# 12. Link Analysis in Web Search

**Prof. Dr. Goran Glavaš**

Center for AI and Data Science (CAIDAS)
Fakultät für Mathematik und Informatik
Universität Würzburg

## After this lecture, you'll...

- Understand why content relevance is not enough for web search
- Perceive web as a large graph
- Know the inner workings of PageRank algorithm
- Understand the mechanisms of the HITS algorithm

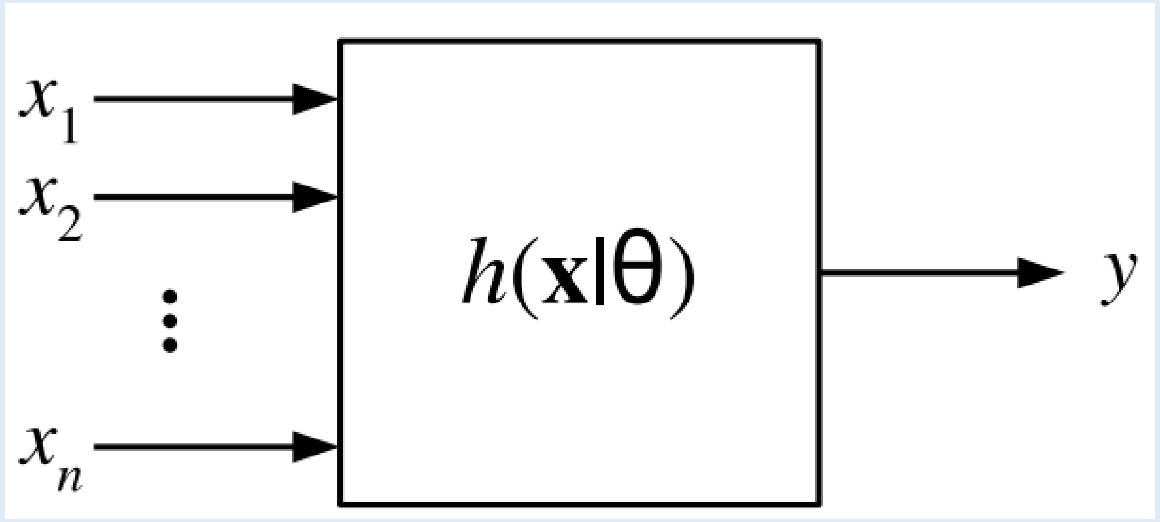
# Outline

- Recap of lectures #9 and #10
- Beyond content: source reputation
- PageRank
- HITS

# Recap of previous lectures

- Machine learning for IR
  - **Q:** What is the difference between supervised and unsupervised machine learning?
  - **Q:** What are the types of supervised classifiers used in IR/NLP?
- Text classification for IR
  - **Q:** Explain the setting in which text classification is beneficial for IR?
  - **Q:** What are the basic ideas behind logistic regression and CNN?
- Text clustering for IR
  - **Q:** Where can we employ text clustering in IR?
  - **Q:** How does single pass clustering work? What about k-means?
- Learning to rank (L2R)
  - **Q:** What is learning to rank? Why do we use it?
  - **Q:** Explain the differences between point-wise, pair-wise, and list-wise approaches to L2R
- IR Evaluation
  - **Q:** What measures we use to evaluate rankings? Why are P, R, and F not enough?
  - **Q:** What is pooling and why do we do it?

## Supervised classification



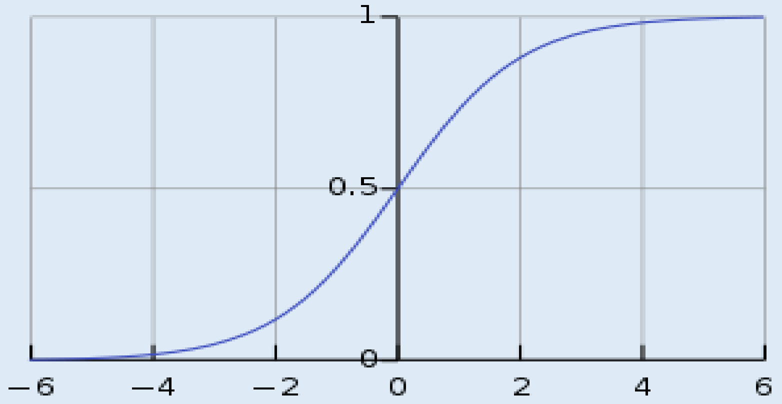
- Types of classifiers in IR/NLP:
  - Binary classification: just two output labels (yes/no, 0/1)
  - Multi-class classification: each instance has one of K labels
  - Multi-label classification: an instance can have more than one label at once
  - Sequence labeling: input is a sequence of instances and the output is the sequence of labels

## Logistic regression

- Despite its name, **logistic regression** is a classification algorithm
  - We will focus on binary classification – logistic regression computes the probability that some instance **x** belongs to some class ($y$ = 1)

$$h(\mathbf{x} \mid \boldsymbol{\theta}) = P(y = 1 \mid \mathbf{x}) = \frac{1}{1+\exp(-\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x})} = \sigma(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x})$$

- Logistic regression is based on a logistic function: $\sigma(a) = 1 / (1 + e^{-a})$
- The logistic function maps the input value to the output interval [-1, 1]



# K-means

- Arguably the most famous and widely used clustering algorithm
- Requires the number of clusters k to be predefined – K clusters, **S** = {$S_1$, $S_2$, ..., $S_k$}, represented by mean vectors $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, ..., $\boldsymbol{\mu}_k$
- **K-means** clusters instances ($\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$) by finding the partition **S** that minimizes the within-cluster distances (maximizing the within-cluster similarities):

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2$$

- **Q:** How to find the optimal clusters (i.e., minimize the above sum of within-cluster distances)?
- **A:** Using iterative optimization

# Learning to Rank

- Learning to rank is a supervised information retrieval paradigm that
  - Describes instances of document-query pairs (d, q) with a range of features
  - Learns (with some ML algorithm) the mapping between these features and relevance

- Three different learning-to-rank approaches:
  1. **Point-wise approach**
     - Classify a single document-query (d, q) pair for relevance
  2. **Pair-wise approach**
     - Classify, for a pair of documents, which one is more relevant for the query, i.e., whether $r(d_1, q) > r(d_2, q)$ or $r(d_1, q) < r(d_2, q)$
  3. **List-wise approach**
     - Classify the whole ranking as either correct or wrong

## Mean average precision

- We would like to have a single-figure measure of retrieval effectiveness across all recall levels
- **Average precision (AP)** for a query *q* with relevant documents $\{d_1, ..., d_m\}$ is computed by averaging the precision scores measure at ranks of relevant docs:

$$\mathrm{AP}(q) = \frac{1}{m} \sum_{k=1}^{m} P(R_k)$$

- $R_k$ is the rank at which we find the k-th relevant document
- **Mean average precision** is AP averaged over the set of queries ***Q***:

$$\mathrm{MAP} \quad = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} P(R_{jk})$$

# Pooling

- In most IR evaluations, we are comparing the performance different models (or different variants of the same model)
- **Pooling** is a method for reducing the number of required relevance judgement annotations in settings where we compare different IR models
- Example: evaluating models $r_1, \ldots, r_K$ (expected N relevant docs for query $q$)
- Pooling involves the following steps:
  1. Rank all documents with each of the models $r_1, \ldots, r_K$
  2. In each of the rankings $R_1, \ldots, R_K$, take only the top N results: $R_{1,N}, \ldots, R_{K,N}$
  3. The documents in the union of retrieved top results are to be annotated for relevance for the given query: $R_{1,N} \cup \ldots \cup R_{K,N}$

- **Q:** Is it still possible to ignore some truly relevant document for relevance judgements? If so, is that a problem?

# Outline

- Recap of lectures #9 and #10
- Beyond content: source reputation
- PageRank
- HITS

# Web search: beyond content

- In the web search setting, ranking based on pure content relevance is not possible due to:

1. **Web's massive size**
   - Impossible (even for Google) to compute relevance scores for all web pages
   - Even with all the tricks we discussed: pre-clustering, reducing the vector length, etc.

2. **Spam websites**
   - If the ranking of web results was based purely on content, it would be very easy to create spam web pages that are very relevant for certain queries
   - E.g., a website that wants to be ranked high for queries relating to *football* would simply contain a lot of football-related terms: *game*, *match*, *ball*, *messi*, *ronaldo*, *UEFA*, ...

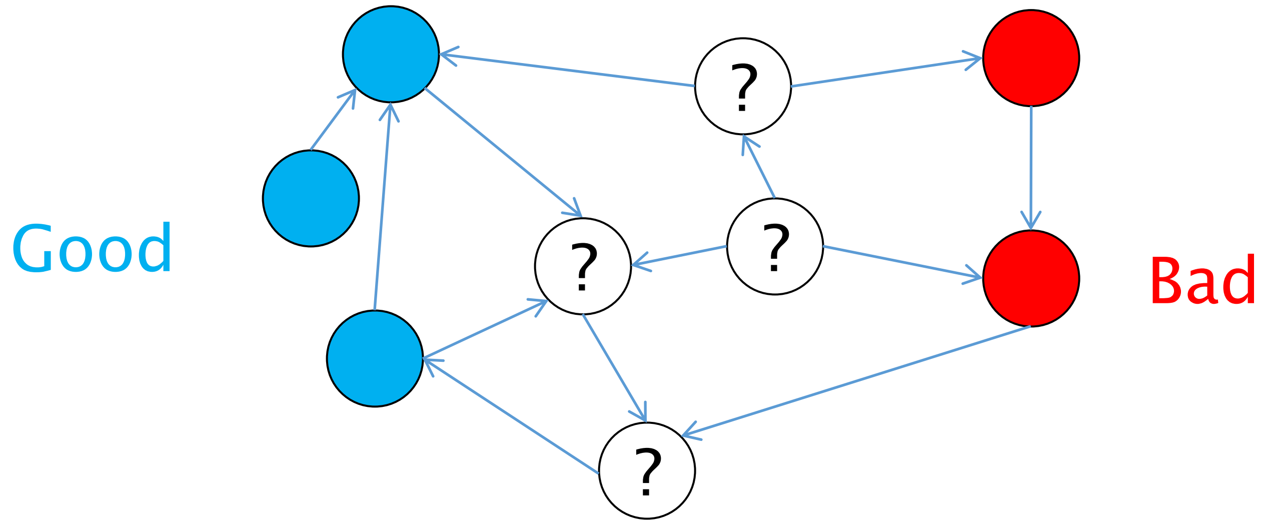
# Web search: beyond content

- On the web, we need an additional measure/metric of popularity/authority/importance that allows the following:
  1. The scores can be pre-computed for the entire web
  2. Spam websites should have low scores with this measure
  3. Pages with low scores not considered for the result set (ranking)

- **Q:** How could we define such a measure of popularity to discriminate between good and bad pages on the web?
- **A:** By exploiting the linked structure of the web!
  - Web is a huge directed graph: web pages are vertices and hyperlinks edges

# Exploiting linked structure of the Web

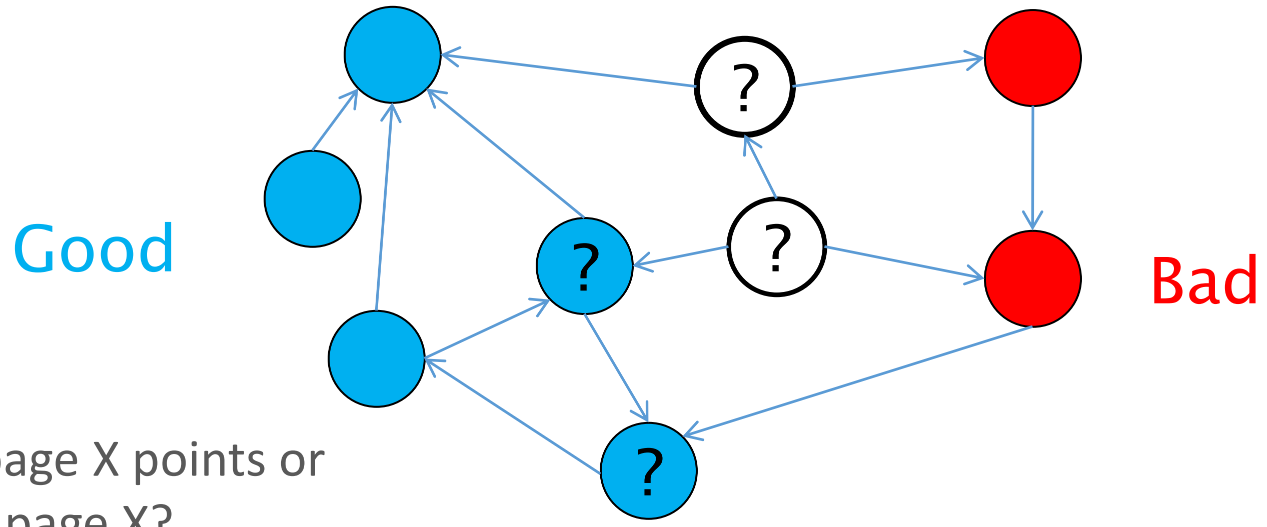
- **First idea**
  - Let’s start from a seed set of good and bad web pages
  - Iteratively label other pages:
    - As bad if they contain a hyperlink to some bad page
    - As good if they contain a hyperlink to some good page



## Exploiting linked structure of the Web

- Iteratively label other pages:
  - As bad if they contain a hyperlink to some bad page
  - As good if they contain a hyperlink to some good page

- Issues:
  - What if a page points to both good and bad pages?
  - If two pages are good/bad, which one is better/worse?
  - How to assign label for page X?
    - Based on pages to which the page X points or
    - Based on pages which point to page X?



# Link Analysis

- We need a real-valued importance measure that exploits the fact that web is a large directed graph

- **Assumptions**
  - A hyperlink between pages denotes a conferral of authority
  - The text in the anchor of the hyperlink determines the target page

- Finding authoritative websites
  - Relies on the above assumption that hyperlinks are „recommendations”
  - More pages point to a page, the more authoritative the page is
  - Algorithms:
    - **PageRank** (computes single PageRank score for each page)
    - **HITS** (computes two scores for each page: authority score and hub score)

## Outline

- Recap of lectures #9 and #10
- Beyond content: source reputation
- PageRank
- HITS

# PageRank hypothesis

- **Key assumption**
  - **Hyperlinks** of the web graph are **recommendations**
  - A page with more recommendations is more important

- The status of the recommender also matters:
  - Recommendations from more important recommenders are worth more
  - The overall number of recommendations issued by the recommender also matters

- **PageRank hypothesis**
  - A web page is important if it is pointed to by other important pages that do not point to too many other pages

## PageRank formula

- Heinrich Hertz (on Maxwell’s equations):

  *One cannot escape the feeling that these mathematical formulae have an independent existence and an intelligence on their own, that they are wiser than we are, wiser even than their discoverers, that we get more out of them than was originally put into them.*

- PageRank formula is one of the most important formulas in Computer Science:

$$\boldsymbol{\pi}^T = \boldsymbol{\pi}^T(\alpha\mathbf{S} + (1-\alpha)\mathbf{E})$$

- But lets take it step by step...

# Original PageRank summation

- **PageRank score** of a page $P_i$ is the sum of importances of all pages that have hyperlinks to $P_i$, each normalized by the total number of hyperlinks of that page

$$r(P_i) = \sum_{P_j \in B_{P_i}} \frac{r(P_j)}{|P_j|}$$

- $B_{P_i}$ is the set of all pages that have hyperlinks to $P_i$
- $|P_j|$ is the number of pages that page $P_j$ hyperlinks to

- **Q:** But PageRank scores of pages $P_j$ linking to $P_i$ are unknown. How do we determine the PageRank scores for pages then?

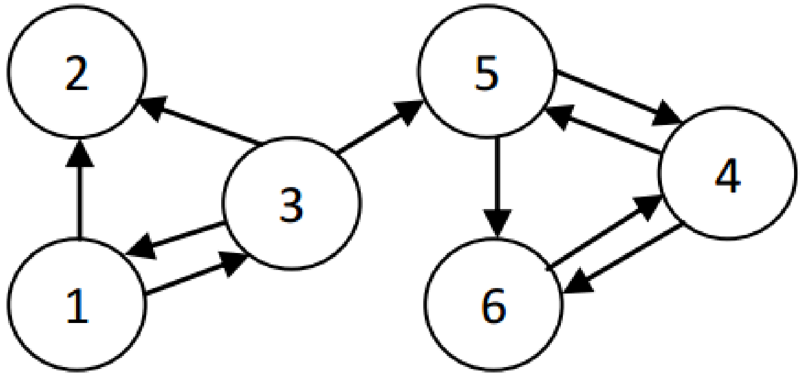
# Iterative computation of PageRank scores

- **Solution:** compute the scores iteratively
- **The idea:**
  1. Assign the same initial scores to all pages, $P_i = \frac{1}{n}$, where *n* is the total number of nodes (i.e., web pages) in the graph
  2. Run the PageRank summation formula iteratively, until the PageRank scores for all pages converge (remain the same between two iterations)

$$r_{k+1}(P_i) = \sum_{P_j \in B_{P_i}} \frac{r_k(P_j)}{|P_j|}$$

- **Q:** How do we know if this iterative computation will converge?

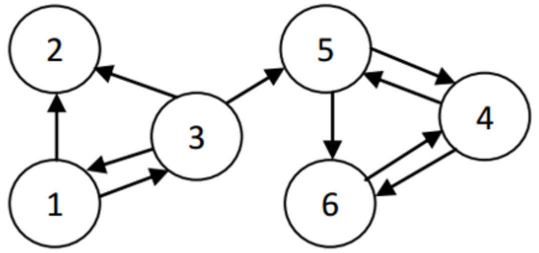
# Iterative PageRank computation – example



| Iteration 0 | Iteration 1 | Iteration 2 | Rank |
|---|---|---|---|
| $r_0(P_1) = 1/6$ | $r_1(P_1) = 1/18$ | $r_2(P_1) = 1/36$ | 5 |
| $r_0(P_2) = 1/6$ | $r_1(P_2) = 5/36$ | $r_2(P_2) = 1/18$ | 4 |
| $r_0(P_3) = 1/6$ | $r_1(P_3) = 1/12$ | $r_2(P_3) = 1/36$ | 5 |
| $r_0(P_4) = 1/6$ | $r_1(P_4) = 1/4$ | $r_2(P_4) = 17/72$ | 1 |
| $r_0(P_5) = 1/6$ | $r_1(P_5) = 5/36$ | $r_2(P_5) = 11/72$ | 3 |
| $r_0(P_6) = 1/6$ | $r_1(P_6) = 1/6$ | $r_2(P_6) = 14/72$ | 2 |

# PageRank summation: matrix form

- Using the previous formula, PageRank scores need to be computed (updated) one page at a time
- With the **matrix form** of the PageRank equation, all PageRank scores can be updated at once
- Let's introduce some **notation**:
  - $\pi$ is the vector of PageRank scores of pages, $\pi_i = r(P_i)$
  - **H** is the row-normalized adjacency matrix of the Web graph
    - $H_{ij}$ equals $1/|P_i|$ if there is a hyperlink from page $P_i$ to page $P_j$
    - $H_{ij}$ equals 0 if there is no hyperlink from page $P_i$ to page $P_j$



$$\mathbf{H} = \begin{array}{c|cccccc} & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\ \hline P_1 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ P_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ P_3 & 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ P_4 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ P_5 & 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ P_6 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}$$

## PageRank summation: matrix form

- The iterative PageRank formula can now be written in matrix form:

$$\boldsymbol{\pi}_{k+1}^T = \boldsymbol{\pi}_k^T \mathbf{H}$$

- Each iteration requires one vector-matrix multiplication: *O(n²)*
- However, our web graph adjacency matrix **H** is a (very) sparse matrix
  - Most pages have links only to few other pages: *O(nnz(**H**))*
    - Effectively, the complexity is *O(kn)* = *O(n)*

- **Key question**: Will this iterative process converge?
  - Does convergence depend on the initial PageRank scores (at iteration 0)?
  - Under what conditions on **H** is the iterative process guaranteed to converge?

# PageRank in matrix form

- The PageRank equation $\boldsymbol{\pi}_{k+1}^{T} = \boldsymbol{\pi}_{k}^{T}\mathbf{H}$ is a **power method** applied to the web graph's row-normalized adjacency matrix **H**

- From linear algrebra, we know that a power method applied to some matrix **P** converges if and only if the matrix **P** is:
  1. **Stochastic** – each row is a probability distribution (i.e., row values sum up to 1)
  2. **Irreducible** – there is a probability assigned to transition from each node to every other node (i.e., we can surely reach each node from every other node)
  3. **Aperiodic** – no cycle of length k exists for which $\mathbf{P} = \mathbf{P}^{k}$, $\mathbf{P}^{1} = \mathbf{P}^{k+1}$, $\mathbf{P}^{2} = \mathbf{P}^{k+2}$, ...

- In general, row-normalized adjacency matrix **H** satisfies none of the above
  - **H** is a substochastic transition probability matrix of a Markov chain

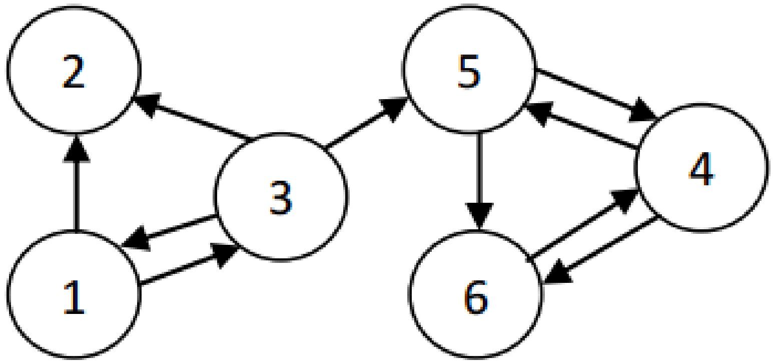
# Adjustments to the basic PageRank model

- To ensure **convergence** of the power method, we adjust the matrix **H**
  - We need to transform **H** into a matrix that is stochastic, irreducible, and aperiodic
- Brin & Page (authors of PageRank and founders of Google) introduce the concept of a **random surfer**
  - A random surfer surfs the web by following the hyperlink structure of the Web (on each page the hyperlink to follow is randomly selected)
- **Stochasticity adjustment**
  - When on a page that has no hyperlinks, surfer may randomly „jump” (with equal probability) to any page (including the current one)

$$\mathbf{S} = \mathbf{H} + \mathbf{a}(\frac{1}{n} \cdot \mathbf{e}^T)$$

- $a_i = 1$ if page $i$ has no hyperlinks and **e** is a vector of ones

# Stochasticity adjustment example



$$S = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{array} \begin{array}{c} \begin{array}{cccccc} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \end{array} \\ \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

## Adjustments to the basic PageRank model

- After the stochasticity adjustment, the obtained matrix **S** is stochastic, but in general it is still not primitive (primitive = irreducible and aperiodic)

- **Primitivity adjustment**
  - Occasionally, a random surfer gets bored and abandons following hyperlinks and randomly jumps to a new page (e.g., by entering URL in browser)
  - We assign probability $\alpha$ to surfer following hyperlinks
  - The probability of random „jump" is then $1 - \alpha$

$$\mathbf{G} = \alpha\mathbf{S} + (1 - \alpha)\mathbf{E}$$

  - Matrix **G** captures both the hyperlinking following ($\alpha$**S**) and random jumps ((1- $\alpha$)**E**)
  - **E** is a normalized matrix of ones, $\mathbf{E} = \frac{1}{n} \cdot \mathbf{e}\mathbf{e}^T$

# Adjustments to the basic PageRank model

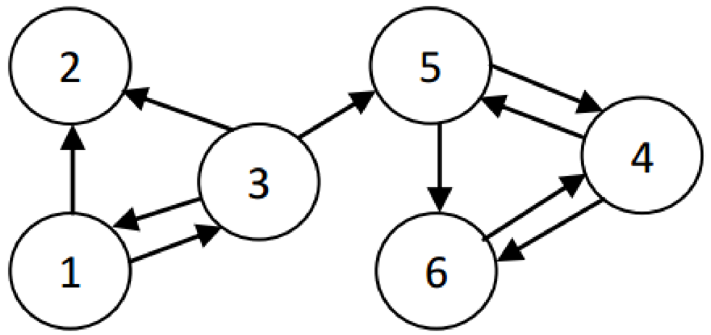
- The matrix G, known as the „Google matrix” is:
  - **Stochastic**
  - **Irreducible**
  - **Aperiodic**

- This means that the power method applied to **G** will **converge** for any graph (i.e., any starting row-normalized adjacency matrix **H**)

$$\boldsymbol{\pi}_{k+1}^{T} = \boldsymbol{\pi}_{k}^{T}\mathbf{G}$$

- At convergence, we obtain the final positive PageRank vector **π**

# Primitivity adjustment example



$$\alpha = \frac{9}{10}, n = 6,\ G_{ij} = \alpha S_{ij} + (1-\alpha)\cdot\frac{1}{n} = \frac{9}{10}\cdot S_{ij} + \frac{1}{10}\cdot\frac{1}{6}$$

$$\mathbf{G} = \begin{array}{c|cccccc} & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\ \hline P_1 & 1/60 & 7/15 & 7/15 & 1/60 & 1/60 & 1/60 \\ P_2 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ P_3 & 19/60 & 19/60 & 1/60 & 1/60 & 19/60 & 1/60 \\ P_4 & 1/60 & 1/60 & 1/60 & 1/60 & 7/15 & 7/15 \\ P_5 & 1/60 & 1/60 & 1/60 & 7/15 & 1/60 & 7/15 \\ P_6 & 1/60 & 1/60 & 1/60 & 11/12 & 1/60 & 1/60 \end{array}$$

## Computational complexity concerns

- Unlike the Web graph adjacency matrix H, Google matrix G is a dense matrix
  - Each iteration of the power method on G will have quadratic complexity *$O(n^2)$*
- Fortunately, the Google matrix **G** can be written as the rank-one update of the very sparse initial adjacency matrix **H**

$$\begin{aligned}\mathbf{G} &= \alpha\mathbf{S} + (1-\alpha)\mathbf{E} \\ &= \alpha(\mathbf{H} + \frac{1}{n}\mathbf{a}\mathbf{e}^T) + (1-\alpha)\frac{1}{n}\mathbf{e}\mathbf{e}^T \\ &= \alpha\mathbf{H} + \big(\alpha\mathbf{a} + (1-\alpha)\mathbf{e}\big)\cdot\frac{1}{n}\mathbf{e}^T\end{aligned}$$

- So, the power method on **G** can be computed without ever explicitly computing matrices **S** or **G** (assuming **π*e** = 1):

$$\begin{aligned}\boldsymbol{\pi}_{k+1}^T &= \boldsymbol{\pi}_k^T\mathbf{G} \\ &= \alpha\boldsymbol{\pi}_k^T\mathbf{H} + (\alpha\boldsymbol{\pi}_k^T\mathbf{a} + \boldsymbol{\pi}_k^T(1-\alpha)\mathbf{e})\frac{\mathbf{e}^T}{n}\end{aligned}$$

# PageRank for weighted graphs

- PageRank is used in many other contexts, not only for web search
- Often, graphs on which we run PageRank are weighted – edges have real values
  - For PageRank to work, all weights must be positive
- Let w(j,i) be the weight of the edge pointing from page $P_j$ to page $P_i$
- Summation version of the PageRank algorithm for weighted graphs:

$$r(P_i) = \sum_{P_j \in BPi} \frac{w(j,i) r(Pj)}{\sum_k w(j,k)}$$

- For the matrix form of the PageRank, we just start from the **weighted** adjacency matrix **H** of the web graph, which we row-normalize

## Outline

- Recap of lectures #9 and #10
- Beyond content: source reputation
- PageRank
- HITS

# HITS algorithm

- Unlike PageRank, **HITS** (Hypertext Induced Topic Search) qualifies pages in terms of two different scores

- HITS defines **hubs** and **authorities**
  - A page is considered a hub if it contains many hyperlinks
  - A page is considered an authority if many hyperlinks point to it

- **HITS hypothesis**
  - A page is a good hub if it points to good authorities, and a good authority if it is pointed to by good hubs
  - So, every page $P_i$ is assigned two scores: authority score $x_i$ and hub score $y_i$

## Original HITS summations

- Lets assume that each page $P_i$ is assigned some initial scores
  - Authority score $x_i^{(0)}$
  - Hub score $y_i^{(0)}$

- HITS iteratively refines both authority and hub scores:

$$x_i^{(k+1)} = \sum_{j:e_{ji} \in E} y_j^{(k)}$$

$$y_i^{(k+1)} = \sum_{j:e_{ij} \in E} x_j^{(k)}$$

## HITS – matrix form

- Similarly as for PageRank, we can compute all updates at once using matrix formulation, instead of updating hub and authority scores page by page
- Let **L** be the pure adjacency matrix of the directed Web graph (**NOTE:** **L** is not raw-normalized as **H** in PageRank)
- HITS in matrix/vector form is given as:

$$\mathbf{x}^{(k+1)} = \mathbf{L}^T \mathbf{y}^{(k)}$$
$$\mathbf{y}^{(k+1)} = \mathbf{L}\mathbf{x}^{(k)}$$

- After each iteration (*k*), the authority vector $\mathbf{x}^{(k)}$ and hub vector $\mathbf{y}^{(k)}$ must be normalized (i.e., must represent a probability distribution) for HITS to converge

## HITS algorithm

- Until convergence repeat:
  1. Compute the new scores $\mathbf{x}^{(k+1)}$ and $\mathbf{y}^{(k+1)}$
  2. Normalize $\mathbf{x}^{(k+1)}$ and $\mathbf{y}^{(k+1)}$

$$\mathbf{x}^{(k+1)} = \mathbf{L}^T\mathbf{y}^{(k)}$$
$$\mathbf{y}^{(k+1)} = \mathbf{L}\mathbf{x}^{(k)}$$

- By simple arithmetics, and substitution, we obtain the following formulas:

$$\mathbf{x}^{(k+1)} = \mathbf{L}^T\mathbf{L}\mathbf{x}^{(k-1)}$$
$$\mathbf{y}^{(k+1)} = \mathbf{L}\mathbf{L}^T\mathbf{y}^{(k-1)}$$

- Above equations denote the power method applied to $\mathbf{L}^T\mathbf{L}$ and $\mathbf{L}\mathbf{L}^T$, respectively

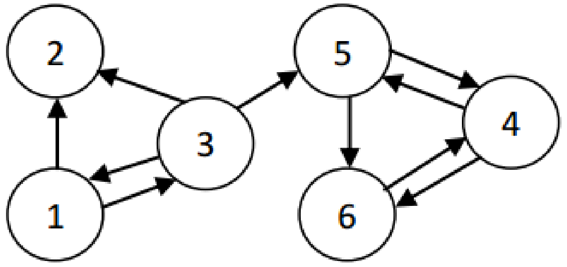
# HITS Convergence

- **Q:** Under which conditions does HITS converge?
  - We would need $\mathbf{L}^T\mathbf{L}$ and $\mathbf{L}\mathbf{L}^T$ to satisfy the same criteria as **G** for PageRank

- However, **L** is a graph adjacency matrix (with all non-negative scores) and because of this $\mathbf{L}^T\mathbf{L}$ and $\mathbf{L}\mathbf{L}^T$ are symmetric, positive semi-definite, and non-negative
  - Such matrices have real and non-negative eigenvalues
  - Power method applied to matrices with such eigenvalues always converges

- As long as we **normalize** the hub and authority scores after every iteration, HITS will converge for any graph (i.e., any raw adjacency matrix **L**)

# HITS – example



$$\mathbf{L} = \begin{array}{c|cccccc} & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\ \hline P_1 & 0 & 1 & 1 & 0 & 0 & 0 \\ P_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ P_3 & 1 & 1 & 0 & 0 & 1 & 0 \\ P_4 & 0 & 0 & 0 & 0 & 1 & 1 \\ P_5 & 0 & 0 & 0 & 1 & 0 & 1 \\ P_6 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}$$

$$\mathbf{L}^T\mathbf{L} = \begin{array}{c|cccccc} & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\ \hline P_1 & 1 & 1 & 0 & 0 & 1 & 0 \\ P_2 & 1 & 2 & 1 & 0 & 1 & 0 \\ P_3 & 0 & 1 & 1 & 0 & 0 & 0 \\ P_4 & 0 & 0 & 0 & 2 & 0 & 1 \\ P_5 & 1 & 1 & 0 & 0 & 2 & 1 \\ P_6 & 0 & 0 & 0 & 1 & 1 & 2 \end{array} \qquad \mathbf{L}\mathbf{L}^T = \begin{array}{c|cccccc} & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\ \hline P_1 & 2 & 0 & 1 & 0 & 0 & 0 \\ P_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ P_3 & 1 & 0 & 3 & 1 & 0 & 0 \\ P_4 & 0 & 0 & 1 & 2 & 1 & 0 \\ P_5 & 0 & 0 & 0 & 1 & 2 & 1 \\ P_6 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}$$

| | $\mathbf{x}^{(0)}$ | $\mathbf{y}^{(0)}$ | $\mathbf{x}^{(1)}$ | $\mathbf{y}^{(1)}$ | ... | $\mathbf{x}^{(5)}$ | $\mathbf{y}^{(5)}$ | ... | $\mathbf{x}^{(10)}$ | $\mathbf{y}^{(10)}$ | ... | $\mathbf{x}^{(40)}$ | $\mathbf{y}^{(40)}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $P_1$ | 0.3 | 0.3 | 0.117 | 0.206 | ... | 0.144 | 0.167 | ... | 0.160 | 0.179 | ... | 0.165 | 0.183 |
| $P_2$ | 0.1 | 0.1 | 0.147 | 0 | ... | 0.205 | 0 | ... | 0.234 | 0 | ... | 0.243 | 0 |
| $P_3$ | 0 | 0 | 0.029 | 0.241 | ... | 0.061 | 0.363 | ... | 0.074 | 0.381 | ... | 0.078 | 0.386 |
| $P_4$ | 0.4 | 0.4 | 0.294 | 0.275 | ... | 0.124 | 0.253 | ... | 0.087 | 0.249 | ... | 0.078 | 0.248 |
| $P_5$ | 0 | 0 | 0.176 | 0.207 | ... | 0.260 | 0.159 | ... | 0.269 | 0.142 | ... | 0.271 | 0.138 |
| $P_6$ | 0.2 | 0.2 | 0.235 | 0.069 | ... | 0.206 | 0.056 | ... | 0.174 | 0.047 | ... | 0.165 | 0.044 |

## Now you...

- Understand why content relevance is not enough for web search
- See the web as a large graph
- Understand the inner workings of the PageRank algorithm
- Understand the mechanisms of the HITS algorithm