open.michigan

Author(s): Rahul Sami and Paul Resnick, 2009

Citation Key

for more information see: http://open.umich.edu/wiki/CitationPolicy

Use + Share + Adapt

{ Content the copyright holder, author, or law permits you to use, share and adapt. }

PD-GOV **Public Domain – Government**: Works that are produced by the U.S. Government. (USC 17 § 105)

PD-EXP **Public Domain – Expired**: Works that are no longer protected due to an expired copyright term.

PD-SELF **Public Domain – Self Dedicated**: Works that a copyright holder has dedicated to the public domain.

ZERO **Creative Commons – Zero Waiver**

BY **Creative Commons – Attribution License**

BY-SA **Creative Commons – Attribution Share Alike License**

BY-NC **Creative Commons – Attribution Noncommercial License**

BY-NC-SA **Creative Commons – Attribution Noncommercial Share Alike License**

GNU-FDL **GNU – Free Documentation License**

Make Your Own Assessment

{ Content Open.Michigan believes can be used, shared, and adapted because it is ineligible for copyright. }

PD-INEL **Public Domain – Ineligible**: Works that are ineligible for copyright protection in the U.S. (USC 17 § 102(b)) *laws in your jurisdiction may differ

{ Content Open.Michigan has used under a Fair Use determination. }

FAIR USE **Fair Use**: Use of works that is determined to be Fair consistent with the U.S. Copyright Act. (USC 17 § 107) *laws in your jurisdiction may differ

Our determination **DOES NOT** mean that all uses of this 3rd-party content are Fair Uses and we **DO NOT** guarantee that your use of the content is Fair.

To use this content you should **do your own independent analysis** to determine whether or not your use will be Fair.

Lecture 8:
Item-to-item; Page Rank

SI583: Recommender Systems

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

Item-Item Collaborative Filtering

High-level approach:

- For each item X find similar items Y,Z..
- For user Joe, recommend items most similar to items Joe has already liked

Users-by-Items Matrix

$$R = \begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{vmatrix}$$

Normalize the Rows for User-User Algorithm

$$X_{iJ} = R_{iJ} - \overline{R_i}$$

$$X = \begin{vmatrix} 1/3 & -2/3 & 1/3 \\ -2/3 & 1/3 & 1/3 \\ 1/3 & -2/3 & 1/3 \\ -2/3 & 1/3 & 1/3 \end{vmatrix} \qquad R = \begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{vmatrix}$$

Normalize the Columns for Item-Item Algorithm

$$W_{jk} = R_{jk} - \overline{W_k}$$

$$X = \begin{vmatrix} 0.5 & -.5 & 0 \\ -.5 & 0.5 & 0 \\ 0.5 & -.5 & 0 \\ -.5 & 0.5 & 0 \end{vmatrix}$$

$$X = \begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{vmatrix}$$

Alternative similarity measure for 0-1 entries: co-occurence

- When X has just 0 or 1 for each entry
- Instead of computing actual covariances from W, compute a similarity score based on count of co-occurrence in X
  - Co-occur(It1, It2) = 0
  - Co-occur(It1, It3) = 2

$$\Xi = \begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{vmatrix}$$

Generalization of co-occurrence similarity: Association Rules

- From a database of purchases, can find significant co-occurence rules, e.g.,

  person who buys bread and butter => 90% chance of also buying milk
- It's possible to precompute these association rules (Agarwal et al)

User-User vs. Item-Item

- Compute pairwise correlations between users

$$X \; X^T$$

- Compute pairwise correlations between items

$$W^T W$$

Computational Complexity

- With n items, m users,
  - user-user algorithm (unoptimized): about m^2n operations
  - item-item algorithm (unoptimized): about mn^2 operations
- #items *may be* < #users
- item-item similarities *may be* stable over long periods of time => batch computing leads to less inaccuracy

Predicted Scores for Target Item

- User-user
  - Weighted average of other user's ratings of this item
    - Weights taken from user-user similarities
- Item-item
  - Weighted average of this user's ratings of other items
    - Weights taken from item-item similarities

Finding Items from Items

- Item-item algorithm
  - Single starting item
    - Find other items with highest correlation
  - Starting from a group of items
    - Union of results for each item
    - (Why are association rules better than the item-item similarity matrix?)
- User-user algorithm
  - ??

Finding Users from Users

- User-user algorithm
  – Find other users with highest correlation
- Item-item algorithm
  – ??

Web search as a recommender

- Use links between pages as implicit “ratings”
- No separate categories of users, items
  - can’t easily use user-user algorithm, etc.
- How are the “best” pages for a query recommended?

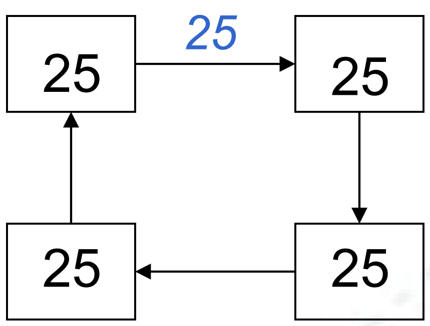
Model

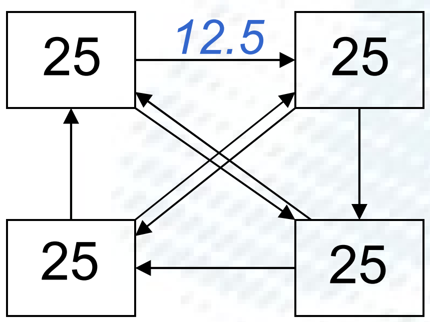
- Page is a node
- html link defines a directional link in the graph
- Terminology
  - If A has an html to B
    - A has an outgoing link to B
    - B has an incoming link from A

PageRank

- Google's big original idea [Brin &Page, 1998]
- Idea: ranking is based on "random web surfer":
  - start from any page at random
  - pick a random link from the page, and follow it
  - repeat!
  - ultimately, this process will converge to a stable distribution over pages (with some tricks...)
  - most likely page in this stable distribution is ranked highest
- Strong points:
  - Pages linked to by many pages *tend* to be ranked higher (not always)
  - A link ("vote") from a highly-ranked page carries more weight
  - Relatively hard to manipulate

PageRank, examples





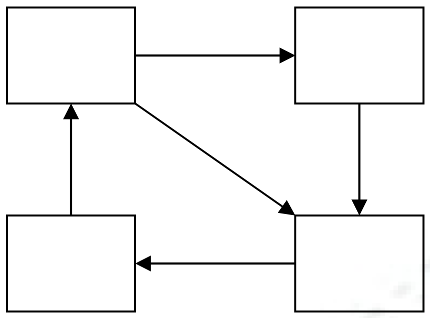
Final distribution properties:

(a) Total weight = 100%

(b) Weight of node is divided among outgoing links.

(c) Weight of node is sum of incoming link weights.

PageRank, examples



Final distribution properties:

(a) Total weight = 100%

(b) Weight of node is divided among outgoing links.

(c) Weight of node is some of incoming links

PageRank, mathematically

- Let the stable probabilities be x_i for page i, $x_i >= 0$
- For each i,j, define a_{ij} as
  - If j links to i, $a_{ij} = (1/\text{number of links of } j)$
  - If j does not link to i, $a_{ij} = 0$
- Form A = square matrix of a_{ij} for all i, j.
- Then, the PageRank probabilities satisfy

$$Ax = x$$

- x is the eigenvector of the link matrix, with eigenvalue 1

* May need to modify A slightly to ensure unique solution

optional

Finding the PageRank eigenvector

- One approach: solve linear equation

  $(\boldsymbol{A}\text{-}\boldsymbol{I})\boldsymbol{x} = (0\ 0\ 0\ ..0\ 0)^T$
- Alternative “power method” is more efficient in practice:
  - Start with an arbitrary X
  - Compute A^x, A^2x, ... A^tx (t large)
  - $\boldsymbol{A}^t\boldsymbol{x}$ is approximately proportional to the correct solution!

Aside: why the power method works (optional)

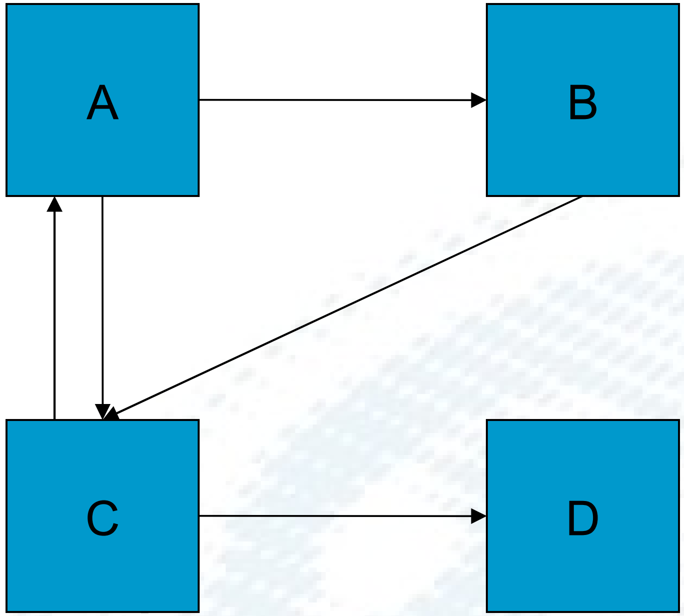
- Known: the link matrix A has
  - eigenvalue 1 for the correct eigenvector *v**
  - all other eigenvalues λ have $|\lambda| < 1$
- Known: any ***x*** can be expressed as a sum of eigenvectors of A

  $\mathbf{x} = a_0\mathbf{v}^* + a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + ..$
- Multiplying by A *t* times,

  $A^t x = a_0\mathbf{v}^* + a_1(\lambda_1)^t\mathbf{v}_1 + a_2(\lambda_2)^t\mathbf{v}_2 + ..$

  but $(\lambda_1)^t$ etc. are very close to 0 for large t

A Sample Graph

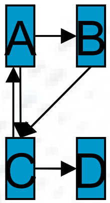


$$A = \begin{matrix} 0 & 0 & .5 & 0 \\ .5 & 0 & 0 & 0 \\ .5 & 1 & 0 & 0 \\ 0 & 0 & .5 & 0 \end{matrix}$$

Handling Loops

- Let E be a set of "source" weight ranks
  - At each node, random surfer goes to nodes with probabilities in E
- Each node's final rank is a scaled multiple of
  - It's source rank PLUS
  - The sum of the rank on its backlinks
- Scale it such that the sum of final ranks is 1

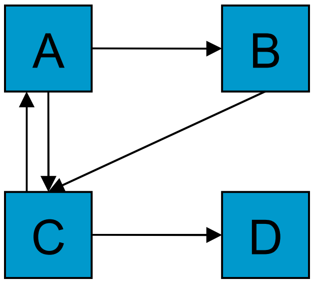
A Sample Graph



$$A = \begin{matrix} 0 & 0 & .5 & 0 \\ .5 & 0 & 0 & 0 \\ .5 & 1 & 0 & 0 \\ 0 & 0 & .5 & 0 \end{matrix}$$

$$E = \begin{matrix} .1 \\ .1 \\ .1 \\ .1 \end{matrix}$$

Some Intuitions



- Will D’s Rank be more or less than ¼?
- Will C’s Rank be more or less than B’s?
- How will A’s Rank compare to D’s?

Mathematical Expression

$$R' = c\,(AR + E)$$

$$\|P'\| = \quad |P_i| = 1$$

Power Method Algorithm

- Multiply by A, and then normalize so that the sum is 1

$$R_{i+1} = \frac{AR_i + E}{|AR_i + E|}$$

Before the First Iteration

- S

r1	.3
r2	.1
r3	.3
r4	.1

First Iteration

- AR+E

r1 .25
r2 .25
r3 .35
r4 .25

- Normalize so sum is 1 (divide by 1.1)

r1 .22727273
r2 .22727273
r3 .31818182
r4 .22727273

Second Iteration

- AR+E

r1 .25909091
r2 .21363636
r3 .44090909
r4 .25909091

- Normalized (divide by 1.17)

r1 .22093023
r2 .18217054
r3 .37596899
r4 .22093023

Third Iteration

- AR+E

r1 .2879845
r2 .21046512
r3 .39263566
r4 .2879845

- Normalized (divide by 1.18)

r1 .24424721
r2 .17850099
r3 .3330046
r4 .24424721

What If More Weight in E?

- Try (1 1 1 1) instead of (.1 .1 .1 .1)

r1	.23825503
r2	.2360179
r3	.28747204
r4	.23825503

- Try (10 10 10 10)

r1	.24848512
r2	.24845498
r3	.25457478
r4	.24848512

Personalized PageRank

- Pick E to be some sites that I like
  - My bookmarks
  - Links from my home page
- Rank flows more from these initial links than from other pages
  - But much of it may still flow to the popular sites, and from them to others that are not part of my initial set