Jul – Dec 2022
IT458

Information Retrieval on the Web

Link Analysis and Web Search

Link Analysis Algorithms

- based on analysis of links for computing importance of nodes in a Web graph :
  - Voting based algorithms
  - Link importance based algorithms
  - Learning based algorithms

Link Analysis Algorithms

- Voting based algorithms
  - Hubs and Authorities -- Hypertext Induced Topical Search (HITS)
- Link importance based algorithms
  - PageRank
  - Topic-Specific (Personalized) PageRank
- Learning based algorithms
  - Learning to Rank (LtR)
  - Learning the Ranking function (LtRf)

Voting based Ranking Algorithms

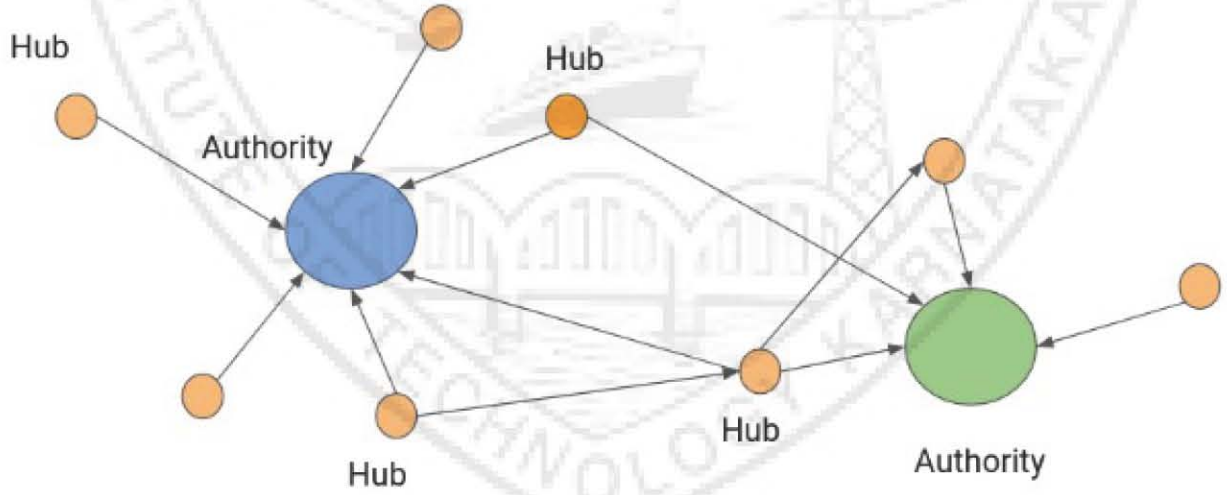
Hubs and Authorities

- Goal :
  - To find the "experts" – pages that link in a coordinated way to "good" content.

- Idea: Links as votes
  - Page is more important if it has more links !!

Hubs and Authorities

- Interesting pages fall into two classes:
  1. **Authorities**: pages containing useful information
     - E.g Newspaper home pages, Course home pages .....
  2. **Hubs**: pages that link to authorities
     - E.g. List of newspapers, University course listings…



HITS Algorithm

- Hypertext-Induced Topical Search
  - Core of early search engines like Altavista, Ask.com etc

- **The Principle of Repeated Improvement**
  - A good hub links to many good authorities
  - A good authority is linked from many good hubs

HITS Algorithm

- Model uses two scores for each node: **Hub score and Authority score**
  - Represented as vectors ***hub(p)*** and ***auth(p)***, where the p^{th} element is the hub/authority score of the p^{th} node
  - Both initially set to 1.

HITS Algorithm

- Voting phases -
  - ***Authority Update Rule:*** For each page p, **update *auth(p)*** to be the **sum of the hub scores** of all pages that point to it.
  - ***Hub Update Rule:*** For each page p, **update *hub(p)*** to be the **sum of the authority scores** of all pages that it points to.

HITS Algorithm - Steps

- Start: hub and authority scores = 1

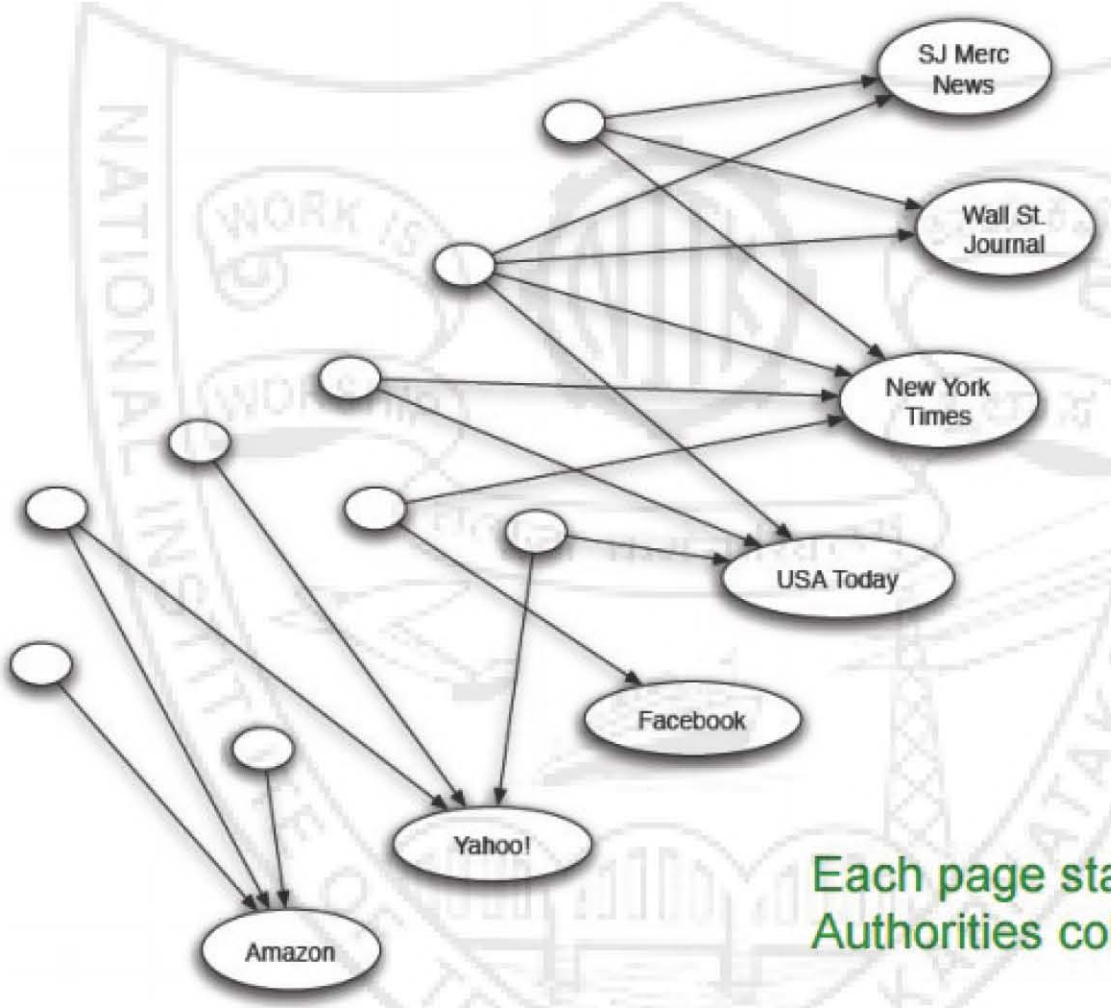
  choose the number of iterative steps k.

- Perform a sequence of k hub-authority updates.
  1. Apply the Authority Update Rule to the current set of scores.
  2. Then, apply the Hub Update Rule to the resulting set of scores.

- After k iterations, normalize all resulting scores.
  1. Divide each authority score by the sum of all authority scores.
  2. Divide each hub score by the sum of all hub scores.

- Repeat from the second step as necessary.

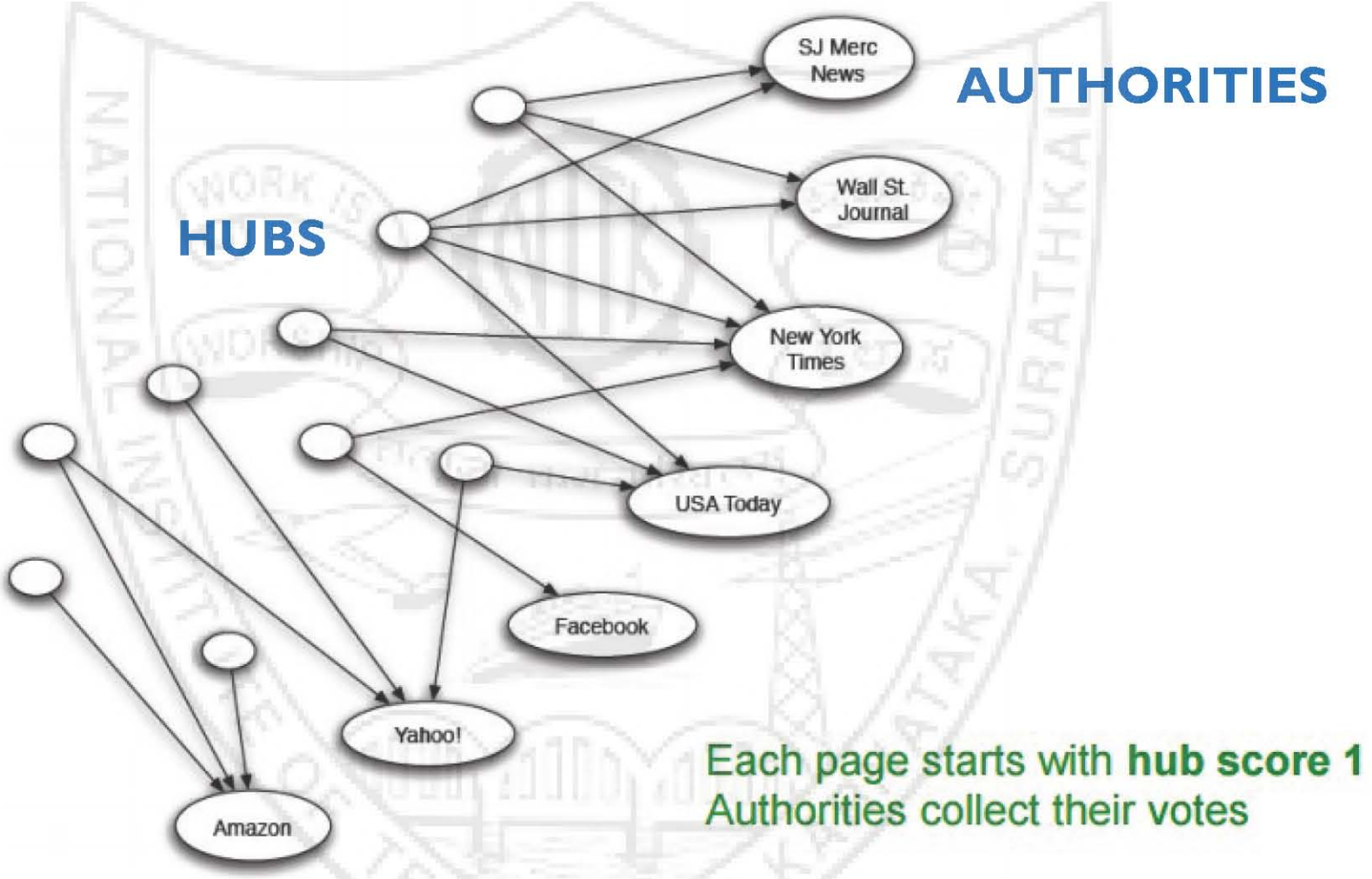
Counting in-links: Authority



Each page starts with **hub score 1**
Authorities collect their votes

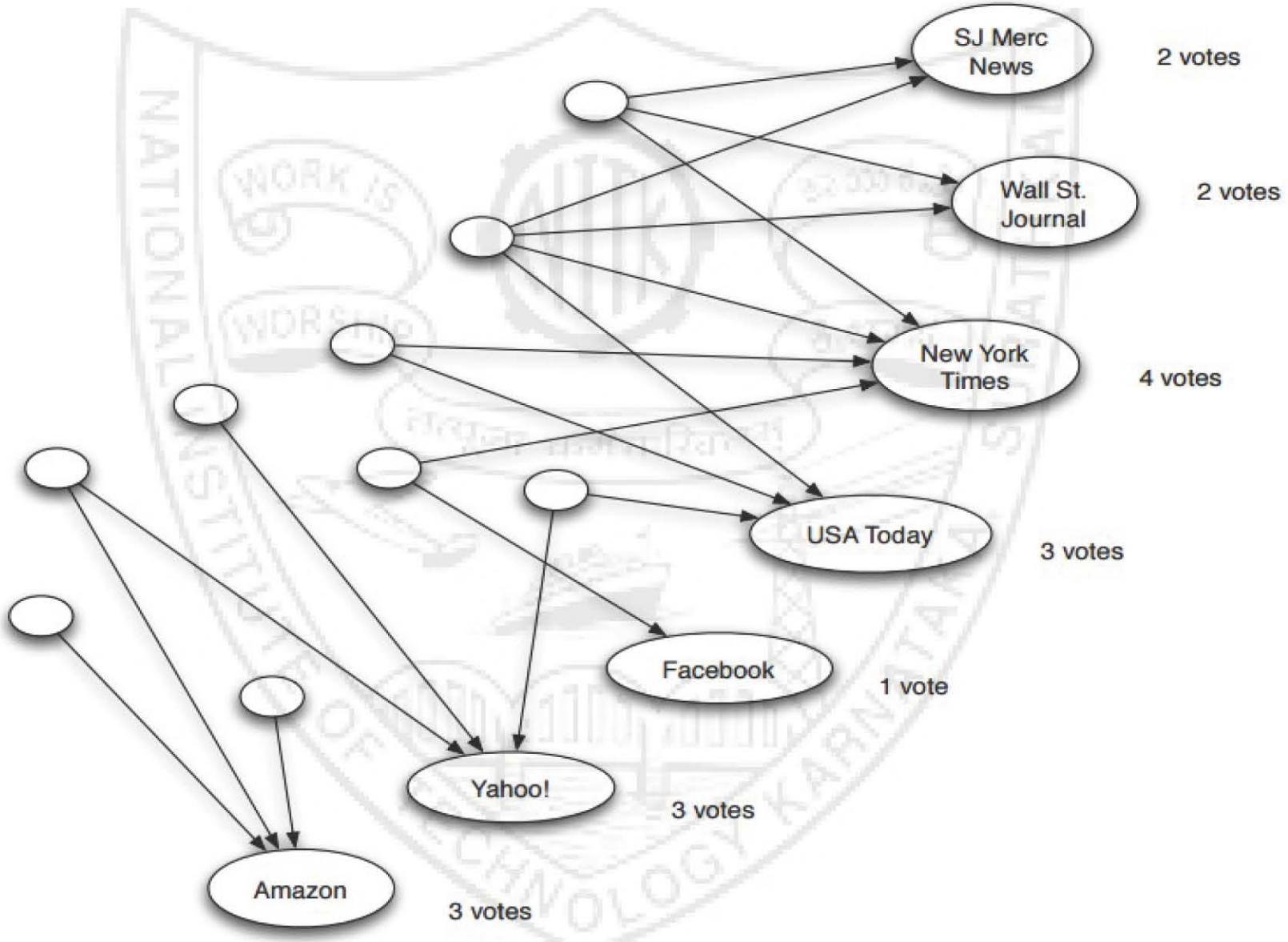
(Note this is idealized example. In reality graph is not bipartite and each page has both the hub and authority score)

Counting in-links: Authority

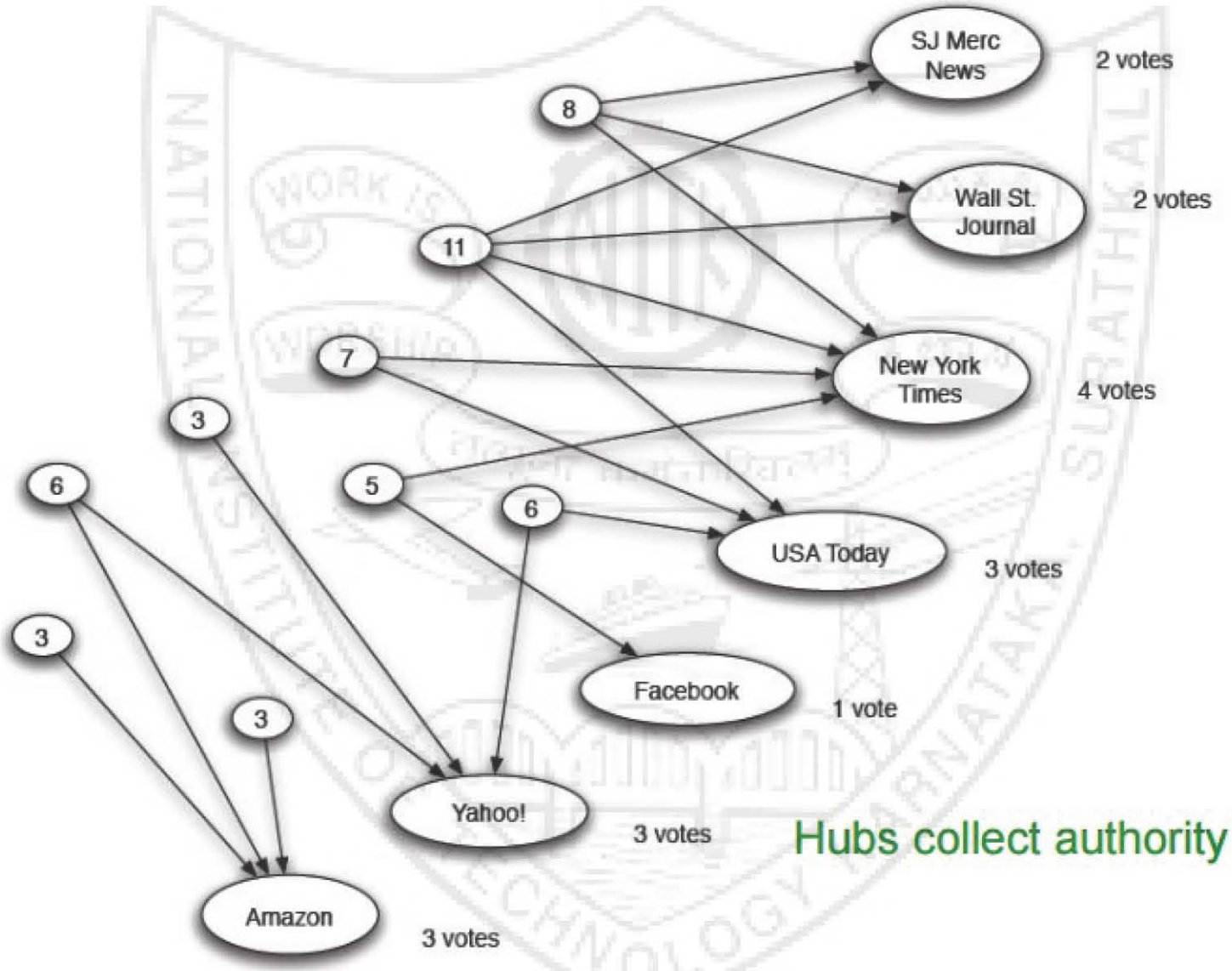


(Note this is idealized example. In reality graph is not bipartite and each page has both the hub and authority score)

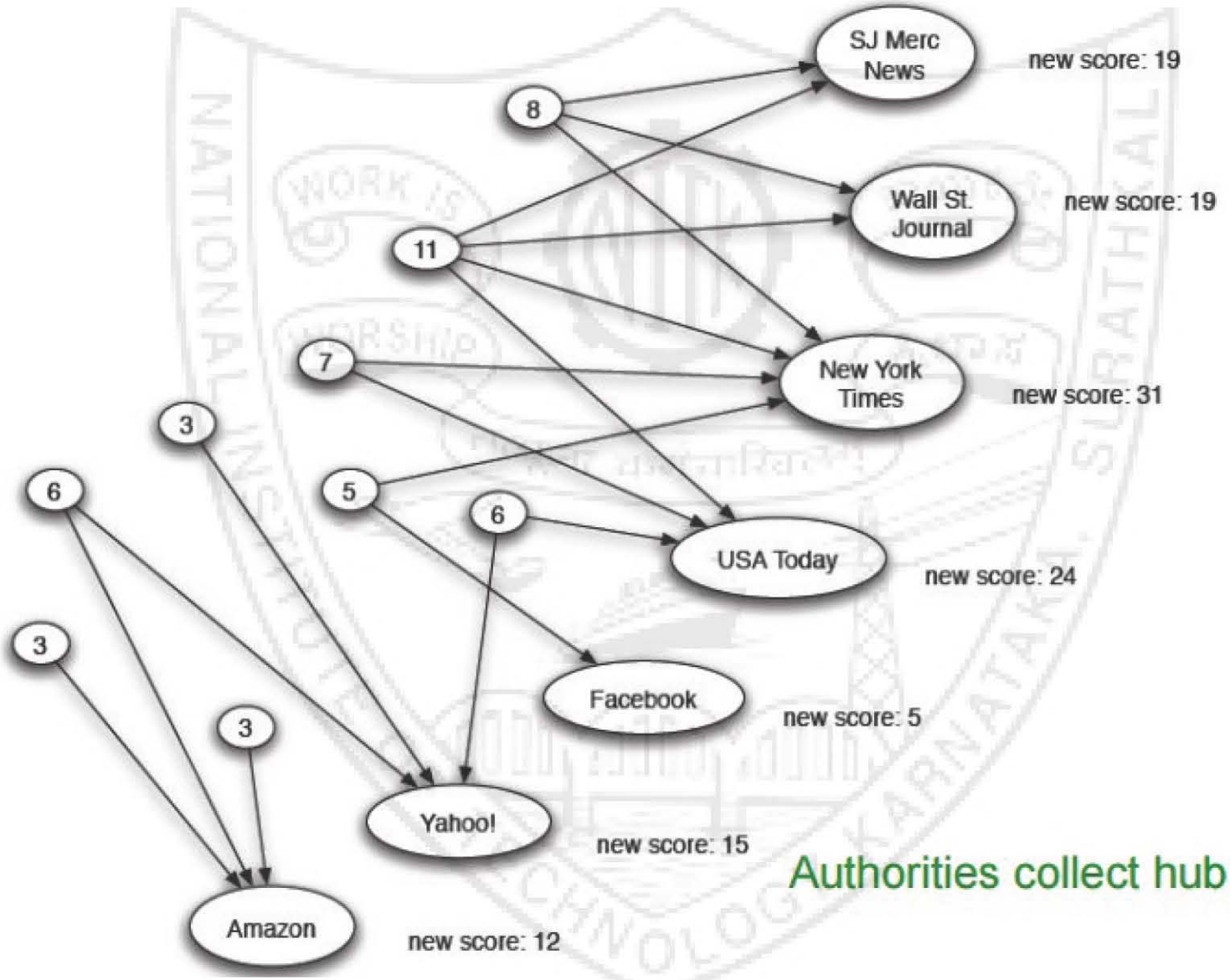
Votes by other hubs: Authority



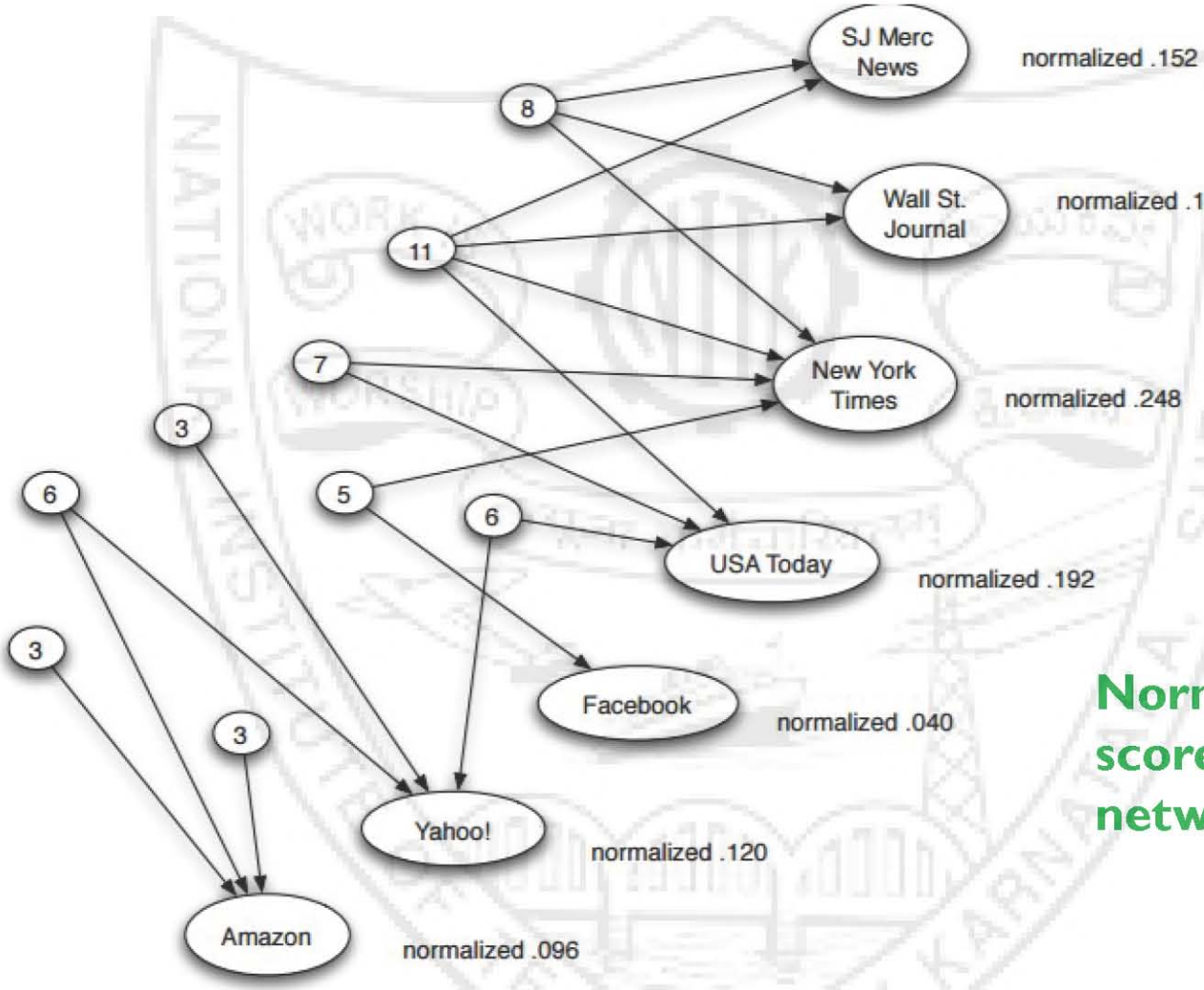
Expert Quality: Hub



Reweighting



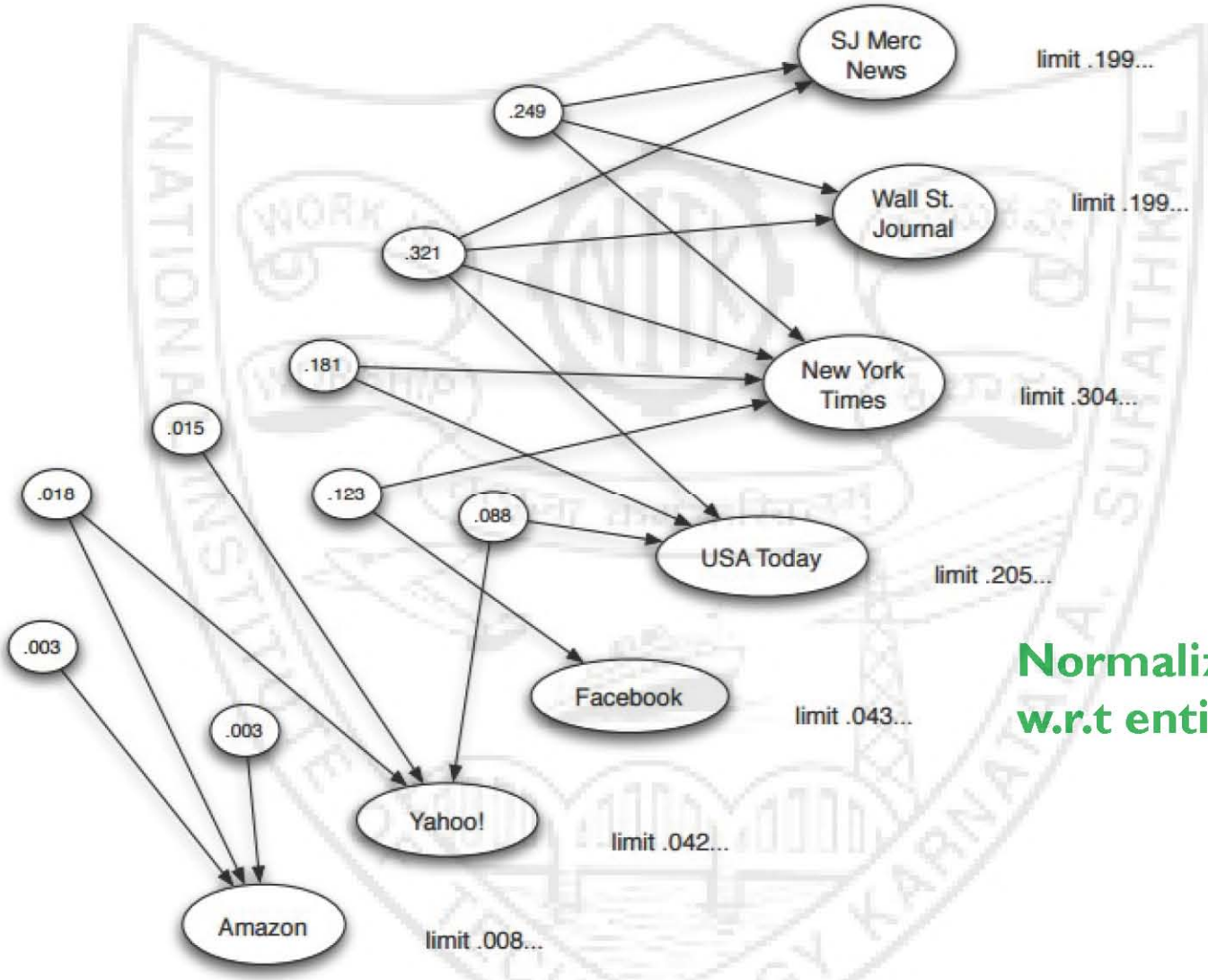
Normalizing Authority score



Normalizing authority scores w.r.t entire network

Divide each authority score by the sum of all authority scores

Normalizing Hub score



Normalizing hub scores w.r.t entire network

Divide each hub score by the sum of all hub scores

HITS - Summary

- Merits:
  - Simple way of calculating relative importance of connected pages
- Demerits:
  - it is executed at query time, not at indexing time
    - High latency due to query-time processing.
  - processed on a small subset of 'relevant' documents, not all documents
    - PageRank addresses this issue.

Link Importance based Ranking Algorithms

Dr. Sowmya Kamath S, Dept of IT, NITK Surathkal

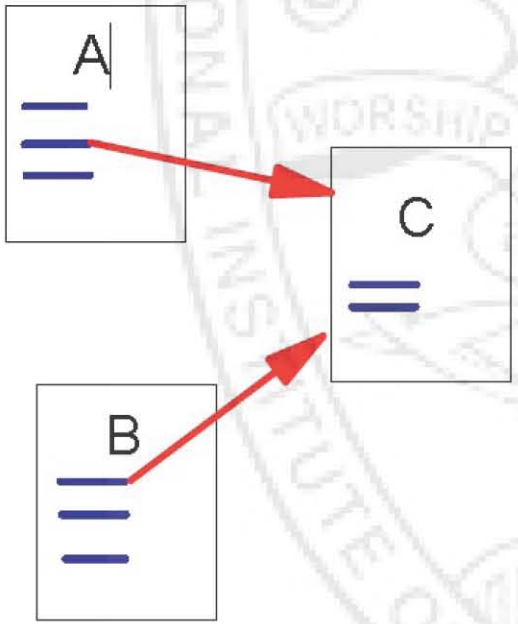
PageRank

- PageRank
  - rates the importance of **web pages** objectively and continuously using the **link structure** of the web.

- developed by Larry Page (hence the name *Page*-Rank) and Sergey Brin.
  - used by the Google Internet search engine.
  - Patent issued to Stanford University!

PageRank Algorithm

- Exploits the link structure of the Web.
- *Stats*: More than 1.98 billion websites on the Internet → Trillions of links.

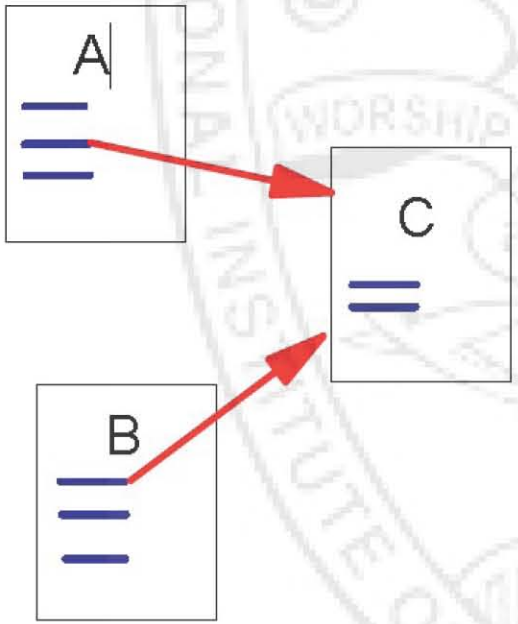


Back (incoming) links and *Forward (outgoing)* links:

- A and B are C's incoming links
- C is A and B's outgoing link

PageRank Algorithm

- Exploits the link structure of the Web.
- *Stats*: More than 1.98 billion websites on the Internet → Trillions of links.



Back (incoming) links and *Forward (outgoing)* links:

- A and B are C's incoming links
- C is A and B's outgoing link

* Intuitively, a webpage is important if it has a lot of incoming links, that keep changing over time.

PageRank Algorithm (contd.)

- Salient features –
  - interprets a link from page A to page B as a vote by page A for page B.
  - All votes don't weigh the same.
  - does not rank the whole website.
  - PR of page A is a recursive function defined by the PR of all those pages which link to page A.

PageRank's Random Surfer Model

PageRank can be thought of as a model of user behavior.

- *"We assume there is a "random surfer" who when given a web page at random, keeps clicking on links, never hitting "back" but eventually gets bored and starts on another random page. The probability that the random surfer visits a page is its* ***PageRank****."*

- *"And, the* ***damping factor d*** *(teleportation probability) is the probability the "random surfer" will get bored and request another random page."*

Brin, Sergey, and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine." *Computer networks and ISDN systems* 30.1-7 (1998): 107-117.

The original PageRank™ algorithm

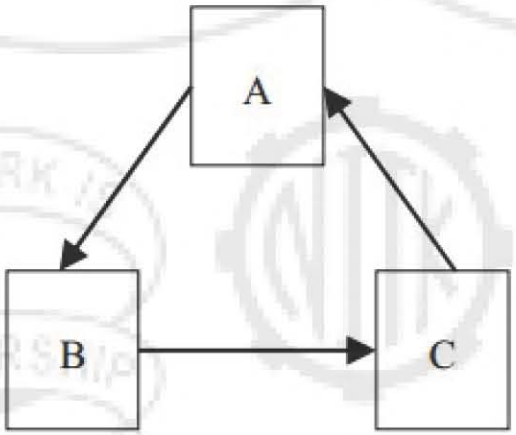
The PageRank (PR) of a page *u* is given as:

$$PR(u) = (1-d) \times (1/N) + d \times \sum (PR(v)/C(v))$$

- Assume page *u* has set of pages *v* which point to it (i.e., are votes).
  - PR(u) - PageRank of page *u*,
  - PR(v) - PageRank of the set of pages *v* that link to page *u*,
  - C(*v*) - number of links going out of pages *v*.
  - *N* - number of pages in the network.
  - *d* - damping factor which can be set between 0 and 1.
    - d is set to 0.7 - 0.85, if not otherwise specified.

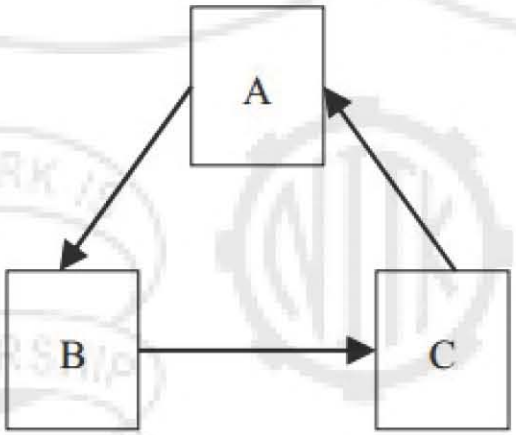
Calculating a Webpage's PageRank (PR)
Examples

Consider an example:



Calculating a Webpage's PageRank (PR)
Examples

Consider an example:

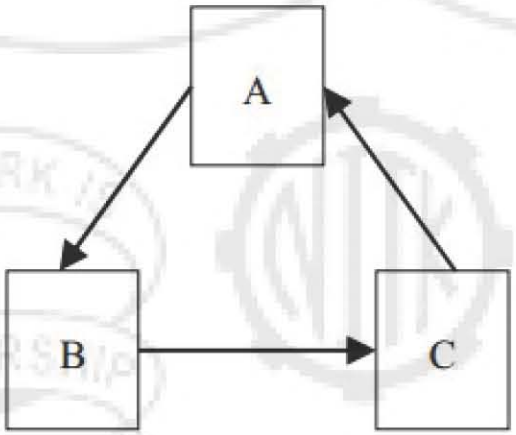


The number of web pages N = 3 ;

Let the damping parameter be d = 0.7

Calculating a Webpage's PageRank (PR)
Examples

Consider an example:



The number of web pages N = 3 ;

Let the damping parameter be d = 0.7

PageRank PR(u) = (1 - d) × (1/N) + d × ∑ (PR(v)/C(v))

Where u can be A, B or C.

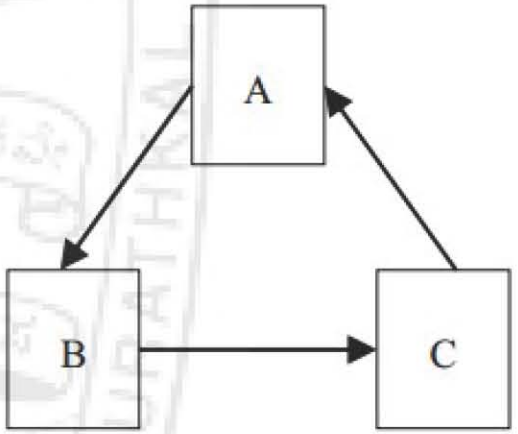
Calculating a Webpage's PageRank (PR)

So,

$$PR(A) = (1 - d) \times (1/N) + d \times (PR(C) / 1)$$

$$PR(B) = (1 - d) \times (1/N) + d \times (PR(A) / 1)$$

$$PR(C) = (1 - d) \times (1/N) + d \times (PR(B) / 1)$$



Substituting value of d and N ,

$$PR(A) = 0.1 + 0.7 \times PR(C) \quad \text{---- (i)}$$

$$PR(B) = 0.1 + 0.7 \times PR(A) \quad \text{---- (ii)}$$

$$PR(C) = 0.1 + 0.7 \times PR(B) \quad \text{---- (iii)}$$

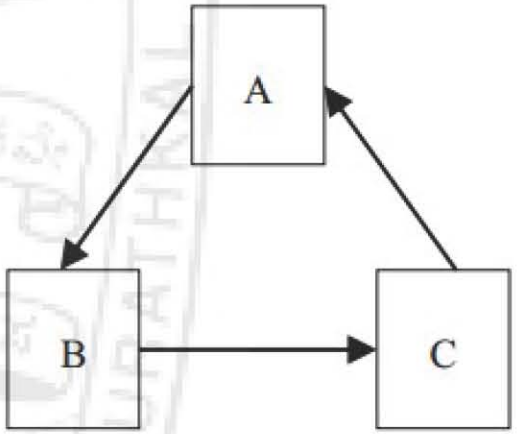
Calculating a Webpage's PageRank (PR)

So,

$$PR(A) = (1 - d) \times (1/N) + d \times (PR(C) / 1)$$

$$PR(B) = (1 - d) \times (1/N) + d \times (PR(A) / 1)$$

$$PR(C) = (1 - d) \times (1/N) + d \times (PR(B) / 1)$$



Substituting value of d and N ,

$$PR(A) = 0.1 + 0.7 \times PR(C) \quad \text{---- (i)}$$

$$PR(B) = 0.1 + 0.7 \times PR(A) \quad \text{---- (ii)}$$

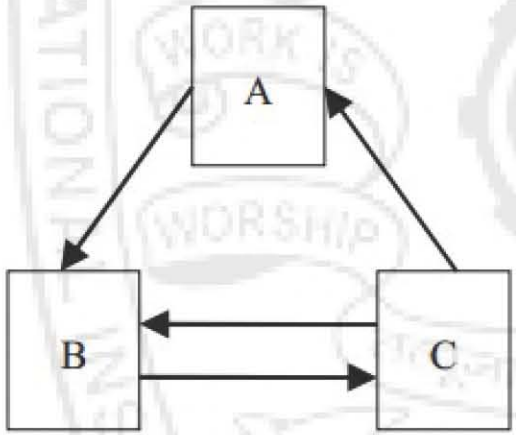
$$PR(C) = 0.1 + 0.7 \times PR(B) \quad \text{---- (iii)}$$

By solving the resulting system of linear equations, we get

PR(A) = 1/3 = 0.33 **PR(B) = 1/3 = 0.33** **PR(C) = 1/3 = 0.33**

Calculating a Webpage's PageRank (PR)

Consider another example:

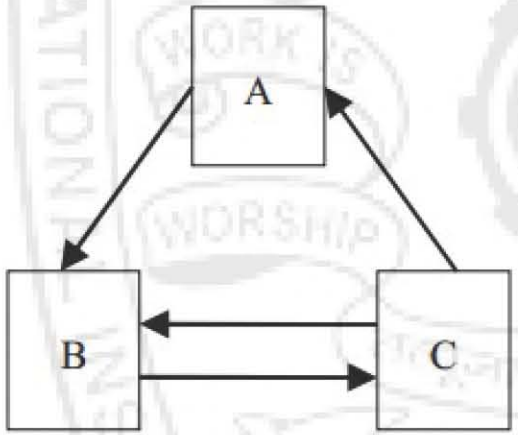


Substituting value of d and N, and by solving the resulting systems of linear equations, we get

PR(A) = **PR(B) =** **PR(C) =**

Calculating a Webpage's PageRank (PR)

Consider another example:

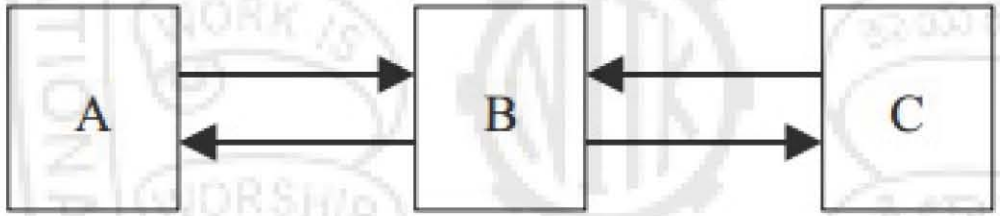


Substituting value of d and N, and by solving the resulting systems of linear equations, we get

PR(A) = 0.2314 **PR(B) = 0.3933** **PR(C) = 0.3753**

Calculating a Webpage's PageRank (PR)

Consider another example:

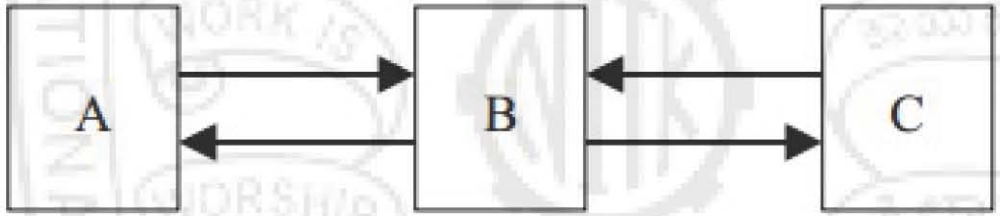


Substituting value of d and N and by solving the resulting systems of linear equations, we get

PR(A) = **PR(B) =** **PR(C) =**

Calculating a Webpage's PageRank (PR)

Consider another example:

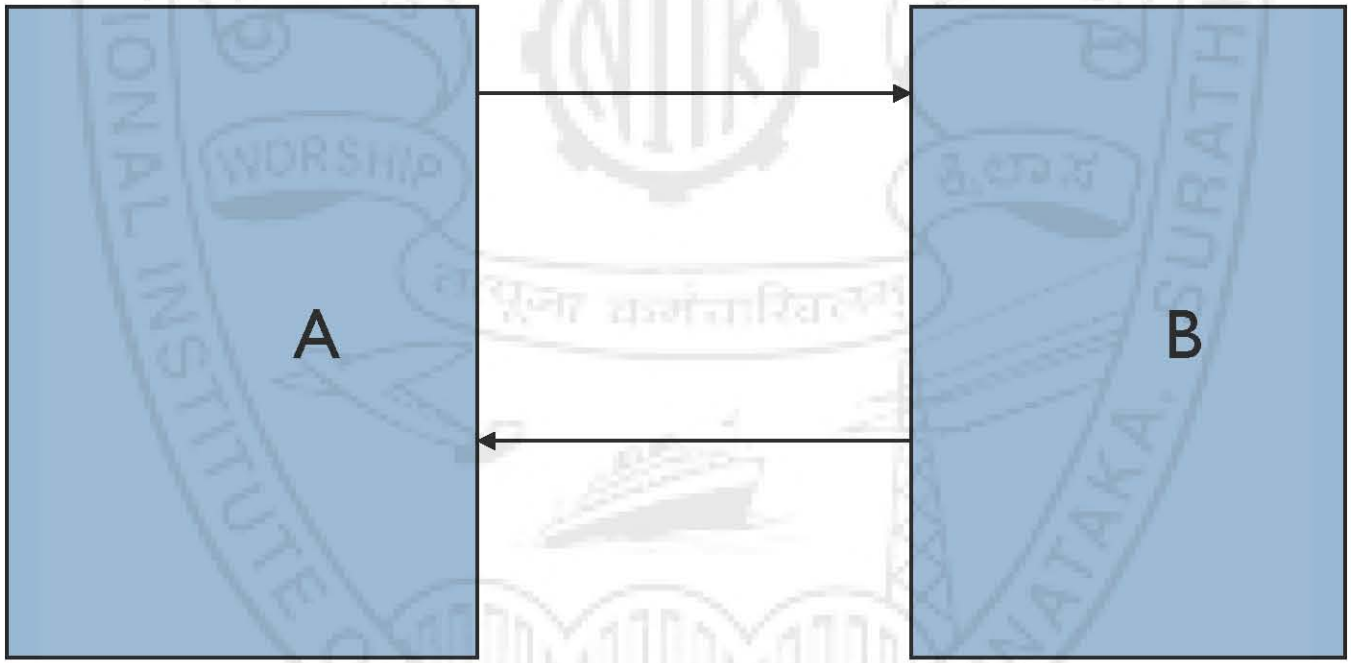


Substituting value of d and N and by solving the resulting systems of linear equations, we get

PR(A) = 0.2647 **PR(B) = 0.4706** **PR(C) = 0.2647**

Calculating a Webpage's PageRank (PR)
Iterative Computation

Consider two pages A and B, pointing to each other.



Calculating a Webpage's PageRank (PR)

Iterative Computation

Lets start with $PR(A) = PR(B) = 10$

1st iteration:

$$PR(A) = (1\text{-}d)*1/N + d*(PR(B)/C(B))$$
$$= 0.15*0.5 + 0.85 * (10/1)$$
$$= 8.58$$
$$PR(B) = (1\text{-}d) *1/N + d*(PR(A)/C(A))$$
$$= 0.15*0.5 + 0.85 * (8.58/1)$$
$$= 7.36$$

Calculating a Webpage's PageRank (PR)

Iterative Computation

After 2nd iteration:

$$PR(A) = (1\text{-}d)*1/N + d*(PR(B)/C(B))$$
$$= 0.15*0.5 + 0.85 * (7.36/1)$$
$$= 6.331$$

$$PR(B) = (1\text{-}d)*1/N + d*(PR(A)/C(A))$$
$$= 0.15*0.5 + 0.85 * (6.331/1)$$
$$= 5.456$$

And so on….. till?

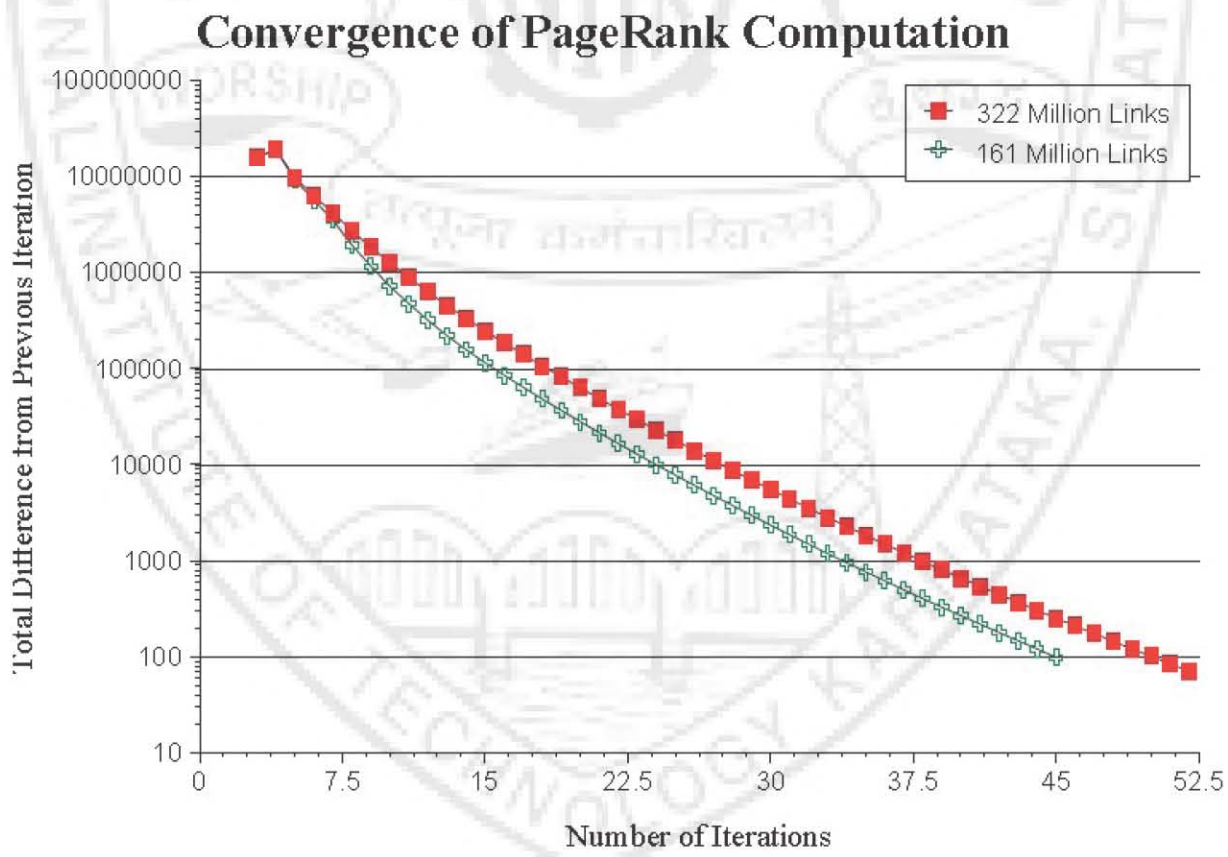
Calculating a Webpage's PageRank (PR)
Iterative Computation

→ Iterations should be repeated till PR values converge…

i.e., start with any value of PR, and repeat iterations till PR values converge.

PageRank: Convergence Behavior

- PR (322 Million Links): 52 iterations
- PR (161 Million Links): 45 iterations
  - Scaling factor is roughly linear !



Personalised PageRank

- Similar to PageRank, but more biased towards user's preferences.
- Instead of teleporting uniformly to any page, the jump is "biased" to prefer some specific topics/pages over others.

$$PR(P) = dv + (1-d)\left(\frac{PR(W_1)}{O(W_1)} + \frac{PR(W_2)}{O(W_2)} + \ldots + \frac{PR(W_n)}{O(W_n)}\right)$$

Personalised PageRank

- For example -
  - v has higher value for your preferences (e.g. most visited page/frequent topic/liked article genre) and 0 otherwise.
  - *i.e.*, algorithm *gives preference* to the topics you are interested in (obtained from Query log analysis/Clickstream analysis).

Personalised PageRank

- For example -
  - v has higher value for your preferences (e.g. most visited page/frequent topic/liked article genre) and 0 otherwise.
  - *i.e.*, algorithm *gives preference* to the topics you are interested in (obtained from Query log analysis/Clickstream analysis).
- Typically used in recommender systems.

PageRank - Summary

- Concepts considered:
  - A page is important if it has *many* links (incoming? outgoing?)
  - Are all links *equally* important?
  - A page is important if it is cited by other *important* pages.
  - A page is important if it is linked to (highly ranked) *topically related* pages.

Learning to Rank

Learning to Rank Concept

- Apply machine learning techniques to **learn the ranking of the results**
  - a learning algorithm fed with training data that contains ranking information
- Define a loss function to be minimized.

Learning to Rank - Process

- Uses query chains – i.e., a sequence of reformulated queries
  - Automatically detect query chains from large-scale search logs
  - Use query chains to infer relevance of results in each query and between results from all queries in the chain
  - Use ML models (SVM/Genetic algorithm/swarm intelligence…..) to learn a retrieval function from the results.

Inferring Relevance from Query Chains

- Some strategies for generating feedback from query chains
  - **Click >q'Skip Above:** A clicked on document is more relevant than any documents above it.

Inferring Relevance from Query Chains

- Some strategies for generating feedback from query chains
  - **Click >q'Skip Above:** A clicked on document is more relevant than any documents above it
  - **Click Second >q'No-Click First: :** Given the first two document results, if the second was clicked, then it is more relevant

Inferring Relevance from Query Chains

- Some strategies for generating feedback from query chains
  - **Click >q'Skip Above:** A clicked on document is more relevant than any documents above it
  - **Click Second >q'No-Click First: :** Given the first two document results, if the second was clicked, then it is more relevant
  - **Click >q' Skip Earlier Query:** A clicked on document is more relevant than any that were skipped in any earlier query

Inferring Relevance from Query Chains

- Some strategies for generating feedback from query chains
  - **Click >q'Skip Above:** A clicked on document is more relevant than any documents above it
  - **Click Second >q'No-Click First:** Given the first two document results, if the second was clicked, it is more relevant.
  - **Click >q' Skip Earlier Query:** A clicked on document is more relevant than any that were skipped in any earlier query
  - **Click >q'Top Two Earlier Query:** If nothing was clicked in this query, the clicked document from earlier query is more relevant than the top two from this query.

Generating relevance training data

- Given query Q, three types of training data are then generated:
  - **pointwise**: a set of relevant pages for Q
  - **pairwise**: a set of pairs of relevant pages indicating the ranking relation between the two pages
  - **listwise**: a set of ordered relevant pages: p1 $\succ$ p2 $\succ$ $\cdots$ $\succ$ pm

Generating relevance training data

- Given query Q, three types of training data are then generated:
  - **pointwise**: a set of relevant pages for Q
  - **pairwise**: a set of pairs of relevant pages indicating the ranking relation between the two pages
  - **listwise**: a set of ordered relevant pages: p1 $\succ$ p2 $\succ$ $\cdots$ $\succ$ pm

Generating relevance training data

- Training data may also be originated from editorial judgements made by human experts.
  - ML classifiers are trained so that ranking can be improved over time.

Learning the Ranking Function

- An effort to learn the ranking function, rather than the ranking order.

- E.g. using a genetic algorithm based approach.
  - members of the population are function instances over a given set of ranking features.
  - at every step of the genetic algorithm, different functions are mutated or mixed.
  - the goodness of each learning function is evaluated through a set of ground truth or training data.
  - after many iterations, the fittest ranking function is selected.

More reading...

- Borodin, A. et al. (2001, April). Finding authorities and hubs from link structures on the world wide web. In *Proceedings of the 10th international conference on World Wide Web* (pp. 415-429).
- Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd. *The PageRank citation ranking: Bringing order to the web*. Stanford InfoLab, 1999.
- Haveliwala, T., Kamvar, S., & Jeh, G. (2003). *An analytical comparison of approaches to personalizing pagerank*. Stanford.
- Liu, Tie-Yan. "Learning to rank for information retrieval." *Foundations and Trends® in Information Retrieval* 3.3 (2009): 225-331.