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 relevancy 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

IN  Root Set  OUT
Query dependent input
Query dependent input
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

\[ A = U \Sigma V^T \]

\[ U = [\vec{u}_1, \ldots, \vec{u}_r] \quad V = [\vec{v}_1 \vec{v}_2 \ldots \vec{v}_r] \]

\[ \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r) \]

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

• **Linear trend** $\mathbf{v}$ in matrix $\mathbf{A}$:
  - the tendency of the row vectors of $\mathbf{A}$ to align with vector $\mathbf{v}$
  - strength of the linear trend: $\mathbf{A}\mathbf{v}$

• SVD discovers the linear trends in the data

• $\mathbf{u}_i, \mathbf{v}_i$: 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

![Diagram of HITS algorithm]
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
  - $w_i = |B(i)|$

1. Red Page
2. Yellow Page
3. Blue Page
4. Purple Page
5. Green Page
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

$$PR(p) = \alpha \sum_{q \rightarrow p} \frac{PR(q)}{|F(q)|} + (1 - \alpha) \frac{1}{n}$$

1. Red Page
2. Purple Page
3. Yellow Page
4. Blue Page
5. Green Page
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 transition probability matrix is the probability that we follow an edge from one node to another
An example

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

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

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

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

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

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

$$P = \begin{bmatrix}
0 & 1/2 & 1/2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 \\
1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\
1/2 & 0 & 0 & 1/2 & 0 & 0
\end{bmatrix}$$
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 $v$ – typically, the uniform vector

\[
P' = P + dv^T
\]

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

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

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

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

\[
P'' = \alpha P' + (1 - \alpha)uv^T, \text{ where } u \text{ is the vector of all 1s}
\]
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

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

Efficient computation of

\[
q^t = (P'')^T q^{t-1}
\]

\[
q^t = aP'^T q^{t-1} \\
\beta = ||q^{t-1}||_1 - ||q^t||_1 \\
q^t = q^t + \beta v
\]
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”