Network Centrality
Part 2

Saptarshi Ghosh
Department of CSE, IIT Kharagpur
Social Computing course, CS60017

Includes material borrowed from various online sources, including slides by Lada Adamic and slides from University of Ioannina
CENTRALITY IN LARGE DIRECTED GRAPHS (WEB GRAPH)
Requirements for Web search

- Results of Web search need to consider
  - Relevance to query
  - Importance / authoritativeness
  - Location / time of query
  - Recency of page
  - ... and many others

- Initial days of the Web: only relevance to query was used to rank webpages
  - Ranking algorithms easily spammed by manipulating the text on spam webpages
Need to consider authoritiveness

- Importance / authoritiveness – centrality on the Web graph (webpages are nodes, hyperlinks are directed edges)

- An edge from node \( p \) to node \( q \) denotes endorsement
  - Node \( p \) endorses/recommends/confirm the authority/centrality/importance of node \( q \)
  - May not be always true (e.g., all pages on a website linking to the Copyright page) but mostly true
  - Use the graph of recommendations to assign an authority value to every node
Hypothesis 1: A hyperlink between pages denotes a conferral of authority (quality signal)

Hypothesis 2: The text in the anchor of the hyperlink on page A describes the target page B
How to compute node centrality on Web?

- First attempt: indegree of webpages used to rank pages according to importance
  - Easily gamed by spammers creating their own webpages

- Subsequent better algorithms: HITS and PageRank
HITS ALGORITHM
HITS algorithm

- Hyperlink-Induced Topic Search, by Kleinberg

- Two types of important pages on the Web
  - Authority: has authoritative content on a topic
  - Hub: pages which link to many authoritative pages, e.g., a directory or catalog
  - A good hub is one which links to many good authorities
  - A good authority node is one which is pointed to by many good hubs
The hope

Mobile telecom companies

Hubs

Alice

Bob

AT&T

ITIM

O2

Authorities

Sec. 21.3
HITS

- HITS computes two scores for each page $p$
  - Authority score: sum of hub scores of all pages which point to $p$
  - Hub score: sum of authority scores of all pages which $p$ points to

- Iterative algorithm
  - The definitions of hubs and authorities are “circular” in nature
  - A series of iterations run, until the scores of all pages converge
HITS run on a query-dependent sub-graph

- Meant to run on a (sub)set of pages that are relevant to a given query
  - Top N pages relevant to query retrieved based on content → called the root set
  - Add to the root set all pages that are linked from it or that links to it → base set
  - Sub-graph of all nodes in base set → focused sub-graph
HITS run on a query-dependent sub-graph

- Why is the root set not sufficient?

- Motivation of building base set
  - A good authority page may not contain the query term
  - Hubs describe authorities through the anchor text / text surrounding hyperlinks
Visualization: hubs & authorities
**HITS Algorithm**

Find focused sub-graph $G$ of pages relevant to given query
for each page $p$ in $G$:

- $p.auth \leftarrow 1$, $p.hub \leftarrow 1$

do until convergence

- for each page $p$ in $G$
  - $p.hub \leftarrow \Sigma r.auth$ for all pages $r$ which $p$ links to
  - $p.auth \leftarrow \Sigma q.hub$ for all pages $q$ which link to $p$

Normalize hub and auth scores for all pages
Check convergence of scores

Output pages with highest authority scores and hub scores
Normalization of scores

- Scores need to be normalized after each iteration

- Different normalization schemes proposed
  - Normalize so that score vectors sum to 1
  - Normalization factor $F$: square root of sum of squares of current scores of all pages; divide score of each page by $F$ at the end of each iteration
Checking for convergence

- Various convergence criteria used
  - Fixed number of iterations
  - Iterate until scores do not change appreciably from one iteration to the next (compute difference of score vectors from previous and current iterations)
  - Iterate until rankings of pages do not change
HITS Algorithm (again)

Find focused sub-graph $G$ of pages relevant to given query
for each page $p$ in $G$:
  $p$.auth $\leftarrow$ 1, $p$.hub $\leftarrow$ 1

do until convergence
  for each page $p$ in $G$
    $p$.hub $\leftarrow$ $\Sigma$ $r$.auth for all pages $r$ which $p$ links to
    $p$.auth $\leftarrow$ $\Sigma$ $q$.hub for all pages $q$ which link to $p$

Normalize hub and auth scores for all pages
Check convergence of scores

Output pages with highest authority scores and hub scores
Matrix version of HITS

- Matrices / vectors
  - A: adjacency matrix of web graph. (u, v)-th element is 1 if page u links to page v
  - h: vector of hub scores of all pages
  - a: vector of authority scores of all pages

- \( h \leftarrow A.a \)
- \( a \leftarrow A^T.h \)
HITS – summary

- HITS is guaranteed to converge

- Reasonably efficient for large Web-scale graphs, since updates involve local operations only

- Still, not very popularly used. Why?
HITS – summary

- HITS is guaranteed to converge

- Reasonably efficient for large Web-scale graphs, since updates involve local operations only

- Still, not very popularly used. Why?
  - Easy for a spam page to obtain high hub score (just by following many authorities)
  - Hubs often transit to authorities
  - Search engines themselves become hubs
PAGERANK ALGORITHM
PageRank

By Larry Page and Sergey Brin

Problem in measuring importance by indegree
- Not all in-links are same
- How important are those pages which link to page \( p \)?

PageRank of a page
- A measure of the ‘authority value’ of the page
- Independent of query
- One of many factors used by Google to rank pages
Idea of PageRank

- Good authorities should be pointed to by other good authorities
  - $PR_v$ of page (node) $v$ is a function of the sum of PRs of all those pages which point to $v$
- Each node $u$ distributes its authority value equally among all those nodes to which $u$ points
  - If page $u$ links to 4 pages, $u$ contributes $PR_u/4$ to the PR of each of those 4 pages

$$PR_v = \sum_{u \rightarrow v} \frac{1}{d_{out}(u)} PR_u$$
Equations for PR (here $w_v \sim PR_v$)

\[
\begin{align*}
  w_1 &= \frac{1}{3} w_4 + \frac{1}{2} w_5 \\
  w_2 &= \frac{1}{2} w_1 + w_3 + \frac{1}{3} w_4 \\
  w_3 &= \frac{1}{2} w_1 + \frac{1}{3} w_4 \\
  w_4 &= \frac{1}{2} w_5 \\
  w_5 &= w_2
\end{align*}
\]

Iterative algorithm used to solve such a system of equations (multiple iterations until convergence)

\[
w_v = \sum_{u \rightarrow v} \frac{1}{d_{out}(u)} w_u
\]
PageRank computation

/* initialization */
for all nodes $u$ in $G$: $d(u) \leftarrow 1/N$, where $N = \#\text{nodes}$
for all nodes $u$ in $G$: $PR(u) \leftarrow d(u)$

/* iteration */
do until $PR$ vector converges
   for all nodes $u$ in $G$
      for all nodes $v$ that links to $u$
         $t = \sum PR(v) / \text{out-degree}(v)$
         $PR(u) \leftarrow \alpha \cdot t + (1 - \alpha) \cdot d(u)$
      normalize scores
      check for convergence
   end
end

$\alpha$ to be explained later
Theoretical basis of PageRank

- Random walks on a graph
  - Start from a node chosen uniformly at random with prob \( \frac{1}{N} \)
  - From the node you are in, pick one of the outgoing links uniformly at random
  - Move to the destination node of the chosen link
  - Repeat

- The **Random Surfer** model
  - Users wander on the web, following links
  - Nodes visited more frequently in this random walk are web-pages with higher PR
Example

- Step 0
Example

- Step 0
Example

- Step 1
Example

- Step 1
Example

- Step 2
Example

- Step 2
Example

- Step 3

![Diagram](image_url)
Example

- Step 3
Example

- Step 4...
Equations for Random Walk

Question: what is the probability $p_i^t$ of being at node $i$ after $t$ steps?

- $p_1^0 = \frac{1}{5}$
- $p_2^0 = \frac{1}{5}$
- $p_3^0 = \frac{1}{5}$
- $p_4^0 = \frac{1}{5}$
- $p_5^0 = \frac{1}{5}$

- $p_1^t = \frac{1}{3}p_4^{t-1} + \frac{1}{2}p_5^{t-1}$
- $p_2^t = \frac{1}{2}p_1^{t-1} + p_3^{t-1} + \frac{1}{3}p_4^{t-1}$
- $p_3^t = \frac{1}{2}p_1^{t-1} + \frac{1}{3}p_4^{t-1}$
- $p_4^t = \frac{1}{2}p_5^{t-1}$
- $p_5^t = p_2^{t-1}$

The equations are the same as those for the PageRank computation.
Equations for PR (again)

\[
\begin{align*}
  w_1 &= \frac{1}{3} w_4 + \frac{1}{2} w_5 \\
  w_2 &= \frac{1}{2} w_1 + w_3 + \frac{1}{3} w_4 \\
  w_3 &= \frac{1}{2} w_1 + \frac{1}{3} w_4 \\
  w_4 &= \frac{1}{2} w_5 \\
  w_5 &= w_2
\end{align*}
\]

\[
w_v = \sum_{u \rightarrow v} \frac{1}{d_{out}(u)} w_u
\]

Iterative algorithm used to solve such a system of equations (multiple iterations until convergence)
Theoretical basis of PageRank

- The random walk defines a Markov chain
  - A discrete time stochastic process following Markov property (next state depends only on current state)
  - $N$ states corresponding to the $N$ nodes; chain is at one of the states at any given time-step
  - $N \times N$ transition probability matrix $P$: $P_{ij}$ is the probability that state at next time-step is $j$, given current state is $i$

$$\forall i, j, P_{ij} \in [0, 1] \quad \forall i, \sum_{j=1}^{N} P_{ij} = 1.$$
An example
An example

- $P$ is a stochastic matrix
  - Every element is in $[0, 1]$
  - Sum of every row is 1
  - Largest eigenvalue is 1
  - Has a principal left eigenvector corresponding to its largest eigenvalue
Another example

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

\[ P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix} \]
Transition matrix for random surfer

- How to derive the transition matrix for the random surfer on the Web graph?

- Adjacency matrix of Web graph
  - $A_{ij} = 1$ if there is a hyperlink from page $i$ to page $j$
  - $A_{ij} = 0$ otherwise

- Derive transition matrix $P$ of Markov chain from $A$
Some practical challenges

- Web graph (or any graph) can have
  - Dead-ends or sink nodes – nodes with no out-edges

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

- Web graph (or any graph) can have
  - Loops
Transition matrix for random surfer

- Derive transition matrix $P$ of Markov chain from $A$
  - If a row of $A$ has no 1’s, replace each element by $1/N$
  - For all other rows: divide each 1 by the number of 1’s in the row
  - Multiply the resulting matrix by $\alpha$
  - Add $(1-\alpha)/N$ to every entry of the resulting matrix
Dealing with sink nodes

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

As if synthetic edges are inserted from the sink node to every other node in the graph

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

- As if synthetic edges are inserted to enable jump from any node to any other node in the graph
- Teleportation: jump to any random node with probability 1/N

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

- Convergence of PageRank is guaranteed only if
  - The transition probability matrix $P$ is irreducible, i.e., all transitions have a non-zero probability
  - In other words, if the graph (on which random surfing is taking place) is strongly connected

To ensure convergence

- To nodes with out-degree 0, add an outgoing edge to every node
- Damp the walk by factor $\alpha$, by adding a complete set of outgoing edges, with weight $(1-\alpha)/N$, to all nodes
Transition matrix for random surfer: Recap

- Derive transition matrix $P$ of Markov chain from $A$
  - If a row of $A$ has no 1’s, replace each element by $1/N$
  - For all other rows: divide each 1 by the number of 1’s in the row
  - Multiply the resulting matrix by $\alpha$
  - Add $(1-\alpha)/N$ to every entry of the resulting matrix
Given $P$, how to compute PageRank?

- Vector $x$ (dimension $N$): probability distribution of surfer’s position at any time
  - At $t = 0$: one entry in $x$ is 1, rest are 0
  - At $t = 1$: $xP$
  - At $t = 2$: $(xP)P = xP^2$
  - ...

- Steady-state $x = \Pi$ gives the PageRank scores
  - At steady-state: $\Pi P = \Pi$
  - In other words, at steady state: $\Pi P = 1.\Pi$
**Given \( P \), how to compute PageRank?**

- Vector \( x \) (dimension \( N \)): probability distribution of surfer’s position at any time
  - At \( t = 0 \): one entry in \( x \) is 1, rest are 0
  - At \( t = 1 \): \( xP \)
  - At \( t = 2 \): \( (xP)P = xP^2 \)
  - ...

- Steady-state \( x = \mathbb{1} \) gives the PageRank scores

- PageRank scores obtained as the principal left eigenvector of \( P \) (corresponding to eigenvalue 1)
PageRank computation

- Need to compute principal left eigenvector of a stochastic matrix

- Several numerical methods, e.g., power iteration

- Difficult to compute for matrices of the size of the Web graph; iterative method (already discussed) can be more efficient
Theoretical basis of PageRank: Recap

- Random surfer model
  - Start at a node, execute a random walk on Web graph
  - At each step, proceed from current node $u$ to a randomly chosen node that $u$ links to
  - Teleport: jump to any random node with probability $1/N$
  - At a node with no outgoing links, teleport
  - At a node that has outgoing links
    - Follow standard random walk with probability $\alpha$ where $0<\alpha<1$
    - Teleport with probability $(1-\alpha)$

- Nodes visited more frequently in this random walk are web-pages with higher PR
PageRank computation: Recap

/* initialization */
for all nodes \( u \) in \( G \): \( d(u) \leftarrow 1/N \), where \( N = \) #nodes
for all nodes \( u \) in \( G \): \( PR(u) \leftarrow d(u) \)

/* iteration */
do until \( PR \) vector converges
   for all nodes \( u \) in \( G \)
      for all nodes \( \nu \) that links to \( u \)
         \( t = \Sigma PR(\nu) / \) out-degree\((\nu) \)
         \( PR(u) \leftarrow \alpha * t + (1 - \alpha) * d(u) \)
      normalize scores
      check for convergence
   end
end
Practical challenges

- All links $u \to v$ do not signify a vote for $v$
  - E.g., links to a copyright page from all pages in a website

- Attempts to spam PageRank: link spam farms or link farms
  - A target page (whose PR the spammer wants to boost)
  - A number of boosting pages, which link to the target page, link to each other and also to external pages
  - Hijacked links – links accumulated from pages outside the link farm
Example link farm

Figure 2: A web of good (white) and bad (black) nodes.
VARIATIONS OF PAGERANK
PageRank computation

/* initialization */
for all nodes u in G: $d(u) \leftarrow 1/N$, where $N = \#nodes$
for all nodes u in G: $PR(u) \leftarrow d(u)$

/* iteration */
do until $PR$ vector converges
  for all nodes $u$ in G
    for all nodes $v$ that links to $u$
      $t = \Sigma PR(v) / \text{out-degree}(v)$
      $PR(u) \leftarrow \alpha \times t + (1 - \alpha) \times d(u)$
    normalize scores
  check for convergence
end
Biased PageRank

Instead of using the uniform vector \( d(u) \leftarrow 1/N \) for all nodes \( u \), use a non-uniform preference vector:

\[
d(u) = \begin{cases} 
1 / |S|, & \text{for all } u \in S \\
0, & \text{otherwise}
\end{cases}
\]

Implication for random surfer:

- With probability \( \alpha \), follow standard random walk
- With probability \( (1-\alpha) \), teleport to a node in \( S \), where the particular node in \( S \) is chosen randomly
Biased PageRank

- Instead of using the uniform vector $d(u) \leftarrow 1/N$ for all nodes $u$, use a non-uniform preference vector:
  \[
  d(u) = \begin{cases} 
  1 / |S|, & \text{for all } u \in S \\
  0, & \text{otherwise}
  \end{cases}
  \]

- Implication for random surfer:
  - With probability $\alpha$, follow standard random walk
  - With probability $(1-\alpha)$, teleport to a node in $S$, where the particular node in $S$ is chosen randomly

- Bias the ranks towards nodes that are closer to nodes with a larger value in the preference vector
Topic-sensitive PageRank [Haveliwala, WWW 2002]

- Webpages are classified into various topics (16 Open Directory Project high-level categories)
- Computes PageRank for a particular topic of interest

For category $c_j$

- $T_j$ is the set of websites for category $c_i$
- Modified teleportation function

$$v_{ji} = \begin{cases} \frac{1}{|T_j|} & i \in T_j, \\ 0 & i \notin T_j. \end{cases}$$
TrustRank [Gyongyi, VLDB 2004]

- Aims to rank trusted pages higher, and push untrusted pages down in the rankings
- Assumes
  - A way of knowing trusted nodes: oracle
  - Trusted (good) nodes will only link to other good nodes but this assumption is violated in the real Web
  - Bad nodes will link to other bad nodes and good nodes

- Run PageRank by biasing the preference vector towards a set of trusted nodes
Figure 10: Bad sites in PageRank and TrustRank buckets.