

---

## CS 3750 Advanced Machine Learning

# Applications of SVD and PCA (LSA and Link analysis)

Cem Akkaya

---

---

## Outline

- SVD and LSI
  - Kleinberg's Algorithm
  - PageRank Algorithm
-

# Vector Space Model

- Vector space model represents database as a vector space
  - each document is represented as a vector. Each component and its weight of the vector represents an indexing term and its semantical importance in the document respectively
  - queries are modeled as vectors
  - a database containing a total of  $d$  documents described by  $t$  terms is represented as a  $t \times d$  term-by-document matrix
  - the semantic content of the database is wholly contained in the column space of  $A$

# Example

Terms ( $t=6$ ):                      Documents titles( $d=5$ ):  
T1:bak(e,ing)                      D1:How to bake bread without recipes  
T2:recipes                          D2:The classic art of Viennese pastry  
T3:bread                              D3:Numerical recipes: The art of scientific computing  
T4:cake                                D4:Breads, pastries, pies and cakes: quantity baking recipes  
T5:pastr(y,ies)                      D5:Pastry: A book of best french recipes  
T6:pie

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

6 x 5 term-by-document matrix

$$\hat{A} = \begin{pmatrix} 0.5774 & 0 & 0 & 0.4082 & 0 \\ 0.5774 & 0 & 1 & 0.4082 & 0.7071 \\ 0.5774 & 0 & 0 & 0.4082 & 0 \\ 0 & 0 & 0 & 0.4082 & 0 \\ 0 & 1 & 0 & 0.4082 & 0.7071 \\ 0 & 0 & 0 & 0.4082 & 0 \end{pmatrix}$$

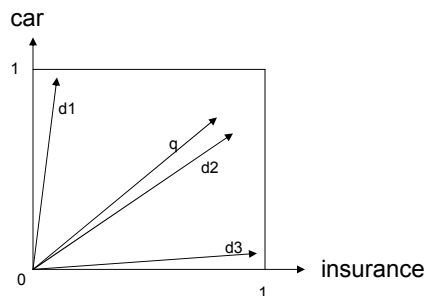
Normalized 6 x 5 term-by-document matrix with unit columns

## Similarity Measure

- Relevant documents are identified by simple vector operations
- Using spatial proximity for semantic proximity
- Most relevant documents for a query are expected to be those represented by the vectors closest to the query
- Cosine measure is the most widespread similarity measure. It gives the cosine of the angle between two vectors.
- If we work on unit vectors, cosine measure becomes just a simple dot product

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| |\vec{y}|} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}}$$

## Example



- A vector space with two dimensions. The two dimensions correspond to terms car and insurance
- Three documents and one query are represented as unit vectors
- D2 is the most similar document to query q, because it has the smallest angle with q

## Term weighting

- Simplest term (vector component) weightings are:
  - count of number of times word occurs in document
  - binary: word does or doesn't occur in document
- However, general experience is that a document is a better match if a word occurs three times than once, but not a three times better match.
- This leads to a series of weighting functions
  - e.g.,  $1 + \log(x)$  if  $x > 0$  else  $\text{squareroot}(x)$
- That doesn't capture that the occurrence of a term in a document is more important if that term does not occur in many other documents.
  - Solution:  $\text{weight} = \text{global weight} \times \text{local weight}$

## Problems

- The vector space representation suffers, from its inability to address two classic problems "*synonymy*" and "*polysemy*".
  - synonymy refers to a case where two different words (say car and automobile) have the same meaning.
  - polysemy on the other hand refers to the case where a term such as charge has multiple meanings
- Synonym causes to underestimate true similarity
  - $q(\text{car}) \text{ document}(\text{car}, \text{automobile})$ . Car and Automobile reside on separate dimensions in the vector space. Thus the similarity measure underestimates the true similarity
- Polysemy causes to overestimate true similarity
  - $q(\text{charge}) \text{ document}(\text{charge})$ . Charge has multiple meanings. Thus the similarity measure overestimates the true similarity
- Solution: LSI
  - could we use the co-occurrences of terms to capture the latent semantic associations of terms?

## Latent Semantic Indexing (LSI)

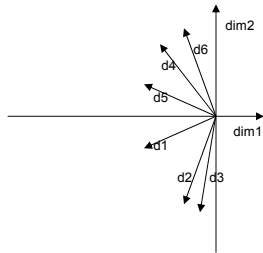
- Approach: Treat token-to-document association data as an unreliable estimate of a larger set of applicable words lying on 'latent' dimensions.
- Goal: Cluster similar documents which may share no terms in the latent semantic space, which is a low-dimensional subspace. (improves recall)
- LSI projects queries and documents into a space with latent semantic dimensions.
  - co-occurring words are projected on the same dimensions
  - non-co-occurring words are projected onto different dimensions
- Thus, LSI can be described as a method for dimensionality reduction
- In the latent semantic space a query and a document can have high cosine similarity even if they do not share any terms, as long as their terms are semantically related (according to co-occurrence)

## Latent Semantic Indexing (LSI)

- Moreover, dimensions of the reduced semantic space correspond to the axes of greatest variation in the original space (closely related to PCA)
- LSI is accomplished by an algebraic technique, called Singular Value Decomposition (SVD), to term-by-document matrix
- Steps:
  - preprocessing: Compute optimal low-rank approximation (latent semantic space) to the original term-by-document matrix with help of SVD
  - evaluation: Rank similarity of terms and docs to query in the latent semantic space via a usual similarity measure
- Optimality dictates that the projection into the latent semantic space should be changed as little as possible measured by the sum of the squares of differences

## Example

$$A = \begin{pmatrix} & d1 & d2 & d3 & d4 & d5 & d6 \\ \text{cosmonaut} & 1 & 0 & 1 & 0 & 0 & 0 \\ \text{astronaut} & 0 & 1 & 0 & 0 & 0 & 0 \\ \text{moon} & 1 & 1 & 0 & 0 & 0 & 0 \\ \text{car} & 1 & 0 & 0 & 1 & 1 & 0 \\ \text{truck} & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$



- A is a term-by-document matrix with rank 5
- In the next figure, the original five dimensional space is reduced to two dimensions (latent dimensions, concepts)
- In the original space the relation between d2 and d3 (space exploration documents) is not clear
- In the reduced latent semantic space it is easy to see that they are closely related
- How we reduce to two dimensions will be shown along SVD

## Singular Value Decomposition (SVD)

- Co-occurrence analysis and dimensionality reduction are two functional ways to understand LSI.
- It is done by the application of SVD projection
- SVD decomposes  $A_{t \times d}$  into the product of three matrices  $T_{t \times n}$ ,  $S_{n \times n}$  and  $D_{d \times n}^T$

$$A_{t \times d} = T_{t \times n} S_{n \times n} (D_{d \times n})^T$$

$$\begin{pmatrix} o & o & o \\ o & o & o \\ o & o & o \\ o & o & o \end{pmatrix} = \begin{pmatrix} o & o \\ o & o \\ o & o \\ o & o \end{pmatrix} \times \begin{pmatrix} o & o \\ o & o \end{pmatrix} \times \begin{pmatrix} o & o & o \\ o & o & o \end{pmatrix}$$

## Singular Value Decomposition (SVD)

- T and D matrices have orthonormal columns. (they are unit vectors and orthogonal to each other)
- S is a diagonal matrix containing singular values of A in descending order. The number of non-zero singular values gives the rank of A
- Columns of T are the orthogonal eigenvectors of  $AA^T$
- Columns of D are the orthogonal eigenvectors of  $A^T A$
- LSI defines:
  - A as term-by-document matrix
  - T as term-to-concept similarity matrix
  - S as concept strengths
  - D as concept-to-doc similarity matrix
- If rank of A is smaller than term count, we can directly project into a reduced dimensionality space. However, we may also want to reduce the dimensionality of A by setting small singular values of S to zero.

## Dimensionality Reduction

- SVD finds the optimal projection to a low-dimensional space (in the case of LSI, it's the latent semantic space)
- Compute SVD of  $A_{t \times d} = T_{t \times n} S_{n \times n} (D_{d \times n})^T$
- Form  $A^{\wedge}_{t \times k} = T_{t \times k} S_{k \times k} (D_{k \times n})^T$  by replacing the  $r - k$  smallest singular values on the diagonal by zeros, which is the optimal reduced rank-k approximation of  $A_{t \times d}$
- $B^{\wedge}_{t \times k} = S_{k \times k} (D_{k \times n})^T$  builds the projection of documents from the original space to the reduced rank-k approximation
  - in the original space, n dimensions correspond to terms
  - in the new reduced space, k dimensions correspond to concepts
- $Q_k = (T_{t \times k})^T Q_t$  builds the projection of the query from the original space to the reduced rank-k approximation
- Then we can rank similarity of documents to query in the reduced latent semantic space via a usual similarity measure
- That process is called LSI. It achieves higher recall than standard vector space search

## Example-SVD

$$A = \begin{pmatrix} & d1 & d2 & d3 & d4 & d5 & d6 \\ \text{cosmonaut} & 1 & 0 & 1 & 0 & 0 & 0 \\ \text{astronaut} & 0 & 1 & 0 & 0 & 0 & 0 \\ \text{moon} & 1 & 1 & 0 & 0 & 0 & 0 \\ \text{car} & 1 & 0 & 0 & 1 & 1 & 0 \\ \text{truck} & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$$T = \begin{pmatrix} & \text{dim1} & \text{dim2} & \text{dim3} & \text{dim4} & \text{dim5} \\ \text{cosmonaut} & -0.44 & -0.30 & 0.57 & 0.58 & 0.25 \\ \text{astronaut} & -0.13 & -0.33 & -0.59 & 0.00 & 0.73 \\ \text{moon} & -0.48 & -0.51 & -0.37 & 0.00 & -0.61 \\ \text{car} & -0.70 & 0.35 & 0.15 & -0.58 & 0.16 \\ \text{truck} & -0.26 & 0.65 & -0.41 & 0.58 & -0.09 \end{pmatrix}$$

$$S = \begin{pmatrix} 2.16 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.59 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.28 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.00 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.39 & 0 \end{pmatrix}$$

$$D^T = \begin{pmatrix} & d1 & d2 & d3 & d4 & d5 & d6 \\ \text{dim1} & -0.75 & -0.28 & -0.20 & -0.45 & -0.33 & -0.12 \\ \text{dim2} & -0.29 & -0.53 & -0.19 & 0.63 & 0.22 & 0.41 \\ \text{dim3} & 0.28 & -0.75 & 0.45 & -0.20 & 0.12 & -0.33 \\ \text{dim4} & 0 & 0 & 0.58 & 0 & -0.58 & 0.58 \\ \text{dim5} & -0.53 & 0.29 & -0.63 & 0.19 & 0.41 & -0.22 \end{pmatrix}$$

## Example-Reduction (rank-2 approx.)

$$S^r = \begin{pmatrix} 2.16 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.59 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

We can get rid of zero valued columns and rows  
And have a 2 x 2 concept strength matrix

$$T^r = \begin{pmatrix} & \text{dim1} & \text{dim2} & \text{dim3} & \text{dim4} & \text{dim5} \\ \text{cosmonaut} & -0.44 & -0.30 & 0 & 0 & 0 \\ \text{astronaut} & -0.13 & -0.33 & 0 & 0 & 0 \\ \text{moon} & -0.48 & -0.51 & 0 & 0 & 0 \\ \text{car} & -0.70 & 0.35 & 0 & 0 & 0 \\ \text{truck} & -0.26 & 0.65 & 0 & 0 & 0 \end{pmatrix}$$

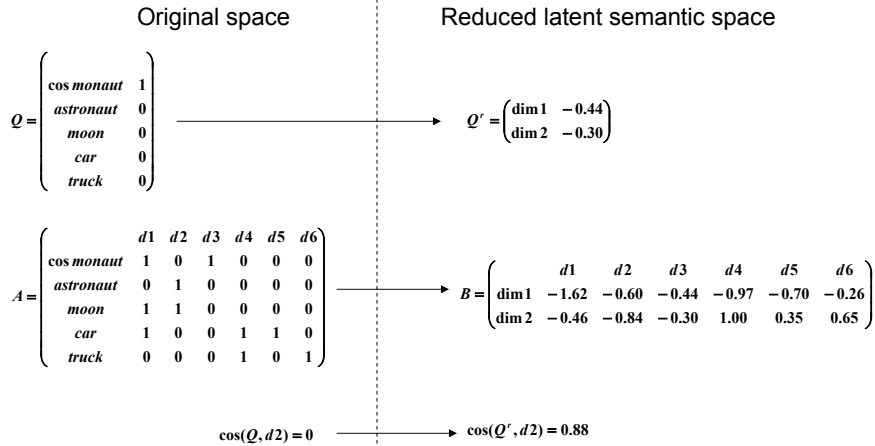
We can get rid of zero valued columns  
And have a 5 x 2 term-to-concept similarity matrix

$$D^{rT} = \begin{pmatrix} & d1 & d2 & d3 & d4 & d5 & d6 \\ \text{dim1} & -0.75 & -0.28 & -0.20 & -0.44 & -0.33 & -0.12 \\ \text{dim2} & -0.29 & -0.53 & -0.19 & 0.65 & 0.22 & 0.41 \\ \text{dim3} & 0 & 0 & 0 & 0 & 0 & 0 \\ \text{dim4} & 0 & 0 & 0 & 0 & 0 & 0 \\ \text{dim5} & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

We can get rid of zero valued columns  
And have a 2 x 6 concept-to-doc similarity matrix

**dim1 and dim2 are the new concepts**

## Example-Projection

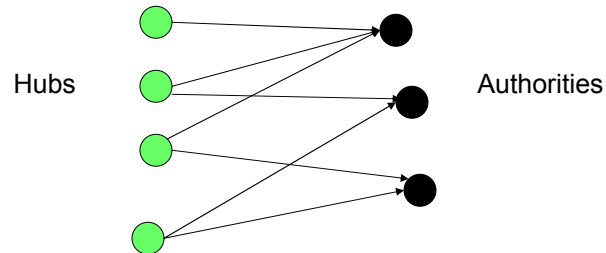


We see that query is not related to the d2 in the original space but in the latent semantic space they become highly related, which is true  
 Max [cos(x,y)]=1

## Kleinberg's Algorithm

- Extracting information from link structures of a hyperlinked environment
- Basic essentials
  - Authorities
  - Hubs
- For a topic, authorities are relevant nodes which are referred by many hubs
- For a topic, hubs are nodes which connect many related authorities for that topic
- Authorities are defined in terms of hubs and hubs defined in terms of authorities
  - Mutually enforcing relationship (global nature)

## Authorities and Hubs



- The algorithm can be applied to arbitrary hyperlinked environments
  - World Wide Web (nodes correspond to web pages with links)
  - Publications Database (nodes correspond to publications and links to co-citation relationship)

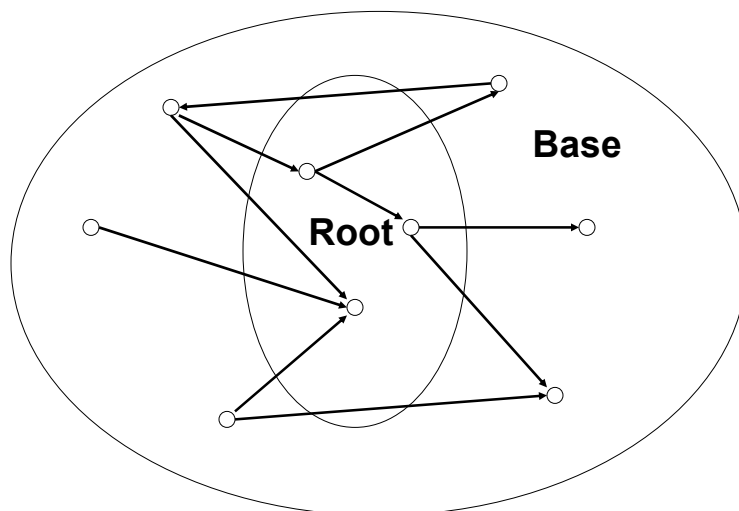
## Kleinberg's Algorithm (WWW)

- Is different from clustering
  - Different meanings of query terms
- Addressed problems by that model
  - Self-description of page may not include appropriate keywords
  - Distinguish between general popularity and relevance
- Three steps
  - Create a focused sub-graph of the Web
  - Iteratively compute hub and authority scores
  - Filter out the top hubs and authorities

## Root and Base Set

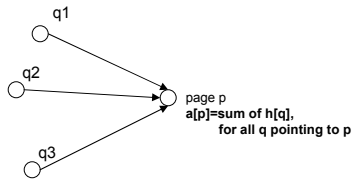
- For the success of the algorithm base set (sub-graph) should be
  - relatively small
  - rich in relevant pages
  - contains most of the strongest authorities
- Start first with a root set
  - obtained from a text-based search engine
  - does not satisfy third condition of a useful subgraph
- Solution: extending root set
  - add any page pointed by a page in the root set to it
  - add any page that points to a page in the root set to it (at most d)
  - the extended root set becomes our base set

## Root and Base Set

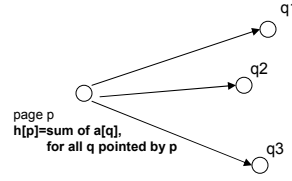


# Two Operations

## Updating authority weight

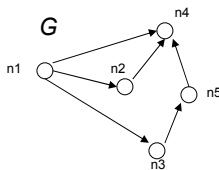


## Updating hub weight



- $a[p]$  ... authority weight for page  $p$
- $h[p]$  ... hub weight for page  $p$
- Iterative algorithm
  1. set all weights for each page to 1
  2. apply both operations on each page from the base set and normalize authority and hub weights separately (sum of squares=1)
  3. repeat step 2 until weights converge

# Matrix Notation



$$A = \begin{pmatrix} & n1 & n2 & n3 & n4 & n5 \\ n1 & 0 & 1 & 1 & 1 & 0 \\ n2 & 0 & 0 & 0 & 1 & 0 \\ n3 & 0 & 0 & 0 & 0 & 1 \\ n4 & 0 & 0 & 0 & 0 & 0 \\ n5 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

- $G$  (root set) is a directed graph with web pages as nodes and their links
- $G$  can be presented as a connectivity matrix  $A$ 
  - $A(i,j)=1$  only if  $i$ -th page points to  $j$ -th page
- Authority weights can be represented as a unit vector  $a$ 
  - $a(i)$  is the authority weight of the  $i$ -th page
- Hub weights can be represented as a unit vector  $h$ 
  - $h(i)$  is the hub weight of the  $i$ -th page

## Convergence

- Two mentioned basic operations can be written as matrix operations (all values are updated simultaneously)
  - Updating authority weights:  $a=A^T h$
  - Updating hub weights:  $h=Aa$
- After k iterations:

$$\begin{array}{l} a_1 = A^T h_0 \\ h_1 = A a_1 \end{array} \longrightarrow h_1 = A A^T h_0 \rightarrow h_k = (A A^T)^k h_0$$

- Thus
  - $h_k$  is a unit vector in the direction of  $(A A^T)^k h_0$
  - $a_k$  is a unit vector in the direction of  $(A^T A)^{k-1} h_0$
- Theorem
  - $a_k$  converges to the principal eigenvector of  $A^T A$
  - $h_k$  converges to the principal eigenvector of  $A A^T$

## Convergence

- $(A^T A)^k x v^T \approx (\text{const}) v_1$  where  $k \gg 1$ ,  $v^T$  is a random vector,  $v_1$  is the eigenvector of  $A^T A$
- Proof:

$$\begin{aligned} (A^T A)^k &= (A^T A) x (A^T A) x \dots = (V \Lambda^2 V^T) x (V \Lambda^2 V^T) x \dots \\ &= (V \Lambda^2 V^T) x \dots = (V \Lambda^4 V^T) x \dots = (V \Lambda^{2k} V^T) \end{aligned}$$

Using spectral decomposition:

$$(A^T A)^k = (V \Lambda^{2k} V^T) = \lambda_1^{2k} v_1 v_1^T + \lambda_2^{2k} v_2 v_2^T + \dots + \lambda_n^{2k} v_n v_n^T$$

because  $\lambda_1 > \lambda_{i \neq 1} \rightarrow \lambda_1^{2k} \gg \lambda_{i \neq 1}^{2k}$

thus  $(A^T A)^k \approx \lambda_1^{2k} v_1 v_1^T$

now  $(A^T A)^k x v^T = \lambda_1^{2k} v_1 v_1^T x v^T = (\text{const}) v_1$

because  $v_1^T x v^T$  is a scalar.

## Sub-communities

- Authority vector converges to the principal eigenvector of  $A^T A$ , which lets us choose strong authorities
- Hub vector converges to the principal eigenvector of  $A A^T$  which lets us choose strong hubs
- These chosen authorities and hubs build a cluster in our network
- However there can exist different clusters of authorities and hubs for a given topic, which correspond to:
  - different meanings of a term (e.g. jaguar → animal, car, team)
  - different communities for a term (e.g. randomized algorithms)
  - polarized thoughts for a term (e.g. abortion)
- Extension:
  - each eigenvector of  $A^T A$  and  $A A^T$  represents distinct authority and hub vectors for a sub-community in Graph  $G$ , respectively.

## PageRank

- PageRank is a link analysis algorithm that assigns weights to nodes of a hyperlinked environment
- It assigns importance scores to every node in the set which is similar to the authority scores in Kleinberg algorithm
- It is an iterative algorithm like Kleinberg algorithm
- Main assumptions:
  - in-degree of nodes are indicators of their importance
  - links from different nodes are not counted equally. They are normalized by the out-degree of its source.

## Simplified PageRank (WWW)

$$\Pr(u) = \sum_{v \in B(u)} \frac{\Pr(v)}{L(v)}$$

$B(u)$  is the set of nodes  
which have a link to  $u$

- Pr value of a page (node) depends on all pages which have a link to it and contribution of each page is divided by its out-degrees
- PageRank Algorithm simulates a random walk over web pages.
- Pr value is interpreted as probabilities
- In each iteration we update Pr values of each page simultaneously
- After several passes, Pr value converges to a probability distribution used to represent the probability that a person randomly clicking on links will arrive at any particular page

## Matrix Notation

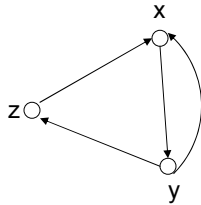
Update step

$$\Pr_{k \times 1} = M_{k \times k} \times \Pr_{k \times 1} \quad M_{ij} = \begin{cases} \frac{1}{|B_j|} & , \text{ if } i \in B_j \\ 0 & , \text{ else} \end{cases}$$

$k$  is the number of total pages  
 $B_i$  is the set of pages which have a link to  $i$ -th page

- $M(i,j)$  is the transition matrix and defines fragment of the  $j$ -th page's Pr value which contributes to the Pr value of the  $i$ -th page
- PageRank essentially defines a Markov Chain on the pages with transition matrix  $M$  and stationary distribution  $\Pr$ 
  - states are pages
  - transitions are the links between pages (all equally probable)
- As a result of Markov theory, Pr value of a page is the probability of being at that page after lots of clicks.

# Matrix Notation



Update step

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 & 0.5 & 1 \\ 1 & 0 & 0 \\ 0 & 0.5 & 0 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$x = 0 \cdot x + 1/2 \cdot y + 1 \cdot z$$

$$y = 1 \cdot x + 0 \cdot y + 0 \cdot z$$

$$z = 0 \cdot x + 1/2 \cdot y + 0 \cdot z$$

# Non-Simplified Pagerank (WWW)

$$\Pr(u) = \frac{1-d}{k} + d \cdot \sum_{v \in B(u)} \frac{\Pr(v)}{L(v)}$$

Matrix Notation

$$\Pr_{k \times 1} = M_{k \times k} \times \Pr_{k \times 1}$$

$k$  is the number of total pages  
 $B_i$  is the set of pages which have a link to  $i$ -th page

$$M_{ij} = \begin{cases} \frac{1-d}{k} + \frac{d}{|B_j|}, & \text{if } i \in B_j \\ \frac{1-d}{k}, & \text{else} \end{cases}$$

- (1-d) defines the probability to jump to a page, to which there is no link from the current page
- Theorem:
  - Pr converges to the principal eigenvector of the transition matrix

---

Thanks for your attention

Special thanks to lyad

---