Ranking the WEB

Data mining lab 9

Problem

- WEB stores a vast amount of data
- We query it for an information
- We want the most important results to be presented first
- What does ***most important*** mean: with the highest rank according to some scheme

Part I. Ranking by context. Latent semantics indexing

D1: How to feed your dog

D2: My pet dog

D3: Customer care

D4: Taking care of your pet

D5: Attracting more customers

Query: pet care

Document-term matrix: A

	D1	D2	D3	D4	D5	query
feed	1					
dog	1	1				
pet		1		1		1
custo mer			1		1	
care			1	1		1
take				1		
attract					1	

Calculate singular value decomposition for A

- Go to http://www.bluebit.gr/matrix-calculator/ and enter matrix A:

1.000 0.000 0.000 0.000 0.000

1.000 1.000 0.000 0.000 0.000

0.000 1.000 0.000 0.000 0.000

0.000 0.000 1.000 0.000 0.000

0.000 0.000 1.000 0.000 0.000

0.000 0.000 0.000 0.000 0.000

0.000 0.000 0.000 0.000 0.000

The results: $A=USV^T$

Singular Value Decomposition:

U:

```
-0.4  0.0  0.7 -0.4 -0.4
-0.8  0.0  0.0  0.4  0.4
-0.4  0.0 -0.7 -0.4 -0.4
 0.0 -0.7  0.0 -0.5  0.5
 0.0 -0.7  0.0  0.5 -0.5
 0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0
```

S:

```
1.7 0.0 0.0 0.0 0.0
0.0 1.4 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0
```

V^T

```
-0.7 -0.7  0.0  0.0  0.0
 0.0  0.0 -1.0  0.0  0.0
 0.7 -0.7  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  1.0
 0.0  0.0  0.0  1.0  0.0
```

Compute rank-2 approximation of word terms: row vectors of $U_2 * S_2$

-0.4	0.0
-0.8	0.0
-0.4	0.0
0.0	-0.7
0.0	-0.7
0.0	0.0
0.0	0.0

x

1.7	0.0
0.0	1.4

=

feed	-0.68	0
dog	-1.36	0
pet	-0.68	0
customer	0	-0.98
care	0	-0.98
take	0	0
attract	0	0

Compute rank-2 approximation of document terms (column vectors of): $S_2 * V_2^T$

1.7	0.0
0.0	1.4

x

-0.7	-0.7	0.0	0.0	0.0
0.0	0.0	-1.0	0.0	0.0

=

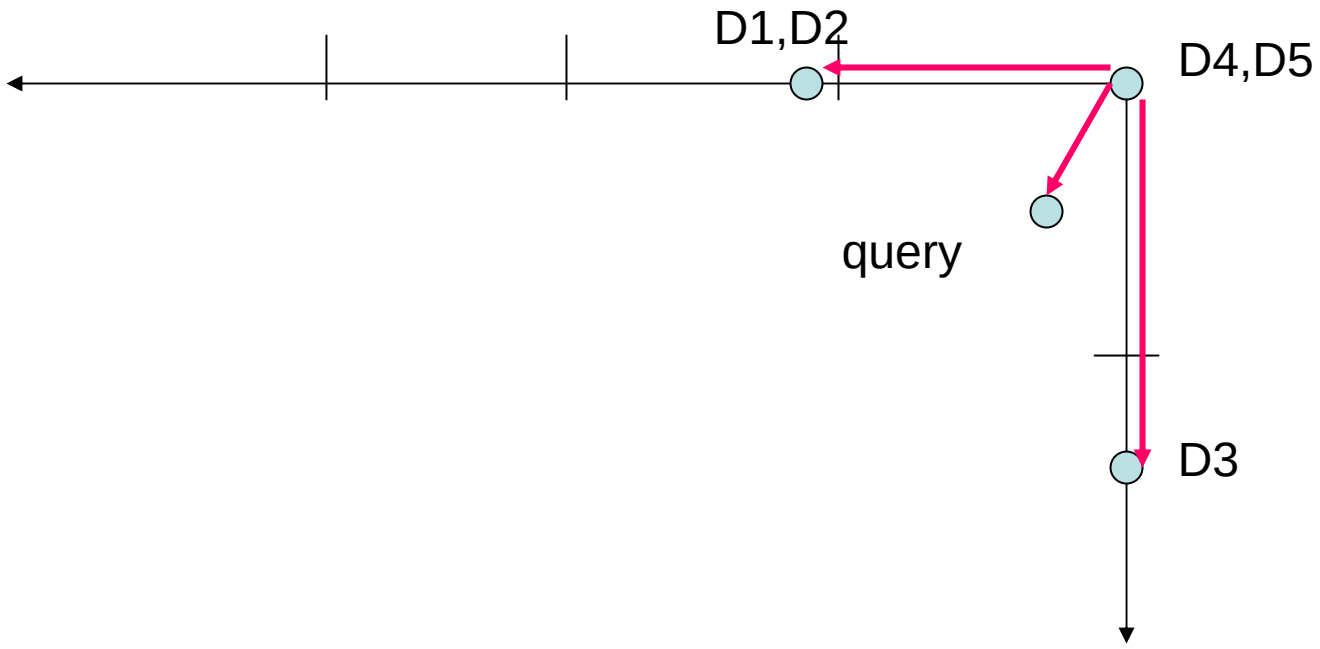
D1	D2	D3	D4	D5
-1.19	-1.19	0.0	0.0	0.0
0.0	0.0	-1.4	0.0	0.0

Compute vector for the query (as a mini-document)

pet	-0.68	0
care	0	-0.98
pet care	-0.34	-0.49

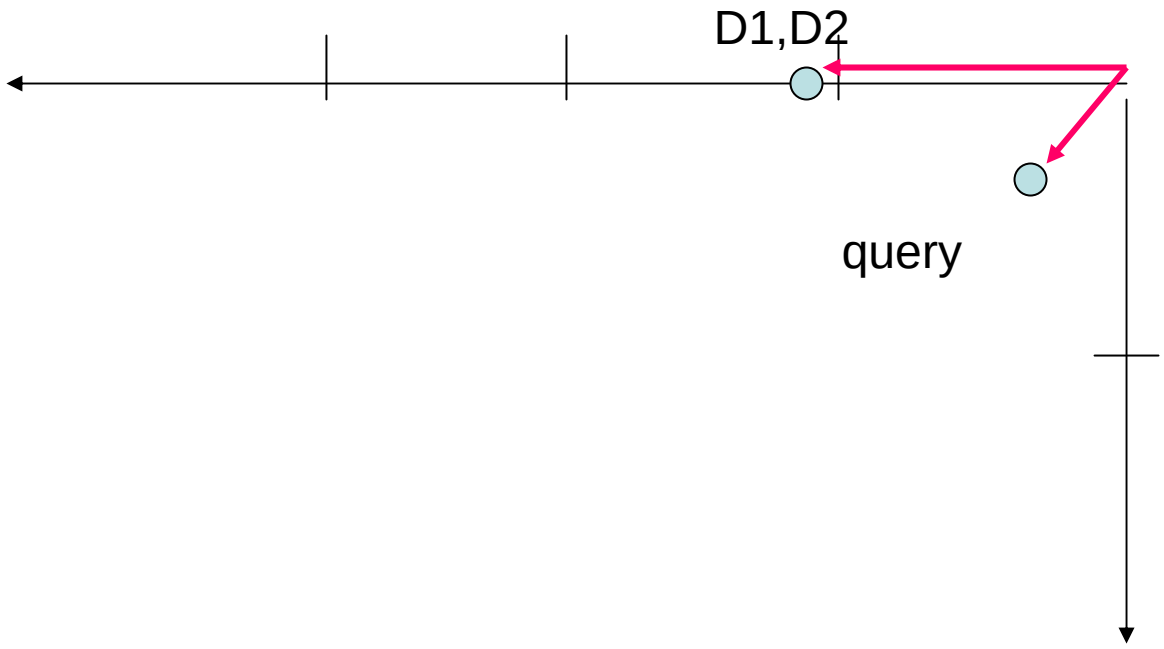
Compare query vector to document vectors

D1	D2	D3	D4	D5	query
-1.19	-1.19	0.0	0.0	0.0	-0.34
0.0	0.0	-1.4	0.0	0.0	-0.49



The cosine similarity of the query and the document D1 is the same as the similarity with D2, even though D1 did not contain query terms (*pet care*)

D1: How to feed your dog
D2: My pet dog

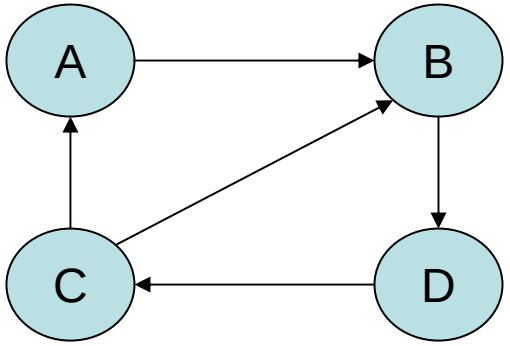


Google uses LSA for ranking

- In the Google search bar type: ***cell phone***
  - Note that results obtained contain ***cell phone***
- Now type ~cell phone
  - Note that the very high ranked document about TELUS does not contain word ***cell***, but ***mobile*** and ***wireless phone*** instead
  - This is because the words ***cell phone*** and ***mobile phone*** occur very often in a similar context

Part II. Ranking by Link Analysis

- Represent WEB pages by a directed graph
- Nodes are pages
- Edges are links
- To be clear: an arrow ending at a given page is a link into that page, and an arrow starting there is a link out to another webpage.



Ideas

- Idea 1: A webpage is important if it has many arrows pointing to it, i.e., many incoming links.
- Why this is too naïve?

Ideas

- Idea 1: A webpage is important if it has many arrows pointing to it, i.e., many incoming links.
- Why this is too naïve?
- Pages from any WEB site have links to the Home page, which will always be rated higher than individual pages

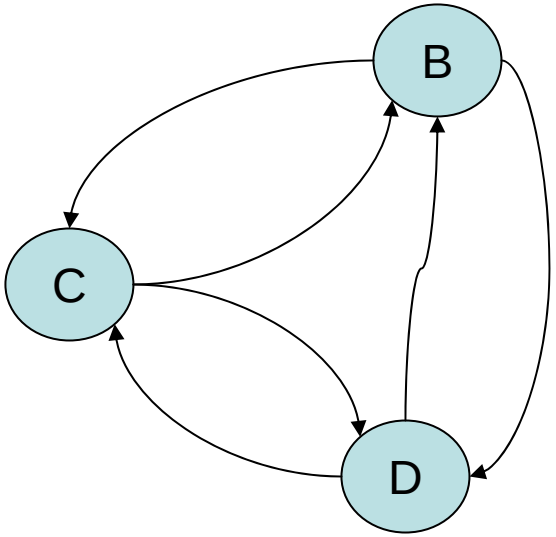
Ideas

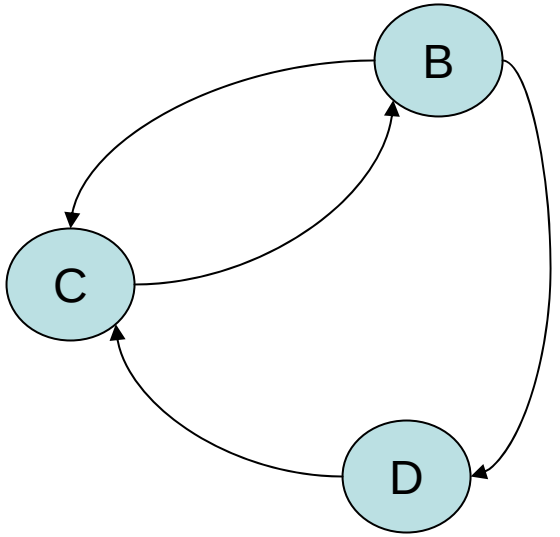
- Idea 2: a webpage is important if many important pages link to it.
- It seems that:

  a problem now is the self-referential nature of this definition;

if we follow this line of reasoning, we might find that the importance of a web page depends on itself.

Models of the WEB





- What can we speculate about the relative importance of pages in each of these models, solely from the structure of the links (which is anyways the only information at hand)?

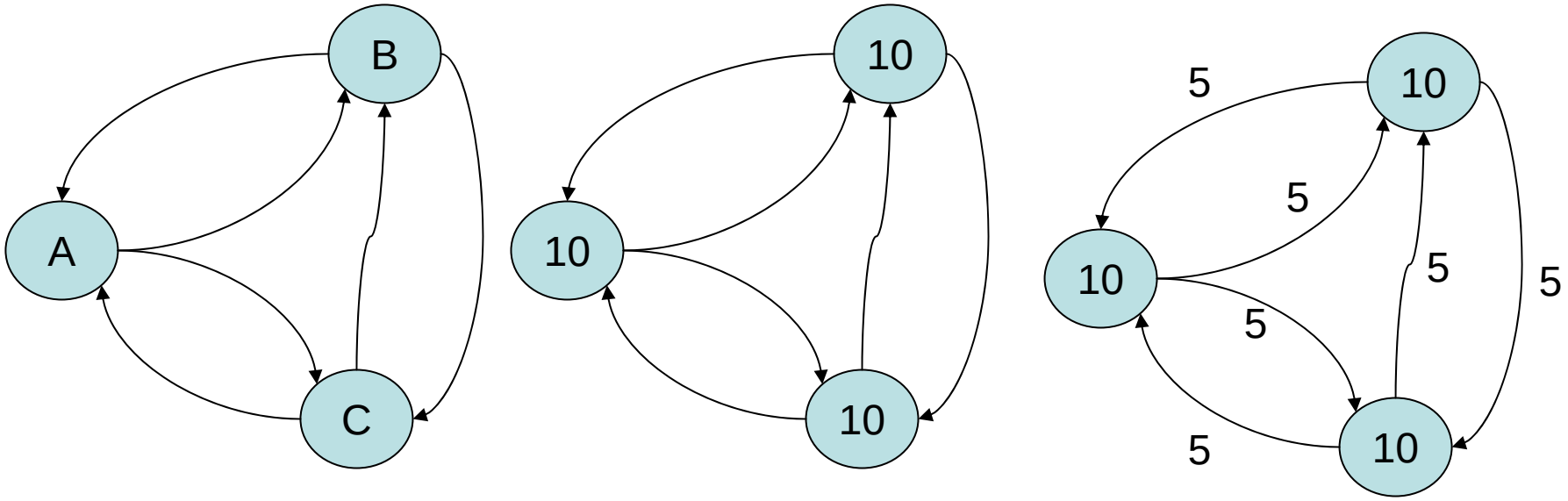
Traffic and mindless surfing.

- Assumptions:
  - The WEB site is important if it gets a lot of traffic.
  - Let us further assume that everyone is surfing spending a second on each page and then randomly following a link to a new page.
  - In this scheme it is convenient

    to make sure a surfer cannot get stuck, so we make the following STANDING ASSUMPTION:

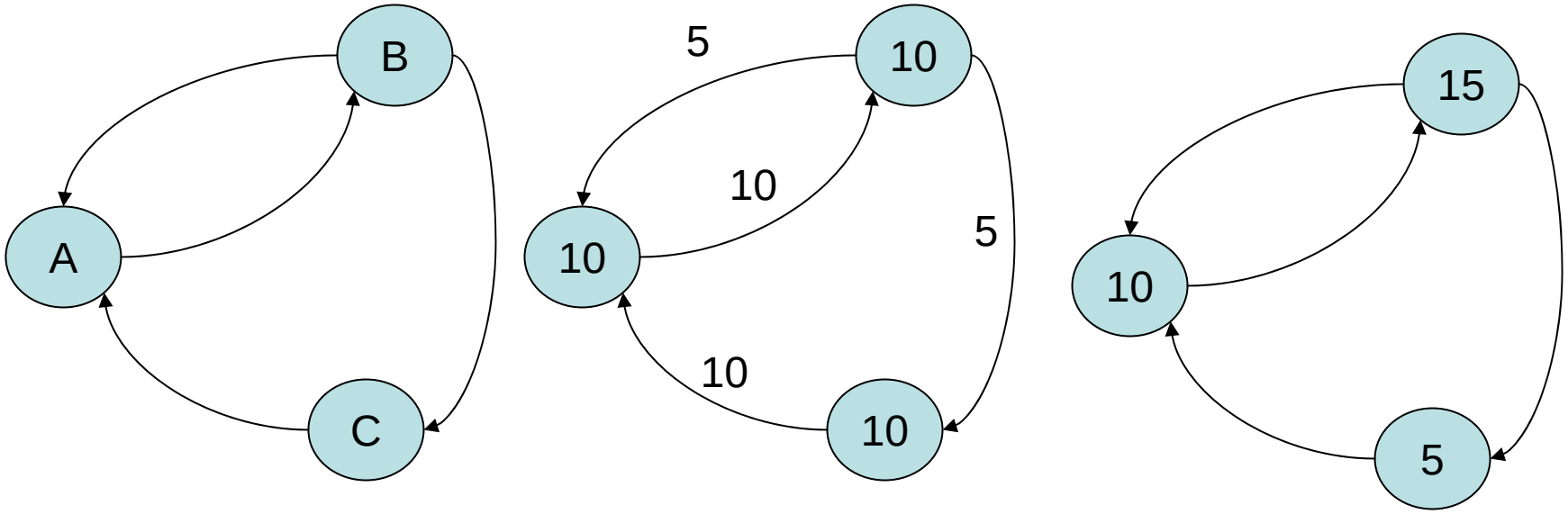
  Each page has at least one outgoing link.

Traffic and mindless surfing. Example 1



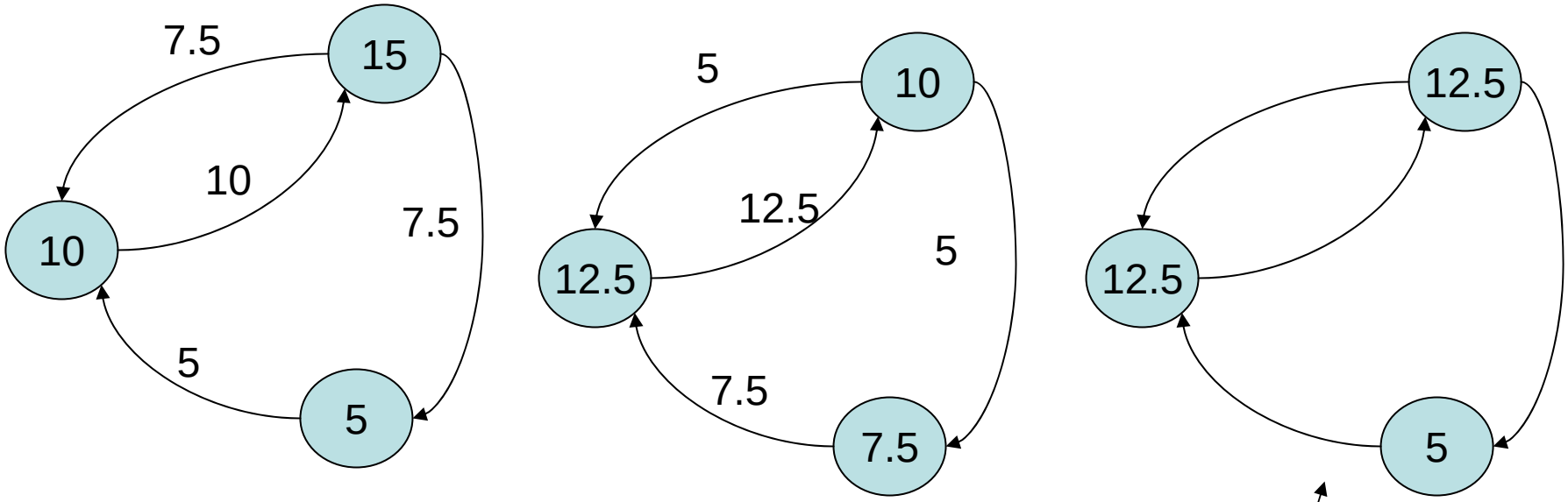
- We start with 10 surfers in each page
- At the first random click, 5 of the surfers at page A, say, go to page B, and the other 5 go to page C. So while each site sees all 10 of its visitors leave, it gets 5 + 5 incoming visitors to replace them: **So the amount of traffic at each page remains constant at 10.**

Traffic and mindless surfing. Example 2



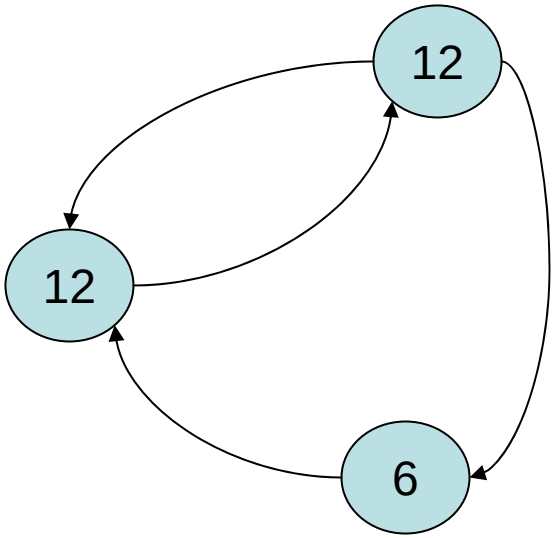
- We start with 10 surfers in each page
- After the first random click, 10 of the surfers at page A go to page B, since there is only 1 outgoing link from A etc…

Traffic and mindless surfing. Example 2



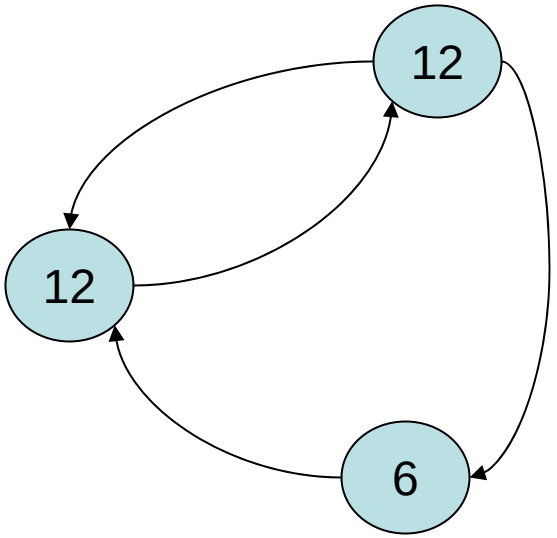
- After the two next clicks it becomes
- Where is this leading? Do we ever reach a stable configuration, as in the first model?

Traffic and mindless surfing. Example 2



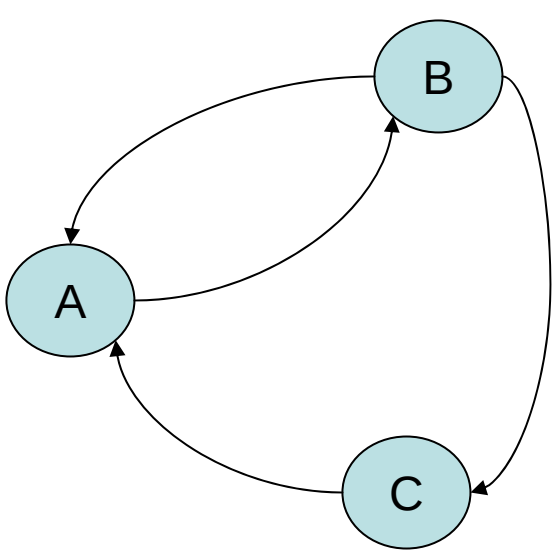
- While the answer is no, it turns out that the process **converges** to the following distribution, which you can check remains the same going forward in time

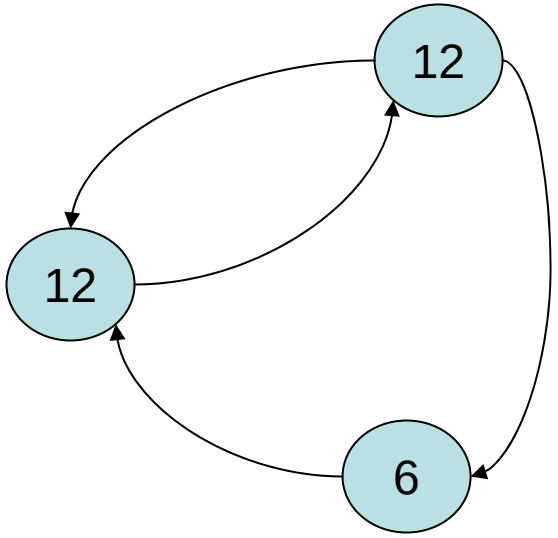
Traffic and mindless surfing. Example 2



- This stable distribution is what the PageRank algorithm (in its most basic form) uses to assign a rank to each page: The two pages with 12 visitors are equally important, and each more important than the remaining page having 6 visitors.

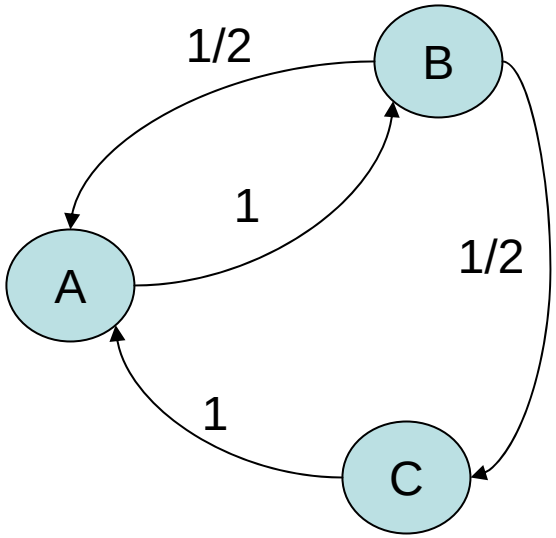
Traffic and mindless surfing. Question?





- How do we qualitatively explain why two of the pages in this model should be ranked equally, even though one has more incoming links than the other?

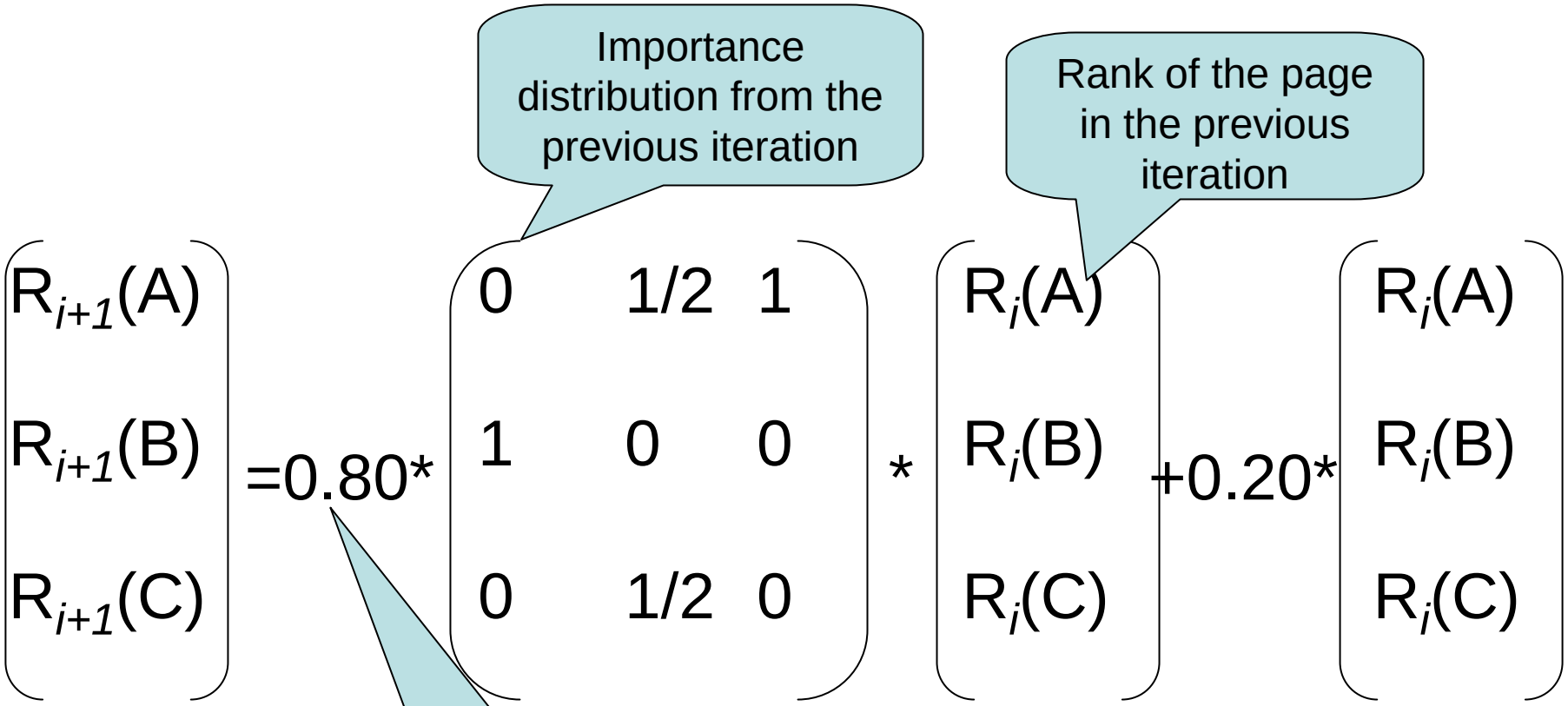
How to compute the stable distribution?



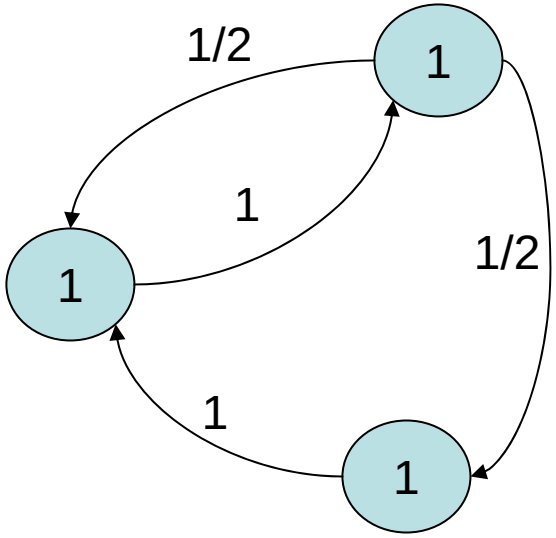
Create matrix of the importance distribution from the current state

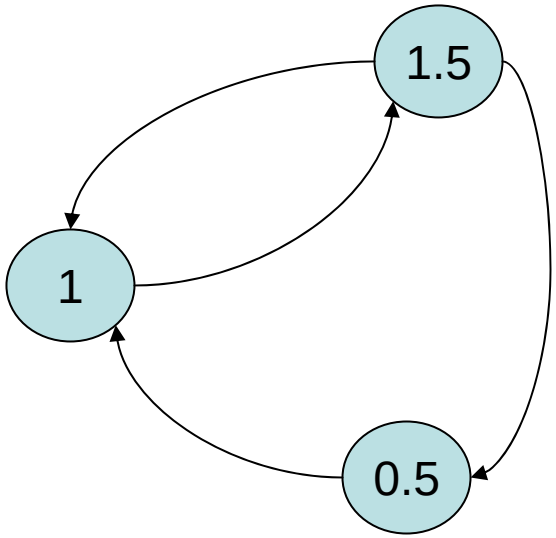
	A	B	C
A	0	1/2	1
B	1	0	0
C	0	1/2	0

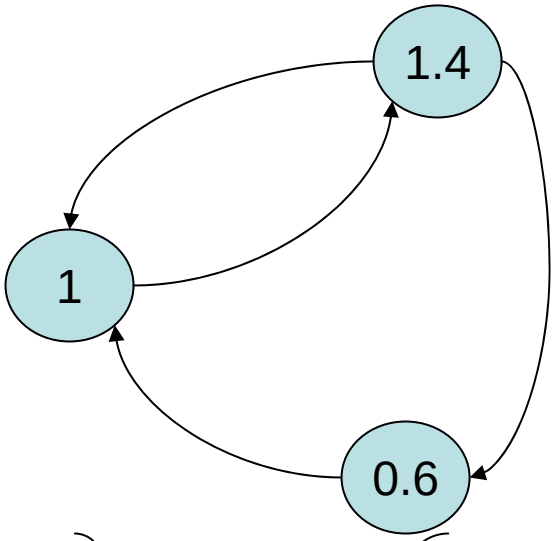
General formula for computing page rank in each iteration *i*



Computing page rank for iteration 1

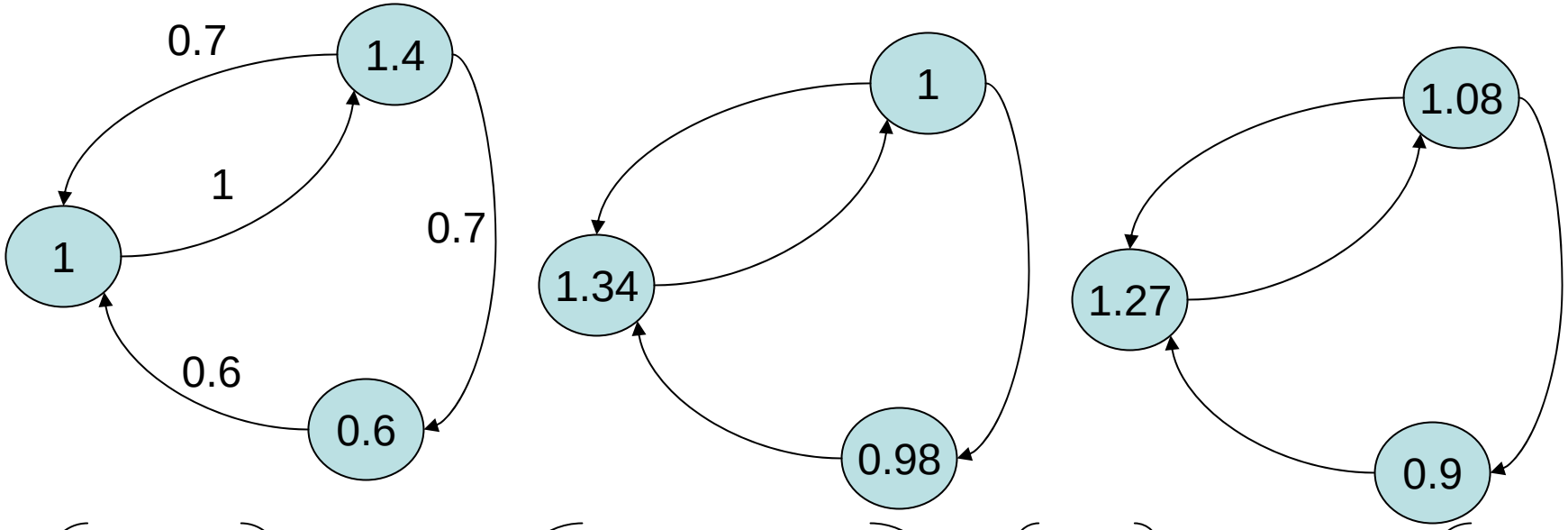






$$\begin{pmatrix} R_2(A) \\ R_2(B) \\ R_2(C) \end{pmatrix} = 0.80 * \begin{pmatrix} 0 & 1/2 & 1 \\ 1 & 0 & 0 \\ 0 & 1/2 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + 0.20 * \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Computing page rank for iteration 2



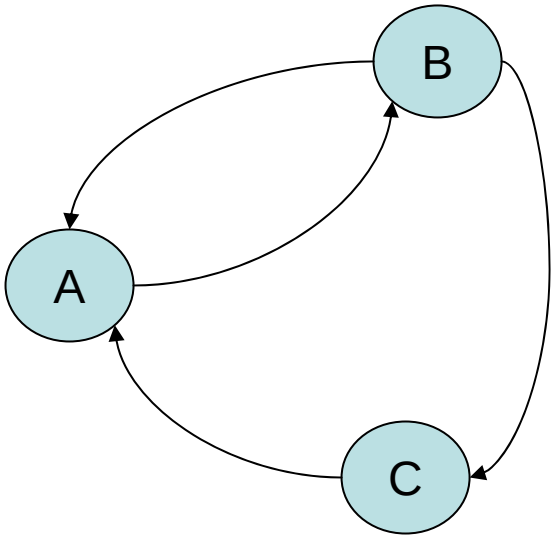
$$\begin{pmatrix} R_3(A) \\ R_3(B) \\ R_3(C) \end{pmatrix} = 0.80 * \begin{pmatrix} 0 & 0.7 & 0.6 \\ 1 & 0 & 0 \\ 0 & 0.7 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 1.4 \\ 0.6 \end{pmatrix} + 0.20 * \begin{pmatrix} 1 \\ 1.4 \\ 0.6 \end{pmatrix}$$

Python code

- Python code for PageRank algorithm is in file: ***PageRank.py***
- In order to run it, you need to install the Python package which works with matrices: ***numpy*** package from **http://numpy.scipy.org/**
- The input is supplied in form of a 2-dimensional array of links between pages

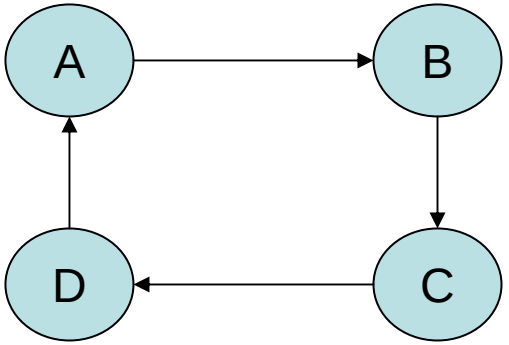
Python code

```
# Example usage
    web = ((0, 1, 0),
           (1, 0, 1),
           (1, 0, 0))

    pr = PageRanker(0.80, web)

    pr.improve_guess(100)
    print pr.getPageRank()
```



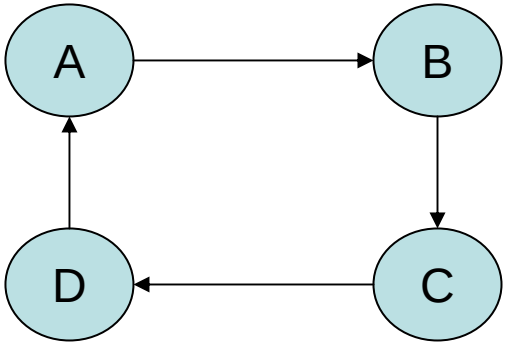

```
>>> execfile('PageRank.py')
[ 0.39739966  0.38778971  0.21481063]
```

PageRank exercise 1.



- Guess what pages in the given model got the highest rank
- Check you guess by running the code

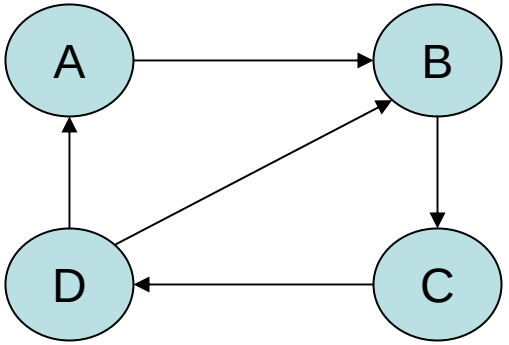
PageRank exercise 1.



- Guess what pages in the given model got the highest rank
- Check you guess by running the code

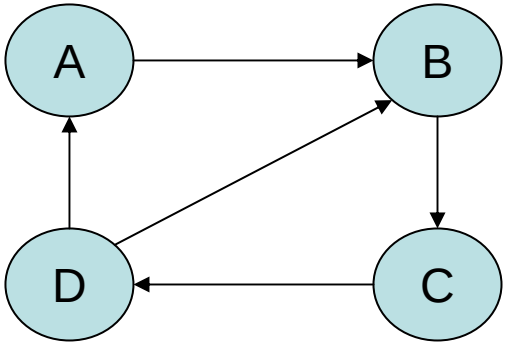
```
>>> execfile('PageRank.py')
[ 0.25  0.25  0.25  0.25]
```

PageRank exercise 2.



- Guess what pages in the given model got the highest rank
- Check you guess by running the code

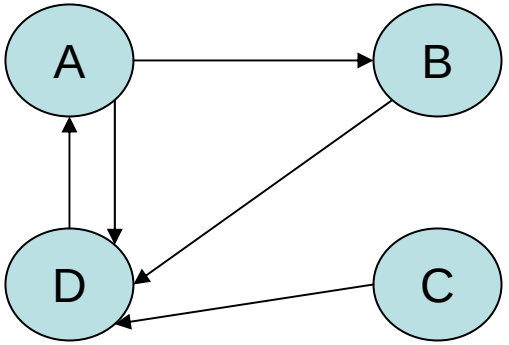
PageRank exercise 2.



- Guess what pages in the given model got the highest rank
- Check you guess by running the code

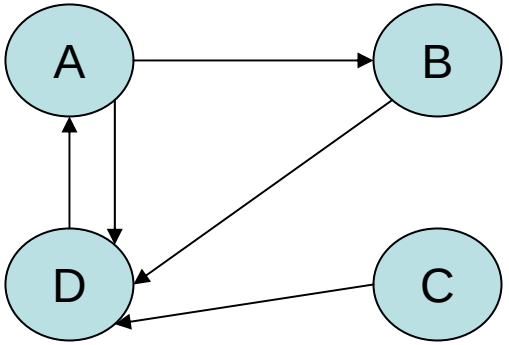
```
>>> execfile('PageRank.py')
[ 0.15507998  0.28689797  0.28136327  0.27665878]
```

PageRank exercise 3.



- Guess what pages in the given model got the highest rank
- Check you guess by running the code

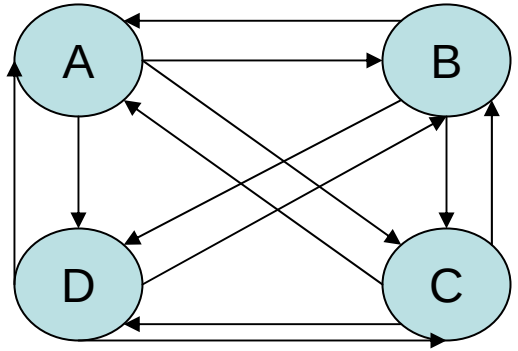
PageRank exercise 3.



- Guess what pages in the given model got the highest rank
- Check you guess by running the code

```
>>> execfile('PageRank.py')
[ 0.37252685  0.19582391  0.0375      0.39414924]
```

PageRank exercise 4.



- Guess what pages in the given model got the highest rank
- Check you guess by running the code

```
>>> execfile('PageRank.py')
[ 0.25  0.25  0.25  0.25]
```