Text Processing on the Web

Week 5
Link Analysis Ranking

The material for these slides are borrowed heavily from the precursor of this course by Tat-Seng Chua, as well as slides from the accompanying recommended texts Baldi et al. and Manning et al.
Recap

• Synonymy and Polysemy affect all standard IR models — not just limited to VSM

• We want to instead model latent topics
  – SVD factors the term-document matrix into orthogonal eigenvectors ("topics"), automatically ranked by salience ("eigenvalue magnitude").
  – LSA does SVD and then drops low order topics to create approximation
  – pLSA does this by taking the unigram LM and injecting a latent variable, k (for k topics)
Outline

• The classics:
  – Page Rank
  – Hubs and Authorities

• Adaptations to the Models
  – Topic Sensitive PageRank
  – SALSA
Citation Networks

• Pioneered by Garfield 1972 to answer questions on impact

• Introduced Impact Factor
  – $C =$ citations to articles in a journal
  – $N =$ total number of articles in a journal

  – Impact Factor = $C/N$
    (Normalized in-degree of a journal)
Query-independent ordering

- How does this translate to the web?
  - Have a graph, not a DAG

- Using link counts as simple measures of prestige
  - number of inlinks (3)
Algorithm

1. Retrieve all pages meeting the text query (say \textit{venture capital}), perhaps by using Boolean model

2. Order these by link popularity

Exercise: How do you spam each of the following heuristics so your page gets a high score?

- score = \# in-links
Link Counts

Min’s Home Page

LKY’s Home Page

Family home page

Min’s sister’s Page

Queen of England’s Page

Linked by 2

Linked by 2

Unimportant pages

Important Pages

www.sg
Definition of PageRank

• The importance of a page is given by the importance of the pages that link to it.

\[ x_i = \sum_{j\in B_i} \frac{1}{N_j} x_j \]

- Importance of page \( i \)
- Pages \( j \) that link to page \( i \)
- Number of outlinks from page \( j \)
Pagerank scoring

• Imagine a browser doing a random walk on web pages:
  – Start at a random page
  – At each step, follow one of the $n$ links on that page, each with $1/n$ probability

• Do this repeatedly. Use the “long-term visit rate” as the page’s score
Markov chains

A Markov chain consists of $n$ states, plus an $n \times n$ transition probability matrix $A$.

- At each step, we are in exactly one of the states.
- For $1 \leq i, k \leq n$, the matrix entry $A_{ik}$ tells us the probability of $k$ being the next state, given we are currently in state $i$.
- **Memorylessness property**: The next state depends only at the current state (first order MC)

\[ A_{ik} > 0 \text{ is OK.} \]
Markov chains

- Clearly, for all $i$, $\sum_{k=1}^{n} A_{ik} = 1$.
- Markov chains are abstractions of random walks

Try this: Calculate the matrix $A_{ik}$ using $1/n$ possibility

\[
A_{ik}:
\begin{array}{ccc}
A & B & C \\
\text{A} & \text{B} & \text{C} \\
\end{array}
\]
Not quite enough

- The web is full of dead ends.
  - What sites have dead ends?
  - Our random walk can get stuck.
Teleporting

- At each step, with probability 10%, teleport to a random web page
- With remaining probability (90%), follow a random link on the page
  - If a dead-end, stay put in this case

\[
\text{rank} = (1 - a)A \times \text{rank} + \alpha \left[ \frac{1}{N} \right] N \times 1
\]
Ergodic Markov chains

- A Markov chain is ergodic if
  - you have a path from any state to any other
  - you can be in any state at every time step, with non-zero probability
  - With teleportation, our Markov chain is ergodic
Markov chains (2\textsuperscript{nd} Try)

Try this: Calculate the matrix $A_{ik}$ using a 10% chance of teleportation.
Probability vectors

• A probability (row) vector \( \mathbf{x} = (x_1, \ldots, x_n) \) tells us where the walk is at any point.

• E.g., (000…1…000) means we’re in state \( i \).

More generally, the vector \( \mathbf{x} = (x_1, \ldots, x_n) \) means the walk is in state \( i \) with probability \( x_i \).

\[
\sum_{i=1}^{n} x_i = 1.
\]
Change in probability vector

• If the probability vector is \( \mathbf{x} = (x_1, \ldots, x_n) \) at this step, what is it at the next step?
• Recall that row \( i \) of the transition prob. Matrix \( \mathbf{A} \) tells us where we go next from state \( i \).
• So from \( \mathbf{x} \), our next state is distributed as \( \mathbf{xA} \).
Pagerank algorithm

• Regardless of where we start, we eventually reach the steady state $a$
  – Start with any distribution (say $x=(10\ldots0)$)
  – After one step, we’re at $xA$
  – After two steps at $xA^2$, then $xA^3$ and so on.
  – “Eventually” means for “large” $k$, $xA^k = a$

• Algorithm: multiply $x$ by increasing powers of $A$ until the product looks stable
Steady State

• For any ergodic Markov chain, there is a unique long-term visit rate for each state
  – Over a long period, we’ll visit each state in proportion to this rate
  – It doesn’t matter where we start
Eigenvector formulation

• The flow equations can be written
  \[ r = Ar \]

• So the rank vector is an eigenvector of the adjacency matrix
  – In fact, it’s the first or principal eigenvector, with corresponding eigenvalue 1
Pagerank summary

• Pre-processing:
  – Given graph of links, build matrix $A$
  – From it compute $a$
  – The pagerank $a_i$ is a scaled number between 0 and 1

• Query processing:
  – Retrieve pages meeting query
  – Rank them by their pagerank
  – Order is query-independent
Hubs and Authorities

- 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.
High-level scheme

• Extract from the web a base set of pages that could be good hubs or authorities.

• From these, identify a small set of top hub and authority pages
  \[\rightarrow\text{iterative algorithm}\]
Base set

1. Given text query (say university), use a text index to get all pages containing university.
   – Call this the root set of pages

2. Add in any page that either:
   – points to a page in the root set, or
   – is pointed to by a page in the root set

3. Call this the base set
Root set

Base set
Assembling the base set

- Root set typically 200-1000 nodes.
- Base set may have up to 5000 nodes.
- How do you find the base set nodes?
  - Follow out-links by parsing root set pages.
  - Get in-links (and out-links) from a connectivity server.
Distilling hubs and authorities

1. Compute, for each page \( x \) in the base set, a **hub score** \( h(x) \) and an **authority score** \( a(x) \).

2. Initialize: for all \( x \), \( h(x) \leftarrow 1; a(x) \leftarrow 1 \);

3. Iteratively update all \( h(x), a(x) \);

4. After iterations:
   - highest \( h() \) scores are hubs
   - highest \( a() \) scores are authorities
Iterative update

- Repeat the following updates, for all x:

\[
h(x) \leftarrow \sum_{y \leftarrow x} a(y)
\]

\[
a(x) \leftarrow \sum_{y \leftarrow x} h(y)
\]

\[
h_t = Aa_{t-1}
\]

\[
a_t = A^T h_{t-1}
\]
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

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

- \( r \): rank of matrix \( A \)
- \( \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \): singular values (square roots of eigenvalues \( A A^T, A^T A \))
- \( \tilde{u}_1, \tilde{u}_2, \cdots, \tilde{u}_r \): left singular vectors (eigenvectors of \( A A^T \))
- \( \tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_r \): right singular vectors (eigenvectors of \( A^T A \))
- \( A = \sigma_1 \tilde{u}_1 \tilde{v}_1^T + \sigma_2 \tilde{u}_2 \tilde{v}_2^T + \cdots + \sigma_r \tilde{u}_r \tilde{v}_r^T \)
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
How many iterations?

• Relative values of scores will converge after a few iterations
• We only require the relative order of the $h()$ and $a()$ scores - not their absolute values
• In practice, ~5 iterations needed
Things to think about

• Use *only* link analysis *after* base set assembled
  – iterative scoring is query-independent
• Iterative computation *after* text index retrieval - significant overhead
Things to think about

• A pagerank score is a global score. Can there be a fusion between H&A (which are query sensitive) and pagerank?

• How does the selection of the base set influence computation of H & As?
• Can we embed the computation of H & A during the standard VS retrieval algorithm?
• How can you update PageRank without recomputing the whole thing from scratch?
• What’s the eigenvector relationship between HITS’ authority and PageRank?
Advanced link structure methods
Topic-Sensitive PageRank

• Basic idea:
  1. Identify topic that might be interesting for the user (e.g. via classification of the query, eval. of context, ...)
  2. Use pre-calculated, topic-sensitive PageRank

• Topic specific PageRank $\text{rank}_{jd}$:

• Now: Topics $c_1$, ..., $c_n$,
  - They used 16 top-level categories from the ODP

• Topic dependent weighting ($1/|T_i|$)

• Advantage: Can be calculated in advance
Offline PageRank Vector Computation

• Play around with Teleportation Rate

\[ \vec{rank} = (1 - \alpha) A \times \vec{rank} + \alpha \left[ \frac{1}{N} \right] N \times 1 \]

• Don’t jump to a random page; jump to a topic page!

\[ \nu_{ij} = \begin{cases} 
\frac{1}{|T_j|} & i \in T_j \\ 
0 & i \notin T_j 
\end{cases} \quad T_j = \text{set of pages relevant to a topic} \]
Run-time TSPageRank (cont.)

- Question: Which one to select during run time?
- Idea: Classification of query $q$ given by the user
- Extension: Consider context $q'$ of query $q$
  - e.g. surrounding text if query was entered via highlighting

- Calculation using a unigram language model:

$$P(c_j | q') = \frac{P(c_j) \cdot P(q'|c_j)}{P(q')} \propto P(c_j) \cdot \prod_i P(q'_i | c_j)$$
Topic-Sensitive PageRank

- Weighted summation of all topic specific PageRanks for one document
  - Weights: Dependent on probability of a particular topic being relevant given the query q
  - Definition: Query-Sensitive Importance Score $s_{qd}$

$$s_{qd} = \sum_j P(c_j | q') \cdot rank_{jd}$$

- Disadvantages:
  - Fixed set of topics
  - Depends on training set
SALSA

• Similarities
  – uses authority and hub score
  – creates a neighborhood graph using authority and hub pages and links

• Differences
  – creates bipartite graph of the authority and hub pages in the neighborhood graph.
  – Each page may be located in both sets
Neighborhood Graph N
Bipartite Graph $G$ of Neighborhood Graph $N$

$V_h = \{1, 2, 3, 6, 10\}$,
$V_a = \{1, 3, 5, 6\}$. 
Markov Chains

• Two matrices formed from bipartite graph G
• A hub Markov chain with matrix $H'$
  – Follow forward link, then backward
    $$h_{uv} = \sum_{w: (u,w) \in E, (v,w) \in E} \frac{1}{\deg(u_h)} \frac{1}{\deg(w_a)}$$

• An authority Markov chain with matrix $A'$
  – Follow backward link, then forward
    $$a_{uv} = \sum_{w: (w,u) \in E, (w,v) \in E} \frac{1}{\deg(v_a)} \frac{1}{\deg(w_h)}$$

• Steps end up on same side of the bipartite graph
Completing SALSA

• Use same power method as in previous methods to compute principal eigenvector
  – Caveat: have to deal with disconnected components!

{1}, {2}

{1,3,6,10}, {3,5,6}

– Link them together in some way
Where does SALSA fit in?

- Matrices $H'$ and $A'$ can be derived from the adjacency matrix used in both methods.
- HITS used unweighted matrix.
- PageRank uses a row weighted version of matrix $A$.
- SALSA uses both row and column weighting.

Why do we say this?
Strengths and Weaknesses

• Not affected as much by topic drift like HITS
• Handles Tightly knit communities better (spammers)
• It gives authority and hub scores.
• Query dependence
Summary

• Ranking needs to account for the graph structure
• Directed structure of the web leads to dichotomy in treatment (giving/receiving ends)
• Global models (propagation) and local models (at run time)
• Linear Algebra strikes again: SVD and Eigenvectors

Still more work to do here:
• Not yet convincingly coupled with standard retrieval models; “content” not really factored in
References

- S. Brin, L. Page, The anatomy of a large scale search engine, WWW 1998
- R. Motwani, P. Raghavan, Randomized Algorithms
- A. Langville, C. Meyer, Deeper Inside PageRank, Internet Mathematics