Web Search

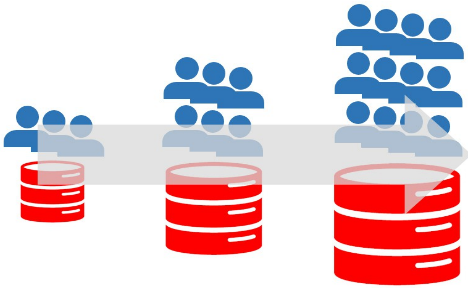
Centro Singular de Investigación en Tecnoloxías Intelixentes

Tecnologías de Gestión de Información No Estructurada

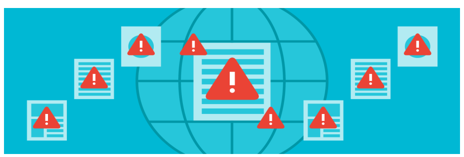
Prof. Dr. David E. Losada

Máster Interuniversitario en Tecnologías de Análisis de Datos Masivos: Big Data

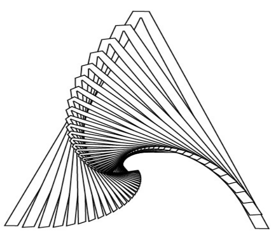
 web search



scalability: coverage, # users



low quality: spam, SEO



dynamism: freshness of the index

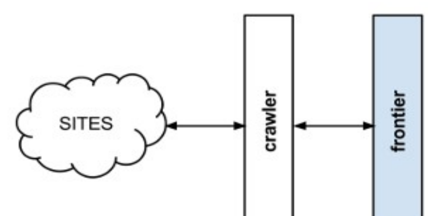
additional evidence: **hyperlinks**

web crawling

The **crawler** (a.k.a. spider or robot) crawls (<u>traverses, parses, and downloads</u>) pages on the web

starts with a set of **seed pages**, fetch pages from the web, and parse these pages’ **new links**.

adds them to a **queue** and then explore those page’s links in a <u>breadth-first search</u>



web crawling

robustness: what if the server doesn't respond or returns unparseable garbage?

spider traps (dynamically generated pages that attract your crawler to keep crawling the same site in circles)

we don't want to **overload** one particular server with too many crawling requests

denial of service

respect robots.txt

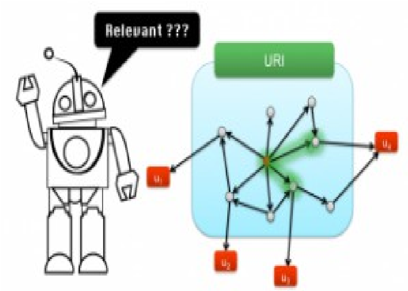
web crawling

handle **different types of files** such as images, PDFs, …

duplicate pages

discover **hidden URLs (**URLs that may not be linked from any page yet still contain content that you’d like to index)

web crawling

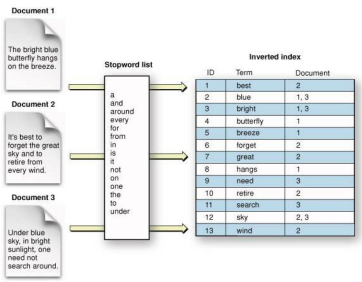


focused crawling: crawl some pages about a particular topic, e.g., all pages about automobiles

revisits: how can we determine when a page needs to be recrawled ?

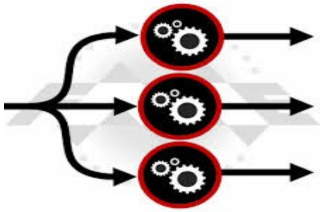
or even when a new page has been created?

web indexing



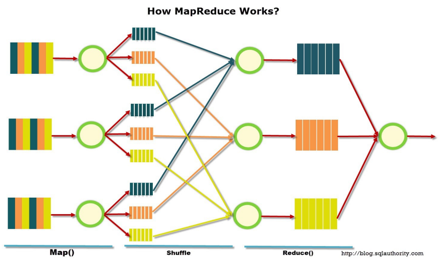
the index will be so **large** that it cannot actually fit into any single machine

or single disk, so we have to store the data on **multiple machines**

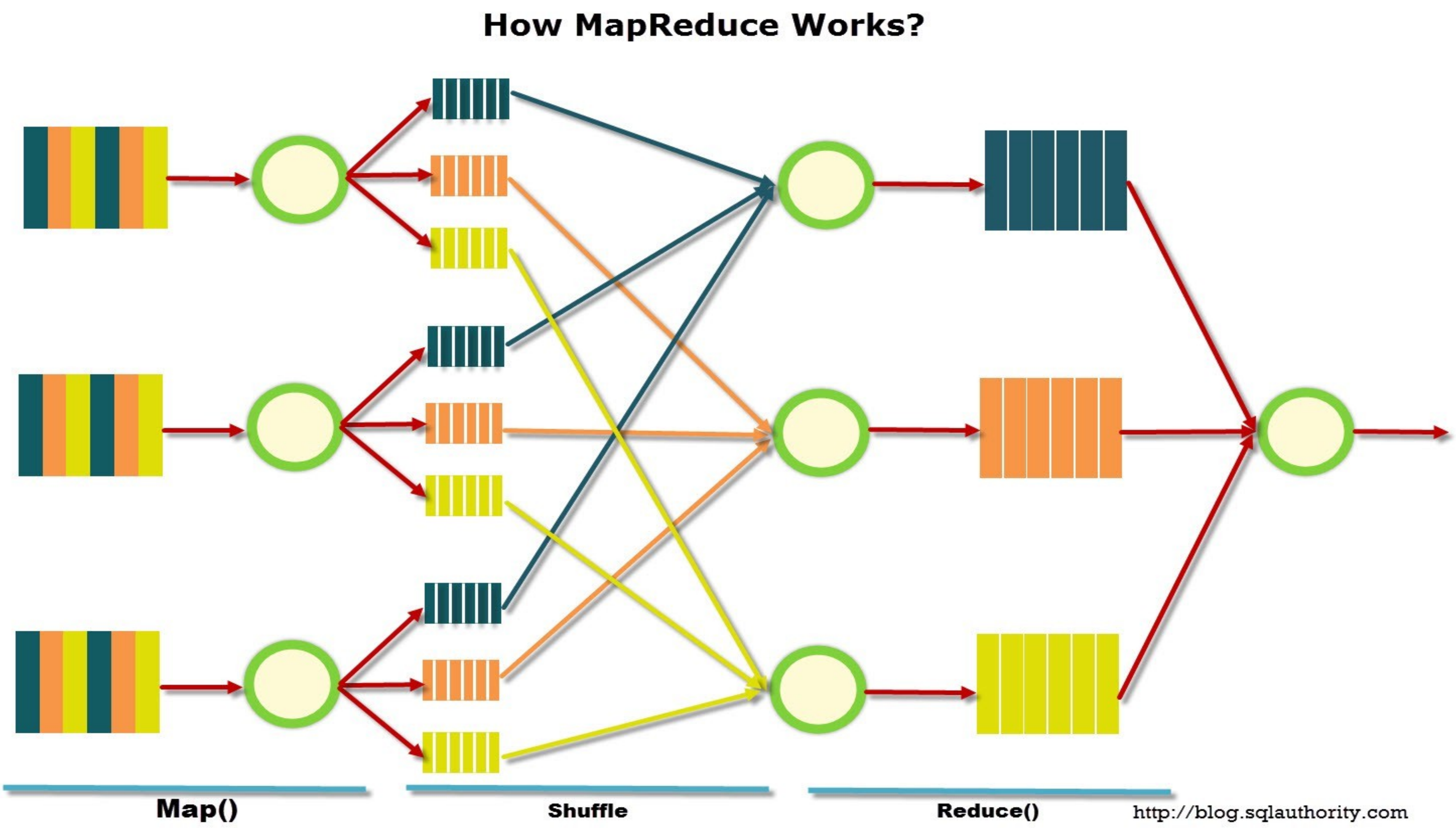


parallel processing

Google File System, a general distributed file system that can help programmers manage files stored on a **cluster of machines**



MapReduce, a general software framework for supporting parallel computation. Hadoop is the most well known open source implementation of MapReduce


How MapReduce Works?
Map()
Shuffle
Reduce()
http://blog.sqlauthority.com

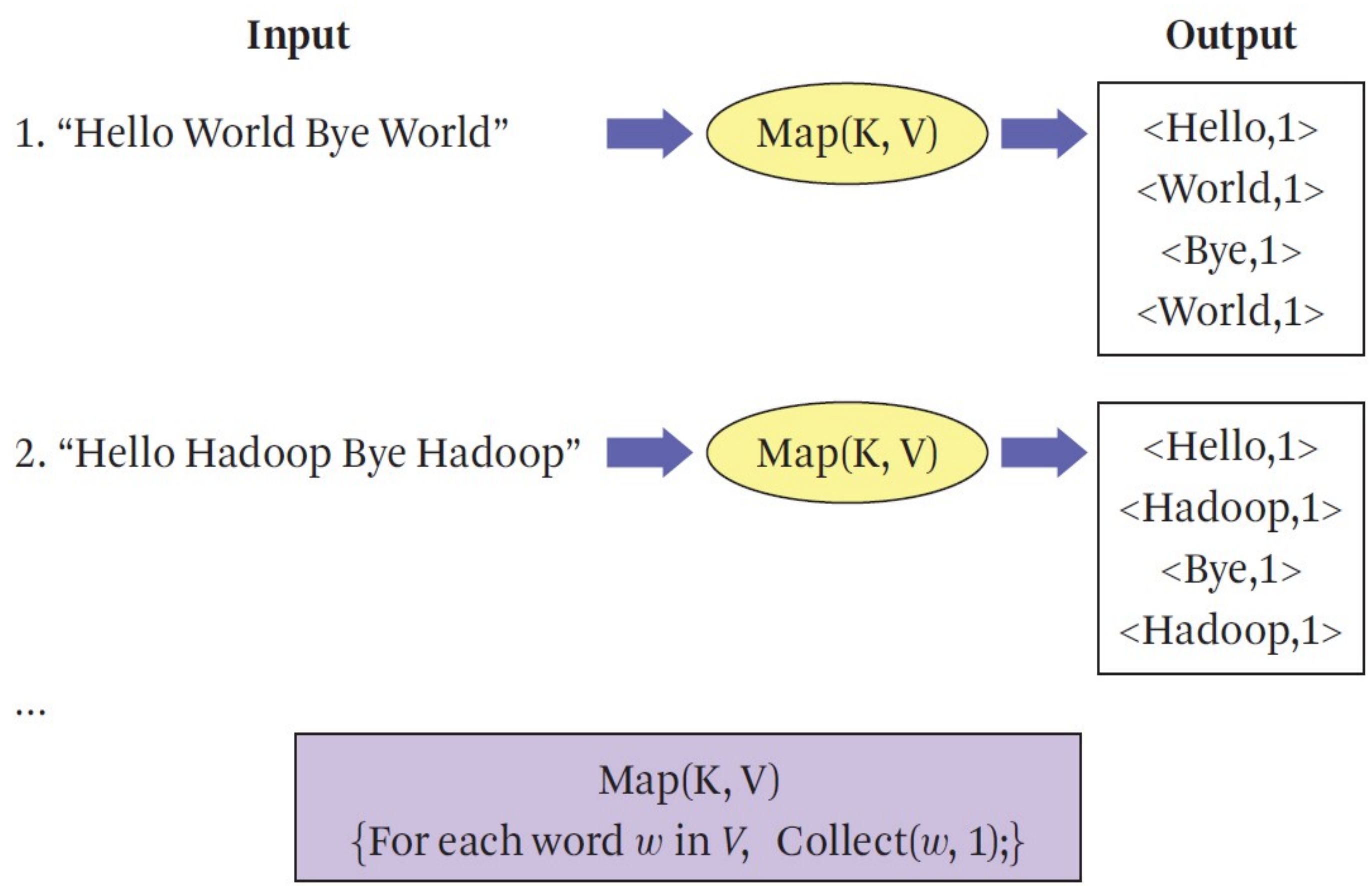


Figure 10.3 The `map` function for word counting.

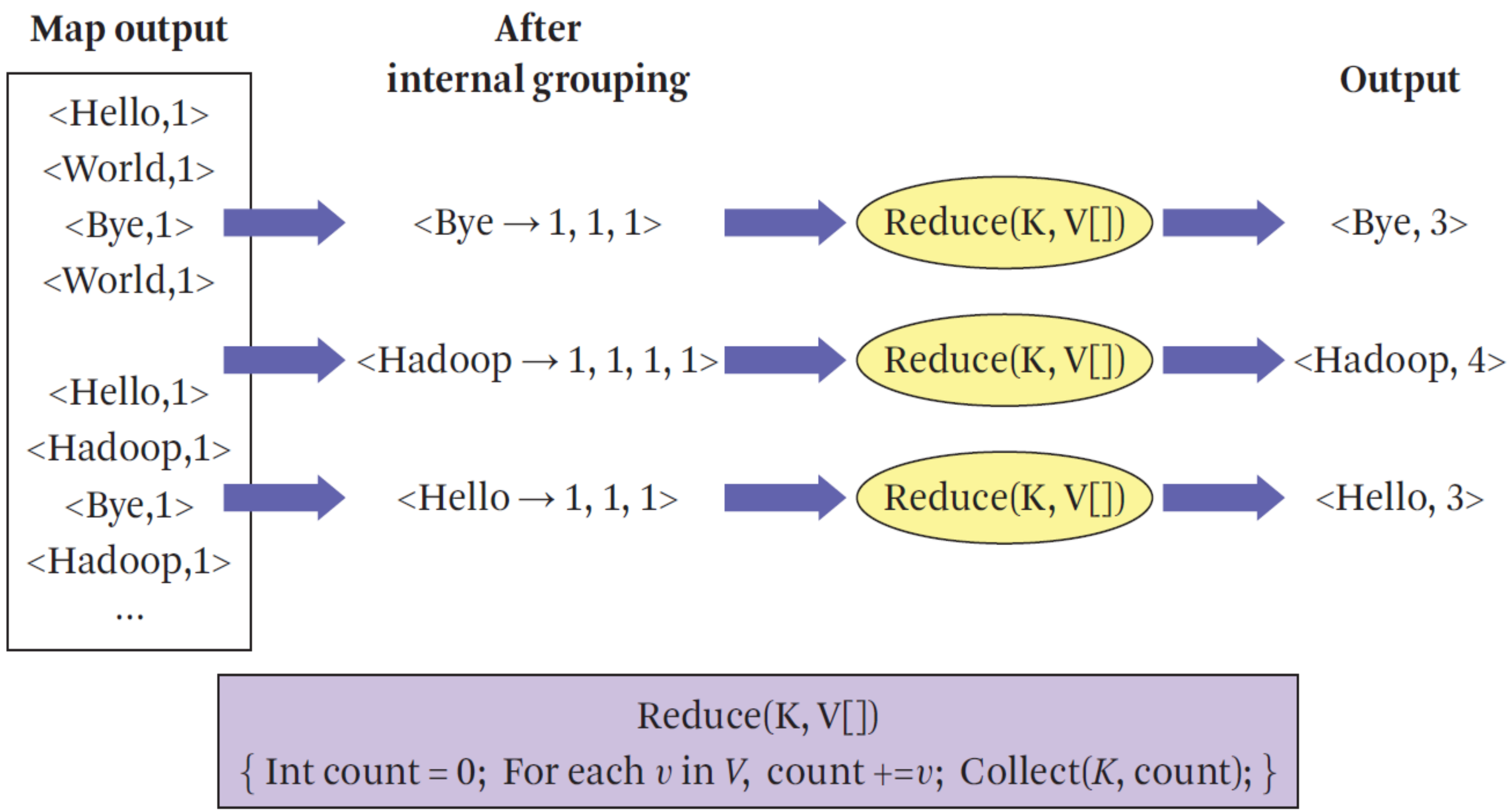
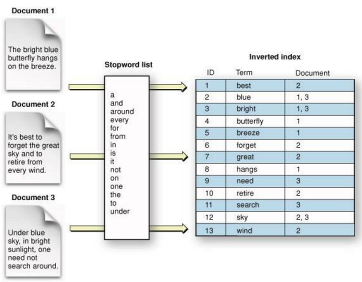


Figure 10.4 The `reduce` function for word counting.

web indexing



D1: java resource java class | D2: java travel resource | D3: …

Map

Key	Value
java	(D1, 2)
resource	(D1, 1)
class	(D1, 1)

Key	Value
java	(D2, 1)
travel	(D2, 1)
resource	(D2, 1)

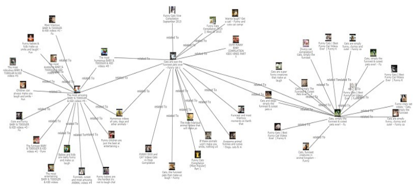
Built in Shuffle and Sort: aggregate values by keys

Reduce

Key	Value
java	{(D1, 2), (D2, 1)}
resource	{(D1, 1), (D2, 1)}
class	{(D1, 1)}
travel	{(D2, 1)}

Figure 10.5 Using MapReduce to create an inverted index.

link analysis



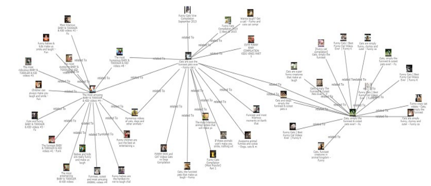
how to utilize **links** between pages to **improve search**

link features provide an opportunity to use **extra context information** of the document to improve scoring

links give us a more **robust way to rank the pages**

more difficult for spammers to manipulate one signal to improve a single page's ranking

link analysis



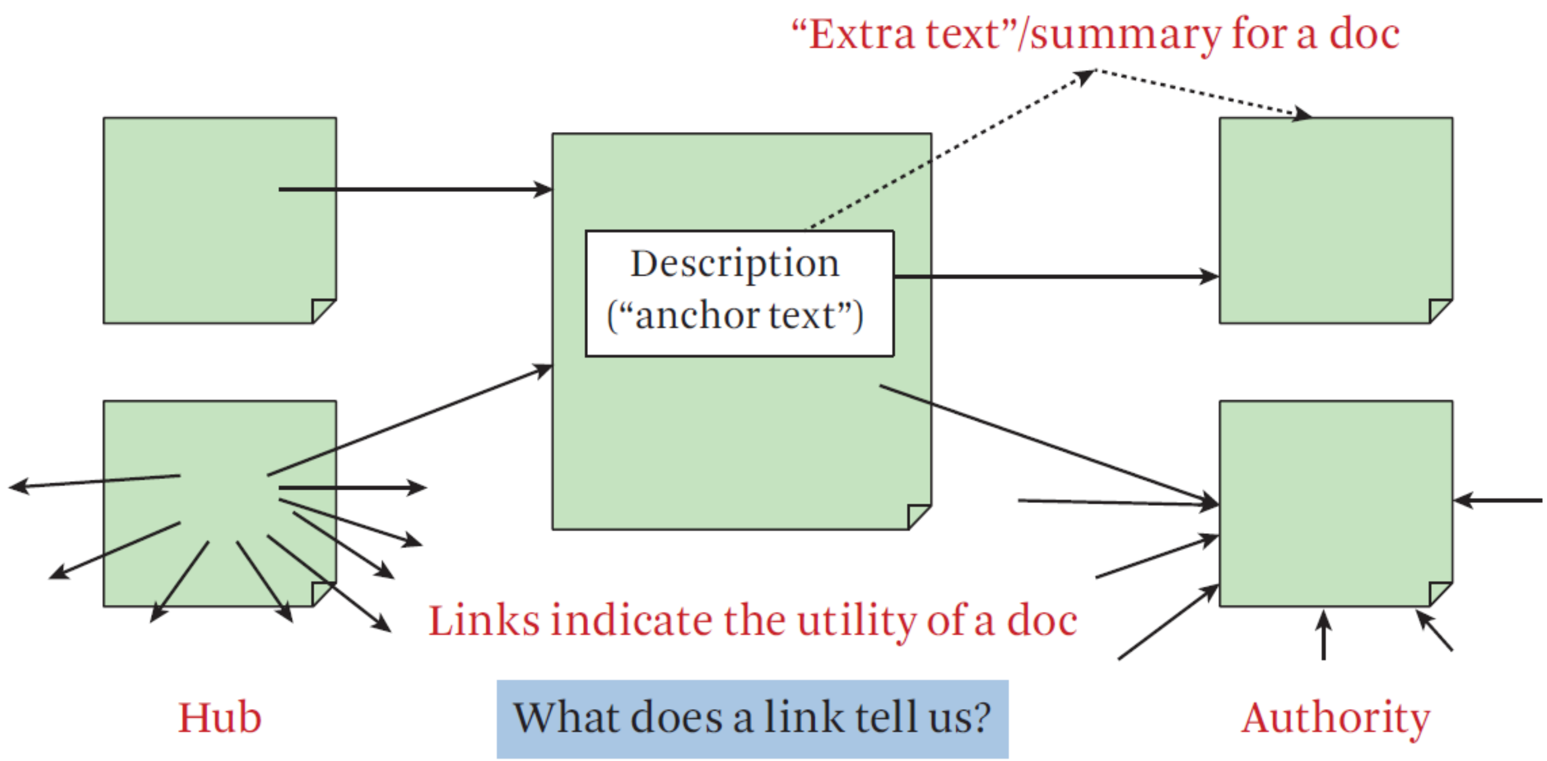


Figure 10.6 Links provide useful information about pages.

pagerank

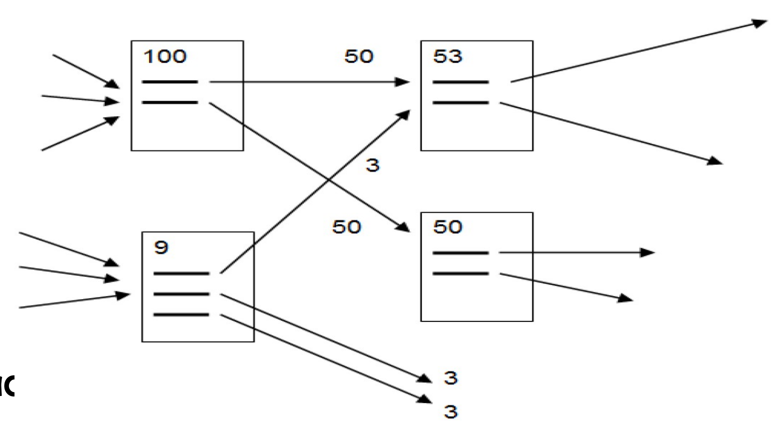
captures page **popularity (authority)**

if a page is cited often, we assume this page is more useful

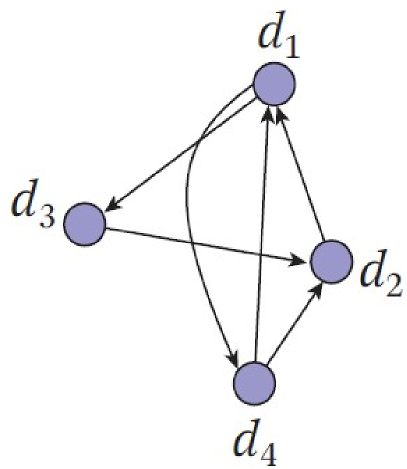
PageRank is essentially doing citation counting or **inlink** counting

But considers **indirect citations**, you don’t just look at the number of inlinks, rather you also look at the inlinks of your inlinks, recursively

if your inlinks themselves have many inlinks, your page gets credit from that.



pagerank – random surfer



Transition matrix

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

M_{ij} = probability of going from d_i to d_j

$$\sum_{j=1}^{N} M_{ij} = 1$$

Figure 10.7 Example of a web graph and the corresponding transition matrix.

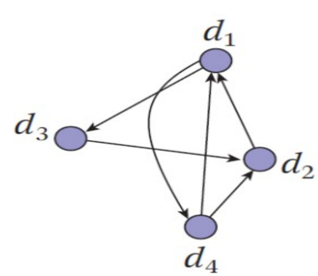
the random surfing model also assumes that the surfer **might get bored** sometimes and decide to ignore the actual links, randomly jumping to any page on the web

<u>“How likely, on average, would the surfer reach a particular page?”</u> this probability is precisely what PageRank computes

pagerank

$$p_{t+1}(d_j) = \underbrace{(1-\alpha)\sum_{i=1}^{N} M_{ij}\, p_t(d_i)}_{\text{reach } d_j \text{ by following a link}} + \underbrace{\alpha \sum_{i=1}^{N} \frac{1}{N} p_t(d_i)}_{\text{reach } d_j \text{ by random jumping}}$$

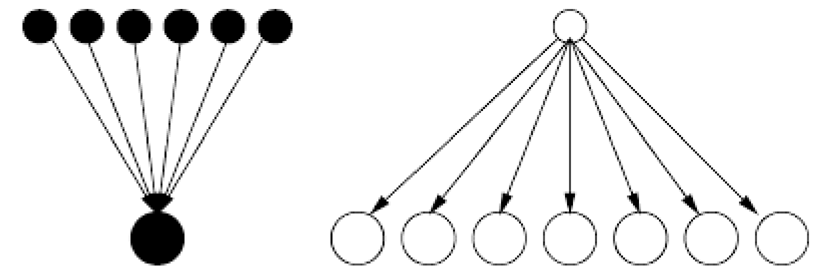
PageRank is **<u>query-independent</u>**, **<u>grows with the number of inlinks</u>** that a page has, and **<u>decreases with the number of outlinks that the linking pages have</u>**

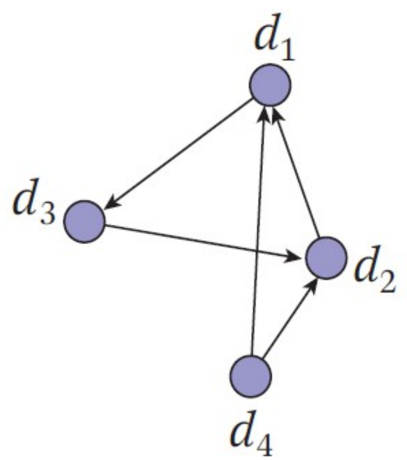


Transition matrix

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

hits





Adjacency matrix

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Initial values: $a(d_i) = h(d_i) = 1$

$$\left.\begin{aligned} h(d_i) &= \sum_{d_j \in OUT(d_i)} a(d_j) \\ a(d_i) &= \sum_{d_j \in IN(d_i)} h(d_j) \end{aligned}\right\} \text{ Iterate}$$

$$\bar{h} = A\bar{a}; \quad \bar{a} = A^T\bar{h}$$
$$\bar{h} = AA^T\bar{h}\,; \quad \bar{a} = A^TA\bar{a}$$

Normalize:

$$\sum_i a(d_i)^2 = \sum_i h(d_i)^2 = 1$$

Figure 10.8 Running the HITS algorithm on a small graph.

learning to rank

using **machine learning** to combine **many different features** into a single ranking function to optimize search results

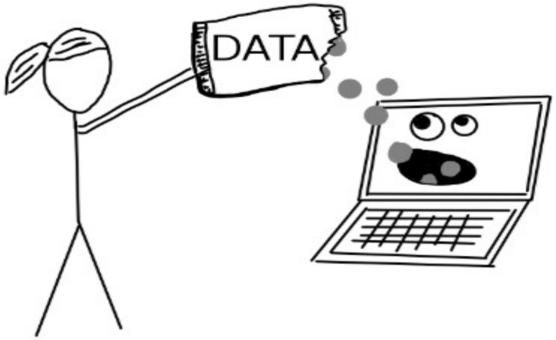
given a query-document pair (q,d)…

- content-based score: e.g., BM25 or query likelihood
- link-based score: e.g., PageRank
- other scores: e.g., the application of retrieval models to the anchor text of the page

how can we **combine all these features** (and potentially many other features) to do ranking?

learning to rank uses machine learning to combine these features, optimizing the **weight on different features** to generate the best ranking function

learning to rank



how do we know which **features** should have high weight and which features should have low weight? this is a task of **training** or learning

training data. data that have been judged by users, so we already know the relevance judgments.

can be based on real judgments by users or can be approximated by just using **clickthrough information**

optimize our search engine's retrieval **accuracy** (using, e.g., MAP or NDCG) on the training data by adjusting these parameters