

# Seminars in Database

---

Prerequisites – 2nd

F.Ricci

# Content

- Information Retrieval
- Web search
- Indexing
- Document Model
- Relevance
- Evaluation of an IR system

# Basic Concepts in Information Retrieval

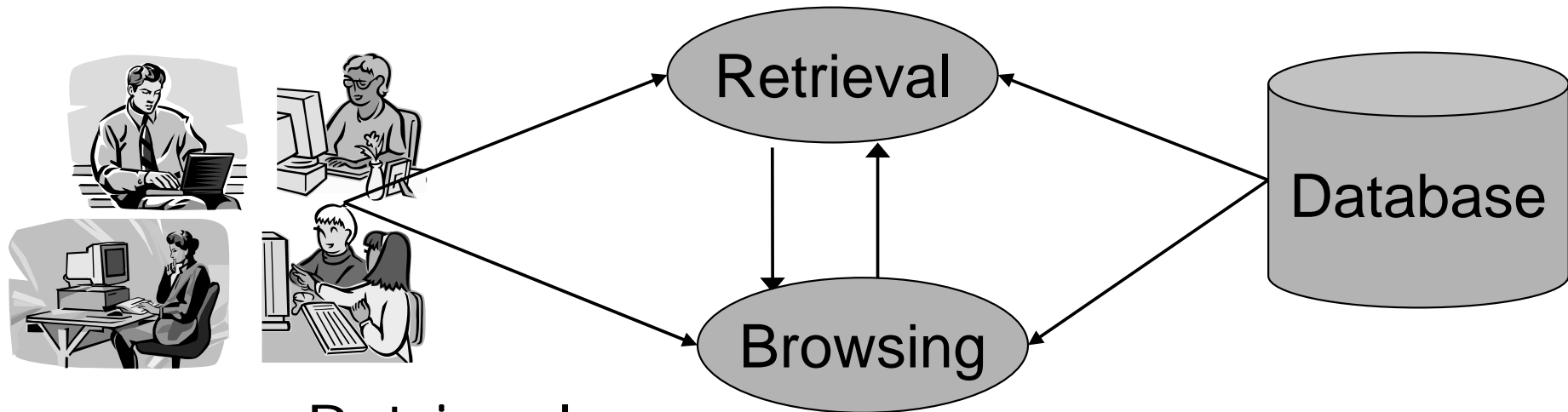
- Information Retrieval (IR) deals with the representation, storage and organization of **unstructured** data
- Its mission is to assist in information “discovery” (more exactly information search)
- 2 main discovery paradigms:

## **Search and Browse**

- *Thanks to Aya Sofer and David Carmel (IBM Haifa) for these IR slides.*

# Basic Concepts

## □ The User Task



### ■ Retrieval

- Search for particular information
- Usually focused and purposeful

### ■ Browsing

- General looking around for information
- For example: Asia -> Thailand -> Phuket  
-> Tsunami

# Data- vs Information- Retrieval (from CJ Risjbergen)

|                     | DR         | IR           |
|---------------------|------------|--------------|
| Matching            | exact      | partial/best |
| Inference Model     | deduction  | induction    |
| Query Language      | artificial | natural      |
| Query Specification | complete   | incomplete   |
| Items wanted        | matching   | relevant     |
| Error Response      | sensitive  | insensitive  |

# Web search

- The Web is not a DB
  - no structure
  - can be seen as a flat collection of documents (What search engines did at the beginning, e.g., AltaVista)
- Yet the Web has a meta-structure:
  - it is a graph
  - using the information embedded in the graph led to a major breakthrough in IR (e.g., PageRank → Google).

# Search: The Basic Concepts

- The user has an **information need**, that is expressed as a **free-text query**
- The query **is a "document"**, to be compared to a collection of documents
- **Effectiveness vs Efficiency**
- How to **compare documents**?  
Similarity metrics needed!
- How to **avoid** doing a **sequential search**? Can we search in parallel in a set of servers?

# Basic Concepts

- **Two** Main Stages

- **Indexing** process (build knowledge base)

- Involves pre-processing and storing of information in a repository

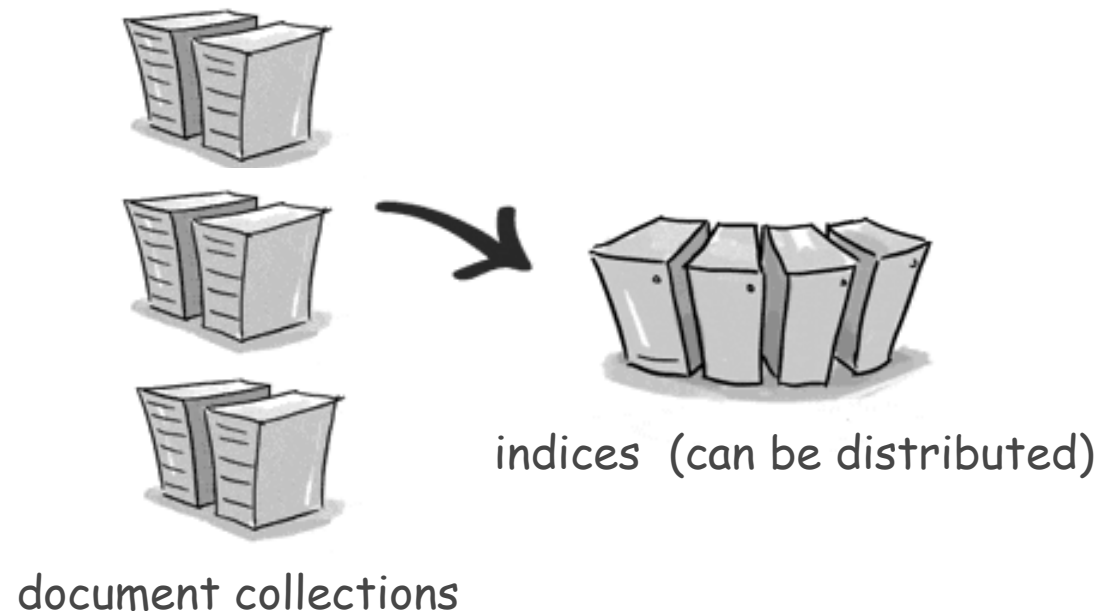
- **Retrieval** process (search knowledge base)

- Involves issuing a query, accessing the repository to find candidate documents most similar to query, and browse them

- Basic object is a document.

# Search Preprocessing Stage - Indexing

- Analyze the distribution of indexing units (words) within a collection of documents
- The **inverted index**, maps every **indexing unit** (word) to a **list of postings** (documents ids associated with frequency of occurrence, location information, etc...)

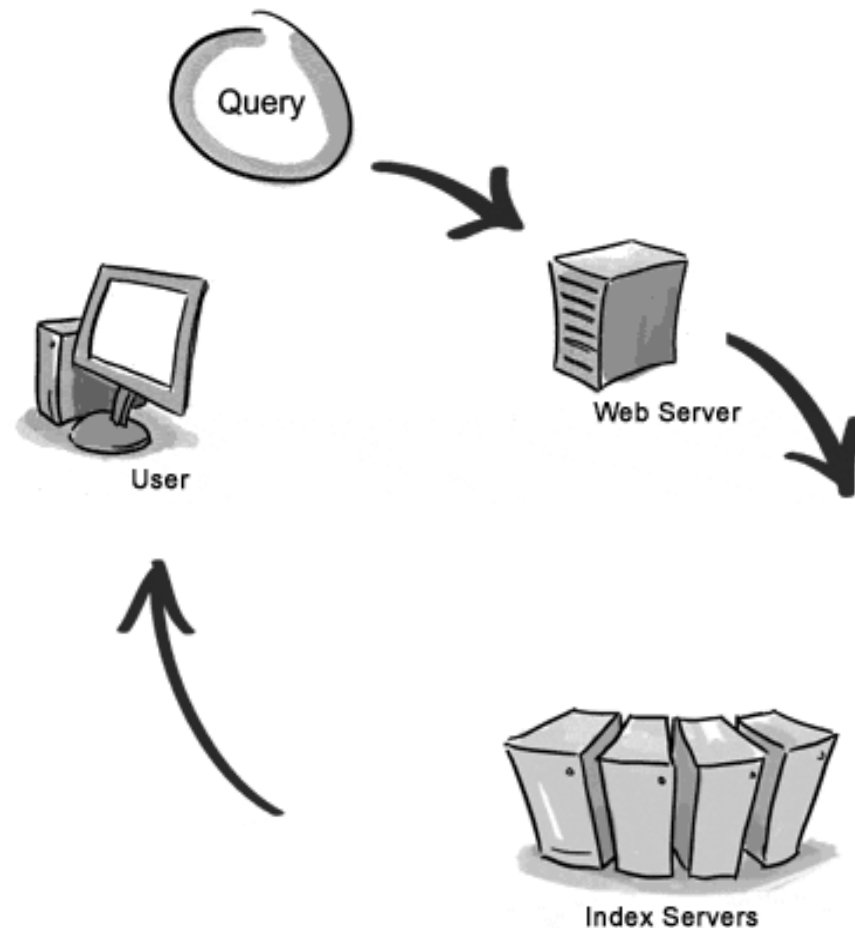


# Inverted Index

- Given the texts  $T_0 = \text{"it is what it is"}$ ,  $T_1 = \text{"what is it"}$  and  $T_2 = \text{"it is a banana"}$
- We have the following *full inverted index*:
  - "a":  $\{(2, 2)\}$
  - "banana":  $\{(2, 3)\}$
  - "is":  $\{(0, 1), (0, 4), \mathbf{(1, 1)}, (2, 1)\}$
  - "it":  $\{(0, 0), (0, 3), \mathbf{(1, 2)}, (2, 0)\}$
  - "what":  $\{(0, 2), \mathbf{(1, 0)}\}$
- In bold face the postings associated to document T1

# The Actual Search Process

- At retrieval time, the query is indexed. Postings are fetched, and analyzed to return the most similar documents
- Effectivity constraint: *"The entire process should not take more than a second per query"*



# Classical IR Models

- **Boolean model** (*Does the term occur in document?*)
  - Simple model based on **set theory**
  - **Queries** are specified as **Boolean expressions**
- **Vector space model** (*Similarity between vectors*)
  - **Query, Documents** are represented as **vectors** in a N-dimensional space, N is the number of terms in the corpus
  - Find documents **most similar** to the query in that space
- **Probabilistic model** (*Find probabilistically best answer*)
  - Goal: model IR with probabilistic framework
  - Given a user's query find out the document with the higher probability to be relevant to the query.

# Text Preprocessing Techniques

- **Goal:** construct a canonical form of the document text called *document profile*
- Stages
  - Sentence splitting – sentence separation
  - Tokenization - word separation
  - Normalization – changing terms to a standard form (e.g., lowercase)
  - Stop-word filtering: ignores frequent terms (e.g. *is, the, as, I, at, in, to...*)
  - Lemmatization – reducing terms to their base form
  - Map between documents and indexing units

# Retrieving – Query Evaluation, Ranking

## Query evaluation steps:

1. Input query
2. Parse query
3. Lexicon lookup
4. Get posting from inverted index
5. Weights accumulator
6. Sort by weights

# Evaluation Criteria

## □ Effectiveness

- How precise is the answer set to the information needs?

## □ Efficiency

- Retrieval time, indexing time, index size?

## □ Usability and Personalization

- Learnability, novice use, expert use

# Web IR- IR on the Web

## □ First Generation

- Classical approach
- Informational: IR/DB techniques on page content. E.g., Lycos, Excite, AltaVista

## □ Second Generation

- Web as a graph
- Navigational: use off-page Web specific data – links topology. E.g., Google

## □ Third Generation

- Open research
- A lot of business potential, “monetization of infomediary role”, matching services

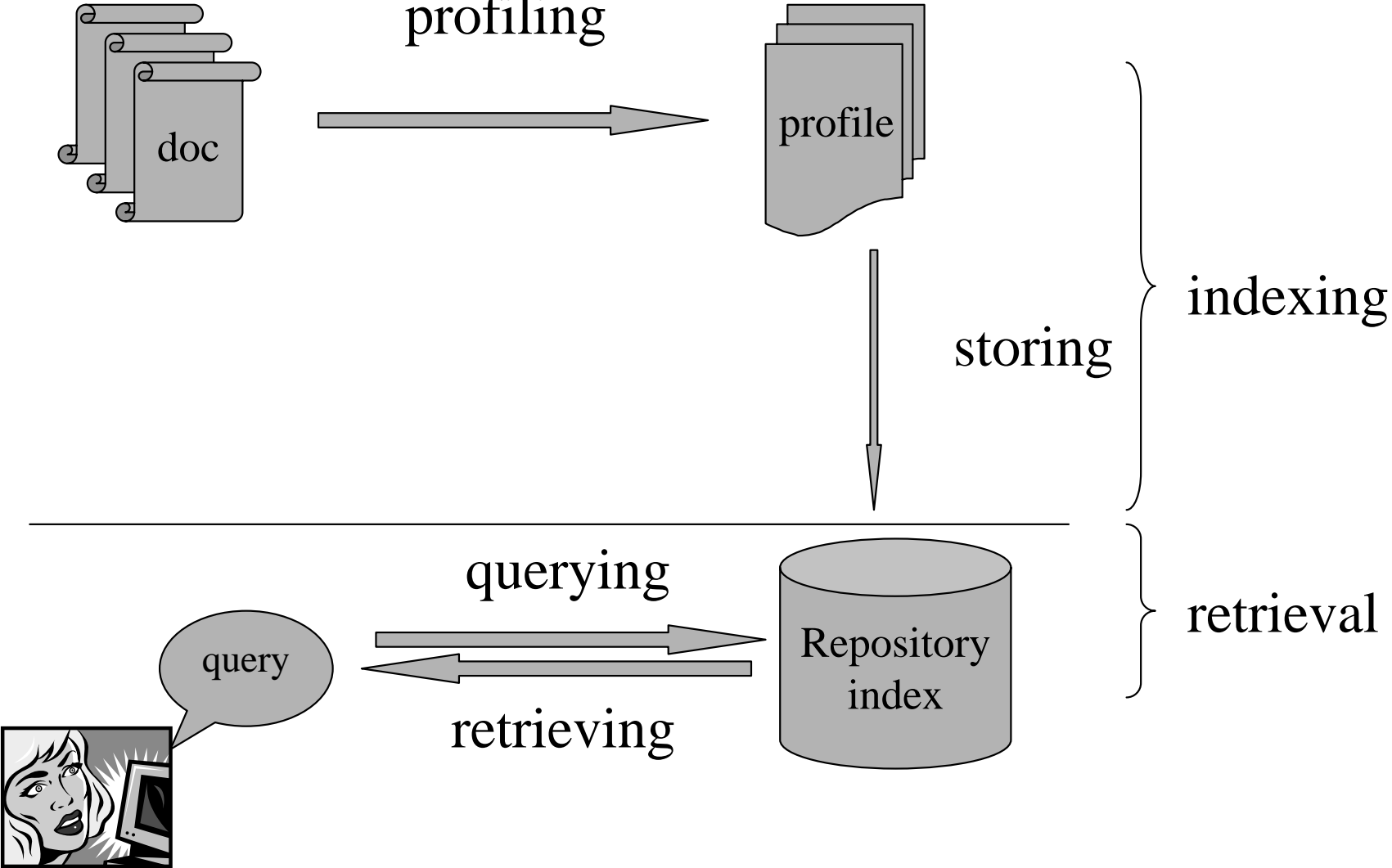
# Problems with Using IR for Web

- Very **large** and **heterogeneous** collection
- Very **short queries**
- **Unsophisticated** users
- **Difficult to judge relevance** and to rank results
- **Synonymy** and **ambiguity**
- Authorship styles
- Search engine **persuasion**, keyword *stuffing* (a web page is loaded with keywords in the meta tags or in content).

# Indexing/Retrieval

- Information Retrieval consists of 2 main stages
  - Indexing - pre-processing and storing of information into a repository (**an Index**)
  - Retrieval - issuing a query, accessing the index to find documents **RELEVANT** to the query

# Typical IR system



# Basic Concepts

## □ Document

- any piece of information (book, article, database record, image)
- usually textual data

## □ Query

- a text representing the user's information need

## □ Relevance

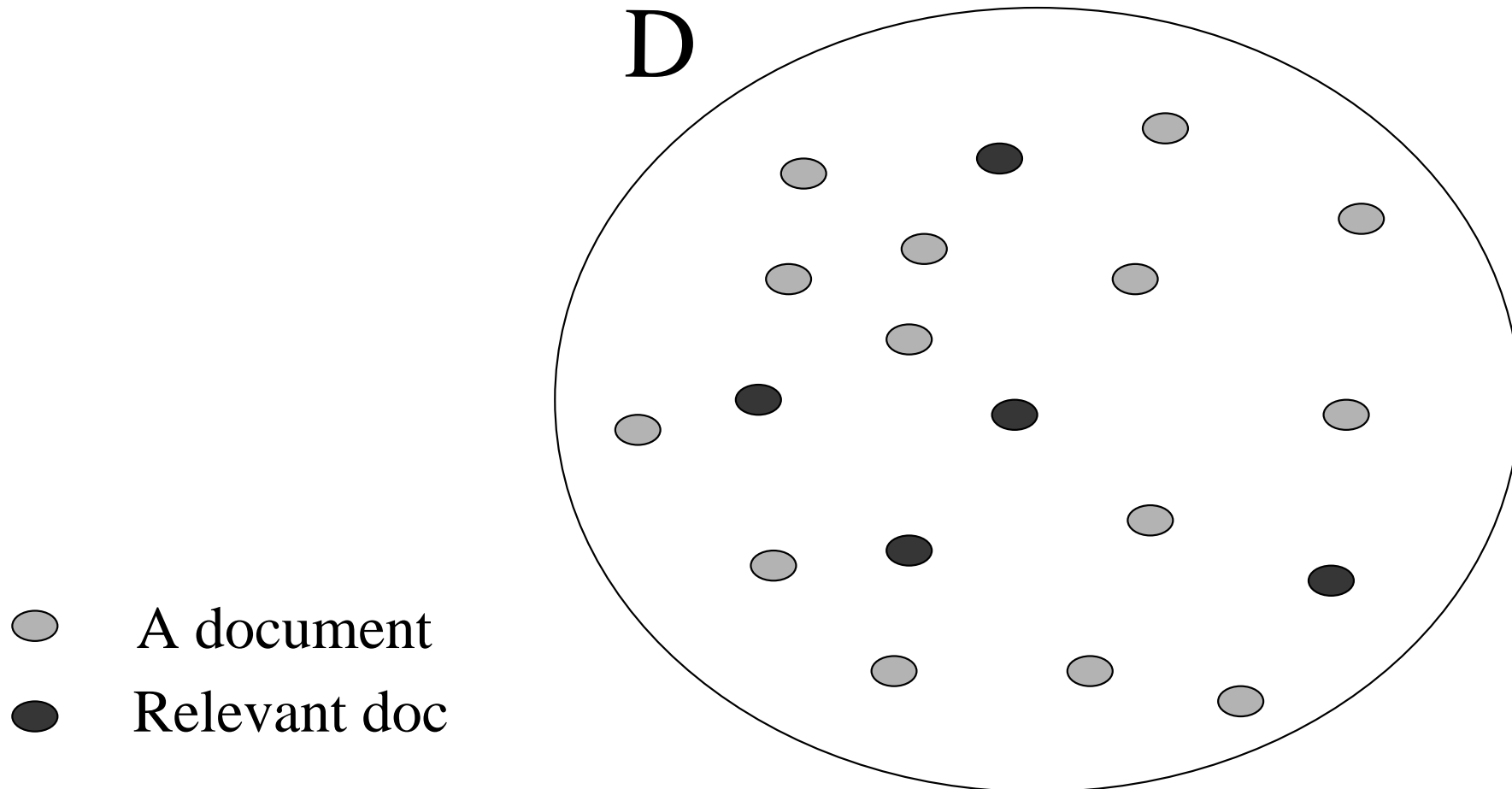
- A **relation** (binary or numeric) between documents and queries  **$R(d,q)$** .

# How to measure Document Relevance

- Although Relevance is a basic concept in IR,  
*there is still no unique definition*
- **Difficulties**
  - User dependent (experienced vs. novice)
  - Time dependent (e.g., news topics)
  - Geography dependent (Paris-FR vs. Paris-USA)
- **Simplified (binary) assumption**
  - $D$  – the set of all documents in the world,
  - $Q$ - the set of all queries in the world,
  - $R: D \times Q \rightarrow \{0,1\}$  is well defined:
    - $d$  is “relevant” to  $q$  if  $R(d,q) = 1$

# IR main task

- retrieve the “relevant” documents to the user’s query



# Index building: Text Profiling

- Both documents and queries are **profiled** to generate a **canonical** representation
- Profile is usually based on the **set of indexing units** in the text
- Indexing units are generally **representative terms** in the text
  - How to select representative terms?
  - For the moment, let's say all the words in the given document/query

In the beginning  
God created the  
heaven and the  
earth. And the  
earth was without  
form and void.

..,and,,beginning,,created,,earth,,form,,god,,heaven,,in,,the,,void,,was,,without  
.., 3 ,, 1 ,, 1 ,, 2 ,, 1 ,, 1 ,, 1 ,, 1,, 3 ,, 1 ,, 1 ,, 1

# Let us formalize a bit

- Given a collection of documents (a corpus)
  - all the terms in the collection are labeled as:  $T = \{t_1, t_2, \dots, t_N\}$ .
  - profile of a document  $d_i$  is a vector of size  $N$ :  $d_i \rightarrow (w_{i1}, w_{i2}, \dots, w_{iN})$

$$w_{ij} = \begin{cases} 0 & t_i \text{ does not appear in } d_j \\ \text{num. of appears of } t_i \text{ in } d_j & \text{otherwise} \end{cases}$$

# Index Representation - sparse matrix

| $A(t,d)$ | $d_1$    | $d_2$    | ... | $d_M$    |
|----------|----------|----------|-----|----------|
| $t_1$    | $w_{11}$ | $w_{12}$ |     | $w_{1M}$ |
| $t_2$    |          |          |     |          |
|          |          |          |     |          |
| $t_N$    | $w_{N1}$ |          |     | $w_{NM}$ |

# Using the index for Relevance Retrieval

- **Assumption:** a document not containing any query term is not relevant (*think about that! Is it true?*)
- Given a simple query of one term  $q = \{t_i\}$
- To find the relevant documents
  - 1. Retrieve all the documents  $d_j$  with  $w_{ij} > 0$
  - 2. Sort them in decreasing order
  - 3. Return the sorted list of “relevant” documents to the user
- In general: given a user’s query  $q = \{t_1, t_2, \dots, t_k\}$  return the sorted list of documents that contain **at least one** of the query terms.

# The Boolean Model

- Simple model based on **set theory**
- Queries are specified as **Boolean expressions**
  - Indexing units are words
  - Boolean Operator: OR, AND, NOT
  - Example:  
q = "java" AND "compilers" AND ("unix" OR "linux")
- **Relevance:** A document is relevant to the query if it satisfies the Boolean expression of the query.

# Boolean Model- Example

| A(t,d) | d <sub>1</sub> | d <sub>2</sub> | d <sub>3</sub> | d <sub>4</sub> | d <sub>5</sub> |
|--------|----------------|----------------|----------------|----------------|----------------|
| a      | 1              | 1              | 1              | 0              | 1              |
| b      | 0              | 1              | 0              | 1              | 1              |
| c      | 0              | 0              | 1              | 0              | 1              |

$$q = a \text{ AND } (b \text{ OR } (\text{NOT } c))$$

# Search the other way around

□ a      -> D1 , D2 , D3 , D5

□ b      -> D2 , D4 , D5

□ c      -> D3 , D5

□ a      -> 1 , 1 , 1 , 0 , 1

□ b      -> 0 , 1 , 0 , 1 , 1

□ NOT c -> 1 , 1 , 0 , 1 , 0

}  $1, 1, 0, 1, 1$  }  $1, 1, 0, 0, 1$

OR

AND

$q = a \text{ AND } (b \text{ OR } (\text{NOT } c))$

Results: d1, d2, d5

# Boolean Model – pros & cons

## □ Pros:

- Fast (bitmap vector operations)
- Binary decision (doc is “relevant” or not)

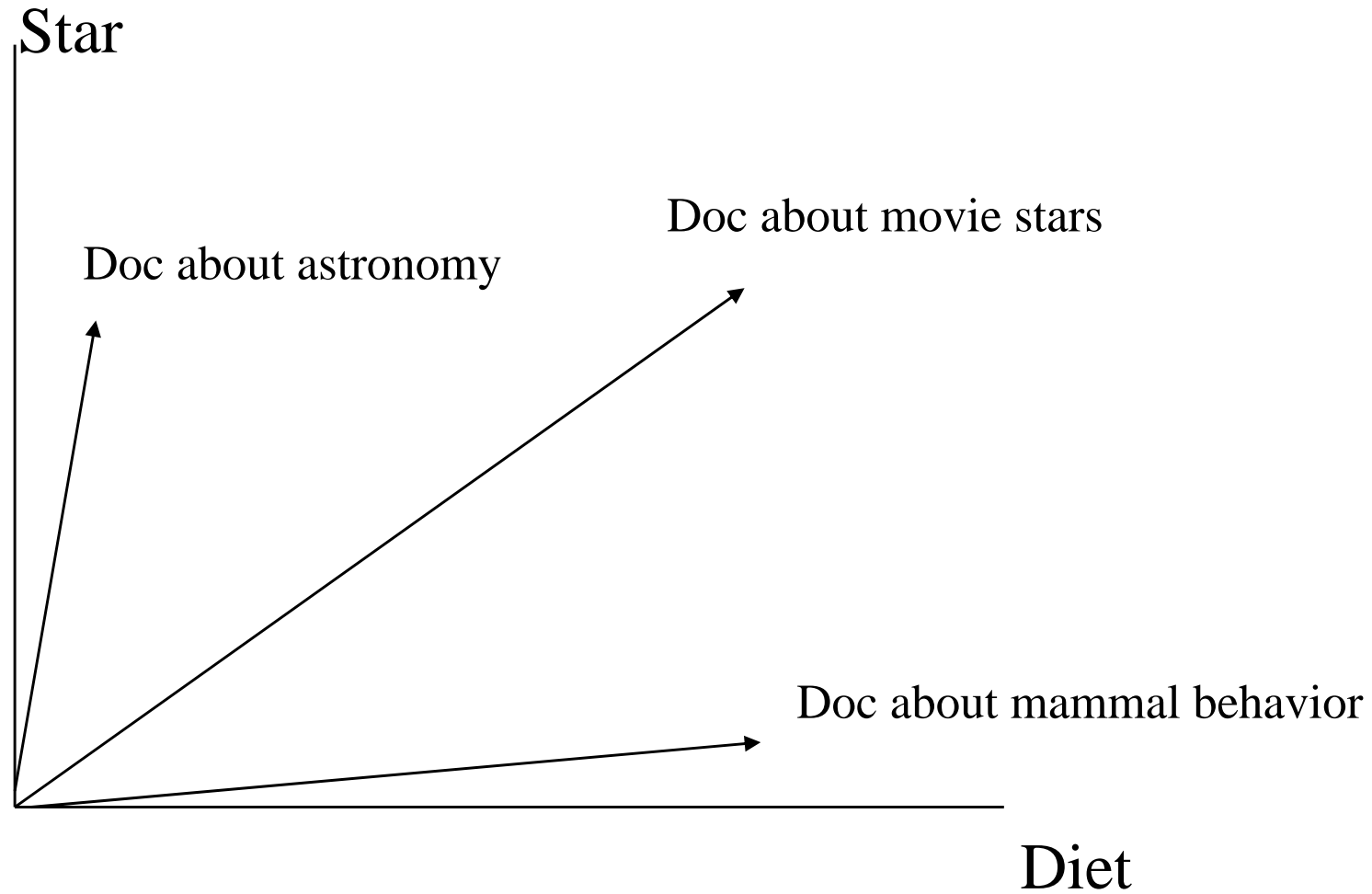
## □ Cons:

- Binary decision – How to rank?
  - Query can fail (no result)
  - Who speaks Boolean?
- Still very popular by commercial IR systems

# Vector Model

- Documents are represented as vectors in a N-dimensional space
  - N is the number of terms in the corpus
- Query is a document like any other document
- Relevance – measured by **similarity** between vectors:
  - A document is relevant to the query if its vector is similar to the vector of the query.

# Documents as Vectors



# Documents in 3D Space

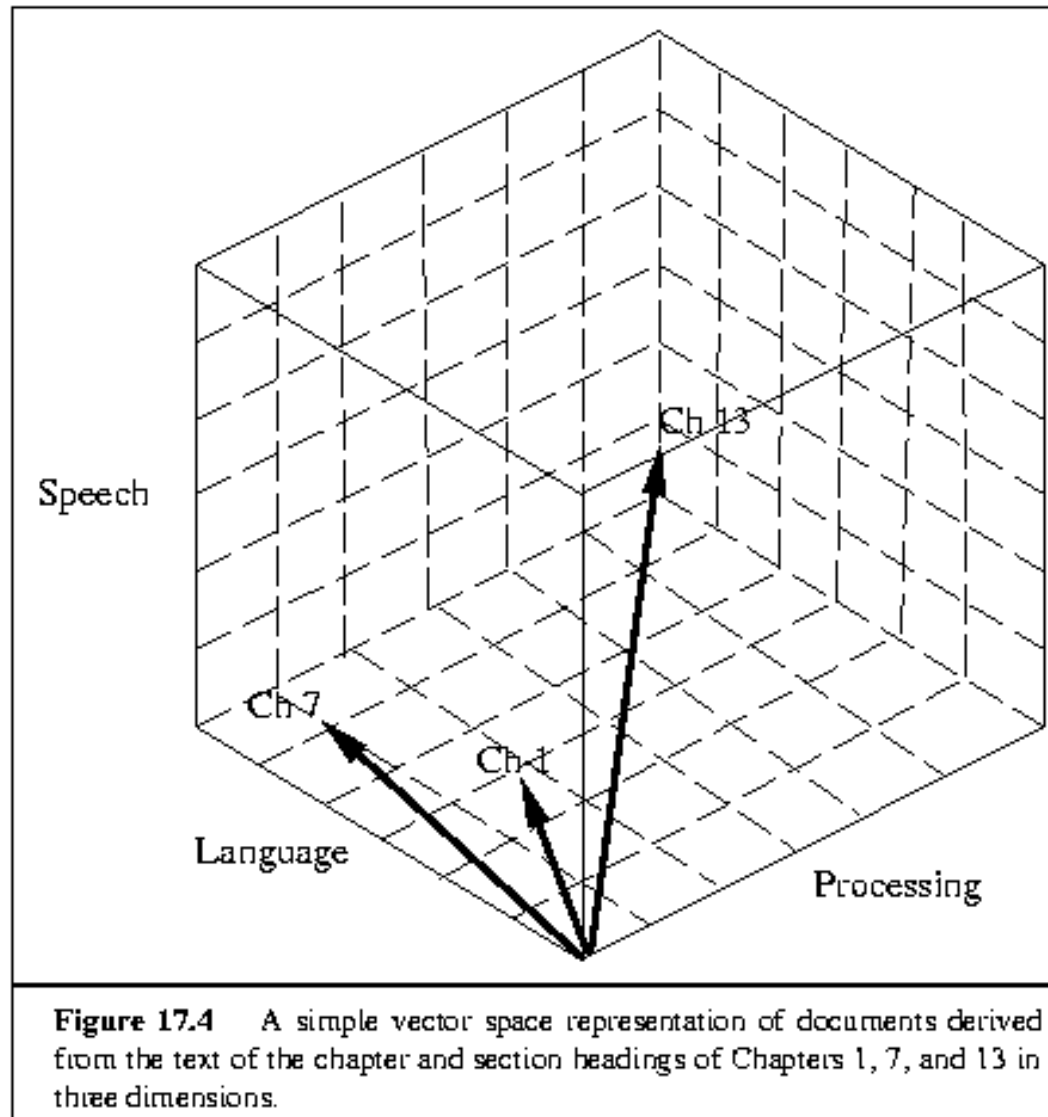
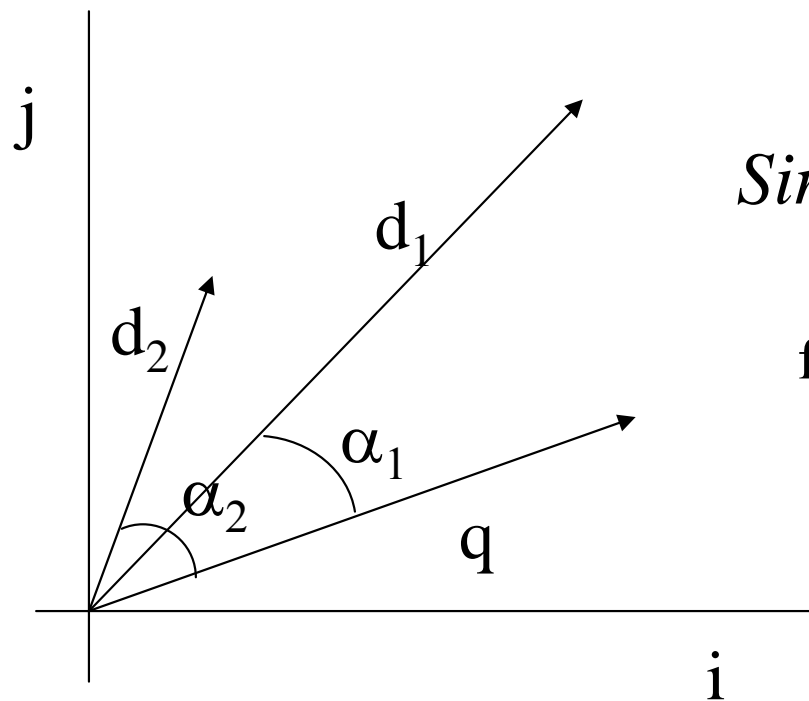


Illustration from Jurafsky & Martin

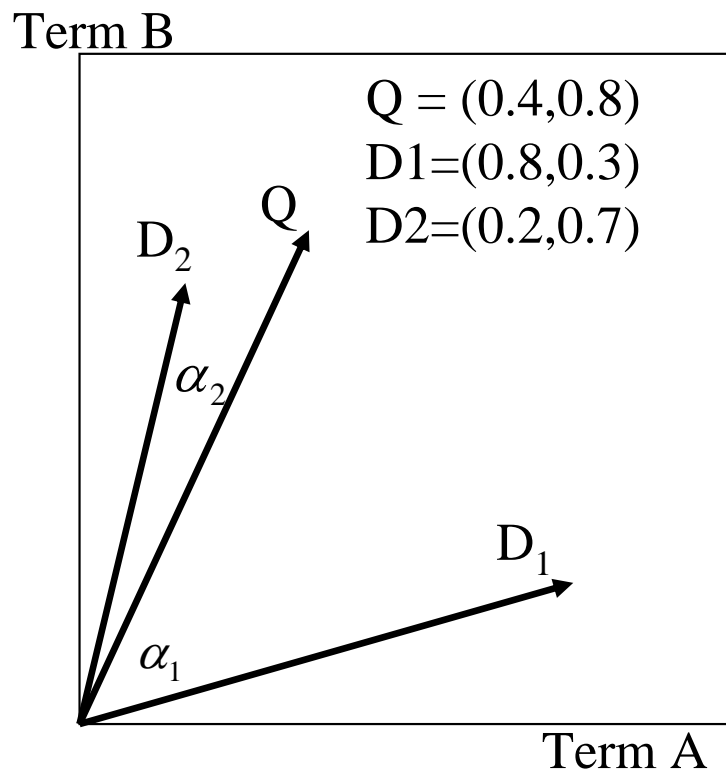
# Vector-space Model

- Represent the query as vector in the same document vector-space
- Relevance is measured by similarity
- Measure the cosines of the angle between doc-vectors and the query vector



$$\begin{aligned} \text{Sim}(q, d) &= \frac{q \bullet d}{|q| \bullet |d|} = \frac{\sum_i w_{iq} w_{id}}{\sqrt{\sum_i w_{iq}^2 \sum_i w_{id}^2}} \\ \text{for } N=2 &\rightarrow = \frac{q_x d_x + q_y d_y}{\sqrt{(q_x^2 + q_y^2)(d_x^2 + d_y^2)}} \end{aligned}$$

# Example



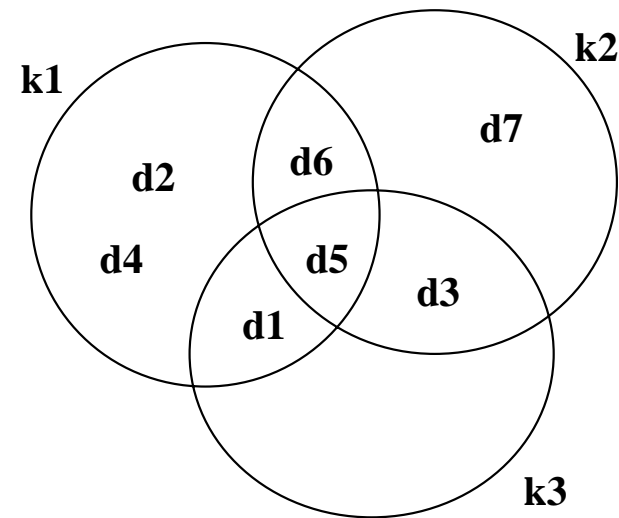
$$Sim(q, d) = \frac{q \bullet d}{|q| \bullet |d|} = \frac{\sum_i w_{iq} w_{id}}{\sqrt{\sum_i w_{iq}^2 \sum_i w_{id}^2}}$$
$$sim(q, d_1) = \frac{(0.4 \cdot 0.8) + (0.8 \cdot 0.3)}{\sqrt{[(0.4)^2 + (0.8)^2] \cdot [(0.8)^2 + (0.3)^2]}}$$

$$= \frac{0.56}{\sqrt{0.58}} = 0.74$$

$$sim(q, d_2) = \frac{(0.4 \cdot 0.2) + (0.8 \cdot 0.7)}{\sqrt{[(0.4)^2 + (0.8)^2] \cdot [(0.2)^2 + (0.7)^2]}}$$

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

# Example 2



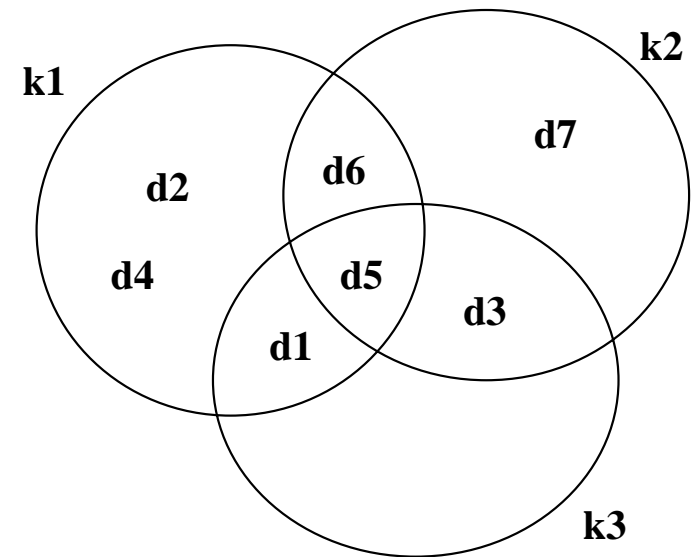
Query:  $(k_1 \ k_2 \ k_3) = (1 \ 1 \ 1)$

|           | <b>k1</b> | <b>k2</b> | <b>k3</b> | <b>q • d<sub>j</sub></b> |
|-----------|-----------|-----------|-----------|--------------------------|
| <b>d1</b> | 1         | 0         | 1         | <b>2</b>                 |
| <b>d2</b> | 1         | 0         | 0         | <b>1</b>                 |
| <b>d3</b> | 0         | 1         | 1         | <b>2</b>                 |
| <b>d4</b> | 1         | 0         | 0         | <b>1</b>                 |
| <b>d5</b> | 1         | 1         | 1         | <b>3</b>                 |
| <b>d6</b> | 1         | 1         | 0         | <b>2</b>                 |
| <b>d7</b> | 0         | 1         | 0         | <b>1</b>                 |
|           |           |           |           |                          |
| <b>q</b>  | 1         | 1         | 1         |                          |

Binary weights for the sake of the simplicity

# Example 3

Query: (1 2 3)



|           | <b>k1</b> | <b>k2</b> | <b>k3</b> | <b>q • dj</b> |
|-----------|-----------|-----------|-----------|---------------|
| <b>d1</b> | 1         | 0         | 1         | <b>4</b>      |
| <b>d2</b> | 1         | 0         | 0         | <b>1</b>      |
| <b>d3</b> | 0         | 1         | 1         | <b>5</b>      |
| <b>d4</b> | 1         | 0         | 0         | <b>1</b>      |
| <b>d5</b> | 1         | 1         | 1         | <b>6</b>      |
| <b>d6</b> | 1         | 1         | 0         | <b>3</b>      |
| <b>d7</b> | 0         | 1         | 0         | <b>2</b>      |
|           |           |           |           |               |
| <b>q</b>  | 1         | 2         | 3         |               |

# How to determine the $w(t,d)$ weights?

## □ **Binary weights:**

- $w_{ij} = 1$  if document  $d_j$  contains term  $t_i$ , otherwise 0.
- Actually it is the Boolean model

## □ **Term Frequency (tf):**

- $w_{ij} =$  (number of occurrences of  $t_i$  in  $d_j$ )

## □ **Inverse Document Frequency (idf):**

- E.g.,  $q =$  "galaxy in space".
- Is the occurrence of the query term "in" in a document should contribute the same as the occurrence of the query term "galaxy"?

# How to determine the $w(t,d)$ weights?

- tf \* idf weighting scheme (Salton 73)
  - tf – **term frequency**: a monotonic function of the term frequency in the document,
    - e.g.,  $tf(t,d) = \log(\text{freq}(t,d) + 1)$
    - e.g.,  $tf(t,d) = \text{freq}(t,d) / (\text{Max}_{u \in d} \{\text{freq}(u,d)\})$
  - idf – the **inverse document frequency of a term**: a decreasing function of the term total frequency  $N_t =$  the number of documents in the corpus where  $t$  occurs
    - e.g.,  $idf(t) = \log(N / N_t)$
  - $w_{ij} = tf(t_i, d_j) * idf(t_i)$

# Computing TF-IDF -- An Example

- Given a document  $D$  containing terms ( $a$ ,  $b$ , and  $c$ ) with given frequencies:
  - $\text{freq}(a, D) = 3$ ,  $\text{freq}(b, D) = 2$ ,  $\text{freq}(c, D) = 1$
- Assume collection contains 10,000 documents and the term total frequencies of these terms are:
  - $N_a = 50$ ,  $N_b = 1300$ ,  $N_c = 250$
- Then:
  - $a$ :  $\text{tf} = 3/3$ ;  $\text{idf} = \log(10.000/50) = 5.3$ ;  $\text{tf-idf} = 5.3$
  - $b$ :  $\text{tf} = 2/3$ ;  $\text{idf} = \log(10.000/1300) = 2.0$ ;  $\text{tf-idf} = 1.3$
  - $c$ :  $\text{tf} = 1/3$ ;  $\text{idf} = \log(10.000/250) = 3.7$ ;  $\text{tf-idf} = 1.2$

# Vector-Space pros & cons

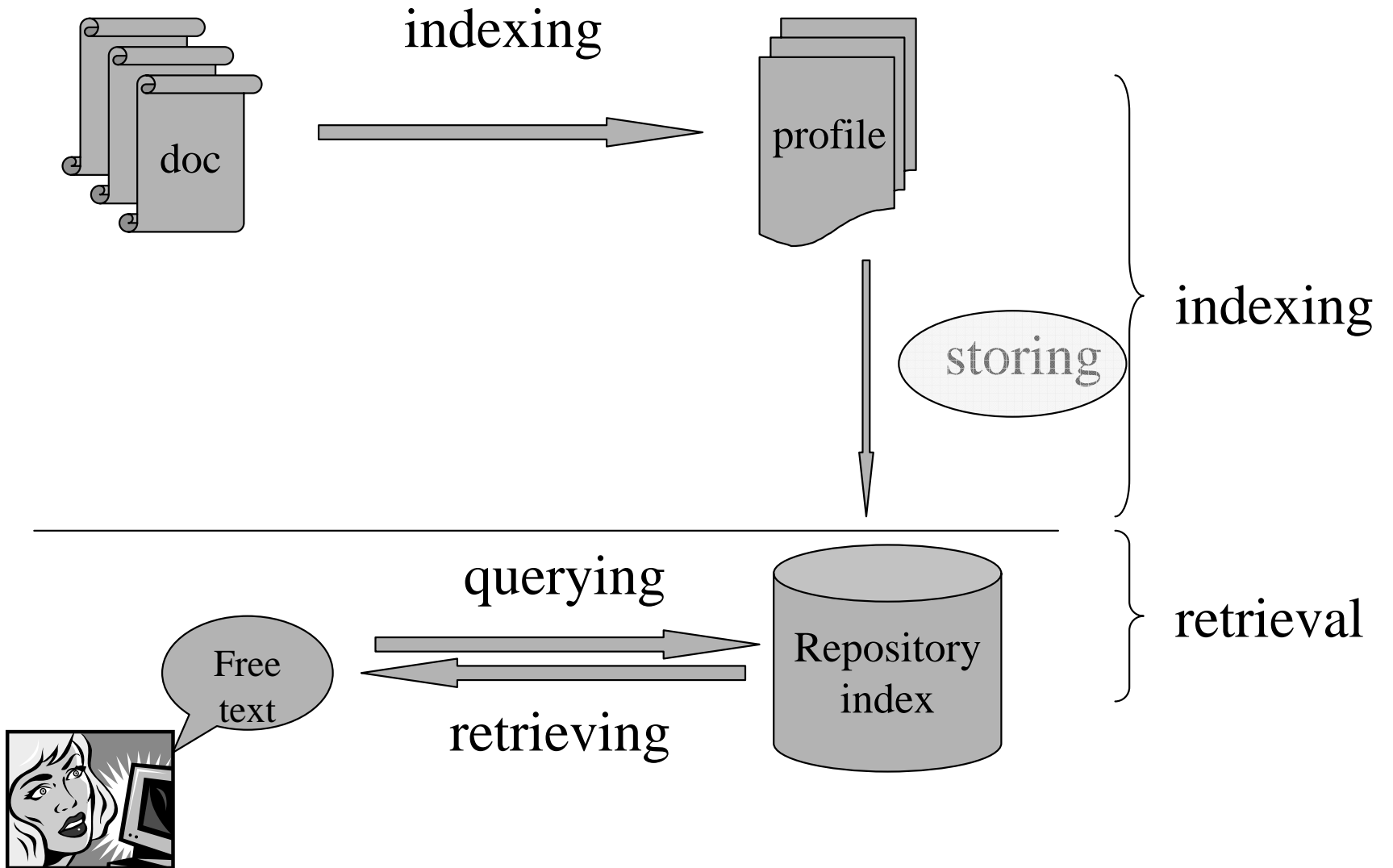
## □ Pros:

- Terms weighting scheme improves retrieval effectiveness
- Approximate query matching
- Cosine similarity is a good ranking measure

## □ Cons:

- Terms are not independent of all other terms
  - Normalization makes incrementally difficult as recomputation is needed for each added term
- Considered superior to other models due to its simplicity and elegance.

# Reminder



# Inverted Index

- ▣ Purpose: to support efficient retrieval, avoid scanning the entire collection sequentially
- ▣ Retrieval should be proportional to the size of the query rather than to size of collection.

Inverted Index

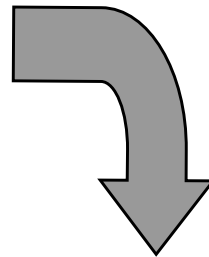
# Inverted Index

- This is the primary data structure for text indexes
- Main Idea:
  - **Invert documents descriptions into a big index**
- Basic steps:
  - Make a “dictionary” of all the tokens in the collection
  - For each token, list all the docs it occurs in

# Inverted Indexes

An Inverted File is a vector file “inverted” so that rows become columns and columns become rows

| <i>docs</i> | <i>t1</i> | <i>t2</i> | <i>t3</i> |
|-------------|-----------|-----------|-----------|
| 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         |



| <i>Terms</i> | D1 | D2 | D3 | D4 | D5 | D6 | D7 | ... |
|--------------|----|----|----|----|----|----|----|-----|
| <i>t1</i>    | 1  | 1  | 0  | 1  | 1  | 1  | 1  | 0   |
| <i>t2</i>    | 0  | 0  | 1  | 0  | 1  | 1  | 1  | 1   |
| <i>t3</i>    | 1  | 0  | 1  | 0  | 1  | 0  | 0  | 0   |

# Evaluation Criteria

## □ Effectiveness

- How precise is the answer set to the information need

## □ Efficiency

- Retrieval time, indexing time, index size

## □ Usability

- Learnability, novice use, expert use

# Effectiveness Evaluation

## □ User-centered strategy

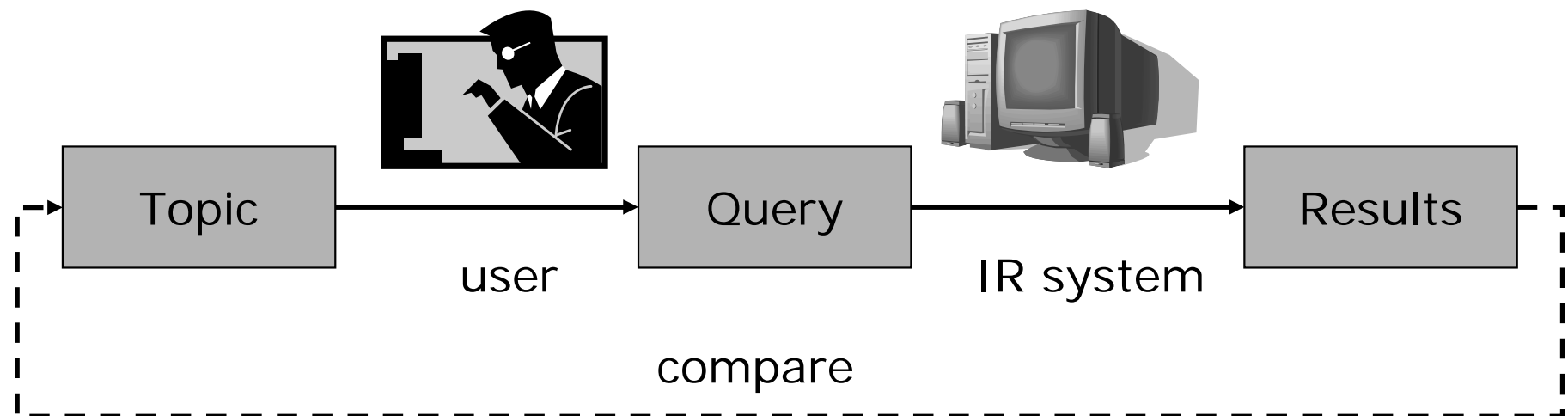
- Try several variations of the retrieval system
- Measure which variation works the “best”
- Given several users, and at least 2 retrieval systems
- Have each user try the same task on both systems (within groups)
- Or have each user try the same task on one system (between groups)
- Measure which system works the “best” (for the “average” user)

# IR Test Collection

- A Collection of Documents
  - Representative sources and quantity
- A Set of Topics
  - Used to form queries
- Relevance judgments
  - For each document, with respect to each topic
  - This is the expensive part!

# Goal of the evaluation

- Given a (generic) user and an information retrieval system
- We want to measure how good is a user, using the system, in retrieving documents relevant to a set of given topics



# Relevance Table

|                    | d1 | d2 | .... | d <sub>N</sub> |
|--------------------|----|----|------|----------------|
| topic1             | +  | -  |      | +              |
| topic2             | -  | -  |      | -              |
|                    |    |    |      |                |
| topic <sub>M</sub> | -  | +  |      | +              |

# Traditional Measures

- Precision
  - How much of what was found is relevant?
    - Particularly for interactive searching
- Recall
  - How much of what is relevant was found?
    - Particularly important for law and patent searches
- Fallout
  - How much of what was irrelevant was rejected?
    - Useful when different size collections are compared

# The Contingency Table

| Action<br>Doc | Retrieved            | Not Retrieved       |
|---------------|----------------------|---------------------|
| Relevant      | Relevant Retrieved   | Relevant Rejected   |
| Not relevant  | Irrelevant Retrieved | Irrelevant Rejected |

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

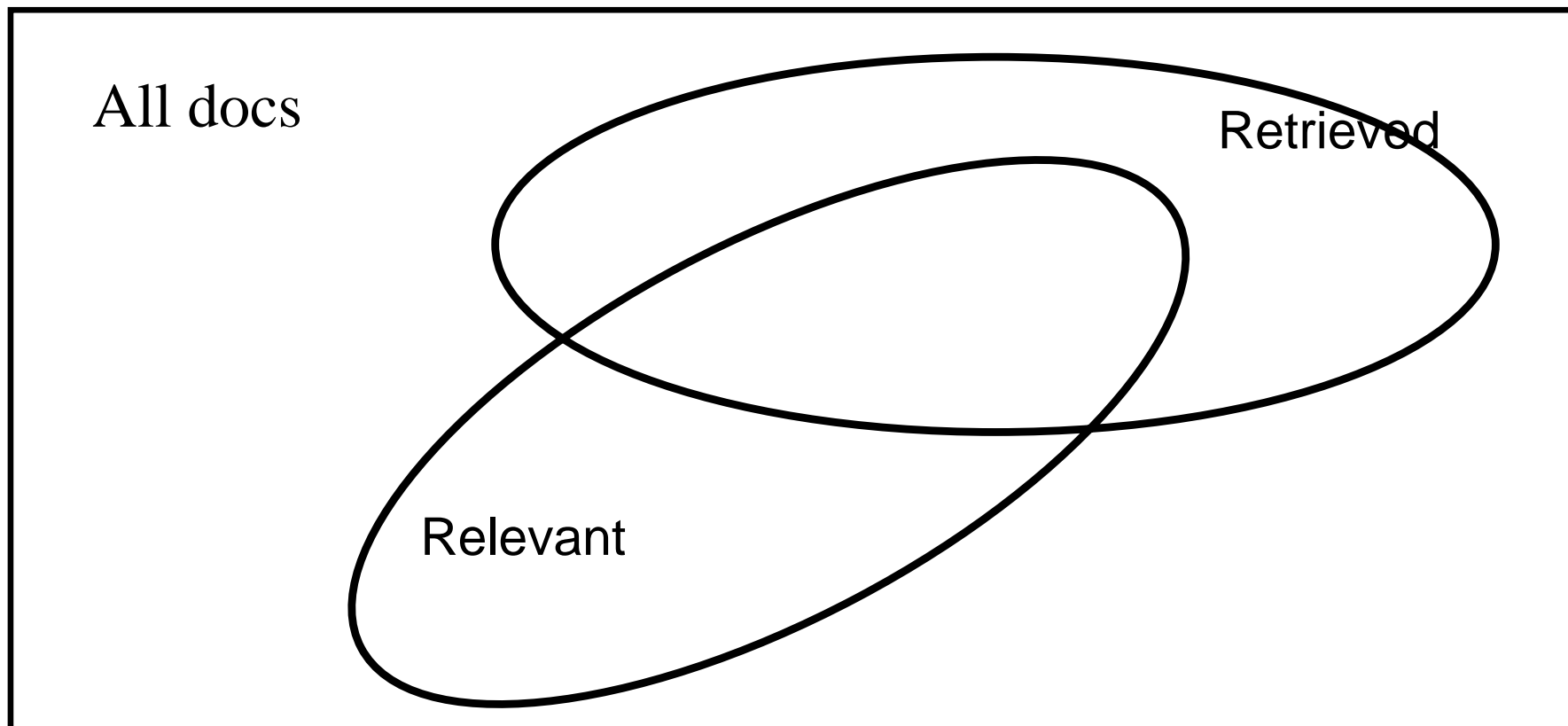
$$\text{Recall} = \frac{\text{Relevant Retrieved}}{\text{Relevant}}$$

$$\text{Fallout} = \frac{\text{Irrelevant Retrieved}}{\text{Not Relevant}}$$

# Precision vs. Recall

$$\text{Precision} = \frac{|\text{RelRetrieved}|}{|\text{Retrieved}|}$$

$$\text{Recall} = \frac{|\text{RelRetrieved}|}{|\text{Rel in Collection}|}$$



# Why Precision and Recall?

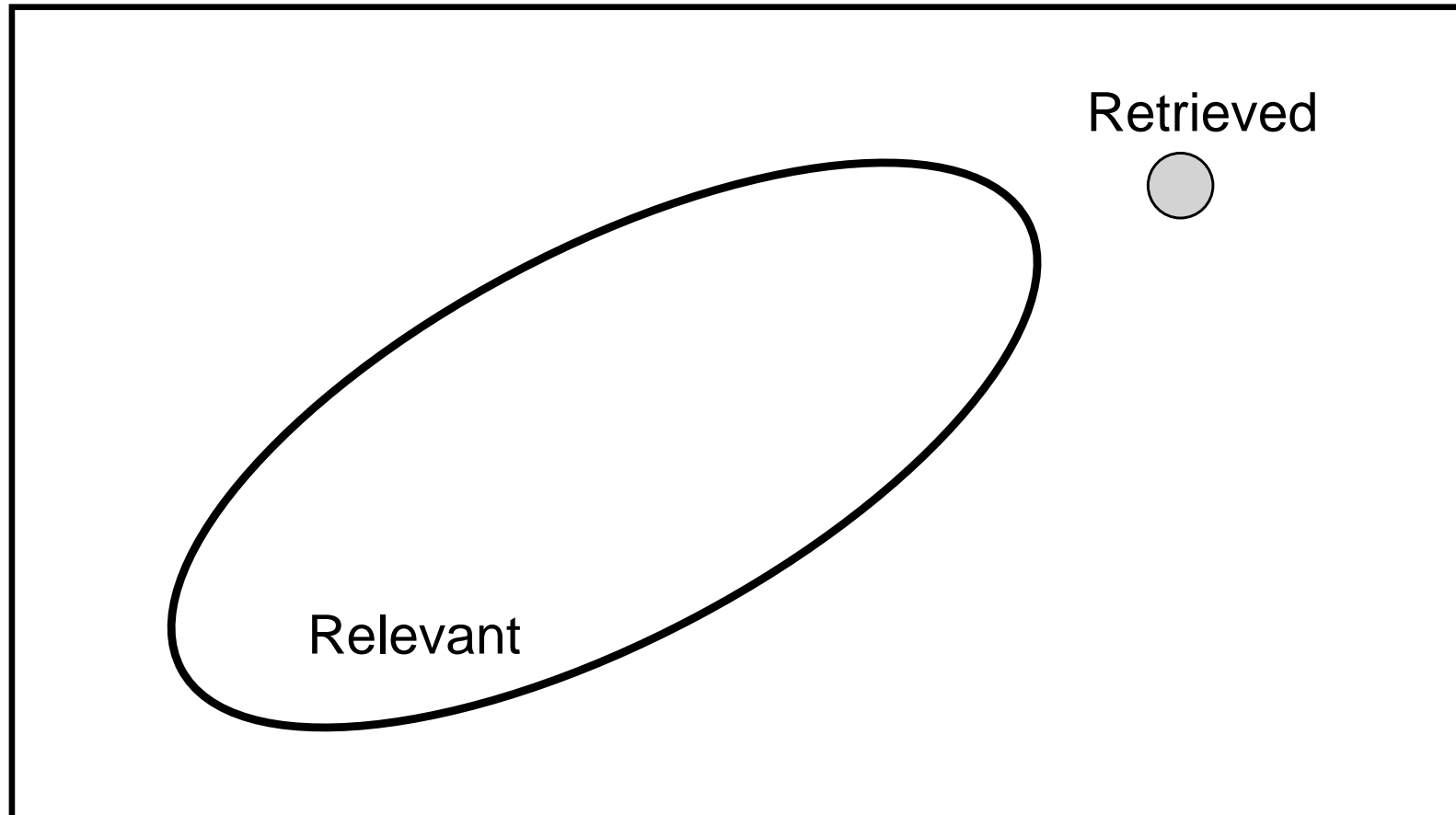
Get as much good stuff while at the same time getting as little junk as possible.

High Recall – all the truth

High Precision – nothing but the truth

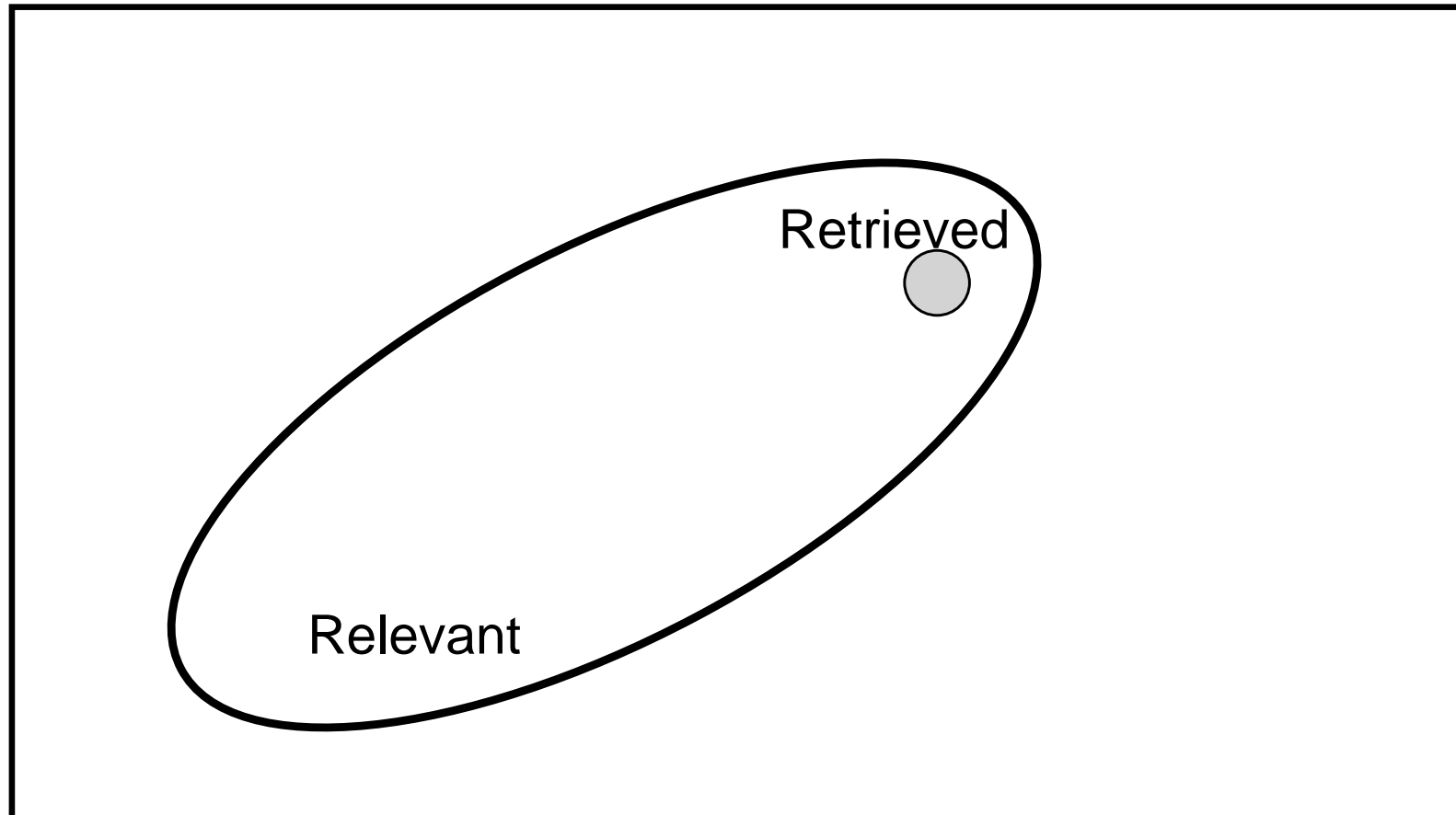
# Retrieved vs. Relevant Documents

Very low precision, very low recall (0 in fact)



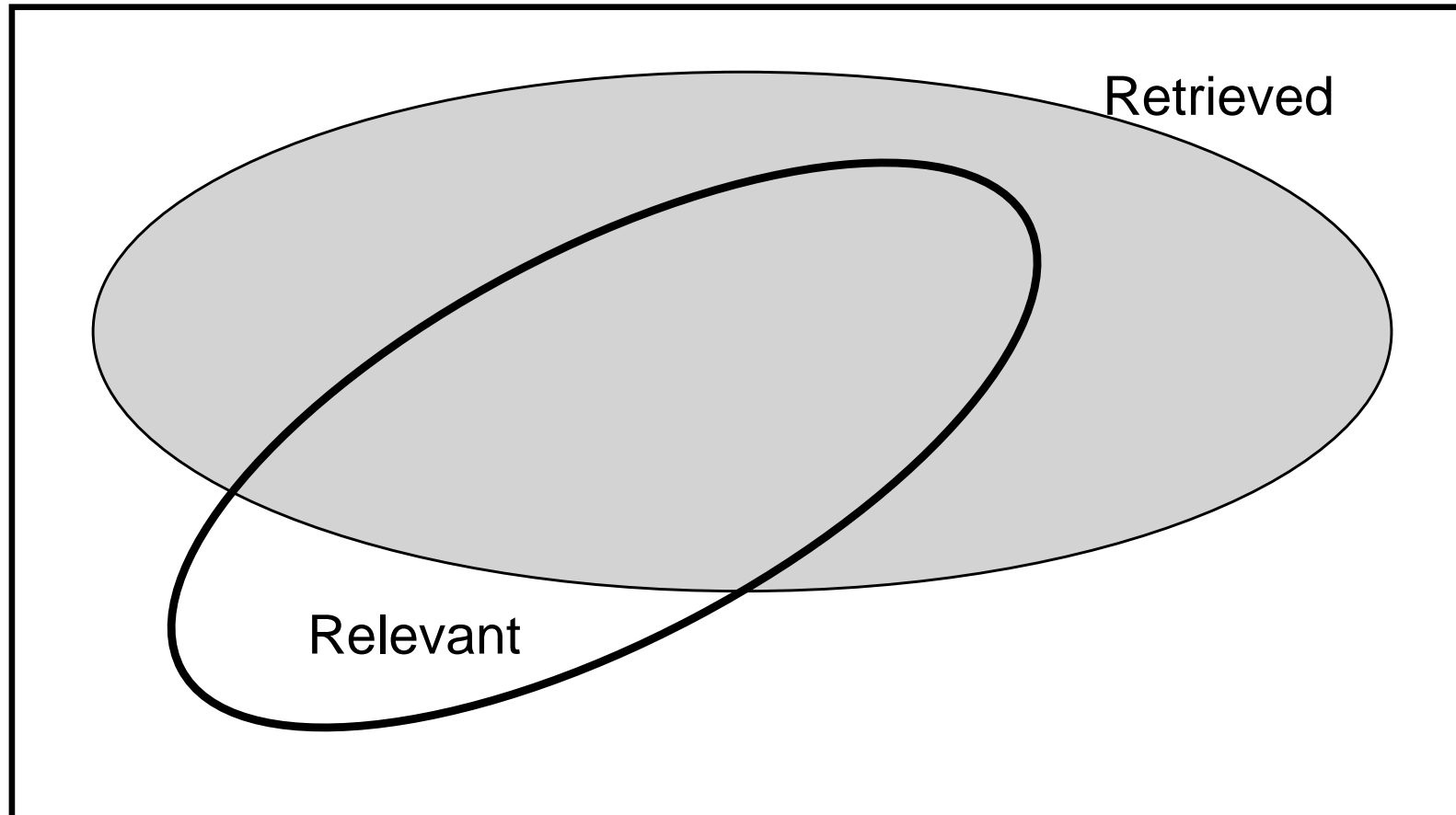
# Retrieved vs. Relevant Documents

Very high precision, very low recall



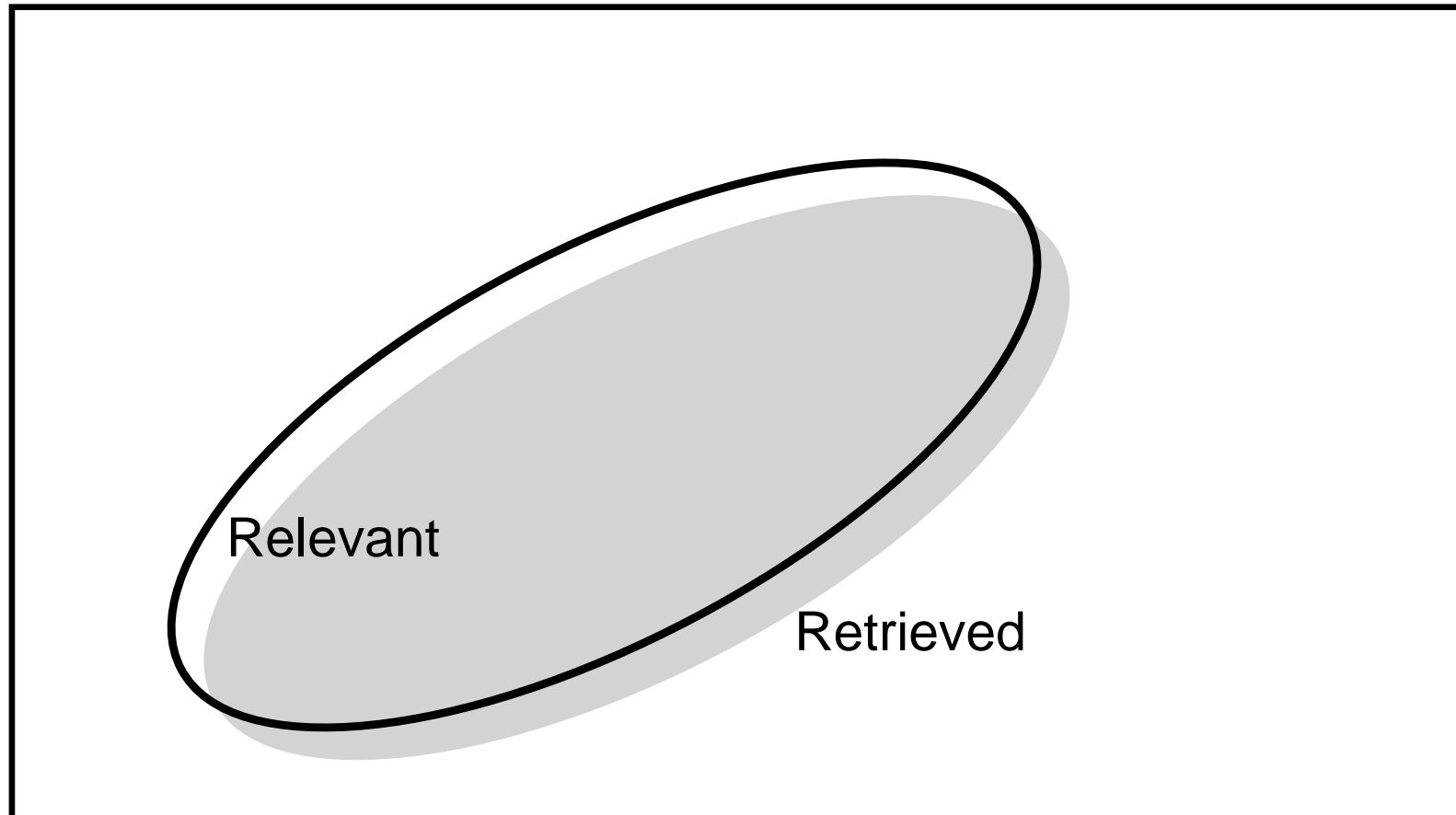
# Retrieved vs. Relevant Documents

High recall, low precision

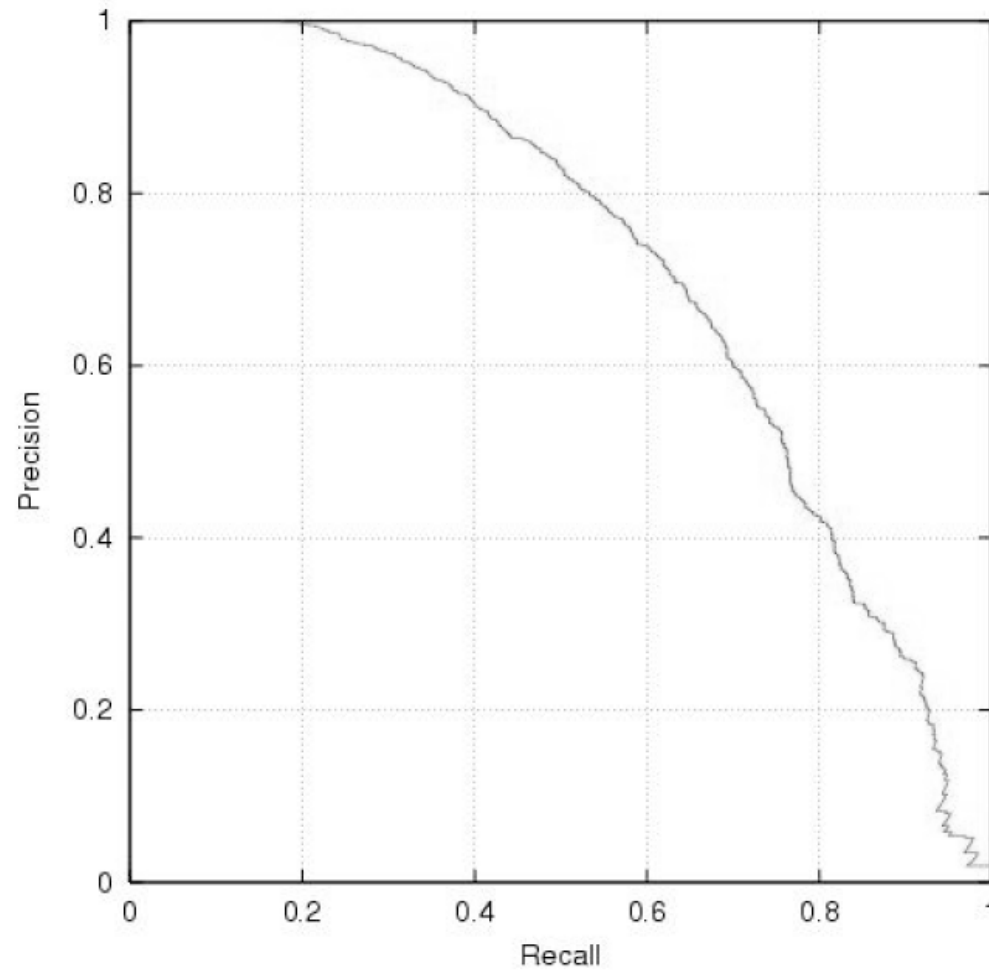


# Retrieved vs. Relevant Documents

High precision, high recall (at last!)



# Precision and recall



A typical precision and recall curve