Ranking web pages

Lecture 23

Information Retrieval (IR)

- IR core problem: Find documents relevant to user queries
- Web search has its root in IR.

From: Bing Liu. Web Data Mining. 2007

IR queries

- Boolean queries (using AND, OR, NOT)
- Bag of keywords queries
- Phrase queries
- Full document queries
- Natural language questions

Information retrieval models

- An IR model governs
  1. how a document and a query are represented and
  2. how the relevance of a document to a user query is defined.

- Main models:
  - Boolean model
  - Vector space model
  - Statistical language model
  - etc

Boolean model

- Each document or query is treated as a **"bag" of words** or **terms**. Word sequence is not considered.

- Given a collection of documents D, let $V = \{t_1, t_2, ..., t_{|V|}\}$ be the set of distinctive words/terms in the collection. V is called the vocabulary.

- A weight $w_{ij} > 0$ is associated with each term t_i of a document $\mathbf{d}_j$ in D.

  $$\mathbf{d}_j = (w_{1j}, w_{2j}, ..., w_{|V|j})$$

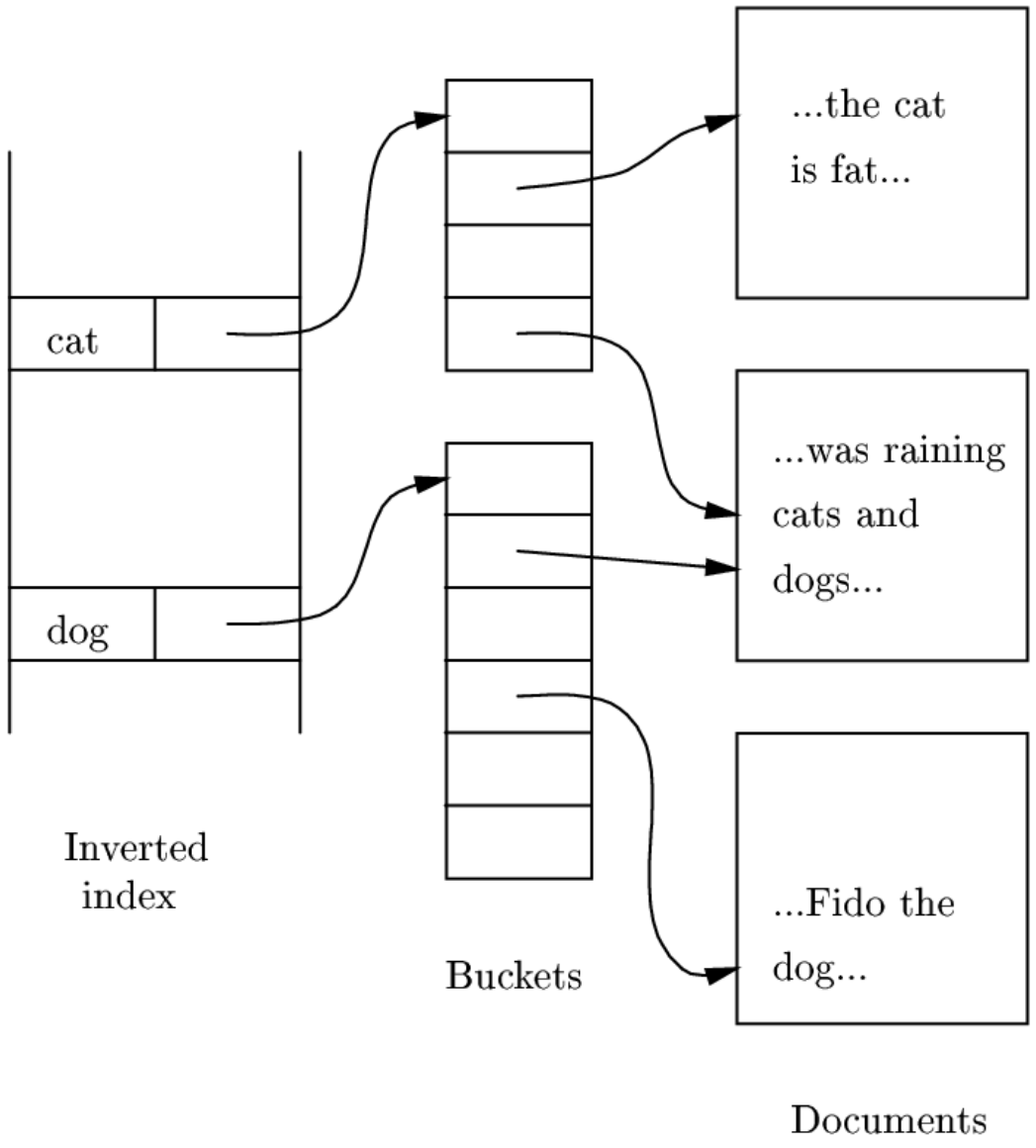
- For a term that does not appear in document $\mathbf{d}_j$, $w_{ij} = 0$.
- For a term that does appear in document $\mathbf{d}_j$, $w_{ij} = 1$.

Boolean model

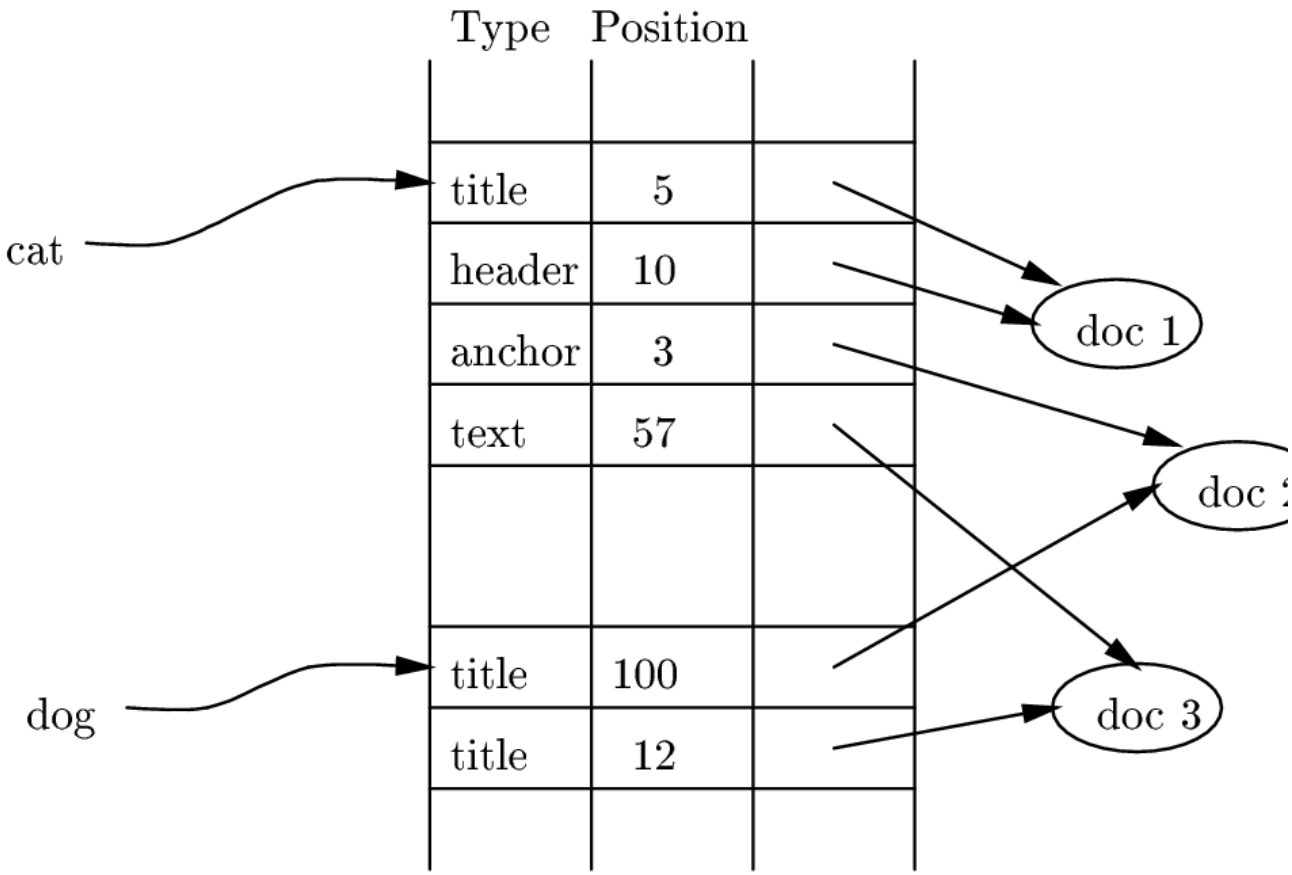
- Query terms are combined logically using the Boolean operators **AND**, **OR**, and **NOT.**
  - E.g., ((*data* AND *mining*) AND (NOT *text*))

- Retrieval
  - Given a Boolean query, the system retrieves every document that makes the query logically true.
  - Called **exact match**.

Inverted Indexes



Additional Information in Buckets



Vector space model

- Documents are also treated as a "bag" of words or terms.
  - Each document is represented as a vector.
  - However, the term weights are no longer 0 or 1.

- Term Frequency (TF) Scheme:
  - Weight of a term t_i in document $\mathbf{d}_j$ is the number of times that t_i appears in $\mathbf{d}_j$, denoted by f_{ij}.

- Shortcoming of the TF scheme is that it doesn't consider the situation where a term appears in many documents of the collection.
  - Such a term may not be discriminative.

Document Frequency

term	df_t
calpurnia	1
animal	100
sunday	1,000
fly	10,000
under	100,000
the	1,000,000

- Suppose query is: calpurnia animal

TF-IDF term weighting scheme

- The most well known weighting scheme
  - TF: (normalized) **term frequency**
  - IDF: **inverse document frequency**.

$$tf_{ij} = \frac{f_{ij}}{\max\{ f_{1j}, f_{2j}, \ldots, f_{|V|j} \}}$$

N: total number of docs

df_i: the number of docs that t_i appears.

$$idf_i = \log \frac{N}{df_i}$$

- The final TF-IDF term weight is:

$$w_{ij} = tf_{ij} \times idf_j$$

Each document will be a vector of such numbers.

IDF

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1,000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

Retrieval in the vector space model

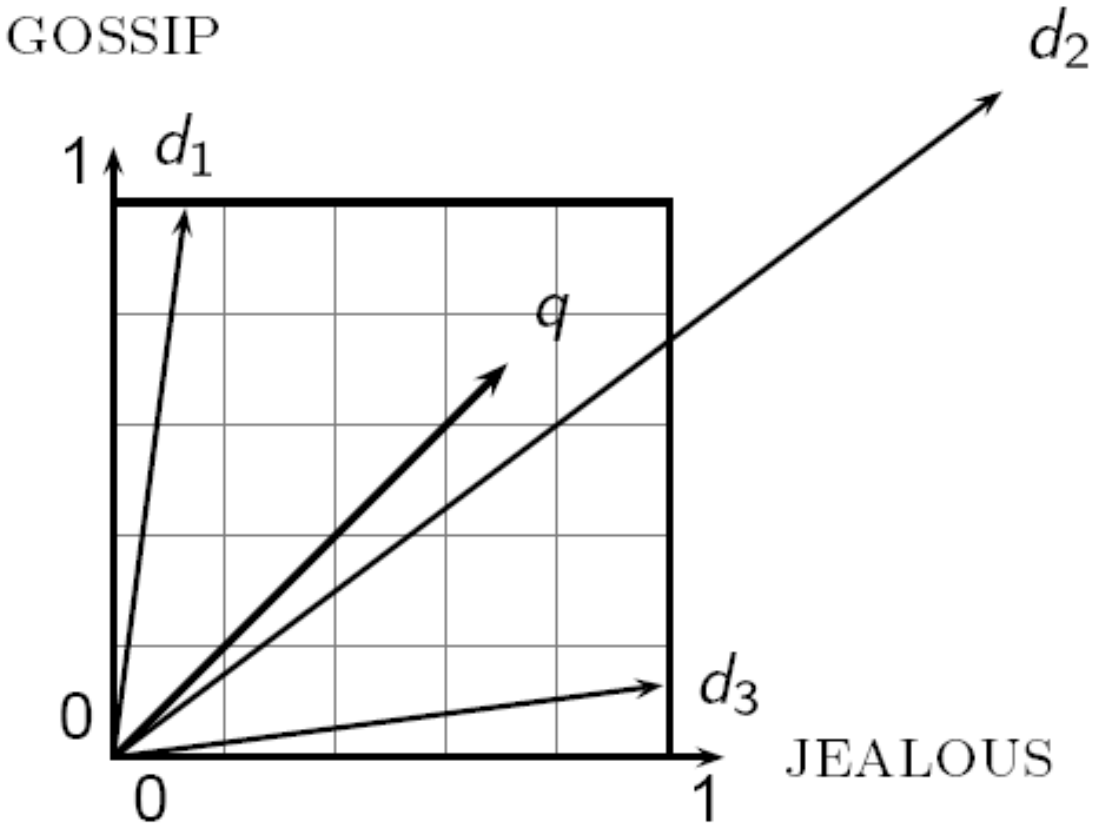
- Query **q** is represented in the same way as a document.
- The term w_{iq} of each term t_i in **q** can also computed in the same way as in normal document.
- Relevance of $\mathbf{d}_j$ to **q**: Compare the similarity of query **q** and document $\mathbf{d}_j$.
- For this, use cosine similarity (the cosine of the angle between the two vectors)
  - The bigger the cosine the smaller the angle and the higher the similarity

Dot product

Unit vectors

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q}\bullet\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|}\bullet\frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

Cosine similarity



Text pre-processing

1. Word (term) extraction: easy
2. Stopwords removal
3. Stemming
4. Frequency counts and computing TF-IDF term weights.

Stopwords removal

- Some of the most frequently used words aren't useful in IR and text mining – these words are called *stop words*.
  - the, of, and, to, ….
  - Typically about 400 to 500 such words
  - For an application, an additional domain specific stopwords list may be constructed

- Why do we need to remove stopwords?
  - Reduce indexing (or data) file size
    - stopwords accounts 20-30% of total word counts.
  - Improve efficiency and effectiveness
    - stopwords are not useful for searching or text mining
    - they may also confuse the retrieval system.

Stemming

- Techniques used to find out the root/stem of a word. E.g.,

user	engineering
users	engineered
used	engineer
using	

- stem: use engineer

Usefulness:

- improves the effectiveness of IR and text mining
- reduces index size
  - combing words with same roots may reduce indexing size as much as 40-50%.

Precision and Recall

In information retrieval (search engines) community, system evaluation revolves around the notion of *relevant* and *not relevant* documents.

Precision is the fraction of retrieved documents that are relevant

$$\text{Precision} = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})} = P(\text{relevant}|\text{retrieved})$$

Recall is the fraction of relevant documents that are retrieved

$$\text{Recall} = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} = P(\text{retrieved}|\text{relevant})$$

How do we compute the precision and recall?

Why having two numbers?

- The advantage of having the two numbers for precision and recall is that one is more important than the other in certain circumstances.

- Typical web surfers:
  - would like every result on the first page to be relevant (**high precision**), but have not the slightest interest in knowing let alone looking at every document that is relevant.

- Professional searchers such as paralegals and intelligence analysts:
  - are very concerned with trying to get as **high recall** as possible, and will tolerate fairly low precision results in order to get it.

What about a single number?

- The combined measure which is standardly used is called the *F measure*, which is the weighted harmonic mean of precision and recall:

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

where $\alpha = 1/(\beta^2+1)$

- The default is to equally weight precision and recall, giving a balanced F measure.
  - This corresponds to making $\alpha = 1/2$ or $\beta = 1$.
  - Commonly written as F_1, which is short for $F_{\beta=1}$

$$F_{\beta=1} = \frac{2PR}{P+R}$$

Why not arithmetic mean?

- Suppose, that only 1 document in 10,000 is relevant to a query.

- We can always get 100% recall by just returning all documents, and therefore we can always get a 50% arithmetic mean by the same process.
- In contrast, the harmonic mean score of the above strategy is 0.02%.

- The harmonic mean is closer to the minimum of two numbers than to their arithmetic mean.

Precision at *k*

- The above measures precision at all recall levels.

- What matters is rather how many good results there are on the first page or the first three pages.
- This leads to measures of precision at fixed low levels of retrieved results, such as 10 or 30 documents.

- This is referred to as "Precision at *k*", for example "Precision at 10."

Web Search as a huge IR system

- A Web crawler (robot) crawls the Web to collect all the pages.
- Servers establish a huge inverted indexing database and other indexing databases
- At query (search) time, search engines conduct different types of vector query matching.
  - There is an *Information Retrieval score* coming out of this.
- The documents have HREF links as well. They are used to compute a *reputation score*.
- The two scores are combined together in order to produce a ranking of the returned documents.

Google Page Ranking

“The Anatomy of a Large-Scale Hypertextual Web Search Engine”

by

Sergey Brin and Lawrence Page

http://www-db.stanford.edu/~backrub/google.html

Outline

1. Page rank, for discovering the most "important" pages on the Web, as used in Google.

2. Hubs and authorities, a more detailed evaluation of the importance of Web pages using a variant of the eigenvector calculation used for Page rank.

Page Rank (PR)

Intuitively, we solve the recursive definition of "importance":

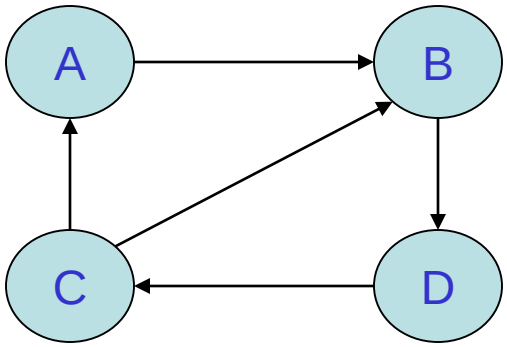
A page is important if important pages link to it.

- Page rank is the estimated page importance.

- In short PageRank is a "vote", by all the other pages on the Web, about how important a page is.
  - A link to a page counts as a vote of support.
  - If there's no link there's no support (but it's an abstention from voting rather than a vote against the page).

Ranking pages by Link Analysis: intuition

- 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

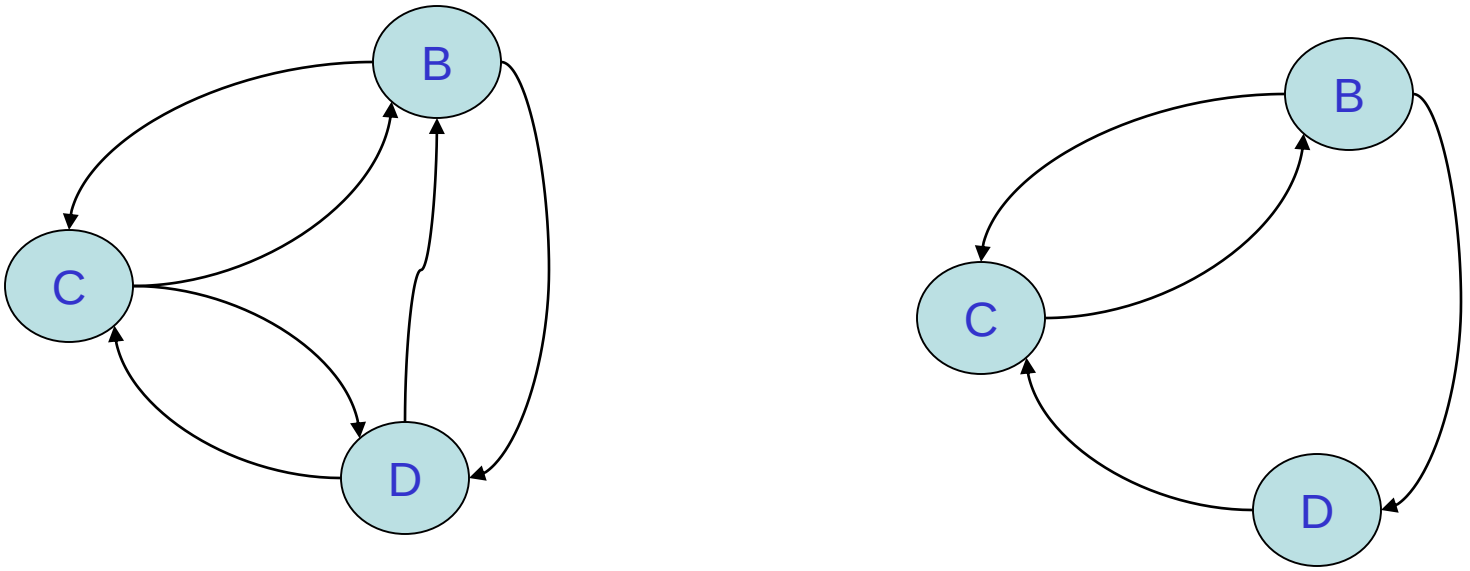
- Idea 2: a webpage is important if many important pages link to it.

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



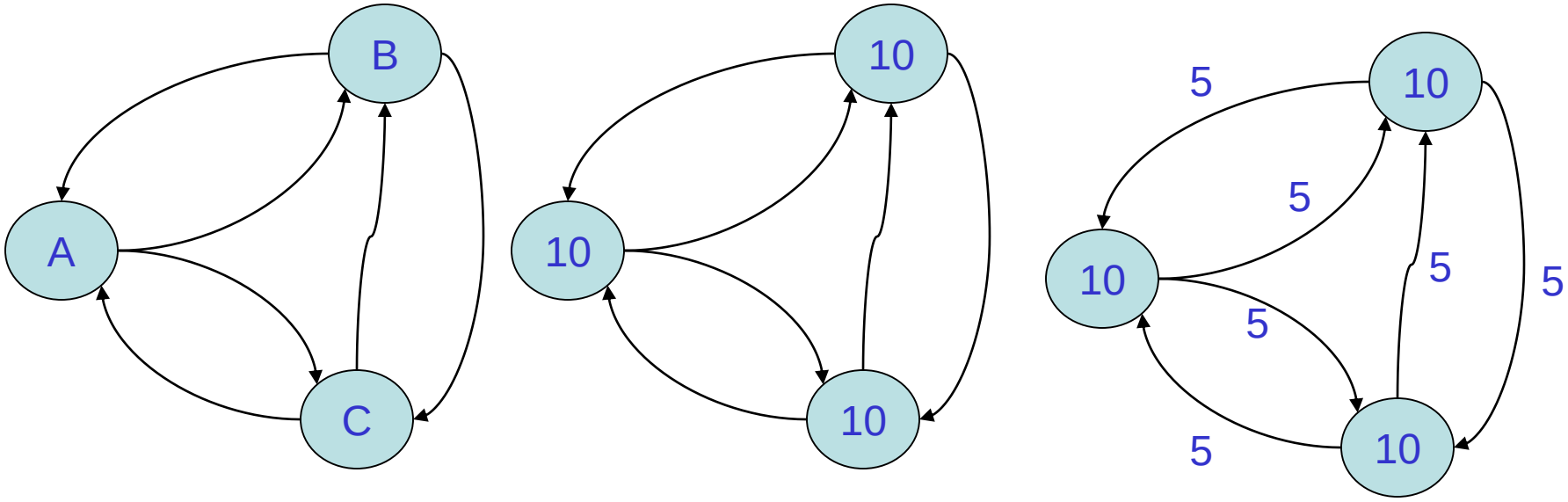
- What can we speculate about the relative importance of pages in each of these models, solely from the structure of the links (which is anyways the only information at hand)?

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:

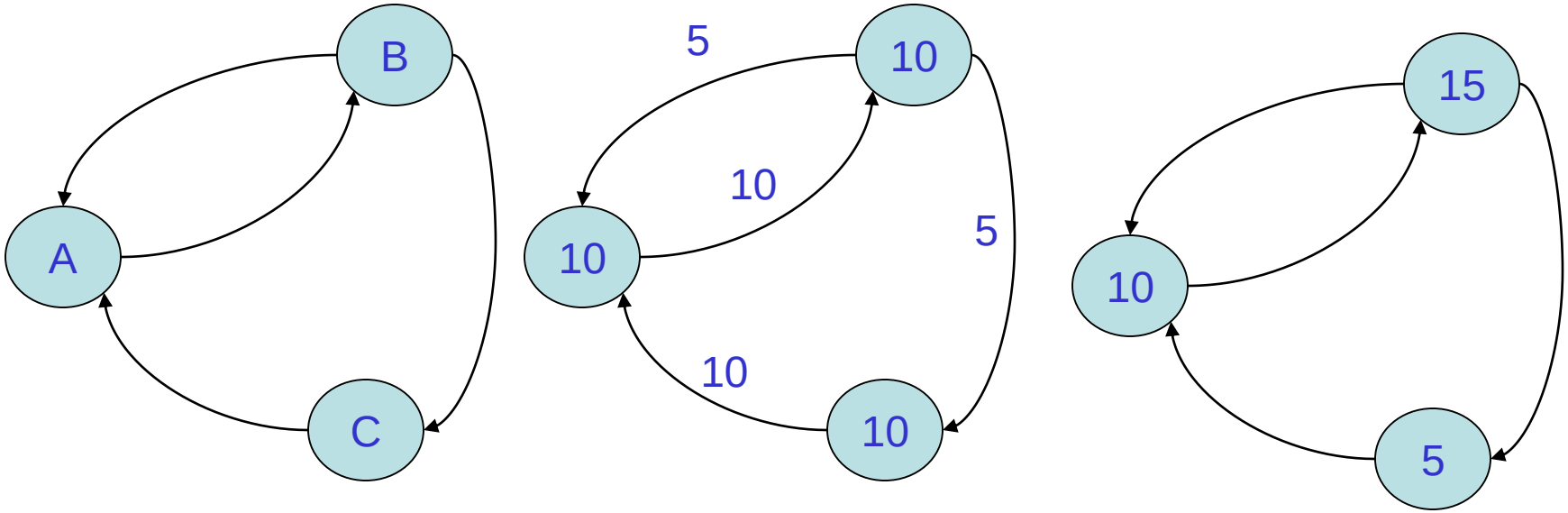
  Each page has at least one outgoing link.

Traffic and mindless surfing. Example 1



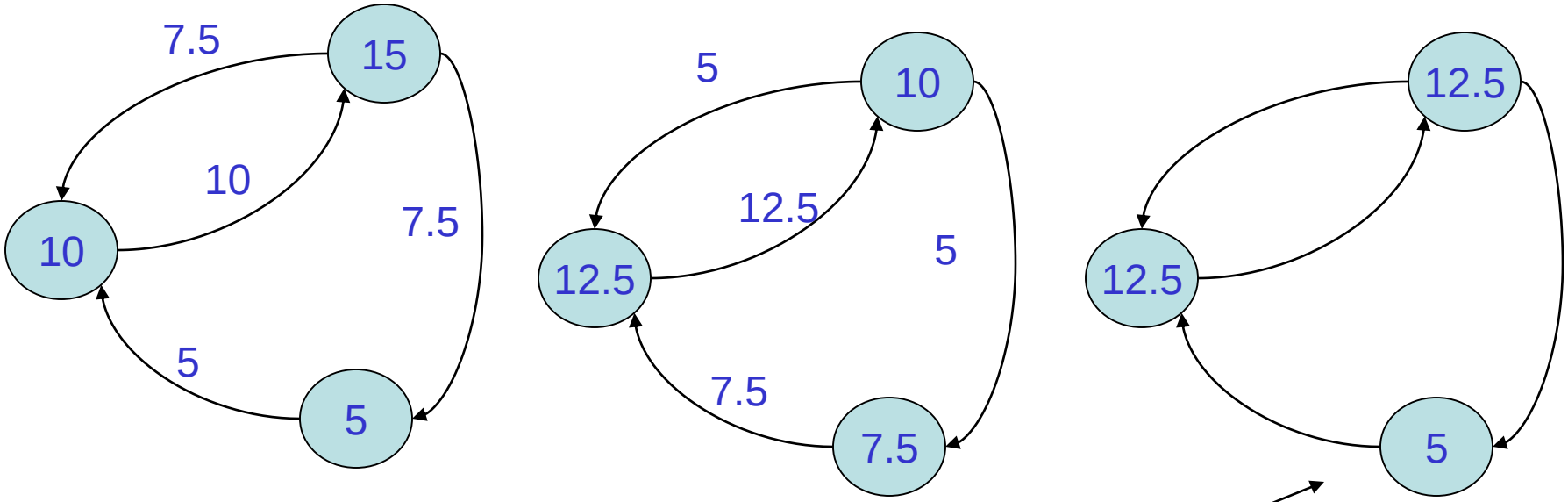
- We start with 10 surfers in each page
- At the first random click, 5 of the surfers at page A, say, go to page B, and the other 5 go to page C. So while each site sees all 10 of its visitors leave, it gets 5 + 5 incoming visitors to replace them: **So the amount of traffic at each page remains constant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

Traffic and mindless surfing. Example 2



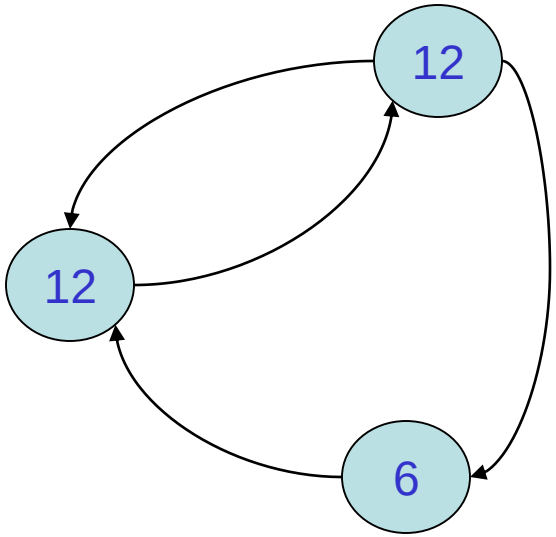
- We start with 10 surfers in each page
- After the first random click, 10 of the surfers at page A go to page B, since there is only 1 outgoing link from A etc…

Traffic and mindless surfing. Example 2



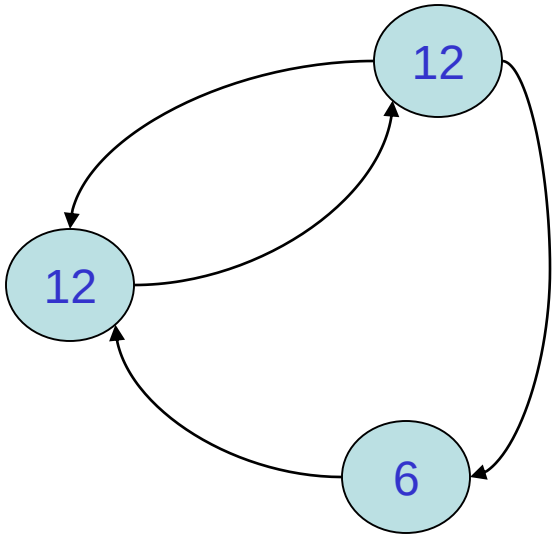
- After the two next clicks it becomes
- Where is this leading? Do we ever reach a stable configuration, as in the first model?

Traffic and mindless surfing. Example 2



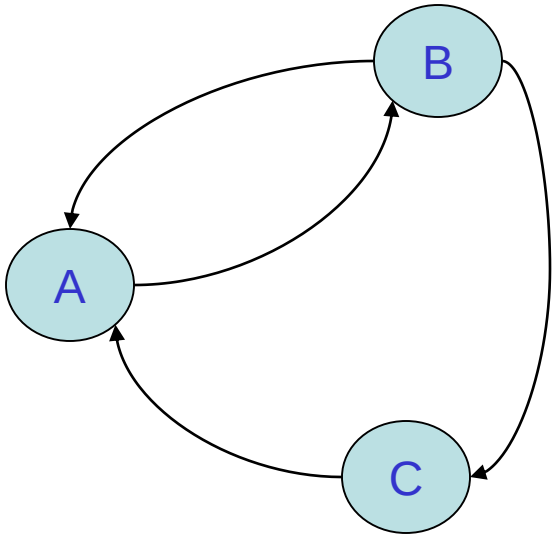
- While the answer is no, it turns out that the process **converges** to the following distribution, which (you can check) remains the same going forward in time

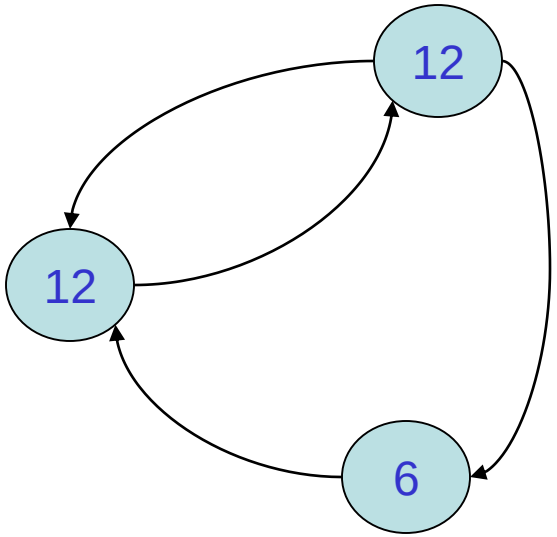
Traffic and mindless surfing. Example 2



- 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.

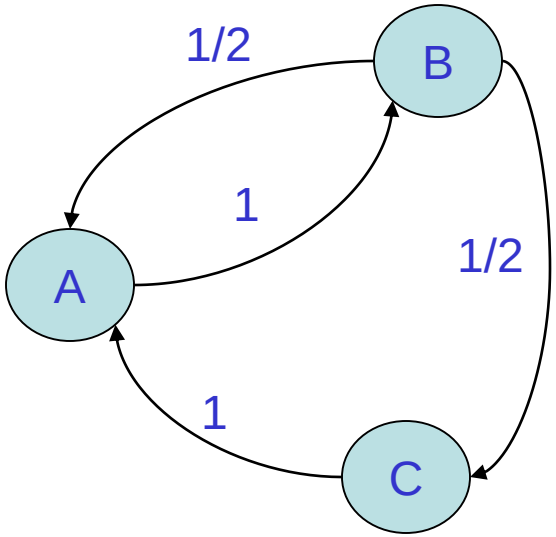
Traffic and mindless surfing. Question?





- 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?

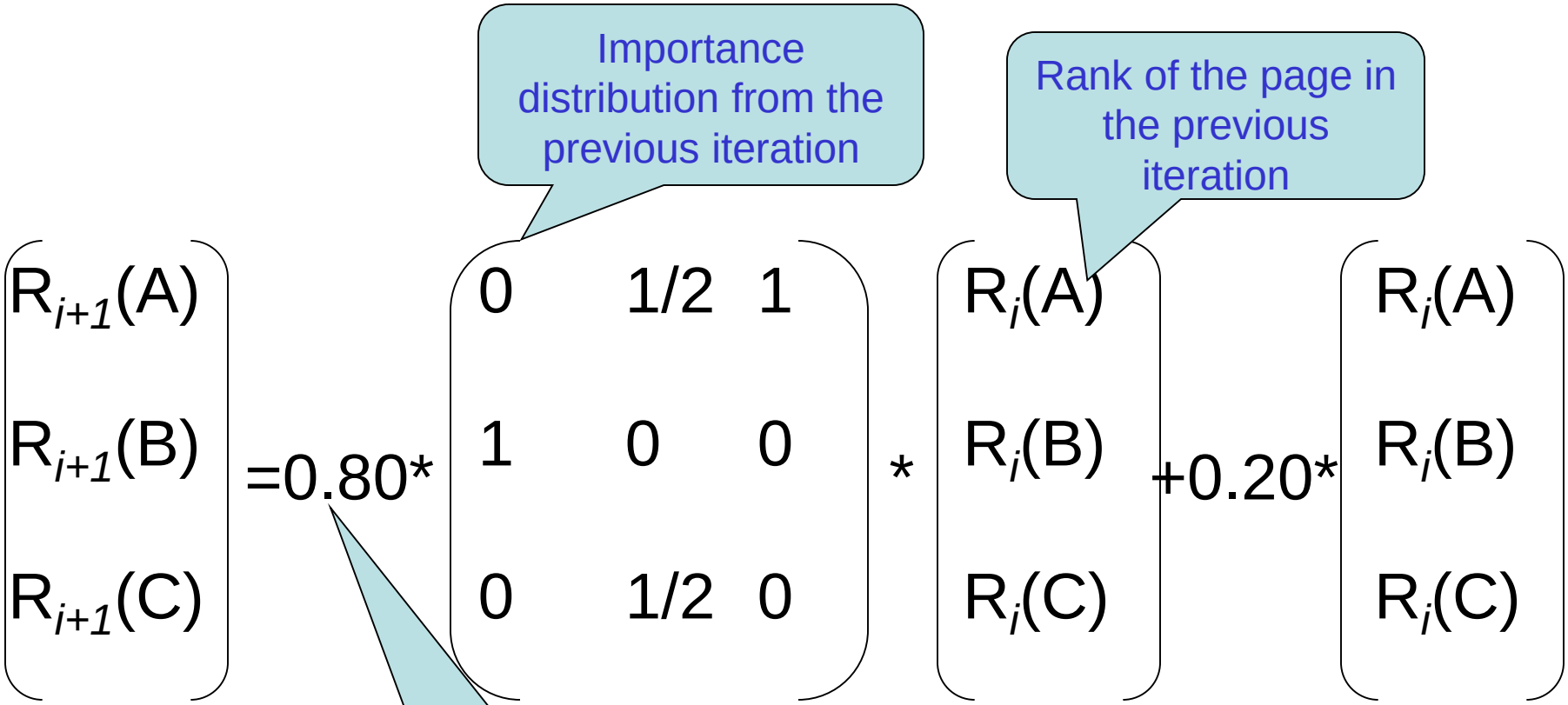
How to compute the stable distribution?



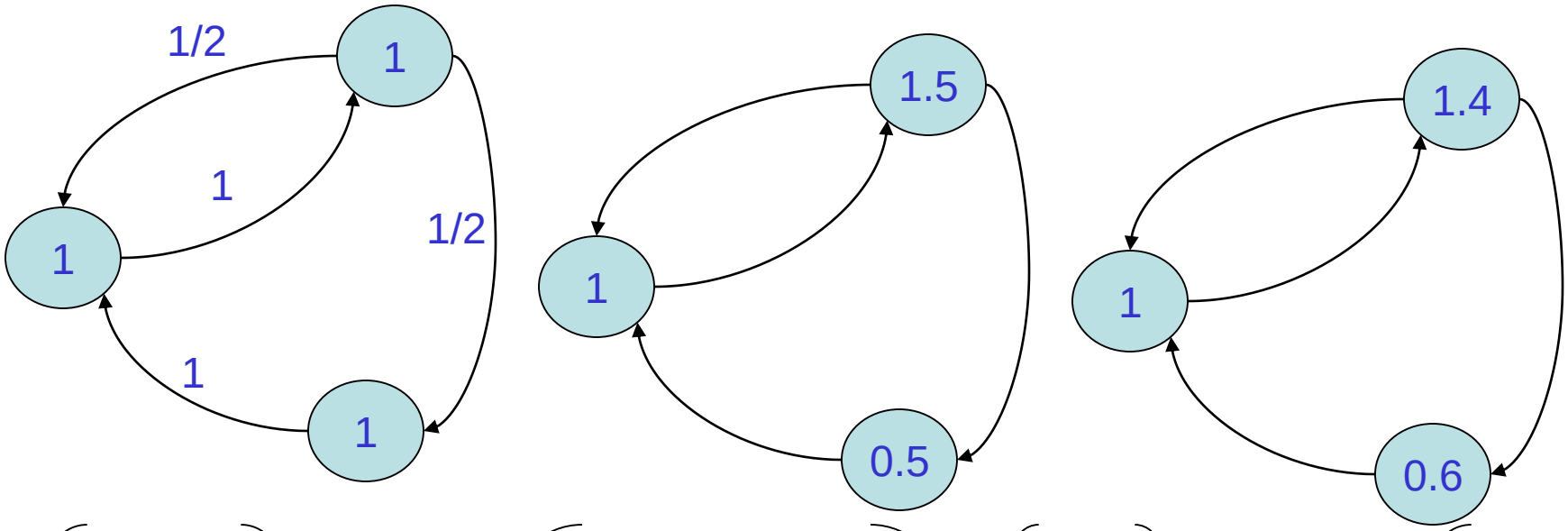
Create matrix of the importance distribution from the current state

	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*

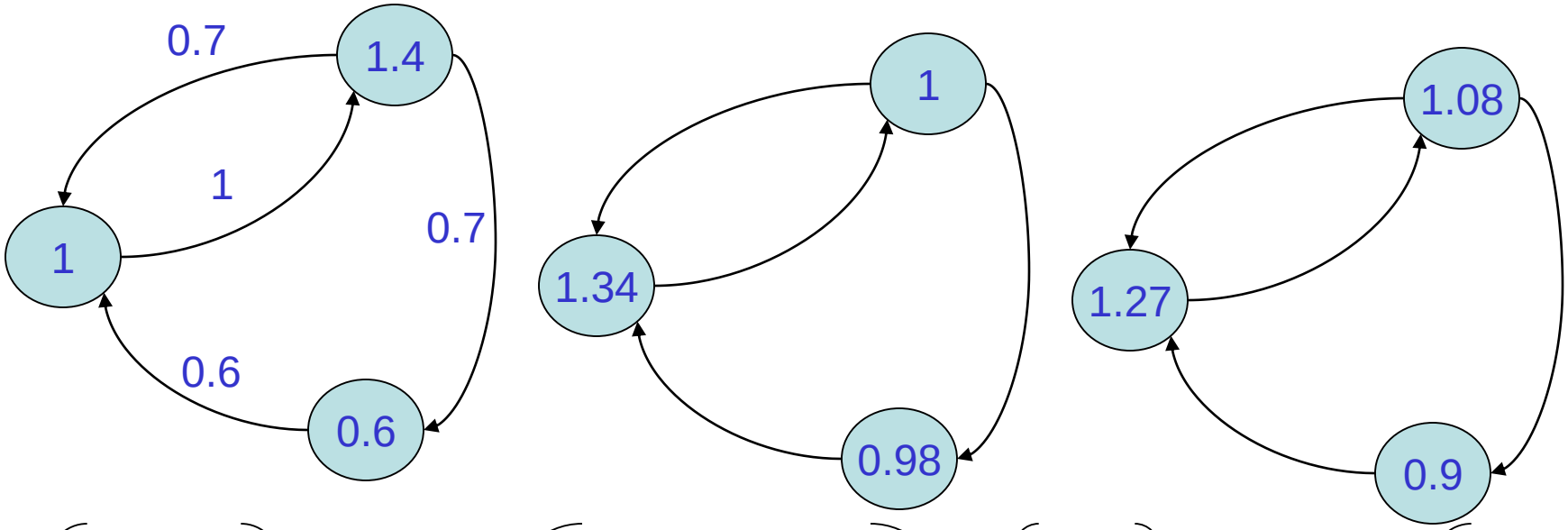


Computing page rank for iteration 1



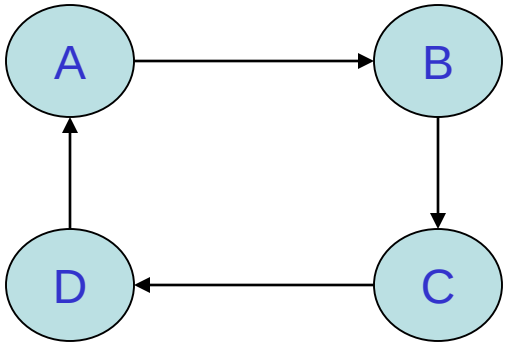
$$\begin{pmatrix} R_2(A) \\ R_2(B) \\ R_2(C) \end{pmatrix} = 0.80 * \begin{pmatrix} 0 & 1/2 & 1 \\ 1 & 0 & 0 \\ 0 & 1/2 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + 0.20 * \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Computing page rank for iteration 2



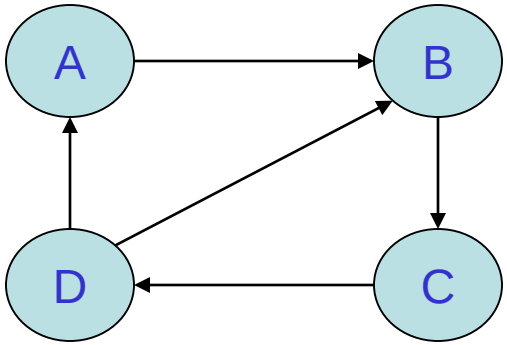
$$\begin{pmatrix} R_3(A) \\ R_3(B) \\ R_3(C) \end{pmatrix} = 0.80 * \begin{pmatrix} 0 & 0.7 & 0.6 \\ 1 & 0 & 0 \\ 0 & 0.7 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 1.4 \\ 0.6 \end{pmatrix} + 0.20 * \begin{pmatrix} 1 \\ 1.4 \\ 0.6 \end{pmatrix}$$

PageRank exercise 1.



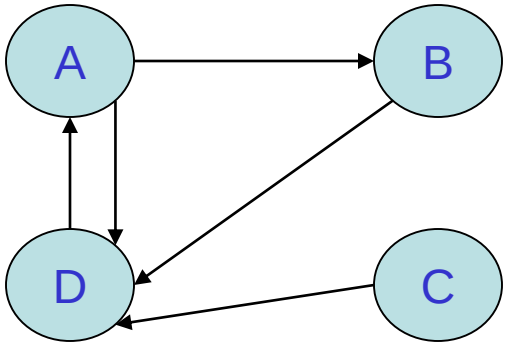
- Guess what pages in the given model got the highest rank

PageRank exercise 2.



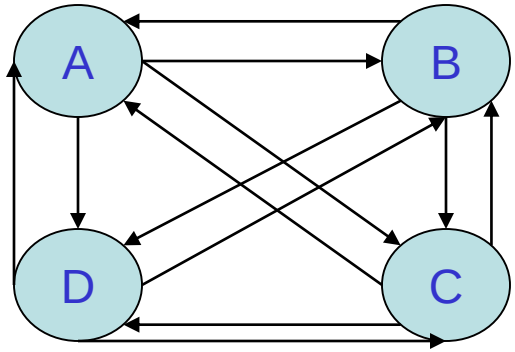
- Guess what pages in the given model got the highest rank

PageRank exercise 3.



- Guess what pages in the given model got the highest rank

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]
```

Page Rank: General Formula

$PR(A) = PR(T_1)/C(T_1) + \ldots + PR(T_n)/C(T_n)$

1. **$PR(T_n)$** - Each page has a notion of its own self-importance, which is say 1 initially.
2. **$C(T_n)$** – Count of outgoing links from page T_n.
    1. Each page spreads its vote out evenly amongst all of it's outgoing links.
3. **$PR(T_n)/C(T_n)$** –
    a) Each page spreads its vote out evenly amongst all of it's outgoing links.
    b) So if our page (say page A) has a back link from page "*n*" the share of the vote page A will get from page "*n*" is "**$PR(T_n)/C(T_n)$.**"

How is Page Rank Calculated?

- The page rank (PR) of each page depends on the PR of the pages pointing to it.
  - We won't know what PR those pages have until the pages pointing to them have their PR calculated and so on…

- Well, just go ahead and calculate a page's PR without knowing the final value of the PR of the other pages.
  - Each time we run the calculation we're getting a closer estimate of the final value.
  - Repeat the calculations lots of times until the numbers converge.

Web Matrix

Capture the formula by the web matrix (M) that is:

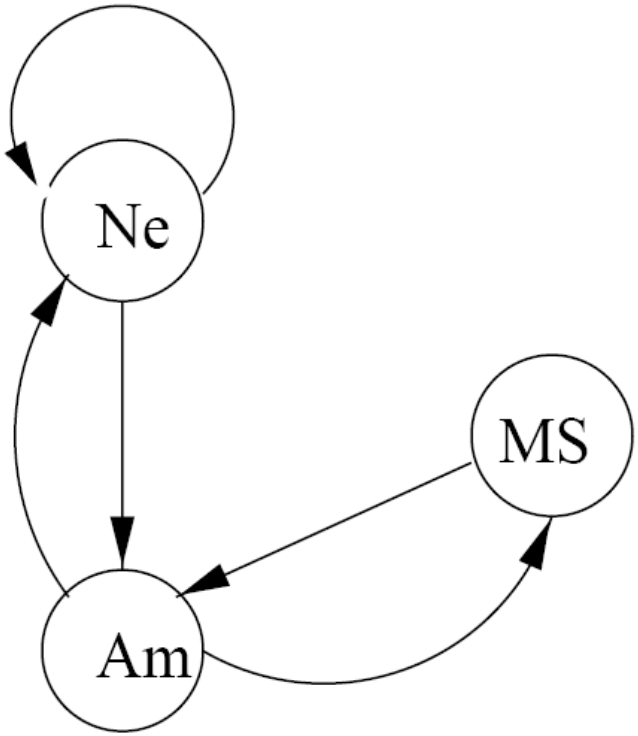
- If page *j* has *n* successors (links), then:
  - M[*i*, *j*] =1/*n* if page *i* is one of these *n* successors of page *j*, and
  - 0 otherwise.

Then, the importance vector containing the rank of each page is calculated by:

$$Rank_{new} = M \bullet Rank_{old}$$

Example

- In 1839, the Web consisted on only three pages: Netscape, Microsoft, and Amazon.



$$\begin{bmatrix} n_{new} \\ m_{new} \\ a_{new} \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 \\ 1/2 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} n_{old} \\ m_{old} \\ a_{old} \end{bmatrix}$$

For example, the first column of the Web matrix reflects the fact that Netscape divides its importance between itself and Amazon.

The second column indicates that Microsoft gives all its importance to Amazon.

Start with n = m = a = 1, then do rounds of improvements.

Based on Jeff Ullman's notes

Example

- The first four iterations give the following estimates:

n = 1	1	5/4	9/8	5/4
m = 1	1/2	3/4	1/2	11/16
a = 1	3/2	1	11/8	17/16

- In the limit, the solution is n = a = 6/5; m = 3/5.

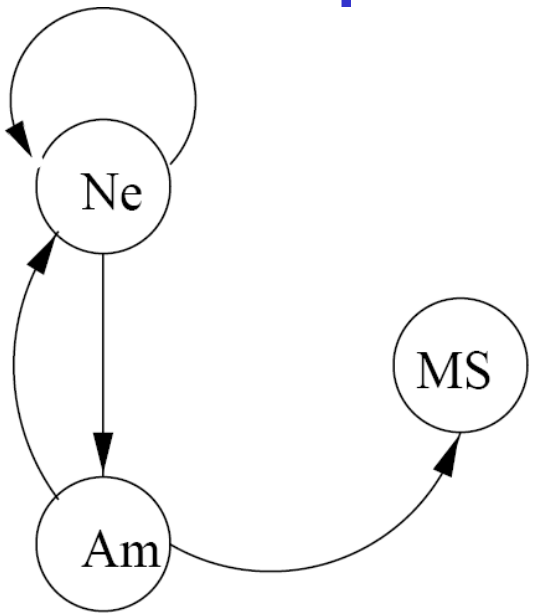
- That is, Netscape and Amazon each have the same importance, and twice the importance of Microsoft (well this was 1839).

Problems With Real Web Graphs

Dead ends: a page that has no successors has nowhere to send its importance.

Eventually, all importance will "leak out of" the Web.

Example: Suppose Microsoft tries to claim that it is a monopoly by removing all links from its site.



The new Web, and the rank vectors for the first 4 iterations are shown.

$$\begin{bmatrix} n_{new} \\ m_{new} \\ a_{new} \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 \\ 1/2 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} n_{old} \\ m_{old} \\ a_{old} \end{bmatrix}$$

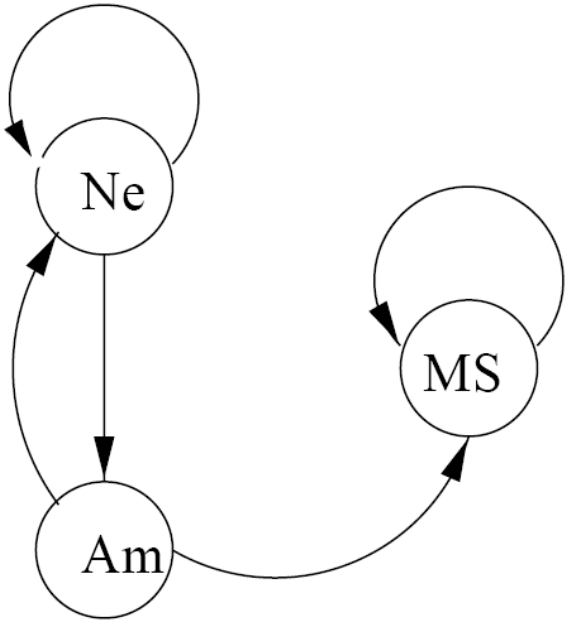
Eventually, each of *n*, *m*, and *a* become 0; i.e., all the importance leaked out.

n =	1	1	3/4	5/8	1/2
m =	1	1/2	1/4	1/4	3/16
a =	1	1/2	1/2	3/8	5/16

Based on Jeff Ullman's notes

Problems With Real Web Graphs

Spider traps: a group of one or more pages that have no links out of the group will eventually accumulate all the importance of the Web.



Example: Angered by the decision, Microsoft decides it will link only to itself from now on. Now, Microsoft has become a spider trap.

The new Web, and the rank vectors for the first 4 iterations are shown.

$$\begin{bmatrix} n_{new} \\ m_{new} \\ a_{new} \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 1 & 1/2 \\ 1/2 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} n_{old} \\ m_{old} \\ a_{old} \end{bmatrix}$$

n =	1	1	3/4	5/8	1/2
m =	1	3/2	7/4	2	35/16
a =	1	1/2	1/2	3/8	5/16

Now, m converges to 3, and $n = a = 0$.

Google Solution to Dead Ends and Spider Traps

Stop the other pages having too much influence.

This total vote is "damped down" by multiplying it by a factor.

Example: If we use a 20% damp-down, the equation of previous example becomes:

$$\begin{bmatrix} n_{new} \\ m_{new} \\ a_{new} \end{bmatrix} = 0.80 \cdot \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 1 & 1/2 \\ 1/2 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} n_{old} \\ m_{old} \\ a_{old} \end{bmatrix} + 0.20 \cdot \begin{bmatrix} n_{old} \\ m_{old} \\ a_{old} \end{bmatrix}$$

The solution to this equation is $n = 7/11$; $m = 21/11$; $a = 5/11$.