Link Analysis Ranking
How do search engines decide how to rank your query results?

• Guess why Google ranks the query results the way it does

• How would you do it?
Naïve ranking of query results

• Given query \( q \)
• Rank the web pages \( p \) in the index based on \( \text{sim}(p,q) \)

• Scenarios where this is not such a good idea?
Why Link Analysis?

• First generation search engines
  – view documents as flat text files
  – could not cope with size, spamming, user needs
    • Example: Honda website, keywords: automobile manufacturer

• Second generation search engines
  – Ranking becomes critical
  – use of Web specific data: Link Analysis
  – shift from relevance to authoritativeness
  – a success story for the network analysis
Link Analysis: Intuition

• A link from page \( p \) to page \( q \) denotes endorsement
  – page \( p \) considers page \( q \) an authority on a subject
  – mine the web graph of recommendations
  – assign an authority value to every page
Link Analysis Ranking Algorithms

- Start with a collection of web pages
- Extract the underlying hyperlink graph
- Run the LAR algorithm on the graph
- Output: an authority weight for each node
Algorithm input

- **Query dependent**: rank a small subset of pages related to a specific query
  - HITS (Kleinberg 98) was proposed as query dependent

- **Query independent**: rank the whole Web
  - PageRank (Brin and Page 98) was proposed as query independent
Query-dependent LAR

• Given a query $q$, find a subset of web pages $S$ that are related to $S$
• Rank the pages in $S$ based on some ranking criterion
Query-dependent input

Root Set
Query-dependent input
Query dependent input
Query dependent input

Base Set

IN

Root Set

OUT
Properties of a good seed set $S$

- $S$ is relatively small.
- $S$ is rich in relevant pages.
- $S$ contains most (or many) of the strongest authorities.
How to construct a good seed set $S$

• For query $q$ first collect the $t$ highest-ranked pages for $q$ from a text-based search engine to form set $\Gamma$

• $S = \Gamma$

• Add to $S$ all the pages pointing to $\Gamma$

• Add to $S$ all the pages that pages from $\Gamma$ point to
Link Filtering

• Navigational links: serve the purpose of moving within a site (or to related sites)
  • www.espn.com → www.espn.com/nba
  • www.yahoo.com → www.yahoo.it
  • www.espn.com → www.msn.com

• Filter out navigational links
  – same domain name
  – same IP address
How do we rank the pages in seed set \( S \)?

• In degree?

• Intuition

• Problems
Hubs and Authorities [K98]

• Authority is not necessarily transferred directly between authorities

• Pages have double identity
  – hub identity
  – authority identity

• Good hubs point to good authorities

• Good authorities are pointed by good hubs
**HITS Algorithm**

- Initialize all weights to 1.
- Repeat until convergence
  - $O$ operation: hubs collect the weight of the authorities
    \[ h_i = \sum_{j:i\rightarrow j} a_j \]
  - $I$ operation: authorities collect the weight of the hubs
    \[ a_i = \sum_{j:j\rightarrow i} h_j \]
  - Normalize weights under some norm
HITS and eigenvectors

- The HITS algorithm is a power-method eigenvector computation
  - in vector terms \( a^t = A^T h^{t-1} \) and \( h^t = A a^{t-1} \)
  - so \( a^t = A^T A a^{t-1} \) and \( h^t = A A^T h^{t-1} \)
  - The authority weight vector \( a \) is the eigenvector of \( A^T A \) and the hub weight vector \( h \) is the eigenvector of \( A A^T \)
  - Why do we need normalization?

- The vectors \( a \) and \( h \) are singular vectors of the matrix \( A \)
Singular Value Decomposition

\[ A = U \Sigma V^T = [\tilde{u}_1 \quad \tilde{u}_2 \quad \ldots \quad \tilde{u}_r] \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_r \end{bmatrix} \begin{bmatrix} \tilde{v}_1 \\ \tilde{v}_2 \\ \vdots \\ \tilde{v}_r \end{bmatrix} \]

- \( r \): rank of matrix \( A \)
- \( \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \): singular values (square roots of eig-vals \( A^T A \), \( A A^T \))
- \( \tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_r \): left singular vectors (eig-vectors of \( A A^T \))
- \( \tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_r \): right singular vectors (eig-vectors of \( A^T A \))
- \[ A = \sigma_1 \tilde{u}_1 \tilde{v}_1^T + \sigma_2 \tilde{u}_2 \tilde{v}_2^T + \ldots + \sigma_r \tilde{u}_r \tilde{v}_r^T \]
Singular Value Decomposition

- **Linear trend** $\boldsymbol{v}$ in matrix $\mathbf{A}$:
  - the tendency of the row vectors of $\mathbf{A}$ to align with vector $\boldsymbol{v}$
  - strength of the linear trend: $\mathbf{A}\boldsymbol{v}$
- SVD discovers the linear trends in the data
- $\boldsymbol{u}_i, \boldsymbol{v}_i$: the $i$-th strongest linear trends
- $\sigma_i$: the strength of the $i$-th strongest linear trend

- HITS discovers the **strongest linear trend** in the authority space
HITS and the TKC effect

- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect
HITS and the TKC effect

- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect
HITS and the TKC effect

- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect
HITS and the TKC effect

- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect
HITS and the TKC effect

• The HITS algorithm favors the most dense community of hubs and authorities
  – Tightly Knit Community (TKC) effect
HITS and the TKC effect

• The HITS algorithm favors the most dense community of hubs and authorities
  – Tightly Knit Community (TKC) effect
HITS and the TKC effect

• The HITS algorithm favors the most dense community of hubs and authorities
  – Tightly Knit Community (TKC) effect

**Weight of node p is proportional to the number of \((BF)^n\) paths that leave node p**

After n iterations
HITS and the TKC effect

• The HITS algorithm favors the most dense community of hubs and authorities
  – Tightly Knit Community (TKC) effect

after normalization with the max element as $n \to \infty$
Query-independent LAR

- Have an a-priori ordering of the web pages

- $Q$: Set of pages that contain the keywords in the query $q$

- Present the pages in $Q$ ordered according to order $\pi$

- What are the advantages of such an approach?
InDegree algorithm

• Rank pages according to in-degree
  \[ w_i = |B(i)| \]
PageRank algorithm [BP98]

- **Good** authorities should be pointed by **good** authorities
- Random walk on the web graph
  - pick a page at random
  - with probability 1- $\alpha$ jump to a random page
  - with probability $\alpha$ follow a random outgoing link
- Rank according to the stationary distribution

\[
PR(p) = \alpha \sum_{q \rightarrow p} \frac{PR(q)}{|F(q)|} + (1 - \alpha) \frac{1}{n}
\]
Markov chains

• A Markov chain describes a discrete time stochastic process over a set of states

\[ S = \{s_1, s_2, \ldots, s_n\} \]

according to a transition probability matrix

\[ P = \{P_{ij}\} \]

– \( P_{ij} = \) probability of moving to state \( j \) when at state \( i \)
  • \( \sum_j P_{ij} = 1 \) (stochastic matrix)

• **Memorylessness property:** The next state of the chain depends only at the current state and not on the past of the process (first order MC)
  – higher order MCs are also possible
Random walks

• Random walks on graphs correspond to Markov Chains
  – The set of states $S$ is the set of nodes of the graph $G$
  – The \textit{transition probability matrix} is the probability that we follow an edge from one node to another
An example

\[ A = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 \\
\end{bmatrix} \]

\[ P = \begin{bmatrix}
0 & 1/2 & 1/2 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 \\
1/3 & 1/3 & 1/3 & 0 & 0 \\
1/2 & 0 & 0 & 0 & 1/2 \\
\end{bmatrix} \]
State probability vector

• The vector $\mathbf{q}^t = (q^t_1, q^t_2, \ldots, q^t_n)$ that stores the probability of being at state $i$ at time $t$
  $- q^0_i =$ the probability of starting from state $i$

$$\mathbf{q}^t = \mathbf{q}^{t-1} \mathbf{P}$$
An example

\[ P = \begin{bmatrix}
0 & 1/2 & 1/2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
1/3 & 1/3 & 1/3 & 0 & 0 \\
1/2 & 0 & 0 & 1/2 & 0 \\
\end{bmatrix} \]

\[
q^{t+1}_1 = \frac{1}{3} q^t_4 + \frac{1}{2} q^t_5 \\
q^{t+1}_2 = \frac{1}{2} q^t_1 + q^t_3 + \frac{1}{3} q^t_4 \\
q^{t+1}_3 = \frac{1}{2} q^t_1 + \frac{1}{3} q^t_4 \\
q^{t+1}_4 = \frac{1}{2} q^t_5 \\
q^{t+1}_5 = q^t_2
\]
Stationary distribution

• A stationary distribution for a MC with transition matrix $P$, is a probability distribution $\pi$, such that $\pi = \pi P$

• A MC has a unique stationary distribution if
  – it is irreducible
    • the underlying graph is strongly connected
  – it is aperiodic
    • for random walks, the underlying graph is not bipartite

• The probability $\pi_i$ is the fraction of times that we visited state $i$ as $t \to \infty$

• The stationary distribution is an eigenvector of matrix $P$
  – the principal left eigenvector of $P$ – stochastic matrices have maximum eigenvalue 1
Computing the stationary distribution

• The Power Method
  – Initialize to some distribution $q^0$
  – Iteratively compute $q^t = q^{t-1}P$
  – After enough iterations $q^t \approx \pi$
  – Power method because it computes $q^t = q^0P^t$

• Why does it converge?
  – follows from the fact that any vector can be written as a linear combination of the eigenvectors
    • $q^0 = v_1 + c_2v_2 + \ldots + c_nv_n$

• Rate of convergence
  – determined by $\lambda_{2}^t$
The PageRank random walk

• Vanilla random walk
  – make the adjacency matrix stochastic and run a random walk

\[ P = \begin{bmatrix}
0 & 1/2 & 1/2 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 \\
1/3 & 1/3 & 1/3 & 0 & 0 \\
1/2 & 0 & 0 & 1/2 & 0
\end{bmatrix} \]
The PageRank random walk

• What about sink nodes?
  – what happens when the random walk moves to a node without any outgoing inks?

\[
P = \begin{bmatrix}
0 & 1/2 & 1/2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
1/3 & 1/3 & 1/3 & 0 & 0 \\
1/2 & 0 & 0 & 1/2 & 0 \\
\end{bmatrix}
\]
The PageRank random walk

• Replace these row vectors with a vector \( \mathbf{v} \)
  – typically, the uniform vector

\[
P' = \begin{bmatrix}
0 & 1/2 & 1/2 & 0 & 0 \\
1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\
0 & 1 & 0 & 0 & 0 \\
1/3 & 1/3 & 1/3 & 0 & 0 \\
1/2 & 0 & 0 & 1/2 & 0 \\
\end{bmatrix}
\]

\( P' = P + dv^T \)

\( d = \begin{cases} 
1 & \text{if } i \text{ is sink} \\
0 & \text{otherwise} 
\end{cases} \)
The PageRank random walk

- How do we guarantee irreducibility?
  - add a random jump to vector \( v \) with prob \( \alpha \)
    - typically, to a uniform vector

\[
P'' = \alpha \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \end{bmatrix} + (1 - \alpha) \begin{bmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{bmatrix}
\]

\( P'' = \alpha P' + (1-\alpha)uv^T \), where \( u \) is the vector of all 1s
Effects of random jump

- Guarantees irreducibility
- Motivated by the concept of random surfer
- Offers additional flexibility
  - personalization
  - anti-spam
- Controls the rate of convergence
  - the second eigenvalue of matrix $P''$ is $\alpha$
A PageRank algorithm

• Performing vanilla power method is now too expensive – the matrix is not sparse

\[
q^0 = v \\
q^t = (P')^T q^{t-1} \\
t = t + 1 \\
\delta = \|q^t - q^{t-1}\| \\
\text{repeat} \\
\text{until } \delta < \varepsilon
\]

Efficient computation of \( y = (P'')^T x \)

\[
y = \alpha P^T x \\
\beta = \|x\|_1 - \|y\|_1 \\
y = y + \beta v
\]
Random walks on undirected graphs

• In the stationary distribution of a random walk on an undirected graph, the probability of being at node $i$ is proportional to the (weighted) degree of the vertex

• Random walks on undirected graphs are not “interesting”
Research on PageRank

• Specialized PageRank
  – personalization [BP98]
    • instead of picking a node uniformly at random favor specific nodes that are related to the user
  – topic sensitive PageRank [H02]
    • compute many PageRank vectors, one for each topic
    • estimate relevance of query with each topic
    • produce final PageRank as a weighted combination

• Updating PageRank [Chien et al 2002]

• Fast computation of PageRank
  – numerical analysis tricks
  – node aggregation techniques
  – dealing with the “Web frontier”
Previous work

• The problem of identifying the most important nodes in a network has been studied before in social networks and bibliometrics

• The idea is similar
  – A link from node $p$ to node $q$ denotes endorsement
  – mine the network at hand
  – assign an *centrality/importance/standing value* to every node
Social network analysis

• Evaluate the centrality of individuals in social networks
  – degree centrality
    • the (weighted) degree of a node
  – distance centrality
    • the average (weighted) distance of a node to the rest in the graph
      \[
      D_c(v) = \frac{1}{\sum_{u \neq v} d(v, u)}
      \]
  – betweenness centrality
    • the average number of (weighted) shortest paths that use node \( v \)
      \[
      B_c(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}
      \]
Counting paths – Katz 53

• The importance of a node is measured by the weighted sum of paths that lead to this node
• \( A^m[i,j] \) = number of paths of length \( m \) from \( i \) to \( j \)
• Compute
  \[
P = bA + b^2A^2 + \cdots + b^mA^m + \cdots = (I - bA)^{-1} - I
  \]
  converges when \( b < \lambda_1(A) \)
• Rank nodes according to the column sums of the matrix \( P \)
Bibliometrics

• Impact factor (E. Garfield 72)
  – counts the number of citations received for papers of the journal in the previous two years

• Pinsky-Narin 76
  – perform a random walk on the set of journals
  – $P_{ij} =$ the fraction of citations from journal i that are directed to journal j