Chapter 4

IR Models

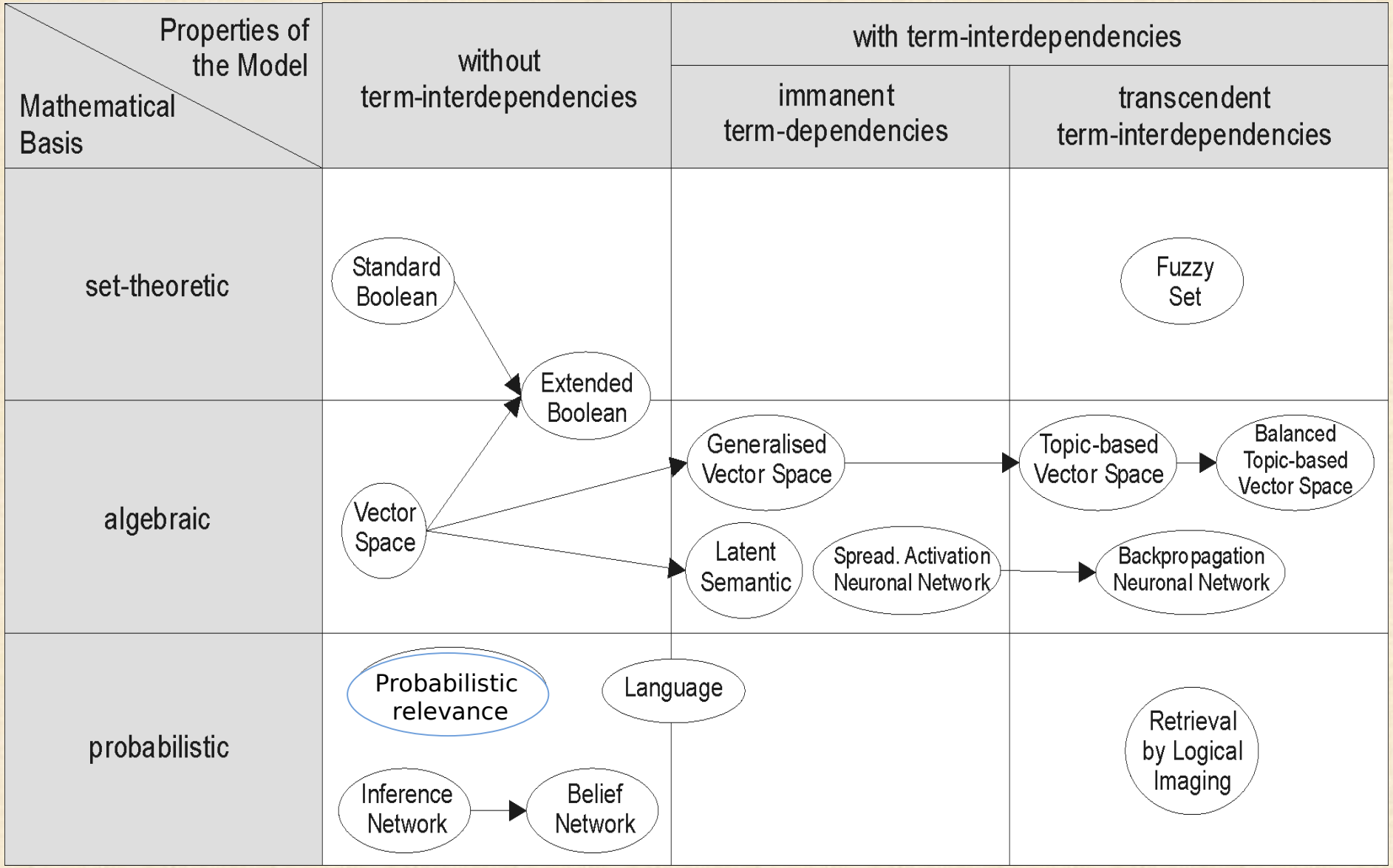
IR Models - Basic Concepts

- Word evidence:
  - ✓ IR systems usually adopt index terms to index and retrieve documents
  - ✓ Each document is represented by a set of representative keywords or index terms (called Bag of Words)
- An index term is a document word useful for remembering the document main themes
- Not all terms are equally useful for representing the document contents:
  - ✓ less frequent terms allow identifying a narrower set of documents
  - But no ordering information is attached to the Bag of Words identified from the document collection.

IR Models - Basic Concepts

- One central problem regarding IR systems is the issue of predicting the degree of relevance of documents for a given query
  - Such a decision is usually dependent on a ranking algorithm which attempts to establish a simple ordering of the documents retrieved
  - Documents appearning at the top of this ordering are considered to be more likely to be relevant
- Thus ranking algorithms are at the core of IR systems
  - The IR models determine the predictions of what is relevant and what is not, based on the notion of

Alternative IR models



How to find relevant documents for a query?

- Step 1: Map documents & queries into term-document vector space. Note that queries are considered as short document
  - Represent both documents & queries as N-dimensional vectors in a term-document matrix, which shows occurrence of terms in the document collection or query

$$d_j = (t_{1,j}, t_{2,j}, \ldots t_{N,j}); q_k = (t_{1,k}, t_{2,k}, \ldots t_{N,k})$$

	T_1	T_2	….	T_N
D_1	…	…	.. …	
	…			
D_2	…	…	.. …	
	…			
:	…	…	…..	
…;	…	…	…	..

  - Document collection is mapped to term-by-document matrix
  - View as vector in multidimensional space
    - Nearby vectors are related

How to find relevant documents for a query?

- Step 2: Queries and documents are represented as weighted vectors, w_{ij}

  - ✓ Why we need weighting techniques?
    - To know the importance of a term in describing the content of a given document?
  - ✓ There are binary weights & non-binary weighting technique. Any difference between the two?
  - ✓ What method you recommend to compute weights for *term i* in *document j* and query q; w_{ij} and w_{iq} ?
    - An entry in the matrix corresponds to the "weight" of a term in the document; zero means the term doesn't exist in the document.
    - Normalize for vector length to avoid the effect of document length

	T_1	T_2	....	T_N
D_1	w_{11}	w_{12}	...	w_{1N}
D_2	w_{21}	w_{22}	...	w_{2N}
:	:	:		

How to find relevant documents for a query?

- Step 3: Rank documents (in increasing or decreasing order) based on their closeness to the query.
  - ✓ Documents are ranked by the degree of their closeness to the query.
  - ✓ How closeness of the document to query measured?
    - ➢ It is determined by a similarity/dissimilarity score calculation
  - ✓ How many matching (similarity/dissimilarity measurements) you know? Which one is best for IR?

$$sim(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{|d_j||q|} = \frac{\sum_{i=1}^{n} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \sqrt{\sum_{i=1}^{n} w_{i,q}^2}}$$

How to evaluate Models?

- We need to investigate what procedures the IR Models follow and what techniques they use:
  - What is the weighting technique used by the IR Models for measuring importance of terms in documents?
    - Are they using binary or non-binary weight?
  - What is the matching technique used by the IR models?
    - Are they measuring similarity or dissimilarity?
  - Are they applying exact matching or partial matching in the course of finding relevant documents for a given query?
  - Are they applying best matching principle to measure the degree of relevance of documents to display in ranked-order?
    - Is there any Ranking mechanism applied before displaying relevant documents for the users?

The Boolean Model

- Boolean model is a simple model based on set theory
  - ✓ The Boolean model imposes a binary criterion for deciding relevance
- Terms are either present or absent. Thus,

$$w_{ij} \in \{0,1\}$$

- $sim(q,d_j)$ = *1 - if document satisfies the boolean query*
  0 - otherwise

- *Note that, no weights assigned in-between 0 and 1, just only values 0 or 1*

	T_1	T_2	....	T_N
D_1	w_{11}	w_{12}	...	w_{1N}
D_2	w_{21}	w_{22}	...	w_{2N}
:	:	:		

The Boolean Model: Example

Given the following three documents, Construct Term – document matrix and find the relevant documents retrieved by the Boolean model for the query "gold silver truck"

- D1: "Shipment of gold damaged in a fire"
- D2: "Delivery of silver arrived in a silver truck"
- D3: "Shipment of gold arrived in a truck"
- Query: "gold silver truck"

Table below shows document –term (t_i) matrix

	arrive	damage	deliver	fire	gold	silver	ship	truck
D1	0	1	0	1	1	0	1	0
D2	1	0	1	0	0	1	0	1
D3	1	0	0	0	1	0	1	1
query	0	0	0	0	1	1	0	1

Also find the documents relevant for the queries:

(a) gold delivery;

(b) ship gold;

(c) silver truck

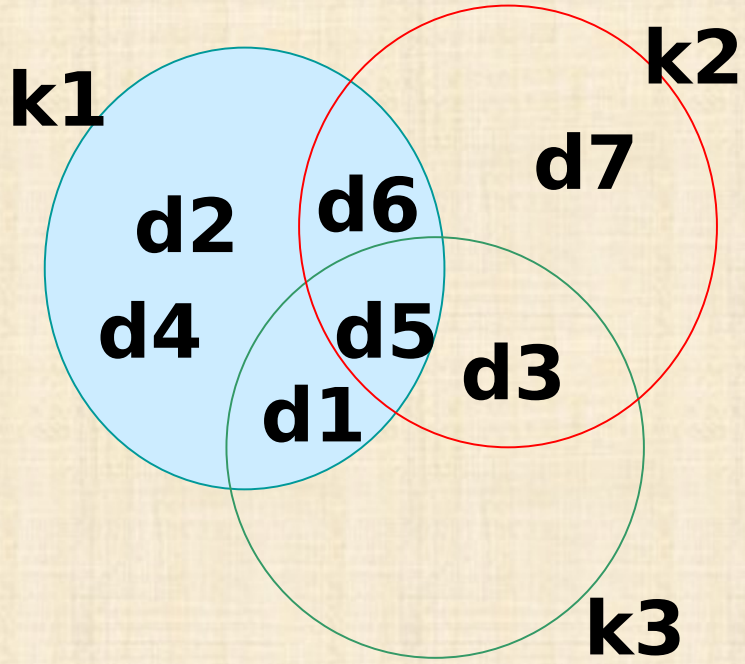
The Boolean Model: Example

- Generate the relevant documents retrieved by the Boolean model for the query :

$q = k1 \wedge (k2 \vee \neg k3)$

1. k1 = {D1, D2, D4, D5, D6}
2. k2 = {D3, D5, D6, D7}
3. k3 = {D1, D3, D5}

Query: $q = k1 \wedge (k2 \vee \neg k3)$



- Answer: {D1, D2, D4, D5, D6} ∩ ({D3, D5, D6, D7} ∪{*D2, D4, D6, D7*})

= {*D2*, *D4*, D5, *D6*}

The Boolean Model: Example

- Given the following determine documents retrieved by the Boolean model based IR system
  - Index Terms: $K1, \ldots, K8$.
  - Documents:
    1. D1 = {K1, K2, K3, K4, K5}
    2. D2 = {K1, K2, K3, K4}
    3. D3 = {K2, K4, K6, K8}
    4. D4 = {K1, K3, K5, K7}
    5. D5 = {K4, K5, K6, K7, K8}
    6. D6 = {K1, K2, K3, K4}
  - Query: K1 ^ (K2 ˅ ¬K3)
- Answer: $\{D1, D2, D4, D6\} \cap (\{D1, D2, D3, D6\} \cup \{D3, D5\})$
  $= \{D1, D2, D6\}$

Exercise

Given the following four documents with the following contents:

- D_1 = “computer information retrieval”
- D_2 = “computer retrieval”
- D_3 = “information”
- D_4 = “computer information”

- What are the relevant documents retrieved for the queries:
  - Q_1 = “information ^ retrieval”
  - Q_2 = “information ^ ¬computer”

Drawbacks of the Boolean Model

- Retrieval based on binary decision criteria with no notion of partial matching
- No ranking of the documents is provided (absence of a grading scale)
- Information need has to be translated into a Boolean expression which most users find awkward
- The Boolean queries formulated by the users are most often too simplistic
  - ✓ As a consequence, the Boolean model frequently returns either too few or too many documents in response to a user query

Vector-Space Model

- This is the most commonly used strategy for measuring relevance of documents for a given query. This is because,
  - ✓ Use of binary weights is too limiting
  - ✓ Non-binary weights provide consideration for partial matches
- These term weights are used to compute a degree of similarity between a query and each document
  - ✓ Ranked set of documents provides for better matching
- The idea behind VSM is that
  - ✓ the meaning of a document is conveyed by the words used in that document

Vector-Space Model

To find relevant documens for a given query,

- First, map documents and queries into term-document vector space.
  - ✓ Note that queries are considered as short document
- Second, in the vector space, queries and documents are represented as weighted vectors, w_{ij}
  - ✓ There are different weighting technique; the most widely used one is computing TF*IDF weight for each term
- Third, similarity measurement is used to rank documents by the closeness of their vectors to the query.
  - ✓ To measure closeness of documents to the query cosine similarity score is used by most search engines

Term-document matrix.

- A collection of *n* documents and query can be represented in the vector space model by a term-document matrix.
  - An entry in the matrix corresponds to the "weight" of a term in the document;
  - zero means the term has no significance in the document or it simply doesn't exist in the document. *Otherwise,* $w_{ij} > 0$ whenever $k_i \in d_j$

	T_1	T_2	....	T_N
D_1	w_{11}	w_{21}	...	w_{1N}
D_2	w_{21}	w_{22}	...	w_{2N}
:	:	:		

Computing weights

- How to compute weights for term i in document j and query q; w_{ij} and w_{iq} ?
- A good weight must take into account two effects:
  - Quantification of intra-document contents (similarity)
    - *tf* factor, the *term frequency* within a document
  - Quantification of inter-documents separation (dissimilarity)
    - *idf* factor, the *inverse document frequency*
- As a result of which most IR systems are using tf*idf weighting technique:

$$w_{ij} = tf(i,j) * idf(i)$$

Computing weights

- Let,
  - ✓ N be the total number of documents in the collection
  - ✓ n_i be the number of documents which contain t_i
  - ✓ freq(i,j) raw frequency of t_i within d_j
- A normalized tf(i,j) factor is given by

  $tf(i,j) = freq(i,j) / max(freq(k,j))$
  - ✓ where the maximum is computed over all terms which occur within the document d_j
- The idf factor is computed as

  $idf(i) = \log (N/n_i)$
  - ✓ The log is used to make the values of tf and idf comparable. It can also be interpreted as the amount of information associated with the term t_i.

Computing weights

- The vector space model with TF*IDF weights is a good ranking strategy with general collections
- For index terms a normalized TF*IDF weight is given by:

$$w_{ij} = \frac{freq(i,j)}{\max(freq(k,j))} * \log(N/n_i)$$

- Users query is typically treated as a short document and also TF-IDF weighted.
  - ✓ For the query term weights, a suggestion is

$$w_{iq} = \left\{ 0.5 + [0.5 * \frac{freq(i,q)}{\max(freq(k,q))}] \right\} * \log(N/n_i)$$

- The vector space model is usually as good as the known ranking alternatives. It is also simple and fast to compute.

Example: Computing weights

- A collection includes 10,000 documents
  - ✓ The term t_A appears 20 times in a particular document j
  - ✓ The maximum appearance of term t_k in document j is 50 times
  - ✓ The term t_A appears in 2,000 of the document collections.

- Compute TF*IDF weight of term A?
  - ✓ $tf(A,j)$ = freq(A,j) / max(freq(k,j)) = 20/50 = 0.4

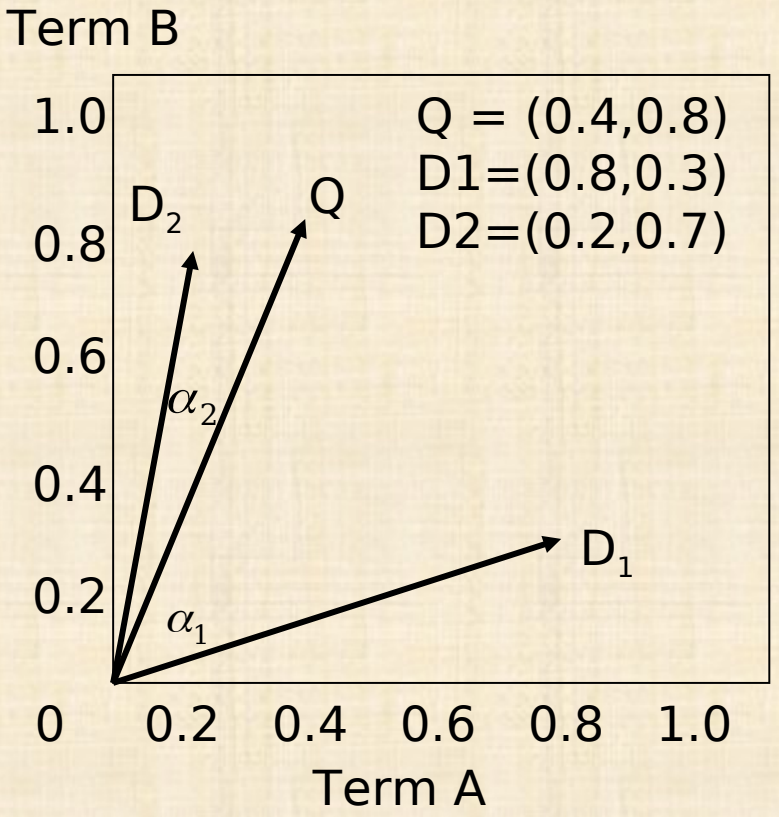
Similarity Measure

- A similarity measure is a function that computes the *degree of similarity/dissimilarity* between document j and users query.

$$sim(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{|d_j||q|} = \frac{\sum_{i=1}^{n} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \sqrt{\sum_{i=1}^{n} w_{i,q}^2}}$$

- Using a similarity score between the query and each document:
  - It is possible to apply best matching such that documents are ranked for retrieval in the order of presumed relevance.
  - It is possible to enforce a certain threshold so that we can control the size of the retrieved set of documents.

Vector Space with Term Weights and Cosine Similarity Measure



$D_i=(d_{1i},w_{1di};d_{2i},\ w_{2di};\ldots;d_{ti},\ w_{tdi})$

$Q=(q_{1i},w_{1qi};q_{2i},w_{2qi};\ldots;q_{ti},w_{tqi})$

$$sim(Q,D_i)=\frac{\sum_{j=1}^{t} w_{jq}\cdot w_{jd_i}}{\sqrt{\sum_{j=1}^{t}(w_{jq})^2\sum_{j=1}^{t}(w_{jd_i})^2}}$$

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

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

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

Example Vector-Space Model

- Suppose user query for: Q = “gold silver truck”. The database collection consists of three documents with the following content.

  D1: “Shipment of gold damaged in a fire”
  D2: “Delivery of silver arrived in a silver truck”
  D3: “Shipment of gold arrived in a truck”

- Show retrieval results in ranked order?
  1.Assume that full text terms are used during indexing, without removing common terms, stop words, & also no terms are stemmed.
  2.Assume that content-bearing terms are selected during indexing
  3.Also compare your result with or without

Example VSM: Weighting T

Terms	Q	Counts TF			DF	IDF	W_i = TF*IDF			
		D1	D2	D3			Q	D1	D2	D3
arrive	0	0	1	1	2	0.176	0	0	0.176	0.176
damage	0	1	0	0	1	0.477	0	0.477	0	0
deliver	0	0	1	0	1	0.477	0	0	0.477	0
fire	0	1	0	0	1	0.477	0	0.477	0	0
gold	1	1	0	1	2	0.176	0.176	0.176	0	0.176
silver	1	0	2	0	1	0.477	0.477	0	0.954	0
ship	0	1	0	1	2	0.176	0	0.176	0	0.176
truck	1	0	1	1	2	0.176	0.176	0	0.176	0.176

Example VSM: Weighting Terms

Terms	Q	D1	D2	D3
arrive	0	0	0.176	0.176
damage	0	0.477	0	0
deliver	0	0	0.477	0
fire	0	0.477	0	0
gold	0.176	0.176	0	0.176
silver	0.477	0	0.954	0
ship	0	0.176	0	0.176
truck	0.176	0	0.176	0.176

Example VSM: similarity Measure

- *Compute similarity using cosine* *Sim(q,d1)*
- *First, for each document and query, compute all vector lengths (zero terms ignored)*

$|d_1| = \sqrt{0.477^2 + 0.477^2 + 0.176^2 + 0.176^2} = \sqrt{0.517} = 0.719$

$|d_2| = \sqrt{0.176^2 + 0.477^2 + 0.954^2 + 0.176^2} = \sqrt{1.199} = 1.095$

$|d_3| = \sqrt{0.176^2 + 0.176^2 + 0.176^2 + 0.176^2} = \sqrt{0.124} = 0.352$

$|q| = \sqrt{0.176^2 + 0.471^2 + 0.176^2} = \sqrt{0.289} = 0.538$

- *Next, compute dot products (zero products ignored)*

*Q*d1* = 0.176*0.176 = 0.0310

*Q*d2* = (0.954*0.477) + (0.176 *0.176) = 0.4862

Q*d3 = (0.176*0.176) + (0.176*0.176) = 0.0620

Example VSM: Ranking

Now, compute similarity score

$Sim(q,d_1) = (0.0310) / (0.538*0.719) = 0.0801$

$Sim(q,d_2) = (0.4862) / (0.538*1.095) = 0.8246$

$Sim(q,d_3) = (0.0620) / (0.538*0.352) = 0.3271$

Finally, we sort and rank documents in descending order according to the similarity scores

Rank 1: Doc 2 = 0.8246
Rank 2: Doc 3 = 0.3271
Rank 3: Doc 1 = 0.0801

- **Exercise**: using normalized TF, rank documents using cosine similarity measure? Hint: Normalize TF of term i in doc j using max frequency of a term k in document j

Vector-Space Model

- Advantages:
  - Term-weighting improves quality of the answer set since it helps to display relevant documents in ranked order
  - Partial matching allows retrieval of documents that approximate the query conditions
  - Cosine ranking formula sorts documents according to degree of similarity to the query

- Disadvantages:
  - Assumes independence of index terms. It doesn’t relate one term with another term
  - Computationally expensive since it measures the similarity between each document and the query

Exercise 1

Suppose the database collection consists of the following documents.

c1: Human machine interface for Lab ABC computer applications
c2: A survey of user opinion of computer system response time
c3: The EPS user interface management system
c4: System and human system engineering testing of EPS
c5: Relation of user-perceived response time to error measure
M1: The generation of random, binary, unordered trees
M2: The intersection graph of paths in trees
M3: Graph minors: Widths of trees and well-quasi-ordering
m4: Graph minors: A survey

Query:

Find documents relevant to "human computer interaction"

Exercise 2

- Consider these documents:

Doc 1 breakthrough drug for schizophrenia
Doc 2 new schizophrenia drug
Doc 3 new approach for treatment of schizophrenia
Doc 4 new hopes for schizophrenia patients

  - Draw the term-document incidence matrix for this document collection.
  - Draw the inverted index representation for this collection.

- For the document collection shown above, what are the returned results for the queries:
  - schizophrenia AND drug
  - for AND NOT(drug OR approach)

Probabilistic Model

- IR is an uncertain process
  - Mapping Information need to Query is not perfect
  - Mapping Documents to index terms is a logical representation
  - Query terms and index terms mostly mismatch
- This situation leads to several statistical approaches: probability theory, fuzzy logic, theory of evidence, language modeling, etc.
- Probabilistic retrieval model is rigorous formal model that attempts to predict the probability that a given document will be relevant to a given query; i.e. Prob(R|(q,d_i))
  - Use probability to estimate the “odds” of relevance of a query to a document.
  - It relies on accurate estimates of probabilities

Probability Ranking Principle

- The relevance of a given document for users query can be determined by the probability score
  - High probability (*prob(rel | d_i q):* means more likely for users to get relevant information by reading document d_i.
- A Probabilistic retrieval model follows Probability ranking principle
  - You have a collection of Documents
    - A set of relevant documents needs to be returned for queries issued by users
    - Intuitively, want the "best" document to be first, second best - second, etc…
  - According to probability ranking principle, documents are ranked in decreasing order of probability of relevance to users information need
  - If p($R|D$) > p($NR|D$) then D is relevant,
    - otherwise D is not relevant

Terms Existence in Relevant Document

N=the total number of documents in the collection

n= the total number of documents that contain term t_i

R=the total number of relevant documents retrieved

r=the total number of relevant documents retrieved that contain term t_i

For term t_i	Document Relevance		
	No of relevant docs	No of non-relevant docs	Total
No of docs including term t_i	r	n-r	n
No of docs excluding term t_i	R-r	N-R-(n-r)	N-n
Total	R	N-R	N

$$w_i = \log \frac{(r+0.5)(N-n-R+r+0.5)}{(n-r+0.5)(R-r+0.5)}$$

Computing term probabilities

- Three cases: Relevance of documents for a given query may be known, partially known or unknown
  - Initially, there are no retrieved documents
  - R is completely unknown
  - Assume $P(t_i|R)$ is constant (usually 0.5)
  - Assume $P(t_i|NR)$ approximated by distribution of t_i across collection – IDF

$$P(t|\overline{R}) = \log \frac{N - n + 0.5}{n + 0.5}$$

- This can be used to compute an initial rank using IDF as the basic term weight

Probabilistic Model Example

d	Document vectors <$tf_{d,t}$>									
	cold	day	eat	hot	lot	nine	old	pea	pizza	pot
1	1			1				1	1	
2								1	1	1
3		1				1	1			
4	1			1						1
5								1	1	
6			1		1					
w_t	0.26	0.56	0.56	0.26	0.56	0.56	0.56	0.0	0.0	0.26

- q_1 = eat =>0.56
- q_2 = eat pizza=> 0.56+0.0
- q_4 = eat hot pizza=>0.56+0.26+0.0

$$P(t|\overline{R}) = \log \frac{N - n + 0.5}{n + 0.5}$$

Improving the Ranking

- Now, suppose
  - we have shown the initial ranking to the user
  - the user has labeled some of the documents as relevant ("relevance feedback")
- We now have
  - N documents in collection, R are known relevant documents
  - n_i documents containing t_i, out of which r_i are relevant

Probabilistic Retrieval Example

- D1: “Cost of paper is up.” (relevant)
- D2: “Cost of jellybeans is up.” (not relevant)
- D3: “Salaries of CEO’s are up.” (not relevant)
- D4: “Paper: CEO’s labor cost up.” (????)

Q. Term	Relevant	Not relevant	Evidence
paper	1	0	for (strong)
CEO	0	1/2	against
labor	0	0	none
cost	1	1/2	for (weak)
up	1	1	none

Pros & Cons Probabilistic model

- Probabilistic model uses probability theory to model the uncertainty in the retrieval process
  - Assumptions are made explicit
  - Term weight without relevance information is IDF
- Relevance feedback can improve the ranking by giving better term probability estimates
- Advantages of probabilistic model over vector-space
  - Strong theoretical basis
  - Since the base is probability theory, it is very well understood
  - Easy to extend
- Disadvantages
  - Models are often complicated
  - No term frequency weighting
- Which is better: vector-space or probabilistic?
  - Both are approximately as good as each other
  - Depends on collection, query, and other factors