Social Networks and Ranking

Generalized Social Networks

- Represent relationship between entities
  - paper cites paper
  - html page links to html page
  - A supervises B
  - A and B are friends
  - papers share an author
  - A and B are co-workers

Hypertext

- document or part of document links to other parts or other documents
  - construct documents of interrelated pieces
  - relate documents to each other
- pre-dates Web
- Web “killer app.”

How use links to improve information search?

- use structure to compute score for ranking
- include more objects to rank
  - redefines “satisfying” of query?
- add to the content of a document

- can deal with objects of mixed types
  - images, PDF, …
Scoring using structure

- Ideas
  1. Link to object suggests it valuable object
  2. Distance between objects in graph represents degree of relatedness reachable by all in 2 links

Pursuing linking and value

- Intuition: when Web page points to another Web page, it confers status/authority/popularity to that page
- Find a measure that captures intuition
- Not just web linking
  - Citations in books, articles
  - Others?

Indegree

- Indegree = number of links into a node
- Most obvious idea: higher indegree => better node
- Doesn’t work well
- Need some feedback in system
- Leads us to Page and Brin’s PageRank

PageRank

- Algorithm that gave Google the leap in quality
  - Link structure centerpiece of scoring
- Framework
  - Given a directed graph with \( n \) nodes
  - Assign each node a score that represents its importance in structure: PageRank: \( pr(node) \)
Conferring importance

Core ideas:

- A node should **confer** some of its importance **to the nodes to which it points**
  - If a node is important, the nodes it links to should be important
- A node should **not transfer more importance than it has**

**Attempt 1**

Refer to nodes by numbers 1, … , n (arbitrary numbering)

Let \( t_i \) denote the number of edges out of node \( i \) (outdegree)

Node i transfers \( 1/t_i \) of its importance on each edge out of it

Define

\[
pr_{\text{new}}(k) = \sum_{i \text{ with edge from } i \text{ to } k} (pr(i) / t_i)
\]

Iterate until converges

**Problems**

- Sinks (nodes with no edges out)
- Cyclic behavior

**Normalized?**

- Would like \( \sum_{1 \leq k \leq n} (pr(k)) = 1 \)
- Consider \( \sum_{1 \leq k \leq n} (pr_{\text{new}}(k)) \)

\[
\begin{align*}
&= \sum_{1 \leq k \leq n} \left( \frac{\alpha}{n} + (1-\alpha) \sum_{i \text{ with edge from } i \text{ to } k} (pr(i) / t_i) \right) \\
&= \sum_{1 \leq k \leq n} \left( \frac{\alpha}{n} \right) + \sum_{1 \leq k \leq n} \left( (1-\alpha) \sum_{i \text{ with edge from } i \text{ to } k} (pr(i) / t_i) \right) \tag{1}
\end{align*}
\]

\[
\begin{align*}
&= \alpha \sum_{1 \leq k \leq n} \sum_{i \text{ with edge from } i \text{ to } k} (pr(i) / t_i) \tag{2}
\end{align*}
\]

\[
\begin{align*}
&= \alpha \sum_{1 \leq i \leq n} \sum_{k \text{ with edge from } i \text{ to } k} \left( \frac{pr(i)}{t_i} \right) \tag{3}
\end{align*}
\]

\[
\begin{align*}
&= \frac{\alpha}{1-t} \sum_{1 \leq i \leq n} \sum_{k \text{ with edge from } i \text{ to } k} (pr(i) / t_i) \tag{4}
\end{align*}
\]

\[
\begin{align*}
&= \frac{\alpha}{1-t} \sum_{1 \leq i \leq n} \sum_{k \text{ with edge from } i \text{ to } k} (pr(i) / t_i) \tag{5}
\end{align*}
\]

**Random walk model**

- Attempt 1 gives movement from node to linked neighbor with probability \( 1/\text{outdegree} \)
- Add random jump to any node

\[
pr_{\text{new}}(k) = \frac{\alpha}{n} + (1-\alpha) \sum_{i \text{ with edge from } i \text{ to } k} (pr(i) / t_i)
\]

- \( \alpha \) parameter chosen empirically

- Break cycles
- Escape from sinks
Problem for desired normalization

- Have
  \[ \sum_{1 \leq k \leq n} (pr_{new}(k)) = \alpha + (1-\alpha) \sum_{i \text{ with edge from } i} pr(i) \]
- Missing \( pr(i) \) for nodes with no edges from them
  - sinks!
- Solution: add \( n \) edges out of every sink
  - Edge to every node including self
  - Gives \( \frac{1}{n} \) contribution to every node

Gives desired normalization:
If \( \sum_{1 \leq k \leq n} (pr_{initial}(k)) = 1 \)
then \( \sum_{1 \leq k \leq n} (pr(k)) = 1 \)

Matrix formulation

- Let \( E \) be the \( n \) by \( n \) adjacency matrix
  - \( E(i,k) = 1 \) if there is an edge from node \( i \) to node \( k \)
  - \( = 0 \) otherwise
- Define new matrix \( L \):
  - For each row \( i \) of \( E \) (\( 1 \leq i \leq n \))
    - If row \( i \) contains \( t_i > 0 \) ones, \( L(i,k) = \frac{1}{t_i} E(i,k), \ 1 \leq k \leq n \)
    - If row \( i \) contains 0 ones, \( L(i,k) = \frac{1}{n}, \ 1 \leq k \leq n \)
- Vector \( pr \) of PageRank values defined by
  \[ pr = (\alpha/n, \alpha/n, \ldots \alpha/n)^T + (1-\alpha) L^T pr \]
- has a solution representing the steady-state values \( pr(k) \)

Calculation Choices

1. \( pr = M pr : \) Find principle eigenvector of \( M \)
   solves \( n \) simultaneous equations:
   \[ pr(k) = \alpha/n + (1-\alpha) \sum_{i \text{sink}} L(i,k)pr(i) \]
2. Use iterative calculation - power method
   (where we started)
   - Initialize \( pr_{initial}(k) = 1/n \) for each node \( k \)
   - Until converges {
     For each node \( k \)
     \[ pr_{new}(k) = \alpha/n + (1-\alpha) \sum_{i \text{sink}} L(i,k)pr(i) \]
     For each node \( k \)
     \[ pr(k) = pr_{new}(k) \]
Power method

- **Convergence**
  - In practice choose convergence criterion
  - e.g. stop iteration when
    \[ \text{Max}_{k=1}^{n} |(\text{pr}_{\text{new}}(k) - \text{pr}(k))| < \varepsilon \]
  - \( \varepsilon = 10^{-3}, 10^{-4}, 10^{-5} \)

- **Choice \( \alpha \)**
  - No single best value
  - \( 1-\alpha \) determines rate of convergence
    - Second eigenvalue
    - \( \alpha = 0.15 \) common
    - gives \( 10^{-4} \) accuracy in about 60 iterations regardless of size of graph [Chu; Wu]

PageRank Observations

- Can be calculated for *any* directed graph
- Google calculates on entire Web graph
  - query independent scoring
- Huge calculation for Web graph
  - precomputed
  - 1998 Google published:
    - 52 iterations for 322 million links
    - 45 iterations for 161 million links
- PageRank must be combined with query-based scoring for final ranking
  - Many variations
  - What Google exactly does secret
  - Can make some guesses by results

HITS

**Hyperlink Induced Topic Search**

- Second well-known algorithm
- By Jon Kleinberg while at IBM Almaden Research Center
- Same general goal as PageRank
- Distinguishes 2 kinds of nodes
  - Hubs: resource pages
    - Point to many authorities
  - Authorities: good information pages
    - Pointed to by many hubs

Mutual reinforcement

- Authority weight node \( j \): \( a(j) \)
  - Vector of weights \( a \)
- Hub weight node \( j \): \( h(j) \)
  - Vector of weights \( h \)
- Update:
  \[
  a_{\text{new}}(k) = \sum_{i \text{ with edge from } i \text{ to } k} (h(i))
  \]
  \[
  h_{\text{new}}(k) = \sum_{j \text{ with edge from } k \text{ to } j} (a(j))
  \]
Mutual reinforcement

• Authority weight node j: \( a(j) \)
  – Vector of weights \( a \)
• Hub weight node j: \( h(j) \)
  – Vector of weights \( h \)

• Update:
  \[
  a_{\text{new}}(k) = \sum_{i \text{ with edge from } i \text{ to } k} (h(i)) \\
  h_{\text{new}}(k) = \sum_{j \text{ with edge from } k \text{ to } j} (a(j))
  \]

Matrix formulation

Steady state:
\[
\begin{align*}
  a &= E^T h \\
  a &= E^T E a \\
  h &= E a \\
  h &= E E^T h
\end{align*}
\]

Interpretation?

Mutual reinforcement

• Authority weight node j: \( a(j) \)
  – Vector of weights \( a \)
• Hub weight node j: \( h(j) \)
  – Vector of weights \( h \)

• Update:
  \[
  a_{\text{new}}(k) = \sum_{i \text{ with edge from } i \text{ to } k} (h(i)) \\
  h_{\text{new}}(k) = \sum_{j \text{ with edge from } k \text{ to } j} (a(j))
  \]

Look inside

• \( E^T(i,k) \) 1 where \( k \rightarrow i \)
• \( E(k,j) \) 1 where \( k \rightarrow j \)

• Row i of \( E^T \):
  1’s where \( k \rightarrow i \)

• Column j of \( E^T \):
  1’s where \( k \rightarrow j \)

\( E^T E(i,j) \) is number of nodes pointing to both i and j

• \( E(i,k) \) 1 where \( i \rightarrow k \)
• \( E^T(k,j) \) 1 where \( j \rightarrow k \)

• Row i of \( E \):
  1’s where \( i \rightarrow k \)’s

• Column j of \( E^T \):
  1’s where \( j \rightarrow k \)’s

\( E E^T(i,j) \) is number of nodes pointed to by both i and j
Matrix formulation

Steady state:

\[ a = E^T h \]
\[ h = E a \]

Interpretation:

- \( E^T E(i,j) \): number nodes point to both node i and node j
  - “Co-citation”
- \( E E^T(i,j) \): number nodes pointed to by both node i and node j
  - “Bibliographic coupling”

Iterative Calculation

\[ a = h = (1, \ldots, 1)^T \]

While (not converged) {

\[ a_{\text{new}} = E^T h \]
\[ h_{\text{new}} = E a \]
\[ a = a_{\text{new}} / ||a_{\text{new}}|| \] normalize to unit vector
\[ h = h_{\text{new}} / ||h_{\text{new}}|| \] normalize to unit vector

Provable convergence by linear algebra

Use of HITS

original use after find Web pages satisfying query:

1. Retrieve documents satisfy query and rank by term-based techniques
2. Keep top \( c \) documents: root set of nodes
   - \( c \) a chosen constant - tunable
3. Make base set:
   a) Root set
   b) Plus nodes pointed to by nodes of root set
   c) Plus nodes pointing to nodes of root set
4. Make base graph: base set plus edges from Web graph between these nodes
5. Apply HITS to base graph

Results using HITS

- Documents ranked by authority score \( a(\text{doc}) \) and hub score \( h(\text{doc}) \)
  - Authority score primary score for search results
- Heuristics:
  - delete all links between pages in same domain
  - Keep only pre-determined number of pages linking into root set (\( \sim 200 \))
- Findings (original paper):
  - Number iterations in original tests \( \sim 50 \)
  - most authoritative pages do not contain initial query terms
Observations

- HITS can be applied to any directed graph
- Base graph much smaller than Web graph
- Kleinberg identified bad phenomena
  - Topic diffusion: generalizes topic when expand root graph to base graph
    - example: want compilers - generalized to programming

PageRank and HITS

- designed independently around 1997
- indicates time was ripe for this kind of analysis
- lots of embellishments by others

Revisit: How use links in ranking documents?

- use structure to compute score for ranking
  - PageRank, HITS
- include more objects to rank
  - saw in use of HITS

➤ use anchor text (HTML)
  - anchor text labels link
  - include anchor text
    as text of document pointed to

Anchor text

- HTML text:
  All assignments will be made available on
  
- Renders as:
  All assignments will be made available on
  the Piazza course account.
  
- Anchor text:
  "the Piazza course account" is anchor text
Using anchor text

“homework” may not occur in content of doc b

terms in doc b for building index:

- homework: anchor
- problem: title 1
- set: title 2

Summary

- Link analysis
  - a principal component of ranking by modern Web search engines
  - must be combined with content analysis
- Extend document content with link info
  - anchor text
  - text of URLs
    - e.g. princeton.edu, aardvarksportsshop.com
- Expand set of satisfying docs using links
  - less often used

General Framework

- Have set of $n$ features (aka signals) to use in determining ranking score
  - Features depend on query:
    - vector $\Psi(d_i, q)$ of feature values $f_k$ for doc $d_i$, query $q$
      - eg tf.idf score is feature
  - Features are conditioned to be comparable
- Have parameterized function to combine signals
  - simple: linear $\alpha_0 + \sum_{i=1}^n \alpha_i^*(f_i)$
  - $\alpha_i$ are adjustable weights - how choose?
    - intuition
    - experimentation
    - machine learning

Ranking documents w.r.t. query

query

anchor text

link analysis + doc. features

personal information

historic information

words in doc + word features

scores of documents for query - use to rank

Secret recipe
Machine Learning

Many possibilities – overview of one

**Ordinal Regression Model**

- Goal: get comparison of docs correct
- capture goal
  - Let $\mathbf{\omega}$ represent vector $[\alpha_1, \ldots, \alpha_n]$
  - want $\mathbf{\omega}^T \Psi(d_i, q) - \mathbf{\omega}^T \Psi(d_j, q) > 0$ if and only if $d_i$ more relevant than $d_j$ for query $q$
  - find $\mathbf{\omega}$ that works
- techniques **train** on known correct data:
  - humans rank a set of documents for various queries

**Ranking documents w.r.t. query**

- **Query**
- **Secret recipe**
- **link analysis + doc. features**
- **personal information**
- **historic information**
- **anchor text**
- **words in doc + word features**
- **scores of documents for query - use to rank**