Retrieval Model:
Vector Space Model

Pu-Jen Cheng

Formal Formulation of IR

- Vocabulary $V=\{w_1, w_2, \ldots, w_N\}$ of language
- Query $q = q_1,\ldots,q_m$, where $q_i \in V$
- Document $d_i = d_{i1},\ldots,d_{in}$, where $d_{ij} \in V$
- Collection $C= \{d_1, \ldots, d_k\}$
- Set of **relevant** documents $R(q) \subseteq C$
  - Generally unknown and user-dependent
  - Query is a "hint" on which doc is in $R(q)$
- Task = compute $R'(q)$, an "approximate $R(q)$"

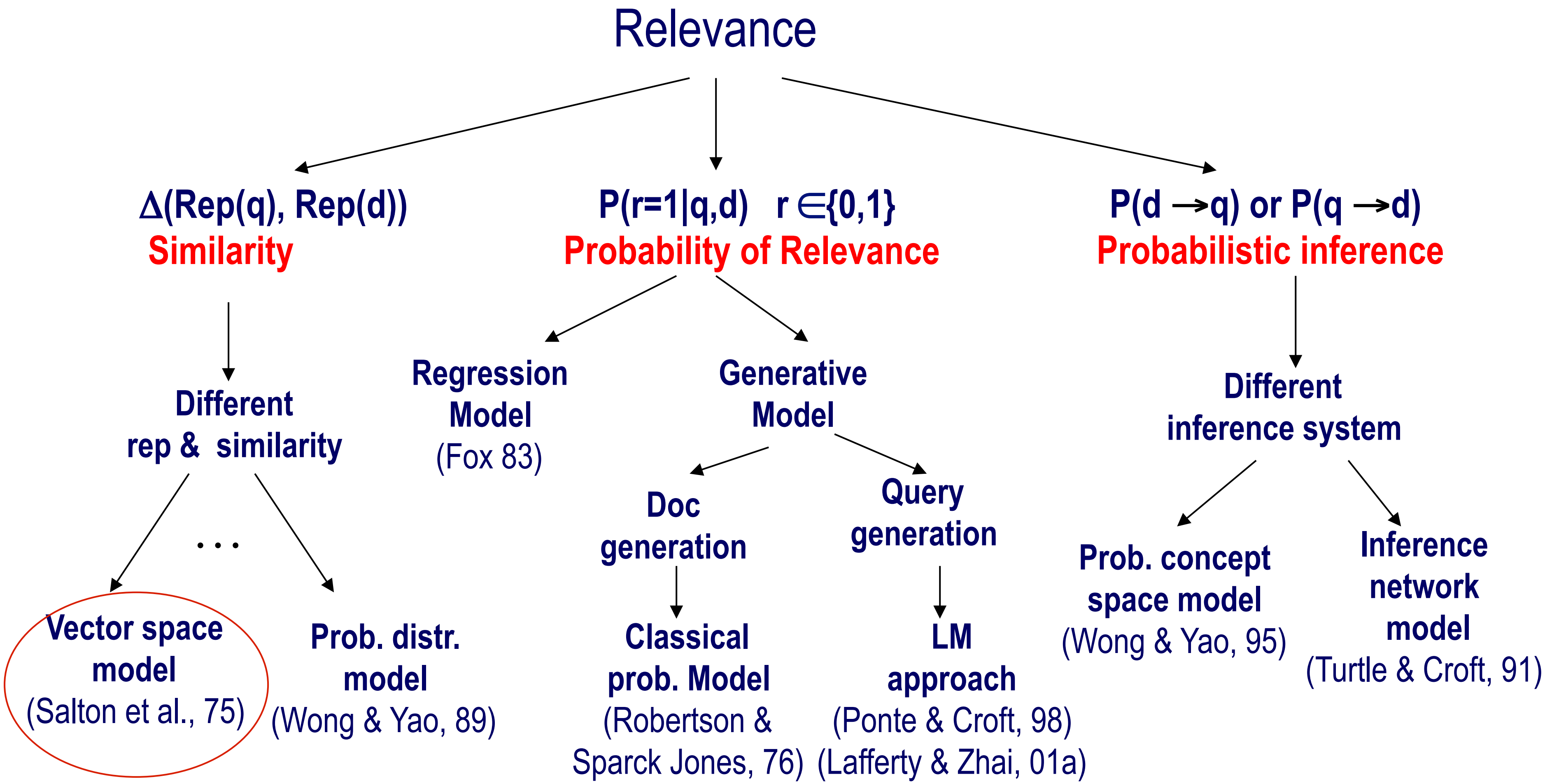
Computing R(q)

- Document ranking
  - $R'(q) = \{d \in C | f(d,q) > \theta\}$, where $f(d,q) \in \Re$ is a relevance measure function; θ is a cutoff
  - System must decide if one doc is more likely to be relevant than another ("relative relevance")
- All we need is

  "A relevance measure function f"
  which satisfies
  For all q, d_1, d_2,
  $f(q,d_1) > f(q,d_2)$ iff $p(Rel|q,d_1) > p(Rel|q,d_2)$

(Rel: relevance)

The Notion of Relevance



Lecture Plan

- Vector Space Model
- Relevance Feedback in VSM
  - Rocchio Feedback
- Dimension Reduction
  - Latent Semantic Indexing

Lecture Plan

- Vector Space Model
- Relevance Feedback in VSM
  - Rocchio Feedback
- Dimension Reduction
  - Latent Semantic Indexing

The Basic Question

Given a query, how do we know if document A is more relevant than B?

One Possible Answer

If document A uses more query words than document B

(Word usage in document A is more similar to that in query)

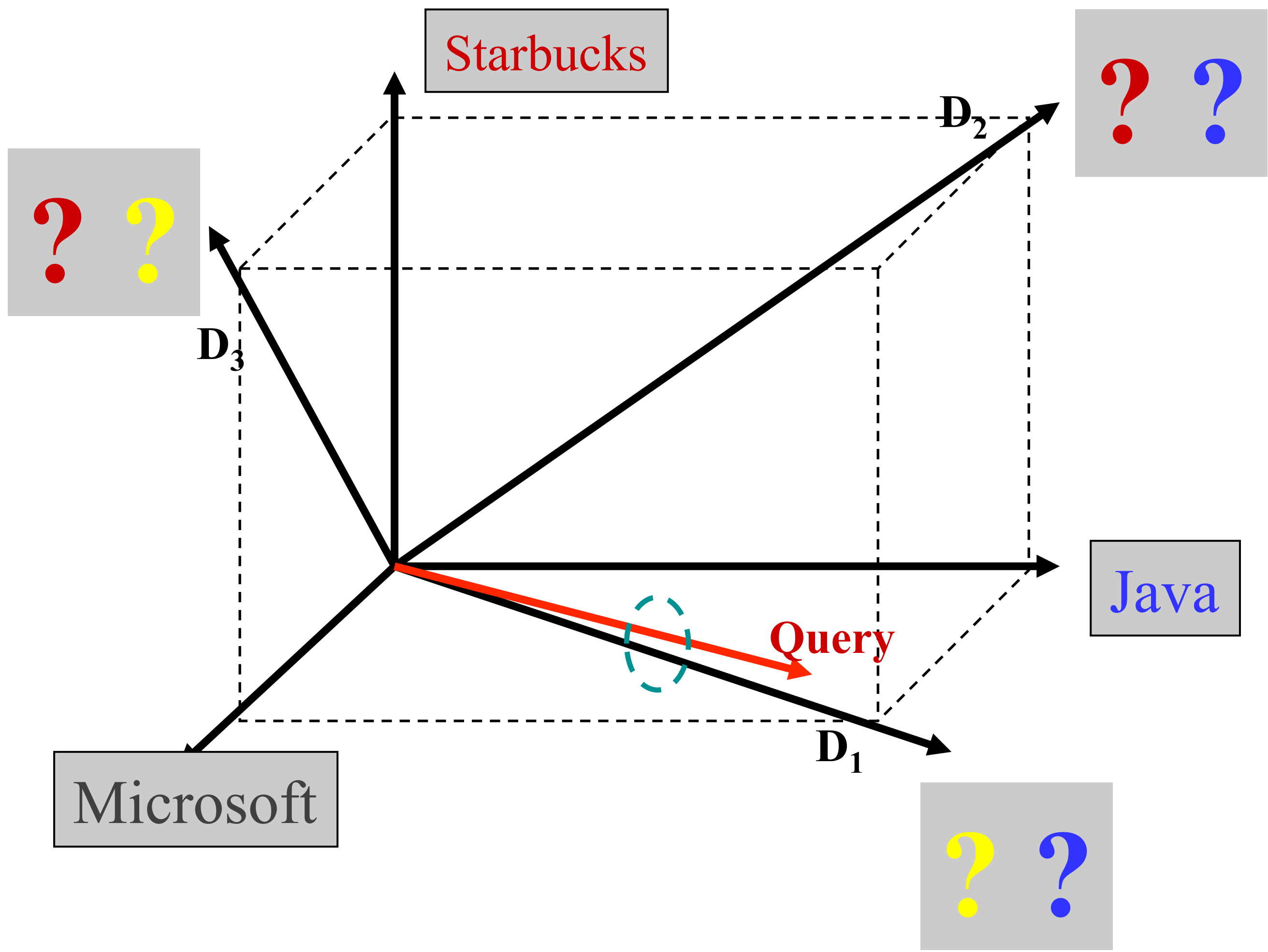
Relevance = Similarity

- Assumptions
  - Query and document are represented similarly
  - A query can be regarded as a document
  - Relevance(d,q) $\propto$ similarity(d,q)
- $R(q) = \{d \in C | f(d,q) > \theta\}$, $f(q,d) = \Delta(Rep(q), Rep(d))$
- Key issues
  - How to represent query/document?
  - How to define the similarity measure Δ?

Vector Space Model (VSM)

- Represent a doc/query by a term vector
  - Term: basic concept, e.g., word or phrase
  - Each term defines one dimension
  - N terms define a high-dimensional space
  - Element of vector corresponds to term weight
  - E.g., $d=(x_1,\ldots,x_N)$, x_i is importance of term I
- Relevance
  - Measured by the distance between query and document vectors in the vector space

VSM: Illustration



What the VSM does not say

- How to define/select the basic concept
  - Concepts are assumed to be orthogonal
- How to assign weights
  - Weight in query indicates importance of term
  - Weight in doc indicates how well the term characterizes the doc
- How to define the similarity/distance measure

What is a good basic concept?

- Orthogonal
  - Linearly independent basis vectors
  - Non-overlapping in meaning
- No ambiguity
- Weights can be assigned automatically and hopefully accurately
- Many possibilities: Words, stemmed words, phrases, latent concept, …

How to assign weights?

- Very important!
- Why weighting
  - Query side: Not all terms are equally important
  - Doc side: Some terms carry more information about contents
- How?
  - Two basic heuristics
    - TF (Term Frequency) = Within-doc-frequency
    - IDF (Inverse Document Frequency)
  - TF normalization

TF Weighting

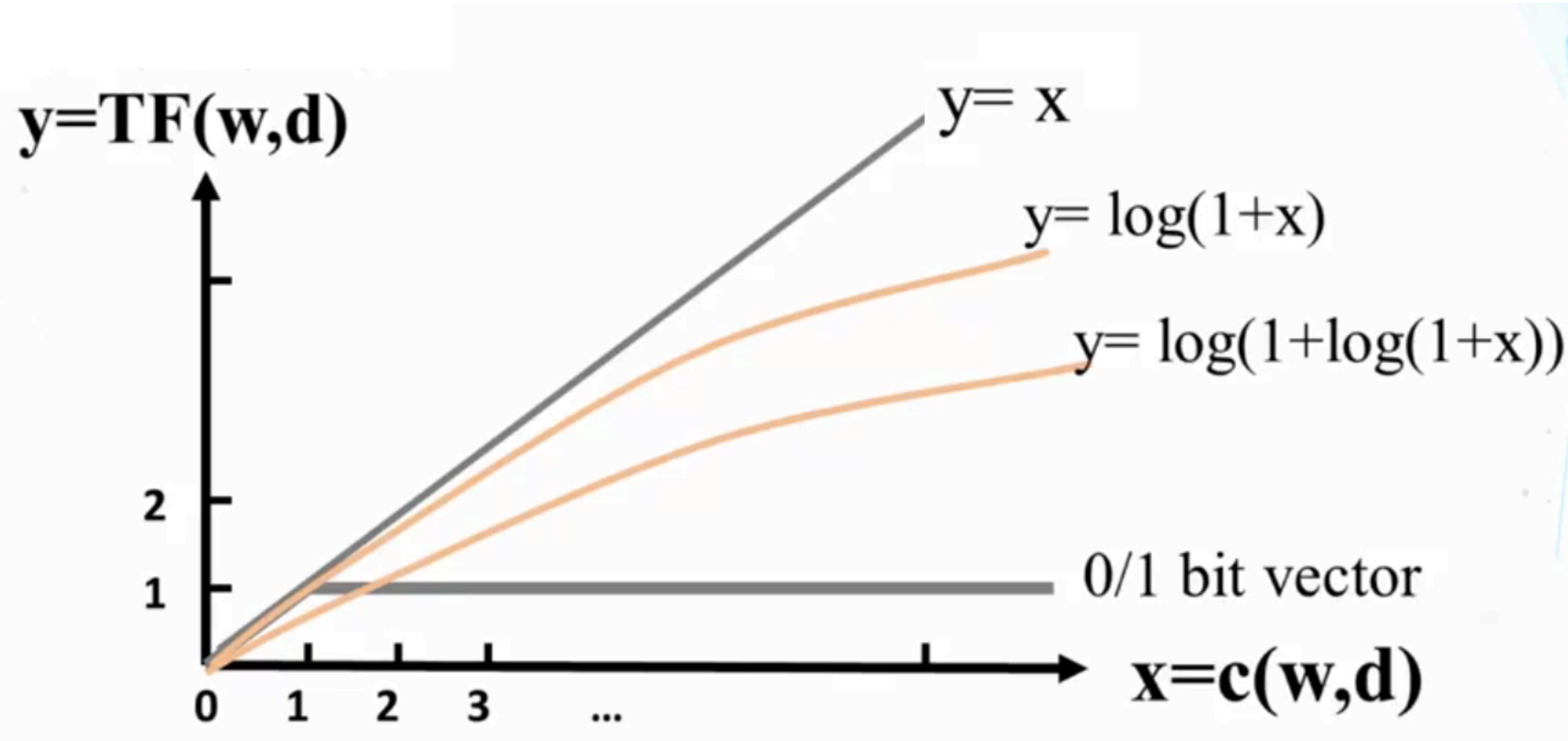
- **Idea: A term is more important if it occurs more frequently in a document**
  - **Restrict the contribution of high-frequency term**
- **Formulas: Let c(t,d) be the frequency count of term *t* in doc *d***
  - **Raw TF:** TF(t,d) = c(t,d)
  - **Maximum frequency normalization:**

    TF(t,d) = 0.5 +0.5*c(t,d) /MaxFreq(d)

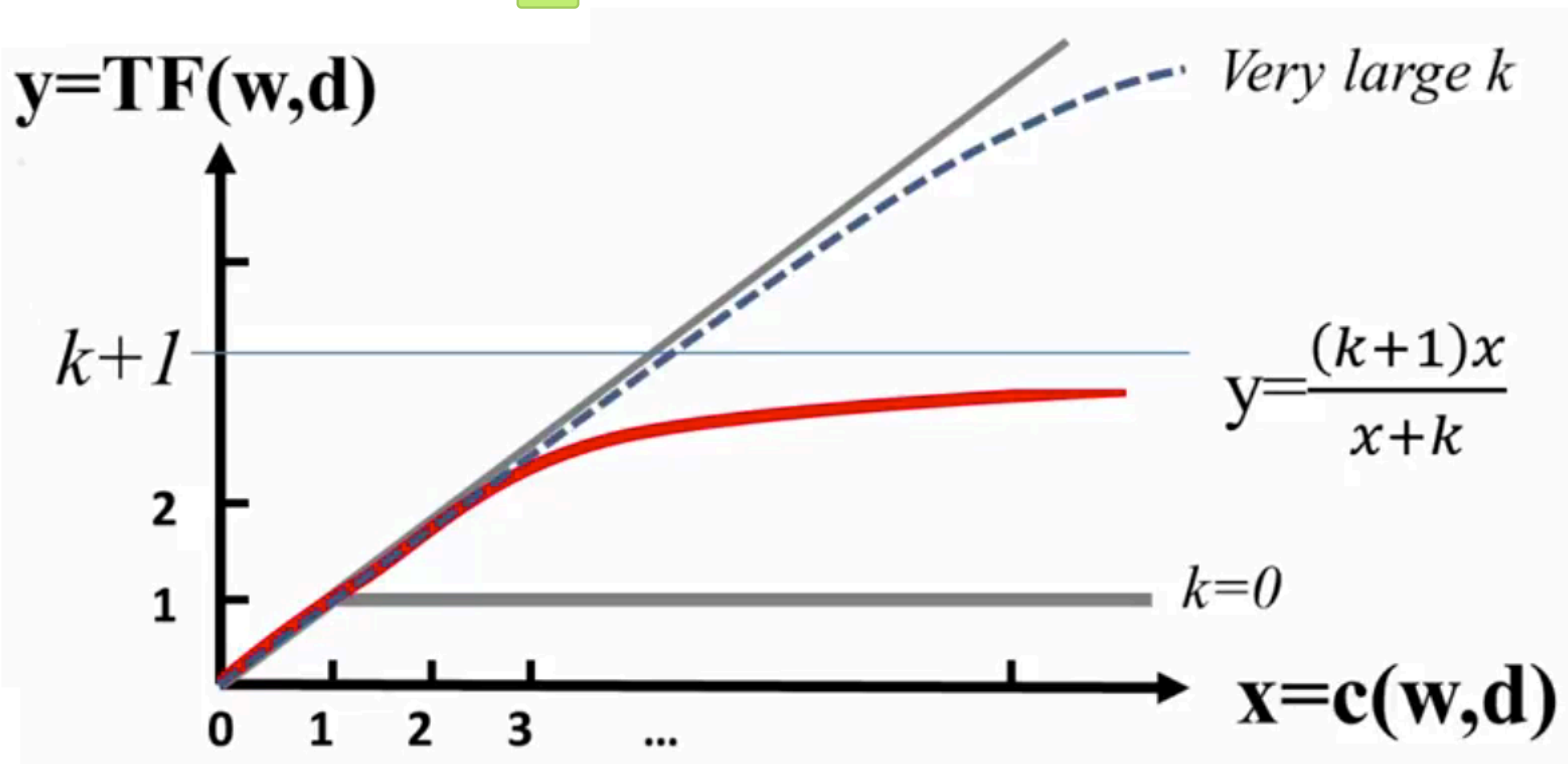
    $$\text{ntf}_{t,d} = a + (1-a)\frac{\text{tf}_{t,d}}{\text{tf}_{\max}(d)}$$
  - **"Okapi/BM25 TF":**

    TF(t,d) = (k+1)c(t,d) / (c(t,d)+k) ($k \geq 0$)
- **Normalization of TF is very important!**

TF Normalization - sublinear transformation



Okapi/BM25 TF Normalization

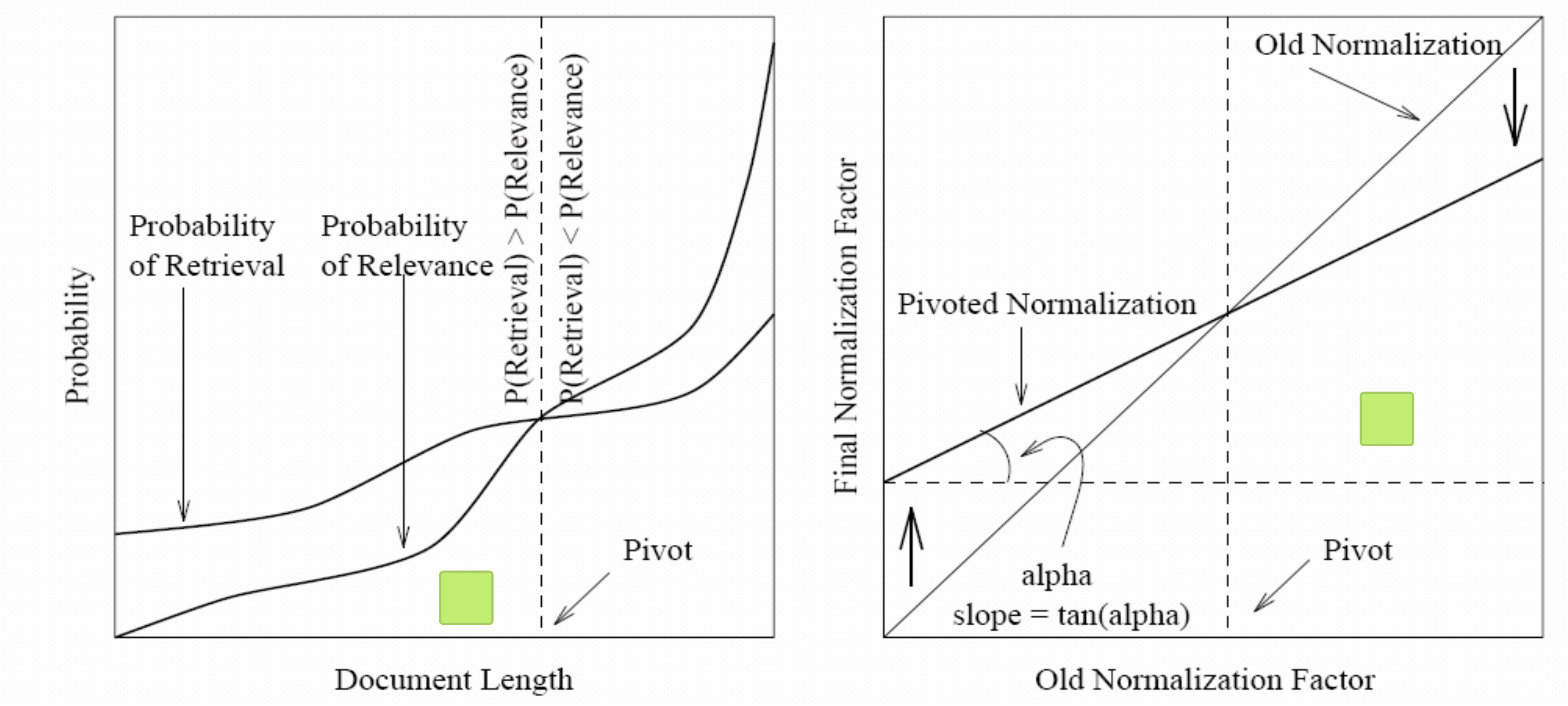


Doc Length Normalization

- Why?
  - Document length variation
  - Repeated occurrences are less informative than the first occurrence
- Two views of document length
  - A doc is long because it uses more words
  - A doc is long because it has more contents
- Generally penalize long doc, but avoid over-penalizing (pivoted normalization)

Pivoted Document Length Normalization

cosine normalization : $\sqrt{w_1{}^2 + w_2{}^2 + \ldots + w_t{}^2}$

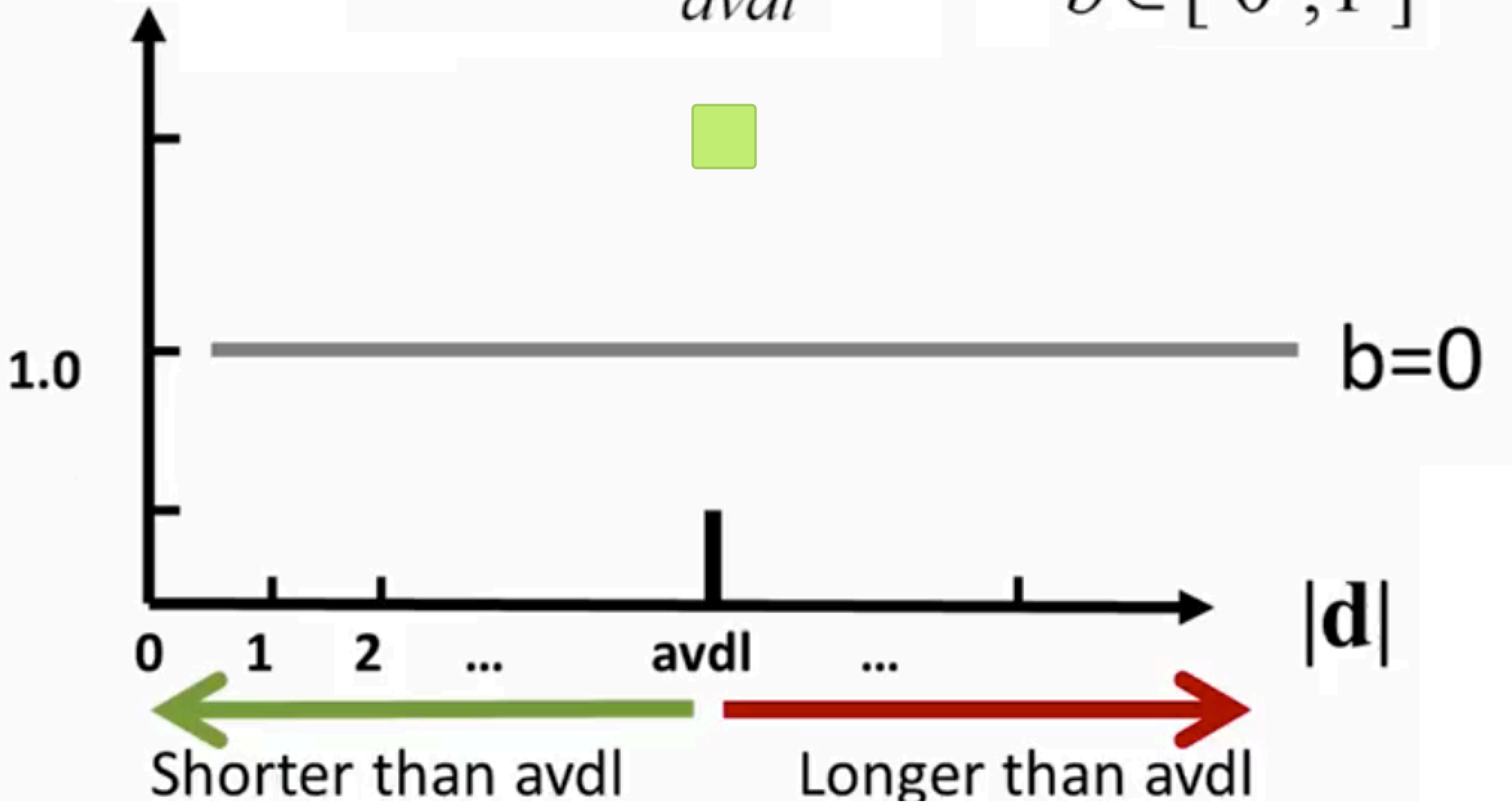


- **pivoted normalization =**
  (1.0 – slope) * pivot + slope * old normalization

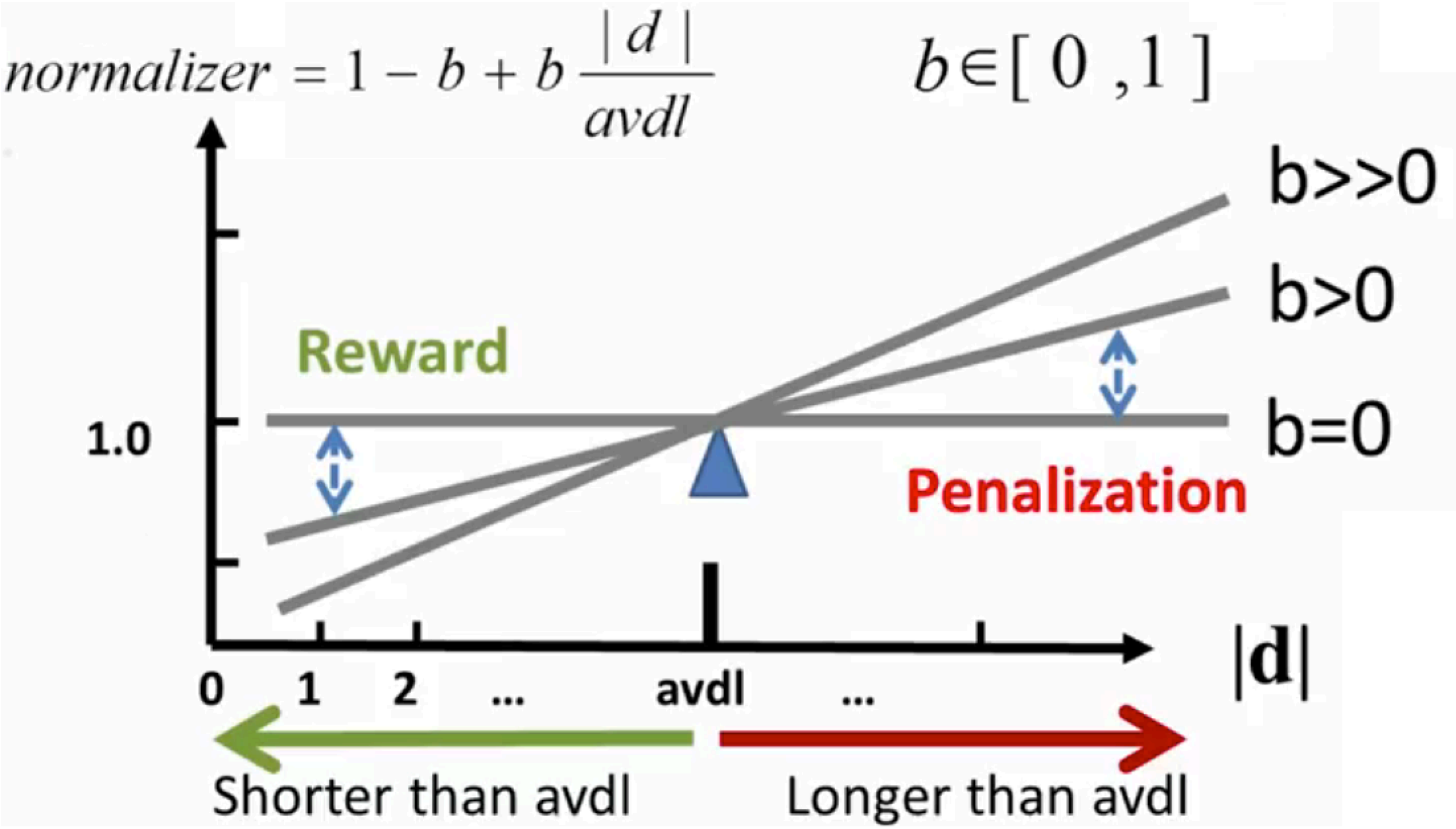
A. Singhal, C. Buckley, and M. Mitra. *Pivoted document length normalization*. In Proc. of SIGIR, 1996.

Okapi/BM25
Doc Length Normalization

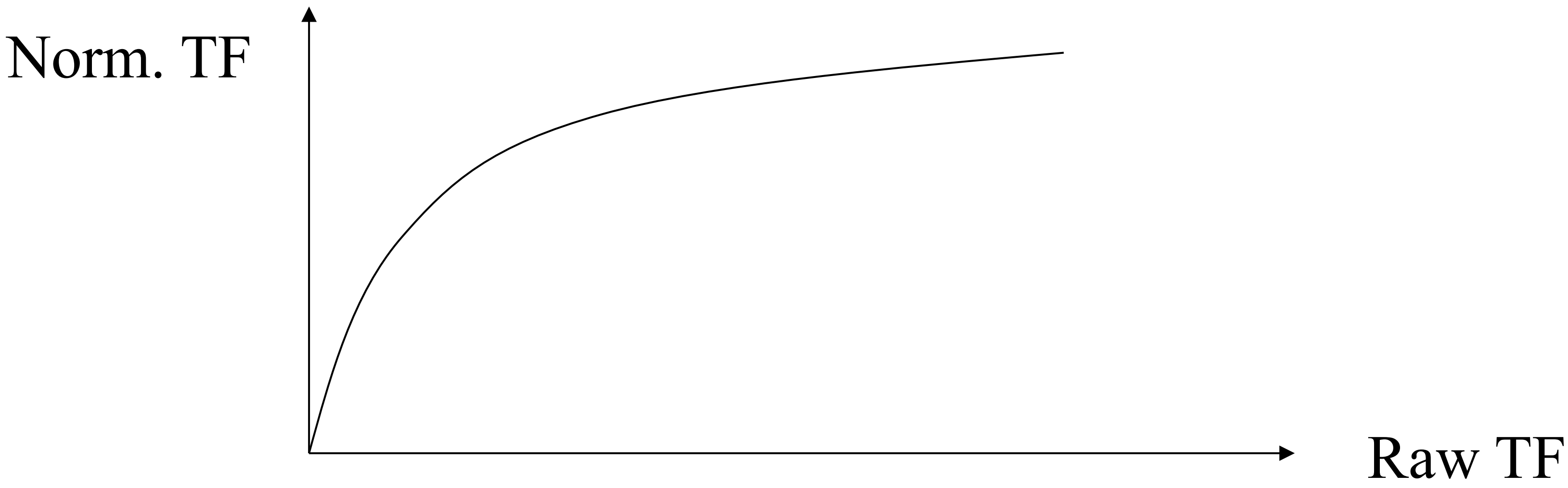
$$normalizer = 1 - b + b\frac{|d|}{avdl} \qquad b \in [0, 1]$$



Okapi/BM25
Doc Length Normalization



Okapi/BM25
Doc Length Normalization



Using avg. doc length to regularize normalization

Norm TF = Raw TF / (1-b+b*doclen/avgdoclen)

b varies from 0 to 1

Collection vs. Document Frequency

- The collection frequency of a term is the number of occurrences of the term in the collection
- Example:

Word	Collection frequency	Document frequency
insurance	10440	3997
try	10422	8760

- Which word is a better search term (and should get a higher weight)?

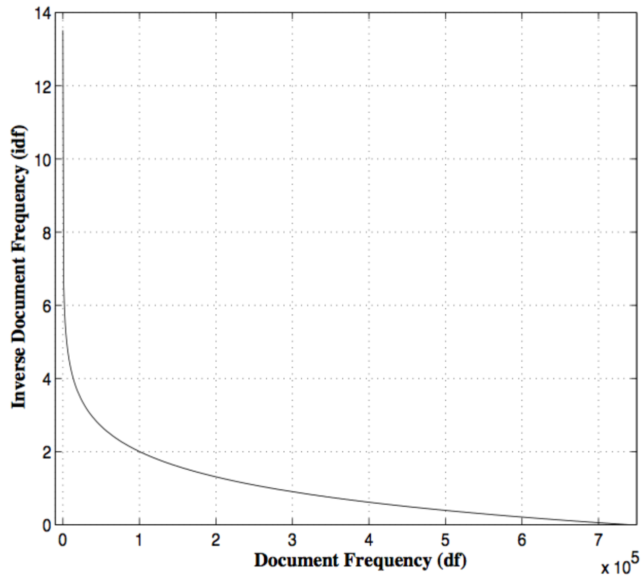
IDF Weighting

- **Idea: A term is more discriminative if it occurs only in fewer documents**

- **Formula:**

$$IDF(t) = \log(n/k)$$

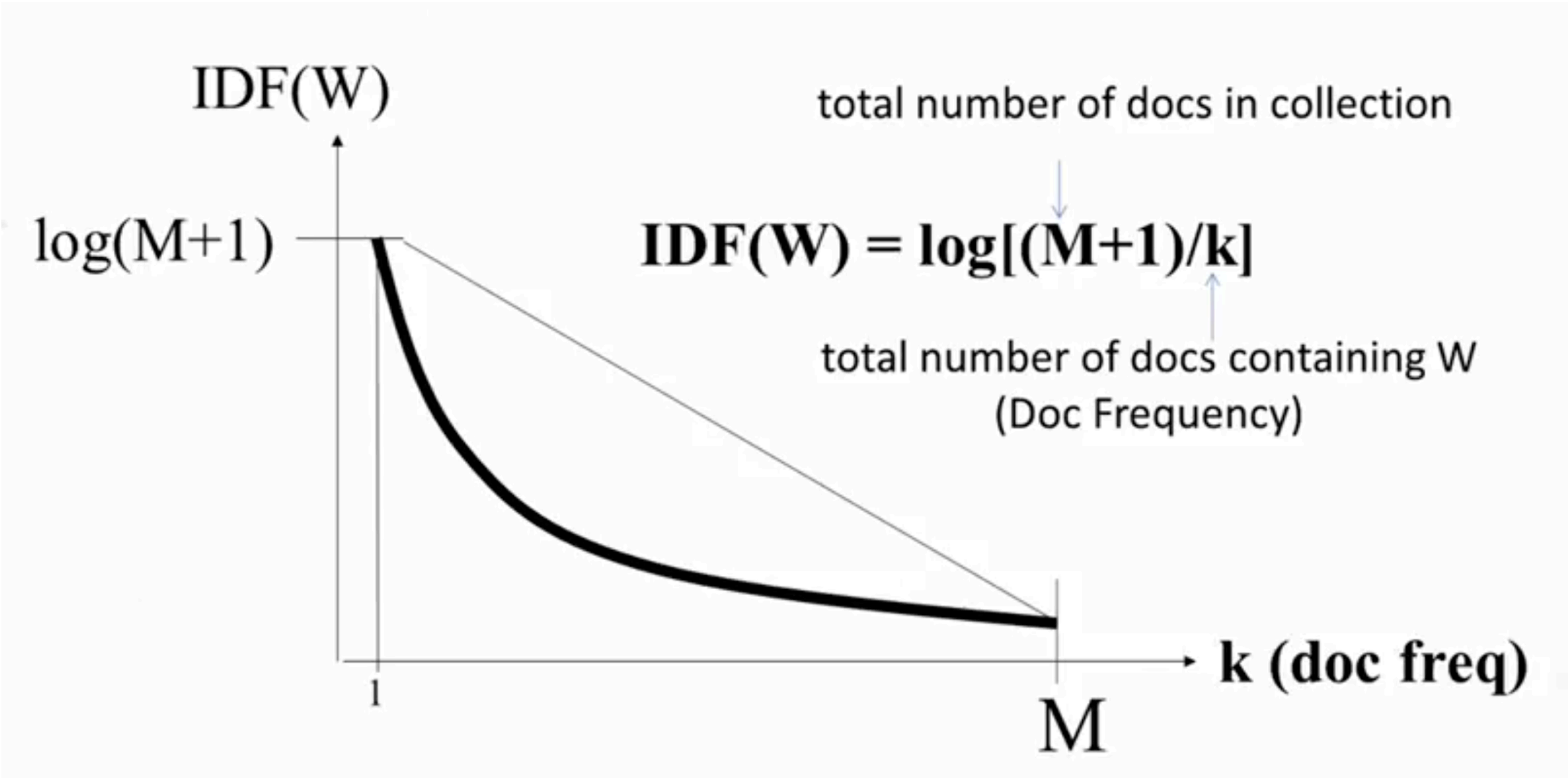
n – total number of docs
k – # docs with term t (doc freq)



- **Why log(n/k) instead of n/k**
  - **to "dampen" the effect of IDF**
  - **from information theory**
    - $P(t) = P(\text{a random doc d containing t}) \approx k/n$
    - $IDF(t) = -\log P(t)$

self-information ($I(x) = -\log P(x)$): measure of the info. content associated with the outcome of a random variable

Additive: $IDF(t1 \wedge t2) = IDF(t1) + IDF(t2)$ (t1,t2 independent)


IDF(W)
log(M+1)
1
M
k (doc freq)
total number of docs in collection
IDF(W) = log[(M+1)/k]
total number of docs containing W
(Doc Frequency)

TF-IDF Weighting

- **TF-IDF weighting :** weight(t,d)=TF(t,d)*IDF(t)
  - **Common in doc → high tf → high weight**
  - **Rare in collection→ high idf→ high weight**

How to measure similarity?

$$\vec{D}_i = (w_{i1}, \ldots, w_{iN})$$

$$\vec{Q} = (w_{q1}, \ldots, w_{qN})$$ $w = 0$ if a term is absent

Dot product similarity: $$sim(\vec{Q}, \vec{D}_i) = \sum_{j=1}^{N} w_{qj} * w_{ij}$$

Cosine: $$sim(\vec{Q}, \vec{D}_i) = \frac{\sum_{j=1}^{N} w_{qj} * w_{ij}}{\sqrt{\sum_{j=1}^{N} (w_{qj})^2 * \sum_{j=1}^{N} (w_{ij})^2}}$$

(= normalized dot product)

Euclidean distance: $$dist(\vec{Q}, \vec{D}_i) = \sqrt{\sum_{j=1}^{N} \left(w_{qj} - w_{ij}\right)^2}$$

VS Example: Raw TF & Dot Product

doc1

information
retrieval
search
engine
information

doc2

travel
information

map
travel

doc3

government
president
congress

……

query="information retrieval"

Sim(q,doc1)=4.8*2.4+4.5*4.5

Sim(q,doc2)=2.4*2.4

Sim(q,doc3)=0

	info	retrieval	travel	map	search	engine	govern	president	congress
IDF(faked)	2.4	4.5	2.8	3.3	2.1	5.4	2.2	3.2	4.3
doc1	2(4.8)	1(4.5)			1(2.1)	1(5.4)			
doc2	1(2.4)		2 (5.6)	1(3.3)					
doc3							1 (2.2)	1(3.2)	1(4.3)
query	1(2.4)	1(4.5)							

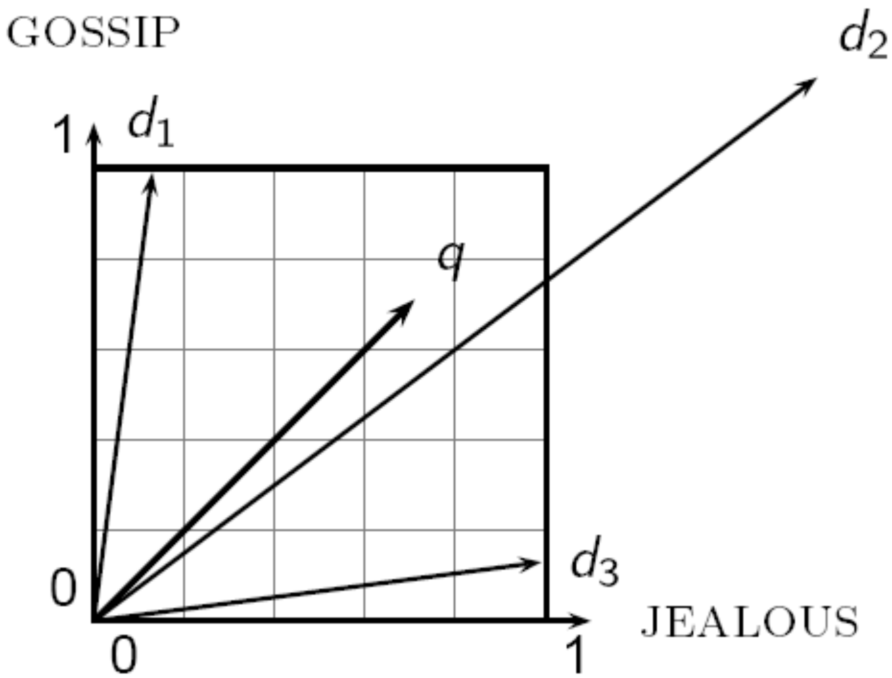
Dot Product and Euclidean Distance are not Good Ideas

Dot product (inner product) favors long documents

Euclidean Distance

The distance between q and d_2 is large even though the distributions of their terms are very similar.

Think about: take a document and append it to itself.



Lecture Plan

- Vector Space Model
- Relevance Feedback in VSM
  - Rocchio Feedback
- Dimension Reduction
  - Latent Semantic Indexing

Relevance Feedback in VSM

- Basic setting: Learn from examples
  - Positive examples: docs known to be relevant
  - Negative examples: docs known to be non-relevant
  - How do you learn from this to improve performance?
- General method: Query modification
  - Adding new (weighted) terms
  - Adjusting weights of old terms
  - Doing both
- The most well-known and effective approach is Rocchio

Rocchio Feedback

- **The Rocchio algorithm uses the vector space model to pick a relevance fed-back query**
- **Rocchio seeks the query $\vec{q}_{opt}$ that maximizes**

$$\vec{q}_{opt} = \arg\max_{\vec{q}} [\cos(\vec{q}, \vec{\mu}(C_r)) - \cos(\vec{q}, \vec{\mu}(C_{nr}))]$$

C_r = set of truly relevant doc vectors
C_{nr} = set of truly irrelevant doc vector

centroid: $\vec{\mu}(C) = \frac{1}{|C|} \sum_{d \in C} \vec{d}$

- **Tries to separate docs marked relevant and non-relevant**

$$\vec{q}_{opt} = \frac{1}{|C_r|} \sum_{\vec{d}_j \in C_r} \vec{d}_j - \frac{1}{|C_{nr}|} \sum_{\vec{d}_j \notin C_r} \vec{d}_j$$

- **Problem: we do not know the truly relevant docs**

Rocchio Feedback: Formula

D_r = set of <u>known</u> relevant doc vectors
D_{nr} = set of <u>known</u> irrelevant doc vectors
Different from C_r and C_{nr}

Parameters

New query

$$\vec{q}_m = \alpha \vec{q} + \frac{\beta}{|D_r|} \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \frac{\gamma}{|D_n|} \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

Origial query

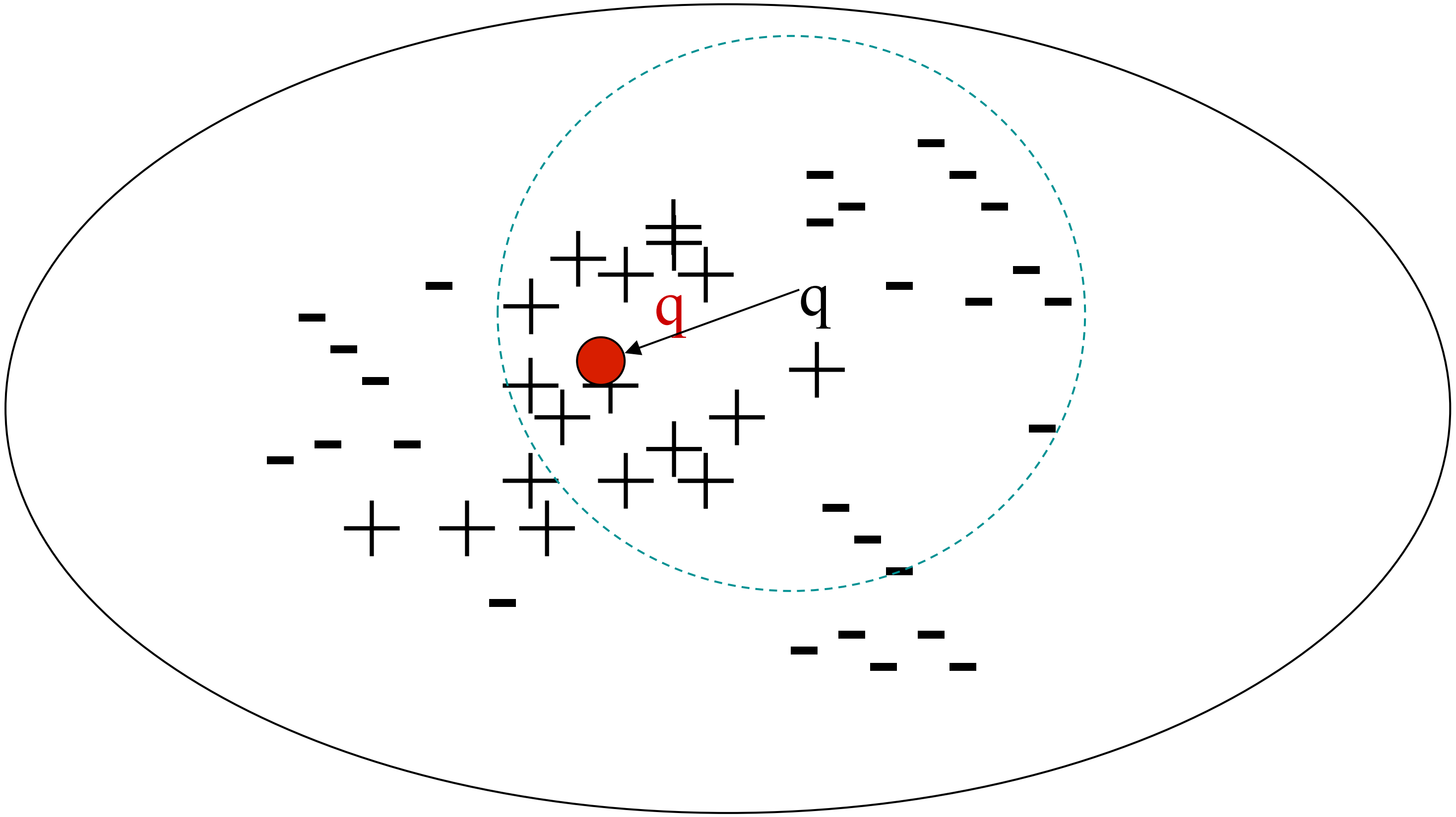
Rel docs

Non-rel docs

term reweighting

query expansion (may introduce new terms)

Rocchio Feedback: Illustration



Rocchio in Practice

- Negative (non-relevant) examples are not very important (why?)
- Often project the vector onto a lower dimension (i.e., consider only a small number of words that have high weights in the centroid vector) (efficiency concern)
- Avoid "training bias" (keep relatively high weight on the original query weights) (why?)
- Can be used for relevance feedback and pseudo feedback
- Robust and effective for qualified feedback info.

Extension of VSM

- Alternative similarity measures
  - Many other choices (tend not to be very effective)
  - P-norm (Extended Boolean)
- Alternative representation
  - Many choices (performance varies a lot)
  - Latent Semantic Indexing (LSI)
- Generalized vector space model
  - Theoretically interesting, not seriously evaluated

Advantages of VSM

- Empirically effective! (Top TREC performance)
  - its *partial matching* strategy allows retrieval of documents that *approximate* the query conditions
- Intuitive
- Easy to implement
- Well-studied/Most evaluated
- The SMART system
  - Developed at Cornell: 1960-1999
  - Still widely used
- Warning: Many many variants of TF-IDF!

SMART notation for tf-idf variants

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1+\log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5+\frac{0.5\times\mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0,\log\frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$ (Section 6.4.4)
b (boolean)	$\begin{cases}1 & \text{if } \mathrm{tf}_{t,d}>0\\ 0 & \text{otherwise}\end{cases}$			b (byte size)	$1/\mathit{CharLength}^{\alpha}$, $\alpha<1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

CharLength: the number of characters in a document.

What works the best?

tf	is the term's frequency in document
qtf	is the term's frequency in query
N	is the total number of documents in the collection
df	is the number of documents that contain the term
dl	is the document length (in bytes), and
$avdl$	is the average document length

Okapi weighting based document score: [23]

$$\sum_{t \in Q,D} \ln \frac{N - df + 0.5}{df + 0.5} \cdot \frac{(k_1 + 1)tf}{(k_1(1 - b + b\frac{dl}{avdl})) + tf} \cdot \frac{(k_3 + 1)qtf}{k_3 + qtf}$$

k_1 (between 1.0–2.0), b (usually 0.75), and k_3 (between 0–1000) are constants.

Pivoted normalization weighting based document score: [30]

$$\sum_{t \in Q,D} \frac{1 + \ln(1 + \ln(tf))}{(1 - s) + s\frac{dl}{avdl}} \cdot qtf \cdot \ln \frac{N + 1}{df}$$

s is a constant (usually 0.20).

- **Use single words**
- **Use stat. phrases**
- **Remove stop words**
- **Stemming**
- **Others(?)**

A. Singhal, "Modern Information Retrieval: A Brief Overview," IEEE Data Engineering Bulletin, vol. 24(4), pp. 35-43, 2001

Disadvantages of VSM

- Assume term independence
  - Solutions: LSI, generalized VSM
- Assume query and document to be the same
- Lack of "predictive adequacy"
  - Arbitrary term weighting
  - Arbitrary similarity measure
- Lots of parameter tuning!

Lecture Plan

- Vector Space Model
- Relevance Feedback in VSM
  - Rocchio Feedback
- Dimension Reduction
  - Latent Semantic Indexing

Problems with Lexical Semantics

- Association in natural language
  - Synonymy: Different terms may have an identical or a similar meaning
  - VSM has no associations between terms

    (VSM assumes term independence)

$$\mathrm{sim}_{\mathrm{true}}(d, q) > \cos(\angle(\vec{d}, \vec{q}))$$

Underestimate!
Mismatching problem

Road

- Linear Algebra
  - Eigenvalues & Eigenvectors
  - Eigen/diagonal Decomposition
  - Singular Value Decomposition
- Latent Semantic Indexing

Eigenvalues & Eigenvectors

- **Eigenvectors (for a square $m \cdot m$ matrix S)**

$$\mathbf{S}\mathbf{v} = \lambda\mathbf{v}$$

(right) eigenvector — eigenvalue

$\mathbf{v} \in \mathbb{R}^m \neq \mathbf{0}$ — $\lambda \in \mathbb{R}$

Example

$$\begin{pmatrix} 6 & -2 \\ 4 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix} = 2 \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$\mathbf{S}\mathbf{v} = \lambda\mathbf{v} \iff (\mathbf{S} - \lambda\mathbf{I})\,\mathbf{v} = \mathbf{0}$$

- **How many eigenvalues are there at most?**

only has a non-zero solution if $|\mathbf{S} - \lambda\mathbf{I}| = 0$

This is a mth order equation in λ which can have at most m distinct solutions

(roots of the characteristic polynomial)

Matrix-vector Multiplication

$$S = \begin{pmatrix} 30 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$ has eigenvalues 30, 20, 1 with corresponding eigenvectors

$$\vec{x_1} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \vec{x_2} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \text{ and } \vec{x_3} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Any vector can be viewed as a combination of eigenvectors:

$$\vec{v} = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} = 2\vec{x_1} + 4\vec{x_2} + 6\vec{x_3}$$

$$\begin{aligned} S\vec{v} &= S(2\vec{x_1} + 4\vec{x_2} + 6\vec{x_3}) \\ &= 2S\vec{x_1} + 4S\vec{x_2} + 6S\vec{x_3} \\ &= 2\lambda_1\vec{x_1} + 4\lambda_2\vec{x_2} + 6\lambda_3\vec{x_3} \\ &= 60\vec{x_1} + 80\vec{x_2} + 6\vec{x_3}. \end{aligned}$$

前往頁面 4

The action of S on the vector is determined by its eigenvalues and eigenvectors

The effect of small eigenvalues is small

Eigenvalues & Eigenvectors

For symmetric matrices, eigenvectors for distinct eigenvalues are **orthogonal**

$$S = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \qquad S - \lambda I = \begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix} \Rightarrow$$

$$|S - \lambda I| = (2-\lambda)^2 - 1 = 0.$$

The eigenvalues are 1 and 3

The eigenvectors are orthogonal $\begin{pmatrix} 1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

Eigen/diagonal Decomposition

- Let $\mathbf{S} \in \mathbb{R}^{m \times m}$ be a square matrix with $\boldsymbol{m}$ linearly independent eigenvectors
- There exists an eigen decomposition

$$\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$$

- Columns of U are the eigenvectors of S
- Diagonal elements of $\mathbf{\Lambda}$ are eigenvalues of S

$$\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_m), \quad \lambda_i \geq \lambda_{i+1}$$

If U has the eigenvectors of S as columns

$$U = (\vec{u_1}\ \vec{u_2} \cdots \vec{u_M})$$

Then

$$\begin{aligned} SU &= S\,(\vec{u_1}\ \vec{u_2} \cdots \vec{u_M}) \\ &= (\lambda_1 \vec{u_1}\ \lambda_2 \vec{u_2} \cdots \lambda_M \vec{u_M}) \\ &= (\vec{u_1}\ \vec{u_2} \cdots \vec{u_M}) \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \cdots & \\ & & & \lambda_M \end{pmatrix} \end{aligned}$$

So $\mathrm{SU}=\mathrm{U}\Lambda \rightarrow \mathrm{SUU}^{-1}=\mathrm{U}\Lambda\mathrm{U}^{-1} \rightarrow \mathrm{S}=\mathrm{U}\Lambda\mathrm{U}^{-1}$

$(\mathrm{UU}^{-1}=\mathrm{I})$

Example

Recall $S = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ eigenvalues are 1 and 3

eigenvectors $\begin{pmatrix} 1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

$$S = U \Lambda U^{=1} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

Normalize each eigenvector with $\sqrt{2}$

$$S = Q \Lambda Q^{=1} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Q $\qquad\qquad \Lambda \qquad\qquad (Q^{-1} = Q^T)$

Symmetric Eigen Decomposition

- Let $S \in \mathbb{R}^{m \times m}$ be a symmetric matrix
- There exists a (unique) eigen decomposition

$$S = Q \Lambda Q^T$$

where Q is orthogonal:

- $Q^{-1} = Q^T$
- Columns of Q are normalized eigenvectors
- Columns are orthogonal.

Singular Value Decomposition

For an $M \cdot N$ matrix $\mathbf{A}$ of rank r there exists a factorization (Singular Value Decomposition, **SVD**) as follows:

$$\mathbf{A = U_{MxM} \Sigma_{MxN} V^T_{NxN}}$$

The *rank* of a matrix is the number of linearly independent rows (or columns) in it

- $\mathbf{AA^T = Q \Lambda Q^T}$
- $\mathbf{AA^T = (U \Sigma V^T)(U \Sigma V^T)^T = (U \Sigma V^T)(V \Sigma U^T) = U \Sigma^2 U^T}$

The columns of $\mathbf{U}$ are orthogonal eigenvectors of $\mathbf{AA^T}$.

The columns of $\mathbf{V}$ are orthogonal eigenvectors of $\mathbf{A^TA}$.

Eigenvalues $\lambda_1 \ldots \lambda_r$ of $\mathbf{AA^T}$ are the eigenvalues of $\mathbf{A^TA}$.

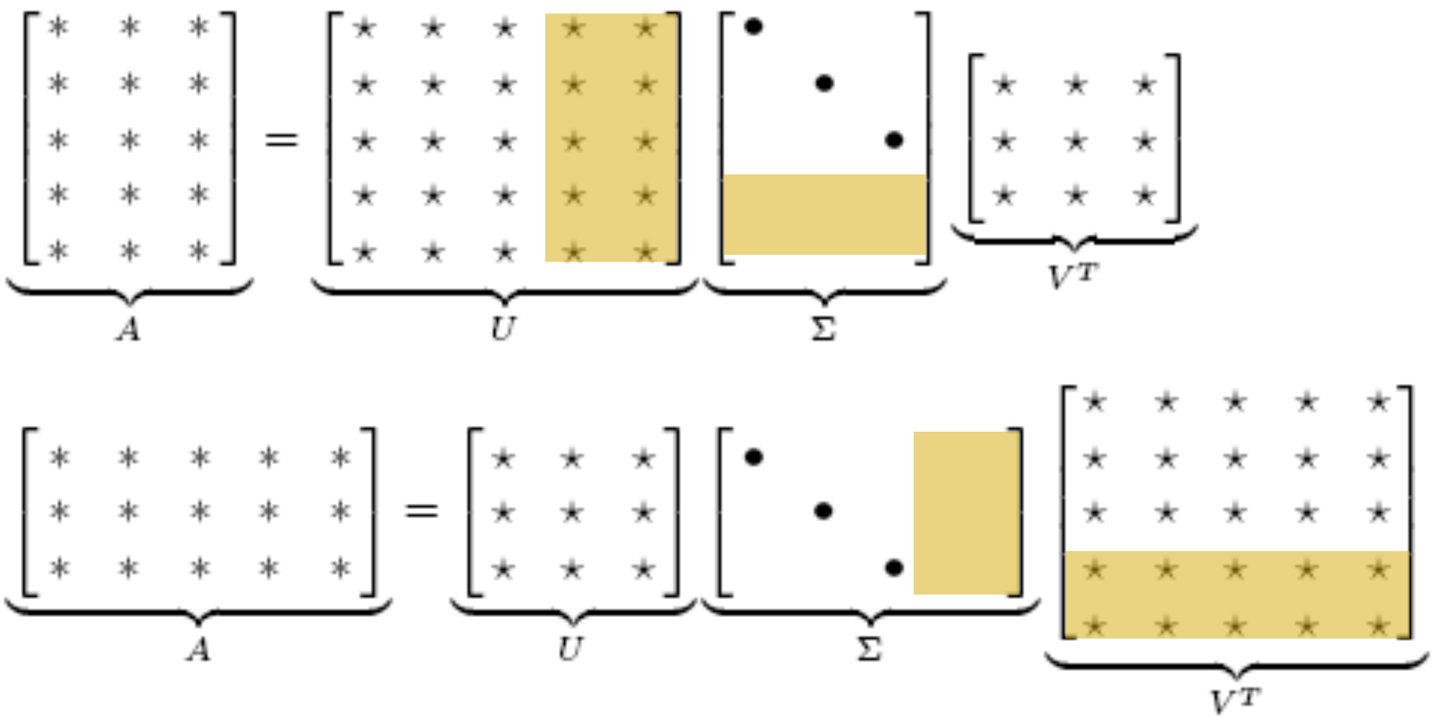
$$\sigma_i = \sqrt{\lambda_i} \quad \lambda_i \geq \lambda_{i+1}$$

$$\Sigma = diag(\sigma_1, \ldots, \sigma_r)$$

Singular values

Singular Value Decomposition

- **Illustration of SVD dimensions and sparseness**



SVD Example

Let $A = \begin{bmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$

Thus M=3, N=2. Its SVD is

$$\begin{bmatrix} 0 & 2/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & 1/\sqrt{6} & -1/\sqrt{3} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{3} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

Typically, the singular values arranged in decreasing order.

Low-rank Approximation

- **Low-rank approximation by reduced SVD**

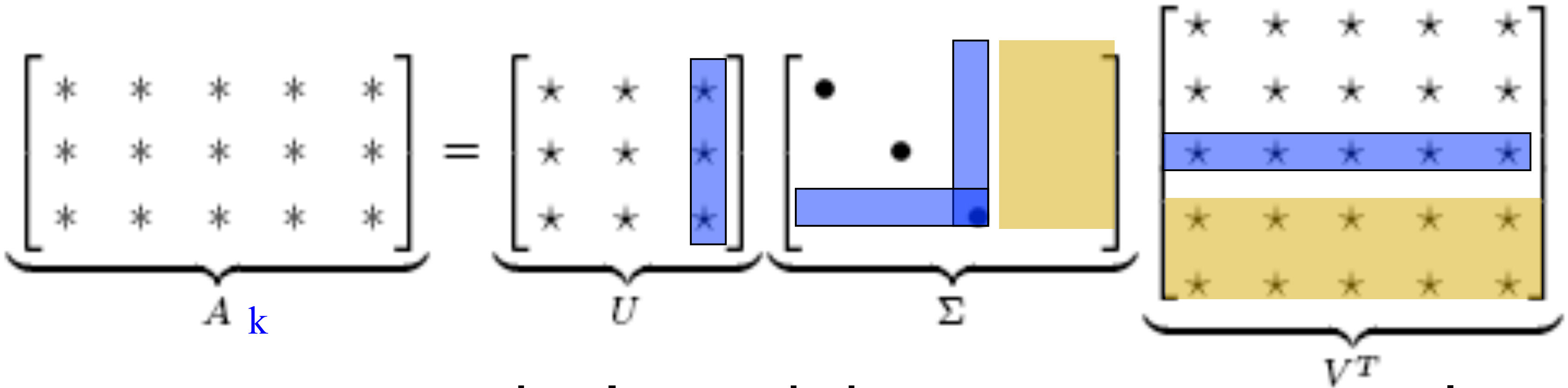
  Given matrix *A* of rank *r* ,

  Find A_k of rank *k* (*k* < *r*) such that A≈A_k

- **Reduced SVD**

$$A_k = U \;\; diag(\sigma_1,\ldots,\sigma_k,0,\ldots,0) \;\; V^T$$

(set smallest r-k signular values to zero)



We don't need the matrix parts in color

Latent Semantic Indexing (LSI) via the SVD

- Perform a low-rank approximation of document-term matrix (typical rank 100–300)
- From term-doc matrix A, we compute the approximation $A_{k.}$
- There is a row for each term and a column for each doc in A_k
- Docs live in a space of $k<<r$ dimensions

Latent Semantic Indexing

- General idea
  - Map documents (and terms) to a low-dimensional representation.
  - Design a mapping such that the low-dimensional space reflects semantic associations (latent semantic space).
  - Compute document similarity based on the inner product in this latent semantic space
  - Dimension reduction brings together related axes in VSM

LSI Example

- **A simple term-document matrix (binary)**

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

LSI Example

- **Example of $C = U \Sigma V^T$: The matrix U**

"Semantic" dimensions that capture distinct topics

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

How strongly the term is to the semantic dimension

LSI Example

- **Example of $C = U \Sigma V^T$: The matrix Σ**

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

The importance of the corresponding semantic dimension

LSI Example

- Example of $C = U \Sigma V^T$: The matrix V^T

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

Reducing the Dimension

U_2	1	2	3	4	5
ship	−0.44	−0.30	0.00	0.00	0.00
boat	−0.13	−0.33	0.00	0.00	0.00
ocean	−0.48	−0.51	0.00	0.00	0.00
wood	−0.70	0.35	0.00	0.00	0.00
tree	−0.26	0.65	0.00	0.00	0.00

Σ_2	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

V_2^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00

Original C vs. Reduced $C_2 = U\Sigma_2 V^T$

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

smoothing

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

Why New Matrix C_2 is Better?

- Similarity of d2 and d3 in the original space: 0.
- Similarity of d2 and d3 in the reduced space: $0.52 * 0.28 + 0.36 * 0.16 + 0.72 * 0.36 + 0.12 * 0.20 + -0.39 * -0.08 \approx 0.52$
- Typically, LSI increases recall and hurts precision

Applying LSI to Retrieval

- Compute SVD of term-document matrix
- Reduce the space and compute reduced document representations
- Map the query into the reduced space

$$\vec{q}_2^T = \Sigma_2^{-1} U_2^T \vec{q}^T.$$

(This follows from: $C_2 = U\Sigma_2 V^T \Rightarrow \Sigma_2^{-1} U^T C = V_2^T$)

- Compute similarity of q_2 with all reduced documents in V_2
- Output ranked list of documents as usual

(LSI is also a retrieval model)

Why LSI ?

- Each singular value tells us how important its dimension is.
- By setting less important dimensions to zero, we keep the important information, but get rid of the "details".
- These details may
  - be noise – in that case, reduced LSI is a better representation because it is less noisy
  - make things dissimilar that should be similar – again reduced LSI is a better representation because it represents similarity better.

What You Should Know

- What is Vector Space Model (a family of models)
- What is TF-IDF weighting
- What is pivoted normalization
- Relevance feedback in VSM
  - How Rocchio works
- Dimension reduction
  - Latent Semantic Indexing