How Google works → Search engine

Query: artificial intelligence => ranked list of web pages

1) Look if query words are in the document
   Frequency of query in doc; high freq indicates high relevance
2) Web page contains query words and these are rare
   - Monte Carlo tree search

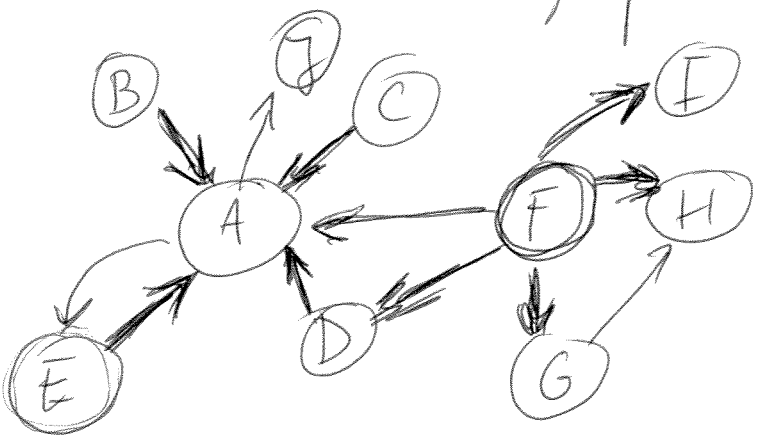
Inverse document frequency: each word → fraction of all documents on web containing it
1/... big => word is specific

$$\sum_{\text{word}} \text{term frequency} * \text{inverse document frequency}$$

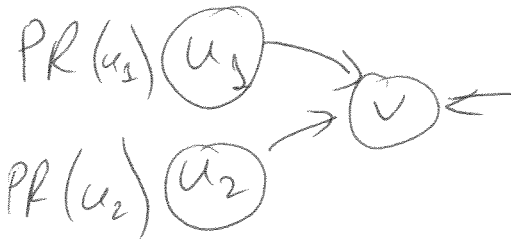
term frequency: fn of document
inverse document frequency: fn of word
Large dictionary (Hash Map)
Computed offline

Sort docs in this way
needs to be computed at query time

Page Rank: Reputation-based system
↑
computed automatically from structure of the web graph.



A has high "in-degree" (links coming in)
A is a "reference" aka high reputation
F has high "out-degree" (links going out)
F is a "repository" so low reputation
Nodes with references into them from authorities are likely to also be good quality.
Node → PageRank (number). High = good.
PR(v): page rank of v (computed for all v's)
Out(v): out-degree of v
$u_1, u_2 \ldots u_k$: nodes pointing to v



$$PR(v) = \sum \frac{PR(u_i)}{Out(u_i)}$$

Giant system of linear equations:

$$\forall v \quad PR(v) = \sum_{u_i} \frac{PR(u_i)}{Out(u_i)}$$

Dampening:

$$PR(v) = (1-d)PR(v) + d \sum_{u_i} \frac{PR(u_i)}{Out(u_i)}$$

$d \in (0, 1)$ → system has a solution

Not going to solve directly → treat each equation as an ~~set of~~ "update rule"

Randomize selection of v

at a picked v: $PR(v) \xleftarrow{\text{update}} (1-d)PR(v) + d \sum_{u_i} \frac{PR(u_i)}{Out(u_i)}$

$Out(u_i)$ needs to be known

$PR(u_i)$ → current guesses

Solving fixed-point equations

$\vec{x} = f(\vec{x})$ fixed-point of f

Special case: f is linear

Iterative algorithm: start with x_0 (guess for sol)

$$x_{i+1} \leftarrow f(x_i)$$

If f is "well-behaved", $x_i \xrightarrow{i \to \infty} x^*$ s.t. $x^* = f(x^*)$

$x_{i+1} = \underbrace{(1-d)}\, x_i + d f(x_i)$

$1-d \approx 1$ $(d \approx 0)$ moving very slowly

$x_{i+1} \approx x_i$ safe

$1-d \approx 0 \Rightarrow$ X's may diverge / oscillate dep on f

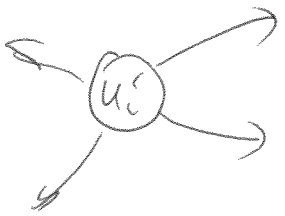
Initially: associate a value to PR(v) → update.

Out-degrees always get updated by <u>crawling</u>

PR also updated by crawling.

word → links of docs containing it (Hash Map)
also updated by crawlers

Query → words → show docs that contain the words in decreasing order of Page Rank



$\frac{PR(u_i)}{Out(u_i)}$ = traffic going out of u_i

start at u_0 w p prop to $PR(u_0)$

what is the chance of hitting some node v?