PURDUE UNIVERSITY

# CS54701: Information Retrieval

*Course Review*

26 April 2016

Prof. Chris Clifton

Indiana Center for Database Systems ™

## Ad-hoc IR: Introduction

### Ad-hoc Information Retrieval:

- Search a collection of documents to find relevant documents that satisfy different information needs (i.e., queries)
  - **Relatively Stable**
  - **Changes**
  - Queries are created and used dynamically; change fast
  - "Ad-hoc": formed or used for specific or immediate problems or needs" – Merriam-Webster's collegiate Dictionary

### Ad-hoc IR vs. Filtering

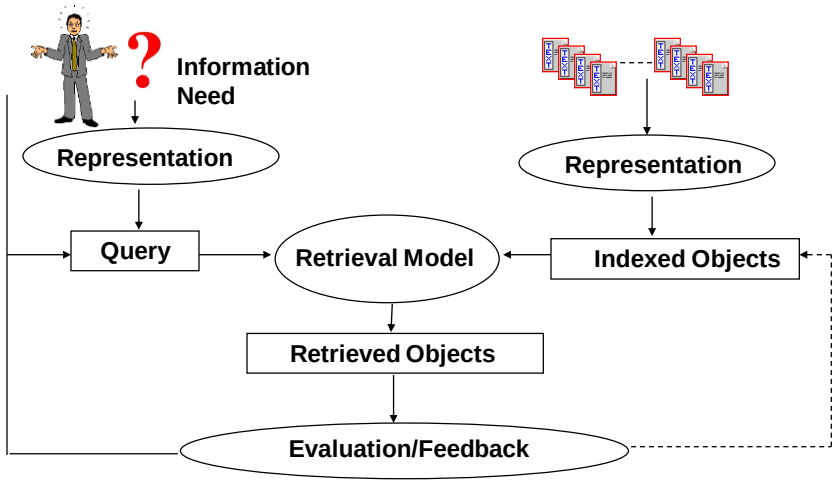
- Filtering: Queries are stable (e.g., Asian High-Tech) while the collection changes (e.g., news)
- More for filtering in later lectures

# Ad-hoc IR: Terminologies

*Terminologies:*

- Query
  - Representative data of user's information need: text (default) and other media
- Document
  - Data candidate to satisfy user's information need: text (default) and other media
- Database|Collection|Corpus
  - A set of documents
- Corpora
  - A set of databases
  - Valuable corpora from TREC (Text Retrieval Evaluation Conference)

# AD-hoc IR: Basic Process



# Text Representation: Indexing

- Indexing
  - Associate document/query with a set of keys
- Manual or human Indexing
  - Indexers assign keywords or key concepts (e.g., libraries, Medline, Yahoo!); often small vocabulary
  - Significant human efforts, may not be thorough
- Automatic Indexing
  - Index program assigns words, phrases or other features; often large vocabulary
  - No human effort ➔ low cost

# Text Representation: Indexing

## Statistical Properties of Text

Observations from language/corpus independent features

- A few words occur very frequently (High Peak)
  - Top 2 words: 8%-15% (e.g., words that carry no semantic meanings like “the”, “to”)
- Most words occur rarely (Heavy Tail)
- Representative words often in the middle
  - e.g., market and stock for WSJ
- Rules formally describe word occurrence patterns: Zipf’s law, Heaps’ Law

# Text Representation: Indexing

## Statistical Properties of Text

Zipf's law: relate a term's frequency to its rank

- Rank all terms with their frequencies in descending order, for a term at a specific rank (e.g., *r*) collects and calculates

$f_r$ : **term frequency**

$p_r = \frac{f_r}{N}$ : **relative term frequency**

**Total number of words**

- Zipf's law (by observation):

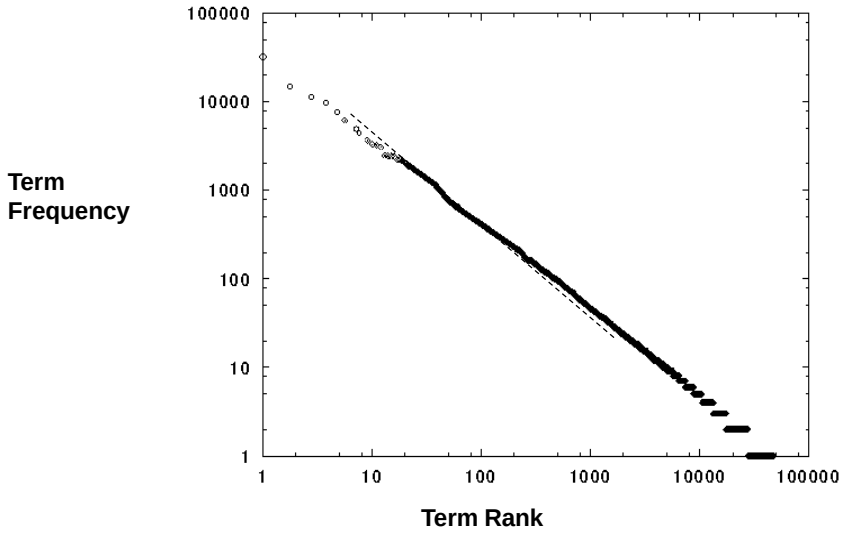
$$p_r = A/r \quad A \approx 0.1$$

So $p_r = \frac{f_r}{N} = \frac{A}{r} \Rightarrow rf_r = AN \Rightarrow \log(r) = -\log(f_r) + \log(AN)$

So Rank X Frequency = Constant

# Text Representation: Indexing

## Statistical Properties of Text



# Text Representation: Text Preprocessing

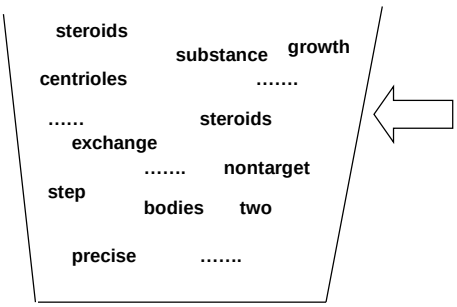
Text Preprocessing: extract representative index terms

- Parse query/document for useful structure
  - E.g., title, anchor text, link, tag in xml…..
- Tokenization
  - For most western languages, words separated by spaces; deal with punctuation, capitalization, hyphenation
  - For Chinese, Japanese: more complex word segmentation…
- Remove stopwords: (remove “the”, “is”,..., existing standard list)
- Morphological analysis (e.g., stemming):
  - Stemming: determine stem form of given inflected forms
- Other: extract phrases; decompounding for some European languages “*rörelseuppskattningssökningsintervallsinställningar*”

# Text Representation: Bag of Words

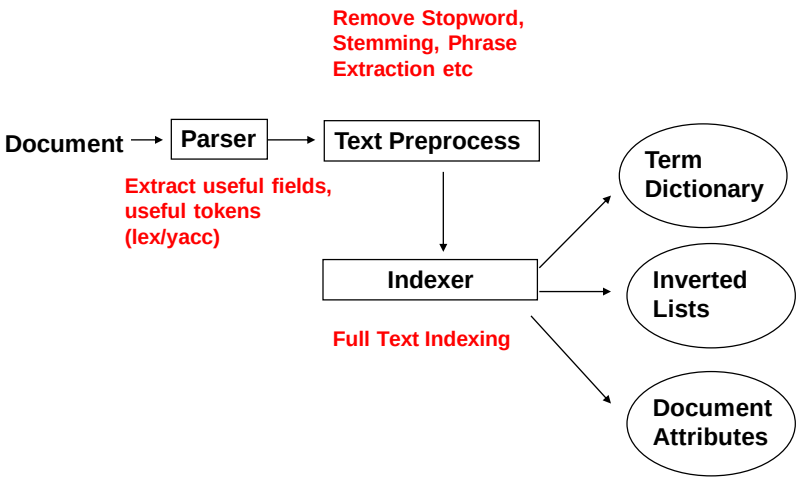
The simplest text representation: “bag of words”

- Query/document: a bag that contains words in it
- Order among words is ignored




| | | | |
|---|---|---|---|
| 3 steroids | 1 cilia-bearing | 1 precise | 1 two |
| 2 centrioles | 1 different | 1 receptor | 1 unexpected |
| 1 affect | 1 exchange | 1 regularly | 1 vitally |
| 1 already | 1 exogenous | 1 reveal | 1 way |
| 1 Although | 1 fluorescent | 1 Specific | |
| 1 antibodies | 1 growth | 1 step | |
| 1 basal | 1 identity | 1 substance | |
| 1 bodies | 1 level | 1 suggests | |
| 1 cell | 1 localization | 1 target | |
| 1 cells | 1 nontarget | 1 technique | |

# Text Representation: Process of Indexing




# AD-hoc IR: Overview of Retrieval Model

## Retrieval Model

Determine whether a document is relevant to query

- Relevance is difficult to define
  - Varies by judgers
  - Varies by context (i.e., jointly by a set of documents and queries)
- Different retrieval methods estimate relevance differently
  - Word occurrence of document and query
  - In probabilistic framework, P(query|document) or P(Relevance|query,document)
  - Estimate semantic consistency between query and document

# Retrieval Models: Latent Semantic Indexing

Latent Semantic Indexing (LSI): Explore correlation between terms and documents

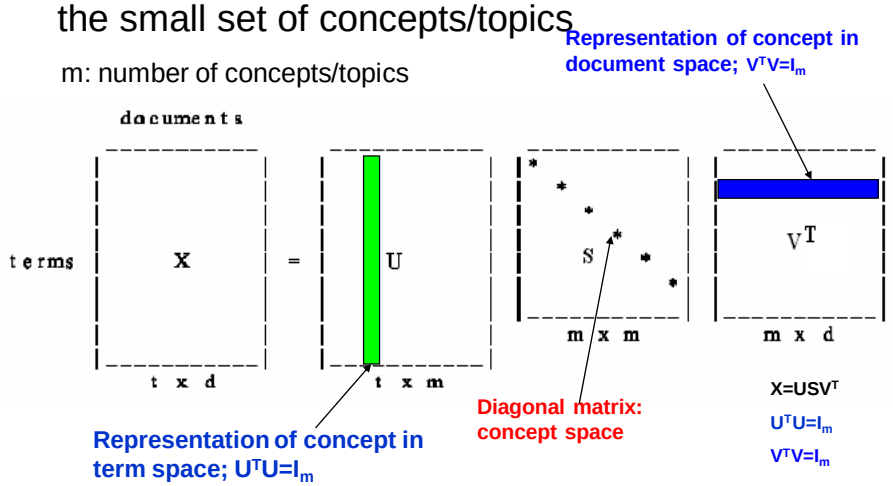
- Two terms are correlated (may share similar semantic concepts) if they often co-occur
- Two documents are correlated (share similar topics) if they have many common words

Latent Semantic Indexing (LSI): Associate each term and document with a small number of semantic concepts/topics

# Retrieval Models: Latent Semantic Indexing

Using singular value decomposition (SVD) to find the small set of concepts/topics

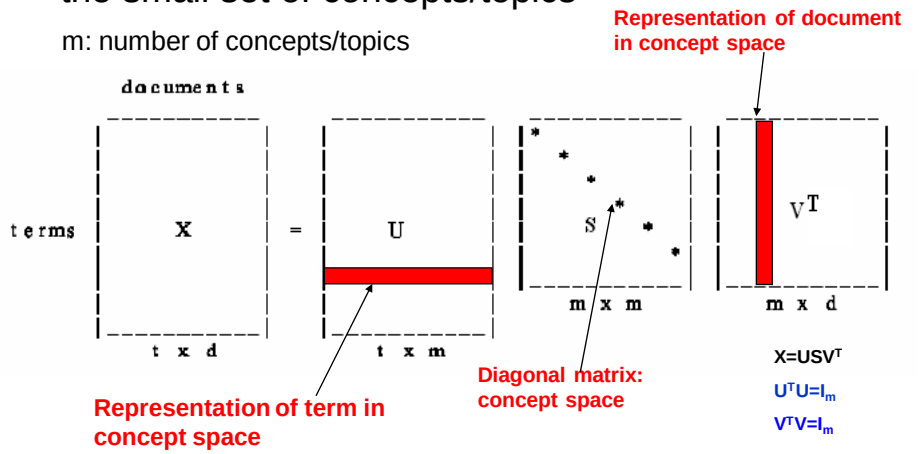
m: number of concepts/topics



# Retrieval Models: Latent Semantic Indexing

Using singular value decomposition (SVD) to find the small set of concepts/topics

m: number of concepts/topics




# Retrieval Models: Latent Semantic Indexing

Query: Machine Learning Protein

| | C1 | C2 | C3 | C4 | B1 | B2 | B3 |
|---|---|---|---|---|---|---|---|
| information | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| retrieval | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| machine | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| learning | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| system | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| protein | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| gene | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| mutation | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| expression | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

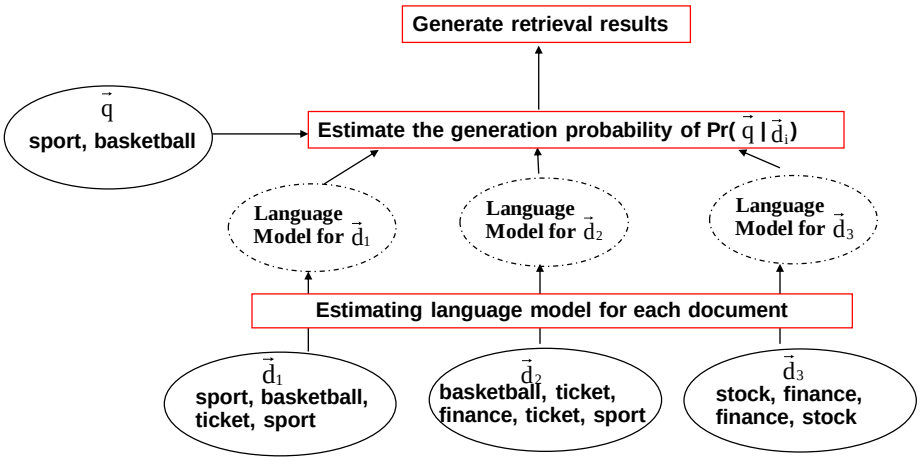
Representation of the query in the term vector space:

$[0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0]^T$

# Introduction to Language Models:

- A document language model defines a probability distribution over indexed terms
  - E.g., the probability of generating a term
  - Sum of the probabilities is 1
- A query can be seen as observed data from unknown models
  - Query also defines a language model (more on this later)
- How might the models be used for IR?
  - Rank documents by Pr($\vec{q} \mid \vec{d}_i$)
  - Rank documents by language models of $\vec{q}$ and $\vec{d}_i$ based on kullback-Leibler (KL) divergence between the models (come later)

# Language Model for IR: Example




19

# Multinomial/Unigram Language Models

- Language model built by multinomial distribution on single terms (i.e., unigram) in the vocabulary

Examples:

Five words in vocabulary (sport, basketball, ticket, finance, stock)

For a document $\vec{d}_i$, its language mode is:

{$P_i$("sport"), $P_i$("basketball"), $P_i$("ticket"), $P_i$("finance"), $P_i$("stock")}

Formally:

The language model is: {$P_i(w)$ for any word w in vocabulary V}

$$\sum_k P_i(w_k) = 1 \qquad 0 \le P_i(w_k) \le 1$$

# Maximum Likelihood Estimation (MLE)

**Maximum Likelihood Estimation:**

- Find model parameters that make generation likelihood reach maximum:

$$M^* = \text{argmax}_M \Pr(D|M)$$

There are K words in vocabulary, $w_1 \ldots w_K$ (e.g., 5)

Data: one document $\vec{d}_i$ with counts $tf_i(w_1)$, …, $tf_i(w_K)$, and length $|\vec{d}_i|$

Model: multinomial M with parameters $\{p_i(w_k)\}$

Likelihood: $\Pr(\vec{d}_i | M)$

$$M^* = \text{argmax}_M \Pr(\vec{d}_i | M)$$

# Maximum A Posterior (MAP) Estimation

**Maximum A Posterior Estimation:**

- Select a model that maximizes the probability of model given observed data

  $M^*=\text{argmax}_M \Pr(M|D)=\text{argmax}_M \Pr(D|M)\Pr(M)$

  - Pr(M): Prior belief/knowledge
  - Use prior Pr(M) to avoid zero probabilities

A specific examples:

Only two words in vocabulary (sport, business)

For a document $\vec{d}_i$:

**Prior Distribution**

$$\Pr(M \mid \vec{d}_i) = \binom{\vec{d}_i}{tf_i(w_1)\, tf_i(w_2)} p_i(w_1)^{tf_i(w_1)}\, p_i(w_2)^{tf_i(w_2)}\, \Pr(M)$$

22

# Dirichlet Smoothing & TF-IDF

**Dirichlet Smoothing:**

**Irrelevant part**

$$\log p(q \mid \vec{d}_i) = \sum_{k=1} tf_q(w_k) \left\{ \log\left(1+\frac{tf_i(w_k)}{\mu p_c(w_k)}\right) - \log(|\vec{d}_i| + \mu) + \log \mu p_c(w_k) \right\}$$

$$\log p(q \mid \vec{d}_i) \cong \sum_{k=1} tf_q(w_k) \left\{ \log\left(1+\frac{tf_i(w_k)}{\mu p_c(w_k)}\right) - \log(|\vec{d}_i| + \mu) \right\}$$

**TF-IDF Weighting:**

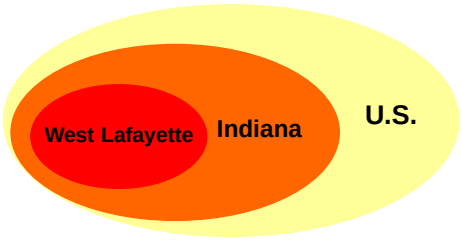
$$sim(q, \vec{d}_i) = \sum_{k=1}^{K} tf_q(w_k)\, tf_i(w_k)\, idf(w_k)\, norm(\vec{d}_i)$$

23

# Shrinkage

- Maximum Likelihood (MLE) builds model purely on document data and generates query word
  - Model may not be accurate when document is short (many unseen words)
- Shrinkage estimator builds more reliable model by consulting more general models (e.g., collection language model)

**Example: Estimate P(Lung_Cancer|Smoke)**






# Shrinkage

- Jelinek Mercer Smoothing
  - Assume for each word, with probability $\lambda$ , it is generated from document language model (MLE), with probability 1-$\lambda$ , it is generated from collection language model (MLE)
  - Linear interpolation between document language model and collection language model

**JM Smoothing:**

$$p_i(w_k) = \lambda \frac{tf_i(w_k)}{|\vec{d}_i|} + (1-\lambda)\, p_c(w_k)$$

# Query Expansion

- Users often start with short queries with ambiguous representations
- Observation: Many people refine their queries by analyzing the results from initial queries, or consulting other resources (thesaurus)
  - By adding and removing terms
  - By reweighting terms
  - By adding other features (e.g., Boolean operators)
- Technique of query expansion:

  Can a better query be created automatically?

# Query Expansion: Relevance Feedback Vector Space Model

**Goal:** Move new query close to relevant documents and far away from irrelevant documents

**Approach:** New query is a weighted average of original query, and relevant and non-relevant document vectors

$$\vec{q}' = \vec{q} + \alpha \frac{1}{|R|} \sum_{\vec{d}_i \in R} \vec{d}_i - \beta \frac{1}{|NR|} \sum_{\vec{d}_i \in NR} \vec{d}_i \quad \text{(Rocchio formula)}$$

**Positive feedback for terms in relevant docs**

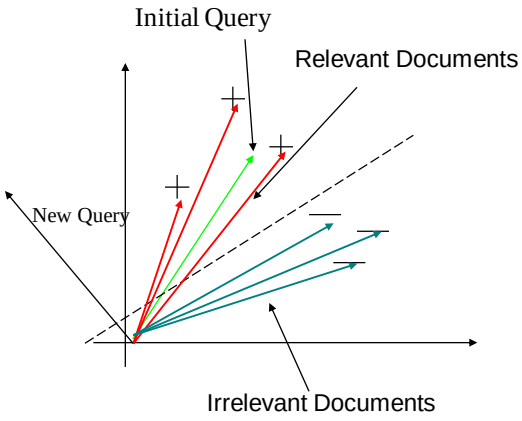
**Relevant documents**

**Irrelevant documents**

**Negative feedback for terms in irrelevant docs**

# Query Expansion: Relevance Feedback Vector Space Model

Desirable weights for $\alpha$ and $\beta$




Try find $\alpha$ and $\beta$ such that

$$\vec{q}(\alpha,\beta)\bullet\vec{d}_i \geq 1 \text{ for } \vec{d}_i \in R$$

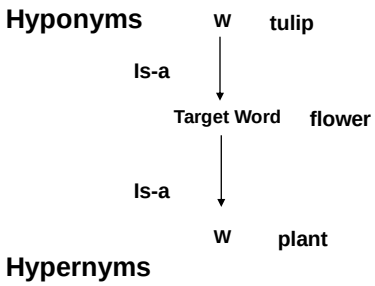
$$\vec{q}(\alpha,\beta)\bullet\vec{d}_i \leq -1 \text{ for } \vec{d}_i \in NR$$

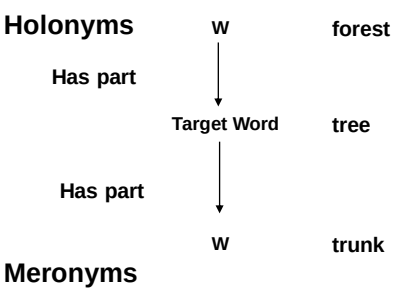
# Query Expansion: Relevance Feedback
## Blind(Pseudo) Relevance Feedback

Approaches

- Pseudo-relevance feedback
  - Assume top N (e.g., 20) documents in initial list are relevant
  - Assume bottom N' (e.g., 200-300) in initial list are irrelevant
  - Calculate weights of term according to some criterion (e.g., Rocchio)
  - Select top M (e.g., 10) terms
- Local context analysis
  - Similar approach to pseudo-relevance feedback
  - But use passages instead of documents for initial retrieval; use different term weight selection algorithms

## Query Expansion via External Resources
# Semantic Network




# Evaluation

Evaluation criteria

- Effectiveness
  - How to define effectiveness? Where can we find the correct answers?
- Efficiency
  - What about retrieval speed? What about the storage space? Particularly important for large-scale real-world system
- Usability
  - What is the most important factor for real user? Is user interface important?

# Evaluation

Evaluation criteria

- Effectiveness
  - Favor returned document ranked lists with more relevant documents at the top
  - Objective measures

Recall and Precision

Mean-average precision

Rank based precision

**For documents in a subset of a ranked lists, if we know the truth**

| | Retrieved | Not retrieved |
|---|---|---|
| Relevant | Relevant docs retrieved | Relevant docs not retrieved |
| Irrelevant | Irrelevant docs retrieved | Irrelevant docs not retrieved |

$$\text{Precision} = \frac{\text{Relevant docs retrieved}}{\text{Retrieved docs}}$$

$$\text{Recall} = \frac{\text{Relevant docs retrieved}}{\text{Relevant docs}}$$

# Evaluation

Pooling Strategy

- Retrieve documents using multiple methods
- Judge top n documents from each method
- Whole retrieved set is the union of top retrieved documents from all methods
- Problems: the judged relevant documents may not be complete
- It is possible to estimate size of true relevant documents by randomly sampling

# Evaluation

Evaluate a ranked list

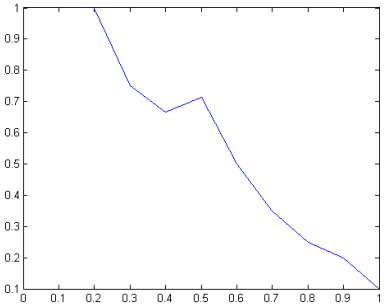
Precision at Recall

- Evaluate at every relevant document

| |
|---|
| + |
| + |
| - |
| + |
| + |
| - |
| + |

Not Retrieved: +++++

| Precision | Recall |
|---|---|
| 1 | 0.1 |
| 1 | 0.2 |
| 0.667 | 0.2 |
| 0.75 | 0.3 |
| 0.8 | 0.4 |
| 0.667 | 0.4 |
| 0.714 | 0.5 |




# Evaluation

Single value metrics

- Mean average precision
  - Calculate precision at each relevant document; average over all precision values
- 11-point interpolated average precision
  - Calculate precision at standard recall points (e.g., 10%, 20%...); smooth the values; estimate 0 % by interpolation
  - Average the results
- Rank based precision
  - Calculate precision at top ranked documents (e.g., 5, 10, 15…)
  - Desirable when users care more for top ranked documents

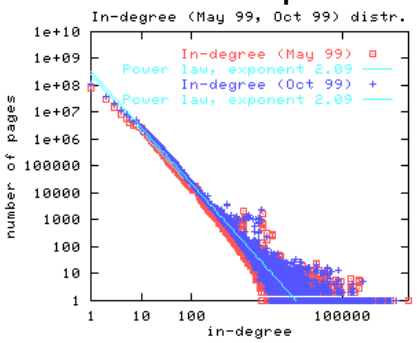
# Ad-Hoc Retrieval: Beyond the Words

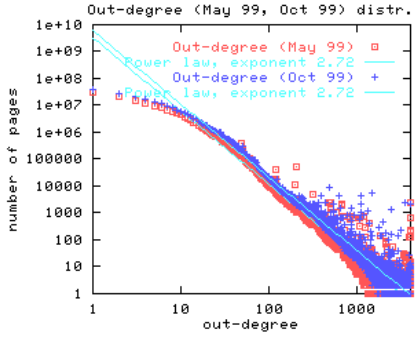
- Web is a graph
  - Each web site correspond to a node
  - A link from one site to another site forms a directed edge
- What does it look like?
  - Web is small world
  - The diameter of the web is 19
    - e.g. the average number of clicks from one web site to another is 19

36

# Inlinks and Outlinks

- Both degrees of incoming and outgoing links follow power law




Broder et al., 2001

37

# Early Approaches

Basic Assumptions

- Hyperlinks contain information about the human judgment of a site
- The more incoming links to a site, the more it is judged important

Bray 1996

- The visibility of a site is measured by the number of other sites pointing to it
- The luminosity of a site is measured by the number of other sites to which it points
- ➢ Limitation: failure to capture the relative importance of different parents (children) sites

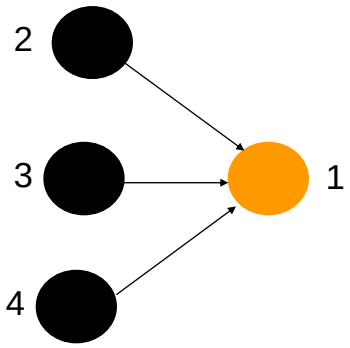
38

# HITS - Kleinberg's Algorithm

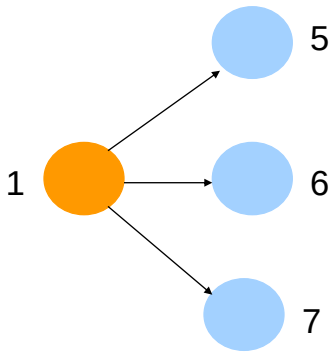
- HITS – Hypertext Induced Topic Selection
- For each vertex v ϵ V in a subgraph of interest:
  - a(v) - the authority of v
  - h(v) - the hubness of v
- A site is very authoritative if it receives many citations.
  - Citation from important sites weight more than citations from less-important sites
- Hubness shows the importance of a site.
  - A good hub is a site that links to many authoritative sites

39

# Authority and Hubness




a(1) = h(2) + h(3) + h(4)




h(1) = a(5) + a(6) + a(7)

40

# Authority and Hubness

- Authority score
  - Not only depends on the number of incoming links
  - But also the 'quality' (e.g., hubness) of the incoming links
- Hubness score
  - Not only depends on the number of outgoing links
  - But also the 'quality' (e.g., hubness) of the outgoing links

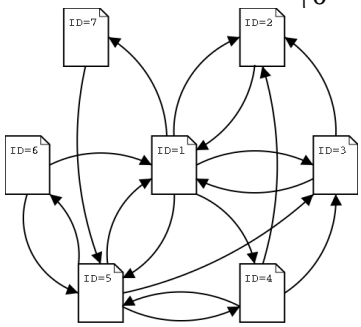
41

# Authority and Hub

- Column vector ***a***: $a_i$ is the authority score for the i-th site
- Column vector ***h***: $h_i$ is the hub score for the i-th site

- Matrix **M**:

$$\mathbf{M}_{i,j} = \begin{cases} 1 & \text{the } i\text{th site points to the } j\text{ th site} \\ 0 & \text{otherwise} \end{cases}$$




$$\mathbf{M} = \begin{pmatrix} 0&1&1&1&1&0&1 \\ 1&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0 \\ 0&1&1&0&1&0&0 \\ 1&0&1&1&0&1&0 \\ 1&0&0&0&1&0&0 \\ 0&0&0&0&1&0&0 \end{pmatrix}$$

42

# Authority and Hub

- Column vector ***a***: $a_i$ is the authority score for the i-th site
- Column vector ***h***: $h_i$ is the hub score for the i-th site

- Matrix **M**:

$$\mathbf{M}_{i,j} = \begin{cases} 1 & \text{the } i\text{th site points to the } j\text{ th site} \\ 0 & \text{otherwise} \end{cases}$$

- Recursive dependency:

a(v) ← $\Sigma_{w \in pa[v]}$ h(w)

h(v) ← $\Sigma_{w \in ch[v]}$ a(w)

$$\mathbf{a}_t = \alpha_t \mathbf{M}^T \mathbf{h}_t$$

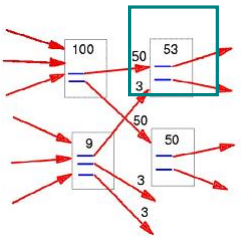
$$\mathbf{h}_t = \beta_t \mathbf{M} \mathbf{a}_t$$

Normalization Procedure

43

# PageRank

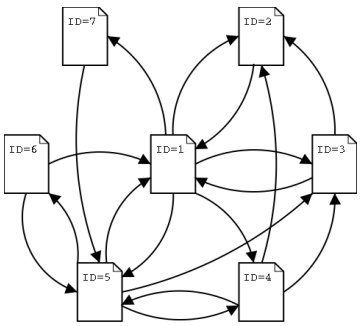
- Introduced by Page et al (1998)
  - The weight is assigned by the rank of parents
- Difference with HITS
  - HITS takes Hubness & Authority weights
  - The page rank is proportional to its parents' rank, but inversely proportional to its parents' outdegree




# Matrix Notation

$$\mathbf{M}_{i,j} = \begin{cases} 1 & \text{the } i\text{th site points to the } j\text{th site} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{B}_{i,j} = \begin{cases} \dfrac{1}{\sum_j \mathbf{M}_{i,j}} & \sum_j \mathbf{M}_{i,j} > 0 \\ 0 & \text{otherwise} \end{cases}$$




$$\mathbf{M} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} 0 & 1/5 & 1/5 & 1/5 & 1/5 & 0 & 1/5 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

# Matrix Notation

$\mathbf{r} : \mathbf{r}_i$ represents the rank score for the i-thweb page

$$r(v) = \alpha \sum_{w \in \mathrm{pa}[v]} \frac{r(w)}{|\mathrm{ch}[w]|'}$$

$\mathbf{r} = \alpha\, \mathbf{B}^T \mathbf{r}$

α : eigenvalue

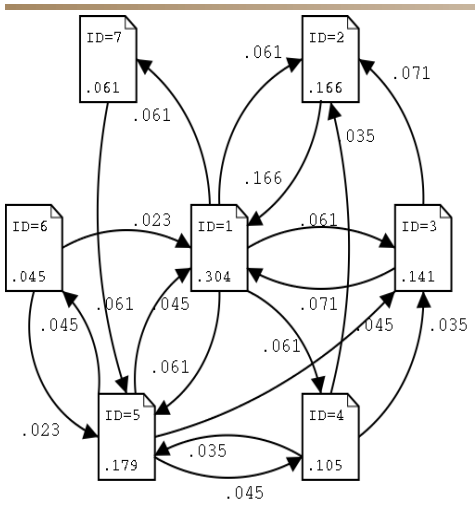
r : eigenvector of **B**

Finding Pagerank

→ finding principle eigenvector of B



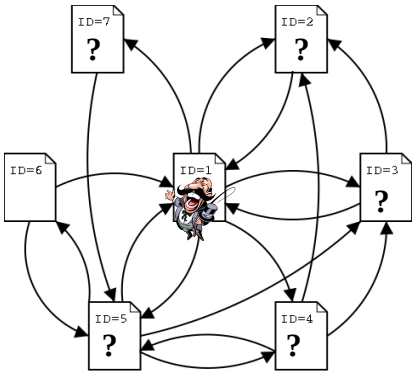
# Matrix Notation




| PR | ID | OutLink | InLink |
|---|---|---|---|
| 0.304 | 1 | 2,3,4,5,7 | 2,3,5,6 |
| 0.179 | 5 | 1,3,4,6 | 1,4,6,7 |
| 0.166 | 2 | 1 | 1,3,4 |
| 0.141 | 3 | 1,2 | 1,4,5 |
| 0.105 | 4 | 2,3,5 | 1,5 |
| 0.061 | 7 | 5 | 1 |
| 0.045 | 6 | 1,5 | 5 |

# Random Walk Model

- Consider a random walk through the Web graph



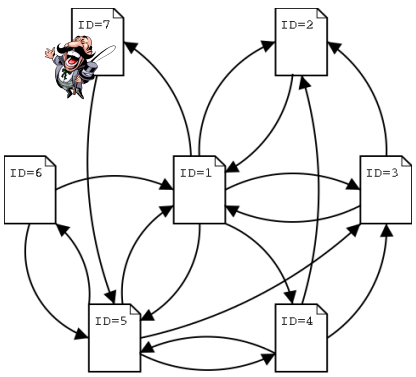

**B** =

$$\begin{pmatrix} 0 & 1/5 & 1/5 & 1/5 & 1/5 & 0 & 1/5 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

48

# Random Walk Model

- Consider a random walk through the Web graph



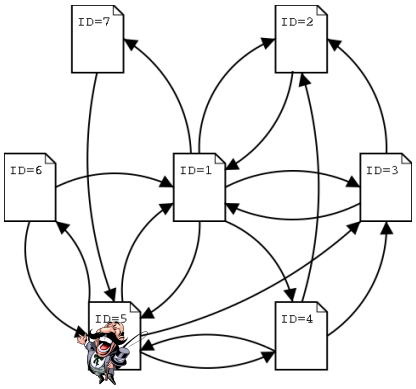

**B** =

$$\begin{pmatrix} 0 & 1/5 & 1/5 & 1/5 & 1/5 & 0 & 1/5 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

49

# Random Walk Model

- Consider a random walk through the Web graph

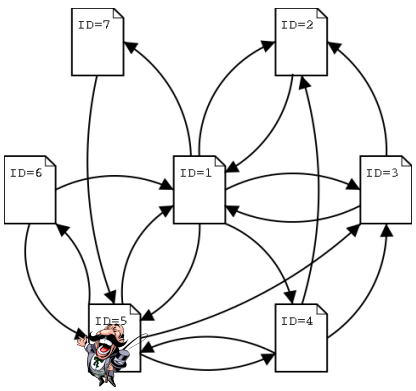



$$B = \begin{pmatrix} 0 & 1/5 & 1/5 & 1/5 & 1/5 & 0 & 1/5 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$



# Random Walk Model

- Consider a random walk through the Web graph

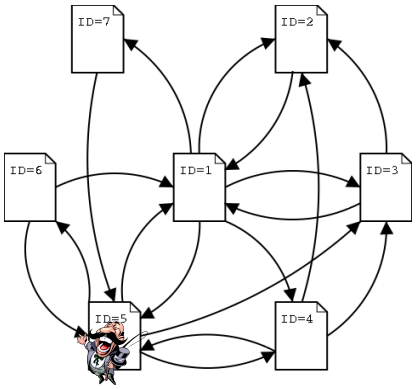



$$B = \begin{pmatrix} 0 & 1/5 & 1/5 & 1/5 & 1/5 & 0 & 1/5 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$T \rightarrow \infty$, what is portion of time that the surfer will spend time on each site?

# Random Walk Model

- Consider a random walk through the Web graph




$$\mathbf{B} = \begin{pmatrix} 0 & 1/5 & 1/5 & 1/5 & 1/5 & 0 & 1/5 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$p(k)$: percentage of time that the surfer will stay at the i-th site

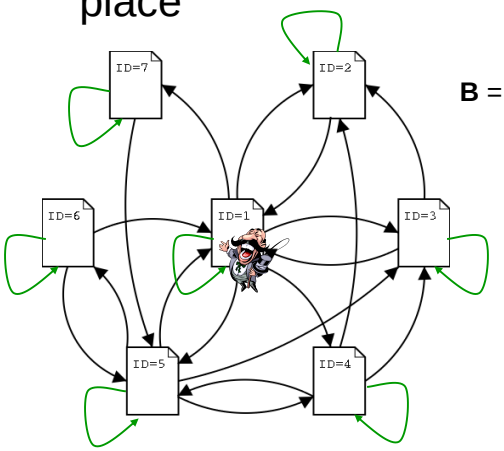
$$p(k) = \sum_i p(i)\mathbf{B}_{i,k}$$

$$\mathbf{p} = \mathbf{B}^T\mathbf{p}$$

52

# Adding Self Loop

- Allow surfer to decide to stay on the same place




$$\mathbf{B} = \begin{pmatrix} 0 & 1/5 & 1/5 & 1/5 & 1/5 & 0 & 1/5 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$\mathbf{B}' = \alpha\mathbf{B} + (1-\alpha)\mathbf{I}$$

53

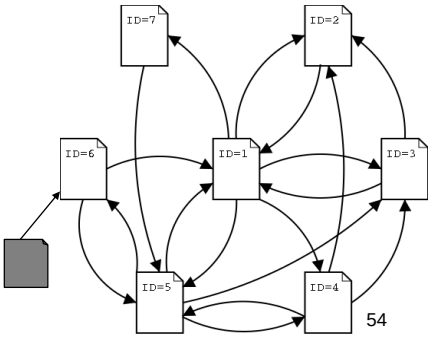
# Problem

- "Rank Sink" Problem
  - In general, many Web pages have no inlinks/outlinks
  - Results in dangling edges in the graph

**B** =

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1/5 | 1/5 | 1/5 | 1/5 | 0 | 1/5 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1/2 | 1/2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1/3 | 1/3 | 0 | 1/3 | 0 | 0 | 0 |
| 1/4 | 0 | 1/4 | 1/4 | 0 | 1/4 | 0 | 0 |
| 1/2 | 0 | 0 | 0 | 1/2 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

r(new page) = 0




54

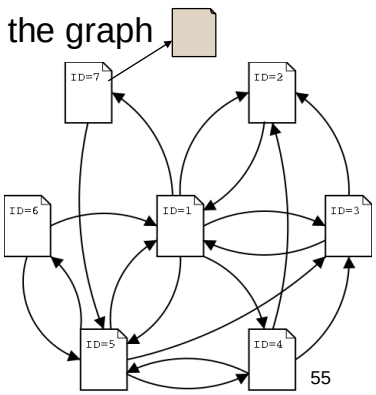
# Problem

- "Rank Sink" Problem
  - In general, many Web pages have no inlinks/outlinks
  - Results in dangling edges in the graph

**B** =

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1/5 | 1/5 | 1/5 | 1/5 | 0 | 1/5 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1/2 | 1/2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1/3 | 1/3 | 0 | 1/3 | 0 | 0 | 0 |
| 1/4 | 0 | 1/4 | 1/4 | 0 | 1/4 | 0 | 0 |
| 1/2 | 0 | 0 | 0 | 1/2 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

r(new page) = 1




55

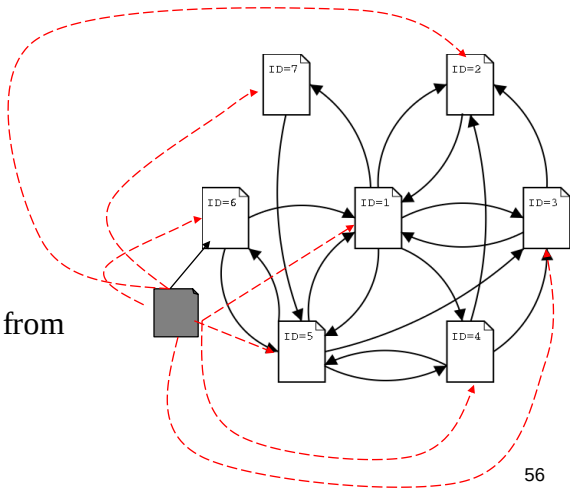
# Distribution of the Mixture Model

$$\mathbf{H}_{i,j} = 1/n$$

$$\mathbf{B}' = \varepsilon\mathbf{H} + (1-\varepsilon)\mathbf{B}$$

$$\mathbf{r} = \mathbf{B}'^{T}\,\mathbf{r}$$

Prevents the page ranks from being 0 or 1




56

# Stability

- Are link analysis algorithms based on eigenvectors stable?
  - Will small changes in graph result in major changes in outcomes?
- What if the connectivity of a portion of the graph is changed arbitrarily?
  - How will this affect the results of algorithms?

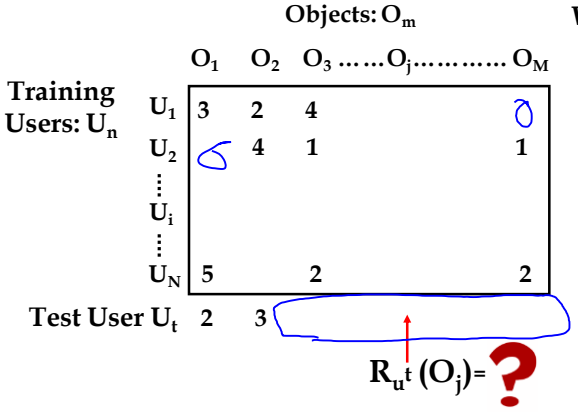
57

# Collaborative Filtering

Outline

- Introduction to collaborative filtering
- Main framework
- Memory-based collaborative filtering approach
- Model-based collaborative filtering approach
  - Aspect model & Two-way clustering model
  - Flexible mixture model
  - Decouple model
- Unified filtering by combining content and collaborative filtering

# Formal Framework for Collaborative Filtering




**What we have:**

- **Assume there are some ratings by training users**
- **Test user provides some amount of additional training data**

**What we do:**

- **Predict test user's rating based training information**

# Memory-Based Approaches

- Memory-Based Approaches
  - Given a specific user *u*, find a set of similar users
  - Predict *u*'s rating based on ratings of similar users
- Issues
  - How to determine the similarity between users?
  - How to combine the ratings from similar users to make the predictions (how to weight different users)?

# Memory-Based Approaches

- How to determine the similarity between users?
  - Measure the similarity in rating patterns between different users

**Pearson Correlation Coefficient Similarity**

$$w_{u,u^t} = \frac{\sum (R_{u^t}(o) - \bar{R}_{u^t})(R_u(o) - \bar{R}_u)}{\sqrt{\sum (R_{u^t}(o) - \bar{R}_{u^t})^2} \sqrt{\sum (R_u(o) - \bar{R}_u)^2}}$$

**Vector Space Similarity**

$$w_{u,u^t} = \frac{\sum R_{u^t}(o) R_u(o)}{\sqrt{\sum R_{u^t}(o)^2} \sqrt{\sum R_u(o)^2}}$$

**Average Ratings**

**Prediction:**

$$\hat{R}_{u^t}(o) = \bar{R}_{u^t} + \frac{\sum_u w_{u,u^t} (R_u(o) - \bar{R}_u)}{\sum_u |w_{u,u^t}|}$$

# Memory-Based Approaches

- How to combine the ratings from similar users for predicting?
  - Weight similar users by their similarity with a specific user; use these weights to combine their ratings.

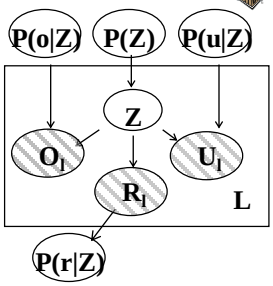
Prediction: $\hat{R}_{u'}(o) = \bar{R}_{u'} + \frac{\sum_u w_{u,u'}(R_u(o) - \bar{R}_u)}{\sum_u |w_{u,u'}|}$

# Memory-Based Approaches

- Problems with memory-based approaches
  - Associated a large amount of computation online costs
    (have to go over all users, any fast indexing approach?)
  - Heuristic method to calculate user similarity and make user rating prediction
- Possible Solution
  - Cluster users/items in offline manner, save for online computation cost
  - Proposal more solid probabilistic modeling method

# Collaborative Filtering

- Model-Based Approaches:
- **Aspect Model (Hofmann et al., 1999)**
  - Model individual ratings as convex combination of preference factors




$$P(o_{(l)}, u_{(l)}, r_{(l)}) = \sum_{z \in Z} P(z) P(o_{(l)} \mid z) P(u_{(l)} \mid z) P(r_{(l)} \mid z)$$

**Two-Sided Clustering Model (Hofmann et al., 1999)**

- Assume each user and item belong to one user and item group.

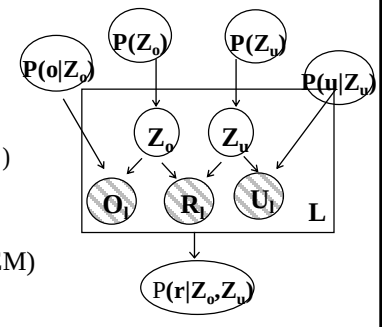
$$P(o_{(l)}, u_{(l)}, r_{(l)}) = P(o_{(l)}) P(u_{(l)}) \sum_{v,u} I_{x_{(l)}v} J_{y_{(l)}u} C_{vu}$$

**$I_{x(l)v}$, $J_{y(l)u}$: Indicator Variables $C_{vu}$: Associaion Parameter**

# Collaborative Filtering

- Flexible Mixture Model (FMM):

Cluster users and objects separately AND allow them to belong to different classes




$$P(o_{(l)}, u_{(l)}, r_{(l)}) = \sum_{Z_o, Z_u} P(Z_o) P(Z_u) P(o_{(l)} \mid Z_o) P(u_{(l)} \mid Z_u) P(r_{(l)} \mid Z_o, Z_u)$$

- Training Procedure:
  Annealed Expectation Maximization (AEM) algorithm

E-Step: Calculate Posterior Probabilities

$$P(z_o, z_u \mid o_{(l)}, u_{(l)}, r_{(l)}) = \frac{(P(Z_o) P(Z_u) P(o_{(l)} \mid Z_o) P(u_{(l)} \mid Z_u) P(r_{(l)} \mid Z_o, Z_u))^b}{\sum_{Z_o, Z_u} (P(Z_o) P(Z_u) P(o_{(l)} \mid Z_o) P(u_{(l)} \mid Z_u) P(r_{(l)} \mid Z_o, Z_u))^b}$$

# Collaborative Filtering

$$P(Z_o);\ P(Z_u);\ P(o_{(l)} \mid Z_o);\ P(u_{(l)} \mid Z_u);\ P(r_{(l)} \mid Z_o, Z_u)$$

M-Step: Update Parameters

- Prediction Procedure:
  Fold-In process to calculate join probabilities

$$P(o, u^t, r_{(l)}) = \sum_{Z_o, Z_u} P(Z_o)P(Z_u)P(o \mid Z_o)P(u^t \mid Z_u)P(r \mid Z_o, Z_u)$$

**Fold-in process by EM algorithm**

Calculate expectation for prediction

$$\hat{R}_{u^t}(o) = \sum_r r \frac{P(o, u^t, r)}{\sum_{r'} P(o, u^t, r')}$$

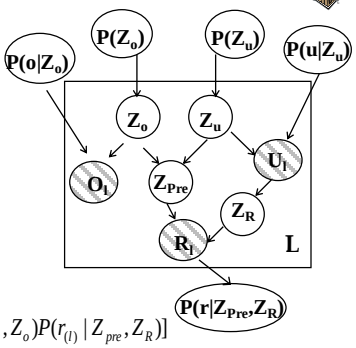
**"Flexible Mixture Model for Collaborative Filtering", ICML'03**

# Decoupled Model (DM)

- Decoupled Model (DM):
  Separate preference value

$Z_{pref} \in [1, \ldots, k]$ (1 disfavor, k favor)

from rating $r \in \{1,2,3,4,5\}$




**Joint Probability:**

$$P(o_{(l)}, u_{(l)}, r_{(l)})$$
$$= \sum_{Z_o, Z_u, Z_R} P(Z_o)P(Z_u)P(o_{(l)} \mid Z_o)P(u_{(l)} \mid Z_u)P(Z_R \mid u_{(l)})[\sum_{Z_{pre}} P(Z_{pre} \mid Z_u, Z_o)P(r_{(l)} \mid Z_{pre}, Z_R)]$$

**"Preference-Based Graphical Model for Collaborative Filtering", UAI'03**

**"A study of Mixture Model for Collaborative Filtering", Journal of IR**

# Techniques Explored in Text Categorization

- Rule-based Expert system (Hayes, 1990)
- Nearest Neighbor methods (Creecy'92; Yang'94)
- Decision symbolic rule induction (Apte'94)
- Naïve Bayes (Language Model) (Lewis'94; McCallum'98)
- Regression method (Furh'92; Yang'92)
- Support Vector Machines (Joachims'98)
- Boosting or Bagging (Schapier'98)
- Neural networks (Wiener'95)
- ……

# Text Categorization: Evaluation

Contingency Table Per Category (for all docs)

| | Truth: True | Truth: False | |
|---|---|---|---|
| Predicted Positive | a | b | a+b |
| Predicted Negative | c | d | c+d |
| | a+c | b+d | n=a+b+c+d |

**a: number of truly positive docs b: number of false-positive docs**

**c: number of false negative docs d: number of truly-negative docs**

**n: total number of test documents**

# Text Categorization: Evaluation

**Recall: r=a/(a+c)** **percentage of positive docs detected**

**Precision: p=a/(a+b)** **how accurate are the predicted positive docs**

**Accuracy: (a+d)/n** **how accurate are all the predicted docs**

**F-measure:** $F_\beta = \frac{(\beta^2+1)pr}{\beta^2 p + r}$ $F_1 = \frac{2pr}{p+r}$

**Harmonic average:** $\frac{1}{\frac{1}{2}\left(\frac{1}{x_1}+\frac{1}{x_2}\right)}$

**Error: (b+c)/n** **error rate of predicted docs**

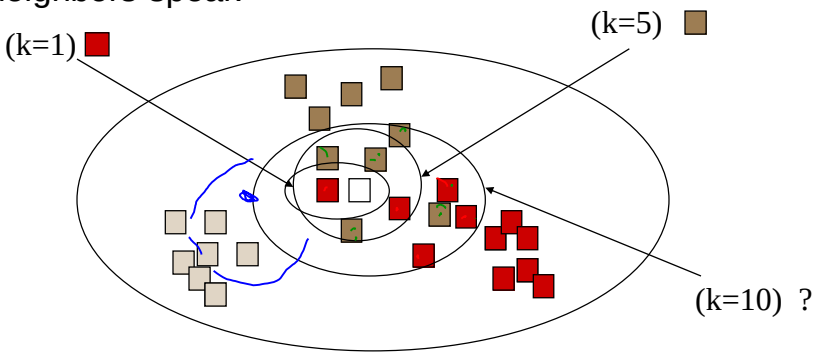
**Accuracy+Error=1**

# Text Categorization: Evaluation

- Micro F1-Measure
  - Calculate a single contingency table for all categories and calculate F1 measure
  - Treat each prediction with equal weight; better for algorithms that work well on large categories
- Macro F1-Measure
  - Calculate a single contingency table for every category calculate F1 measure separately and average the values
  - Treat each category with equal weight; better for algorithms that work well on many small categories

# K-Nearest Neighbor Classifier

- Idea: find your language by what language your neighbors speak




- Use K nearest neighbors to vote

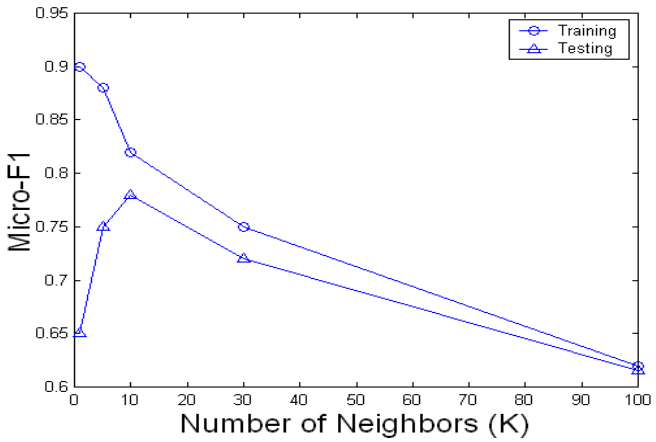
1-NN:Red; 5-NN:Brown; 10-NN:?; Weighted 10-NN:Brown

# Choices of Similarity Functions

| | |
|---|---|
| Euclidean distance | $d(\vec{x}_1, \vec{x}_2) = \sqrt{\sum_v (x_{1v} - x_{2v})^2}$ |
| Kullback Leibler distance | $d(\vec{x}_1, \vec{x}_2) = \sum_v x_{1v} \log \frac{x_{1v}}{x_{2v}}$ |
| Dot product | $\vec{x}_1 * \vec{x}_2 = \sum_v x_{1v} * x_{2v}$ |
| Cosine Similarity | $\cos(\vec{x}_1, \vec{x}_2) = \frac{\sum_v x_{1v} * x_{2v}}{\sqrt{\sum_v x_{1v}^2} \sqrt{\sum_v x_{2v}^2}}$ |
| Kernel functions | $k(\vec{x}_1, \vec{x}_2) = e^{-d(\vec{x}_1, \vec{x}_2)/2\sigma^2}$ (Gaussian Kernel) |

Automatic learning of the metrics

# Choices of Number of Neighbors (K)




**Trade off between small number of neighbors and large number of neighbors**

# Choices of Number of Neighbors (K)

- Find desired number of neighbors by cross validation
  - Choose a subset of available data as training data, the rest as validation data
  - Find the desired number of neighbors on the validation data
  - The procedure can be repeated for different splits; find the consistent good number for the splits

# Naïve Bayes Classification

- Naïve Bayes (NB) Classification
  - Generative Model: Model both the input data (i.e., document contents) and output data (i.e., class labels)
  - Make strong assumption of the probabilistic modeling approach
- Methodology
  - Similar with the idea of language modeling approaches for information retrieval
  - Train a language model for all the documents in one category

# Naïve Bayes Classification

- Methodology
  - Train a language model for all the documents in one category

$$\text{Category 1}: \left(\vec{d}_{1,1}, \vec{d}_{1,2}, ..., \vec{d}_{1,n_1}\right) \rightarrow \text{Language model } \theta_1$$

$$\text{Category 2}: \left(\vec{d}_{2,1}, \vec{d}_{2,2}, ..., \vec{d}_{2,n_2}\right) \rightarrow \text{Language model } \theta_2$$

.....

$$\text{Category C}: \left(\vec{d}_{C,1}, \vec{d}_{C,2}, ..., \vec{d}_{C,n_K}\right) \rightarrow \text{Language model } \theta_C$$

  - What is the language model? (Multinomial distribution)
  - How to estimate the language model for all the documents in one category?

# Naïve Bayes Classification

Maximum Likelihood Estimation:

- Find model parameters for a category that maximizes generation likelihood:

$$\theta_c^* = \arg\max_{\theta_c} p(\vec{d}_{c1},..,\vec{d}_{cn_c} \mid \theta_c)$$

There are K words in vocabulary, $w_1...w_K$

Data: documents $\vec{d}_{c1},...,\vec{d}_{cn_c}$

For $\vec{d}_{ci}$ with counts $c_i(w_1)$, …, $c_i(w_k)$, and length $|\vec{d}_{ci}|$

Model: multinomial M with parameters $\{p(w_k)\}$

Likelihood: $\Pr(\vec{d}_{c1},...,\vec{d}_{cn_c} \mid \theta)$

$$\theta_c^* = \arg\max_{\theta_c} p(\vec{d}_{c1},..,\vec{d}_{cn_c} \mid \theta_c)$$

81

# Naïve Bayes Classification

- **MLE Estimator: Normalization by simple counting**
  - Train a language model for all the documents in one category

$$p(w \mid \theta_c^*) = \frac{\sum_{i=1}^{n_c} c_{ci}(w)}{\sum_{i=1}^{n_c} |\vec{d}_{ci}|}$$

- **Category Prior:**
  - Number of documents in the category divided by the total number of documents

$$p(c) = \frac{n_c}{\sum_{c'} n_{c'}}$$

# Naïve Bayes Classification

- **Smoothed Estimator:**
  - Laplace Smoothing

$$p(w \mid \theta_c^*) = \frac{1 + \sum_{i=1}^{n_c} c_{ci}(w)}{K + \sum_{i=1}^{n_c} \left| \vec{d}_{ci} \right|}$$

Number of Words in Vocabulary

  - Hierarchical Smoothing

$$p(w \mid \theta_c^*) = \lambda_1 P(w \mid \theta_c^*) + \lambda_2 P(w \mid \theta_{c^{up1}}^*) .... + \lambda_m P(w \mid \theta_{c^{root}}^*)$$

  - Dirichlet Smoothing

# Naïve Bayes Classification

- **Example of Binary Classification**

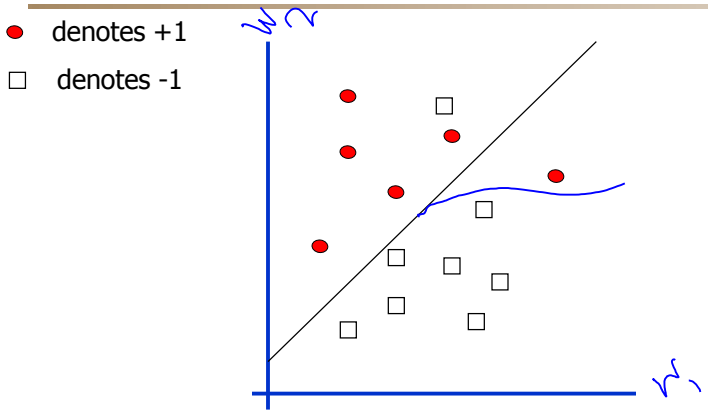
$$c^* = \arg\max_{l \in [-,+]} p(c_l \mid \vec{d}_i) \rightarrow \frac{p(c_+ \mid \vec{d}_i)}{p(c_- \mid \vec{d}_i)}$$

$$\log \frac{p(c_+ \mid \vec{d}_i)}{p(c_- \mid \vec{d}_i)} = \log \left\{ \frac{\prod_k \left[ p(w_k \mid c_+) \right]^{c_i(w_k)} \frac{n_+}{n_+ + n_-}}{\prod_k \left[ p(w_k \mid c_-) \right]^{c_i(w_k)} \frac{n_-}{n_+ + n_-}} \right\}$$

$$= \log \left( \frac{n_+}{n_-} \right) + \sum_k c_i(w_k) \log \left( \frac{p(w_k \mid c_+)}{p(w_k \mid c_-)} \right)$$

$$\log \frac{p(c_+ \mid \vec{d})}{p(c_- \mid \vec{d})} \propto b_0 + \sum_k c_i(w_k) \times weight(w_k)$$

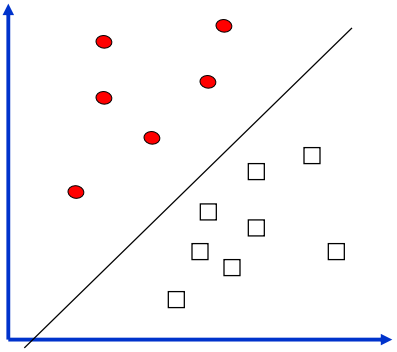
# Naïve Bayes = Linear Classifier




$$\log \frac{p(c_+ \mid \vec{d}_i)}{p(c_- \mid \vec{d}_i)} \propto b_0 + \sum_k c_i(w_k) \times weight(w_k)$$

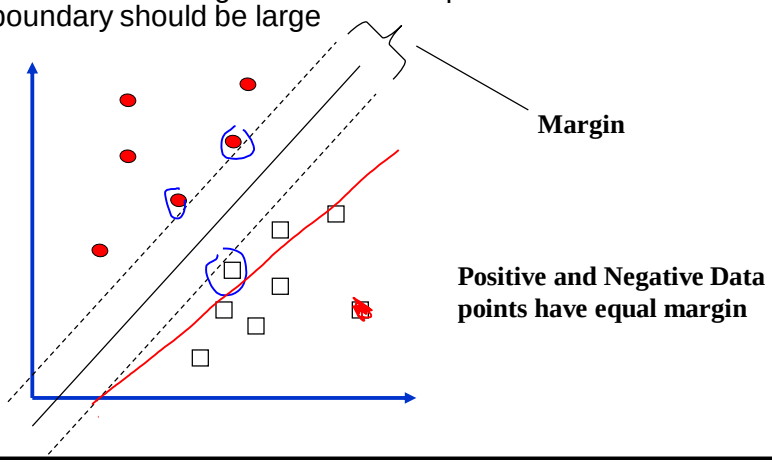
# Support Vector Machine

- Consider a two-class (binary classification problem like text categorization)
  - Find a line to separate data points in two classes
- There are many possible solutions!
  - Are those decision boundaries equally good?



# Large-Margin Decision Criterion

- The decision boundary should be far away from the data points of two classes as much as possible
- Indicates the margin between data points and the decision boundary should be large




# Linear SVM

•Let $\{x_1, ..., x_n\}$ denote input data. For example, vector representation of all documents

•Let $y_i$ be the binary indicator 1 or -1 that indicates whether $x_i$ belongs to a particular category c or not

The decision boundary should classify all points correctly

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \qquad \forall i$$

The decision boundary can be found by solving the following constrained optimization problem

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

# Linear SVM

•Set the derivative of the Lagrangian to be zero and calculate W by $a_i$, plug new form of w into the Lagrangian, the optimization problem can be written in terms of $a_i$ (the dual problem)

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

Plug new form of w into the Lagrangian, the optimization problem can be written in terms of $a_i$ (the dual problem)

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

The above optimization problem is a quadratic program problem, which means there is a global maximum of $a_i$ can always be found

# Soft Margin Linear SVM

Introduction "slack variables", slack variables are always positive

$$\begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T \mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

Introduce const C to balance error for linear boundary and the margin

$$\tfrac{1}{2}||\mathbf{w}||^2 + C \textstyle\sum_i \xi_i$$

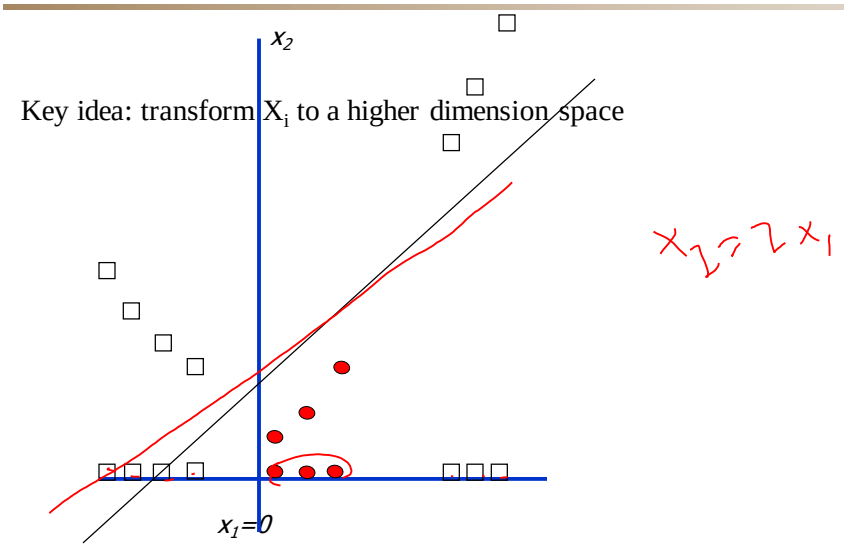
The optimization problem becomes

$$\text{Minimize } \tfrac{1}{2}||\mathbf{w}||^2 + C \textstyle\sum_{i=1}^{n} \xi_i$$
$$\text{subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

# Non-linear SVM

- Linear SVM only uses a line to separate data points, how to generalize it to non-linear case?
- Key idea: transform $X_i$ to a higher dimension space
  - Input space: the space the point $x_i$ are located
  - Feature space: the space of $f(x_i)$ after transformation

# Non-linear SVM



# Non-linear SVM

Key idea: transform $X_i$ to a higher dimension space

- Input space: the space the point $\mathbf{x}_i$ are located
- Feature space: the space after transformation

Use $\Phi(\mathbf{x}_i)$ to transform low level feature to high level feature

Sometimes, the $\Phi(\mathbf{x}_i)$ transformation maps to very high dimensional space or even infinite dimensional space

How can we calculate the high dimensional representation for all data points?

# The Kernel Trick

- Recall the SVM optimization problem

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Only need inner product

The data points only appear as inner product

As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly

Many common geometric operations (angles, distances) can be expressed by inner products

Define the kernel function $K$ by $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

## Examples for the Kernel trick

- Suppose f(.) is given as follows

$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

- An inner product in the feature space is

$$\langle \phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}), \phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) \rangle = (1 + x_1y_1 + x_2y_2)^2$$

- So, if we define the kernel function as follows, there is no need to carry out f(.) explicitly

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

## More Kernel Functions

- Polynomial kernel with degree d

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$$

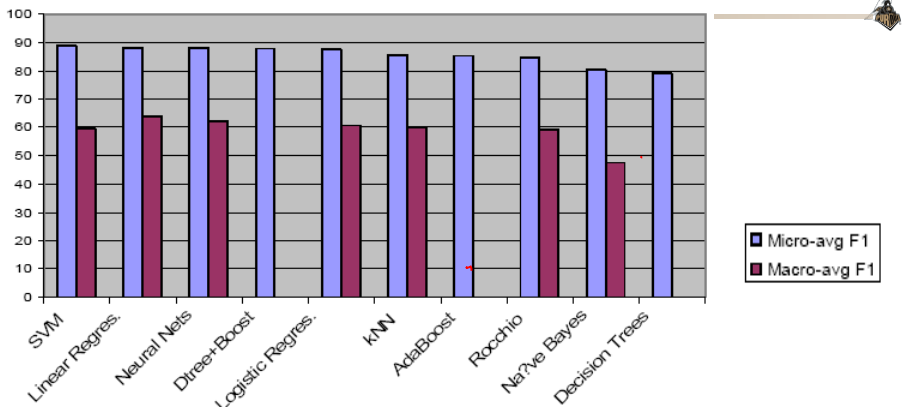
- Gaussian Radial basis function kernel with width σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2))$$

- Two-layer sigmoid neural network

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa\mathbf{x}^T\mathbf{y} + \theta)$$

# Text Categorization: Evaluation




**Performance of different algorithms on Reuters-21578 corpus: 90 categories, 7769 Training docs, 3019 test docs, (Yang, JIR 1999)**

# What is clustering?

- Clustering is the process of grouping a set of physical or abstract objects into classes of similar objects
  - It is the commonest form of unsupervised learning
    - Unsupervised learning = learning from raw data, as opposed to supervised data where the correct classification of examples is given
  - It is a common and important task that finds many applications in IR and other places

98

# Issues for clustering

- Representation for clustering
  - Document representation
    - Vector space? Normalization?
  - Need a notion of similarity/distance
- How many clusters?
  - Fixed a priori?
  - Completely data driven?
    - Avoid "trivial" clusters - too large or small
      - In an application, if a cluster's too large, then for navigation purposes you've wasted an extra user click without whittling down the set of documents much.

99

# Clustering Algorithms

- Partitioning "flat" algorithms
  - Usually start with a random (partial) partitioning
  - Refine it iteratively
    - *k* means/medoids clustering
    - Model based clustering
- Hierarchical algorithms
  - Bottom-up, agglomerative
  - Top-down, divisive

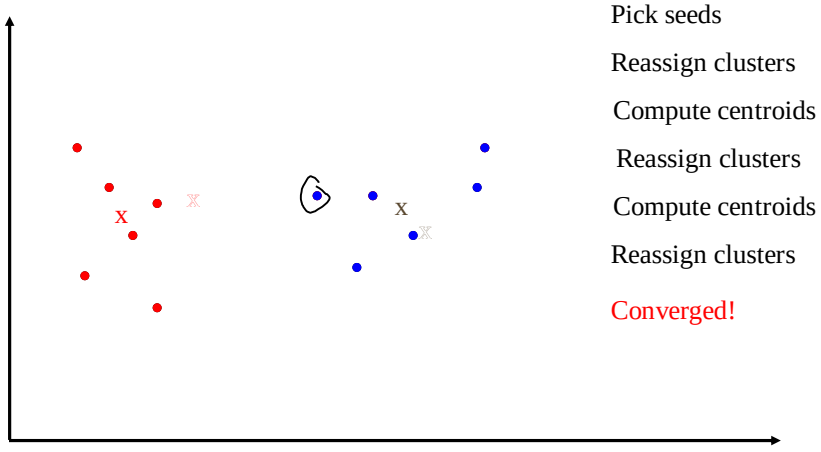
100

# K-Means Algorithm

Let $d$ be the distance measure between instances.
Select $k$ random instances $\{s_1, s_2, \ldots s_k\}$ as seeds.
Until clustering converges or other stopping criterion:
    For each instance $x_i$:
        Assign $x_i$ to the cluster $c_j$ such that $d(x_i, s_j)$ is minimal.
    (*Update the seeds to the centroid of each cluster*)
    For each cluster $c_j$
        $s_j = \mu(c_j)$

101

# K Means Example
## (K=2)




102

# How Many Clusters?

- Number of clusters *k* is given
  - Partition *n* docs into predetermined number of clusters
- Finding the "right" number of clusters is part of the problem
  - Given docs, partition into an "appropriate" number of subsets.
  - E.g., for query results - ideal value of *k* not known up front - though UI may impose limits.
- Can usually take an algorithm for one flavor and convert to the other.

# Penalize lots of clusters

- For each cluster, we have a Cost *C*.
- Thus for a clustering with *k* clusters, the Total Cost is *kC*.
- Define the Value of a clustering to be =

  Total Benefit - Total Cost.
- Find the clustering of highest value, over all choices of *k*.
  - Total benefit increases with increasing K. But can stop when it doesn't increase by "much". The Cost term enforces this.

# Soft Clustering

- Clustering typically assumes that each instance is given a “hard” assignment to exactly one cluster.
- Does not allow uncertainty in class membership or for an instance to belong to more than one cluster.
- *Soft clustering* gives probabilities that an instance belongs to each of a set of clusters.
- Each instance is assigned a probability distribution across a set of discovered categories (probabilities of all categories must sum to 1).

# Model based clustering

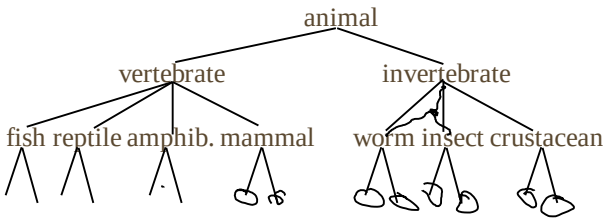
- Algorithm optimizes a probabilistic model criterion
- Clustering is usually done by the Expectation Maximization (EM) algorithm
  - Gives a soft variant of the K-means algorithm
  - Assume $k$ clusters: $\{c_1, c_2, \ldots c_k\}$
  - Assume a probabilistic model of categories that allows computing $P(c_i \mid E)$ for each category, $c_i$, for a given example, $E$.
  - For text, typically assume a naïve Bayes category model.
  - Parameters $\theta = \{P(c_i), P(w_j \mid c_i): i \in \{1,\ldots k\}, j \in \{1,\ldots,|V|\}\}$

## Expectation Maximization (EM) Algorithm

- Iterative method for learning probabilistic categorization model from unsupervised data.
- Initially assume random assignment of examples to categories.
- Learn an initial probabilistic model by estimating model parameters $\theta$ from this randomly labeled data.
- Iterate following two steps until convergence:
  - Expectation (E-step): Compute $P(c_i \mid E)$ for each example given the current model, and probabilistically re-label the examples based on these posterior probability estimates.
  - Maximization (M-step): Re-estimate the model parameters, $\theta$, from the probabilistically re-labeled data.

## Hierarchical Clustering

- Build a tree-based hierarchical taxonomy (*dendrogram*) from a set of unlabeled examples.




- One option to produce a hierarchical clustering is recursive application of a partitional clustering algorithm to produce a hierarchical clustering.

# HAC Algorithm

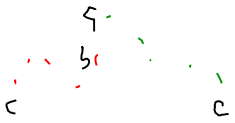
Start with all instances in their own cluster.
Until there is only one cluster:
Among the current clusters, determine the two clusters, $c_i$ and $c_j$, that are most similar.
Replace $c_i$ and $c_j$ with a single cluster $c_i \cup c_j$

# Hierarchical Clustering algorithms

- **Agglomerative (bottom-up):**
  - Start with each document being a single cluster.
  - Eventually all documents belong to the same cluster.
- **Divisive (top-down):**
  - Start with all documents belong to the same cluster.
  - Eventually each node forms a cluster on its own.
- Does not require the number of clusters ***k*** in advance
- Needs a termination/readout condition
  - The final mode in both Agglomerative and Divisive is of no use.

# "Closest pair" of clusters

- Many variants to defining closest pair of clusters
- "Center of gravity"
  - Clusters whose centroids (centers of gravity) are the most cosine-similar
- Average-link
  - Average cosine between pairs of elements
- Single-link
  - Similarity of the most cosine-similar (single-link)
- Complete-link
  - Similarity of the "furthest" points, the least cosine-similar



# Single Link Agglomerative Clustering

- Use maximum similarity of pairs:

$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)$$

- Can result in "straggly" (long and thin) clusters due to chaining effect.
  - Appropriate in some domains, such as clustering islands: "Hawaii clusters"
- After merging $c_i$ and $c_j$, the similarity of the resulting cluster to another cluster, $c_k$, is:

$$sim((c_i \cup c_j), c_k) = \max(sim(c_i, c_k), sim(c_j, c_k))$$

# Complete Link Agglomerative Clustering

- Use minimum similarity of pairs:

$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

- Makes "tighter," spherical clusters that are typically preferable.
- After merging $c_i$ and $c_j$, the similarity of the resulting cluster to another cluster, $c_k$, is:

$$sim((c_i \cup c_j), c_k) = \min(sim(c_i, c_k), sim(c_j, c_k))$$

# Evaluation of Clustering

- Internal criterion: A good clustering will produce high quality clusters in which:
  - the intra-class (that is, intra-cluster) similarity is high
  - the inter-class similarity is low
  - The measured quality of a clustering depends on both the document representation and the similarity measure used
- External criterion: The quality of a clustering is also measured by its ability to discover some or all of the hidden patterns or latent classes
  - Assessable with gold standard data

114

## External Evaluation of Cluster Quality

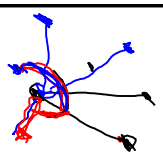
- Assesses clustering with respect to ground truth
- Assume that there are C gold standard classes, while our clustering algorithms produce *k* clusters, $\pi_1, \pi_2, \ldots, \pi_k$ with $n_i$ members.
- Simple measure: purity, the ratio between the dominant class in the cluster $\pi_i$ and the size of cluster $\pi_i$

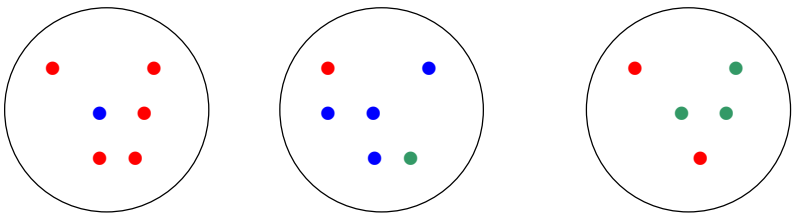
$$Purity(\pi_i) = \frac{1}{n_i} \max_j (n_{ij}) \quad j \in C$$

- Others are entropy of classes in clusters (or mutual information between classes and clusters)

115

## Purity





Cluster I  Cluster II  Cluster III

Cluster I: Purity = 1/6 (max(5, 1, 0)) = 5/6

Cluster II: Purity = 1/6 (max(1, 4, 1)) = 4/6

Cluster III: Purity = 1/5 (max(2, 0, 3)) = 3/5

116

# Federated Search: Resource Representation

- Previous Research on Resource Representation

**Resource descriptions of words and the occurrences**

- **STARTS protocol** (Gravano et al., 1997): Cooperative protocol
- **Query-Based Sampling** (Callan et al., 1999):
  - Send random queries and analyze returned docs
  - Good for uncooperative environments

**Centralized sample database: Collect docs from Query-Based Sampling (QBS)**

- For query-expansion (Ogilvie & Callan, 2001), not very successful
- Successful utilization for other problems, throughout this proposal

# Federated Search: Resource Representation

- Research on Resource Representation

**Information source size estimation**

Important for resource selection and provide users useful information

- Capture-Recapture Model (Liu and Yu, 1999)
  Use two sets of independent queries, analyze overlap of returned doc ids
  But require large number of interactions with information sources

Sample-Resample Model (Si and Callan, 2003)
**Assume:** Search engine indicates num of docs matching a one-term query
**Strategy:** Estimate df of a term in sampled docs
Get total df from by resample query from source
Scale the number of sampled docs to estimate source size

# Federated Search: Resource Selection

Goal of Resource Selection of Information Source Recommendation

High-Recall**: Select the (few) information sources that have the most relevant documents**

Research on Resource Selection

**Resource selection algorithms that need training data**

- **Decision-Theoretic Framework** (DTF) (Nottelmann & Fuhr, 1999, 2003)

  DTF causes large human judgment costs

- **Lightweight probes** (Hawking & Thistlewaite, 1999)

  Acquire training data in an online manner, large communication costs

# Language Model Resource Selection

$$P(db_i \mid Q) = \frac{P(Q \mid db_i) * P(db_i)}{P(Q)}$$

DB independent constant

$$P(Q \mid db_i) = \prod_{q \in Q} \left( \lambda P(q \mid db_i) + (1-\lambda) P(q \mid G) \right)$$

Calculate on Sample Docs

**In Language Model Framework, $P(C_i)$ is set according to DB Size**

$$P(C_i) = \frac{\hat{N}_{C_i}}{\sum_j \hat{N}_{C_j}}$$

# Federated Search: Resource Selection

## Relevant Doc Distribution Estimation (ReDDE) Algorithm

$$Rel\_Q(i) = \sum_{d \in db_i} P(rel|d) * P(d|db_i) * N_{db_i} \approx \sum_{d \in db_{i\_samp}} P(rel|d) * SF_{db_i}$$

Source Scale Factor: $SF_{db_i} = \frac{\hat{N}_{db_i}}{N_{db_{i\_samp}}}$

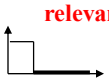
Estimated Source Size

Number of Sampled Docs

Rank on Centralized Complete DB

"Everything at the top is (equally) relevant"

$$P(rel|d) = \begin{cases} C_Q & \text{if } Rank_{CCDB}(Q,d) < ratio * \sum_i N_{db_i} \\ 0 & \text{otherwise} \end{cases}$$



Problem: To estimate doc ranking on Centralized Complete DB

# Federated Search: Result Merging

## Goal of Results Merging

**Make different result lists comparable and merge them into a single list**

**Difficulties:**

- Information sources may use **different retrieval algorithms**
- Information sources have **different corpus statistics**

## Previous Research on Results Merging

**Most accurate methods directly calculate comparable scores**

- **Use same retrieval algorithm and same corpus statistics** (Viles & French, 1997)(Xu and Callan, 1998), need source cooperation
- **Download retrieved docs and recalculate scores** (Kirsch, 1997), large communication and computation costs

# Research Problems (Results Merging)

Research on Results Merging

**Methods approximate comparable scores**

- **Round Robin** (Voorhees et al., 1997), only use source rank information and doc rank information, fast but less effective
- **CORI merging formula** (Callan et al., 1995), linear combination of doc scores and source scores
  - Use linear transformation, a hint for other method
  - Work in uncooperative environment, effective but need improvement

# Federated Search

**In resource representation:**

- Build representations by QBS, collapse sampled docs into centralized sample DB

**In resource selection:**

- Rank sources, calculate centralized scores for docs in centralized sample DB

**In results merging:**

- Find overlap docs, build linear models, estimate centralized scores for all docs