DOK 324: Principles of Information Retrieval

Hacettepe University

Department of Information Management

IR Models: Boolean, Vector Space

Slides taken from Prof. Ray R. Larson, http://www.sims.berkeley.edu

Review: Central Concepts in IR

- Documents
- Queries
- Collections
- Evaluation
- Relevance

Relevance

- “Intuitively, we understand quite well what relevance means. It is a primitive ‘y’ know’ concept, as is information, for which we hardly need a definition. … if and when any productive contact [in communication] is desired, consciously or not, we involve and use this intuitive notion of relevance.”
  - Saracevic, 1975 p. 324

Relevance

- How relevant is the document
  - for this user for this information need.
- Subjective, but
- Measurable to some extent
  - How often do people agree a document is relevant to a query
- How well does it answer the question?
  - Complete answer? Partial?
  - Background Information?
  - Hints for further exploration?

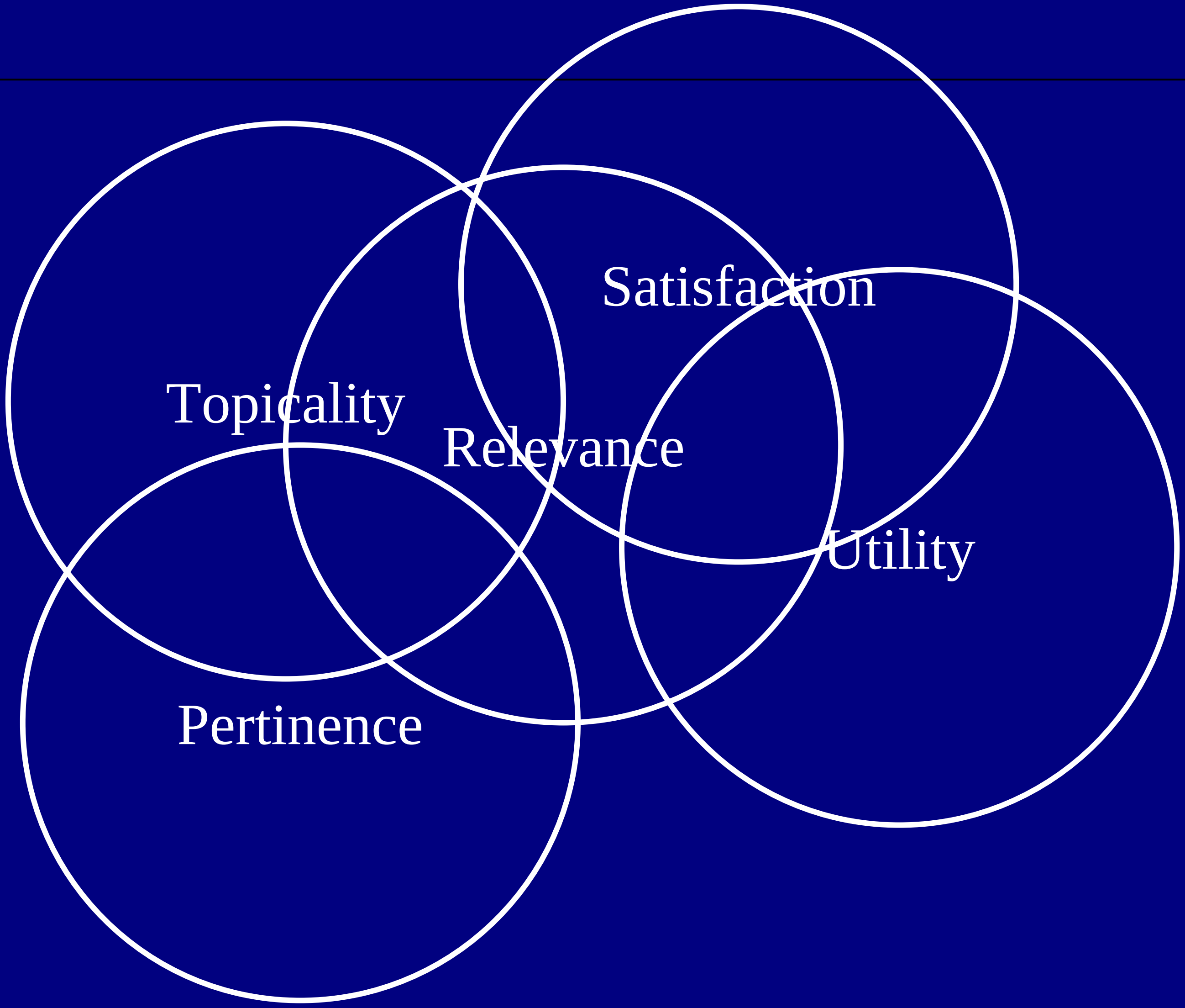
Saracevic

- Relevance is considered as a measure of effectiveness of the contact between a source and a destination in a communications process
  - – Systems view
  - – Destinations view
  - – Subject Literature view
  - – Subject Knowledge view
  - – Pertinence
  - – Pragmatic view

Froehlich

- Centrality and inadequacy of Topicality as the basis for relevance
- Suggestions for a synthesis of views

Janes' View



IR Models

- Set Theoretic Models
  - Boolean
  - Fuzzy
  - Extended Boolean
- Vector Models (Algebraic)
- Probabilistic Models (probabilistic)
- Others (e.g., neural networks)

Boolean Model for IR

- Based on Boolean Logic (Algebra of Sets).
- Fundamental principles established by George Boole in the 1850’s
- Deals with set membership and operations on sets
- Set membership in IR systems is usually based on whether (or not) a document contains a keyword (term)

Boolean Logic

$$C = A$$

$$C = \overline{A}$$

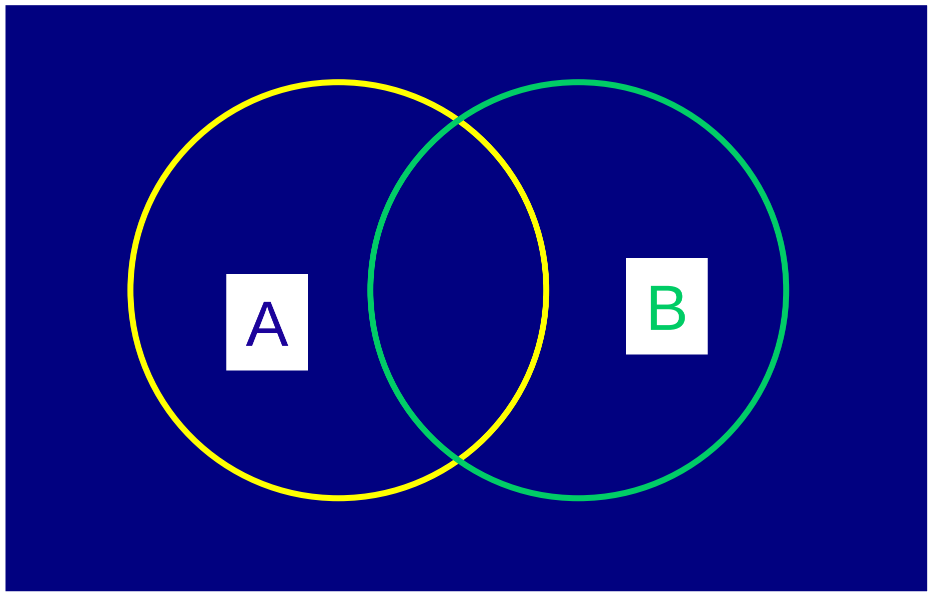
$$C = A \cap B$$

$$C = A \cup B$$

DeMorgan's Law :

$$\overline{A \cap B} = \overline{A} \cup \overline{B}$$

$$\overline{A \cup B} = \overline{A} \cap \overline{B}$$



Query Languages

- A way to express the query (formal expression of the information need)
- Types:
  - Boolean
  - Natural Language
  - Stylized Natural Language
  - Form-Based (GUI)

Simple query language: Boolean

- Terms + Connectors
  - terms
    - words
    - normalized (stemmed) words
    - phrases
    - thesaurus terms
  - connectors
    - AND
    - OR
    - NOT

Boolean Queries

- Cat
- Cat OR Dog
- Cat AND Dog
- (Cat AND Dog)
- (Cat AND Dog) OR Collar
- (Cat AND Dog) OR (Collar AND Leash)
- (Cat OR Dog) AND (Collar OR Leash)

Boolean Queries

- (Cat OR Dog) AND (Collar OR Leash)
  - *Each* of the following combinations satisfies this statement:

Cat		x		x		x	x
Dog		x		x	x	x	x
Collar	x			x	x		x
Leash		x	x			x	x

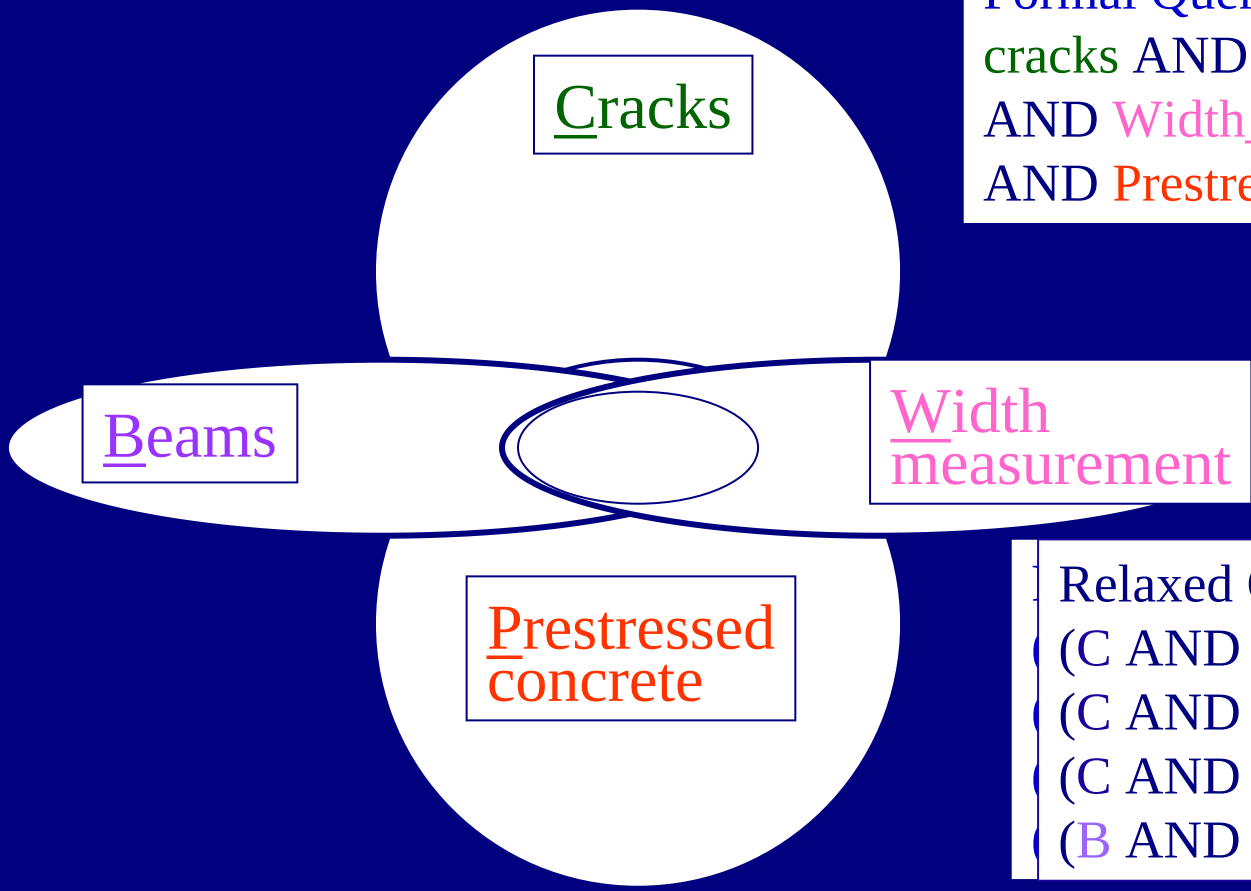
Boolean Queries

- (Cat OR Dog) AND (Collar OR Leash)
  - *None* of the following combinations work:

Cat	x		x				
Dog	x			x			
Collar		x			x		
Leash		x				x	

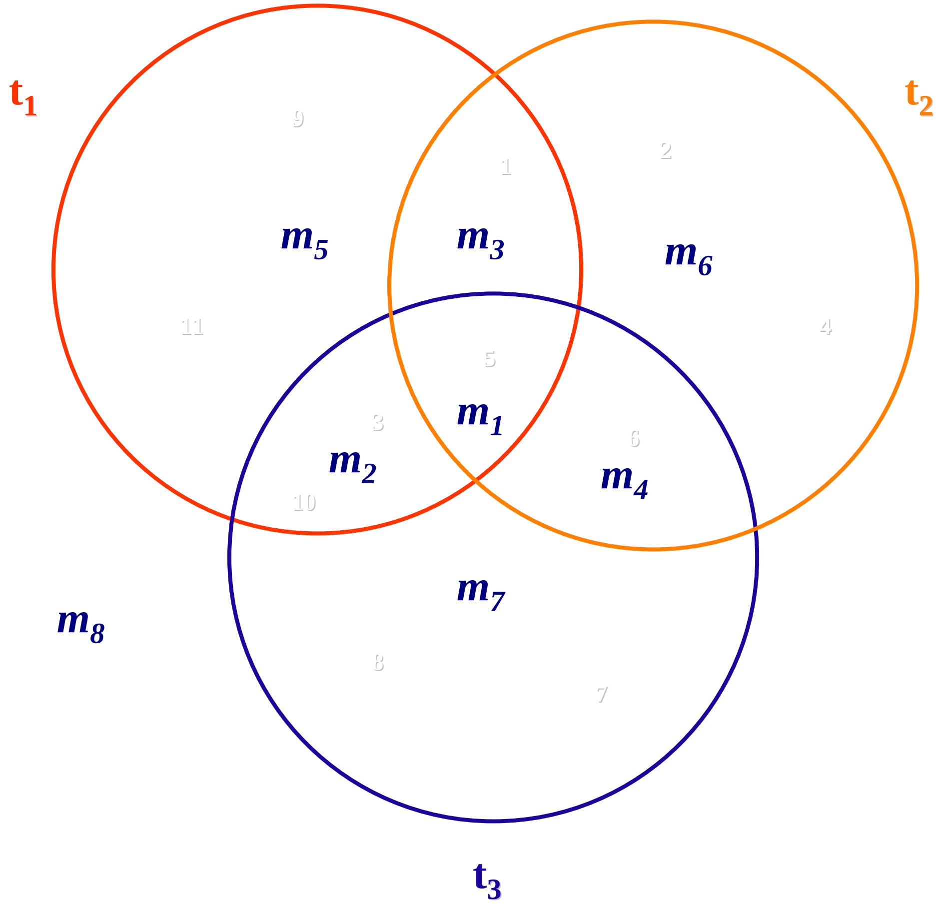
Boolean Searching

Formal Query:
cracks AND beams
AND Width_measurement
AND Prestressed_concrete



Relaxed Query:
(C AND B AND P) OR
(C AND B AND W) OR
(C AND W AND P) OR
(B AND W AND P)

Boolean Logic



m_1	$t_1\ t_2\ t_3$
m_2	$t_1\ t_2\ t_3$
m_3	$t_1\ t_2\ t_3$
m_4	$t_1\ t_2\ t_3$
m_5	$t_1\ t_2\ t_3$
m_6	$t_1\ t_2\ t_3$
m_7	$t_1\ t_2\ t_3$
m_8	$t_1\ t_2\ t_3$

Precedence Ordering

- In what order do we evaluate the components of the Boolean expression?
  - Parenthesis get done first
    - (a or b) and (c or d)
    - (a or (b and c) or d)
  - Usually start from the left and work right (in case of ties)
  - Usually (if there are no parentheses)
    - NOT before AND
    - AND before OR

Pseudo-Boolean Queries

- A new notation, from web search
  - +cat dog +collar leash
  - These are prefix operators
- Does not mean the same thing as AND/OR!
  - + means “mandatory, must be in document”
  - - means “cannot be in the document”
- Phrases:
  - “stray cat” AND “frayed collar”
  - is equivalent to
  - +“stray cat” +“frayed collar”

Result Sets

- Run a query, get a result set
- Two choices
  - Reformulate query, run on entire collection
  - Reformulate query, run on result set
- Example: Dialog query
  - (Redford AND Newman)
  - -> S1 1450 documents
  - (S1 AND Sundance)
  - ->S2 898 documents

Faceted Boolean Query

- Strategy: break query into facets (polysemous with earlier meaning of facets)
  - conjunction of disjunctions

    (a1 OR a2 OR a3)
    (b1 OR b2)
    (c1 OR c2 OR c3 OR c4)
    } AND

  - each facet expresses a topic

    ("rain forest" OR jungle OR amazon)
    (medicine OR remedy OR cure)
    (Smith OR Zhou)
    } AND

Ordering of Retrieved Documents

- Pure Boolean has no ordering
- In practice:
  - order chronologically
  - order by total number of “hits” on query terms
    - What if one term has more hits than others?
    - Is it better to one of each term or many of one term?
- Fancier methods have been investigated
  - p-norm is most famous
    - usually impractical to implement
    - usually hard for user to understand

Boolean Implementation: Inverted Files

- We have not yet seen "Vector files" in detail conceptually, an Inverted File is a vector file "inverted" so that rows become columns and columns become rows

docs	*t1*	*t2*	*t3*
D1	**1**	**0**	**1**
D2	**1**	**0**	**0**
D3	**0**	**1**	**1**
D4	**1**	**0**	**0**
D5	**1**	**1**	**1**
D6	**1**	**1**	**0**
D7	**0**	**1**	**0**
D8	**0**	**1**	**0**
D9	**0**	**0**	**1**
D10	**0**	**1**	**1**

Terms	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**…**
t1	1	1	0	1	1	1	0	
t2	0	0	1	0	1	1	1	
t3	1	0	1	0	1	0	0	

How Are Inverted Files Created

- Documents are parsed to extract words (or stems) and these are saved with the Document ID.

Doc 1

Now is the time for all good men to come to the aid of their country

Doc 2

It was a dark and stormy night in the country manor. The time was past midnight

→

Term	Doc #
now	1
is	1
the	1
time	1
for	1
all	1
good	1
men	1
to	1
come	1
to	1
the	1
aid	1
of	1
their	1
country	1
it	2
was	2
a	2
dark	2
and	2
stormy	2
night	2
in	2
the	2
country	2
manor	2
the	2
time	2
was	2
past	2
midnight	2

How Inverted Files are Created

- After all documents have been parsed the inverted file is sorted

Term	Doc #
now	1
is	1
the	1
time	1
for	1
all	1
good	1
men	1
to	1
come	1
to	1
the	1
aid	1
of	1
their	1
country	1
it	2
was	2
a	2
dark	2
and	2
stormy	2
night	2
in	2
the	2
country	2
manor	2
the	2
time	2
was	2
past	2
midnight	2

Term	Doc #
a	2
aid	1
all	1
and	2
come	1
country	1
country	2
dark	2
for	1
good	1
in	2
is	1
it	2
manor	2
men	1
midnight	2
night	2
now	1
of	1
past	2
stormy	2
the	1
the	1
the	2
the	2
their	1
time	1
time	2
to	1
to	1
was	2
was	2

How Inverted Files are Created

- Multiple term entries for a single document are merged and frequency information added

Term	Doc #
a	2
aid	1
all	1
and	2
come	1
country	1
country	2
dark	2
for	1
good	1
in	2
is	1
it	2
manor	2
men	1
midnight	2
night	2
now	1
of	1
past	2
stormy	2
the	1
the	1
the	2
the	2
their	1
time	1
time	2
to	1
to	1
was	2
was	2

Term	Doc #	Freq
a	2	1
aid	1	1
all	1	1
and	2	1
come	1	1
country	1	1
country	2	1
dark	2	1
for	1	1
good	1	1
in	2	1
is	1	1
it	2	1
manor	2	1
men	1	1
midnight	2	1
night	2	1
now	1	1
of	1	1
past	2	1
stormy	2	1
the	1	2
the	2	2
their	1	1
time	1	1
time	2	1
to	1	2
was	2	2

How Inverted Files are

- The file is commonly split into a *Dictionary* and a *Postings* file

Term	Doc #	Freq
a	2	1
aid	1	1
all	1	1
and	2	1
come	1	1
country	1	1
country	2	1
dark	2	1
for	1	1
good	1	1
in	2	1
is	1	1
it	2	1
manor	2	1
men	1	1
midnight	2	1
night	2	1
now	1	1
of	1	1
past	2	1
stormy	2	1
the	1	2
the	2	2
their	1	1
time	1	1
time	2	1
to	1	2
was	2	2

Term	N docs	Tot Freq
a	1	1
aid	1	1
all	1	1
and	1	1
come	1	1
country	2	2
dark	1	1
for	1	1
good	1	1
in	1	1
is	1	1
it	1	1
manor	1	1
men	1	1
midnight	1	1
night	1	1
now	1	1
of	1	1
past	1	1
stormy	1	1
the	2	4
their	1	1
time	2	2
to	1	2
was	1	2

Doc #	Freq
2	1
1	1
1	1
2	1
1	1
1	1
2	1
2	1
1	1
1	1
2	1
1	1
2	1
2	1
1	1
2	1
2	1
1	1
1	1
2	1
2	1
1	2
2	2
1	1
1	1
2	1
1	2
2	2

Boolean AND Algorithm

2		2		2
5		8		8
7		9		15
8		12		100
15		15		135
29		22		155
35	AND	28	=	189
100		50		195
135		68		
140		77		
155		84		
189		100		
190		120		
195		128		
198		135		
		138		
		141		
		150		
		155		
		188		
		189		
		195		

Boolean OR Algorithm

2
5
7
8
15
29
35
100
135
140
155
189
190
195
198

OR

2
8
9
12
15
22
28
50
68
77
84
100
120
128
135
138
141
150
155
188
189
195

=

2
5
7
8
9
12
15
22
28
29
35
50
68
77
84
100
120
128
135
138
140
141
150
155
188
189
190
195
198

Boolean AND NOT Algorithm

	AND NOT	=
2	2	5
5	8	7
7	9	29
8	12	35
15	15	140
29	22	190
35	28	198
100	50	
135	68	
140	77	
155	84	
189	100	
190	120	
195	128	
198	135	
	138	
	141	
	150	
	155	
	188	
	189	
	195	

Inverted files

- Permit fast search for individual terms
- Search results for each term is a list of document IDs (and optionally, frequency and/or positional information)
- These lists can be used to solve Boolean queries:
  - country: d1, d2
  - manor: d2
  - country and manor: d2

Boolean Summary

- Advantages
  - simple queries are easy to understand
  - relatively easy to implement
- Disadvantages
  - difficult to specify what is wanted, particularly in complex situations
  - too much returned, or too little
  - ordering not well determined
- Dominant IR model in commercial systems until the WWW

IR Models: Vector Space

Non-Boolean?

- Need to measure some similarity between the query and the document
  - Need to consider the characteristics of the document and the query
  - Assumption that similarity of language use between the query and the document implies similarity of topic and hence, potential relevance.

Similarity Measures

$\lvert Q \cap D \rvert$	Simple matching (coordination level match)
$2\frac{\lvert Q \cap D \rvert}{\lvert Q \rvert + \lvert D \rvert}$	Dice's Coefficient
$\frac{\lvert Q \cap D \rvert}{\lvert Q \cup D \rvert}$	Jaccard's Coefficient
$\frac{\lvert Q \cap D \rvert}{\lvert Q \rvert^{1/2} \times \lvert D \rvert^{1/2}}$	Cosine Coefficient
$\frac{\lvert Q \cap D \rvert}{\min(\lvert Q \rvert, \lvert D \rvert)}$	Overlap Coefficient

What form should these take?

- Each of the queries and documents might be considered as:
  - A set of terms (Boolean approach)
    - "index terms"
    - "words", stems, etc.
  - Some other form?

Vector Representation

(see Salton article in *Readings*)

- Documents and Queries are represented as vectors.
- Position 1 corresponds to term 1, position 2 to term 2, position t to term t
- The weight of the term is stored in each position

$$D_i = w_{d_{i1}}, w_{d_{i2}}, \ldots, w_{d_{it}}$$

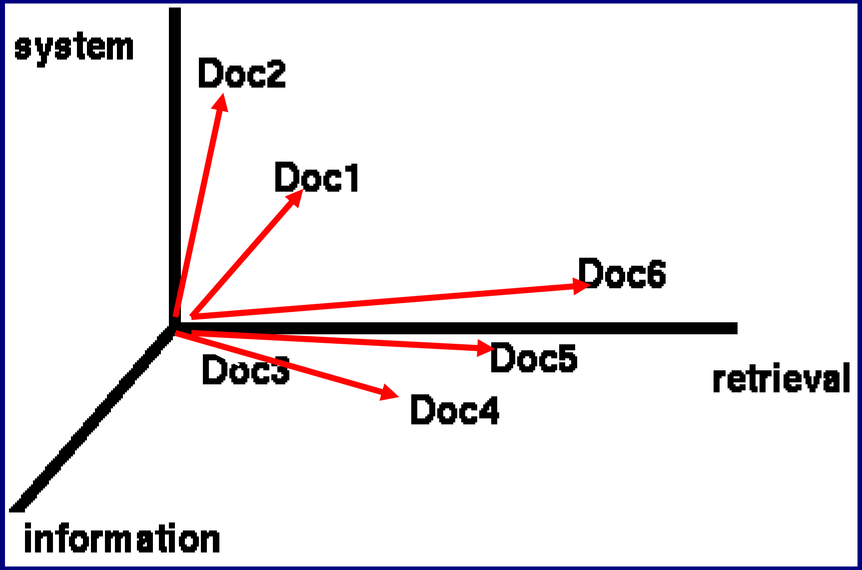
$$Q = w_{q1}, w_{q2}, \ldots, w_{qt}$$

$$w = 0 \text{ if a term is absent}$$

Vector Space Model

- Documents are represented as vectors in term space
  - Terms are usually stems or individual words, but may also be phrases, word pairs, etc.
  - Documents represented by weighted vectors of terms
- Queries represented the same as documents
- Query and Document weights for retrieval are based on length and direction of their vector
- A vector distance measure between the query and documents is used to rank retrieved documents

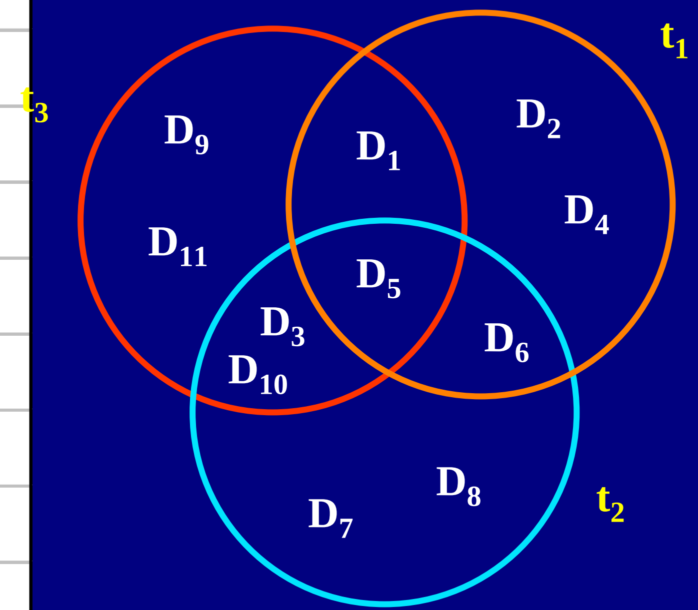
Documents in 3D Space



Assumption: Documents that are "close together" in space are similar in meaning.

Vector Space Documents and Queries

docs	*t1*	*t2*	*t3*	RSV=Q.Di
D1	1	0	1	4
D2	1	0	0	1
D3	0	1	1	5
D4	1	0	0	1
D5	1	1	1	6
D6	1	1	0	3
D7	0	1	0	2
D8	0	1	0	2
D9	0	0	1	3
D10	0	1	1	5
D11	1	0	1	4
Q	1 *q1*	2 *q2*	3 *q3*	



Document Space has High Dimensionality

- What happens beyond 2 or 3 dimensions?
- Similarity still has to do with how many tokens are shared in common.
- More terms -> harder to understand which subsets of words are shared among similar documents.
- We will look in detail at ranking methods
- One approach to handling high dimensionality:Clustering

Word Frequency vs. Resolving Power (from van Rijsbergen 79)

The most frequent words are not the most descriptive.

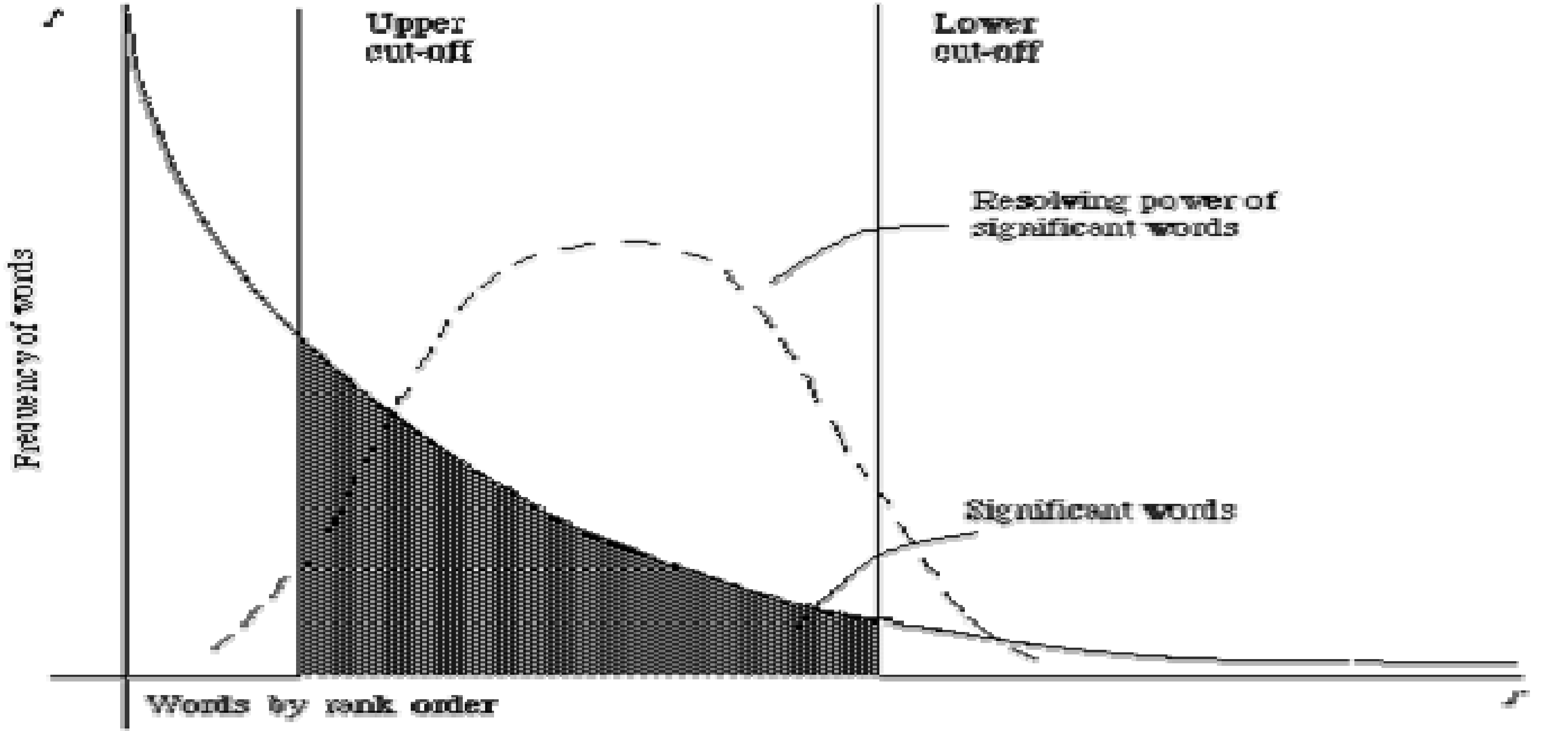


Figure 2.1. A plot of the hyperbolic curve relating f, the frequency of occurrence and r, the rank order (Adapted from Schultz[44] page 120)

tf x idf

$$w_{ik} = tf_{ik} * \log(N / n_k)$$

T_k = term k in document D_i

tf_{ik} = frequency of term T_k in document D_i

idf_k = inverse document frequency of term T_k in C

N = total number of documents in the collection C

n_k = the number of documents in C that contain T_k

$$idf_k = \log\left(\frac{N}{n_k}\right)$$

Inverse Document Frequency

- IDF provides high values for rare words and low values for common words

$$\log\left(\frac{10000}{10000}\right) = 0$$

$$\log\left(\frac{10000}{5000}\right) = 0.301$$

$$\log\left(\frac{10000}{20}\right) = 2.698$$

$$\log\left(\frac{10000}{1}\right) = 4$$

tf x idf normalization

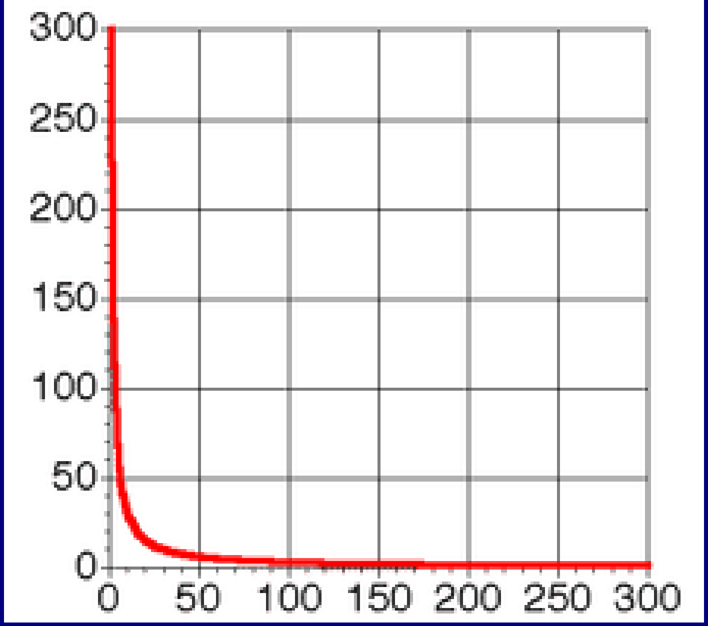
- Normalize the term weights (so longer documents are not unfairly given more weight)
  - normalize usually means force all values to fall within a certain range, usually between 0 and 1, inclusive.

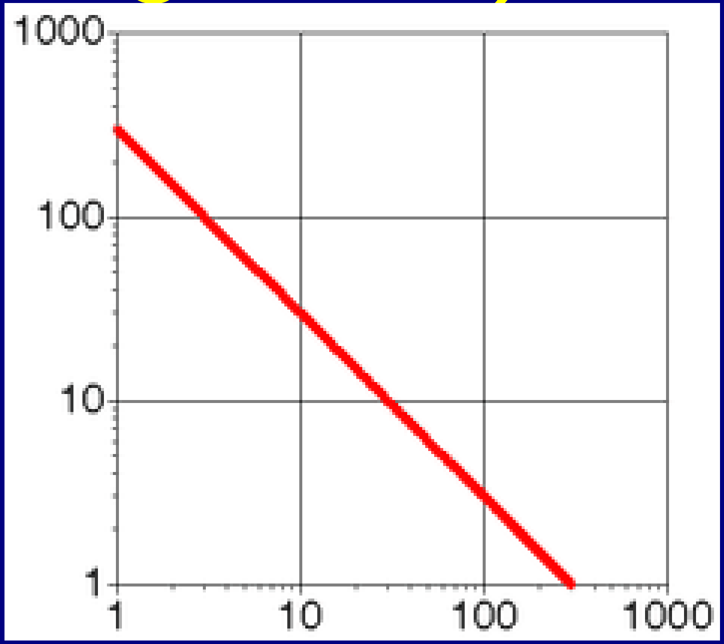
$$w_{ik} = \frac{tf_{ik} \log(N / n_k)}{\sqrt{\sum_{k=1}^{t} (tf_{ik})^2 [\log(N / n_k)]^2}}$$

Assigning Weights to Terms

- Binary Weights
- Raw term frequency
- tf x idf
  - Recall the Zipf distribution (next slide)
  - Want to weight terms highly if they are
    - frequent in relevant documents … BUT
    - infrequent in the collection as a whole
- Automatically derived thesaurus terms

Zipf Distribution (linear and log scale)





Zipf Distribution

- The product of the frequency of words (f) and their rank (r) is approximately constant
  - Rank = order of words' frequency of occurrence

$$f = C * 1/r$$

$$C \cong N/10$$

- Another way to state this is with an approximately correct rule of thumb:
  - Say the most common term occurs C times
  - The second most common occurs C/2 times
  - The third most common occurs C/3 times
  - …

Assigning Weights

- tf x idf measure:
  - term frequency (tf)
  - inverse document frequency (idf) -- a way to deal with the problems of the Zipf distribution
- Goal: assign a tf * idf weight to each term in each document

Binary Weights

- Only the presence (1) or absence (0) of a term is included in the vector

docs	*t1*	*t2*	*t3*
D1	1	0	1
D2	1	0	0
D3	0	1	1
D4	1	0	0
D5	1	1	1
D6	1	1	0
D7	0	1	0
D8	0	1	0
D9	0	0	1
D10	0	1	1
D11	1	0	1

Raw Term Weights

- The frequency of occurrence for the term in each document is included in the vector

docs	*t1*	*t2*	*t3*
D1	2	0	3
D2	1	0	0
D3	0	4	7
D4	3	0	0
D5	1	6	3
D6	3	5	0
D7	0	8	0
D8	0	10	0
D9	0	0	1
D10	0	3	5
D11	4	0	1

Vector space similarity
(use the weights to compare the documents)

Now, the similarity of two documents is :

$$sim(D_i, D_j) = \sum_{k=1}^{t} w_{ik} * w_{jk}$$

This is also called the cosine, or normalized inner product.

(Normalization was done when weighting the terms.)

Vector Space Similarity Measure

$$D_i = w_{d_{i1}}, w_{d_{i2}}, \ldots, w_{d_{it}}$$

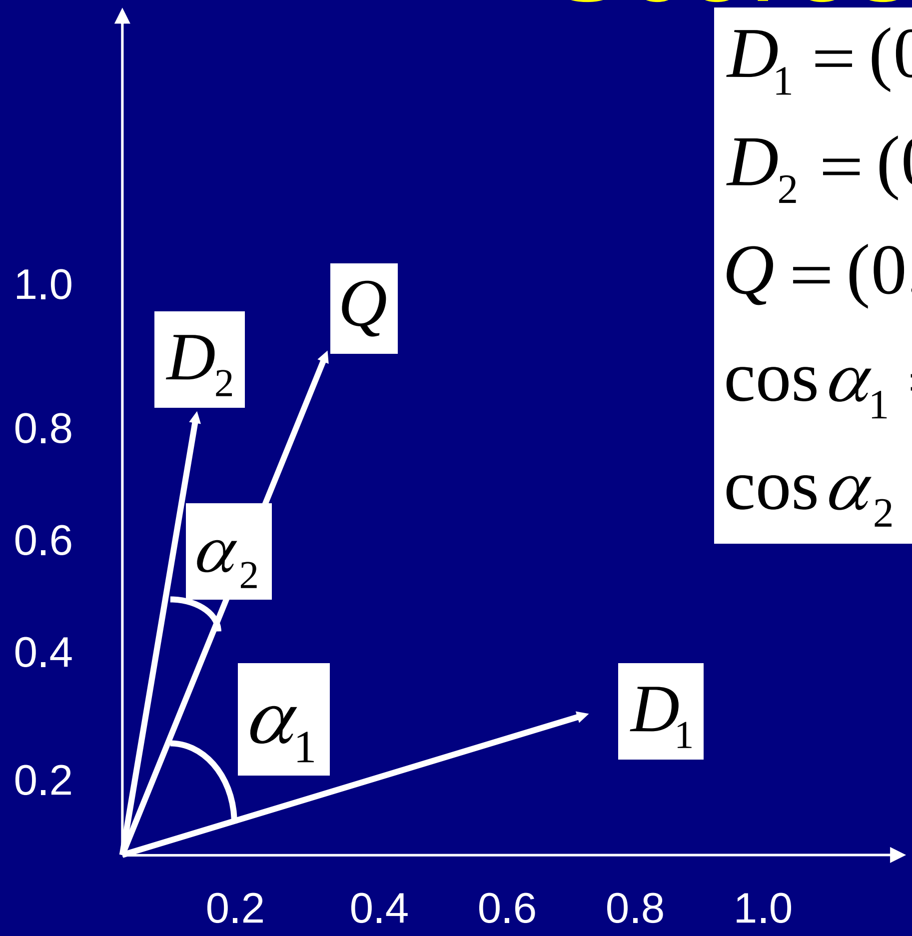
$$Q = w_{q1}, w_{q2}, \ldots, w_{qt} \qquad w = 0 \text{ if a term is absent}$$

$$\text{if term weights normalized:} \quad sim(Q, D_i) = \sum_{j=1}^{t} w_{qj} * w_{d_{ij}}$$

otherwise normalize in the similarity comparison:

$$sim(Q, D_i) = \frac{\sum_{j=1}^{t} w_{qj} * w_{d_{ij}}}{\sqrt{\sum_{j=1}^{t} (w_{qj})^2 * \sum_{j=1}^{t} (w_{d_{ij}})^2}}$$

Computing Cosine Similarity Scores



$$D_1 = (0.8, 0.3)$$

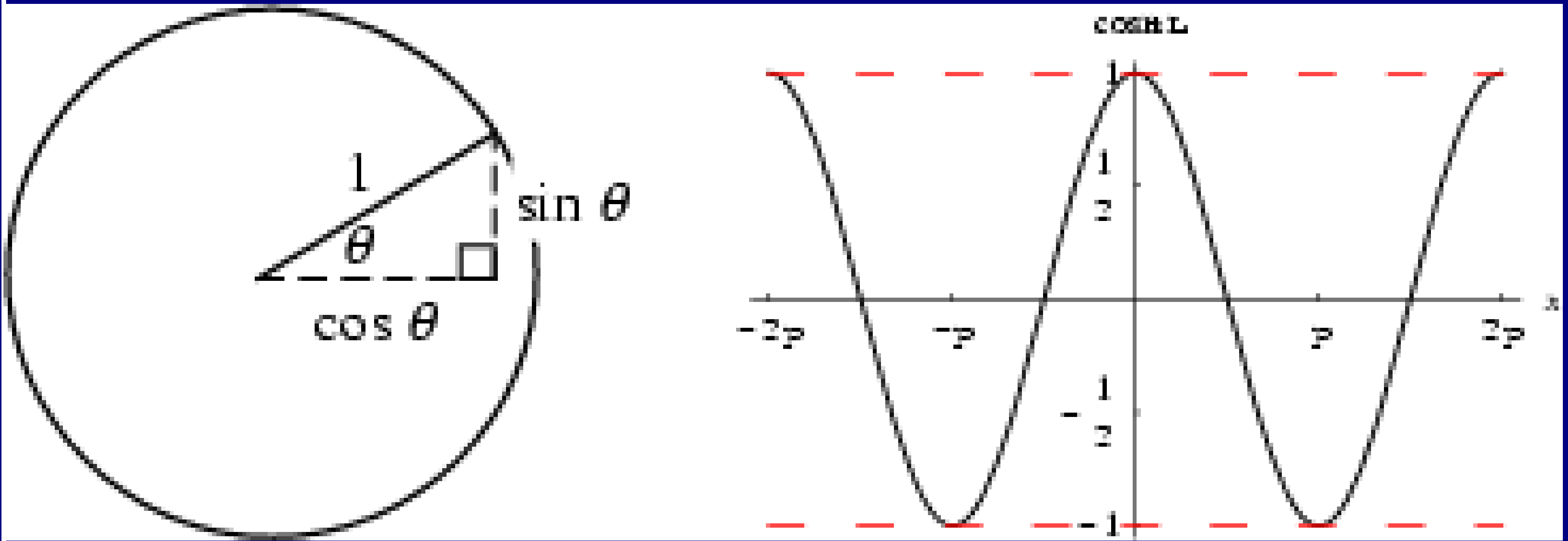
$$D_2 = (0.2, 0.7)$$

$$Q = (0.4, 0.8)$$

$$\cos \alpha_1 = 0.74$$

$$\cos \alpha_2 = 0.98$$

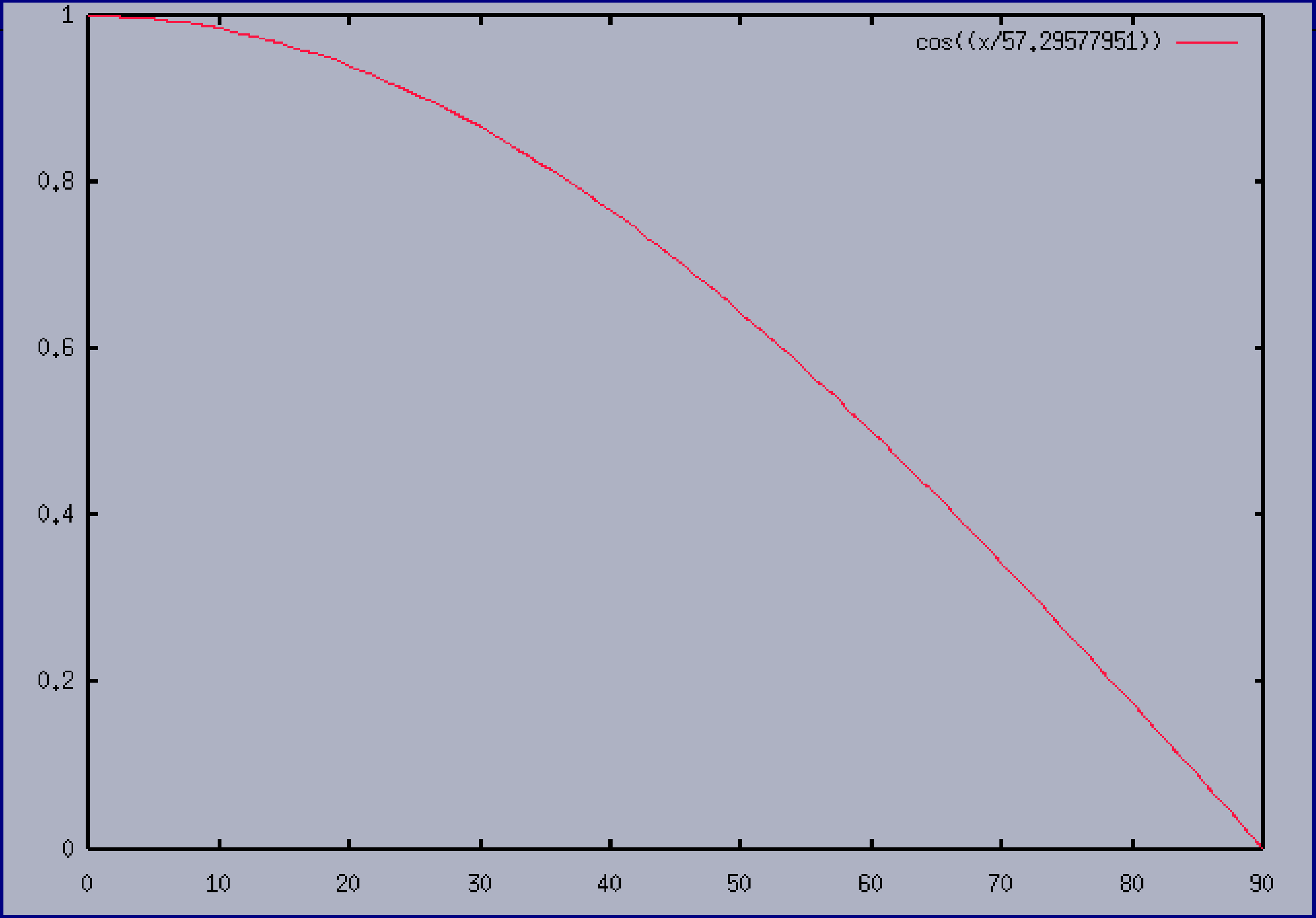
What's Cosine anyway?



One of the basic trigonometric functions encountered in trigonometry. Let theta be an angle measured counterclockwise from the x-axis along the arc of the unit circle. Then cos(theta) is the horizontal coordinate of the arc endpoint. As a result of this definition, the cosine function is periodic with period 2pi.

From http://mathworld.wolfram.com/Cosine.html

Cosine Detail (degrees)



Computing a similarity score

Say we have query vector $Q = (0.4, 0.8)$

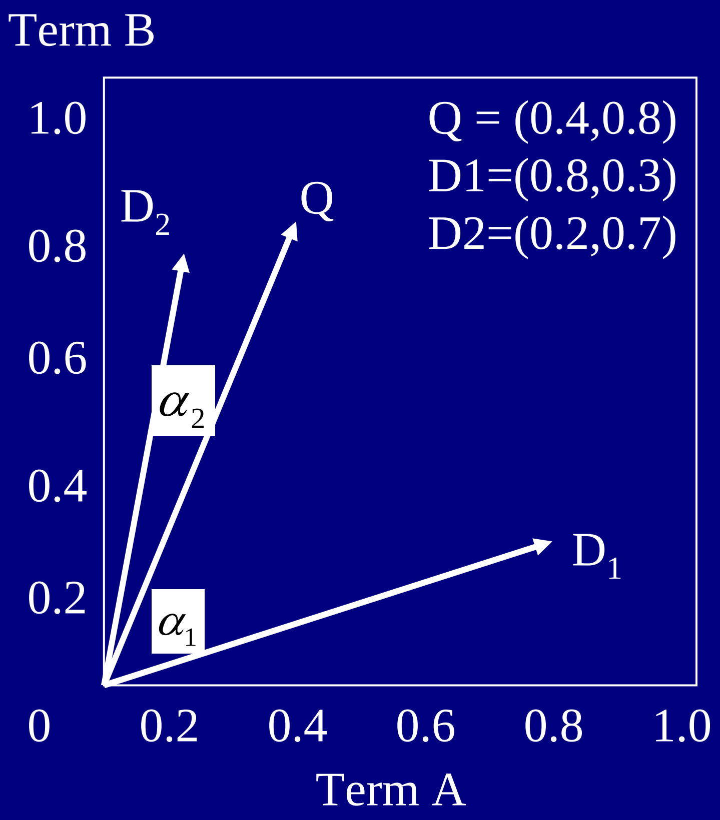
Also, document $D_2 = (0.2, 0.7)$

What does their similarity comparison yield?

$$sim(Q, D_2) = \frac{(0.4*0.2) + (0.8*0.7)}{\sqrt{[(0.4)^2 + (0.8)^2]*[(0.2)^2 + (0.7)^2]}}$$

$$= \frac{0.64}{\sqrt{0.42}} = 0.98$$

Vector Space with Term Weights and Cosine Matching



$D_i=(d_{i1},w_{di1};d_{i2},\, w_{di2};\ldots;d_{it},\, w_{dit})$

$Q=(q_{i1},w_{qi1};q_{i2},\, w_{qi2};\ldots;q_{it},\, w_{qit})$

$$sim(Q,D_i)=\frac{\sum_{j=1}^{t} w_{q_j} w_{d_{ij}}}{\sqrt{\sum_{j=1}^{t}(w_{q_j})^2 \sum_{j=1}^{t}(w_{d_{ij}})^2}}$$

$$sim(Q,D2)=\frac{(0.4\cdot 0.2)+(0.8\cdot 0.7)}{\sqrt{[(0.4)^2+(0.8)^2]\cdot[(0.2)^2+(0.7)^2]}}$$

$$=\frac{0.64}{\sqrt{0.42}}=0.98$$

$$sim(Q,D_1)=\frac{.56}{\sqrt{0.58}}=0.74$$

Weighting schemes

- We have seen something of
  - Binary
  - Raw term weights
  - TF*IDF
- There are many other possibilities
  - IDF alone
  - Normalized term frequency