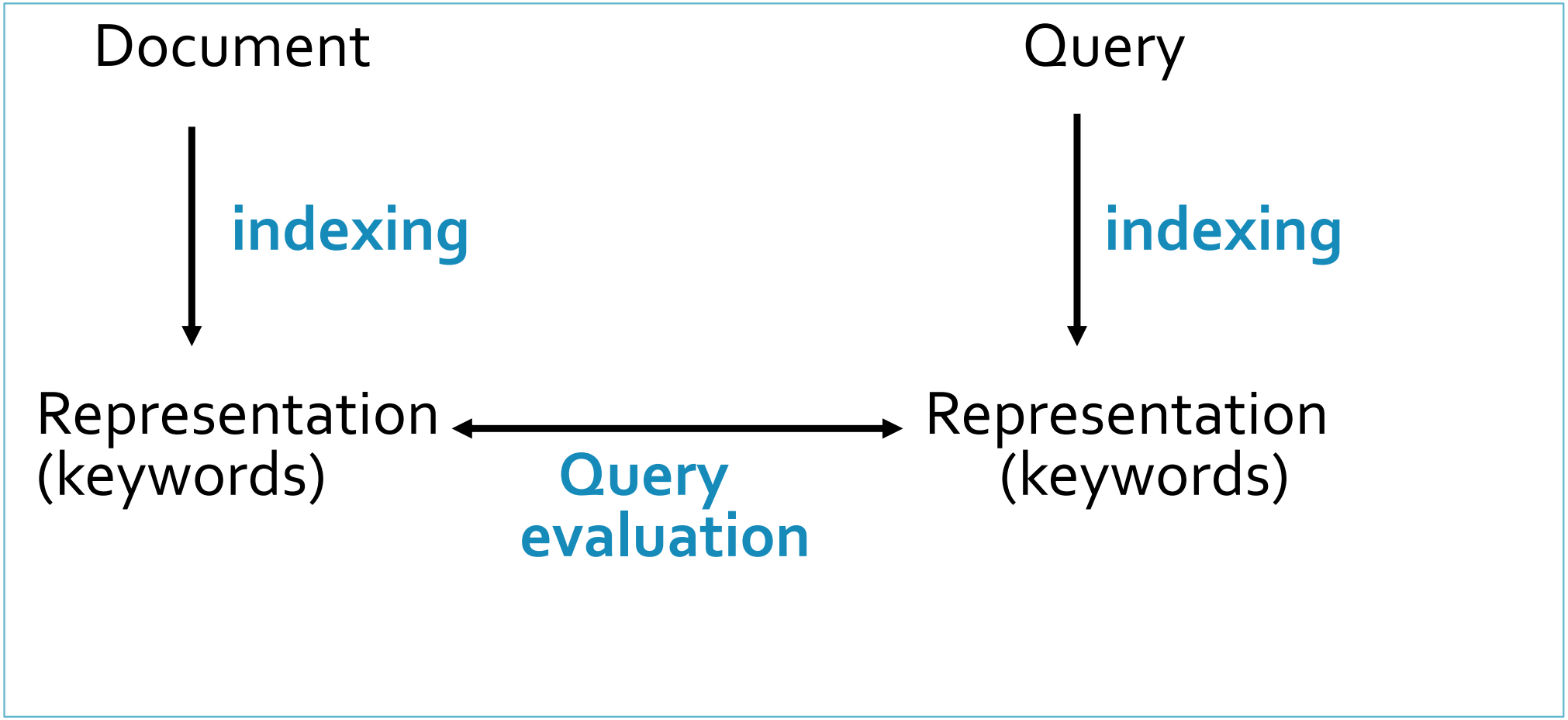
Information Retrieval

Dr Pradipta Biswas, PhD (Cantab)
Assistant Professor
Indian Institute of Science
https://cambum.net/

Indexing-based IR



Main problems in IR

- Document and query indexing
  - How to best represent their contents?
- Query evaluation (or retrieval process)
  - To what extent does a document correspond to a query?
- System evaluation
  - How good is a system?
  - Are the retrieved documents relevant? (precision)
  - Are all the relevant documents retrieved? (recall)

Document indexing

- Goal = Find the important meanings and create an internal representation
- Factors to consider:
  - Accuracy to represent meanings (semantics)
  - Exhaustiveness (cover all the contents)
  - Facility for computer to manipulate
- What is the best representation of contents?
  - Char. string (char trigrams): not precise enough
  - Word: good coverage, not precise
  - Phrase: poor coverage, more precise
  - Concept: poor coverage, precise

tf*idf weighting schema

- tf = term frequency
  - frequency of a term/keyword in a document

  The higher the tf, the higher the importance (weight) for the doc.
- df = document frequency
  - no. of documents containing the term
  - distribution of the term
- idf = inverse document frequency
  - the unevenness of term distribution in the corpus
  - the specificity of term to a document

  The more the term is distributed evenly, the less it is specific to a document

  weight(t,D) = tf(t,D) * idf(t)

Some common *tf*idf* schemes

- tf(t, D)=freq(t,D)
- tf(t, D)=log[freq(t,D)]
- tf(t, D)=log[freq(t,D)]+1
- tf(t, D)=freq(t,d)/Max[f(t,d)]

idf(t) = log(N/n)

n = #docs containing t

N = #docs in corpus

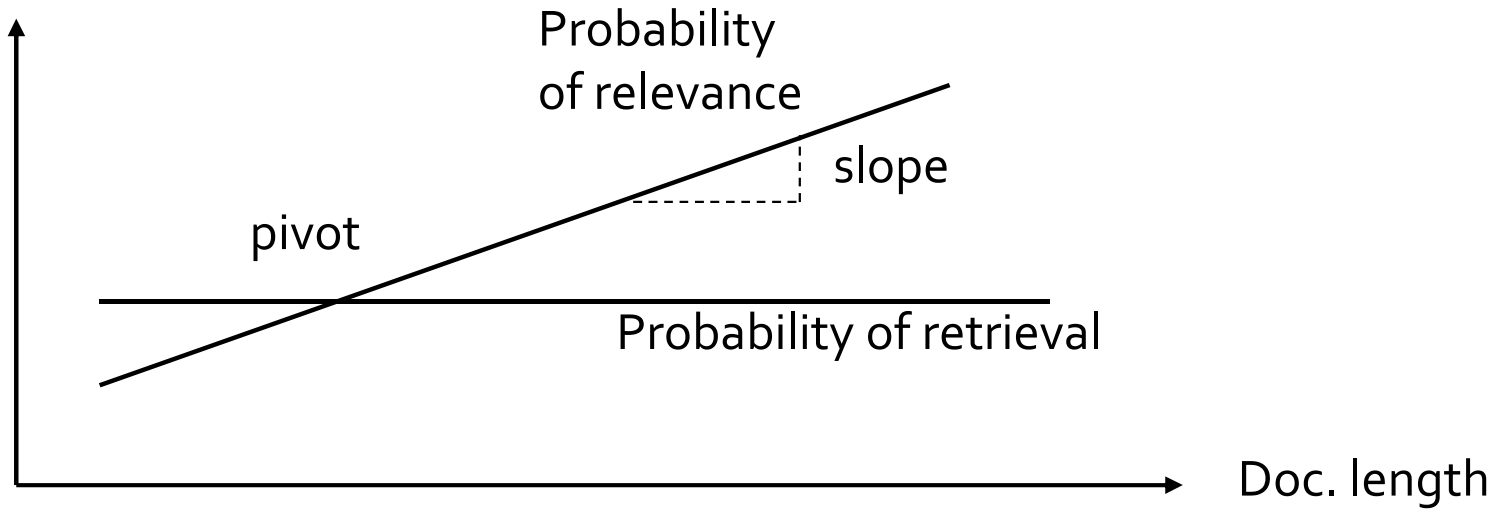
weight(t,D) = tf(t,D) * idf(t)

- Normalization: Cosine normalization, /max, …

Document Length Normalization

- Sometimes, additional normalizations e.g. length:

$$pivoted(t,D) = \frac{weight(t,D)}{1 + \frac{slope}{(1-slope)\times povot} normalized\_weight(t,D)}$$

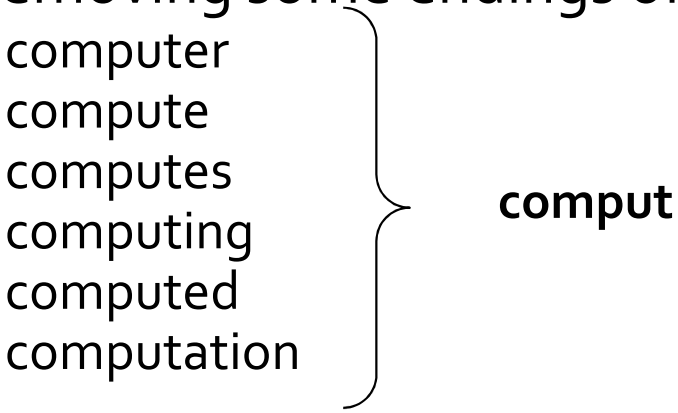


Stopwords / Stoplist

- function words do not bear useful information for IR
  of, in, about, with, I, although, …
- Stoplist: contain stopwords, not to be used as index
  - Prepositions
  - Articles
  - Pronouns
  - Some adverbs and adjectives
  - Some frequent words (e.g. document)
- The removal of stopwords usually improves IR effectiveness
- A few "standard" stoplists are commonly used.

Stemming

- Reason:
  - Different word forms may bear similar meaning (e.g. search, searching): create a "standard" representation for them
- Stemming:
  - Removing some endings of word



Porter algorithm

(Porter, M.F., 1980, An algorithm for suffix stripping, *Program*, 14(3) :130-137)

- Step 1: plurals and past participles
  - SSES -> SS caresses -> caress
  - (*v*) ING -> motoring -> motor
- Step 2: adj->n, n->v, n->adj, …
  - (m>0) OUSNESS -> OUS callousness -> callous
  - (m>0) ATIONAL -> ATE relational -> relate
- Step 3:
  - (m>0) ICATE -> IC triplicate -> triplic
- Step 4:
  - (m>1) AL -> revival -> reviv
  - (m>1) ANCE -> allowance -> allow
- Step 5:
  - (m>1) E -> probate -> probat
  - (m > 1 and *d and *L) -> single letter controll -> control

Lemmatization

- transform to standard form according to syntactic category.
  E.g. verb + *ing* → verb
  noun + *s* → noun
  - Need POS tagging
  - More accurate than stemming, but needs more resources
- crucial to choose stemming/lemmatization rules
  noise v.s. recognition rate
- compromise between precision and recall

light/no stemming
-recall +precision

severe stemming
+recall -precision

Result of indexing

- Each document is represented by a set of weighted keywords (terms):

  $D_1 \rightarrow \{(t_1, w_1), (t_2, w_2), \ldots\}$

  e.g. $D_1 \rightarrow \{(comput, 0.2), (architect, 0.3), \ldots\}$

  $D_2 \rightarrow \{(comput, 0.1), (network, 0.5), \ldots\}$

- Inverted file:

  $comput \rightarrow \{(D_1, 0.2), (D_2, 0.1), \ldots\}$

  Inverted file is used during retrieval for higher efficiency.

Retrieval

- The problems underlying retrieval
  - Retrieval model
    - How is a document represented with the selected keywords?
    - How are document and query representations compared to calculate a score?
  - Implementation

Vector space model

- Vector space = all the keywords encountered

  $<t_1, t_2, t_3, \ldots, t_n>$

- Document

  $D = < a_1, a_2, a_3, \ldots, a_n>$

  a_i = weight of t_i in D

- Query

  $Q = < b_1, b_2, b_3, \ldots, b_n>$

  b_i = weight of t_i in Q

- R(D,Q) = Sim(D,Q)

Matrix representation

Document space

	t_1	t_2	t_3	…	t_n
D_1	a_{11}	a_{12}	a_{13}	…	a_{1n}
D_2	a_{21}	a_{22}	a_{23}	…	a_{2n}
D_3	a_{31}	a_{32}	a_{33}	…	a_{3n}
…					
D_m	a_{m1}	a_{m2}	a_{m3}	…	a_{mn}
Q	b_1	b_2	b_3	…	b_n

← Term vector space

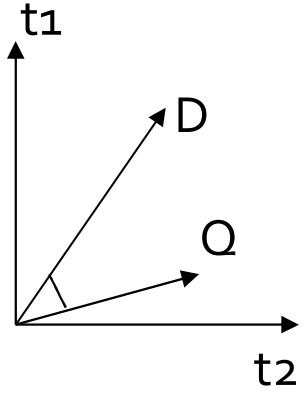
Some formulas for Sim

Dot product

$$Sim(D,Q) = \sum (a_i * b_i)$$

Cosine

$$Sim(D,Q) = \frac{\sum_i (a_i * b_i)}{\sqrt{\sum_i a_i^2 * \sum_i b_i^2}}$$



Dice

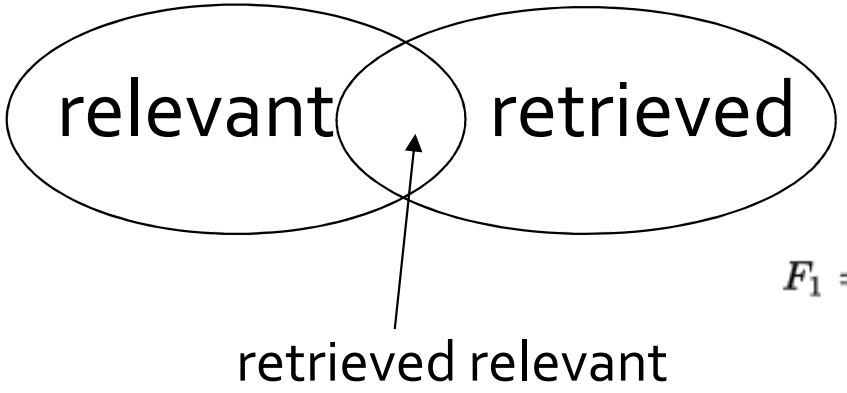
$$Sim(D,Q) = \frac{2\sum_i (a_i * b_i)}{\sum_i a_i^2 + \sum_i b_i^2}$$

Jaccard

$$Sim(D,Q) = \frac{\sum_i (a_i * b_i)}{\sum_i a_i^2 + \sum_i b_i^2 - \sum_i (a_i * b_i)}$$

System evaluation

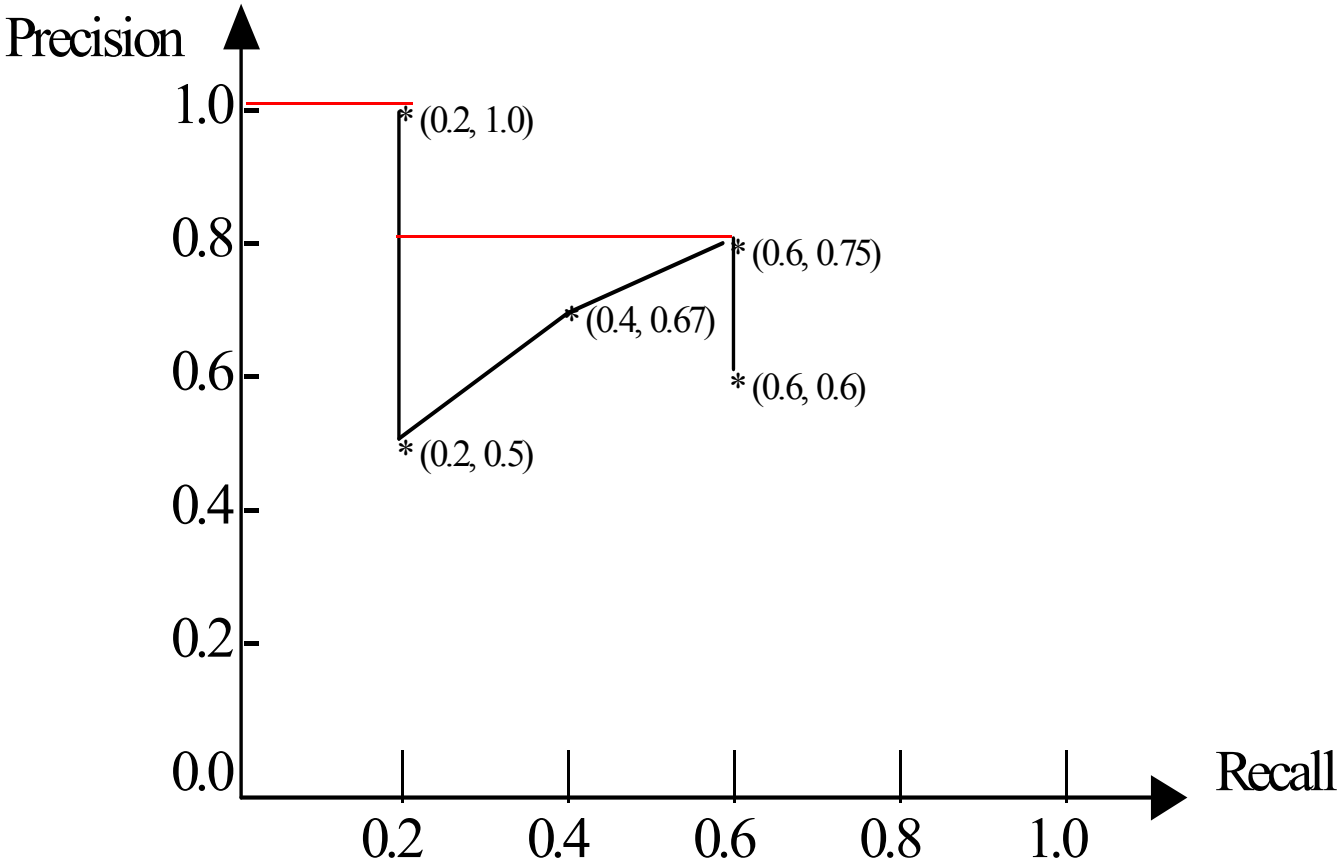
- Efficiency: time, space
- Effectiveness:
  - How is a system capable of retrieving relevant documents?
  - Is a system better than another one?
- Metrics often used (together):
  - Precision = retrieved relevant docs / retrieved docs
  - Recall = retrieved relevant docs / relevant docs



$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

An illustration of P/R calculation

List	Rel?
Doc1	Y
Doc2	
Doc3	Y
Doc4	Y
Doc5	
…	



Assume: 5 relevant docs.

Goal of IR

- Assumptions:
  - The goal is maximizing precision and recall simultaneously
  - The information need remains static
  - The value is in the resulting document set

Link Analysis

Importance

- Internet Surfers generally do not bother to go through the first 10 to 20 pages
- So the ordering of pages is important to compose an effective and efficient search result.

Problems

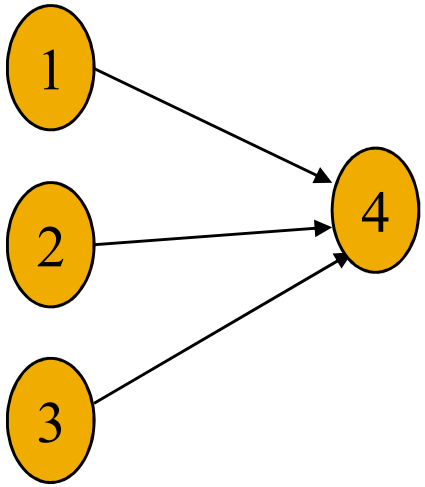
- Huge size of Web
- Exponential increase in size
- Unstructured nature of web pages
- No control on content

Hypertext Induced Topic Search

Ref: Kleinberg, Jon; "Authoritative Sources in a Hyperlinked Environment;" Proc. ACM-SIAM Symposium on Discrete Algorithms, 1998; pp. 668-677

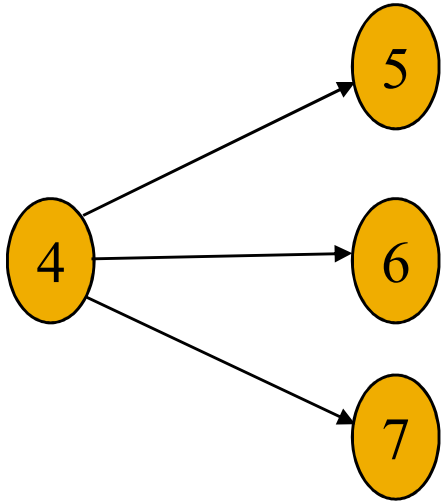
Hubs and Authorities

Authorities are pages that are recognized as providing significant, trustworthy, and useful information on a topic.



$$a_4 = h_1 + h_2 + h_3$$

$$h_4 = a_5 + a_6 + a_7$$



Hubs are index pages that provide lots of useful links to relevant content pages (topic authorities).

HITS Iterative Algorithm

Initialize for all $p \in S$: $a_p = h_p = 1$

For i = 1 to k:

For all $p \in S$:

$$a_p = \sum_{q:q \to p} h_q$$ (*update auth. scores*)

For all $p \in S$:

$$h_p = \sum_{q:p \to q} a_q$$ (*update hub scores*)

For all $p \in S$: $a_p = a_p / c$ c:

$$\sum_{p \in S} (a_p / c)^2 = 1$$ (*normalize* $\boldsymbol{a}$)

For all $p \in S$: $h_p = h_p / c$ c:

$$\sum_{p \in S} (h_p / c)^2 = 1$$ (*normalize* $\mathbf{h}$)

Convergence

- Algorithm converges to a *fix-point* if iterated indefinitely.
- Define A to be the adjacency matrix for the subgraph defined by S.
  - $A_{ij} = 1$ for $i \in S, j \in S$ iff $i \rightarrow j$
- Authority vector, $\boldsymbol{a}$, converges to the principal eigenvector of $A^T A$
- Hub vector, $\boldsymbol{h}$, converges to the principal eigenvector of AA^T
- In practice, 20 iterations produces fairly stable results.

Drawbacks

- Pure link based computation-textual content is ignored
- Topic Drifting-Appear when hub discusses multiple topic

Page Rank

Ref: Brin, Sergey; Page, Lawrence; "The Anatomy of a Large-Scale Hypertextual Web Search Engine;" 7th Int. WWW Conf. Proceedings, Brisbane, Australia; April 1998.

PageRank

- Alternative link-analysis method used by Google (Brin & Page, 1998).
- Does not attempt to capture the distinction between hubs and authorities.
- Ranks pages just by authority.
- Applied to the entire web rather than a local neighborhood of pages surrounding the results of a query.

Random Surfer Model

- PageRank can be seen as modeling a "random surfer" that starts on a random page and then at each point:
  - With probability $E(p)$ randomly jumps to page p.
  - Otherwise, randomly follows a link on the current page.
- $R(p)$ models the probability that this random surfer will be on page p at any given time.
- "Jumps" are needed to prevent the random surfer from getting "trapped" in web sinks with no outgoing links.

Page Rank Algorithm

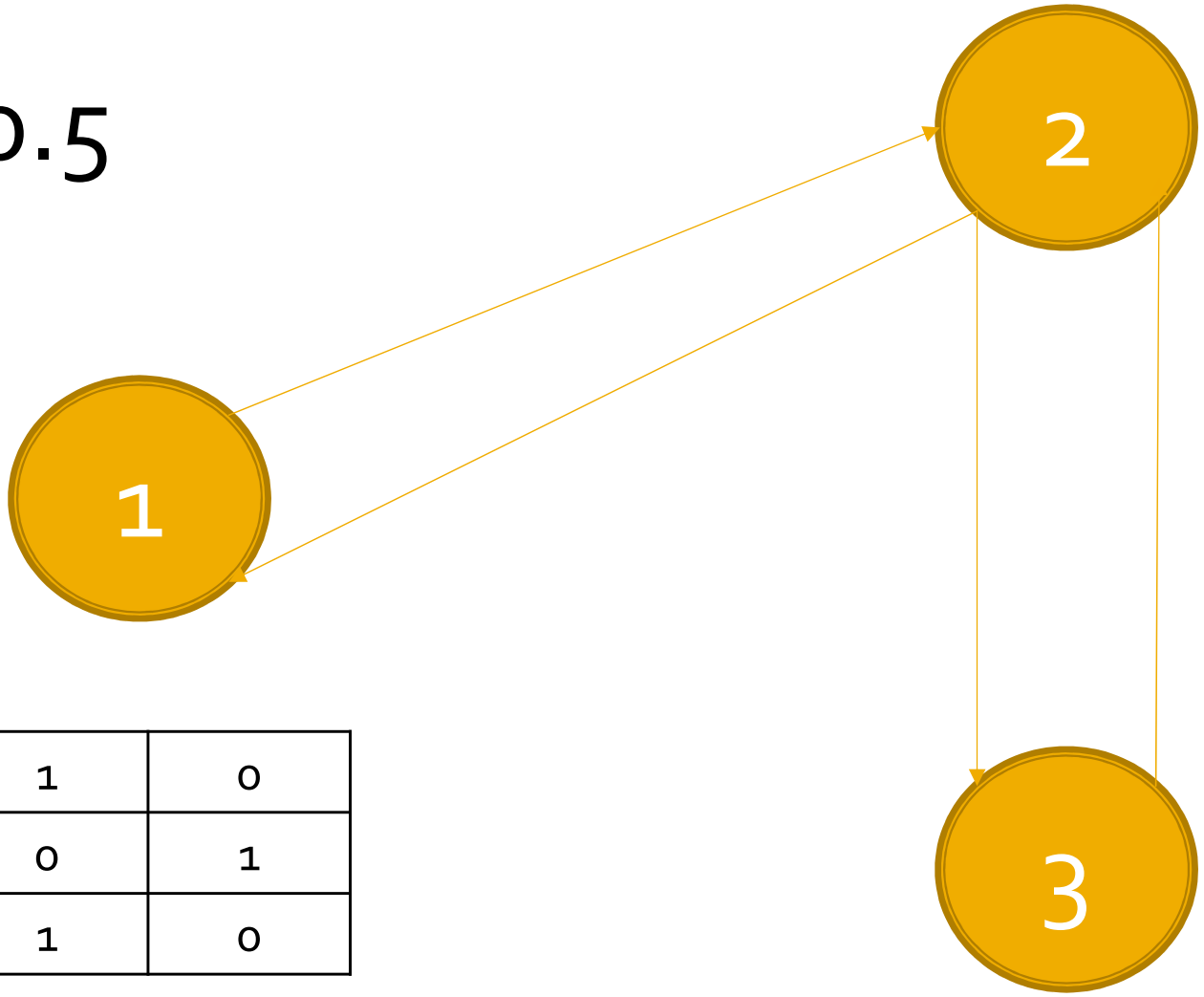
- When at a node with no out-links, the surfer invokes the teleport (jump) operation
- At any node that has outgoing links, the surfer invokes the teleport (jump) operation with probability $0<\alpha<1$
- Otherwise, users undertake the standard random walk means, follow an out-link chosen uniformly at random with $(1-\alpha)$ probability

Page Rank Algorithm

- If a row of adjacency matrix has no 1's, then replace each element by 1/N. For all other rows proceed as follows.
- Divide each 1 in adjacency matrix by the number of 1's in its row.
- Multiply the resulting matrix by (1- α).
- Add α/N to every entry of the resulting matrix.

Example – Step 1

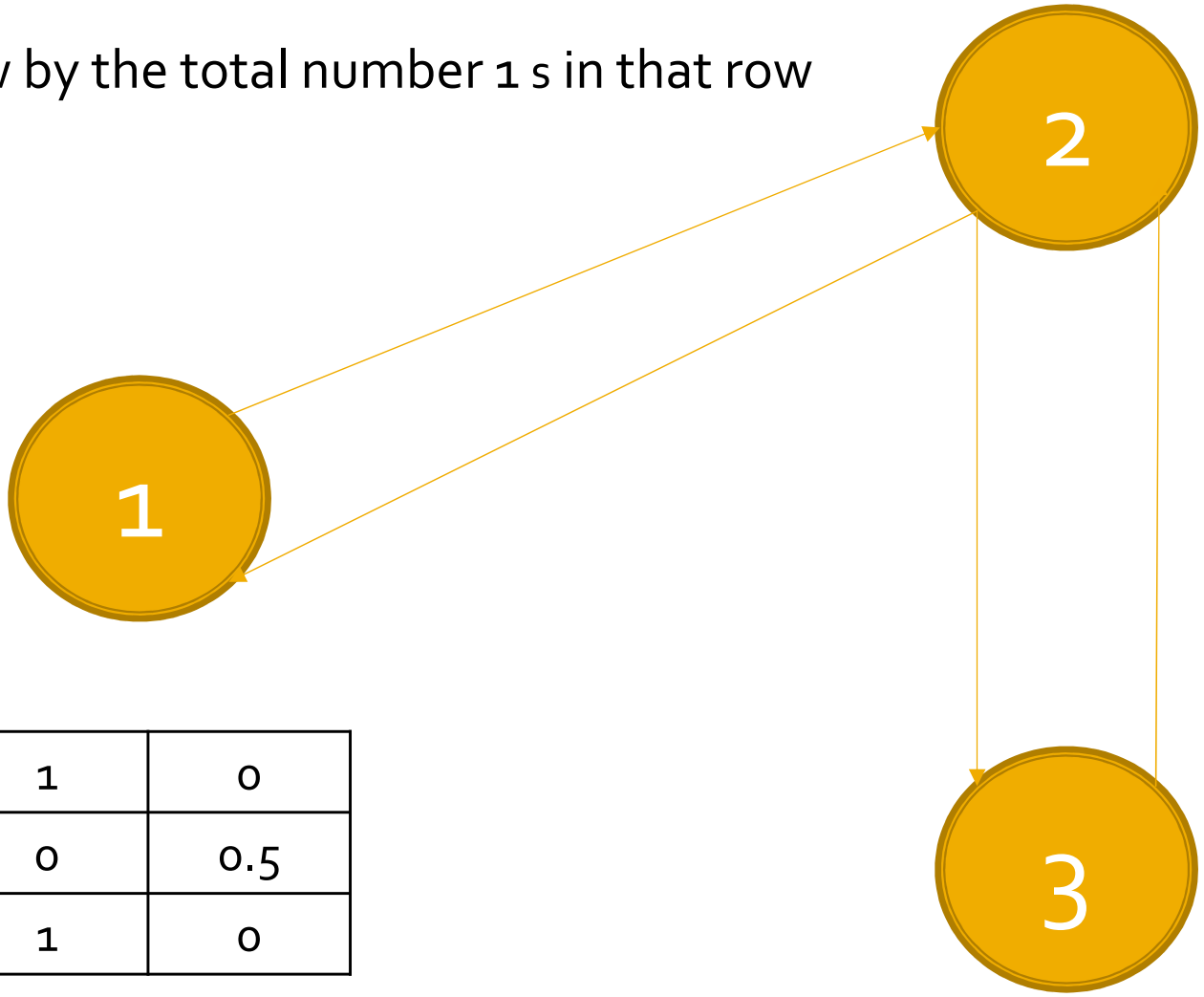
$\alpha = 0.5$



0	1	0
1	0	1
0	1	0

Example – Step 2

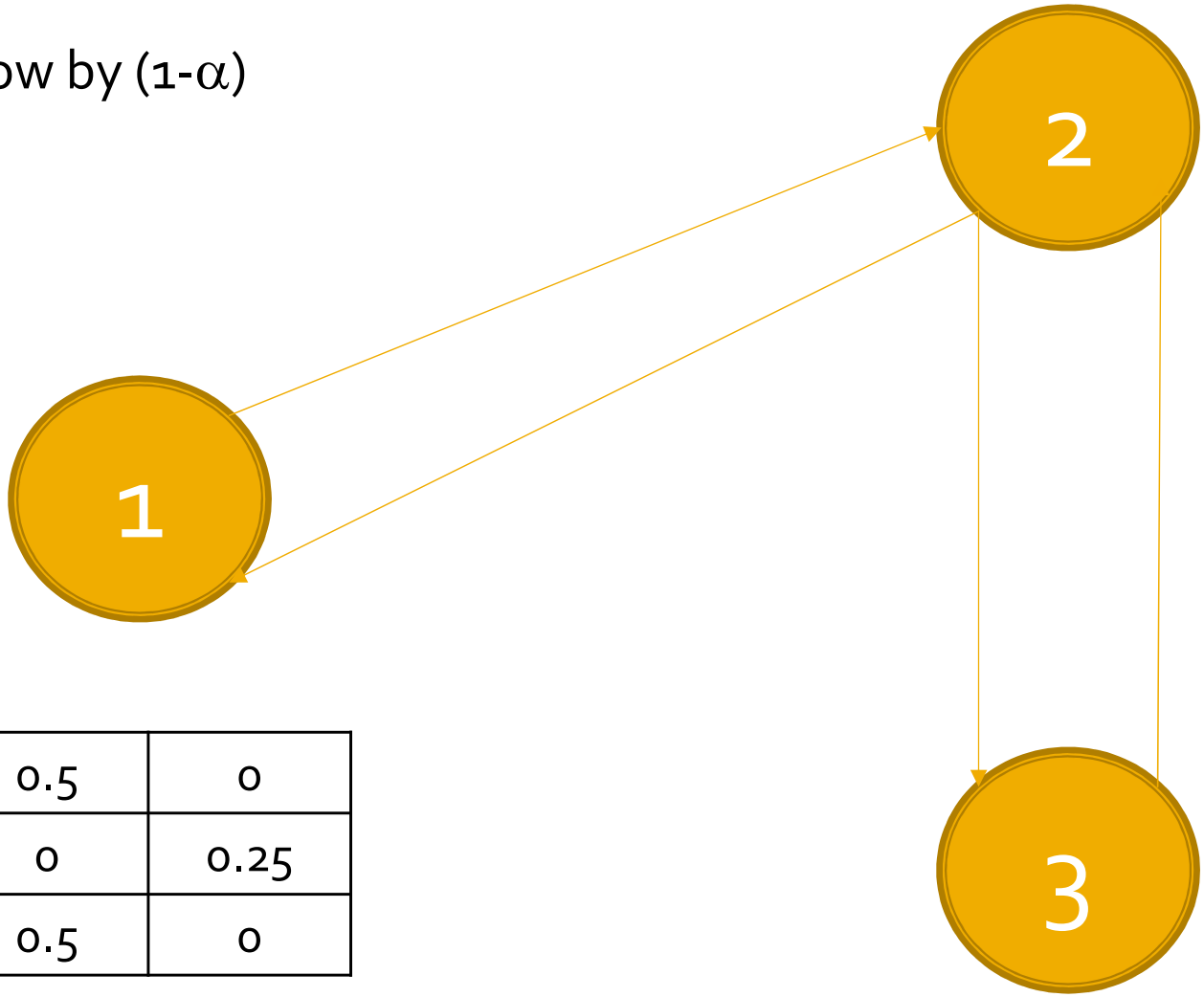
Divide each row by the total number 1 s in that row



0	1	0
0.5	0	0.5
0	1	0

Example – Step 3

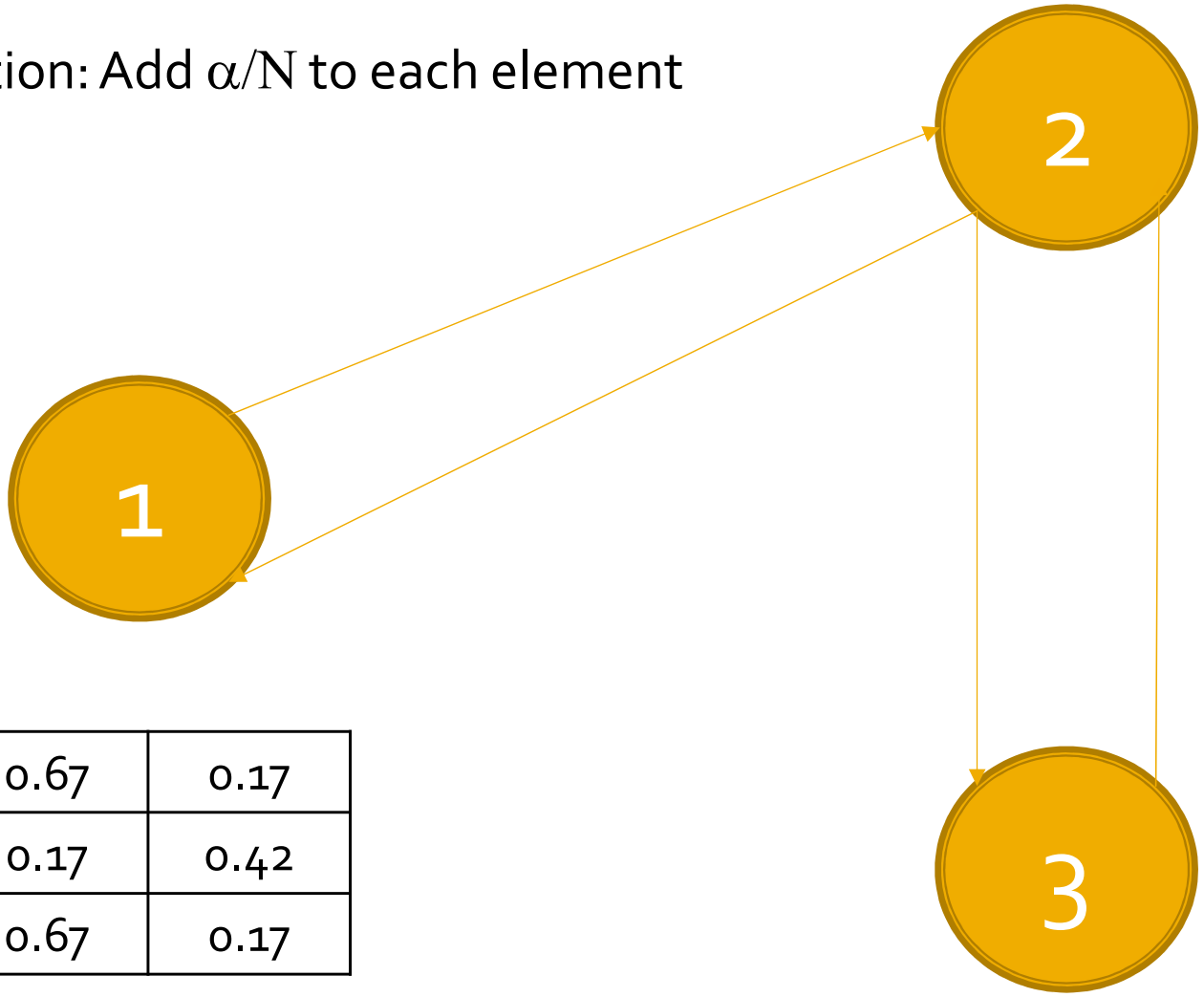
Multiply each row by (1-α)



0	0.5	0
0.25	0	0.25
0	0.5	0

Example – Step 4

Teleport operation: Add α/N to each element



0.17	0.67	0.17
0.42	0.17	0.42
0.17	0.67	0.17

Whats next

- Surfer starts at any page and browse through the links with teleportation
- After a large number/iteration of surfing the probability of visiting each page 'settles down'
- It even turns independent of initial state
- It is the principal left eigen vector of the initial probability matrix (previous slide)

Speed of Convergence

- Early experiments on Google used 322 million links.
- PageRank algorithm converged (within small tolerance) in about 52 iterations.
- Number of iterations required for convergence is empirically O(log *n*) (where *n* is the number of links).
- Therefore calculation is quite efficient.

Comparison of HITS and PageRank

- HITS
  - Assembles different root set and prioritizes pages in the context of query
  - Looks forward and backward direction
- Page Rank
  - Assigns initial ranking and retains them independently from queries (fast)
  - In the forward direction from link to link

Take Away Points

- Basic Process of Information Retrieval
  - Data Preprocesing
  - Tf-Idf
  - Vector Space Model
  - Precision & Recall
- Link Analysis
  - HITS and Page Rank
  - Calculating page rank from adjacency matrix