

2. Information Retrieval Models

Growth of textual information



Literature



Email



WWW

How can we help manage and exploit all the information?



Desktop



News



Intranet



Blog

What is Information Retrieval (IR)?

- Narrow-sense:
 - IR= Search Engine Technologies (IR=Google, library info system)
 - IR= Text matching/classification
- Broad-sense: IR = Text Information Management:
 - General problem: how to manage text information?
 - How to find useful information? (retrieval)
 - **Example: Google**
 - How to organize information? (text classification)
 - **Example: Automatically assign emails to different folders**
 - How to discover knowledge from text? (text mining)
 - **Example: Discover correlation of events**

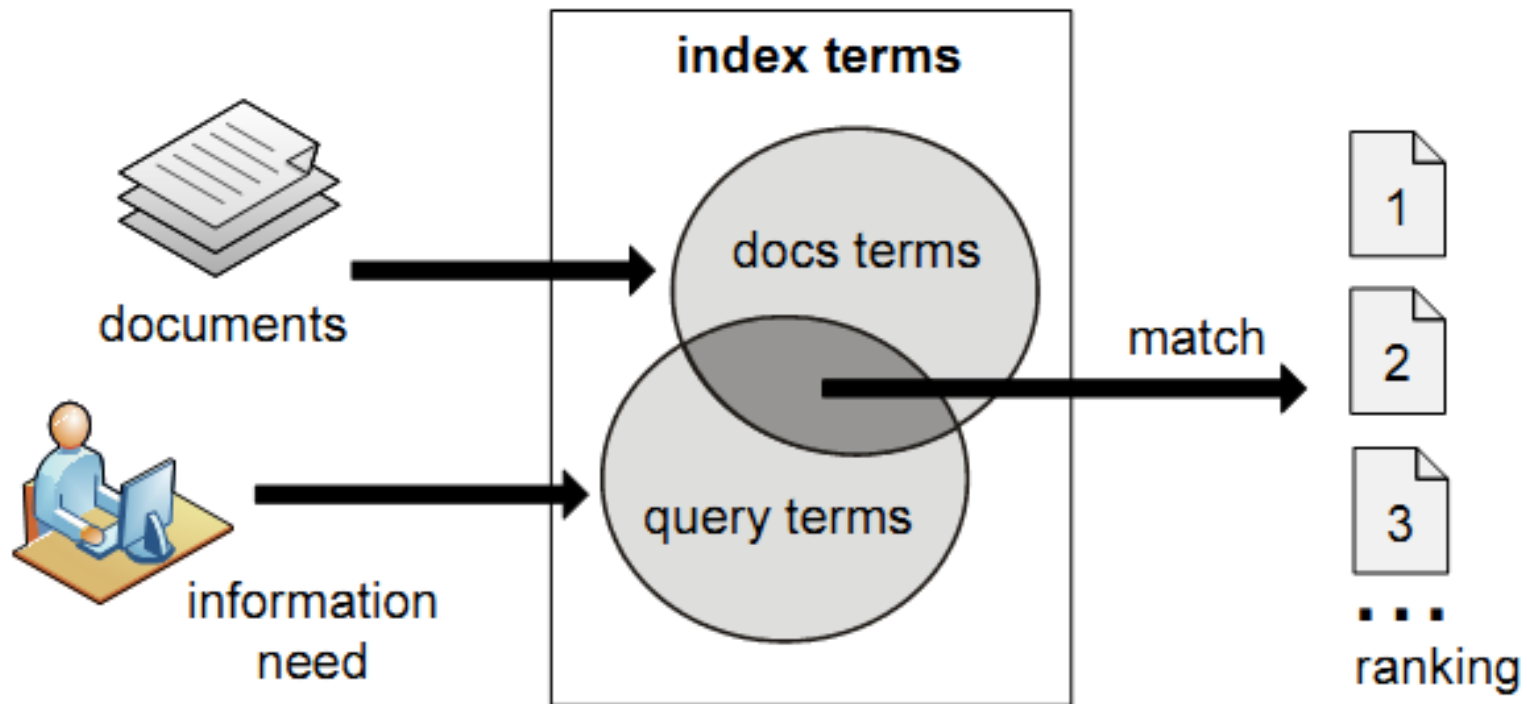
First, nomenclature...

- Information retrieval (IR)
 - Focus on textual information (= text/document retrieval)
 - Other possibilities include image, video, music, ...
- What do we search?
 - Generically, “collections”
 - Less-frequently used, “corpora”
- What do we find?
 - Generically, “documents”
 - Even though we may be referring to web pages, PDFs, PowerPoint slides, paragraphs, etc.

Documents

- Let $E_1, \dots, E_j, \dots, E_m$ denote entities in general. They can be:
 - Texts (books, journal articles, newspaper articles, papers, lecture notes, abstracts, titles, etc.),
 - Images (photographs, pictures, drawings, etc.),
 - Sounds (pieces of music, songs, speeches, etc.),
 - Multimedia (a collection of texts, images, and sounds),
 - A collection of Web pages,
 - And so on.

Information Retrieval Process



What's a word?

Tax payers will have to link their PAN with Aadhaar by the stipulated deadline, which is this month-end, as the Supreme Court verdict on privacy has no bearing on the requirement, Unique Identification Authority of India (UIDAI) CEO Ajay Bhushan Pandey said today.

भारत सरकार ने आर्थिक सर्वेक्षण में वित्तीय वर्ष 2005-06 में सात फ्रीसदी विकास दर हासिल करने का आकलन किया है और कर सुधार पर ज़ोर दिया है

**天主教教宗若望保祿二世因感冒再度住進醫院。
這是他今年第二度因同樣的病因住院。**

وقال مارك ريجيف - الناطق باسم
الخارجية الإسرائيلية - إن شارون قبل
الدعوة وسيقوم للمرة الأولى بزيارة
تونس، التي كانت لفترة طويلة المقر
الرسمي لمنظمة التحرير الفلسطينية بعد خروجها من لبنان عام 1982

日米連合で台頭中国に対処...アーミテージ前副長官提言

**조재영 기자= 서울시는 25일 이명박 시장이 "행정중심복합도시" 건설안
에 대해 "군대라도 동원해 막고싶은 심정"이라고 말했다는 일부 언론의
보도를 부인했다.**

How do we represent text?

- Computers don't “understand” anything!
- “Bag of words”
 - Treat all the words in a document as index terms
 - Assign a “weight” to each term based on “importance” (or, in simplest case, presence/absence of word)
 - Disregard order, structure, meaning, etc. of the words
 - Simple, yet effective!
- Assumptions
 - Term occurrence is independent
 - Document relevance is independent
 - “Words” are well-defined

Sample Document

McDonald's slims down spuds

Fast-food chain to reduce certain types of fat in its french fries with new cooking oil.

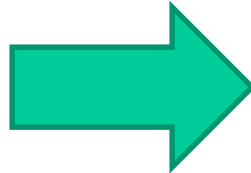
NEW YORK (CNN/Money) - McDonald's Corp. is cutting the amount of "bad" fat in its french fries nearly in half, the fast-food chain said Tuesday as it moves to make all its fried menu items healthier.

But does that mean the popular shoestring fries won't taste the same? The company says no. "It's a win-win for our customers because they are getting the same great french-fry taste along with an even healthier nutrition profile," said Mike Roberts, president of McDonald's USA.

But others are not so sure. McDonald's will not specifically discuss the kind of oil it plans to use, but at least one nutrition expert says playing with the formula could mean a different taste.

Shares of Oak Brook, Ill.-based McDonald's (MCD: down \$0.54 to \$23.22, Research, Estimates) were lower Tuesday afternoon. It was unclear Tuesday whether competitors Burger King and Wendy's International (WEN: down \$0.80 to \$34.91, Research, Estimates) would follow suit. Neither company could immediately be reached for comment.

...



"Bag of Words"

14 × McDonalds

12 × fat

11 × fries

8 × new

7 × french

6 × company, said, nutrition

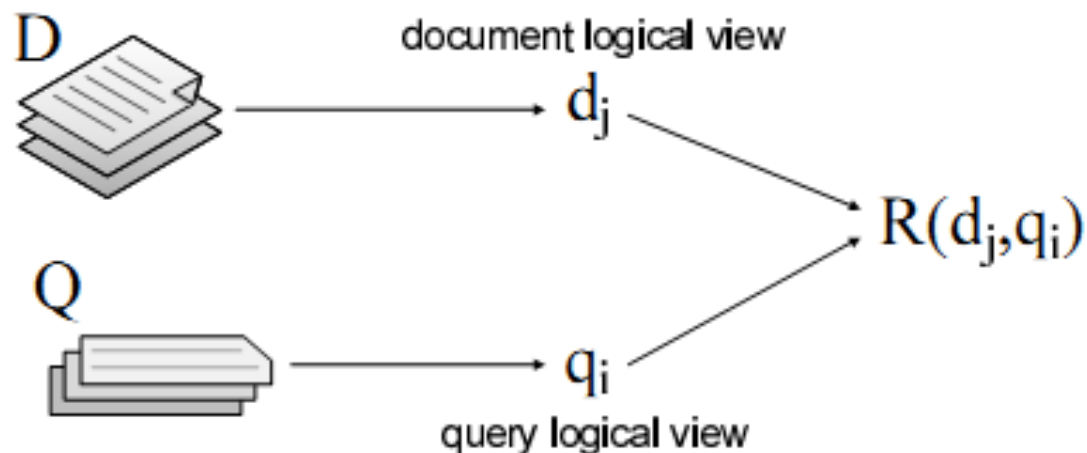
5 × food, oil, percent, reduce,
taste, Tuesday

...

IR Models

An **IR model** is a quadruple $[D, Q, \mathcal{F}, R(q_i, d_j)]$ where

1. D is a set of logical views for the documents in the collection
2. Q is a set of logical views for the user queries
3. \mathcal{F} is a framework for modeling documents and queries
4. $R(q_i, d_j)$ is a ranking function

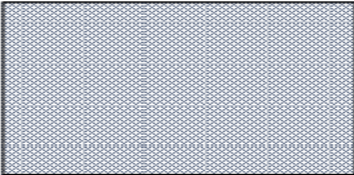
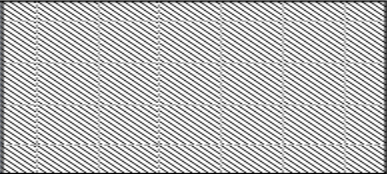
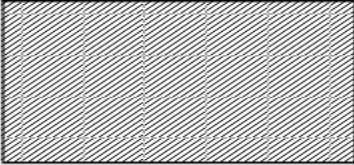



Formalizing IR Tasks

- Vocabulary: $V = \{w_1, w_2, \dots, w_T\}$ of a language
- Query: $q = q_1, q_2, \dots, q_m$ where $q_i \in V$.
- Document: $d_i = d_{i1}, d_{i2}, \dots, d_{im_i}$ where $d_{ij} \in V$.
- Collection: $C = \{d_1, d_2, \dots, d_N\}$
- Relevant document set: $R(q) \subseteq C$: Generally unknown and user-dependent
- Query provides a “hint” on which documents should be in $R(q)$

- IR: find the approximate relevant document set $R'(q)$

Precision & Recall

	Relevant	Irrelevant
Retrieved		
Non-retrieved		

$$\text{Precision} = \frac{\text{Relevant \& Retrieved}}{\text{Retrieved}}$$

$$\text{Recall} = \frac{\text{Relevant \& Retrieved}}{\text{Relevant}}$$

- **Precision** is the percentage of relevant items in the returned set
- **Recall** is the percentage of all relevant documents in the collection that is in the returned set.

Evaluating Retrieval Performance

Total relevant docs = 8

1.	d1	yes
2.	d2	yes
3.	d3	no
4.	d4	yes
5.	d5	no
6.	d6	no
7.	d7	no
8.	d8	no
9.	d9	no
10.	d10	yes

$$\text{Precision} = \frac{\text{Relevant\&Retrieved}}{\text{Retrieved}} = 4/10 = 0.4$$

$$\text{Recall} = \frac{\text{Relevant\&Retrieved}}{\text{Relevant}} = 4/8 = 0.5$$

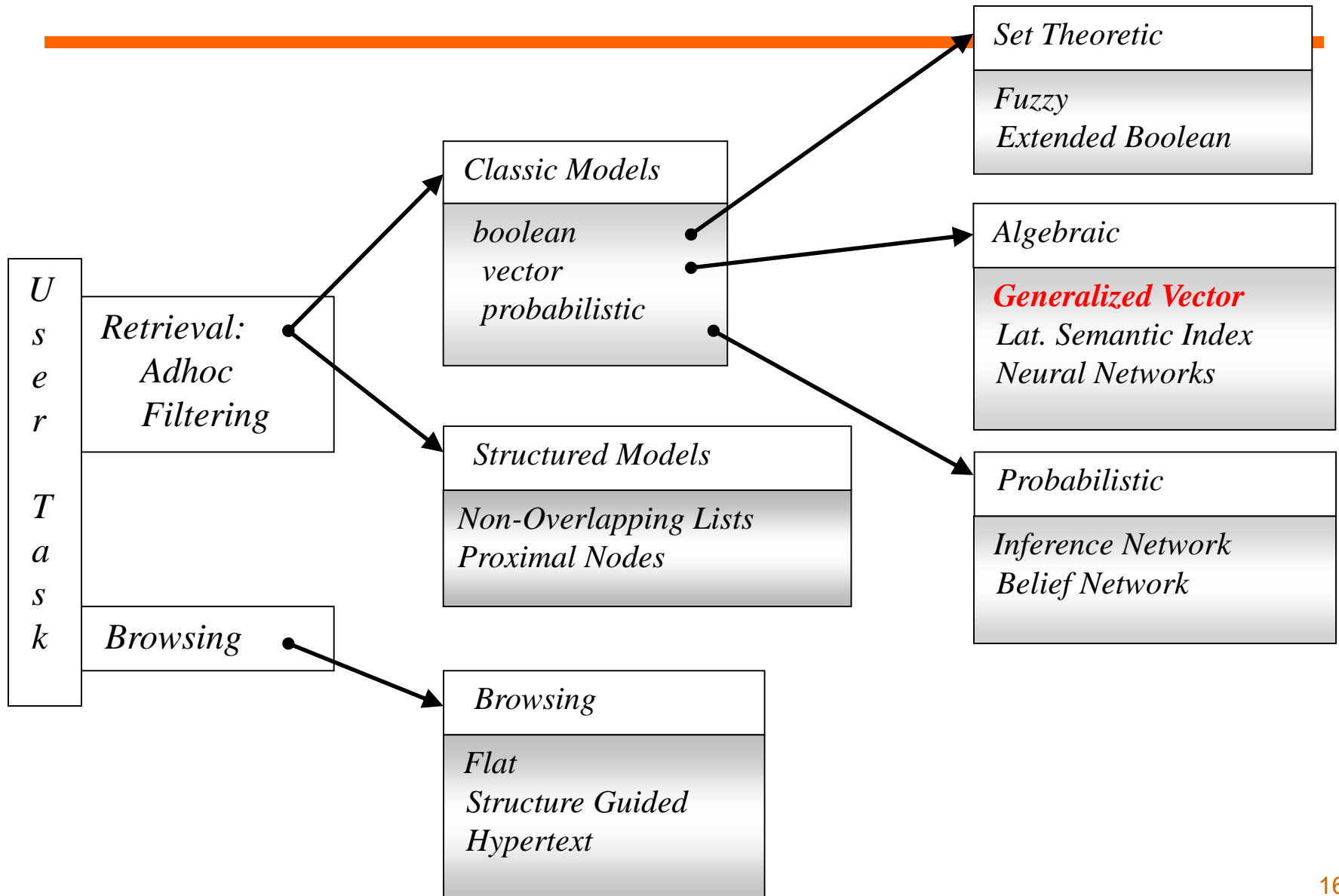
Searching

- Given a query, score documents efficiently
- The basic question:
 - Given a query, how do we know if document A is more relevant than B?
 - If document A uses more query words than document B
 - Word usage in document A is more similar to that in query
 -
- We should find a way to compute relevance
 - Query and documents

Retrieval Models

- A retrieval model specifies the details of:
 - Document representation
 - Query representation
 - Retrieval function
- Determines a notion of relevance.
- Notion of relevance can be binary or continuous (i.e. *ranked retrieval*).

IR Models



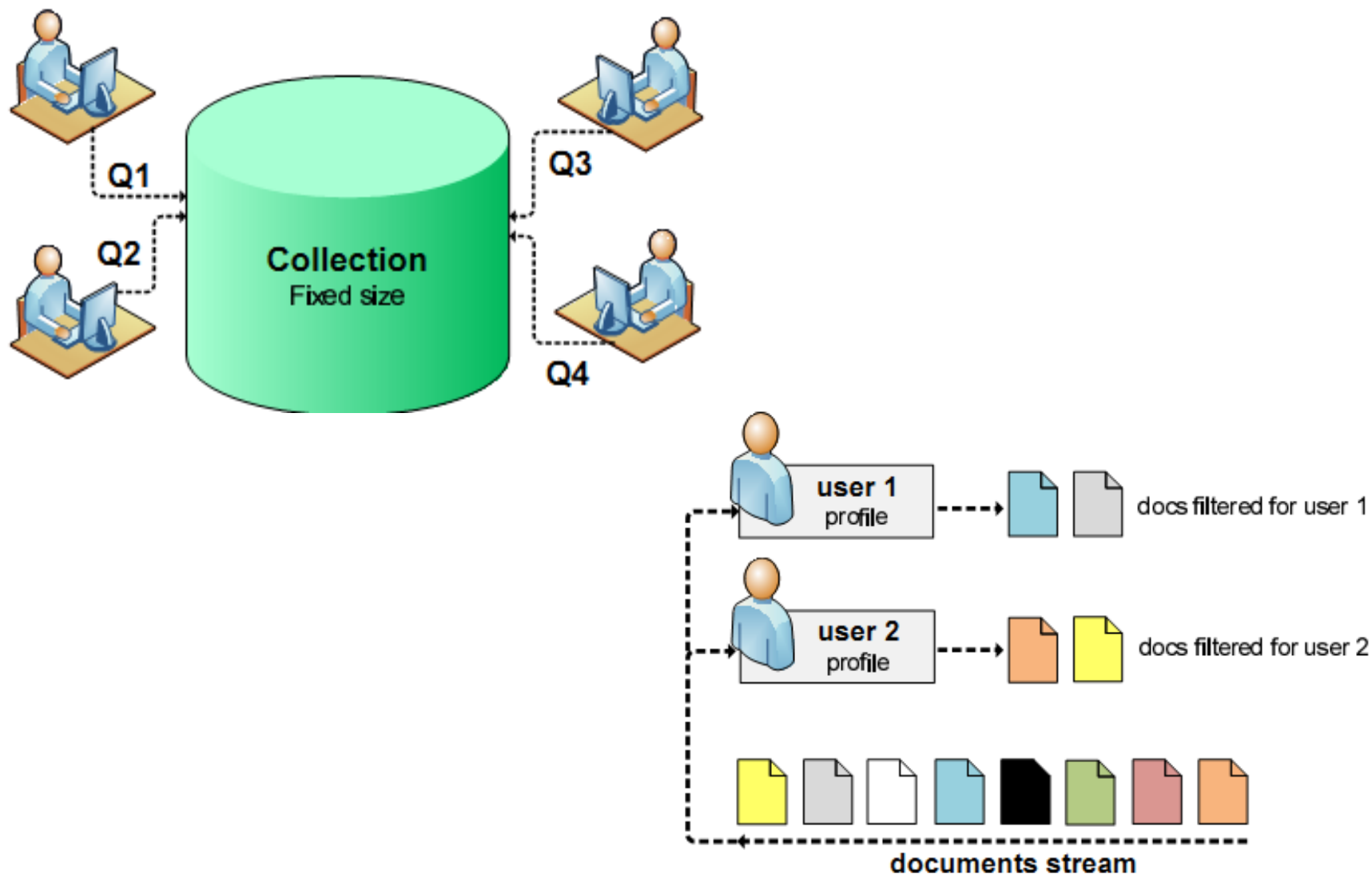
Classes of Retrieval Models

- Boolean models (set theoretic)
 - Extended Boolean
- Vector space models (statistical/algebraic)
 - Generalized VS
 - Latent Semantic Indexing
- Probabilistic models

Other Model Dimensions

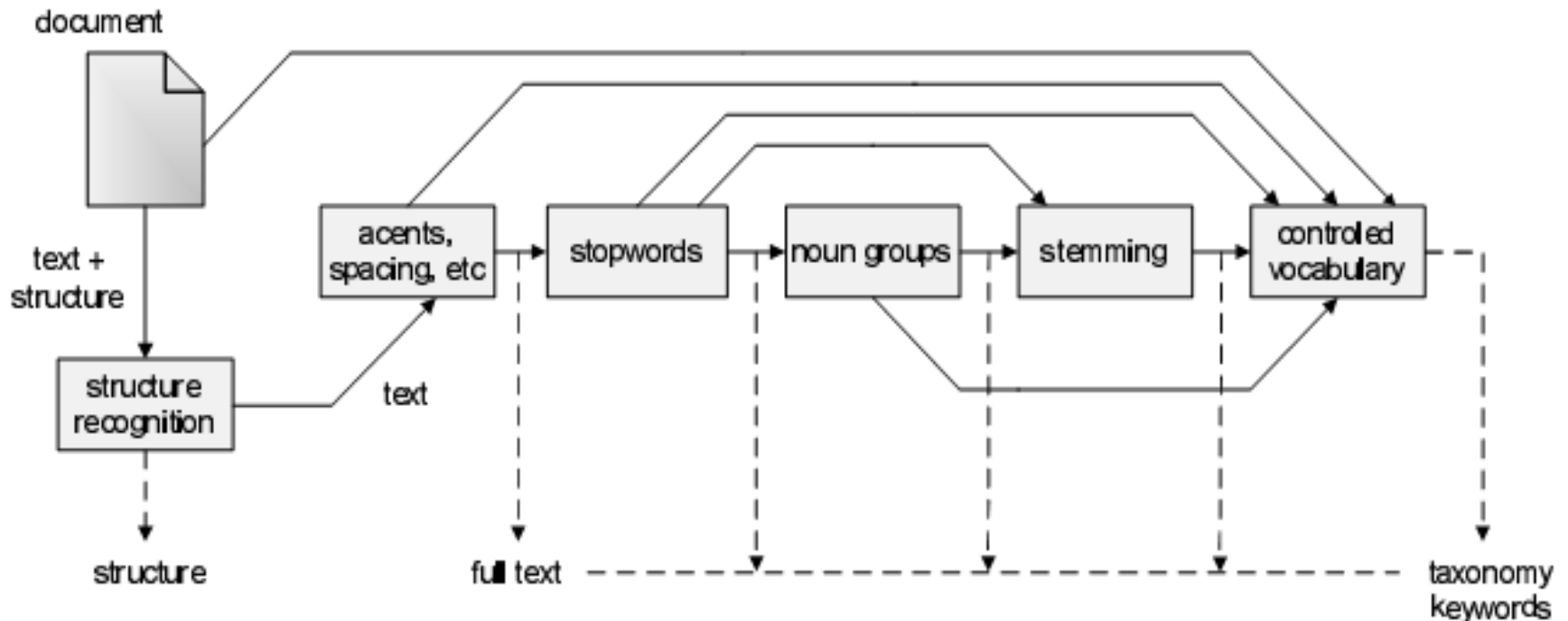
- Logical View of Documents
 - Index terms
 - Full text
 - Full text + Structure (e.g. hypertext)
- User Task
 - Retrieval
 - Browsing

Retrieval : Ad Hoc x Filtering



Logical view of a document:

- from full text to a set of index terms



Stop List

- Function words do not bear useful information for IR
of, not, to, or, in, about, with, I, be, ...
- Stop list: contain stop words, not to be used as index
 - Prepositions
 - Articles
 - Pronouns
 - Some adverbs and adjectives
 - Some frequent words (e.g. document)
- The removal of stop words usually improves IR effectiveness
- A few “standard” stop lists are commonly used.

Stemming

- *Reason:*
 - *Different word forms may bear similar meaning (e.g. search, searching): create a “standard” representation for them*
- *Stemming:*
 - *Removing some endings of word*

dancer
dancers
dance
danced
dancing

} **dance**

- **2 Types**
 - *Dictionary-based stemming*
 - *high stemming accuracy*
 - *Porter-style stemming*

Boolean Model

Boolean Model

The Boolean model of IR (BIR) is

- *a classical IR model and, at the same time,*
- *the first and most adopted one.*

It is used by virtually all commercial IR systems today.

The BIR is based on:

- *Boolean Logic and*
- *classical Sets Theory*

in that both the documents to be searched and the user's query are conceived as sets of terms.

Retrieval is based on whether or not the documents contain the query terms.

Boolean model

- Each document or query is treated as a “**bag**” of **words** or **terms**. Word sequence is not considered.
- Given a collection of documents D ,
 - let $V = \{t_1, t_2, \dots, t_{|V|}\}$ be the set of distinctive words/terms in the collection.
 - V is called the **vocabulary**.
- A weight $w_{ij} > 0$ is associated with each term t_i of a document $\mathbf{d}_j \in D$.
- For a term that does not appear in document \mathbf{d}_j , $w_{ij} = 0$.

$$\mathbf{d}_j = (w_{1j}, w_{2j}, \dots, w_{|V|j}),$$

Boolean Logic

$$C = A$$

$$C = \overline{A}$$

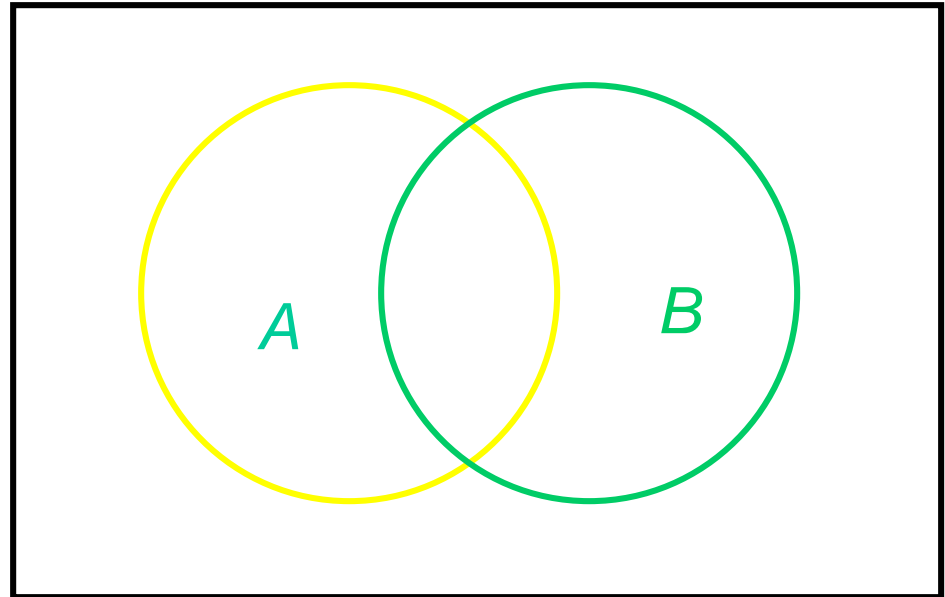
$$C = A \cap B$$

$$C = A \cup B$$

DeMorgan's Law :

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

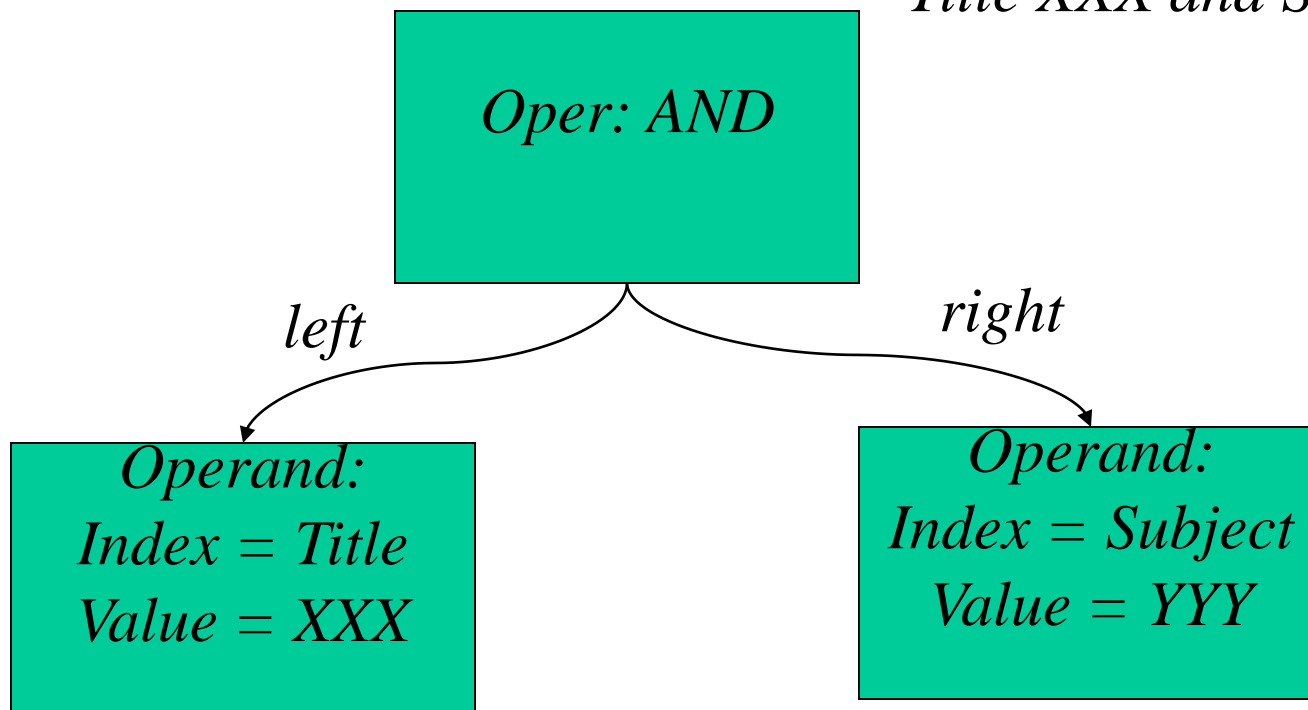
$$\overline{A \cup B} = \overline{A} \cap \overline{B}$$



Parse Result (Query Tree)

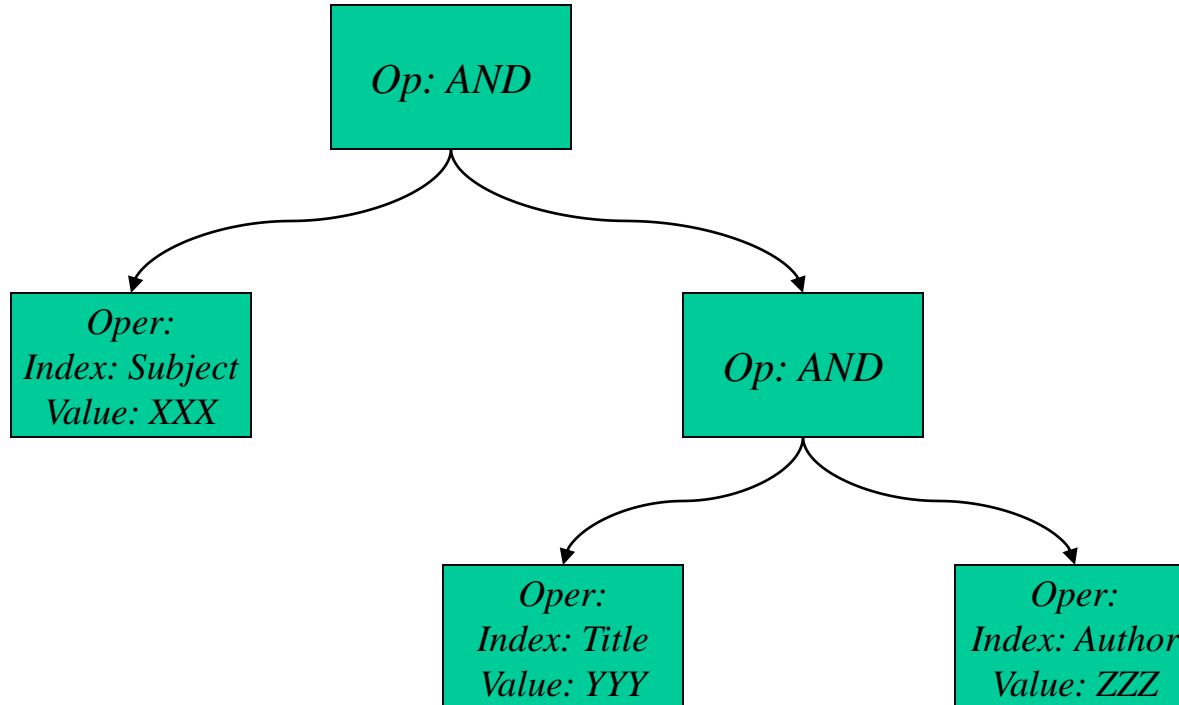
- Z39.50 queries...

Title XXX and Subject YYY



Parse Results

- Subject XXX and (title yyy and author zzz)



Boolean AND Algorithm

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

AND

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

=

2
8
15
100
135
155
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
141
150
155
188
189
190
195
198

Boolean AND NOT Algorithm

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

AND NOT

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

=

5
7
15
29
35
140
190
198

Boolean model (contd)

- Query terms are combined logically using the Boolean operators **AND**, **OR**, and **NOT**.
 - E.g., *((data AND mining) AND (NOT text))*
- Retrieval
 - Given a Boolean query, the system retrieves every document that makes the query logically true.
 - Called **exact match**.
- The retrieval results are usually quite poor because term frequency is not considered.

Boolean queries: Exact match

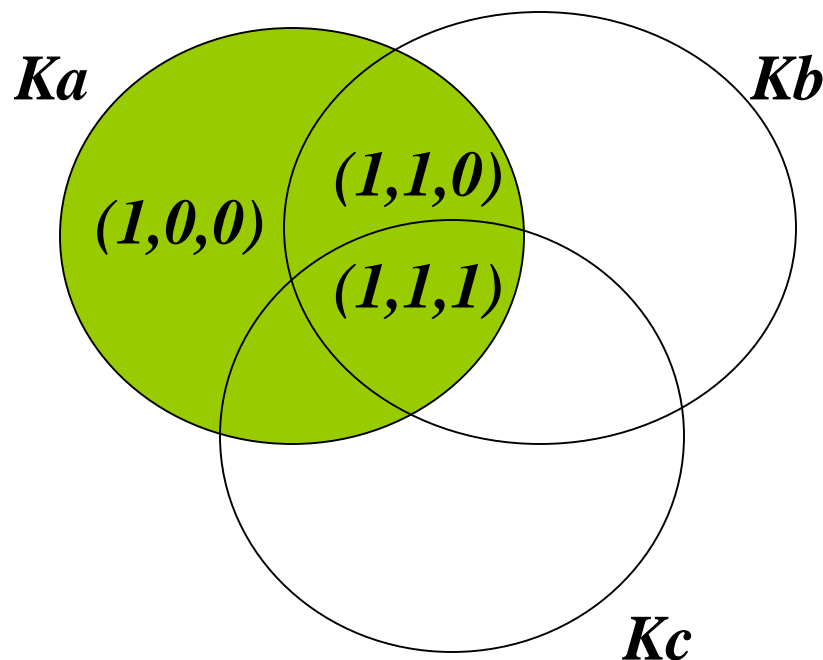
- The **Boolean retrieval model** is being able to ask a query that is a Boolean expression:
 - Boolean Queries are queries using *AND*, *OR* and *NOT* to join query terms
 - Views each document as a set of words
 - Is precise: document matches condition or not.
 - Perhaps the simplest model to build an IR system on
- Primary commercial retrieval tool for 3 decades.
- Many search systems you still use are Boolean:
 - Email, library catalog, Mac OS X Spotlight

The Boolean Model

- *Simple model based on set theory*
- *Queries specified as boolean expressions*
 - *precise semantics*
 - *neat formalism*
 - $q = ka \wedge (kb \vee \neg kc)$
- *Terms are either present or absent. Thus, $w_{ij} \in \{0,1\}$*
- *Consider*
 - $q = ka \wedge (kb \vee \neg kc)$
 - $vec(qdnf) = (1,1,1) \vee (1,1,0) \vee (1,0,0)$
 - $vec(qcc) = (1,1,0)$ is a conjunctive component
- *Each query can be transformed in Disjunctive normal form (DNF) form*

The Boolean Model

- $q = ka \wedge (kb \vee \neg kc)$



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

- no in-between, only 0 or 1

Retrieval is defined as follows:

1. The set S_i of documents are obtained that contain or not the term under focus:

for term A : $S_i = \{D \mid A \in D\}$

for negated term A , i.e., $\neg A$: $S_i = \{D \mid A \notin D\}$

2. Those documents are retrieved in response to Q , which belong to the set obtained as a result of the corresponding sets operations:

intersection \cap corresponds to logical AND,

union \cup corresponds to logical OR.

For example,

$Q = A \text{ OR } (B \text{ AND } C)$

S_1 results set for term A ,

S_2 results set for term B ,

S_3 results set for term C ,

The retrieved set in response to Q :

$S_1 \cup (S_2 \cap S_3)$

EXAMPLE

Let the set of original (real) documents $D = \{D1, D2, D3\}$

D1 = Bayes' Principle: The principle that, in estimating a parameter, one should initially assume that each possible value has equal probability (a uniform prior distribution).

D2 = Bayesian Decision Theory: A mathematical theory of decision-making which presumes utility and probability functions, and according to which the act to be chosen is the Bayes act, i.e. the one with highest Subjective Expected Utility. If one had unlimited time and calculating power with which to make every decision, this procedure would be the best way to make any decision.

D3 = Bayesian Epistemology: A philosophical theory which holds that the epistemic status of a proposition (i.e. how well proven or well established it is) is best measured by a probability and that the proper way to revise this probability is given by Bayesian conditionalisation or similar procedures. A Bayesian epistemologist would use probability to define, and explore the relationship between, concepts such as epistemic status, support or explanatory power.

Let the set T of terms be:

$T = \{$
 t1 = Bayes' Principle,
 t2 = probability,
 t3 = decision-making,
 t4 = Bayesian Epistemology }

Example

The set D of documents is as follows:

$D = \{D1, D2, D3\}$, where

$D1 = \{\text{Bayes' Principle, probability}\}$

$D2 = \{\text{probability, decision-making}\}$

$D3 = \{\text{probability, Bayesian Epistemology}\}$

Let the query Q be:

$Q = \text{probability AND decision-making}$

Solution

Q = probability AND decision-making

1. Firstly, the following sets $S1$ and $S2$ of documents Dj are obtained (retrieved):

$$S1 = \{Dj \mid \text{probability} \in Dj\} = \{D1, D2, D3\}$$

$$S2 = \{Dj \mid \text{decision-making} \in Dj\} = \{D2\}$$

2. Finally, the following documents Dj are retrieved in response to Q :

$$\begin{aligned} \{Dj \mid Dj \in S1 \cap S2\} &= \{D1, D2, D3\} \cap \{D2\} \\ &= \{D2\} \end{aligned}$$

Boolean Model example

Doc 1: “Computers have brought the world to our fingertips. We will try to understand at a basic level the **science** -- old and new -- underlying this new Computational Universe. Our quest takes us on a broad sweep of scientific **knowledge** and related technologies... Ultimately, this study makes us look anew at ourselves -- our genome; language; music; "**knowledge**"; and, above all, the mystery of our intelligence. (cos 116 description)

Doc 2: “An introduction to computer science in the context of scientific, engineering, and commercial applications. The goal of the course is to teach basic principles and practical issues, while at the same time preparing students to use computers effectively for applications in computer science ...” (cos 126 description)

Query: (principles AND knowledge) OR (science AND engineering)

Doc 1:	0	1	1	0	FALSE
--------	---	---	---	---	-------

Boolean Model example

Doc 1: “Computers have brought the world to our fingertips. We will try to understand at a basic level the science -- old and new -- underlying this new Computational Universe. Our quest takes us on a broad sweep of scientific knowledge and related technologies... Ultimately, this study makes us look anew at ourselves -- our genome; language; music; "knowledge"; and, above all, the mystery of our intelligence. (cos 116 description)

Doc 2: “An introduction to computer **science** in the context of scientific, **engineering**, and commercial applications. The goal of the course is to teach basic **principles** and practical issues, while at the same time preparing students to use computers effectively for applications in computer **science** ...”
(cos 126 description)

Query: (principles AND knowledge) OR (science AND engineering)

Doc 2:	1	0	1	1	TRUE
--------	---	---	---	---	------

Exercise

- $D_1 = \text{“computer information retrieval”}$
 - $D_2 = \text{“computer retrieval”}$
 - $D_3 = \text{“information”}$
 - $D_4 = \text{“computer information”}$
-
- $Q_1 = \text{“information} \wedge \text{retrieval”}$
 - $Q_2 = \text{“information} \wedge \neg \text{computer”}$

Boolean Retrieval Model

- Popular retrieval model because:
 - Easy to understand for simple queries.
 - Clean formalism.
- Strengths
 - Precise, if you know the right strategies
 - Precise, if you have an idea of what you're looking for
 - Implementations are fast and efficient

Boolean Models – Weaknesses

- Very rigid: AND means all; OR means any.
- Difficult to express complex user requests.
- Difficult to control the number of documents retrieved.
 - *All* matched documents will be returned.
- Difficult to rank output.
 - *All* matched documents logically satisfy the query.
- Difficult to perform relevance feedback.
 - If a document is identified by the user as relevant or irrelevant, how should the query be modified?

Boolean model Weaknesses

- Weaknesses
 - Users must learn Boolean logic
 - Boolean logic insufficient to capture the richness of language
 - No control over size of result set: either too many hits or none
 - **When do you stop reading?** All documents in the result set are considered “equally good”
 - **What about partial matches?** Documents that “don’t quite match” the query may be useful also

-
- The Boolean model imposes a binary criterion for deciding relevance
 - The question of how to extend the Boolean model to accomodate partial matching and a ranking has attracted considerable attention in the past
 - Two extensions of boolean model:
 - Fuzzy Set Model
 - Extended Boolean Model

Statistical Models

- A document is typically represented by a *bag of words* (unordered words with frequencies).
- Bag = set that allows multiple occurrences of the same element.
- User specifies a set of desired terms with optional weights:
 - Weighted query terms:
 $Q = \langle \text{database } 0.5; \text{ text } 0.8; \text{ information } 0.2 \rangle$
 - Unweighted query terms:
 $Q = \langle \text{database}; \text{ text}; \text{ information} \rangle$
 - No Boolean conditions specified in the query.

Statistical Retrieval

- Retrieval based on *similarity* between query and documents.
- Output documents are ranked according to similarity to query.
- Similarity based on occurrence *frequencies* of keywords in query and document.
- Automatic relevance feedback can be supported:
 - Relevant documents “added” to query.
 - Irrelevant documents “subtracted” from query.

Issues for Vector Space Model

- How to determine important words in a document?
 - Word sense?
 - Word n -grams (and phrases, idioms,...) \rightarrow terms
- How to determine the degree of importance of a term within a document and within the entire collection?
- How to determine the degree of similarity between a document and the query?
- In the case of the web, what is a collection and what are the effects of links, formatting information, etc.?

Vector-Space Model

- t distinct terms remain after preprocessing
 - Unique terms that form the VOCABULARY
- These “orthogonal” terms form a vector space.
 - Dimension = $t = |\text{vocabulary}|$
 - 2 terms \rightarrow bi-dimensional; ...; n -terms \rightarrow n -dimensional
- Each term, i , in a document or query j , is given a real-valued weight, w_{ij} .
- Both documents and queries are expressed as t -dimensional vectors:

$$d_j = (w_{1j}, w_{2j}, \dots, w_{tj})$$

Vector-Space Model

Query as vector:

- We regard query as short document
- We return the documents ranked by the closeness of their vectors to the query, also represented as a vector.
- Vectorial model was developed in the SMART system (Salton, c. 1970) and standardly used by TREC participants and web IR systems

Algorithm for Vector Space Retrieval

1. Given a set D of documents.
2. Identify terms.
3. Exclude stopwords.
4. Apply stemming to remaining words.
5. Compute for each document D_j and term t_i a weight w_{ij} .
6. A query Q_k coming from a user is also conceived as being a document; a weight vector \mathbf{v}_k can be computed for it as well, in a similar way.
7. Retrieval is defined as follows:
Document D_j is retrieved in response to query Q_k if the document and the query are "similar enough," i.e., a similarity measure s_{jk} between the document (identified by \mathbf{v}_j) and the query (identified by \mathbf{v}_k) is over some threshold K .

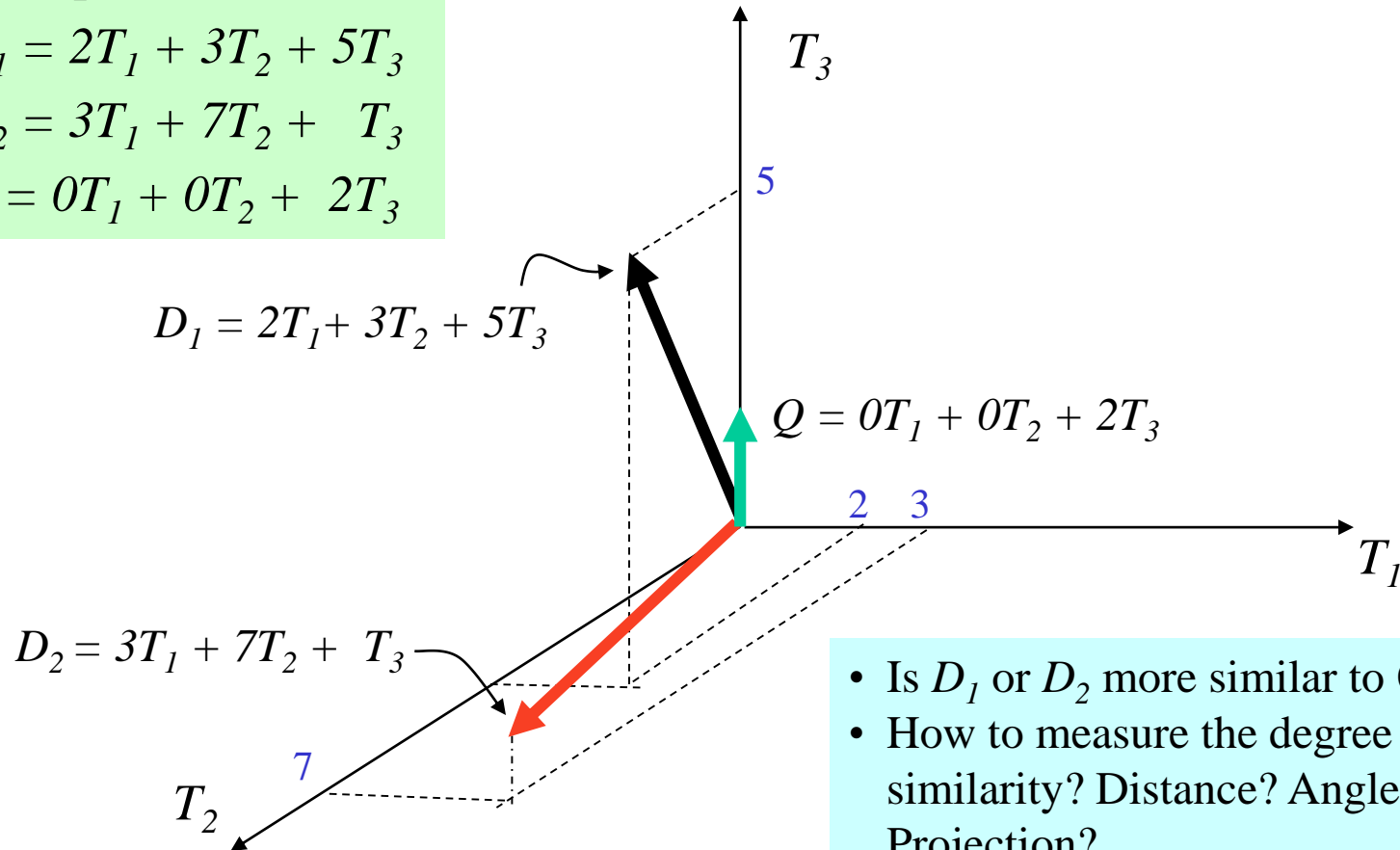
Graphic Representation

Example:

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

$$D_2 = 3T_1 + 7T_2 + T_3$$

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



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

Document Collection

- A collection of n documents 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.

$$\begin{pmatrix} & T_1 & T_2 & \dots & T_t \\ D_1 & w_{11} & w_{21} & \dots & w_{t1} \\ D_2 & w_{12} & w_{22} & \dots & w_{t2} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ D_n & w_{1n} & w_{2n} & \dots & w_{tn} \end{pmatrix}$$

Term Weights: Term Frequency

- More frequent terms in a document are more important, i.e. more indicative of the topic.

f_{ij} = frequency of term i in document j

- May want to **normalize term frequency (tf)** by dividing by the frequency of the most common term in the document:

$$tf_{ij} = f_{ij} / \max_i \{f_{ij}\}$$

Term Weights: Inverse Document Frequency (IDF)

- Terms that appear in many *different* documents are *less* indicative of overall topic.

df_i = document frequency of term i

= number of documents containing term i

idf_i = inverse document frequency of term i ,

= $\log_2 (N / df_i)$

(N : total number of documents)

- An indication of a term's *discrimination* power.
- Log used to dampen the effect relative to tf .

TF-IDF Weighting

- A typical combined term importance indicator is *tf-idf weighting*:

$$w_{ij} = tf_{ij} idf_i = tf_{ij} \log_2 (N / df_i)$$

- *A term occurring frequently in the document but rarely in the rest of the collection is given high weight.*
- Many other ways of determining term weights have been proposed.
- Experimentally, *tf-idf* has been found to work well.

Computing TF-IDF -- An Example

Given a document containing terms with given frequencies:

A(3), B(2), C(1)

Assume collection contains 10,000 documents and document frequencies of these terms are:

A(50), B(1300), C(250)

Then:

A: $tf = 3/3$; $idf = \log_2(10000/50) = 7.6$; $tf-idf = 7.6$

B: $tf = 2/3$; $idf = \log_2(10000/1300) = 2.9$; $tf-idf = 2.0$

C: $tf = 1/3$; $idf = \log_2(10000/250) = 5.3$; $tf-idf = 1.8$

Query Vector

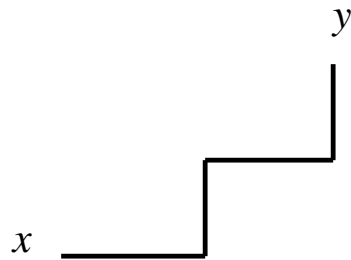
- Query vector is typically treated as a document and also tf-idf weighted.
- Alternative is for the user to supply weights for the given query terms.

Similarity Measure

- We now have vectors for all documents in the collection, a vector for the query, how to compute similarity?
- A **similarity measure** is a function that computes the *degree of similarity* between two vectors.
- Using a similarity measure between the query and each document:
 - It is possible to rank the retrieved documents in the order of presumed relevance.
 - It is possible to enforce a certain threshold so that the size of the retrieved set can be controlled.

One cut: Manhattan Distance

- Or “city block” measure
 - Based on the idea that generally in American cities you cannot follow a direct line between two points.



- Uses the formula: , also known as hamming distance

$$ManhDist(X, Y) = \sum_{i=1}^n |x_i - y_i|$$

- **Exercise:** Determine the Manhattan distance between the vectors (0, 3, 2, 1, 10) and (2, 7, 1, 0, 0)

Second cut: Euclidean distance

- Distance between vectors d_1 and d_2 is the length of the vector $|d_1 - d_2|$.

- Euclidean distance

$$\begin{aligned}d(\mathbf{p}, \mathbf{q}) &= d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^n (q_i - p_i)^2}.\end{aligned}$$

- Exercise:** Determine the Euclidean distance between the vectors $(0, 3, 2, 1, 10)$ and $(2, 7, 1, 0, 0)$
- We still haven't dealt with the issue of length normalization
 - Long documents would be more similar to each other by virtue of length, not topic
- However, we can implicitly normalize by looking at *angles* instead

Third CUT: Similarity Measure - Inner Product

- Similarity between vectors for the document \mathbf{d}_j and query \mathbf{q} can be computed as the vector inner product (a.k.a. dot product):

$$\text{sim}(\mathbf{d}_j, \mathbf{q}) = \mathbf{d}_j \bullet \mathbf{q} = \sum_{i=1}^t w_{ij} w_{iq}$$

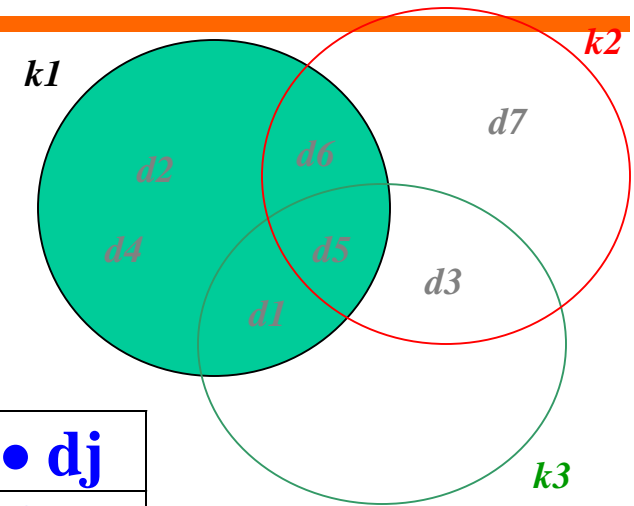
where w_{ij} is the weight of term i in document j and w_{iq} is the weight of term i in the query

- For binary vectors, the inner product is the number of matched query terms in the document (size of intersection).
- For weighted term vectors, it is the sum of the products of the weights of the matched terms.

Properties of Inner Product

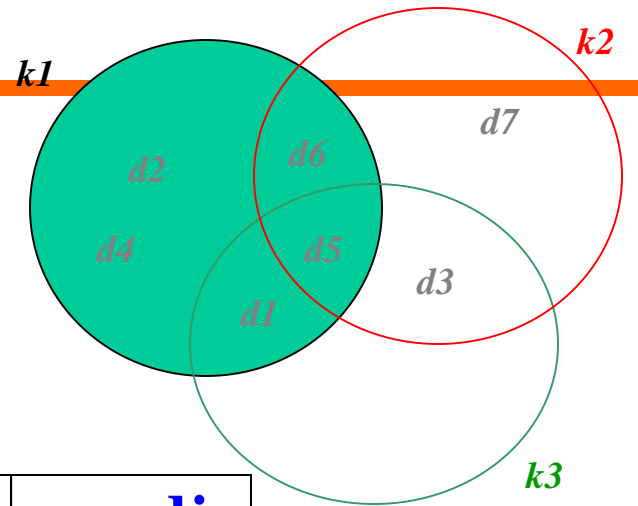
- The inner product is unbounded.
- Favors long documents with a large number of unique terms.
- Measures how many terms matched but not how many terms are *not* matched.

Inner Product: Example 1



	k1	k2	k3	$q \bullet d_j$
d1	1	0	1	2
d2	1	0	0	1
d3	0	1	1	2
d4	1	0	0	1
d5	1	1	1	3
d6	1	1	0	2
d7	0	1	0	1
q	1	1	1	

Inner Product: Exercise



	k1	k2	k3	q • d_j
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	?
q	1	2	3	

Inner Product -- Examples

Binary: retrieval database architecture computer text management information

$$- D = 1, 1, 1, 0, 1, 1, 0$$

$$- Q = 1, 0, 1, 0, 0, 1, 1$$

Size of vector = size of vocabulary = 7
0 means corresponding term not found in document or query

$$\text{sim}(D, Q) = 3$$

Weighted:

$$D_1 = 2T_1 + 3T_2 + 5T_3 \quad D_2 = 3T_1 + 7T_2 + 1T_3$$

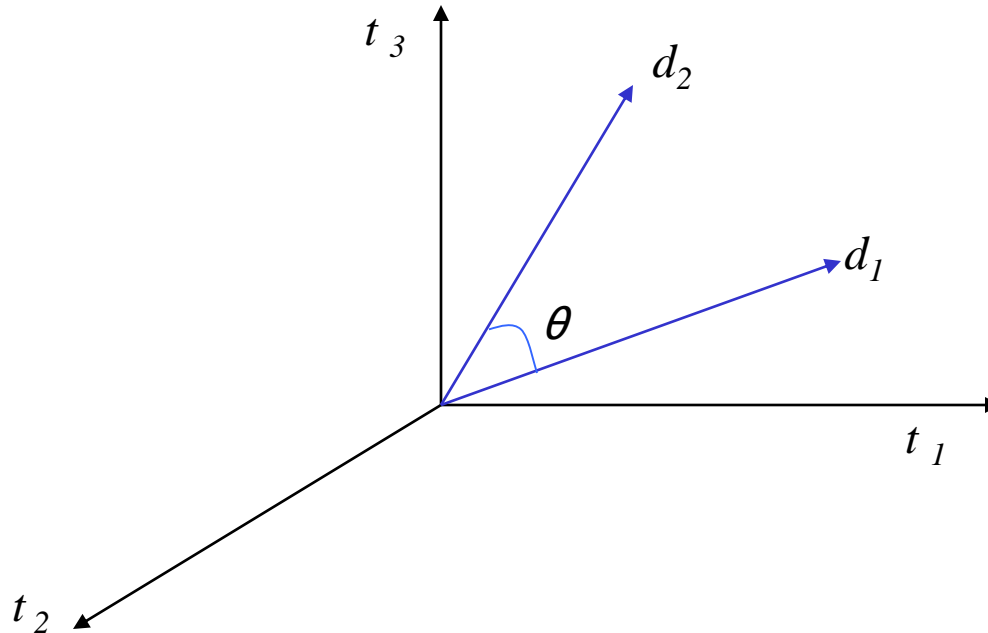
$$Q = 0T_1 + 0T_2 + 2T_3$$

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

$$\text{sim}(D_2, Q) = 3*0 + 7*0 + 1*2 = 2$$

Cosine similarity

- Distance between vectors d_1 and d_2 captured by the cosine of the angle x between them.
- Note – this is *similarity*, not distance



Cosine similarity

$$\text{sim}(d_j, d_k) = \frac{\vec{d}_j \cdot \vec{d}_k}{|\vec{d}_j| |\vec{d}_k|} = \frac{\sum_{i=1}^n w_{i,j} w_{i,k}}{\sqrt{\sum_{i=1}^n w_{i,j}^2} \sqrt{\sum_{i=1}^n w_{i,k}^2}}$$

- Cosine of angle between two vectors
- The denominator involves the lengths of the vectors
- So the cosine measure is also known as the *normalized inner product*

$$\text{Length } |\vec{d}_j| = \sqrt{\sum_{i=1}^n w_{i,j}^2}$$

Example

- Documents: Austen's *Sense and Sensibility*, *Pride and Prejudice*; Bronte's *Wuthering Heights*

	SaS	PaP	WH
<i>affection</i>	115	58	20
<i>jealous</i>	10	7	11
<i>gossip</i>	2	0	6

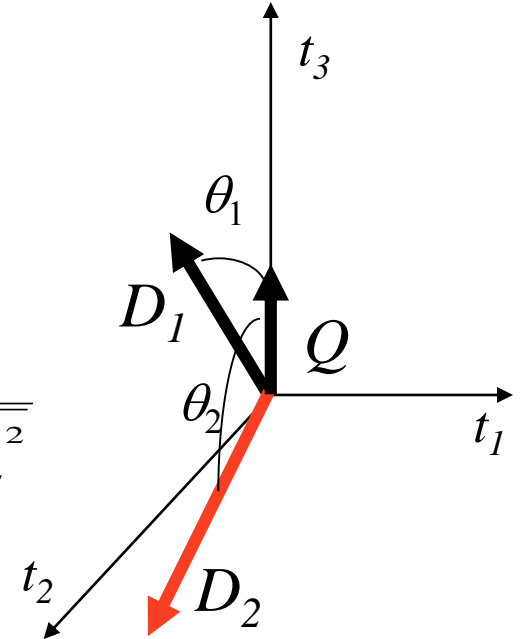
	SaS	PaP	WH
<i>affection</i>	0.996	0.993	0.847
<i>jealous</i>	0.087	0.120	0.466
<i>gossip</i>	0.017	0.000	0.254

- $\cos(\text{SAS}, \text{PAP}) = .996 \times .993 + .087 \times .120 + .017 \times 0.0 = 0.999$
- $\cos(\text{SAS}, \text{WH}) = .996 \times .847 + .087 \times .466 + .017 \times .254 = 0.929$

Cosine Similarity Measure

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

$$\text{CosSim}(\vec{d}_j, \vec{q}) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \cdot |\vec{q}|} = \frac{\sum_{i=1}^t (w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^t w_{ij}^2 \cdot \sum_{i=1}^t w_{iq}^2}}$$



$$\begin{aligned} D_1 &= 2T_1 + 3T_2 + 5T_3 & \text{CosSim}(D_1, Q) &= 10 / \sqrt{(4+9+25)(0+0+4)} = 0.81 \\ D_2 &= 3T_1 + 7T_2 + 1T_3 & \text{CosSim}(D_2, Q) &= 2 / \sqrt{(9+49+1)(0+0+4)} = 0.13 \\ Q &= 0T_1 + 0T_2 + 2T_3 \end{aligned}$$

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

Comments on Vector Space Models

- Simple, mathematically based approach.
- Considers both local (*tf*) and global (*idf*) word occurrence frequencies.
- Provides partial matching and ranked results.
- Tends to work quite well in practice despite obvious weaknesses.
- Allows efficient implementation for large document collections.

Problems with Vector Space Model

- Missing semantic information (e.g. word sense).
- Missing syntactic information (e.g. phrase structure, word order, proximity information).
- Assumption of term independence (e.g. ignores synonymy).
- Lacks the control of a Boolean model (e.g., *requiring* a term to appear in a document).
 - Given a two-term query “A B”, may prefer a document containing A frequently but not B, over a document that contains both A and B, but both less frequently.

Extended Boolean

Basic Concepts

- Instead of binary values, terms in documents and queries have a weight (importance or some other statistical property)
- *Instead of binary set membership, sets are “fuzzy” and the weights are used to determine degree of membership.*
- Degree of set membership can be used to rank the results of a query

Alternative Set Theoretic models

Extended Boolean model

- Combination of Boolean and Vector
- In comparison with Boolean model, adds “distance from query”
 - some documents satisfy the query better than others
- In comparison with Vector model, adds the distinction between AND and OR combinations
- There is a parameter (degree of norm) allowing to adjust the behavior between Boolean-like and Vector-like
- This can be even different within one query
- Not investigated well. Why not investigate it?

In the Extended Boolean model, a document is represented as a vector (similarly to in the vector model). Each i dimension corresponds to a separate term associated with the document.

The weight of term K_x associated with document d_j is measured by its normalized Term frequency and can be defined as:

$$w_{x,j} = f_{x,j} * \frac{Idf_x}{\max_i Idf_x}$$

where Idf_x is inverse document frequency.

The weight vector associated with document d_j can be represented as:

$$\mathbf{V}_{d_j} = [w_{1,j}, w_{2,j}, \dots, w_{i,j}]$$

Considering the space composed of two terms K_x and K_y only, the corresponding term weights are w_1 and w_2 .

Thus, for query $q_{or} = (K_x \vee K_y)$, we can calculate the similarity with the following formula:

$$\text{sim}(q_{or}, d) = \sqrt{\frac{w_1^2 + w_2^2}{2}}$$

For query $q_{and} = (K_x \wedge K_y)$, we can use:

$$\text{sim}(q_{and}, d) = 1 - \sqrt{\frac{(1 - w_1)^2 + (1 - w_2)^2}{2}}$$

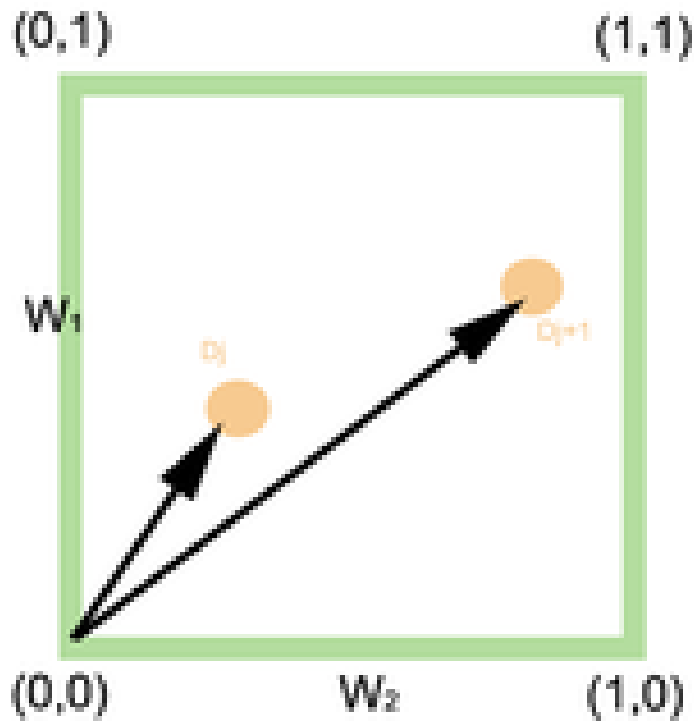


Figure 1: The similarities of $q = (Kx \vee Ky)$ with documents d_j and d_{j+1} .

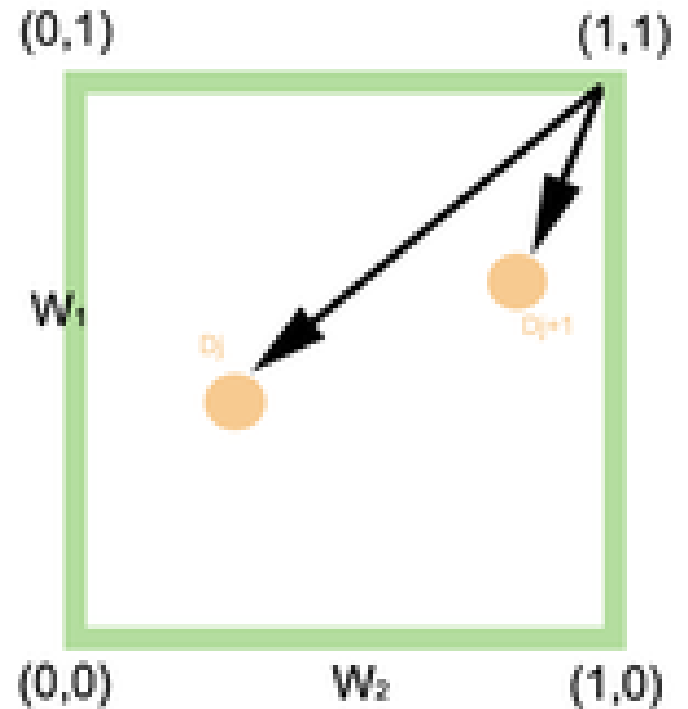


Figure 2: The similarities of $q = (Kx \wedge Ky)$ with documents d_j and d_{j+1} .



Generalizing the idea and P-norms

We can generalize the previous 2D extended Boolean model example to higher t -dimensional space using Euclidean distances.

This can be done using P-norms which extends the notion of distance to include p -distances, where $1 \leq p \leq \infty$ is a new parameter.

A generalized conjunctive query is given by:

$$q_{or} = k_1 \vee^p k_2 \vee^p \dots \vee^p k_t$$

•The similarity of  and  can be defined as:

$$sim(q_{or}, d_j) = \sqrt[p]{\frac{w_1^p + w_2^p + \dots + w_t^p}{t}}$$

A generalized disjunctive query is given by

$$q_{and} = k_1 \wedge^p k_2 \wedge^p \dots \wedge^p k_t$$

The similarity of



and



can be defined as :

$$sim(q_{and}, d_j) = 1 - \sqrt[p]{\frac{(1 - w_1)^p + (1 - w_2)^p + \dots + (1 - w_t)^p}{t}}$$

Examples

Consider the query $q = (K1 \wedge K2) \vee K3$. The similarity between query q and document d can be computed using the formula:

$$\text{sim}(q, d) = \sqrt[p]{\frac{(1 - \sqrt[p]{\frac{(1-w_1)^p + (1-w_2)^p}{2}})^p + w_3^p}{2}}$$

Fuzzy Sets

- The set membership function can be, for example, the relative term frequency within a document, the IDF or any other function providing weights to terms
- This means that the fuzzy methods use sets of criteria for term weighting that are the same or similar to those used in other ranked retrieval methods (e.g., vector and probabilistic methods)

Fuzzy Sets

Matching of a document to a query terms is approximate or vague

This **vagueness** can be modeled using a fuzzy framework, as follows:

- each query term defines a **fuzzy** set
- each doc has a **degree of membership** in this set

This interpretation provides the foundation for many IR models based on fuzzy theory

In here, we discuss the model proposed by Ogawa, Morita, and Kobayashi

Fuzzy Sets

Fuzzy set theory deals with the representation of classes whose boundaries are not well defined

Key idea is to introduce the notion of a **degree of membership** associated with the elements of the class

This degree of membership varies from 0 to 1 and allows modelling the notion of **marginal** membership

Thus, membership is now a **gradual** notion, contrary to the crispy notion enforced by classic Boolean logic

Fussy Set Theory

Introduced by Zadeh in 1965.

Definition

- *A fuzzy subset A of a universe of discourse U is characterized by a membership function $\mu_A(u)$ which associate with each element u of U a number in the interval $[0,1]$.*
- *Set Theory: $A=\{a, b, c\}$. Subset of A : $\{a, c\}$.*
- *An element is either in a set or not in a set. $\mu_A(u)$ is either 0 or 1.*

$$\mu_A : U \rightarrow [0,1]$$

Set Theory

- Let U be the set of all elements (universe)
- There are three basic operations:
 - $A \cup B = \{\text{elements in } A \text{ or in } B\}.$
 - $A \cap B = \{\text{elements in both } A \text{ and } B\}$
 - $\text{Not } A = U - A.$

Fuzzy Sets

- *Definition Let U be the universe of discourse, A and B be two fuzzy subsets of U , and \bar{A} be the complement of A relative to U . Also, let u be an element of U . Then,*

$$\mu_{\bar{A}} = 1 - \mu_A(u)$$

$$\mu_{A \cup B}(u) = \max\{\mu_A(u), \mu_B(u)\}$$

$$\mu_{A \cap B}(u) = \min\{\mu_A(u), \mu_B(u)\}$$

Fuzzy Sets

- If we have three documents and three terms...
 - $D_1=(.4,.2,1)$, $D_2=(0,0,.8)$, $D_3=(.7, .4,0)$

For search: $t_1 \cup t_2 \cup t_3$

$$v(D_1) = \max(.4, .2, 1) = 1$$

$$v(D_2) = \max(0, 0, .8) = .8$$

$$v(D_3) = \max(.7, .4, 0) = .7$$

For search: $t_1 \cap t_2 \cap t_3$

$$v(D_1) = \min(.4, .2, 1) = .2$$

$$v(D_2) = \min(0, 0, .8) = 0$$

$$v(D_3) = \min(.7, .4, 0) = 0$$

Fuzzy Sets

- Fuzzy set membership of term to document is $f(A) \rightarrow [0,1]$

$$D_1 = \{(\text{mesons}, .8), (\text{scattering}, .4)\}$$

$$D_2 = \{(\text{mesons}, .5), (\text{scattering}, .6)\}$$

Query = MESONS AND SCATTERING

$$RSV(D_1) = \text{MIN}(.8, .4) = .4$$

$$RSV(D_2) = \text{MIN}(.5, .6) = .5$$

D_2 is ranked before D_1 in the result set.

- However, consistent with the Boolean model:
 - Query = $t_1 \cap t_2 \cap t_3 \cap \dots \cap t_{100}$
 - If D not indexed by t_1 then it fails, even if D is indexed by t_2, \dots, t_{100}

Fuzzy Information Retrieval

We first set up *term-term correlation* matrix:

For terms k_i and k_l ,

$$C_{i,l} = \frac{n_{i,l}}{n_i + n_l - n_{i,l}}$$

Where n_i is the number of documents containing k_i ,

n_l is the number of documents containing k_l

And $n_{i,l}$ is the number of documents containing both k_i
and k_l .

Note $C_{i,i}=1$.

Fuzzy Information Retrieval

We define a fuzzy set for each term k_i .

In the fuzzy set for k_i , a document d_j has a degree of membership

μ_{ij} computed as
$$\mu_{i,j} = 1 - \prod_{k_l \in d_j} (1 - c_{i,l})$$

Example:

$c_{1,2}=0.1, c_{1,3}=0.21$.

$D1=(0, 1, 1, 0)$. $\mu_{1,1}= 1-0.9*0.79$.

$D2=(1, 0, 0, 0)$. $\mu_{1,2}= 1-0$. (since $c_{1,1}=1$.)

How is $d3=(1, 0, 1,0)$?

Fuzzy Information Retrieval

Whenever, the document d_j contains a term that is strongly related to k_i , then the document d_j is belong to the fuzzy set of term k_i , i.e.,

$\mu_{i,j}$ is very close to 1.

Example, $c_{1,2}=0.9$,
 $d_1=(0, 1, 0, 0)$.

$$\mu_{1,1} = 1 - (1 - 0.9) = 0.9$$

Query:

- Query is a Boolean formula, e.g.,
- $q = K_a \text{ and } (K_b \text{ or not } K_c)$.

$$q = k_a \wedge (k_b \vee \neg k_c)$$

- $q = (1, 1, 1)$ or $(1, 1, 0)$ or $(1, 0, 0)$.
- Suppose q is

$$\vec{q}_{dnf} = cc_1 \vee cc_2 \vee \cdots \vee cc_p$$

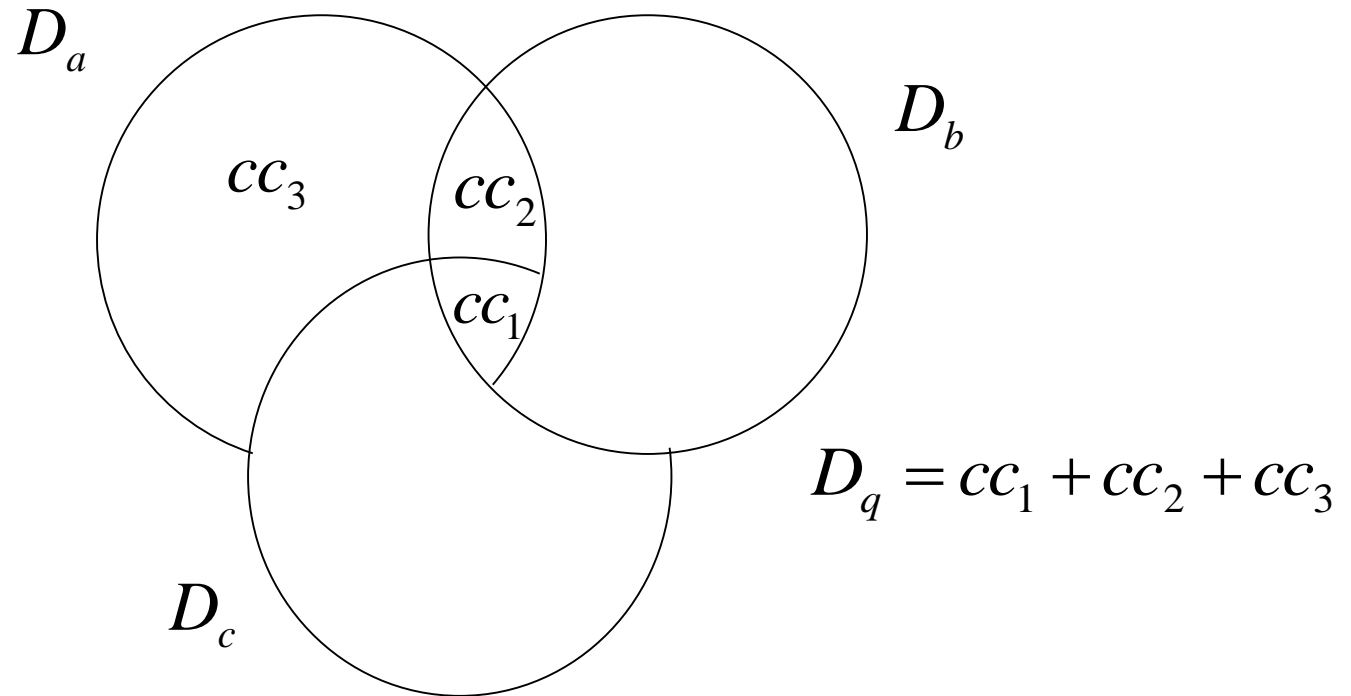


Figure 1. Fuzzy document sets for the query $[q = k_a \wedge (k_b \vee \neg k_c)]$ Each $cc_i, i \in \{1, 2, 3\}$, is a conjunctive component. D_q is the query fuzzy set.

$$\begin{aligned}
\mu_{q,j} &= \mu_{cc_1+cc_2+cc_3,j} \\
&= 1 - \prod_{i=1}^3 (1 - \mu_{cc_i,j}) \\
&= 1 - (1 - \mu_{a,j} \mu_{b,j} \mu_{c,j}) \times \\
&\quad (1 - \mu_{a,j} \mu_{b,j} (1 - \mu_{c,j})) \times (1 - \mu_{a,j} (1 - \mu_{b,j}) (1 - \mu_{c,j}))
\end{aligned}$$

Where $\mu_{i,j}, i \in \{a,b,c\}$, is the membership of d_j in the fuzzy set associated with k_i
 $\mu_{q,j}$ is the membership of document j for query q .

Example - Fuzzy Set Model

- Q: “**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**”

Eg:::: Fuzzy Set Model

- Q: “**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**”
- IDF (Select Keywords)
 - $a = \text{in} = \text{of} = 0 = \log \frac{3}{3}$
arrived = gold = shipment = truck = $0.176 = \log \frac{3}{2}$
damaged = delivery = fire = silver = $0.477 = \log \frac{3}{1}$
- 8 Keywords (Dimensions) are selected
 - arrived(1), damaged(2), delivery(3), fire(4), gold(5), silver(6), shipment(7), truck(8)

Fuzzy Set Model

$$\begin{aligned}\mu_{\text{gold},d1} &= 1 - \prod_{k_1 \in d_1} (1 - C_{\text{gold},k_1}) \\&= 1 - (1 - C_{\text{gold},\text{shipment}}) * (1 - C_{\text{gold},\text{gold}}) * (1 - C_{\text{gold},\text{damaged}}) * (1 - C_{\text{gold},\text{fire}}) \\&= 1 - \left(1 - \frac{2}{2+2-2}\right) * \left(1 - \frac{1}{2+1-1}\right) * \left(1 - \frac{2}{2+2-2}\right) * \left(1 - \frac{2}{2+1-1}\right) \\&= 1 - 0 * \frac{1}{2} * 0 * \frac{1}{2} \\&= 1 \\ \mu_{\text{silver},d1} &= 1 - 1 * 1 * 1 * 1 = 0 \\ \mu_{\text{truck},d1} &= 1 - \prod_{k_1 \in d_1} (1 - C_{\text{truck},k_1}) \\&= 1 - (1 - C_{\text{truck},\text{shipment}}) * (1 - C_{\text{truck},\text{gold}}) * (1 - C_{\text{truck},\text{damaged}}) * (1 - C_{\text{truck},\text{fire}}) \\&= 1 - \left(1 - \frac{1}{2+2-1}\right) * \left(1 - \frac{1}{2+2-1}\right) * \left(1 - \frac{0}{2+1-0}\right) * \left(1 - \frac{0}{2+1-0}\right) \\&= 1 - \frac{2}{3} * \frac{2}{3} * 1 * 1 \\&= \frac{5}{9}\end{aligned}$$

Fuzzy Set Model

$$\mu_{\text{gold},d2} = 1 - 1 * 1 * \frac{2}{3} * \frac{2}{3} = \frac{5}{9}$$

$$\mu_{\text{silver},d2} = 1$$

$$\mu_{\text{truck},d2} = 1$$

$$\mu_{\text{gold},d3} = 1$$

$$\mu_{\text{silver},d3} = 1 - 1 * 1 * \frac{1}{2} * \frac{1}{2} = \frac{3}{4}$$

$$\mu_{\text{truck},d3} = 1$$

Fuzzy Set Model

- Sim(q,d): Alternative 1

$$\mu_{q,d1} = \mu_{\text{gold} \wedge \text{silver} \wedge \text{truck},d1} = \mu_{\text{gold},d1} * \mu_{\text{silver},d1} * \mu_{\text{truck},d1} = 0$$

$$\mu_{q,d2} = \mu_{\text{gold} \wedge \text{silver} \wedge \text{truck},d1} = \mu_{\text{gold},d2} * \mu_{\text{silver},d2} * \mu_{\text{truck},d2} = \frac{5}{9}$$

$$\mu_{q,d3} = \mu_{\text{gold} \wedge \text{silver} \wedge \text{truck},d1} = \mu_{\text{gold},d3} * \mu_{\text{silver},d3} * \mu_{\text{truck},d3} = \frac{3}{4}$$

$$\text{Sim}(q,d_3) > \text{Sim}(q,d_2) > \text{Sim}(q,d_1)$$

- Sim(q,d): Alternative 2

$$\mu_{q,d1} = \mu_{\text{gold} \wedge \text{silver} \wedge \text{truck},d1} = \min(\mu_{\text{gold},d1}, \mu_{\text{silver},d1}, \mu_{\text{truck},d1}) = 0$$

$$\mu_{q,d2} = \mu_{\text{gold} \wedge \text{silver} \wedge \text{truck},d1} = \min(\mu_{\text{gold},d2}, \mu_{\text{silver},d2}, \mu_{\text{truck},d2}) = \frac{5}{9}$$

$$\mu_{q,d3} = \mu_{\text{gold} \wedge \text{silver} \wedge \text{truck},d1} = \min(\mu_{\text{gold},d3}, \mu_{\text{silver},d3}, \mu_{\text{truck},d3}) = \frac{3}{4}$$

$$\text{Sim}(q,d_3) > \text{Sim}(q,d_2) > \text{Sim}(q,d_1)$$

Fuzzy set Model

- Advantages
 - The correlations among index terms are considered
 - Degree of relevance between queries and docs can be achieved
- Disadvantages
 - Fuzzy IR models have been discussed mainly in the literature associated with fuzzy theory
 - Experiments with standard test collections are not available
 - Do not consider the frequency (or counts) of a term in a document or a query

Limitation of Fuzzzyset

- Fuzzy sets suffer from the same kind of lack of discrimination among the retrieval results almost to the same extent as standard Boolean
- The rank of a document depends entirely on the lowest or highest weighted term in an AND or OR operation

Example

- Q: “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”

Solve it with Vector Space model

Vector Space Model

- Q: “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”
- 8 Dimensions (arrived, damaged, delivery, fire, gold, silver, shipment, truck)

- $\text{Weight} = \text{TF} * \text{IDF}$

- $Q = (0, 0, 0, 0, .176, .477, 0, .176)$

$D1 = (0, .477, 0, .477, .176, 0, .176, 0)$

$D2 = (.176, 0, .477, 0, 0, .954, 0, .176)$

$D3 = (.176, 0, 0, 0, .176, 0, .176, .176)$

*Construction of
Matrix T*

Probabilistic Model

- **Objective:** to capture the IR problem using a probabilistic framework
- Given a user query, there is an *ideal* answer set
- Querying as specification of the properties of this ideal answer set (clustering)
 - Guess at the beginning what they could be (i.e., guess initial description of ideal answer set)
 - Improve by iteration

Probabilistic Model

- An initial set of documents is retrieved somehow
- User inspects these docs looking for the relevant ones (in truth, only top 10-20 need to be inspected)
- IR system uses this information to refine description of ideal answer set
- By repeating this process, it is expected that the description of the ideal answer set will improve
- Have always in mind the need to guess at the very beginning the description of the ideal answer set
- Description of ideal answer set is modeled in probabilistic terms

Probabilistic model

- Asks the question “what is probability that user will see **relevant** information if they read **this** document”
 - $P(rel|d_i)$ – **probability** of relevance after reading d_i
 - How **likely** is the user to get relevance information from reading this document
 - **high** probability means more likely to get relevant info.

Probability Ranking Principle

1. Collection of documents

2. Representation of documents

3. User uses a query

4. Representation of query

5. A set of documents to return

Question: In what order documents to present to user?

Logically: Best document first and then next best and so on

Requirement: A formal way to judge the goodness of documents with respect to query

Possibility: Probability of relevance of the document with respect to query

Probability Basics

Bayes' Rule

Let a and b are two events

$$p(a | b) p(b) = p(a \cap b) = p(b | a) p(a)$$

$$p(a | b) = \frac{p(b | a) p(a)}{p(b)}$$

$$p(\bar{a} | b) = \frac{p(b | \bar{a}) p(\bar{a})}{p(b)}$$

Odds of an event a is defined as

$$O(a) = \frac{p(a)}{p(\bar{a})} = \frac{p(a)}{1 - p(a)}$$

Probabilistic Ranking Principle

- *Given a user query q and a document d_j , the probabilistic model tries to estimate the probability that the user will find the document d_j interesting (i.e., relevant).*
- *The model assumes that this probability of relevance depends on the query and the document representations only.*
- *Ideal answer set is referred to as R and should maximize the probability of relevance. Documents in the set R are predicted to be relevant.*

Probability Ranking Principle

Let x be a document in the collection.

Let R represent **relevance** of a document w.r.t. given (fixed) query and let NR represent **non-relevance**.

$p(R/x)$ - probability that a retrieved document x is **relevant**.

$p(NR/x)$ - probability that a retrieved document x is **non-relevant**.

$$p(R | x) = \frac{p(x | R) p(R)}{p(x)}$$

$$p(NR | x) = \frac{p(x | NR) p(NR)}{p(x)}$$

$p(R), p(NR)$ - prior probability of retrieving a relevant and non-relevant document respectively

$p(x/R), p(x/NR)$ - probability that if a relevant (non-relevant) document is retrieved, it is x .

Probability Ranking Principle

$$p(R | x) = \frac{p(x | R) p(R)}{p(x)}$$

$$p(NR | x) = \frac{p(x | NR) p(NR)}{p(x)}$$

Ranking Principle (Bayes' Decision Rule):

***If $p(R/x) > p(NR/x)$ then x is relevant,
otherwise x is not relevant***

$$sim(d_j, q) = \frac{P(R|\vec{d}_j)}{P(\bar{R}|\vec{d}_j)}$$

Probabilistic model

- “if a reference retrieval system's response to each request is a **ranking** of the documents in the collection in order of decreasing **probability of relevance** ... the overall **effectiveness** of the system to its user will be the **best** that is obtainable...”, **Probability ranking principle**
 - Rank documents based on **decreasing** probability of relevance to user
 - Calculate $P(rel/d_i)$ *for each document and rank*
- Suggests **mathematical** approach to IR and matching
 - Can predict (somewhat) good retrieval **models** based on their **estimates** of $P(rel/d_i)$

Probabilistic Models

- Most probabilistic models based on combining probabilities of relevance and non-relevance of individual terms
 - Probability that a term will appear in a relevant document
 - Probability that the term will *not appear in a non-relevant document*
- These probabilities are estimated based on counting term appearances in document descriptions

Example

- Assume we have a collection of 100 documents
 - $N=100$
- 20 of the documents contain the term *UPES*
 - $n_{UPES} = 20$
- Searcher has read and marked 10 documents as relevant
 - $R=10$
- Of these relevant documents 5 contain the term *UPES*
 - $r_{UPES} = 5$
- How important is the word *UPES* to the searcher?

Probability of Relevance

- from these four numbers we can *estimate probability of UPES* given relevance information
 - i.e. how *important* term *UPES* is to relevant documents

# of relevant docs which contain UPES	\rightarrow	r_{UPES}	
# of relevant docs which don't contain UPES	\rightarrow	$(R - r_{UPES})$	$- /$

- r_{UPES} is number of relevant documents containing UPES (5)
- $R - r_{UPES}$ is number of relevant documents that do not contain UPES (5)
- Eq. (I) is
 - higher if most relevant documents contain UPES
 - lower if most relevant documents do not contain UPES
 - high* value means UPES is important *term* to user in our example (5/5=1)

Probability of Non-relevance

- Also we can estimate probability of UPES given *non-relevance* information
 - i.e. how important term *UPES* is to *non-relevant documents*

$$\frac{\text{\# of non-relevant docs which contain UPES}}{\text{\# of docs which don't contain UPES}} = \frac{(n_{UPES} - r_{UPES})}{(N - n_{UPES}) - (R - r_{UPES})} \quad - II$$

The diagram shows three boxes with arrows pointing to the formula:

- Box 1 (top left): "# of non-relevant docs which contain UPES" points to the numerator $(n_{UPES} - r_{UPES})$.
- Box 2 (bottom left): "# of docs which don't contain UPES" points to the denominator $(N - n_{UPES}) - (R - r_{UPES})$.
- Box 3 (right): "# of relevant docs which don't contain UPES" points to the term $(R - r_{UPES})$ in the denominator.

- $n_{UPES} - r_{UPES}$ is number of non-relevant documents that contain term sausage $(20-5)=15$
- $N - n_{UPES} - R + r_{UPES}$ is number of non-relevant documents that do not contain sausage $(100-20-10+5)=75$

- Eq(II)

- higher if more documents containing term UPES are non-relevant
- lower if more documents that do not contain UPES are non-relevant
- low value means UPES is important term to user in our example ($15/75=0.2$)

F4 reweighting formula

- F4 gives new weights to all terms in collection (or just query)
 - high weights to important terms
 - Low weights to unimportant terms
 - replaces *idf*, *tf*, or any other weights
 - document score is based on sum of query terms in documents

$$Similarity(d_j, q) = \sum_{i=1}^n F4_{qi}$$

F4 reweighting formula

how important is *UPES* *being present in relevant documents*

$$\log \left(\frac{\frac{r_{UPES} + 0.5}{0.5 + R - r_{UPES}}}{\frac{0.5 + n_{UPES} - r_{UPES}}{0.5 + N - n_{UPES} - R + r_{UPES}}} \right)$$

how important is *UPES* *being absent from non-relevant documents*

Probabilistic model

- can also use to **rank** terms for **addition** to query
 - rank terms in *relevant documents* by *term* reweighting formula
 - i.e. by how **good** the terms are at retrieving relevant documents
 - add all terms
 - add some, e.g. top 5
- in probabilistic formula query expansion and term reweighting done **separately**

Probabilistic model

- Advantages over **vector-space**
 - Strong **theoretical** basis
 - Based on probability theory (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

More IR models

- Alternative Set and Vector Models
 - Set-Based Model
 - Extended Boolean Model
 - Fuzzy Set Model
 - The Generalized Vector Model
 - Latent Semantic Indexing
 - Neural Network for IR

More IR models

- Alternative Algebraic Models
 - Generalized Vector Model
 - Latent Semantic Indexing
 - Neural Network Model
- Part III: Alternative Probabilistic Models
 - BM25
 - Language Models
 - Divergence from Randomness
 - Belief Network Models
 - Other Models

More IR models

- Other Models
 - Hypertext Model
 - Web-based Models
 - Structured Text Retrieval
 - Multimedia Retrieval
 - Enterprise and Vertical Search

Structured Text Retrieval

- All the IR models discussed here treat the text as a string with no particular structure
- However, information on the structure might be important to the user for particular searches
 - Ex: retrieve a book that contains a figure of the Eiffel tower in a section whose title contains the term “France”
 - Classical information model – Eiffel tower and France
- The solution to this problem is to take advantage of the text structure of the documents to improve retrieval

Early Text Retrieval Models

- We discuss now two early structured text retrieval models
- We use:
 - The term **match point** to refer to the position in the text of a word which matches the user query
 - The term **region** to refer to a contiguous portion of the text
 - The term **node** to refer to a structural component of the document

Structured Text Retrieval

- Keyword-based query answering considers that the documents are flat
 - a word in the title has the same weight as a word in the body of the document
- Document structure is one additional piece of information which can be taken advantage of
 - For instance, words appearing in the title or in sub-titles within the document could receive higher importance

Structured Text Retrieval

- Consider the following information need:
 - Retrieve all documents which contain a page in which the string “atomic holocaust” appears in italic in the text surrounding a Figure whose label contains the word earth
- The corresponding query could be:
 - **same-page(near(italic(“atomic holocaust”),
Figure(label(“earth”))))**
- Advanced interfaces that facilitate the specification of the structure are also highly desirable
- *Models which allow combining information on text content with information on document structure are called structured text models*
- Structured text models include no ranking (open research problem)

Basic Definitions

- **Match point:** the position in the text of a sequence of words that match the query
 - Query: “atomic holocaust in Hiroshima”
 - Doc dj: contains 3 lines with this string
 - Then, doc dj contains 3 match points
- **Region:** a contiguous portion of the text
- **Node:** a structural component of the text such as a chapter, a section, etc

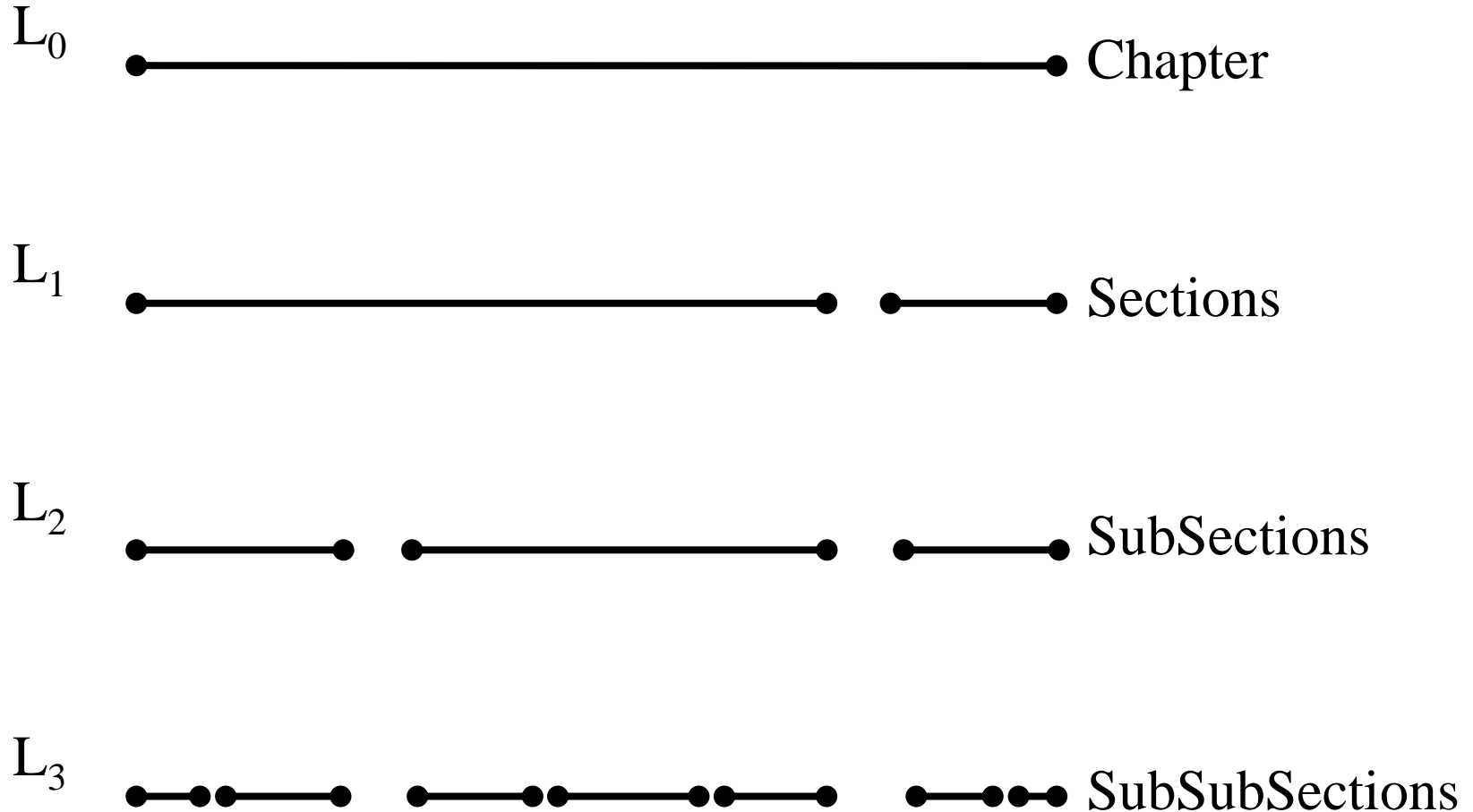
Structured Text Retrieval

- Two type
 - Model based on Non-Overlapping Lists
 - Proximal Nodes Model

Model basd on Non-Overlapping Lists

- Due to Burkowski, 1992.
- Idea: **divide the text in non-overlapping regions which are collected in a list**
- Multiple ways to divide the text in non-overlapping parts yield multiple lists:
 - a list for chapters
 - a list for sections
 - a list for subsections
- Text regions from distinct lists might overlap

Non-Overlapping Lists



Non-Overlapping Lists

- Implementation:
 - single inverted file that combines keywords and text regions
 - to each entry in this inverted file is associated a list of text regions
 - lists of text regions can be merged with lists of keywords

Non-Overlapping Lists

- Regions are non-overlapping which limits the queries that can be asked
- Types of queries:
 - select a region that contains a given word
 - select a region A that does not contain a region B (regions A and B belong to distinct lists)
 - select a region not contained within any other region

Comment : Non-Overlapping Lists

- Simple and allows efficient implementation
- Limited types of queries can be asked
- Model does not include any provision for ranking the documents by degree of similarity to the query

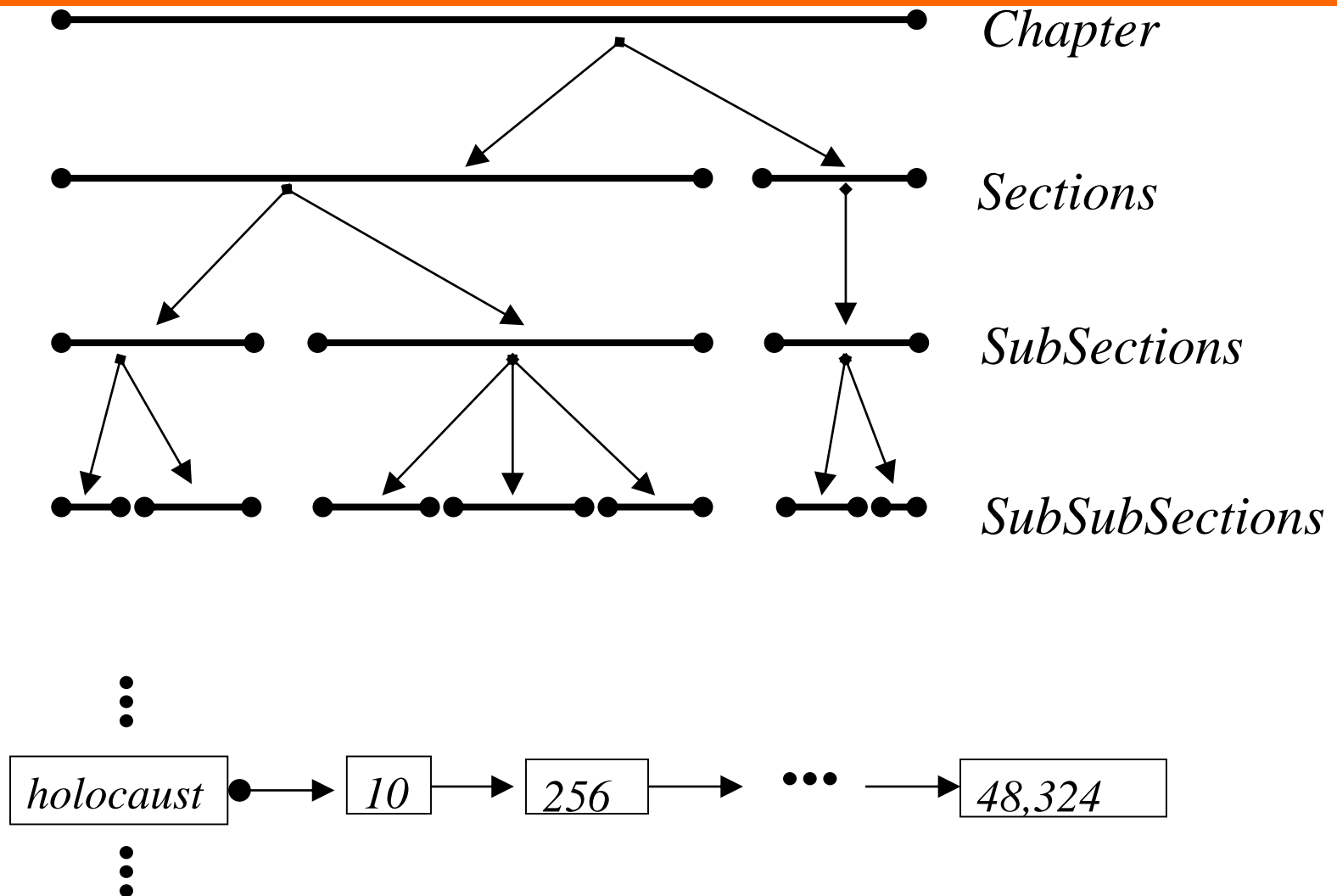
Proximal Nodes

- Navarro and Baeza-Yates, 1997
- Idea: define a strict hierarchical index over the text. This enrichs the previous model that used flat lists.
- Multiple index hierarchies might be defined
 - allows defining independent hierarchical indexing structures over the same document text
- Two distinct index hierarchies might refer to text regions that overlap

Definitions

- Each indexing structure is a strict hierarchy composed of
 - chapters
 - sections
 - subsections
 - paragraphs
 - lines
- Each of these components is called a node
- To each node is associated a text region

Proximal Nodes



Overlapped Lists

- The original idea was to have a lists of disjoint segments, originated by textual searches or by “regions” like chapters.
- It enhances the algebra with overlapping capabilities, some new operators and a framework for an implementation.
- The new operators allow to perform set union, and to combine areas.
- Combination means selecting the minimal text areas including two segments, for any two segments taken from two sets.

Lists of References

- Structure of documents is hierarchical (with only one strict hierarchy),
 - answers to queries are at (only the top-level elements qualify), and all elements must be from the same type (e.g. only sections, or only paragraphs).
- Answers to queries are seen as lists of “references”.
 - A reference is a pointer to a region of the database.
 - Integrates in an elegant way answers to queries and hypertext links, since all are lists of references.

Lists of References

- The model has also navigational features to traverse those lists.
- This model is very powerful, and because of this, has efficiency problems.
- To make the model suitable for our comparisons, we consider only the portion related to querying structures.
- Even this portion is quite powerful, and allows to efficiently solve queries by first locating the text matches and then filtering the candidates with the structural restrictions.

Proximal Nodes

- Key points:
 - In the hierarchical index, one node might be contained within another node
 - But, two nodes of a same hierarchy cannot overlap
 - The inverted list for keywords complements the hierarchical index
 - The implementation here is more complex than that for non-overlapping lists

Proximal Nodes

- Queries are now regular expressions:
 - search for strings
 - references to structural components
 - combination of these
- Model is a compromise between expressiveness and efficiency
- Queries are simple but can be processed efficiently
- Further, model is more expressive than non-overlapping lists

Proximal Nodes

- Query: find the sections, the subsections, and the subsubsections that contain the word “holocaust”
 - [(**section*) with (“holocaust”)]
- Simple query processing:
 - traverse the inverted list for “holocaust” and determine all match points
 - use the match points to search in the hierarchical index for the structural components

Proximal Nodes

- Query: [(**section*) with (“holocaust”)]
- Sophisticated query processing:
 - get the first entry in the inverted list for “holocaust”
 - use this match point to search in the hierarchical index for the structural components
 - Innermost matching component: smaller one
 - Check if innermost matching component includes the second entry in the inverted list for “holocaust”
 - If it does, check the third entry and so on
 - This allows matching efficiently the nearby (or proximal) nodes

Proximal Nodes

- Model allows formulating queries that are more sophisticated than those allowed by non-overlapping lists
- To speed up query processing, nearby nodes are inspected
- Types of queries that can be asked are somewhat limited (all nodes in the answer must come from a same index hierarchy!)
- Model is a compromise between efficiency and expressiveness

Evaluation measures

- The quality of many retrieval systems depends on how well they manage to rank relevant documents.
- How can we evaluate rankings in IR?
 - IR researchers have developed evaluation measures specifically designed to evaluate rankings.
 - Most of these measures combine **precision** and **recall** in a way that takes account of the ranking.

Conclusion

- No model is the best for all applications, especially because the more expressive the model, the less efficient can it be.
- Each application has its own set of requirements, and should select the most efficient model supporting them.
- Another important issue is the perspective of the user. When we incorporate operators and evaluate the cost of implementing them, we are implicitly assuming that they are useful for the user of the system.

Step 1: Preprocessing

- Implement the preprocessing functions:
 - For tokenization
 - For stop word removal
 - For stemming
- Input: Documents that are read one by one from the collection
- Output: Tokens to be added to the index
 - No punctuation, no stop-words, stemmed

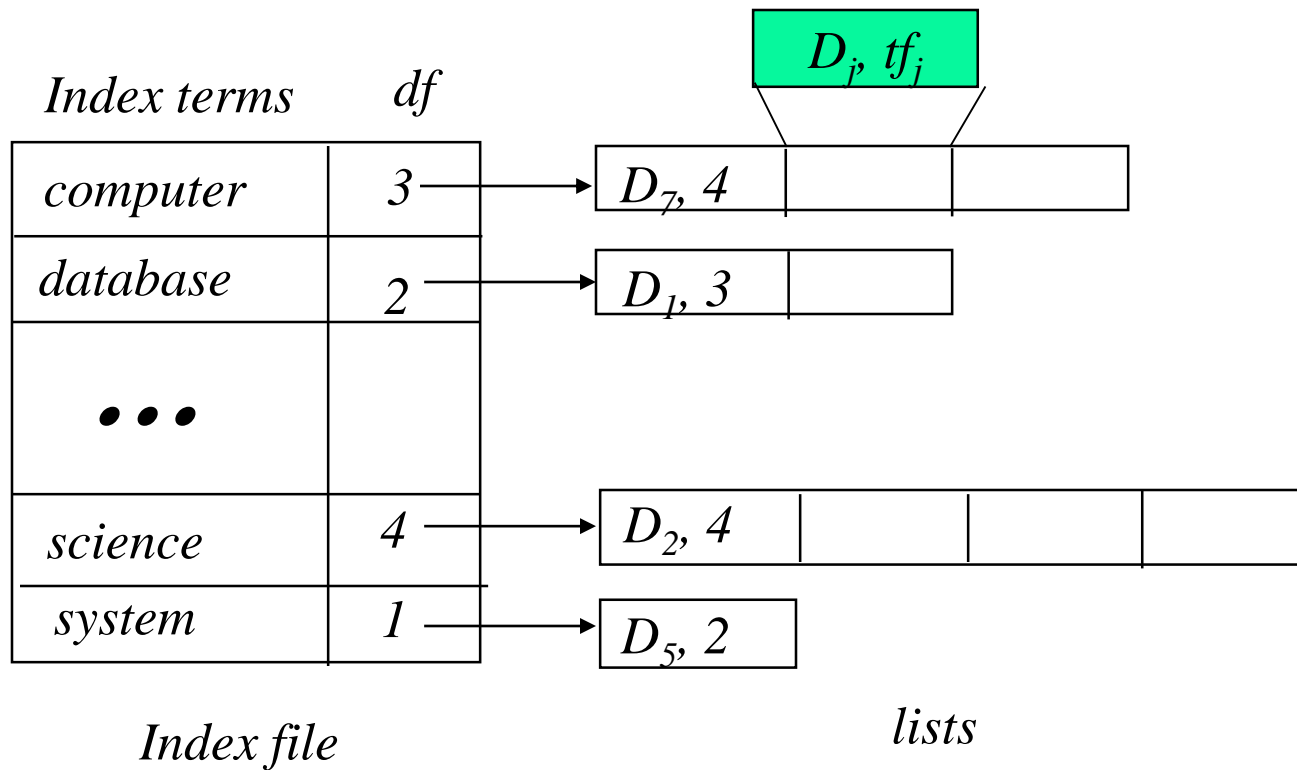
Step 2: Indexing

- Build an inverted index, with an entry for each word in the vocabulary
- Input: Tokens obtained from the preprocessing module
- Output: An inverted index for fast access

Step 2 (cont' d)

- Many data structures are appropriate for fast access
 - B-trees, skipped lists, hashtables
- We need:
 - One entry for each word in the vocabulary
 - For each such entry:
 - Keep a list of all the documents where it appears together with the corresponding frequency \rightarrow TF
 - For each such entry, keep the total number of occurrences in all documents:
 - \rightarrow IDF

Step 2 (cont' d)



Step 2 (cont' d)

- TF and IDF for each token can be computed in one pass
- Cosine similarity also required document lengths
- Need a second pass to compute document vector lengths
 - Remember that the length of a document vector is the square-root of sum of the squares of the weights of its tokens.
 - Remember the weight of a token is: $TF * IDF$
 - Therefore, must wait until IDF's are known (and therefore until all documents are indexed) before document lengths can be determined.
- Do a second pass over all documents: keep a list or hashtable with all document id-s, and for each document determine its length.

Time Complexity of Indexing

- Complexity of creating vector and indexing a document of n tokens is $O(n)$.
- So indexing m such documents is $O(m n)$.
- Computing token IDFs can be done during the same first pass
- Computing vector lengths is also $O(m n)$.
- Complete process is $O(m n)$, which is also the complexity of just reading in the corpus.

Step 3: Retrieval

- Use inverted index (from step 2) to find the limited set of documents that contain at least one of the query words.
 - Incrementally compute cosine similarity of each indexed document as query words are processed one by one.
 - To accumulate a total score for each retrieved document, store retrieved documents in a hashtable, where the document id is the key, and the partial accumulated score is the value.
-
- Input: Query and Inverted Index (from Step 2)
 - Output: Similarity values between query and documents

Step 4: Ranking

- Sort the hashtable including the retrieved documents based on the value of cosine similarity
 - $\text{sort } \{\$retrieved\{\$b\} \Leftrightarrow \$retrieved\{\$a\}\} \text{ keys}$
 $\%retrieved$
- Return the documents in descending order of their relevance
- Input: Similarity values between query and documents
- Output: Ranked list of documents in reversed order of their relevance