

PageRank

Serve também de revisão dos conceitos essenciais das aulas anteriores sobre Cadeias de Markov

Motivação

E um pouco de História ..

Como descobrir informação na web?

- Primeiras tentativas:

- Listas criadas por humanos

- Directórios da Web**

- Yahoo, DMOZ, LookSmart

- Segunda geração: **Web Search**

- **Information Retrieval :**

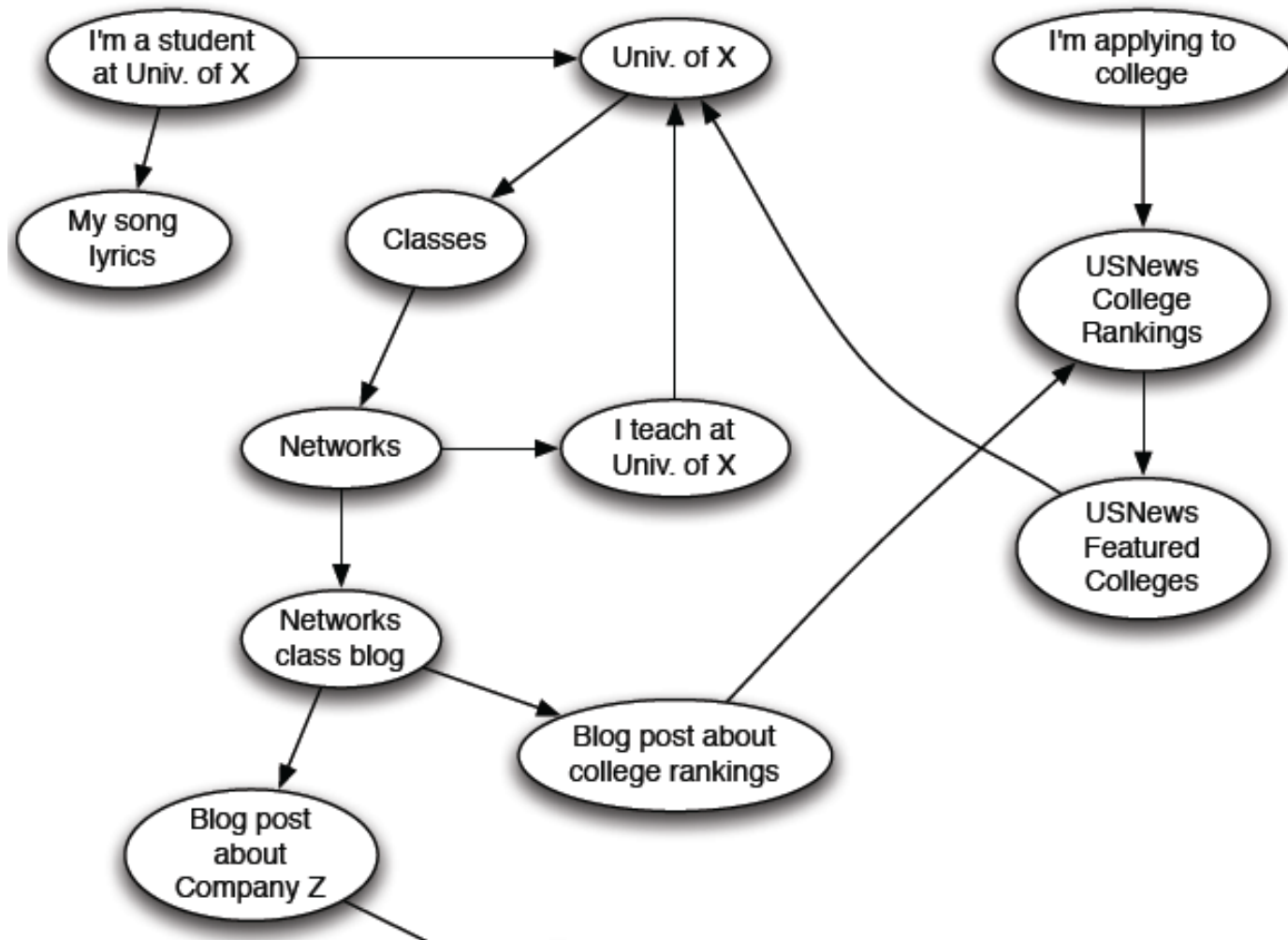
- Procura de documentos relevantes num conjunto pequeno e de confiança

- Artigos de jornais, Patentes, etc.

- **MAS:** A Web é gigantesca, cheia de documentos sobre os quais não temos garantias de ser de confiança, cheia de SPAM, etc.

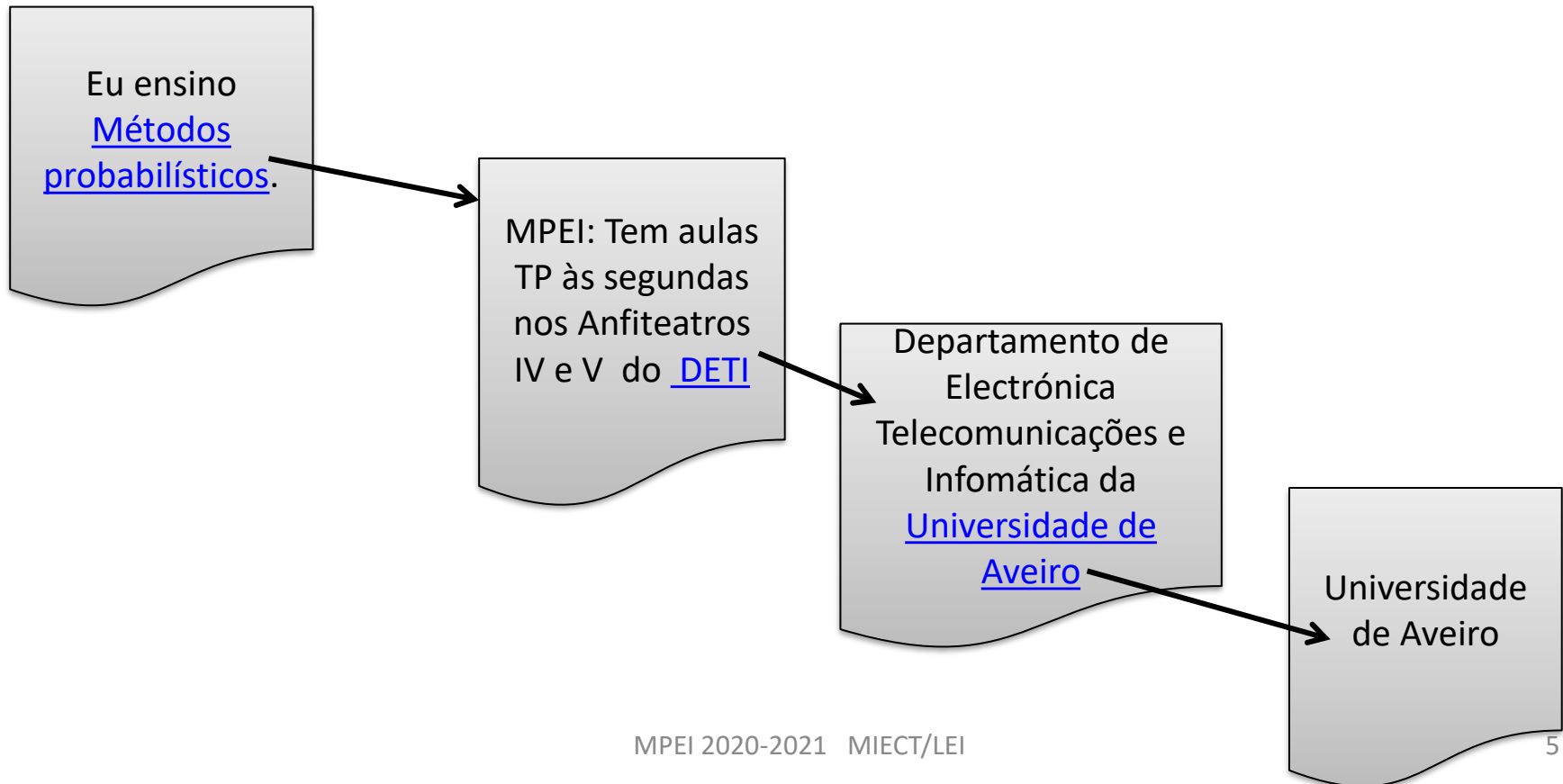


A Web como um grafo

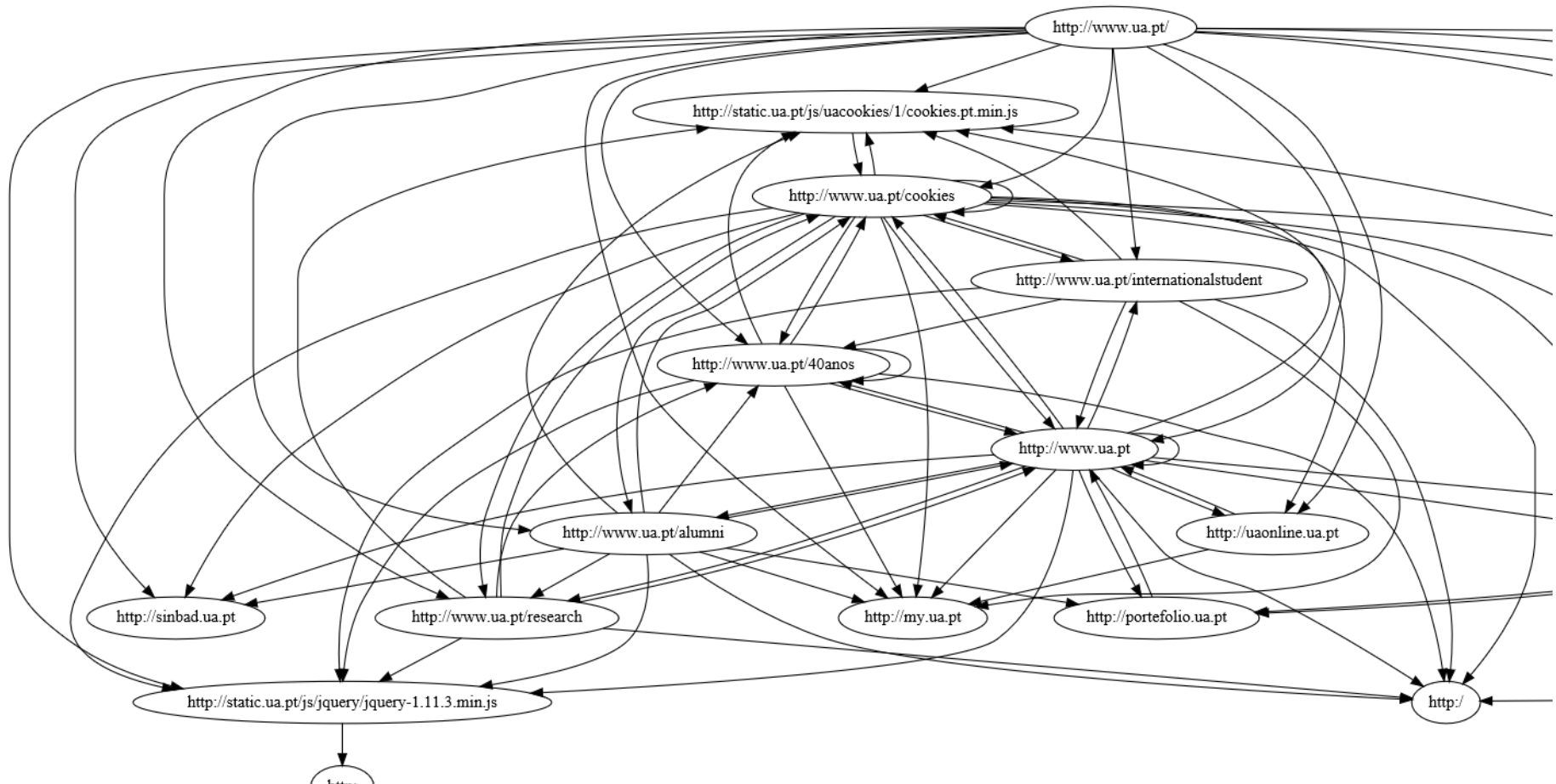


A Web como um Grafo

- Nós /nodos/vértices : Páginas Web
- Ligações/arcos: Hyperlinks



Uma pequena parte das páginas da UA



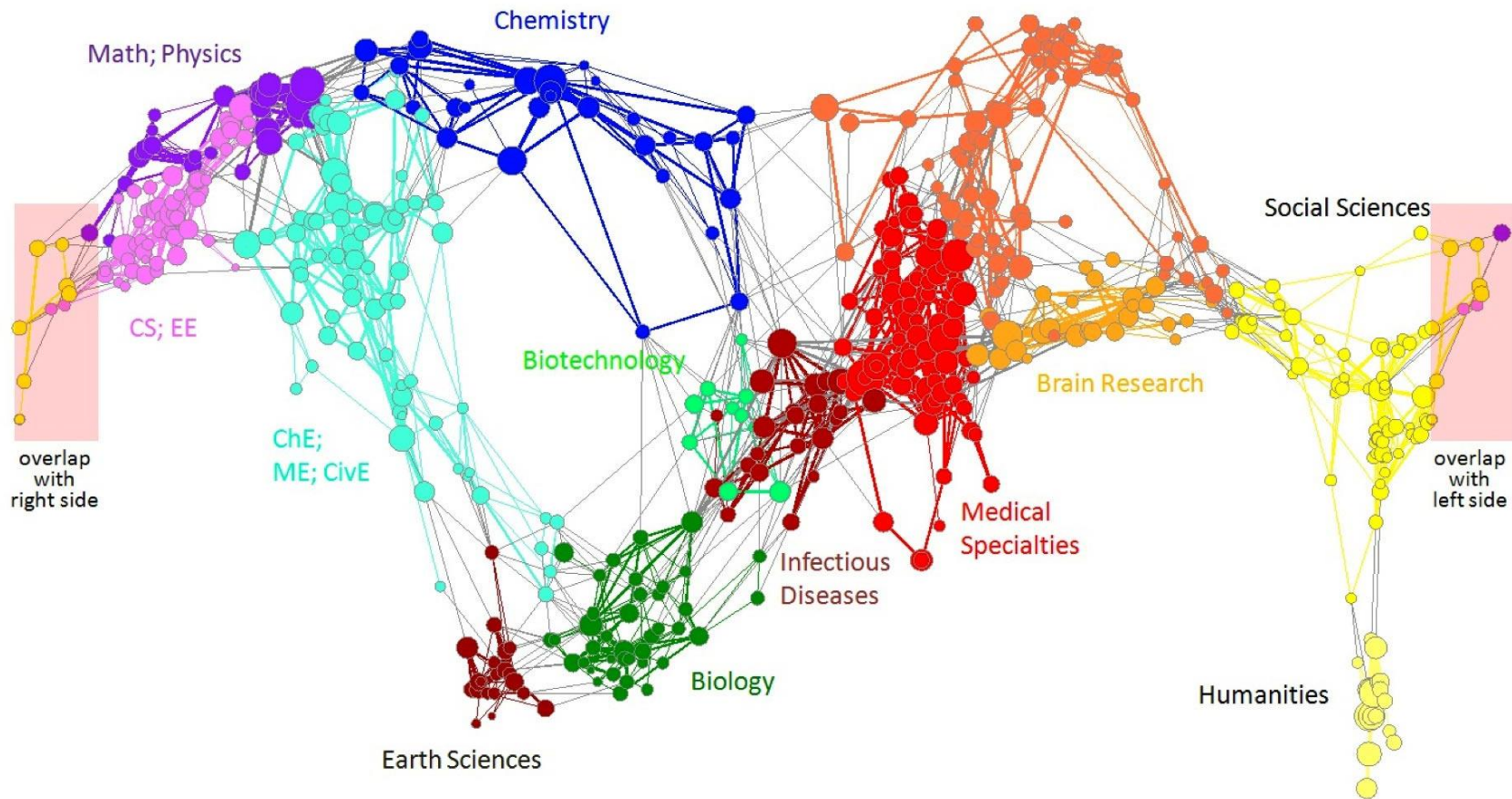
Outros grafos na web: Social Networks



Facebook social graph

4-degrees of separation [Backstrom-Boldi-Rosa-Ugander-Vigna, 2011]

Outros grafos na web: Redes de informação



Citation networks and Maps of science
[Börner et al., 2012]

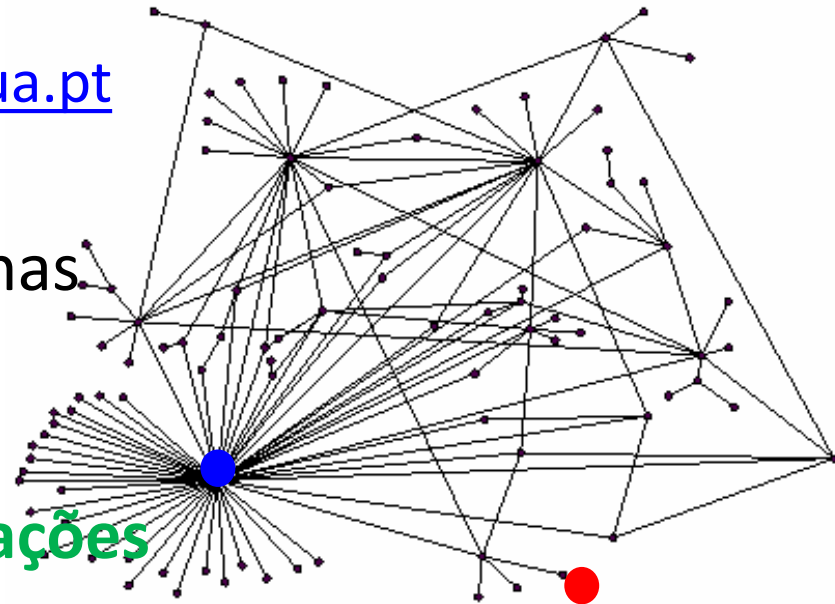
Estrutura do grafo

- As páginas da web Não são todas igualmente “importantes”

www.joe-schmoe.com vs. www.ua.pt

- Existe um grande diversidade nas ligações

- Ideia: Usar a estrutura das ligações para saber quais as páginas “importantes”



Os primeiros motores de procura

- Baseavam-se em percorrer (crawl) a web e **listar os termos** (palavras ou outras sequências de caracteres excluindo os espaços) de cada página, num **índice invertido**



SAPO

- Um índice invertido/inversion (**inverted index**) é uma estrutura de dados que torna simples, **dado um termo, descobrir** (apontar para) **todas as páginas em que o termo ocorre**.
 - Assunto da área de Information Retrieval

Ataques de spam

- Rapidamente estes primeiros [motores de procura](#) foram atacados.



SAPO



- Sendo **sensíveis às palavras nas páginas**, facilmente os detentores de páginas com menos escrúpulos podiam inflacionar a importância das suas páginas:
 - Adicionando muitas cópias de uma ou várias palavras ao conteúdo da página e tornando essa parte invisível quando mostrada num browser
 - Usando o motor de procura para saber a página mais importante segundo o seu algoritmo e copiando o seu conteúdo para as suas páginas (mantendo esta parte invisível no browser)

Contribuição da Google

- Não calcular a relevância das páginas apenas com base nos termos que contém mas usar também informação sobre as ligações a essa página
 - Desenvolvendo e patenteando o Pagerank
 - Que começou como um projecto de investigação

Ideia base

Random surfers / Passeios aleatórios

- Simular onde **passeios aleatórios** (de surfistas aleatórios / random surfers) pelas páginas, começando numa página aleatória, tendem a passar mais se se **escolherem aleatoriamente os links de saída de uma página** (em que se encontram)
 - E permitindo que o processo se repita muitas vezes
- As **páginas visitadas por muitos passeios (ou surfers) serão “mais importantes”** do que páginas raramente visitadas.
- O Google dá preferência a páginas mais importante ao decidir quais as páginas a mostrar primeiro em resposta a um *query*
 - Mas obviamente as páginas têm de conter os termos ...

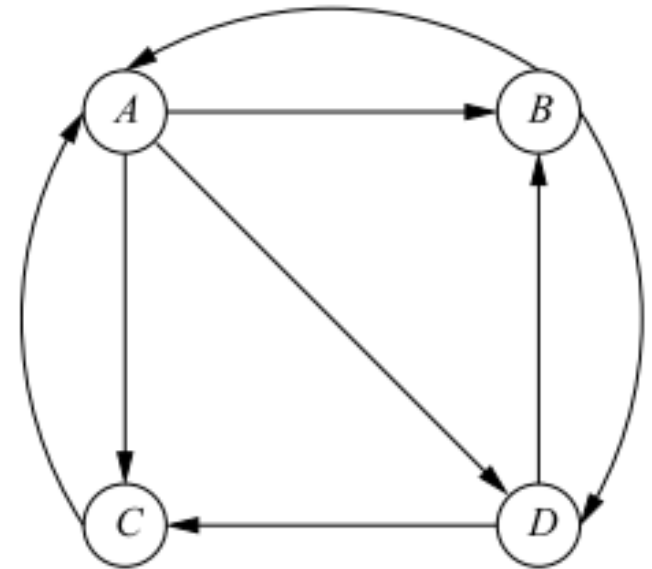
Versão base ‘ideal’

- Consideremos a web como um **grafo orientado** (*directed*),
- em que as **páginas são os nodos** (ou vértices)
- e existe um arco (ou **ligação**) da página P_1 para a página P_2 se existe um ou mais links de P_1 para P_2

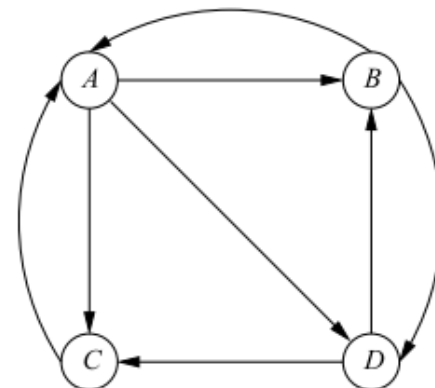
- Exemplo:

- Muito pequena rede: apenas 4 páginas

- A página A tem links para as outras 3
- A página B tem ligações apenas para a A e a D;
- A página C tem apenas um link, para a A
- E a página D tem links apenas para B e C



...



- Suponhamos que o surfista aleatório começa na página A:
- Existem links para B, C e D, logo o surfista estará de seguida numa dessas 3 páginas, com probabilidade $1/3$ [1 a dividir pelos links de saída]
 - E probabilidade zero de estar em A
- O surfista aleatório B terá, no próximo passo, probabilidade $1/2$ de estar em A, $1/2$ de estar em D e 0 de estar em C

Questão

- Estes passeios permitem mesmo aproximar a noção intuitiva de “importância” de uma página ?
- Possíveis Justificações:
 - Os utilizadores da web “votam com os seus pés”. Tendem a colocar links para páginas que consideram boas ou com informação útil
 - E não ligam a páginas de má qualidade ou inúteis
 - O comportamento dos *random surfers* indica quais as páginas que utilizadores da web visitarão com maior probabilidade.
 - Os utilizadores visitam mais páginas úteis do que não úteis.
- Independentemente das justificações anteriores, este método provou na prática que é capaz de atribuir uma medida de “importância” que permite um bom desempenho em procuras na web.
- **Veremos de seguida como funciona ...**

Definição de pagerank

- O PageRank é uma função/algoritmo que atribui um número real a cada página da Web
 - (ou a porção dela que foi processada e as ligações obtidas)
 - Designamos esse número por pagerank
 - Quanto maior é o valor mais “importante” é a página.
- Baseia-se na ideia dos random surfers
- Não existe propriamente um algoritmo fixo, havendo variações do algoritmo
 - Que podem dar valores diferentes de pagerank

pagerank

- O pagerank (r) de uma página P_j é, por **definição**:

$$r(P_j) = \sum_i \frac{r(P_i)}{d_i}$$

- sendo:
 - i o índice das **páginas que apontam para P_j**
 - d_i o número de páginas para as quais P_i aponta
ou seja, número de links de saída

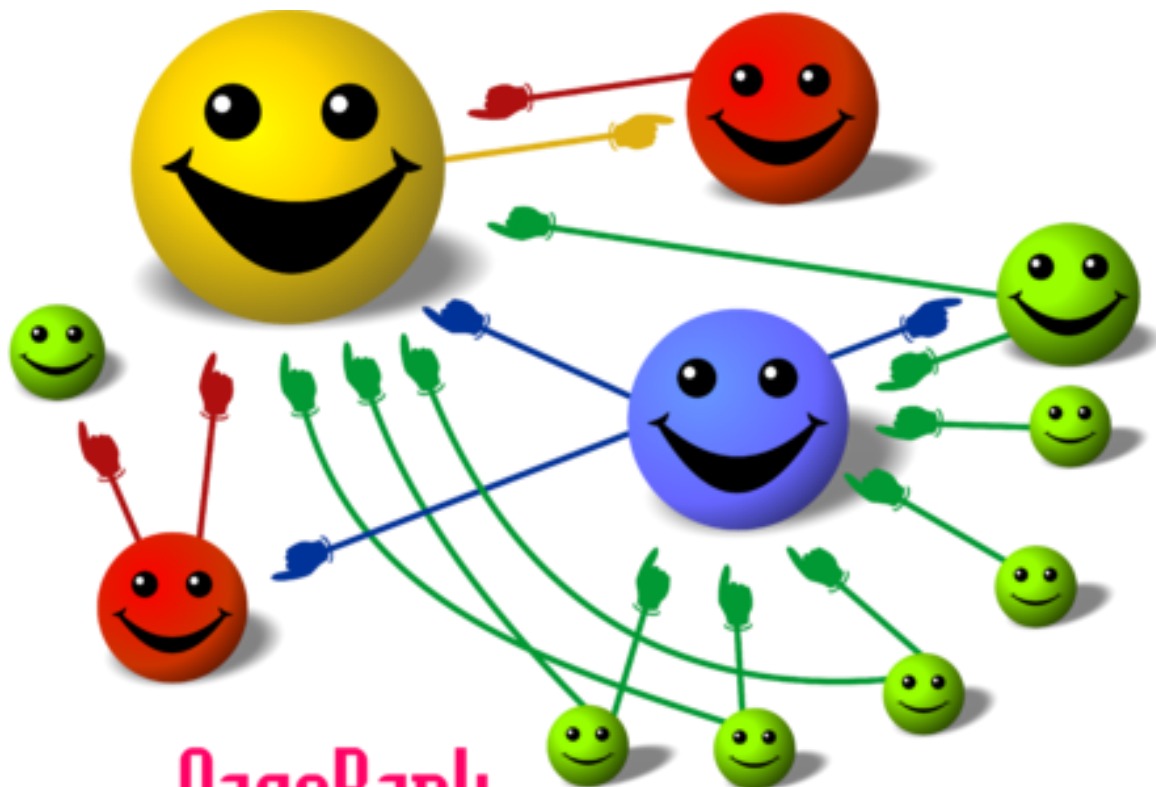
Calculo do pagerank

- O pagerank de uma página **depende do pagerank das páginas que têm links para ela**
 - identificadas pelo índice i

- O que sugere um cálculo iterativo

$$r_{k+1}(P_j) = \sum_i \frac{r_k(P_i)}{d_i}$$

- A **condição inicial** é $r_0(P_i) = 1/n$, com n igual ao número de páginas



PageRank

- Uma simplificação do sistema do PageRank,
- Cada bola representa uma página e o tamanho de cada uma a sua importância (PageRank).
- Quanto maior a bola, mais valor tem seu voto:
- Repare que a bola superior vermelha é grande mesmo recebendo só um voto, pois o voto que ela recebe, da bola maior amarela, tem mais valor

DE: <https://pt.wikipedia.org/wiki/PageRank>.

Forma matricial

- Definindo a **matriz de hyperlinks** H como

- $$H_{ji} = \begin{cases} \frac{1}{d_i} & , \text{se existir link de } i \text{ para } j \\ 0 & , \text{caso contrário} \end{cases}$$

- Teremos $r^{(k+1)} = H r^{(k)}$

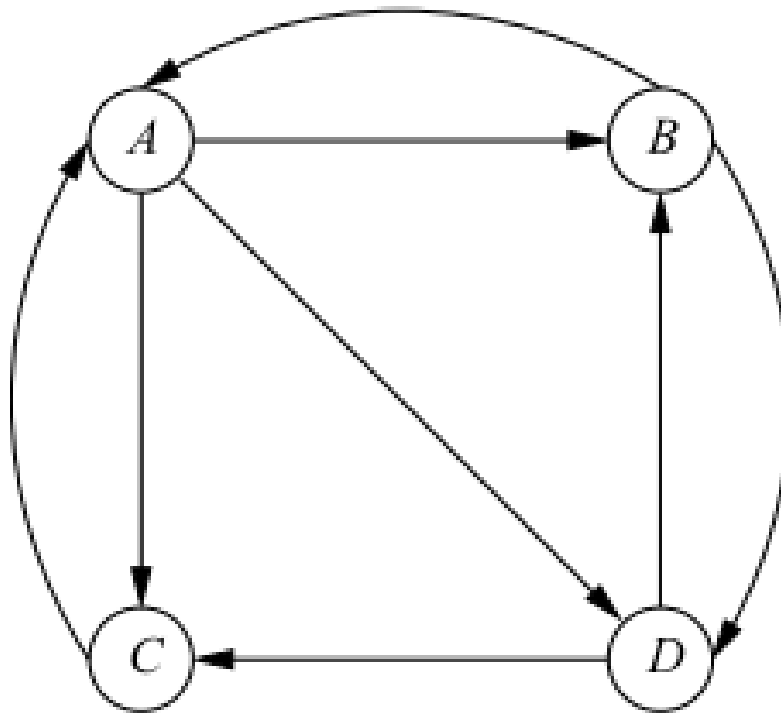
– Sendo $r^{(k)}$ o vector com *pageranks* na iteração k

Forma matricial

- A **matriz H** pode ser interpretada como **contendo as probabilidades de transição entre páginas** (os estados).
- Em consequência pode aplicar-se o que aprendemos nas aulas anteriores e calcular probabilidades após múltiplas transições, estudar o comportamento quando o número de transições (iterações) tende para infinito, etc .

Matriz para o nosso exemplo ?

- Qual será então a matriz H para a nossa mini web ?



Solução

$$\begin{bmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

- Acertaram ?

Limite

- Sabemos que a distribuição dos pageranks atingirá um **estado estacionário**, em que $r = Hr$
 - Pelo menos em certas condições:
 - O grafo ser fortemente ligado, sendo possível ir de qualquer página para qualquer página
 - Não existirem becos sem saída (*dead ends*): páginas que não têm links de saída
- O limite é atingido quando multiplicando os pageranks por H mais uma vez a distribuição de pageranks não se altera

Aplicando ao nosso exemplo

- Aplicando $r^{(k+1)} = H r^{(k)}$ sucessivamente
– e iniciando com $1/n$

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}, \begin{bmatrix} 9/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{bmatrix}, \begin{bmatrix} 15/48 \\ 11/48 \\ 11/48 \\ 11/48 \end{bmatrix}, \begin{bmatrix} 11/32 \\ 7/32 \\ 7/32 \\ 7/32 \end{bmatrix}, \dots, \begin{bmatrix} 3/9 \\ 2/9 \\ 2/9 \\ 2/9 \end{bmatrix}$$

Comentário

- Esta diferença em probabilidade é pequena
- Mas na web real, com biliões de páginas the grande variedade de importância, a verdadeira probabilidade de uma página como www.amazon.com é ordens de magnitude superior à probabilidade de outras páginas, como uma página pessoal

Questões

- É mesmo assim tão simples ?
- Converge sempre?
- Converge para o que queremos?
- Os resultados são razoáveis?

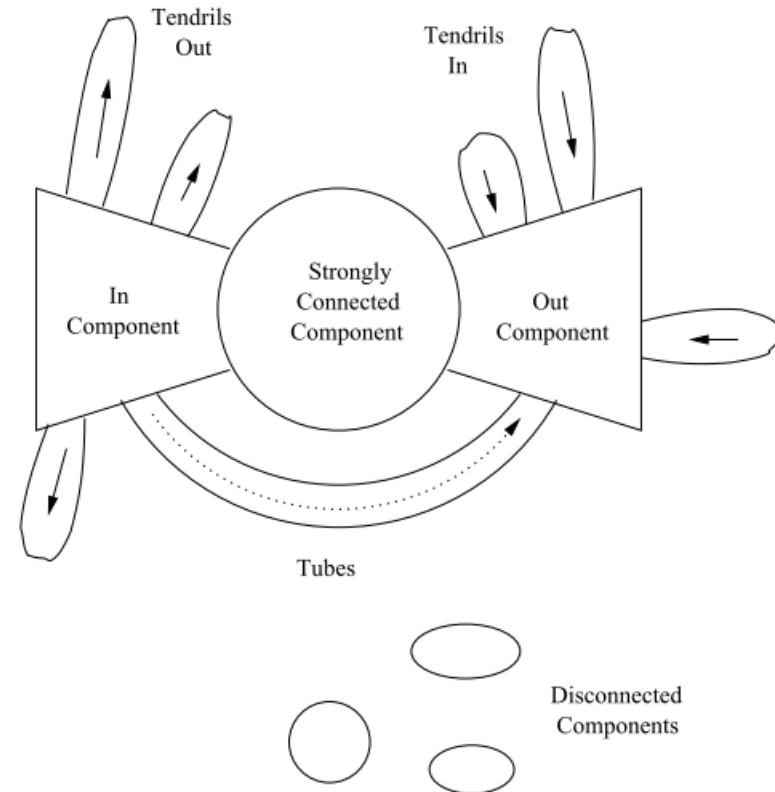
A realidade é sempre mais
complicada ...

Estrutura da web

- Será a **web tão fortemente ligada** como o nosso exemplo ?
- Seria bom que fosse...
- Mas, na prática, não é
 - Pelo menos não na sua totalidade

Estrutura da web

- Um estudo antigo da web revelou a estrutura à direita
- Existe uma parte fortemente ligada (o SCC)
- Mas também muitas páginas com:
 - Ligações ao SCC mas às quais não é possível chegar a partir do SCC (in-component)
 - Ligações a partir do SCC mas sem forma de chegar ao SCC (out-component)
 - Ligações do in-component mas incapazes de aceder a esse componente
 - Etc ..



Isto traz problemas

Temos **dois tipos de problemas** que têm de ser resolvidos

1. Becos sem saída (*dead ends*)

2. Grupos de páginas que têm links de saída mas apenas para esse grupo, impedindo a ida para outras páginas

— Estas estruturas são chamadas de *spider traps*

- Porquê ?

A spider is a program run by a [search engine](#) to build a summary of a website's content (*content index*). Spiders create a text-based summary of content and an address ([URL](#)) for each webpage.

Problemas (continuação)

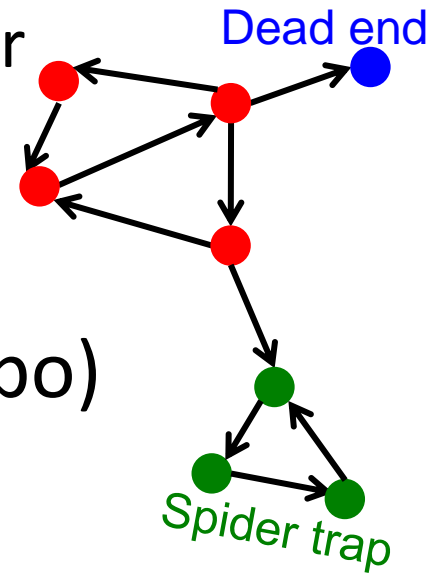
1. **Dead ends** (sem links de saída)

- Passeio aleatório não tem para onde ir

2. **Spider traps:**

(todos os links de saída para o grupo)

- O passeio aleatório fica “preso” na armadilha (trap)



- Qual o efeito nos pageranks ?

Efeito de *dead ends* e *spider traps* no pagerank

E como resolver.

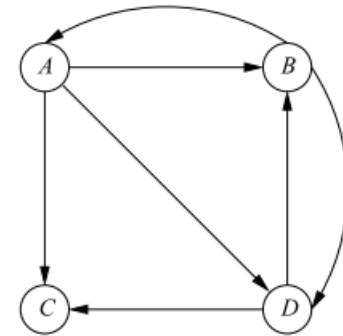
Efeitos dos *Dead ends*

- Neste caso as colunas correspondentes ficam com zeros e a sua soma é zero
- Em consequência a matriz de transição deixa de ser estocástica
- O que implica ?

Dead Ends – Exemplo

- Removendo a ligação de C para A, C passa a ser um *dead end*.

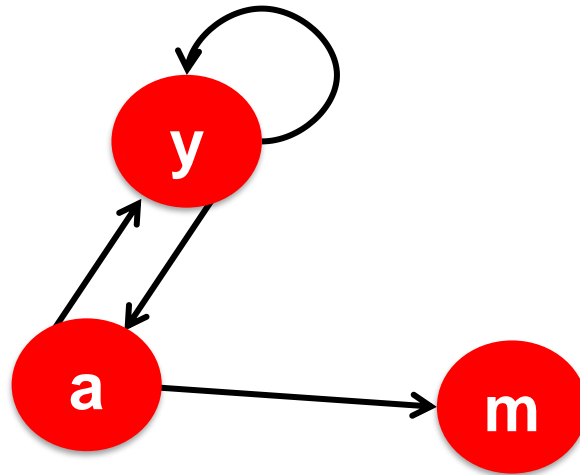
$$H = \begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$



- Multiplicando várias vezes H pelo estado inicial temos:

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}, \begin{bmatrix} 3/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{bmatrix}, \begin{bmatrix} 5/48 \\ 7/48 \\ 7/48 \\ 7/48 \end{bmatrix}, \begin{bmatrix} 21/288 \\ 31/288 \\ 31/288 \\ 31/288 \end{bmatrix}, \dots, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Dead Ends – Exemplo 2



	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	0
a	$\frac{1}{2}$	0	0
m	0	$\frac{1}{2}$	0

- Exemplo:**

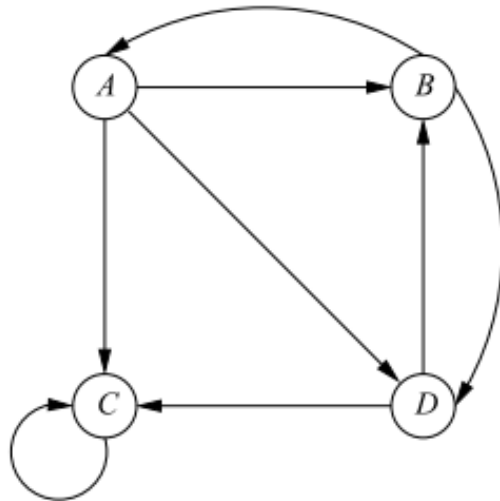
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \dots & 0 \\ 1/3 & 1/6 & 1/12 & 2/24 & & 0 \end{matrix}$$

Iteração 0, 1, 2, ...

O PageRank “desaparece” pois a matriz não é estocástica.

Spider Traps – Exemplo 1

- Consideremos a seguinte rede e matriz

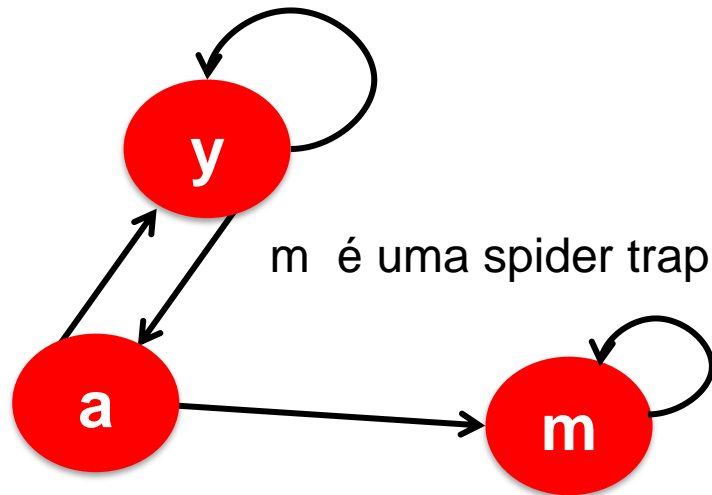


$$\begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 1 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

- Teremos para o vector estado

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}, \begin{bmatrix} 3/24 \\ 5/24 \\ 11/24 \\ 5/24 \end{bmatrix}, \begin{bmatrix} 5/48 \\ 7/48 \\ 29/48 \\ 7/48 \end{bmatrix}, \begin{bmatrix} 21/288 \\ 31/288 \\ 205/288 \\ 31/288 \end{bmatrix}, \dots, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Spider Traps – Exemplo 2



	y	a	m
y	$1/2$	$1/2$	0
a	$1/2$	0	0
m	0	$1/2$	1

• Exemplo:

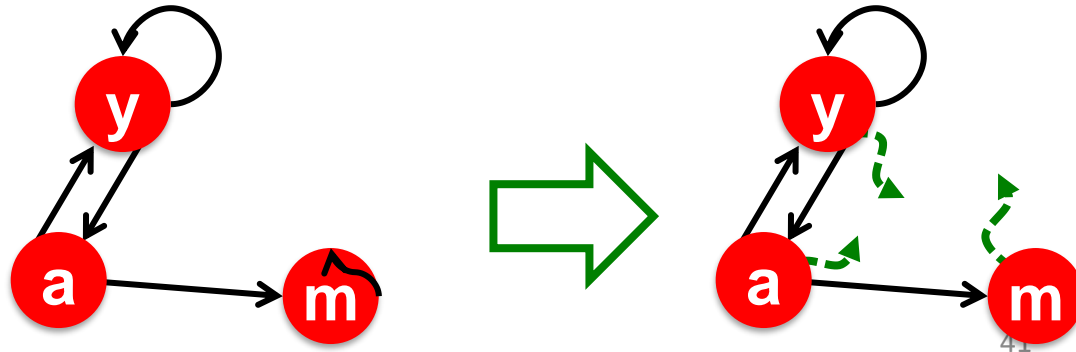
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{pmatrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \dots & 0 \\ 1/3 & 3/6 & 7/12 & 16/24 & & 1 \end{pmatrix}$$

Iteração 0, 1, 2, ...

O PageRank é todo “apanhado” pelo nó m.

Solução para *spider traps*

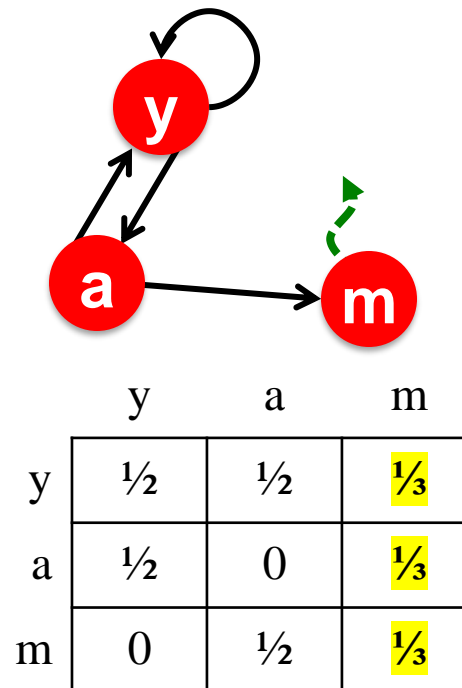
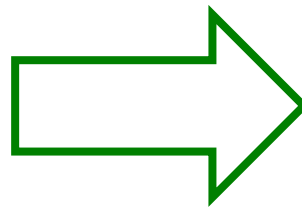
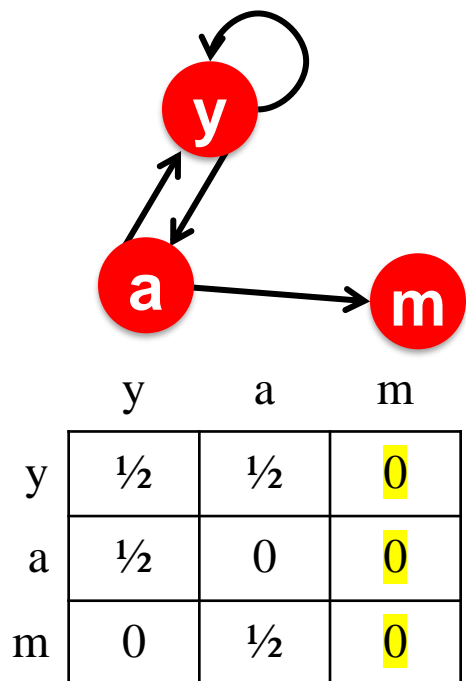
- Em cada passo, **o surfista aleatório tem duas opções**
 - Com probabilidade β :
 - **Seguir um link** aleatoriamente
 - Com probabilidade $1-\beta$,
 - **Saltar aleatoriamente para uma página qualquer**



- **O surfista teletransporta-se para fora da *spider trap* ao fim de alguns passos**
- Valores usuais para β : intervalo 0.8 to 0.9

Solução para *dead ends*

- Teletransportar (teleport) sempre
- **Implica ajustar a matriz** por forma a:
 - seguir um link com probabilidade $1/n$



Os teletransportes resolvem os dead-ends ?

- **SIM**
- **Deixa de haver dead-ends**
 - Existe sempre para onde ir
 - Matriz volta a ser estocástica

Solução: *Random Teleports* (teletransportes aleatórios)

- A solução da Google resolve a possibilidade de haver spider traps
- Em cada passo, o surfista aleatório tem duas opções
 - Com probabilidade β :
 - Seguir um link aleatoriamente
 - Com probabilidade $1-\beta$:
 - Saltar aleatoriamente para uma página qualquer
- PageRank equation [Brin-Page, 98]

$$r_j = \sum_{i \rightarrow j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{N}$$

A Matriz da Google

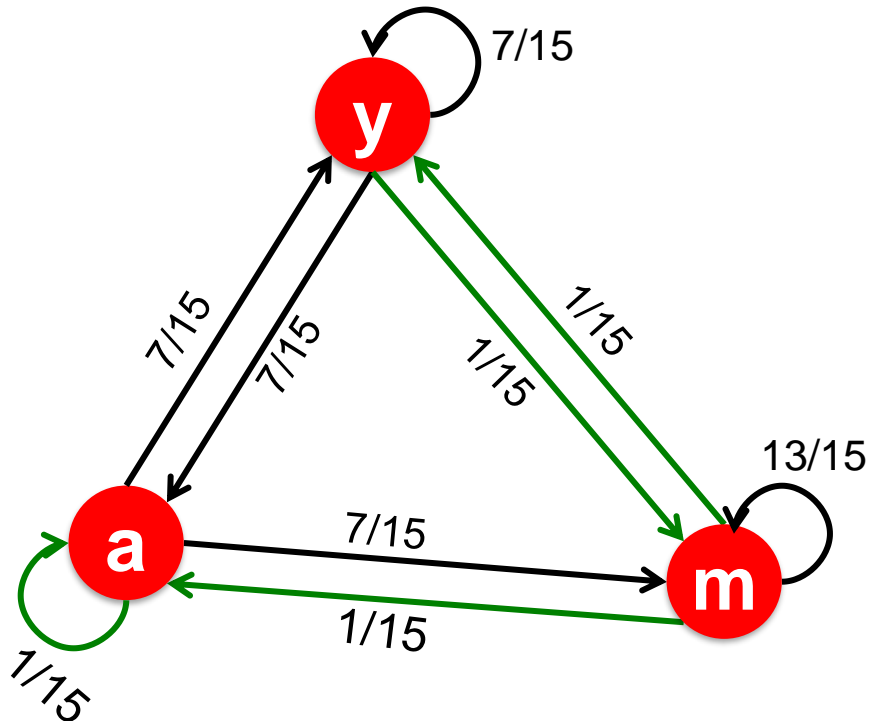
- Matriz da Google:

$$A = \beta H + (1 - \beta) \left[\frac{1}{N} \right]_{N \times N}$$

$[1/N]_{N \times N}$...matriz N por N com todas entradas iguais a $1/N$

- Temos um problema recursivo: $r = A \cdot r$
- A que se pode aplicar o método das potências (Power)
- β ?
 - Na prática $\beta = 0.8, 0.9$ (5 passos em média para saltar)

Exemplo: Random Teleports ($\beta = 0.8$)



$$0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

y	7/15	7/15	1/15
a	7/15	1/15	1/15
m	1/15	7/15	13/15

A

y		1/3	0.33	0.24	0.26		7/33
a	=	1/3	0.20	0.20	0.18	...	5/33
m		1/3	0.46	0.52	0.56		21/33

PageRank in Matlab

Ver

<http://www.mathworks.com/moler/exm/chapters/pagerank.pdf> para mais
informação

Aplicação a uma “pequena rede”

- Consideremos apenas um número reduzido de páginas
 - exemplo: $N=20$ páginas acedíveis a partir de <http://www.ua.pt>
- Principais passos:
 - Obter informação das páginas e suas ligações
 - Com base nos links obter M (ou H)
 - Criar a matriz A aplicando o método da Google para evitar dead ends e spider traps
 - Aplicar power method
 - Apresentar resultados

Obter informação das páginas e suas ligações

- Uma forma simples, em Matlab, é usando a função **surfer()**
 - **disponibilizada** em <http://www.mathworks.com/moler/exm/exmfilelist.html>
 - Copyright 2013 Cleve Moler & The MathWorks, Inc.
- Fazendo: `[U,L]=surfer('http://www.ua.pt',20);`
- Temos em U as URLs
- E em L as ligações

Resultados exemplificativos

```
>> U{1:6}
```

```
ans =
```

```
'http://www.ua.pt'
```

```
'http://static.ua.pt/js/jquery/jquery-1.11.3.min.js'
```

```
'http://static.ua.pt/js/uacookies/1/cookies.pt.min.js'
```

```
'http:/'
```

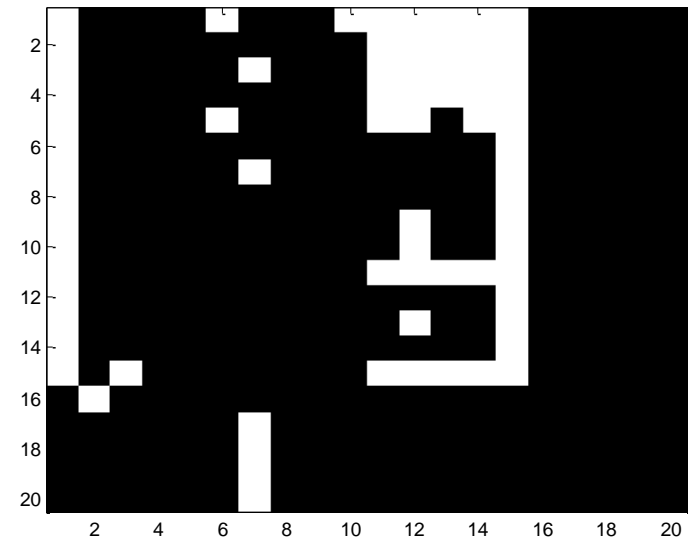
```
'http://my.ua.pt'
```

```
'http://uaonline.ua.pt'
```

L:

```
imagesc(L);
```

```
colormap(gray);
```



Obter H e A

```
H=full(L);
```

```
c=sum(full(L)); % número de ligações (d)
```

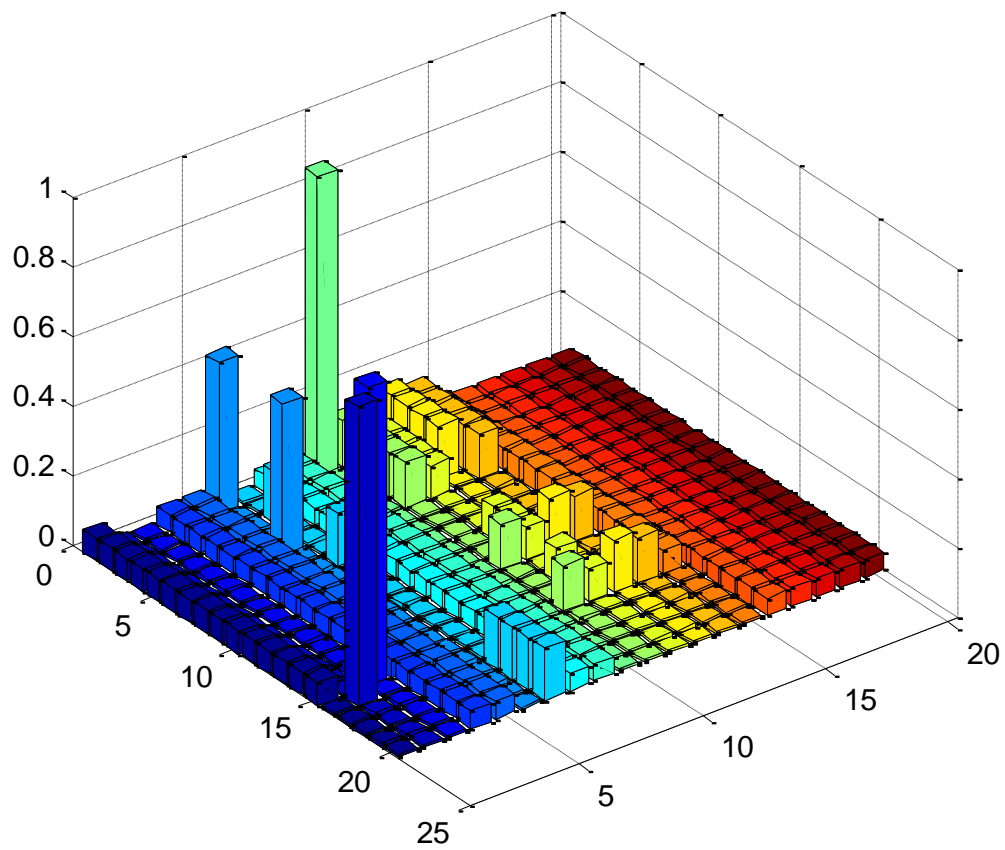
```
H=H./repmat(c,N,1)
```

```
p=0.85
```

```
A=p*H+(1-p)*ones(N)/N % matriz da Google
```

```
A(isnan(A))=1/N % resolver dead ends
```

A



Aplicar “power method”

```
x0=ones(N,1)/N;
```

```
% -----
```

```
iter=1;
```

```
x=x0;
```

```
epsilon=1e-3;
```

```
while 1
```

```
    fprintf(1,'iteração %d\n',iter);
```

```
    xold=x;
```

```
    x=A*x;
```

```
    if max(abs(x-xold))<epsilon break ; end
```

```
    iter=iter+1;
```

```
end
```

```
x
```

Apresentar resultados

```
[xs idx]=sort(x,'descend');  
  
for p=1:N  
    fprintf(1,'PageRank=%.3f : %s \n',  
            x(idx(p)), U{idx(p)});  
end
```

Resultados finais

PageRank=0.108 : <http://www.ua.pt/cookies> ,
PageRank=0.104 : <http://www.ua.pt> ,
PageRank=0.071 : <http://> ,
PageRank=0.065 : <http://my.ua.pt> ,
PageRank=0.062 : <http://static.ua.pt/js/uacookies/1/cookies.pt.min.js> ,
PageRank=0.056 : <http://static.ua.pt/js/jquery/jquery-1.11.3.min.js> ,
PageRank=0.056 : <http://> ,
PageRank=0.056 : <http://www.ua.pt/40anos> ,
PageRank=0.042 : <http://elearning.ua.pt> ,
PageRank=0.039 : <http://sinbad.ua.pt> ,
PageRank=0.039 : <http://portefolio.ua.pt> ,

- Estes resultados podem ser confirmados usando
- $r = \text{pagerank}(U, L)$
 - A diferença entre o vector x obtido e r tem de ser zero

Para casa

- Obter `surfer()` e `pagerank()` de Cleve Moler
 - <https://www.mathworks.com/matlabcentral/fileexchange/4822-using-numerical-computing-with-matlab-in-the-classroom/content/surfer.m>
- Fazer com que o Matlab consiga encontrar essas funções
 - Adicionando o directório em que as colocar à path ou colocando-as no seu directório de trabalho
- Fazer os exemplos em <http://www.mathworks.com/moler/exm/chapters/pagerank.pdf>
- Experimentar o código dos slides anteriores
- Experimentar com outras “mini redes”

E na realidade ?

Apenas 2 ou 3 slides para terem uma ideia, pois começa a sair do âmbito de MPEI

Calcular para toda a web ...

- O passo mais importante é a multiplicação

$$r^{k+1} = A \cdot r^k$$

- Fácil se desse para ter tudo em memória: A , r^{k+1} , r^k
- Se $N = 10^9$ páginas
 - 1 bilhão na América, 1000 milhões na Europa
- E considerarmos 4 bytes por entrada
- Temos:
 - 2×10^9 posições para os 2 vectores (aprox. 8GB)
 - Uma matriz A com N^2 elementos
 - Ou seja 10^{18}

Partes da solução ...

- Aproveitar o facto da matriz ser esparsa
- Codificá-la com base apenas nas entradas não-nulas

origem	grau	Nós destino
0	3	1, 5, 7
1	5	17, 64, 113, 117, 245
2	2	13, 23

- Espaço proporcional aproximadamente ao número de links
- Por exemplo: 10N, or $4 \cdot 10^9 \cong 40\text{GB}$
- **Contínua a “não caber” em memória mas cabe em disco**

Algoritmo básico: Passo de actualização

- **Assumindo que r^{k+1} cabe em memória**
 - r^k e a matriz no disco
- **1 passo da iteração do método das potências :**

Inicializar todas as entradas de r^{k+1} com $(1-\beta) / N$
Para cada página i (com grau d_i):
 Ler para memória: $i, d_i, dest_1, \dots, dest_{d_i}, r^k(i)$
 Para $j = 1 \dots d_i$
 $r^{k+1}(dest_j) += \beta r^k(i) / d_i$

Exemplo

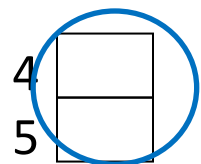
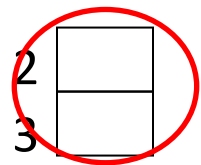
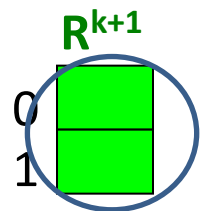


61

Inicializar todas as entradas de r^{k+1} com $(1-\beta) / N$
 Para cada página ORIGEM i (com grau d_i):
 Ler para memória: $i, d_i, \text{dest}_1, \dots, \text{dest}_{d_i}, r^k(i)$
 Para $j = 1 \dots d_i$
 $r^{k+1}(\text{dest}_j) += \beta r^k(i) / d_i$

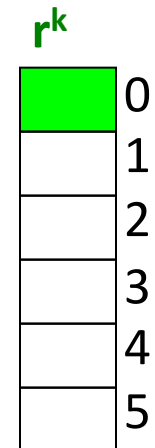
E se nem r^{k+1} cabe em memória ?

- Dividir r^{k+1} em k blocos que caibam em memória
- Processar a Matriz e r^k uma vez para cada bloco



origem	grau	destino
0	4	0, 1, 3, 5
1	2	0, 5
2	2	3, 4

H



Alguns problemas do Page Rank

- **Mede a importância genérica**
 - Não tem em conta “autoridades” num tópico específico
- **Solução:** Topic-Specific PageRank
- **Usa uma medida única de importância**
- **Solução:** Hubs-and-Authorities
- **Susceptível a spam de links**
 - Por exemplo “spam farms”: topografias artificiais de links criadas para aumentar o pagerank
- **Solução:** TrustRank

Para saber mais

- Capítulo *Link Analysis* do livro **Mining of Massive Datasets**
 - Disponível em <http://infolab.stanford.edu/~ullman/mmds/book.pdf>
- Capítulo *PageRank* do livro **Experiments with MATLAB** de C. Moler
 - e respectivo software
 - <http://www.mathworks.com/moler/exm/chapters.html>
- Notas do Prof. Paulo Jorge Ferreira “MPEI - pagerank”
 - Disponíveis no elearning
- Artigos dos autores do PageRank e fundadores da Google
 - É uma questão de usar o Google 😊

Nesta apresentação foram usados e/ou adaptados slides da seguinte apresentação:

Analysis of Large Graphs: Link Analysis, PageRank

Mining of Massive Datasets

Jure Leskovec, Anand Rajaraman, Jeff Ullman Stanford
University

<http://www.mmds.org>



Note to other teachers and users of these slides: We would be delighted if you found this our material useful in giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. If you make use of a significant portion of these slides in your own lecture, please include this message, or a link to our web site: <http://www.mmds.org>