Lecture 12
Text Mining and Web Mining

Zhou Shuigeng

June 10, 2007

Part I: Information Retrieval and Text Mining

Zhou Shuigeng

May 28, 2006

Text Databases and IR

- Text databases (document databases)
  - Large collections of documents from various sources: news articles, research papers, books, digital libraries, e-mail messages, and Web pages, library database, etc.
  - Data stored is usually *semi-structured*
- Information retrieval
  - A field developed in parallel with database systems
  - Information is organized into (a large number of) documents
  - Information retrieval problem: locating relevant documents based on user input, such as keywords or example documents

Information Retrieval

- Typical IR systems
  - Online library catalogs
  - Online document management systems
- Information retrieval vs. database systems
  - Some DB problems are not present in IR, e.g., update, transaction management, complex objects
  - Some IR problems are not addressed well in DBMS, e.g., unstructured documents, approximate search using keywords and relevance

IR Techniques(1)

- Basic Concepts
  - A document can be described by a set of representative keywords called index terms.
  - Different index terms have varying relevance when used to describe document contents.
  - This effect is captured through the assignment of numerical weights to each index term of a document. (e.g.: frequency, tf-idf)
- DBMS Analogy
  - Index Terms → Attributes
  - Weights → Attribute Values

IR Techniques(2)

- Index Terms (Attribute) Selection:
  - Term extraction
  - Stop list
  - Word stem
  - Index terms weighting methods
- Terms × Documents Frequency Matrices
- Information Retrieval Models:
  - Boolean Model
  - Vector Model
  - Probabilistic Model

Keyword Extraction

- Goal:
  - given N documents, each consisting of words,
  - extract the most significant subset of words → keywords
  - Example
    - [All the students are taking exams] -- >[student, take, exam]
- Keyword Extraction Process
  - remove stop words
  - stem remaining terms
  - collapse terms using thesaurus
  - build inverted index
  - extract key words - build key word index
  - extract key phrases - build key phrase index

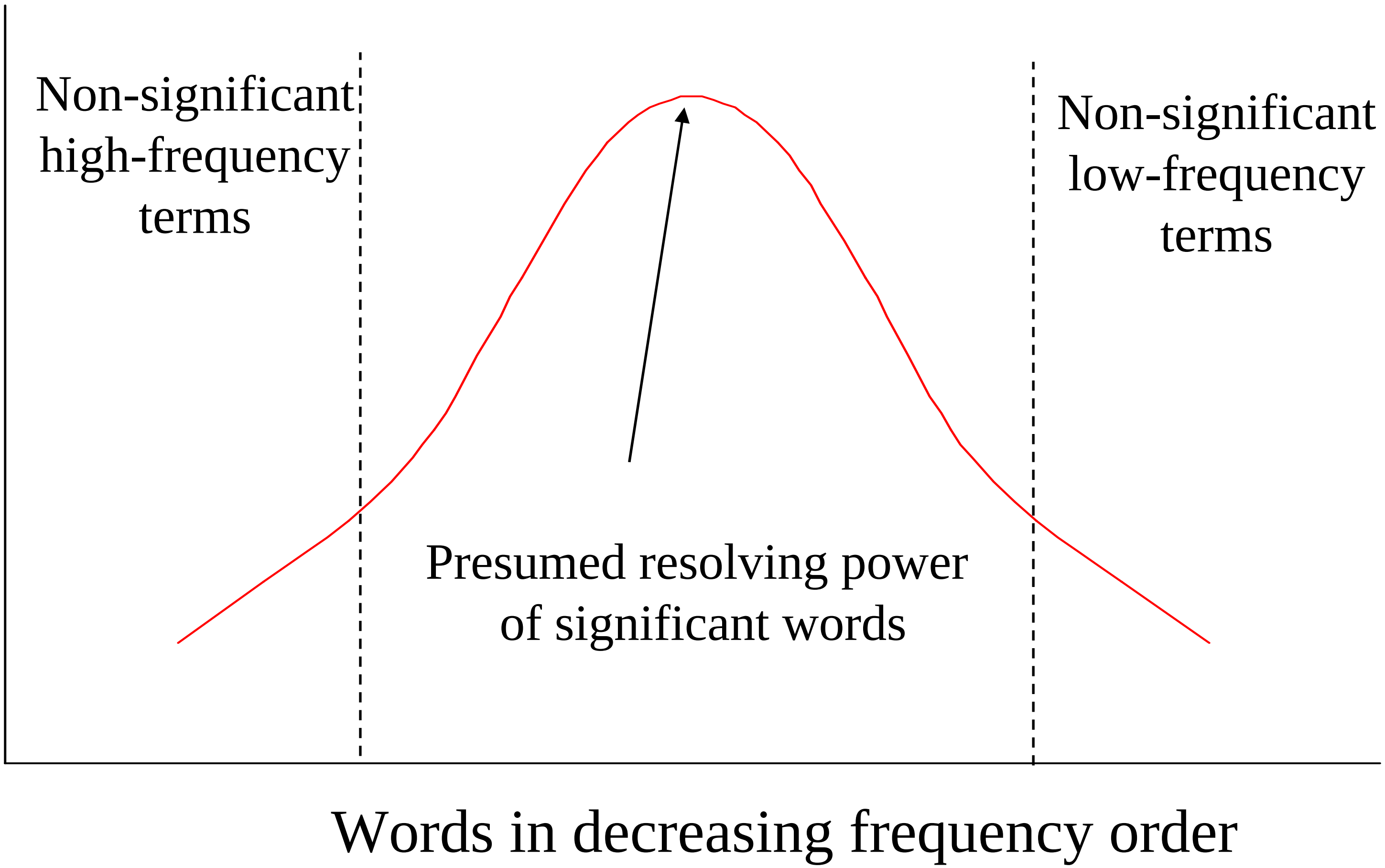
Stop Words and Stemming

- From a given Stop Word List
  - [a, about, again, are, the, to, of, …]
  - Remove them from the documents
- Or, determine stop words
  - Given a large enough corpus of common English
  - Sort the list of words in decreasing order of their occurrence frequency in the corpus
  - Zipf's law: Frequency * rank ≈ constant
    - most frequent words tend to be short
    - most frequent 20% of words account for 60% of usage

Zipf's Law -- An illustration

Rank(R)	Term	Frequency (F)	R*F (10**6)
1	the	69,971	0.070
2	of	36,411	0.073
3	and	28,852	0.086
4	to	26,149	0.104
5	a	23,237	0.116
6	in	21,341	0.128
7	that	10,595	0.074
8	is	10,009	0.081
9	was	9,816	0.088
10	he	9,543	0.095

Resolving Power of Word



Simple Indexing Scheme Based on Zipf's Law

Use term frequency information only:

- Compute frequency of term k in document i, $Freq_{ik}$
- Determine total collection frequency

  $TotalFreq_k = \sum Freq_{ik}$ for $i = 1, 2, \ldots, n$
- Arrange terms in order of collection frequency
- Set thresholds - eliminate high and low frequency terms
- Use remaining terms as index terms

Stemming

- The next task is stemming: transforming words to root form
  - Computing, Computer, Computation →comput
- Suffix based methods
  - Remove “ability” from “computability”
  - “…”+ness, “…”+ive, → remove
- Suffix list + context rules

Thesaurus Rules

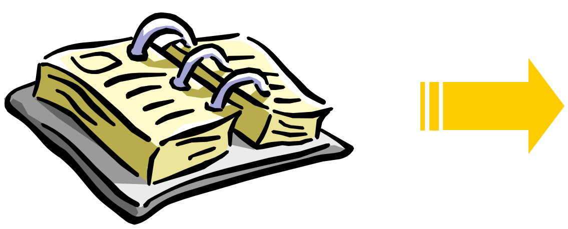
- A thesaurus aims at
  - classification of words in a language
  - for a word, it gives related terms which are broader than, narrower than, same as (synonyms) and opposed to (antonyms) of the given word (other kinds of relationships may exist, e.g., composed of)
- Static Thesaurus Tables
  - [anneal, strain], [antenna, receiver], …
  - Roget's thesaurus
  - WordNet at Preinceton

Thesaurus Rules can also be Learned

- From a search engine query log
  - After typing queries, browse…
  - If query1 and query2 leads to the same document
    - Then, Similar(query1, query2)
  - If query1 leads to Document with title keyword K,
    - Then, Similar(query1, K)
  - Then, transitivity…
- Microsoft Research China's work in WWW10 (Wen, et al.) on Encarta online

The Vector-Space Model

- *The* distinct terms are available; call them *index terms* or the *vocabulary*
- The index terms represent important terms for an application → a vector to represent the document
  - <T1,T2,T3,T4,T5> or <W(T1),W(T2),W(T3),W(T4),W(T5)>



computer science collection

T1=architecture
T2=bus
T3=computer
T4=database
T5=xml

index terms or vocabulary of the collection

The Vector-Space Model

- Assumptions: words are uncorrelated

Given:

1. N documents and a Query
2. Query considered a document too
2. Each represented by *t* terms

3. Each term *j* in document *i* has weight d_{ij}

4. We will deal with how to compute the weights later

$$
\begin{array}{ccccc}
 & T_1 & T_2 & \cdots & T_t \\
D_1 & d_{11} & d_{12} & \cdots & d_{1t} \\
D_2 & d_{21} & d_{22} & \cdots & d_{2t} \\
\vdots & \vdots & \vdots & & \vdots \\
\vdots & \vdots & \vdots & & \vdots \\
D_n & d_{n1} & d_{n2} & \cdots & d_{nt}
\end{array}
$$

$$Q \quad q_1 \quad q_2 \quad \ldots \; q_t$$

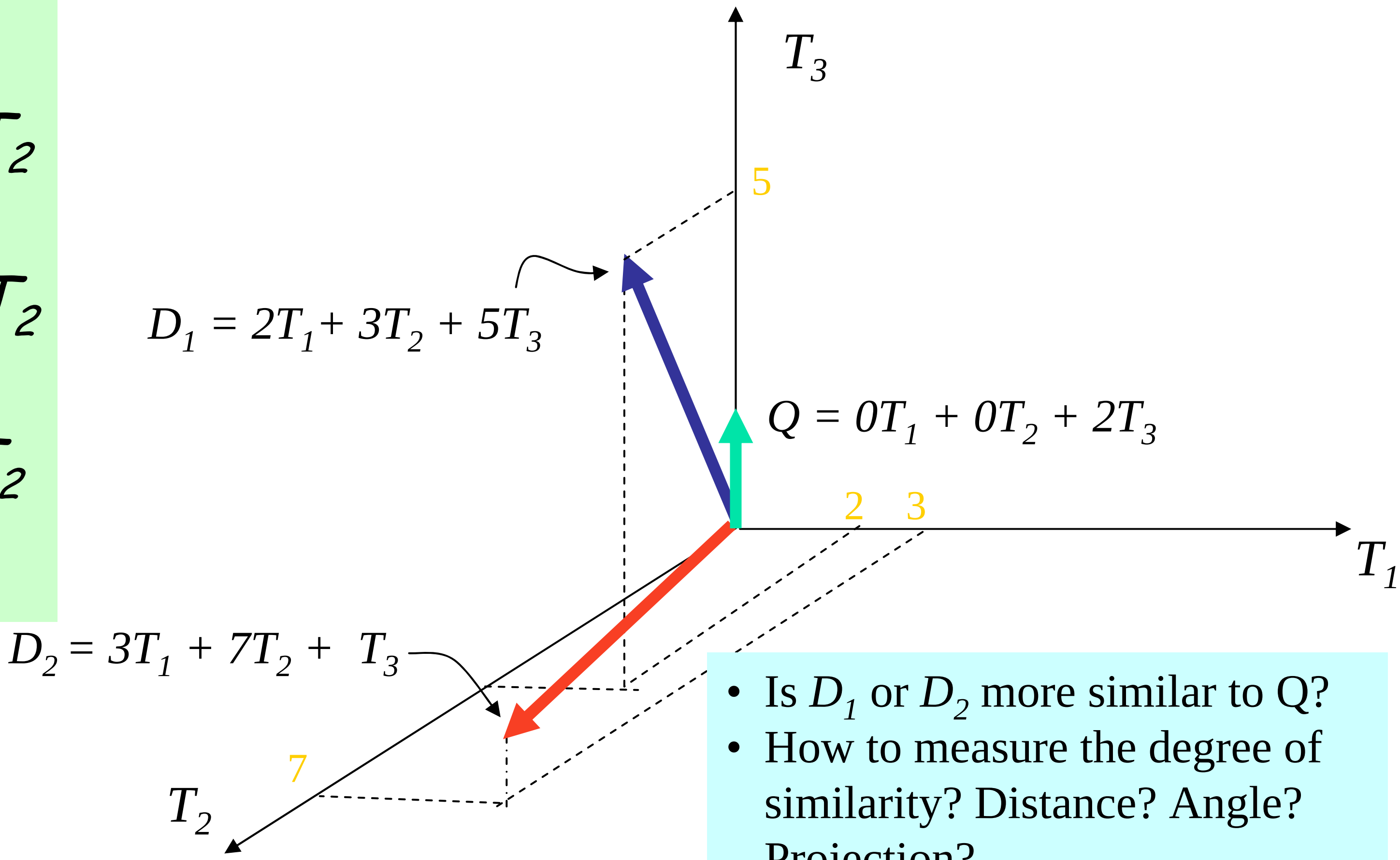
Graphic Representation

Example:

$D_1 = 2T_1 + 3T_2 + 5T_3$

$D_2 = 3T_1 + 7T_2 + T_3$

$Q = 0T_1 + 0T_2 + 2T_3$



- Is D_1 or D_2 more similar to Q?
- How to measure the degree of similarity? Distance? Angle? Projection?

Similarity Measure - Inner Product

- Similarity between documents D_i and query Q can be computed as the inner vector product:

$$\text{sim}(D_i, Q) = \sum_{k=1}^{t} (D_i \bullet Q)$$

$$= \sum_{j=1}^{t} d_{ij} * q_j$$

- Binary: weight = 1 if word present, 0 o/w
- Non-binary: weight represents degree of similary
  - Example: TF/IDF we explain later

Inner Product -- Examples

Size of vector = size of vocabulary = 7

Binary:

	retrieval	database	architecture	computer	text	management	information
D =	1,	1,	1,	0,	1,	1,	0
Q =	1,	0 ,	1,	0,	0,	1,	1

→ sim(D, Q) = 3

Weighted

$$D_1 = 2T_1 + 3T_2 + 5T_3$$

$$Q = 0T_1 + 0T_2 + 2T_3$$

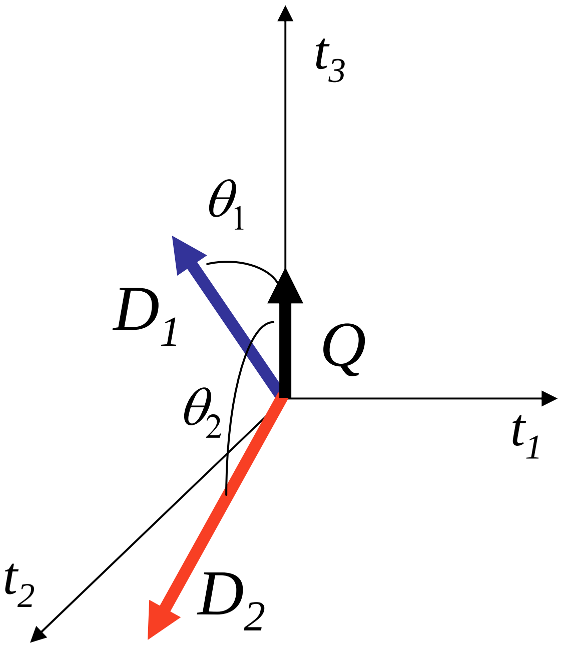
$$\text{sim}(D_1, Q) = 2*0 + 3*0 + 5*2 = 10$$

Properties of Inner Product

- The inner product similarity is unbounded
- Favors long documents
  - long document $\Rightarrow$ a large number of unique terms, each of which may occur many times
  - measures how many terms matched but not how many terms *not* matched

Cosine Similarity Measures

- Cosine similarity measures the cosine of the angle between two vectors
- Inner product normalized by the vector lengths



$$\text{CosSim}(D_i, Q) = \frac{\sum_{k=1}^{t} (d_{ik} \bullet q_k)}{\sqrt{\sum_{k=1}^{t} d_{ik}^{\ 2} \bullet \sum_{k=1}^{t} q_k^{\ 2}}}$$

Cosine Similarity: an Example

$D_1 = 2T_1 + 3T_2 + 5T_3$ CosSim(D_1 , Q) = 5 / √ 38 = 0.81
$D_2 = 3T_1 + 7T_2 + T_3$ CosSim(D_2 , Q) = 1 / √ 59 = 0.13
$Q = 0T_1 + 0T_2 + 2T_3$

D_1 is 6 times better than D_2 using cosine similarity but only 5 times better using inner product

Document and Term Weights

- Document term weights are calculated using frequencies in documents (*tf*) and in collection (*idf*)

  tf_{ij} = frequency of term *j* in document *i*

  df_j = document frequency of term *j*

  = number of documents containing term *j*

  idf_j = inverse document frequency of term *j*

  = $\log_2 (N/df_j)$ (N: number of documents in collection)

- Inverse document frequency -- an indication of term values as a document discriminator.

Term Weight Calculations

- Weight of the jth term in ith document:

$$d_{ij} = tf_{ij} \bullet idf_j = tf_{ij} \bullet \log_2 (N / df_j)$$

  - TF → Term Frequency
    - A term occurs frequently in the document but rarely in the remaining of the collection has a high weight
    - *Let* $max_l\{tf_{lj}\}$ be the term frequency of the most frequent term in document *j*
    - *Normalization: term frequency* $= tf_{ij} / max_l\{tf_{lj}\}$

An example of TF

- Document=(A Computer Science Student Uses Computers)
- Vector Model based on keywords (Computer, Engineering, Student)

  Tf(Computer) = 2

  Tf(Engineering)=0

  Tf(Student) = 1

  Max(Tf)=2

  TF weight for:

  Computer = 2/2 = 1

  Engineering = 0/2 = 0

  Student = ½ = 0.5

Inverse Document Frequency

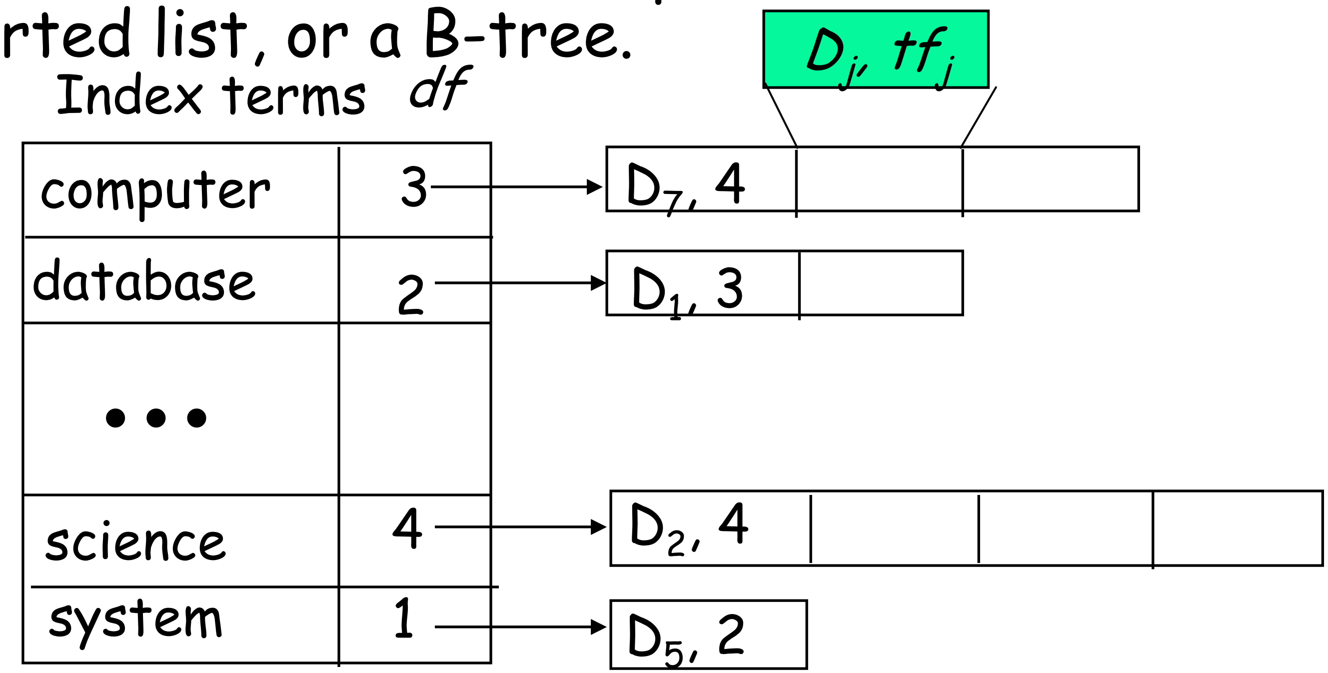
- Df_j gives the number of times term *j* appeared among N documents
- IDF = 1/DF
- Typically use $\log_2 (N/ df_j)$ for IDF
- Example: given 1000 documents, computer appeared in 200 of them,
  - IDF= $\log_2 (1000/ 200) = \log_2(5)$

TF IDF

- $d_{ij} = (tf_{ij} / max_l\{tf_{lj}\}) \bullet idf_j$
  $= (tf_{ij} / max_l\{tf_{lj}\}) \bullet \log_2 (N / df_j)$
- Can use this to obtain non-binary weights
- Used in the SMART Information Retrieval System by the late Gerald Salton and MJ McGill, Cornell University to tremendous success, 1983

Implementation based on Inverted Files

- In practice, document vectors are not stored directly; an inverted organization provides much better access speed.
- The index file can be implemented as a hash file, a sorted list, or a B-tree.



A Simple Search Engine

Now we have got enough tools to build a simple Search engine (documents == web pages)

1. Starting from well known web sites, *crawl* to obtain N web pages (for very large N)
2. Apply stop-word-removal, stemming and thesaurus to select K keywords
3. Build an inverted index for the K keywords
4. For any incoming user query Q,
    1. For each document D
        1. Compute the Cosine similarity *score* between Q and document D
    2. Select all documents whose *score* is over a certain threshold T
    3. Let this result set of documents be M
    4. Return M to the user

Remaining Questions

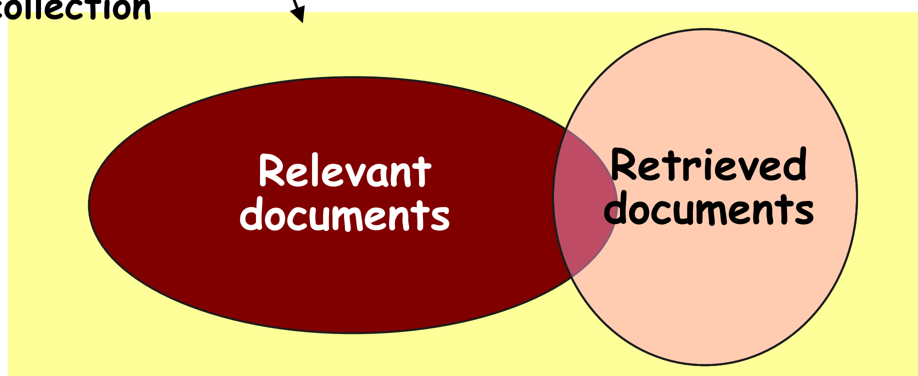
- How to crawl?
- How to evaluate the result
  - Given 3 search engines, which one is better?
    - Is there a quantitative measure?

Measurement

- Let M documents be returned out of a total of N documents;
- N=N1+N2
  - N1 total documents are relevant to query
  - N2 are not
- M=M1+M2
  - M1 found documents are relevant to query
  - M2 are not
- Precision = M1/M
- Recall = M1/N1

Retrieval effectiveness: recall & precision

Entire document collection



	retrieved	not retrieved
irrelevant	retrieved & irrelevant	Not retrieved & irrelevant
relevant	retrieved & relevant	not retrieved but relevant

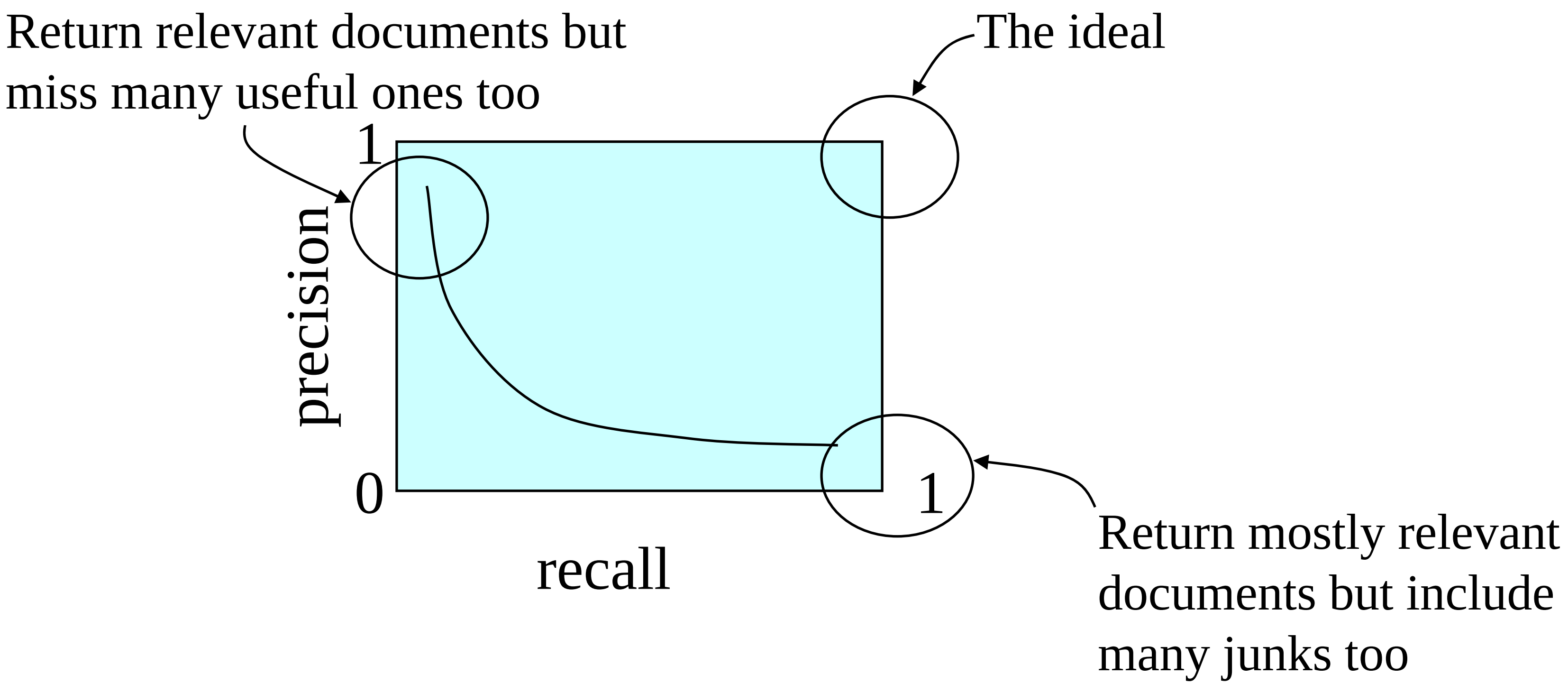
$$recall = \frac{Number\ of\ relevant\ documents\ retrieved}{Total\ number\ of\ relevant\ documents}$$

$$precision = \frac{Number\ of\ relevant\ documents\ retrieved}{total\ Number\ of\ documents\ retrieved}$$

Precision and Recall

- Precision
  - evaluates the correlation of the query to the database
  - an indirect measure of the completeness of indexing algorithm
- Recall
  - the ability of the search to find all of the relevant items in the database
- Among three numbers,
  - only two are always available
    - *total number of items retrieved*
    - *number of relevant items retrieved*
  - *total number of relevant items* is usually not available

Relationship between Recall and Precision



Fallout Rate

- Problems with precision and recall:
  - A query on “Hong Kong” will return most relevant documents but it doesn't tell you how good or how bad the system is !
  - number of irrelevant documents in the collection is not taken into account
  - recall is undefined when there is no relevant document in the collection
  - precision is undefined when no document is retrieved

$$Fallout = \frac{no.of\ \ nonrelevant\ items\ retrieved}{total\ no.of\ \ nonrelevant\ items\ in\ the\ collection}$$

Fallout can be viewed as the inverse of recall. A good system should have high recall and low fallout

Total Number of Relevant Items

- In an uncontrolled environment (e.g., the web), it is unknown.
- Two possible approaches to get estimates
  - Sampling across the database and performing relevance judgment on the returned items
  - Apply different retrieval algorithms to the same database for the same query. The aggregate of relevant items is taken as the total relevant algorithm

Computation of Recall and Precision

n	doc #	relevant	Recall	Precision
1	588	x	0.2	1.00
2	589	x	0.4	1.00
3	576		0.4	0.67
4	590	x	0.6	0.76
5	986		0.6	0.60
6	592	x	0.8	0.67
7	984		0.8	0.57
8	988		0.8	0.50
9	578		0.8	0.44
10	985		0.8	0.40
11	103		0.8	0.36
12	591		0.8	0.33
13	772	x	1.0	0.38
14	990		1.0	0.36

Suppose:
total no. of relevant docs = 5

R=1/5=0.2; p=1/1=1

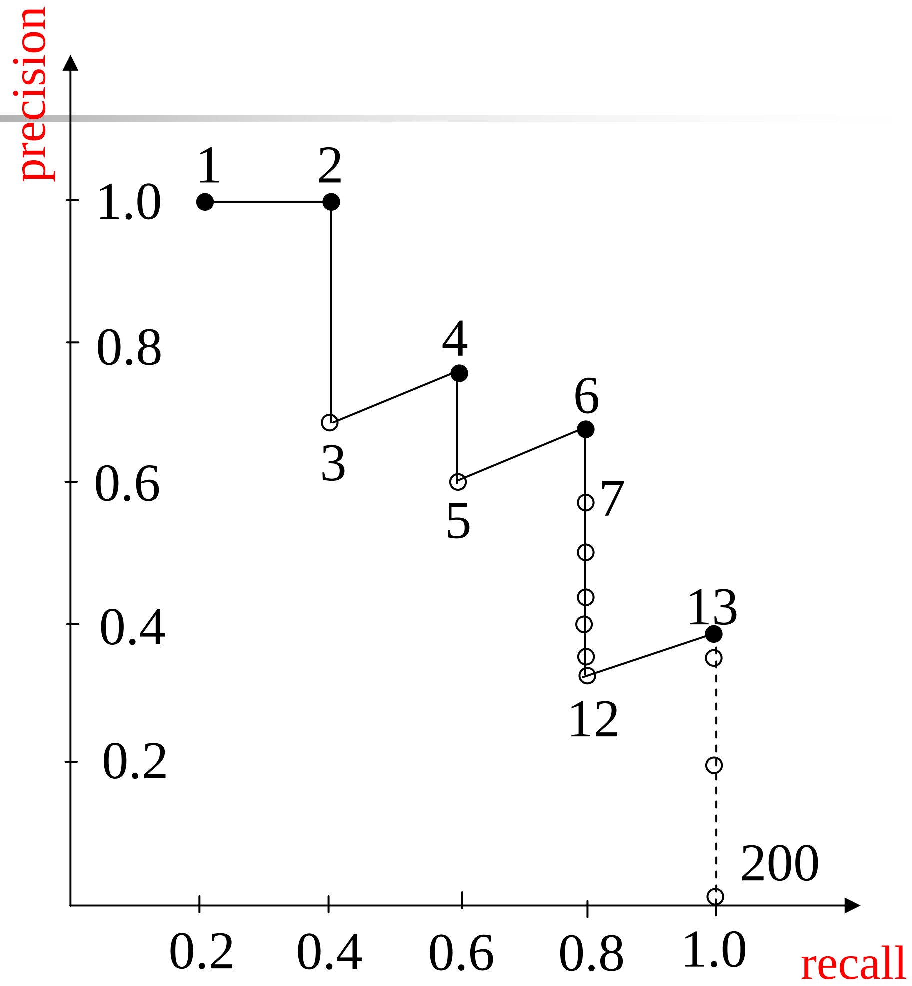
R=2/5=0.4; p=2/2=1

R=2/5=0.4; p=2/3=0.67

R=5/5=1; p=5/13=0.38

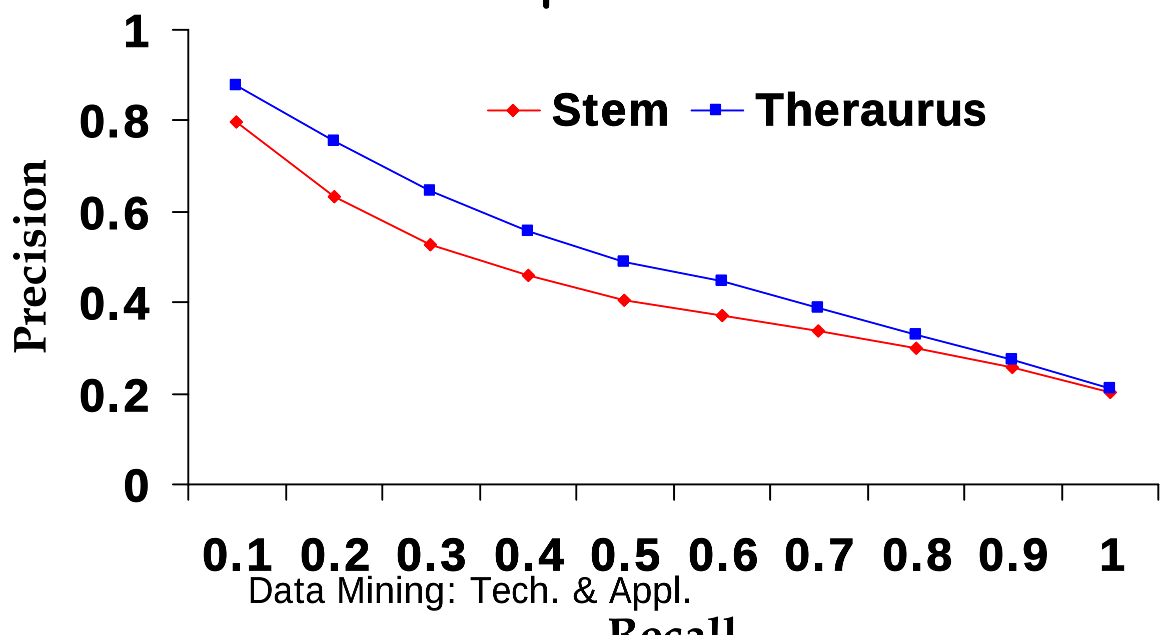
Computation of Recall and Precision

n	Recall	Precision
1	0.2	1.00
2	0.4	1.00
3	0.4	0.67
4	0.6	0.76
5	0.6	0.60
6	0.8	0.67
7	0.8	0.57
8	0.8	0.50
9	0.8	0.44
10	0.8	0.40
11	0.8	0.36
12	0.8	0.33
13	1.0	0.38
14	1.0	0.36



Compare Two or More Systems

- Computing recall and precision values for two or more systems
- Superimposing the results in the same graph
- The curve closest to the upper right-hand corner of the graph indicates the best performance



The TREC Benchmark

TREC: Text Retrieval Conference

Originated from the TIPSTER program sponsored by Defense Advanced Research Projects Agency (DARPA)

Became an annual conference in 1992, co-sponsored by the National Institute of Standards and Technology (NIST) and DARPA

Participants are given parts of a standard set of documents and queries in different stages for testing and training

Participants submit the P/R values on the final document and query set and present their results in the conference

http://trec.nist.gov/

Interactive Search Engines

- Aims to improve their search results incrementally,
  - often applies to query "Find *all* sites with certain property"
  - Content based Multimedia search: given a photo, find all other photos similar to it
    - Large vector space
    - Question: which feature (keyword) is important?
- Procedure:
  - User submits query
  - Engine returns result
  - *User marks some returned result as relevant or irrelevant, and continues search*
  - Engine returns new results
  - Iterates until user satisfied

Query Reformulation

- Based on user's feedback on returned results
  - Documents that are relevant D_R
  - Documents that are irrelevant D_N
  - Build a new query vector Q' from Q
    - <w1, w2, … wt> → <w1', w2', … wt'>
  - Best known algorithm: Rocchio's algorithm
  - Also extensively used in multimedia search

Query Modification

- Using the previously identified relevant and nonrelevant document set D_R and D_N to repeatedly modify the query to reach optimality
- Starting with an initial query in the form of

$$Q' = \alpha * Q + (\frac{1}{R} \sum_{i \in D_R} D_i) - \gamma (\frac{1}{N} \sum_{j \in D_N} D_j)$$

where Q is the original query, and α, β, and γ are suitable constants

An Example

Q: original query

D1: relevant doc.

D2: non-relevant doc.

α = 1, β = 1/2, γ = 1/4

Assume: dot-product similarity measure

	T1	T2	T3	T4	T5
Q =	(5,	0,	3,	0,	1)
D1 =	(2,	1,	2,	0,	0)
D2 =	(1,	0,	0,	0,	2)

$$S(Q, D_i) = \sum_{j=1}^{t} (Q_j \times D_{ij})$$

Sim(Q,D1) = (5•2)+(0 • 1)+(3 • 2)+(0 • 0)+(1 • 0) = 16
Sim(Q,D2) = (5•1)+(0 • 0)+(3 • 0)+(0 • 0)+(1 • 2) = 7

Example (Cont.)

$$Q' = Q + \frac{1}{2}\left(\sum_{i \in D_{R'}} D_i\right) - \frac{1}{4}\left(\frac{1}{N'}\sum_{i \in D_{N'}} D_i\right)$$

$$Q' = (5,0,3,0,1) + \frac{1}{2}(2,1,2,0,0) - \frac{1}{4}(1,0,0,0,2)$$

$$Q' = (5.75, 0.5, 4, 0, 0.5)$$

New Similarity Scores:

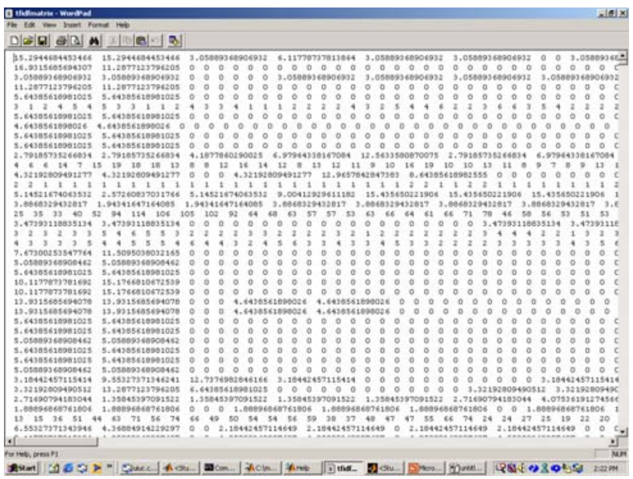
Sim(Q', D1)=(5.75 • 2)+(0.5 • 1)+(4 • 2)+(0 • 0)+(0.5 • 0)=20
Sim(Q', D2)=(5.75 • 1)+(0.5 • 0)+(4 • 0)+(0 • 0)+(0.5 • 2)=6.75

Latent Semantic Indexing (1)

- Basic idea
  - Similar documents have similar word frequencies
  - Difficulty: the size of the term frequency matrix is very large
  - Use a singular value decomposition (SVD) techniques to reduce the size of frequency table
  - Retain the *K* most significant rows of the frequency table
- Method
  - Create a term x document weighted frequency matrix *A*
  - SVD construction: $A = U * S * V'$
  - Define K and obtain U_k ,, S_k , and V_k.
  - Create query vector q' .
  - Project q' into the term-document space: $Dq = q' * U_k * S_k^{-1}$
  - Calculate similarities: $Dq . D / ||Dq|| * ||D||$

Latent Semantic Indexing (2)

Weighted Frequency Matrix



Query Terms:
- Insulation
- Joint

DOCUMENTS:

'CM031.txt'

'CM046.txt'

'CM001.txt'

'CM029.txt'

'CM040.txt'

K>> return

TERMS:

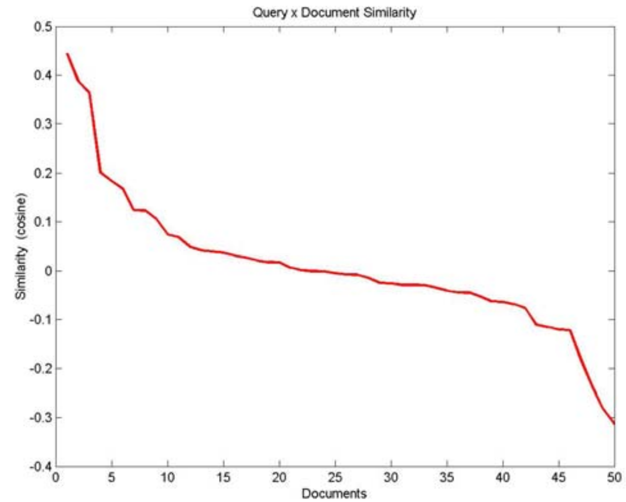
'joint'

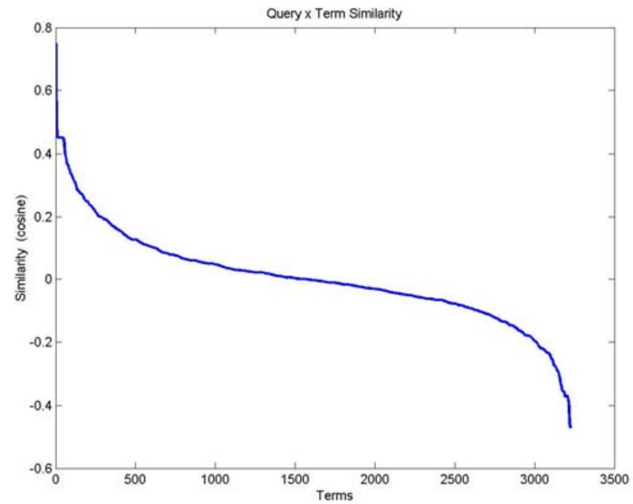
'insulation'

'roofing'

'expansion'

'saw'





Types of Text Data Mining

- Keyword-based association analysis
- Automatic document classification
- Similarity detection
  - Cluster documents by a common author
  - Cluster documents containing information from a common source
- Link analysis: unusual correlation between entities
- Sequence analysis: predicting a recurring event
- Anomaly detection: find information that violates usual patterns
- Hypertext analysis
  - Patterns in anchors/links
    - Anchor text correlations with linked objects

Keyword-Based Association Analysis

- Motivation
  - Collect sets of keywords or terms that occur frequently together and then find the association or correlation relationships among them
- Association Analysis Process
  - Preprocess the text data by parsing, stemming, removing stop words, etc.
  - Evoke association mining algorithms
    - Consider each document as a transaction
    - View a set of keywords in the document as a set of items in the transaction
  - Term level association mining
    - No need for human effort in tagging documents
    - The number of meaningless results and the execution time is greatly reduced

Text Classification(1)

- Motivation
  - Automatic classification for the large number of on-line text documents (Web pages, e-mails, corporate intranets, etc.)
- Classification Process
  - Data preprocessing
  - Definition of training set and test sets
  - Creation of the classification model using the selected classification algorithm
  - Classification model validation
  - Classification of new/unknown text documents
- Text document classification differs from the classification of relational data
  - Document databases are not structured according to attribute-value pairs

Text Classification(2)

- Classification Algorithms:
  - Support Vector Machines
  - K-Nearest Neighbors
  - Naïve Bayes
  - Neural Networks
  - Decision Trees
  - Association rule-based
  - Boosting

			#1	#2	#3	#4	#5
		# of documents	21,450	14,347	13,272	12,902	12,902
		# of training documents	14,704	10,667	9,610	9,603	9,603
		# of test documents	6,746	3,680	3,662	3,299	3,299
		# of categories	135	93	92	90	10
System	Type	Results reported by					
Word	(non-learning)	[Yang 1999]	.150	.310	.290		
	probabilistic	[Dumais et al. 1998]				.752	.815
	probabilistic	[Joachims 1998]					.720
	probabilistic	[Lam et al. 1997]	.443 (MF_1)				
PropBayes	probabilistic	[Lewis 1992a]	.650				
Bim	probabilistic	[Li and Yamanishi 1999]				.747	
	probabilistic	[Li and Yamanishi 1999]				.773	
Nb	probabilistic	[Yang and Liu 1999]				.795	
	decision trees	[Dumais et al. 1998]					.884
C4.5	decision trees	[Joachims 1998]					.794
Ind	decision trees	[Lewis and Ringuette 1994]	.670				
Swap-1	decision rules	[Apté et al. 1994]		.805			
Ripper	decision rules	[Cohen and Singer 1999]	.683	.811		.820	
SleepingExperts	decision rules	[Cohen and Singer 1999]	**.753**	.759		.827	
Dl-Esc	decision rules	[Li and Yamanishi 1999]				.820	
Charade	decision rules	[Moulinier and Ganascia 1996]		.738			
Charade	decision rules	[Moulinier et al. 1996]		.783 (F_1)			
Llsf	regression	[Yang 1999]		.855	.810		
Llsf	regression	[Yang and Liu 1999]				.849	
BalancedWinnow	on-line linear	[Dagan et al. 1997]	.747 (M)	.833 (M)			
Widrow-Hoff	on-line linear	[Lam and Ho 1998]				.822	
Rocchio	batch linear	[Cohen and Singer 1999]	.660	.748		.776	
FindSim	batch linear	[Dumais et al. 1998]				.617	.646
Rocchio	batch linear	[Joachims 1998]					.799
Rocchio	batch linear	[Lam and Ho 1998]				.781	
Rocchio	batch linear	[Li and Yamanishi 1999]				.625	
Classi	neural network	[Ng et al. 1997]		.802			
Nnet	neural network	[Yang and Liu 1999]				.838	
	neural network	[Wiener et al. 1995]			**.820**		
Gis-W	example-based	[Lam and Ho 1998]				.860	
k-NN	example-based	[Joachims 1998]					.823
k-NN	example-based	[Lam and Ho 1998]				.820	
k-NN	example-based	[Yang 1999]	.690	.852	**.820**		
k-NN	example-based	[Yang and Liu 1999]				.856	
	SVM	[Dumais et al. 1998]				.870	**.920**
SvmLight	SVM	[Joachims 1998]					.864
SvmLight	SVM	[Li and Yamanishi 1999]				.841	
SvmLight	SVM	[Yang and Liu 1999]				.859	
AdaBoost.MH	committee	[Schapire and Singer 2000]		**.860**			
	committee	[Weiss et al. 1999]				**.878**	
	Bayesian net	[Dumais et al. 1998]				.800	.850
	Bayesian net	[Lam et al. 1997]	.542 (MF_1)				

Document Clustering

- Motivation
  - Automatically group related documents based on their contents
  - No predetermined training sets or taxonomies
  - Generate a taxonomy at runtime
- Clustering Process
  - Data preprocessing: remove stop words, stem, feature extraction, lexical analysis, etc.
  - Hierarchical clustering: compute similarities applying clustering algorithms.
  - Model-Based clustering (Neural Network Approach): clusters are represented by “exemplars”. (e.g.: SOM)

Part II: Web Mining

Zhou Shuigeng

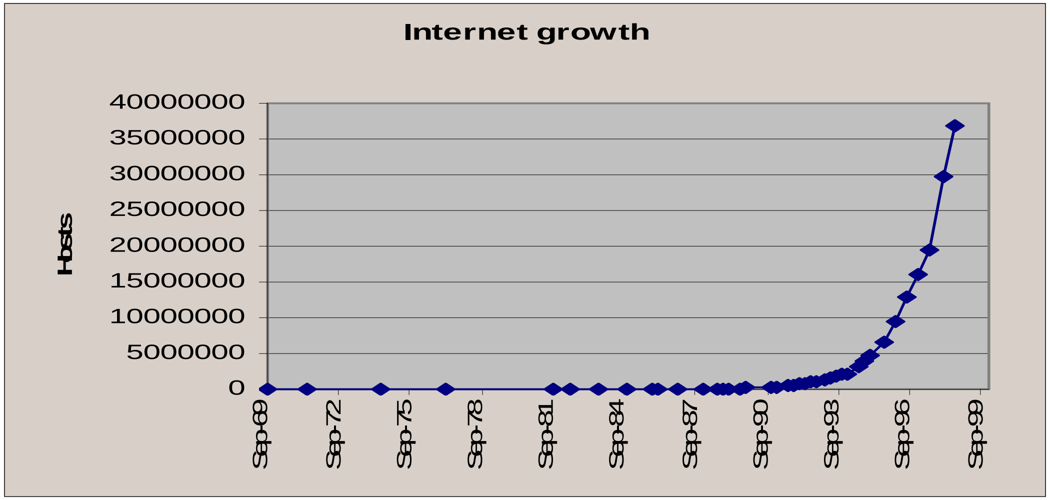
May 28, 2004

Mining the World-Wide Web

- The WWW is huge, widely distributed, global information service center for
  - Information services: news, advertisements, consumer information, financial management, education, government, e-commerce, etc.
  - Hyper-link information
  - Access and usage information
- WWW provides rich sources for data mining
- Challenges
  - Too huge for effective data warehousing and data mining
  - Too complex and heterogeneous: no standards and structure

Mining the World-Wide Web

- Growing and changing very rapidly



- Broad diversity of user communities
- Only a small portion of the information on the Web is truly relevant or useful
  - 99% of the Web information is useless to 99% of Web users
  - How can we find high-quality Web pages on a specified topic?

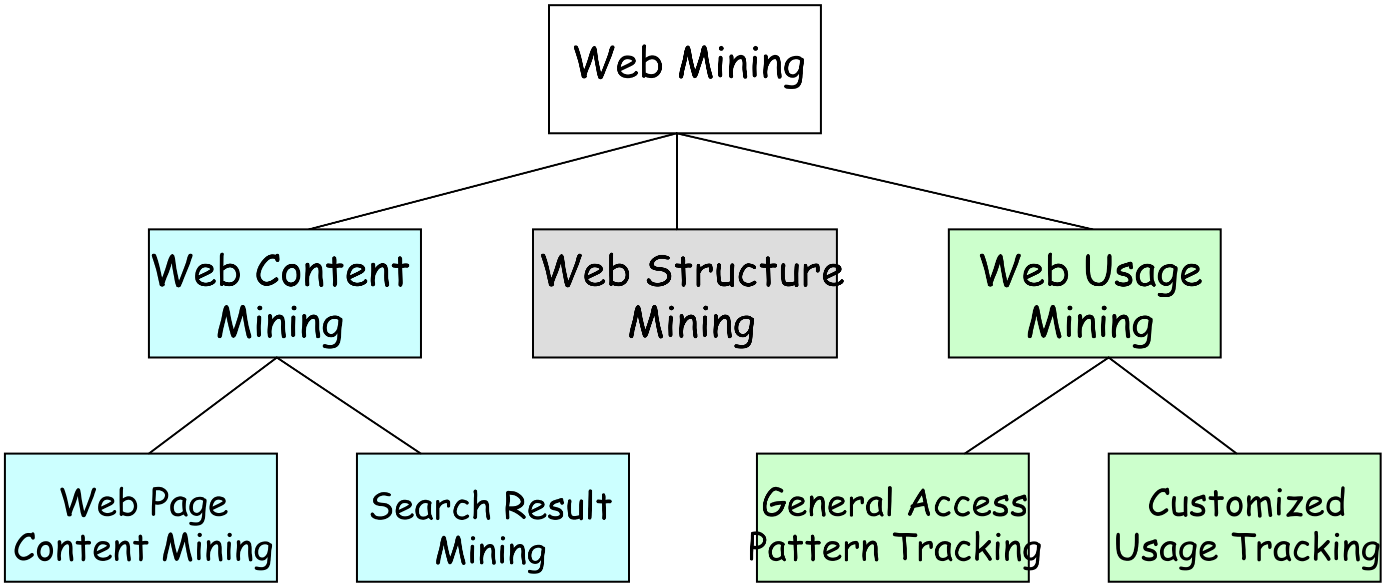
Web search engines

- Index-based: search the Web, index Web pages, and build and store huge keyword-based indices
- Help locate sets of Web pages containing certain keywords
- Deficiencies
  - A topic of any breadth may easily contain hundreds of thousands of documents
  - Many documents that are highly relevant to a topic may not contain keywords defining them (polysemy)

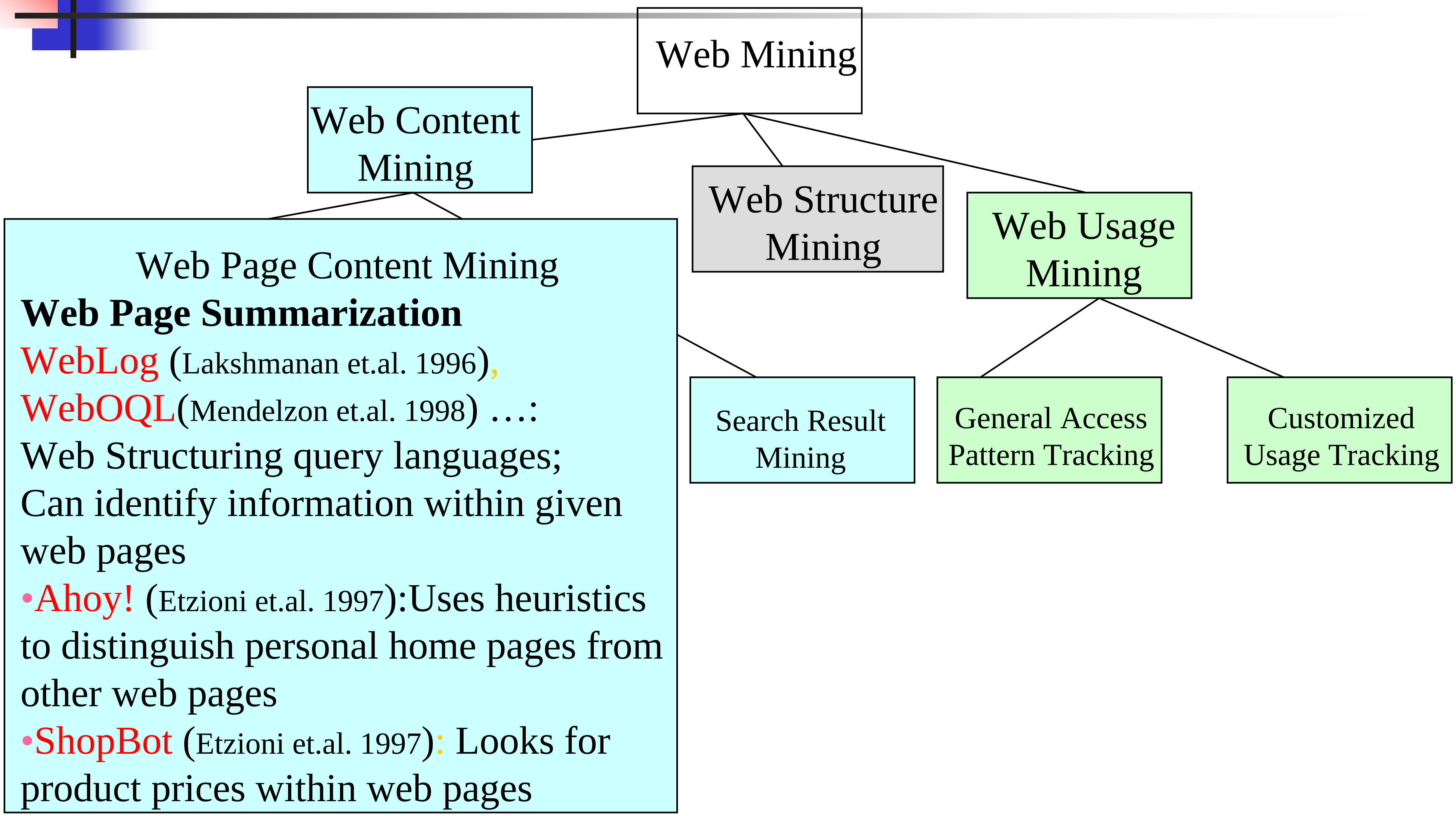
Web Mining: A more challenging task

- Searches for
  - Web access patterns
  - Web structures
  - Regularity and dynamics of Web contents
- Problems
  - The "abundance" problem
  - Limited coverage of the Web: hidden Web sources, majority of data in DBMS
  - Limited query interface based on keyword-oriented search
  - Limited customization to individual users

Web Mining Taxonomy



Mining the World-Wide Web



Mining the World-Wide Web

Web Mining

Web Content Mining

Web Structure Mining

Web Usage Mining

Web Page Content Mining

Search Result Mining

Search Engine Result Summarization

- Clustering Search Result (*Leouski and Croft, 1996, Zamir and Etzioni, 1997*):
Categorizes documents using phrases in titles and snippets

General Access Pattern Tracking

Customized Usage Tracking

Mining the World-Wide Web

Web Mining

- Web Content Mining
  - Web Page Content Mining
  - Search Result Mining
- Web Structure Mining
- Web Usage Mining
  - General Access Pattern Tracking
  - Customized Usage Tracking

Web Structure Mining

Using Links

- PageRank (Brin et al., 1998)
- CLEVER (Chakrabarti et al., 1998)

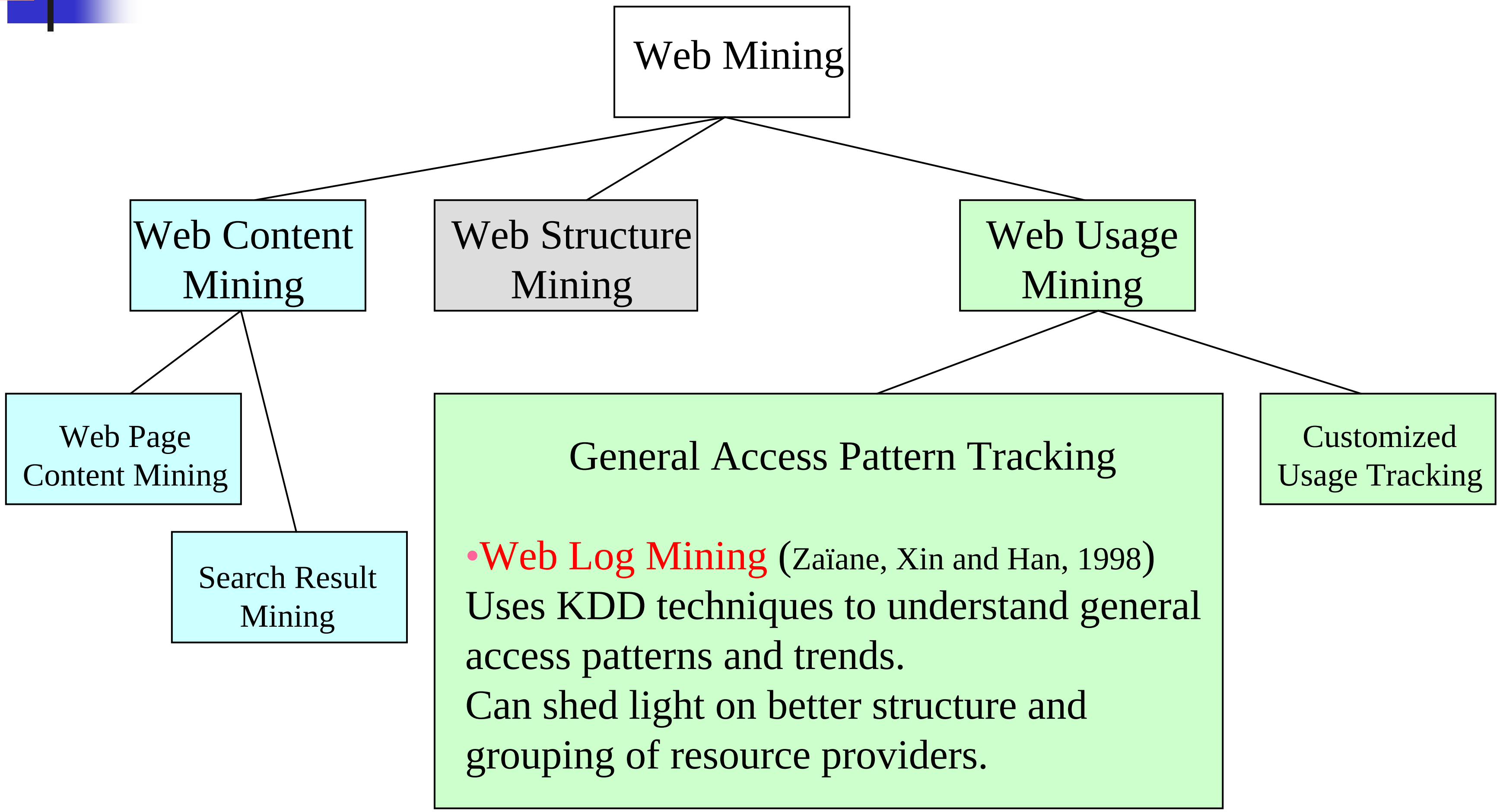
Use interconnections between web pages to give weight to pages.

Using Generalization

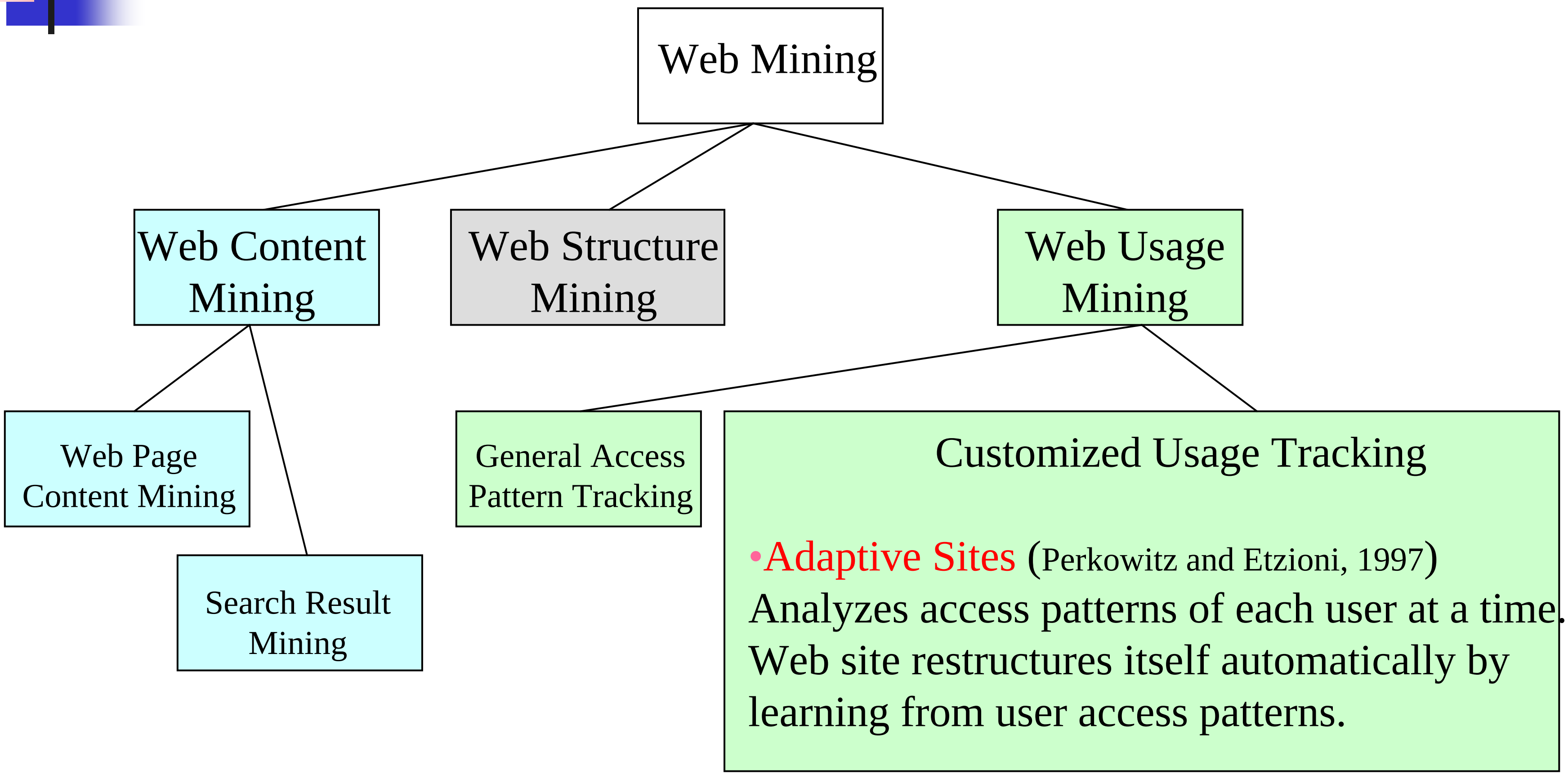
- MLDB (1994), VWV (1998)

Uses a multi-level database representation of the Web. Counters (popularity) and link lists are used for capturing structure.

Mining the World-Wide Web



Mining the World-Wide Web



Search Engine Topics

- Text-based Search Engines
  - Document based
    - Ranking: TF-IDF, Vector Space Model
    - No relationship between pages modeled
    - Cannot tell which page is important without query
- Link-based search engines: Google, Hubs and Authorities Techniques
  - Can pick out *important* pages

The PageRank Algorithm

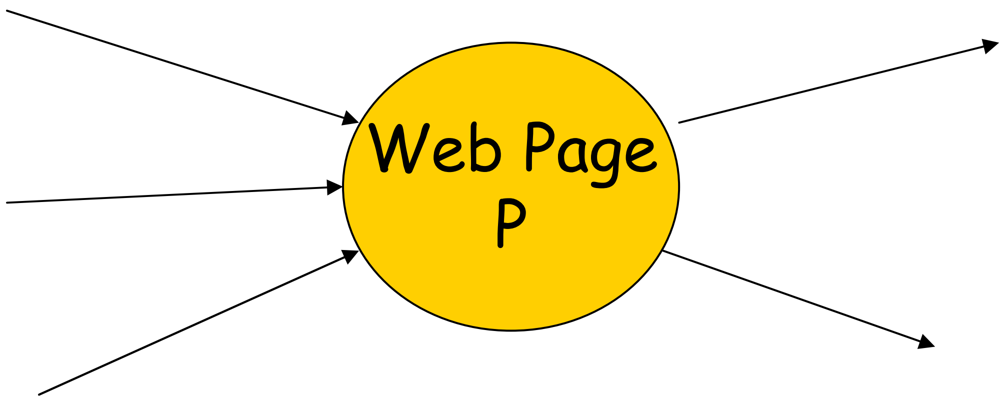
- Fundamental question to ask
  - What is the importance level of a page P, $I(P)$
- Information Retrieval
  - Cosine + TF IDF → does not give related hyperlinks
- Link based
  - Important pages (nodes) have many other links point to it
  - Important pages also point to other important pages

The Google Crawler Algorithm

- *"Efficient Crawling Through URL Ordering",*
  - *Junghoo Cho, Hector Garcia-Molina, Lawrence Page, Stanford*
  - http://www.www8.org
  - http://www-db.stanford.edu/~cho/crawler-paper/
- "Modern Information Retrieval", BY-RN
  - Pages 380—382
- Lawrence Page, Sergey Brin. The Anatomy of a Search Engine. *The Seventh International WWW Conference* (WWW 98). Brisbane, Australia, April 14-18, 1998.
  - http://www.www7.org

Back Link Metric

IB(P)=3



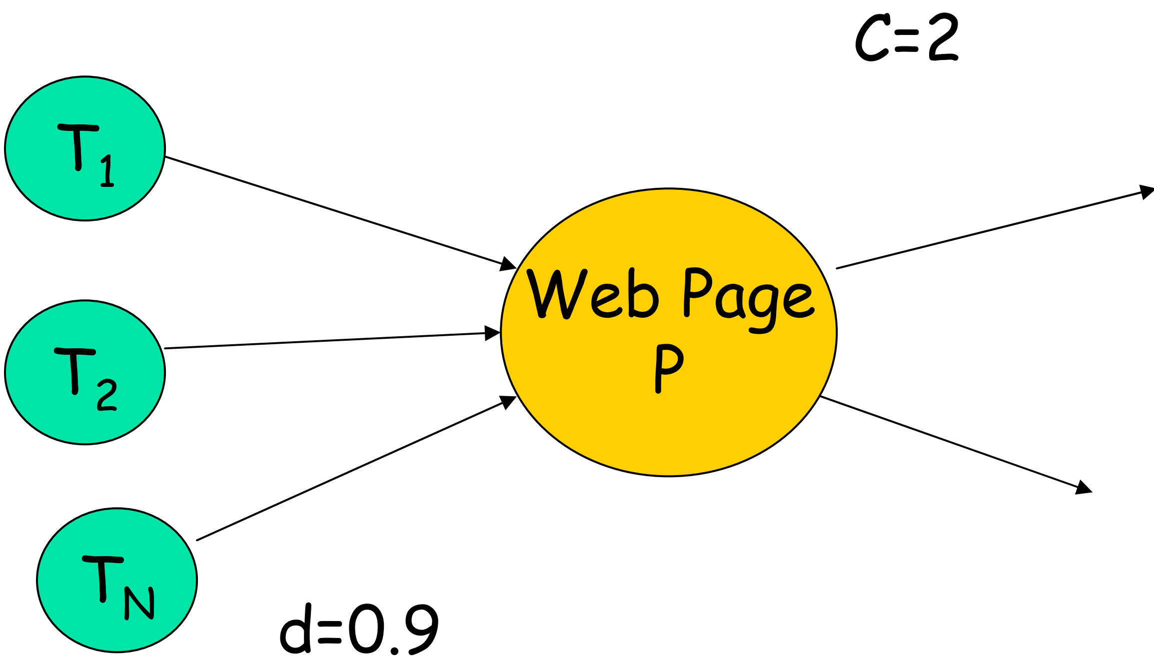
- IB(P) = total number of backlinks of P
- IB(P) impossible to know, thus, use IB'(P) which is the number of back links crawler has seen so far

Page Rank Metric

Let 1-*d* be probability that user randomly jump
to page P;

"d" is the damping factor

Let C_i be the number of
out links from each T_i



$$IR(P) = (1-d) + d * \sum_{i=1}^{N} IR(T_i) / C_i$$

Matrix Formulation

- Consider a random walk on the web *(denote IR(P) by r(P))*
  - Let B_{ij} = probability of going directly from *i* to *j*
  - Let r_i be the limiting probability (page rank) of being at page *i*

$$\begin{pmatrix} b_{11} & b_{21} & \dots & b_{n1} \\ b_{12} & b_{22} & \dots & b_{n2} \\ \dots & \dots & \dots & \dots \\ b_{1n} & b_{2n} & \dots & b_{nn} \end{pmatrix} \begin{pmatrix} r_1 \\ r_2 \\ \dots \\ r_n \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \dots \\ r_n \end{pmatrix} \qquad B^T r = r$$

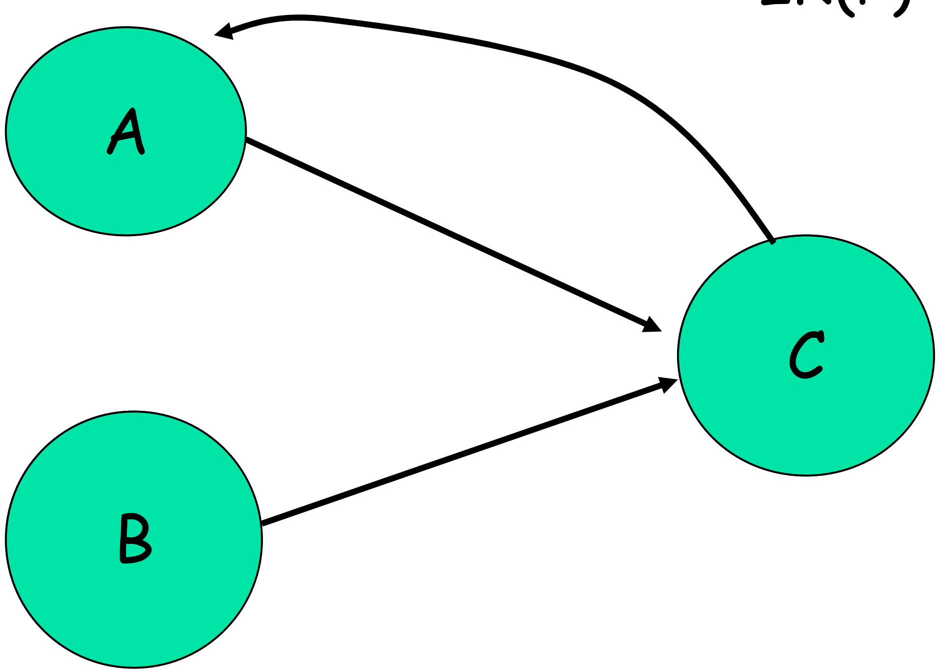
Thus, the final page rank *r* is a principle eigenvector of B^T

How to compute page rank?

- For a given network of web pages,
  - Initialize page rank for all pages (to one)
  - Set parameter (d=0.90)
  - Iterate through the network, L times

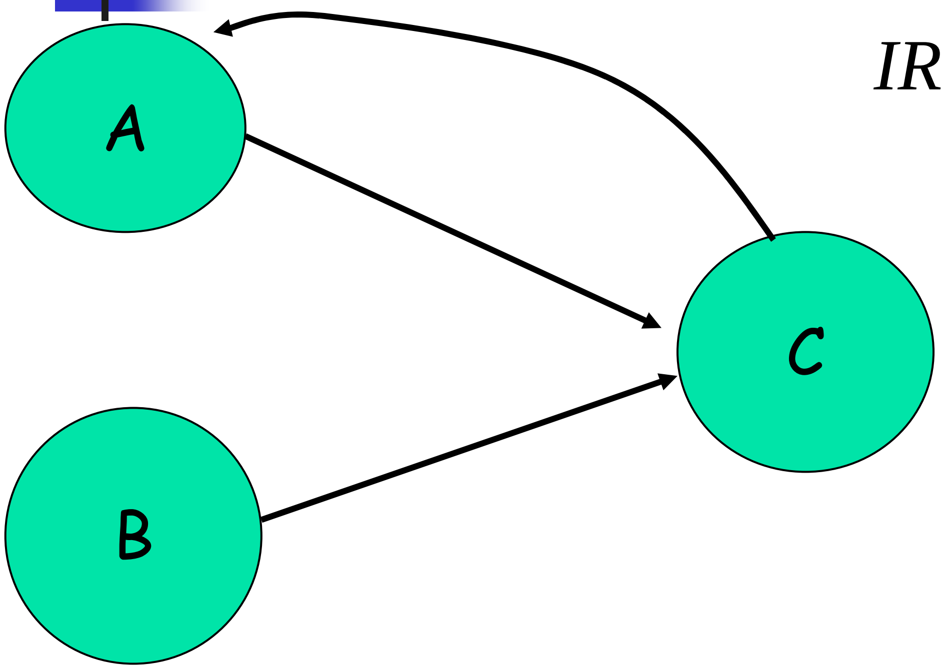
Example: iteration K=1

IR(P)=1/3 for all nodes, d=0.9



node	IP
A	1/3
B	1/3
C	1/3

Example: k=2



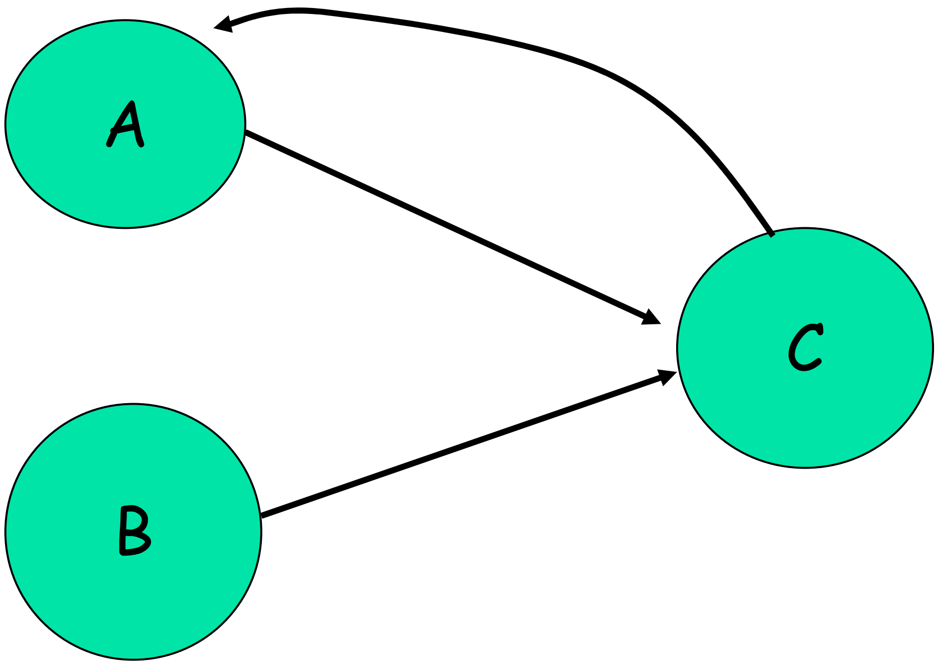
$$IR(P) = 0.1 + 0.9 * \sum_{i=1}^{l} IR(T_i) / C_i$$

l is the in-degree of P

node	IP
A	0.4
B	0.1
C	0.55

Note: A, B, C's IP values are Updated in order of A, then B, then C Use the new value of A when calculating B, etc.

Example: k=2 (normalize)



node	IP
A	0.38
B	0.095
C	0.52

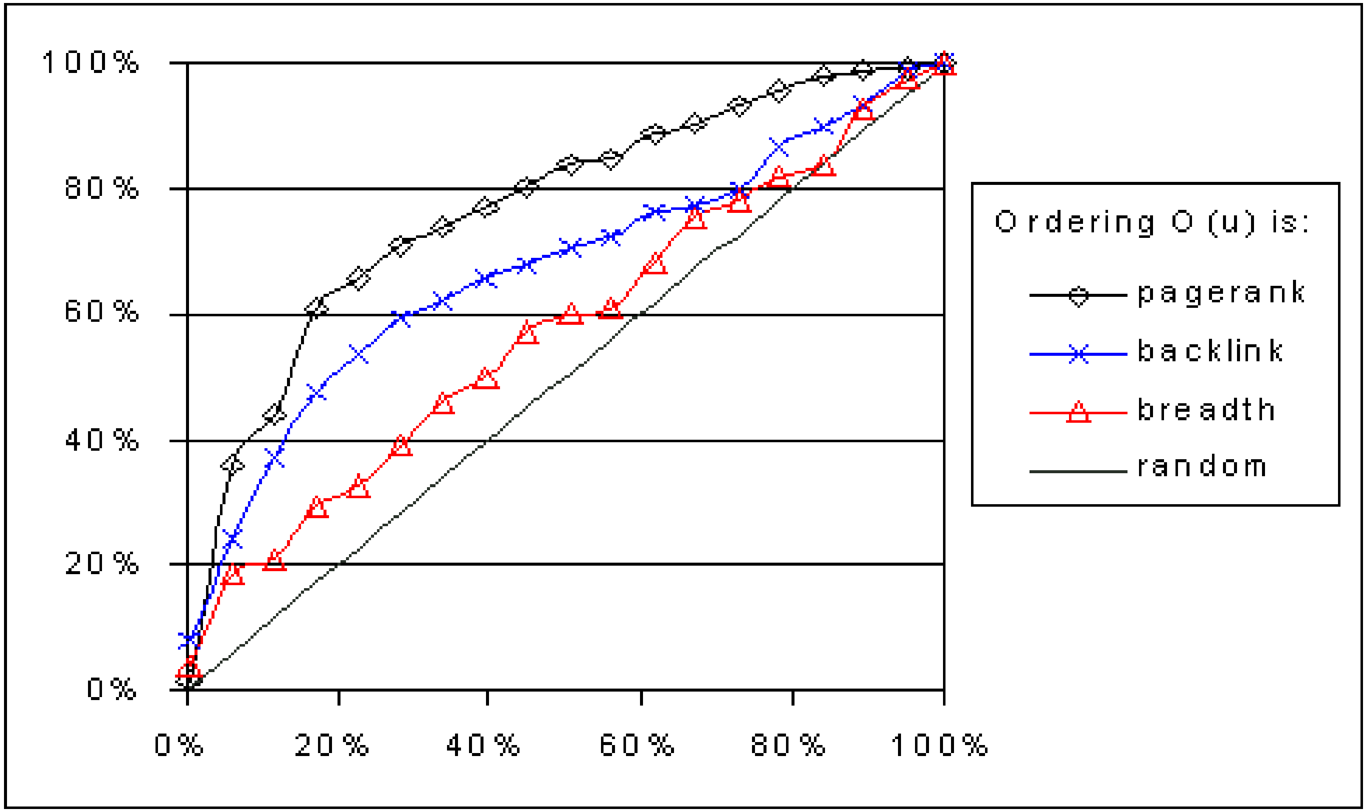
Crawler Control

- All crawlers maintain several queues of URL's to pursue next
  - Google initially maintains 500 queues
  - Each queue corresponds to a web site pursuing
- Important considerations:
  - Limited buffer space
  - Limited time
  - Avoid overloading target sites
  - Avoid overloading network traffic

Crawler Control

- Thus, it is important to visit important pages *first*
- Let G be a lower bound threshold on I(P)
- *Crawl and Stop*
  - Select only pages with IP>G to crawl,
  - Stop after crawled *K* pages

Test Result: 179,000 pages



Percentage of Stanford Web crawled vs. P_{ST} – the percentage of hot pages visited so far

Google Algorithm (very simplified)

- First, compute the page rank of each page on WWW
  - Query independent
- Then, in response to a query *q*, return pages that contain *q* and have highest page ranks
- A problem/feature of Google: favors big commercial sites

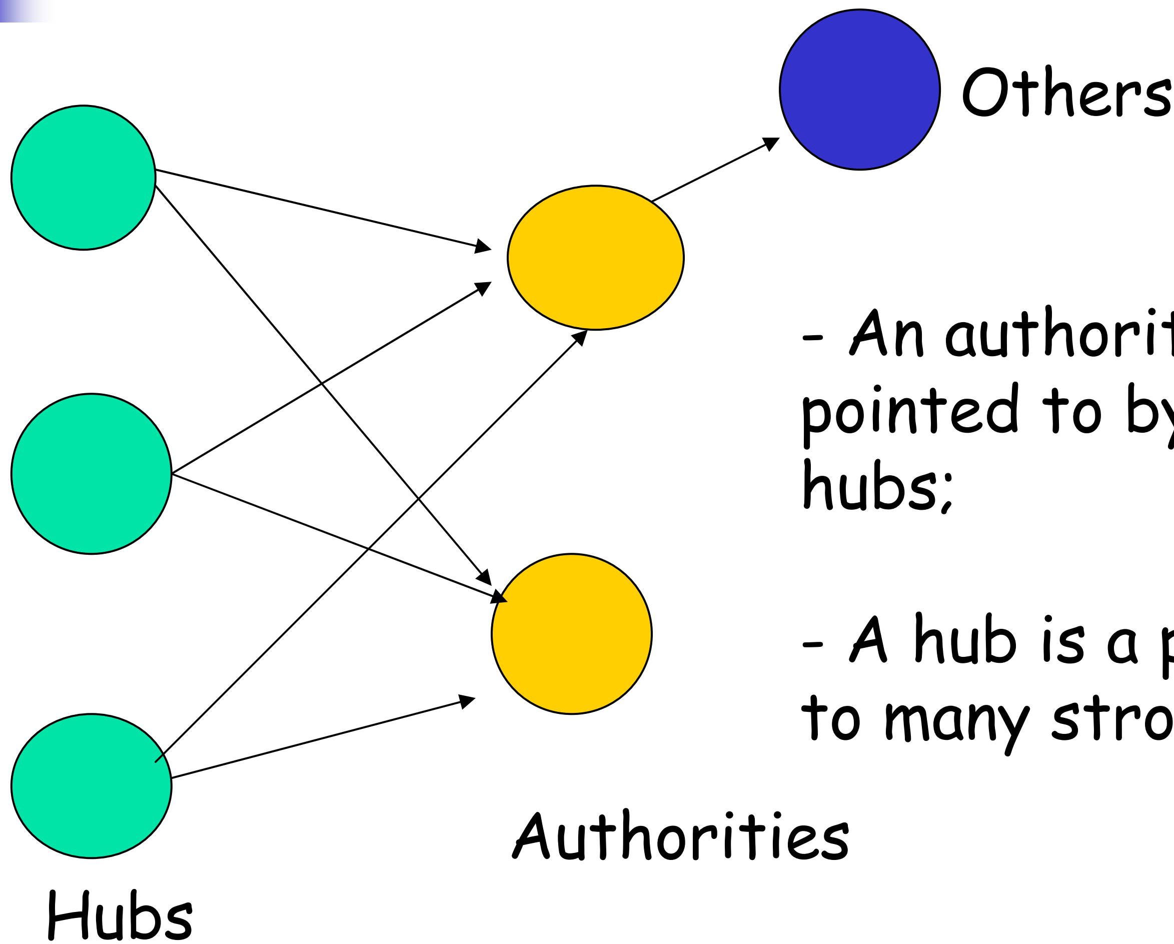
How powerful is Google?

- A PageRank for 26 million web pages can be computed in a few hours on a medium size workstation
- Currently has indexed a total of 1.3 Billion pages

Hubs and Authorities 1998

- Kleinburg, Cornell University
- http://www.cs.cornell.edu/home/kleinber/
- Main Idea: type "java" in a text-based search engine
  - Get 200 or so pages
  - Which one's are authoritive?
    - http://java.sun.com
  - What about others?
    - www.yahoo.com/Computer/ProgramLanguages

Hubs and Authorities



- An authority is a page pointed to by many strong hubs;

- A hub is a page that points to many strong authorities

H&A Search Engine Algorithm

- First submit query Q to a text search engine
- Second, among the results returned
  - select ~200, find their neighbors,
  - compute *Hubs* and *Authorities*
- Third, return *Authorities* found as final result
- *Important Issue*: how to find *Hubs* and *Authorities?*

Link Analysis: weights

- Let B_{ij}=1 if i links to j, 0 otherwise
  - h_i=hub weight of page i
  - a_i = authority weight of page i
  - *Weight normalization*

$$\sum_{i=1}^{N} (h_i)^2 = 1$$

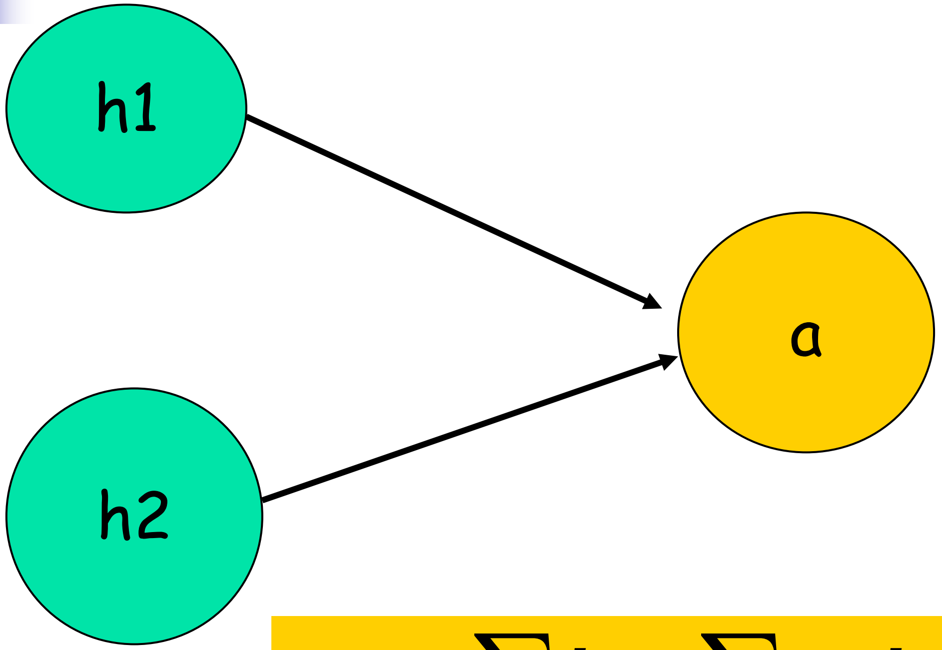
$$\sum_{i=1}^{N} (a_i)^2 = 1 \qquad (3)$$

But, for simplicity, we will use

$$\sum_{i=1}^{N} h_i = 1$$

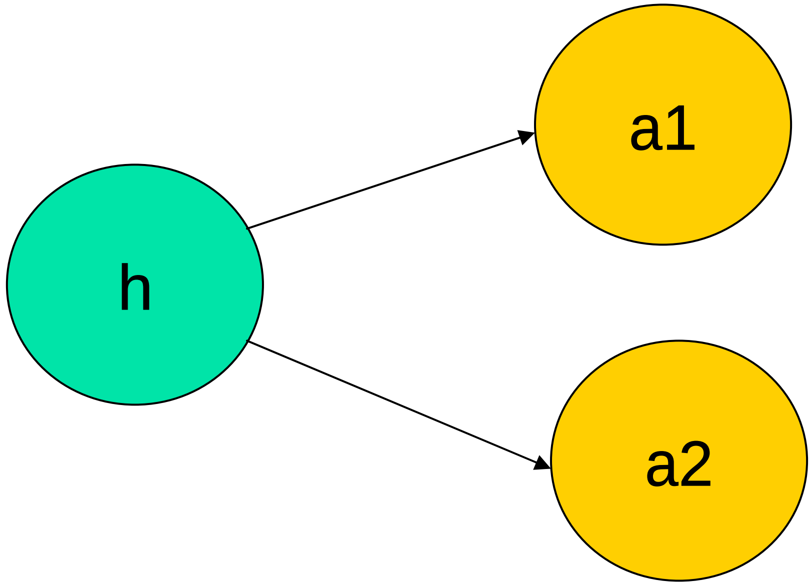
$$\sum_{i=1}^{N} a_i = 1 \qquad (3')$$

Link Analysis: update a-weight



$$a_i \leftarrow \sum_{B_{ji} \neq 0} h_j = \sum B_{ji} h_j = B^T h \qquad (1)$$

Link Analysis: update h-weight



$$h_i \leftarrow \sum_{B_{ij} \neq 0} a_j = \sum B_{ij} a_j = Ba \qquad (2)$$

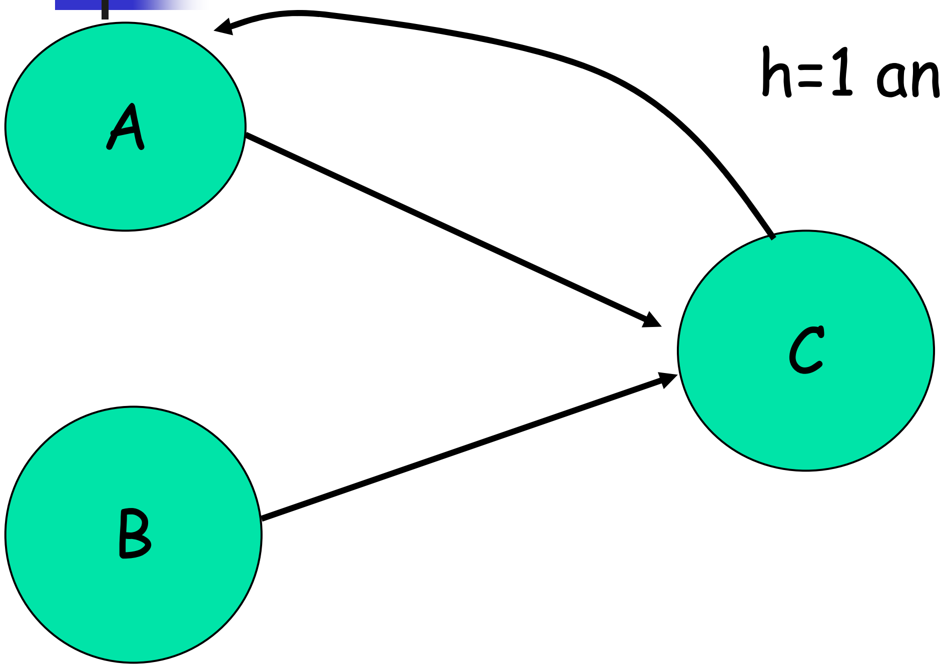
H&A: algorithm

1. Set value for K, the number of iterations
2. Initialize all *a* and *h* weights to 1
3. For l=1 to K, do
   a. Apply equation (1) to obtain new a_i weights
   b. Apply equation (2) to obtain all new h_i weights, using the new a_i weights obtained in the last step
   c. Normalize a_i and h_i weights using equation (3)

DOES it converge?

- Yes, the Kleinberg paper includes a proof
- Needs to know Linear algebra and eigenvector analysis
- We will skip the proof but only using the results:
  - The a and h weight values will converge after sufficiently large number of iterations, K.

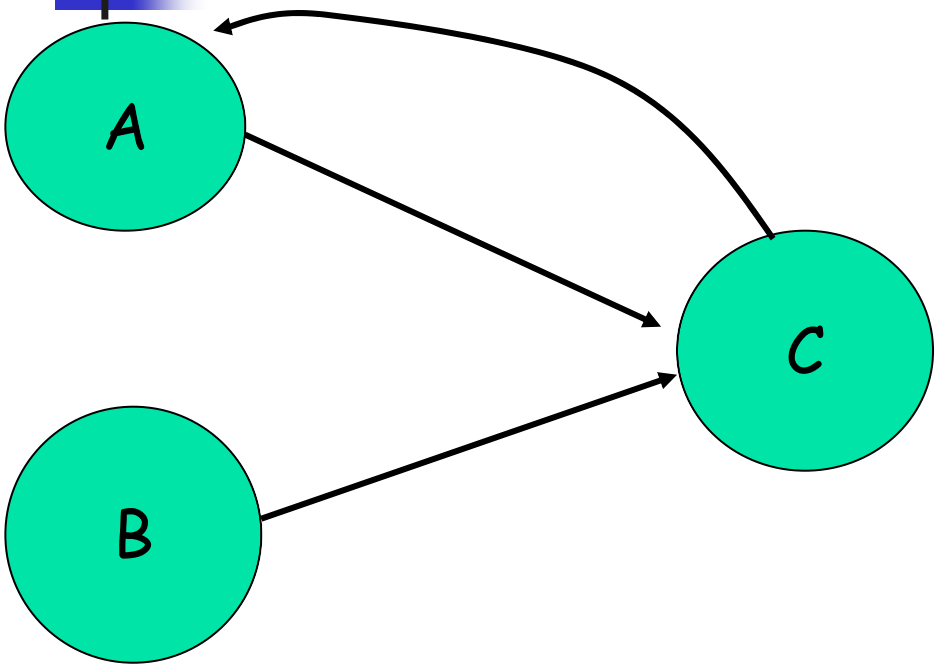
Example: K=1



h=1 and a=1 for all nodes

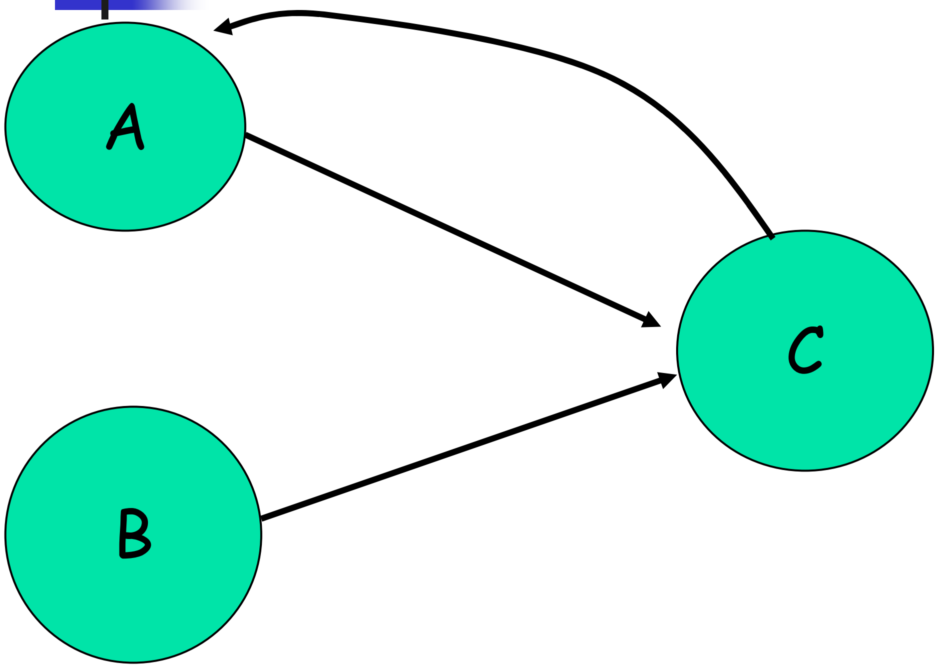
node	a	h
A	1	1
B	1	1
C	1	1

Example: k=1 (update a)



node	a	h
A	1	1
B	0	1
C	2	1

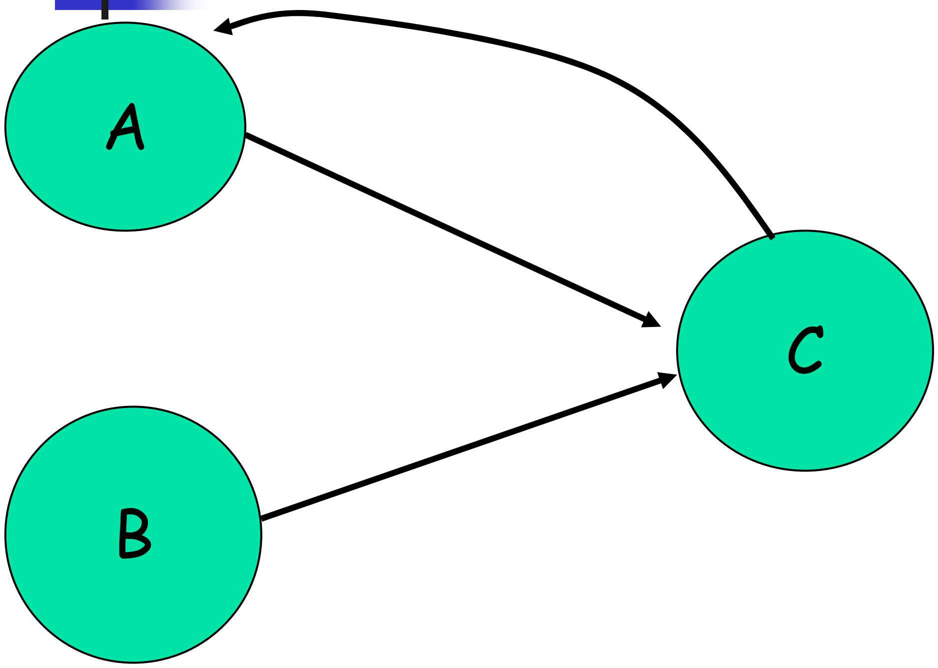
Example: k=1 (update h)



node	a	h
A	1	2
B	0	2
C	2	1

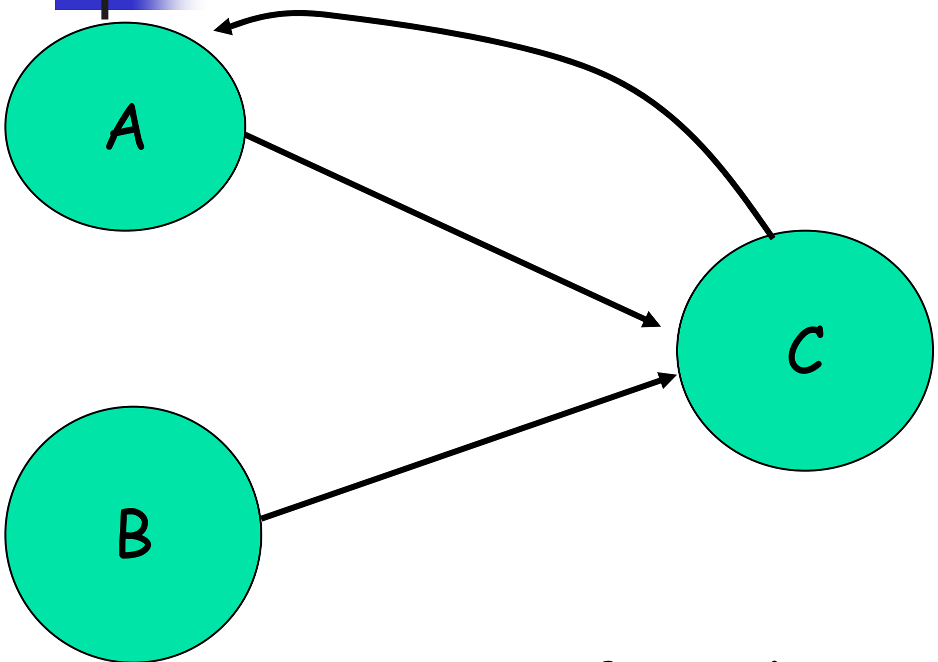
Example: k=1 (normalize)

Use Equation (3')



node	a	h
A	1/3	2/5
B	0	2/5
C	2/3	1/5

Example: k=2 (update a, h,normalize)



Use Equation (1)

node	a	h
A	1/5	4/9
B	0	4/9
C	4/5	1/9

If we choose a threshold of $\frac{1}{2}$, then C is an Authority, and there are no hubs.

Search Engine Using H&A

- For each query q,
  - Enter q into a text-based search engine
  - Find the top 200 pages
  - Find the neighbors of the 200 pages by one link, let the set be S
  - Find hubs and authorities in S
  - Return authorities as final result

Summary

- Link based analysis is very powerful in find out the important pages
- Models the web as a graph, and based on in-degree and out-degree
- Google: crawl only important pages
- H&A: post analysis of search result

Automatic Classification of Web Documents

- Assign a class label to each document from a set of predefined topic categories
- Based on a set of examples of preclassified documents
- Example
  - Use Yahoo!'s taxonomy and its associated documents as training and test sets
  - Derive a Web document classification scheme
  - Use the scheme classify new Web documents by assigning categories from the same taxonomy
- Keyword-based document classification methods
- Statistical models

Web Usage Mining

- Mining Web log records to discover user access patterns of Web pages
- Applications
  - Target potential customers for electronic commerce
  - Enhance the quality and delivery of Internet information services to the end user
  - Improve Web server system performance
  - Identify potential prime advertisement locations
- Web logs provide rich information about Web dynamics
  - Typical Web log entry includes the URL requested, the IP address from which the request originated, and a timestamp

Techniques for Web usage mining

- Construct multidimensional view on the Weblog database
  - Perform multidimensional OLAP analysis to find the top *N* users, top *N* accessed Web pages, most frequently accessed time periods, etc.
- Perform data mining on Weblog records
  - Find association patterns, sequential patterns, and trends of Web accessing
  - May need additional information,e.g., user browsing sequences of the Web pages in the Web server buffer
- Conduct studies to
  - Analyze system performance, improve system design by Web caching, Web page prefetching, and Web page swapping

Mining the World-Wide Web

- Design of a Web Log Miner
  - Web log is filtered to generate a relational database
  - A data cube is generated form database
  - OLAP is used to drill-down and roll-up in the cube
  - OLAM is used for mining interesting knowledge