Corso di Biblioteche Digitali

- Vittore Casarosa
  - casarosa@isti.cnr.it
  - Office: 050 621 3115
  - Mobile: 348 397 2168
  - Skype: vittore1201
- “Ricevimento” at the end of the lessons or by appointment
- Final assessment
  - 70% oral examination
  - 30% project (development of a small digital library))
- Reference material:
  - Ian Witten, David Bainbridge, David Nichols, How to build a Digital Library, Morgan Kaufmann, 2010, ISBN 978-0-12-374857-7 (Second edition)
  - Material provided by the teacher
- **http://cloudone.isti.cnr.it/casarosa/BDG/**

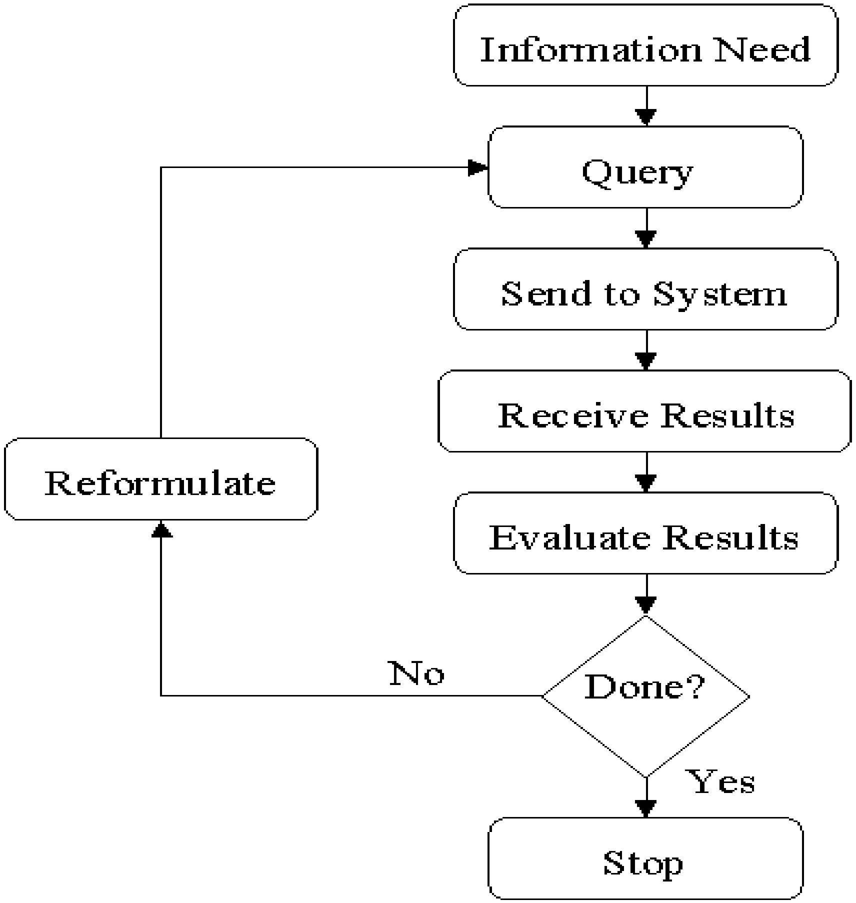
Modules

- Computer Fundamentals and Networking
- A conceptual model for Digital Libraries
- Bibliographic records and metadata
- Information Retrieval and Search Engines
- Knowledge representation
- Digital Libraries and the Web
- Hands-on laboratory: the Greenstone system

Information Retrieval

- Information Retrieval and Search Engines
  - – Indexing a collection of documents
  - – Ranking query results ←
  - – Search engines in the Web
  - – Ranking in Web search engines

Retrieval Interaction Model



IS 240 – Stanford University

Query execution

- The execution of a query usually depend on the underlying “data model”
  - Structured data
  - Semi-structured data
  - Unstructured data
- It also depends on whether we want
  - an **exact match** between the query terms and the documents (**boolean query**) or
  - a “**relevance-based retrieval**” (**non Boolean query**)
- **Ranking** of the result is important in both cases (there can be a high number of results)

Structured data (Boolean query)

- Structured data tends to refer to information in “tables”

Employee	Manager	Salary
Smith	Jones	50000
Chang	Smith	60000
Ivy	Smith	50000

Typically allows query with numerical range and text exact match (see relational DB and SQL).
Salary < 60000 AND Manager = Smith.

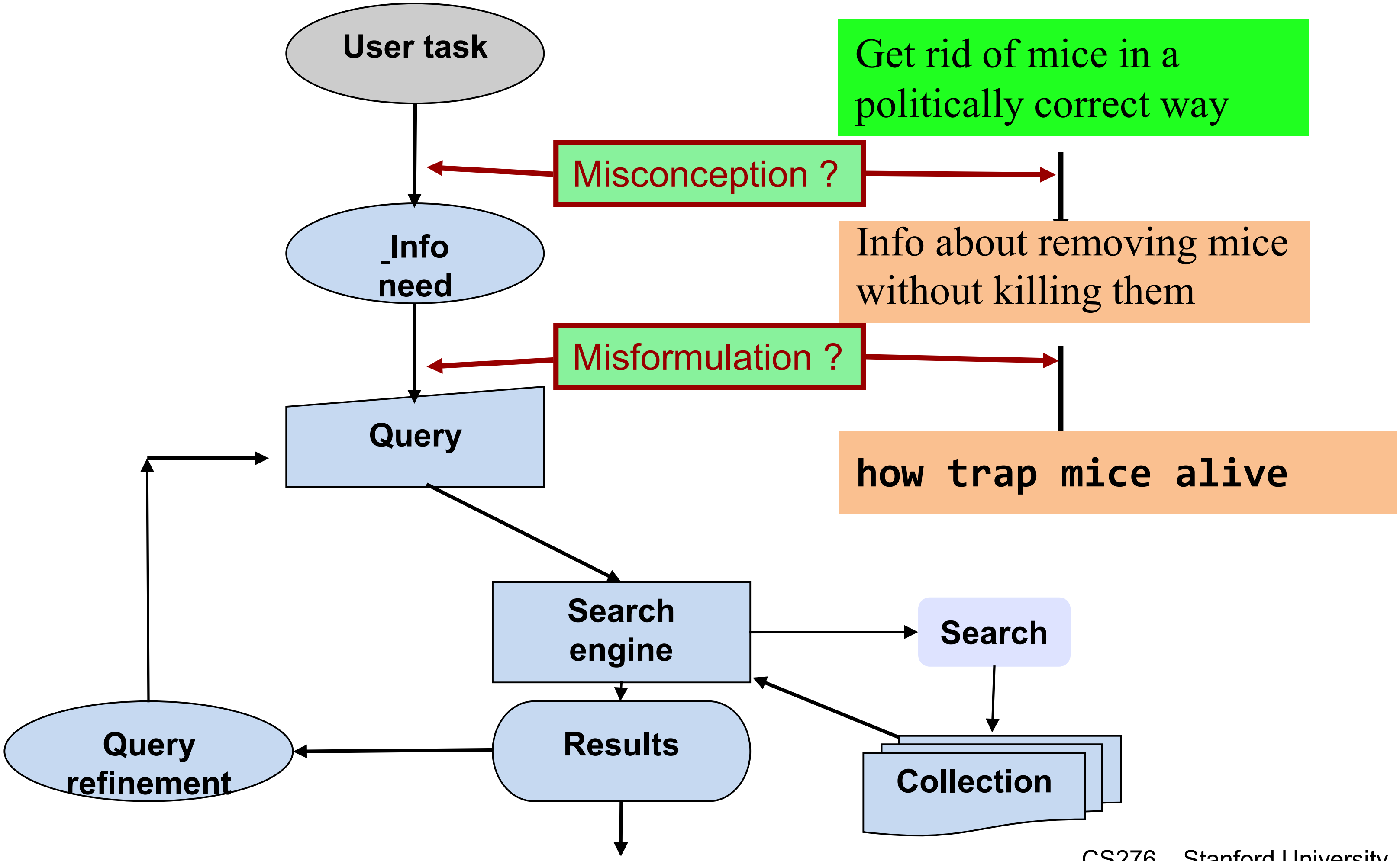
Semi-structured data (Boolean query)

- In fact almost no data is “unstructured”
- E.g., this slide has distinctly identified zones such as the *Title* and *Bullets*
- Facilitates “semi-structured” search (often called “fielded search”)
  - *Title* contains data AND *Bullets* contain search
  - Possibility to consider also the linguistic structure (i.e. subject, verb, etc.)
- Typical of searching in a catalog (OPAC)

Unstructured data

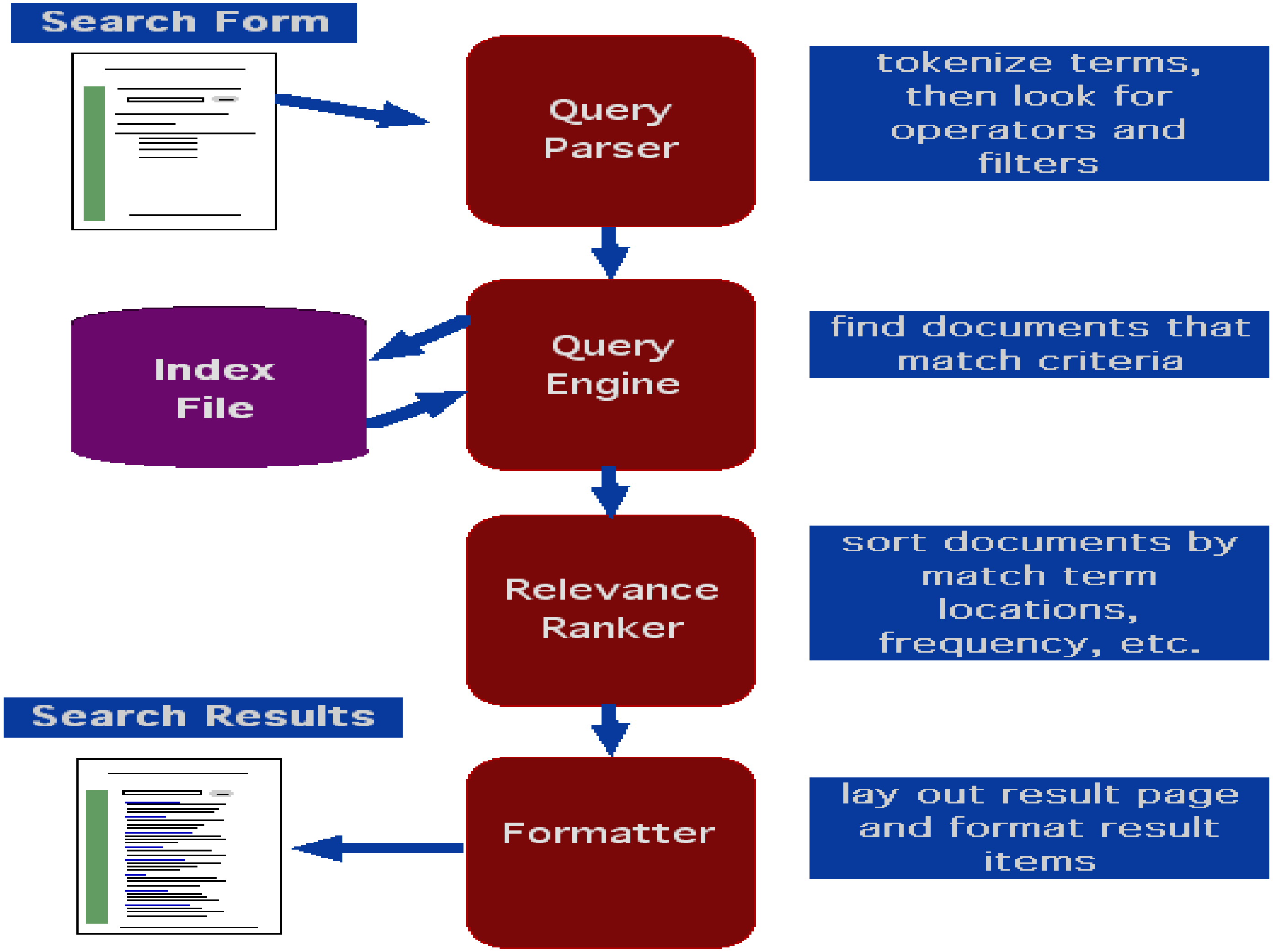
- Typically refers to free text retrieval
- Allows
  - Keyword (search terms) queries, possibly including operators
  - More sophisticated “concept” queries e.g.,
    - find all documents dealing with *drug abuse*
- Classic model for searching text documents

The Information need



CS276 – Stanford University

Components of a Search engine



Search Form

Query Parser

tokenize terms, then look for operators and filters

Index File

Query Engine

find documents that match criteria

Relevance Ranker

sort documents by match term locations, frequency, etc.

Search Results

Formatter

lay out result page and format result items

Model for “free text queries” (non Boolean queries)

- The query is a sequence of “query terms” without (explicit) Boolean connectives
- Not practical to consider the AND of all the query terms (i.e. look for documents that contain ALL the query terms)
  - Some of the query terms might be missing in a document
- Not practical to consider the OR of all the query terms (i.e. look for documents that contain SOME of the query terms)
  - Too many documents would be retrieved
- Need to define a method to compute a measure of relevance (or similarity) between the query and a document
- Results will be ranked according to the **relevance measure**

Representation of documents

- The basic idea is to represent a document with the list of all the distinct words it contains, in alphabetical order (bag of words)
- The computer (as the name says) can only compute, and therefore we need to define an algorithm that takes as input the query and a document, and gives as result a "number" representing the relevance of that document for that query
- To do that we need to represent both the query and each document in some "mathematical form", so that they can be fed into the formula

Documents as vectors

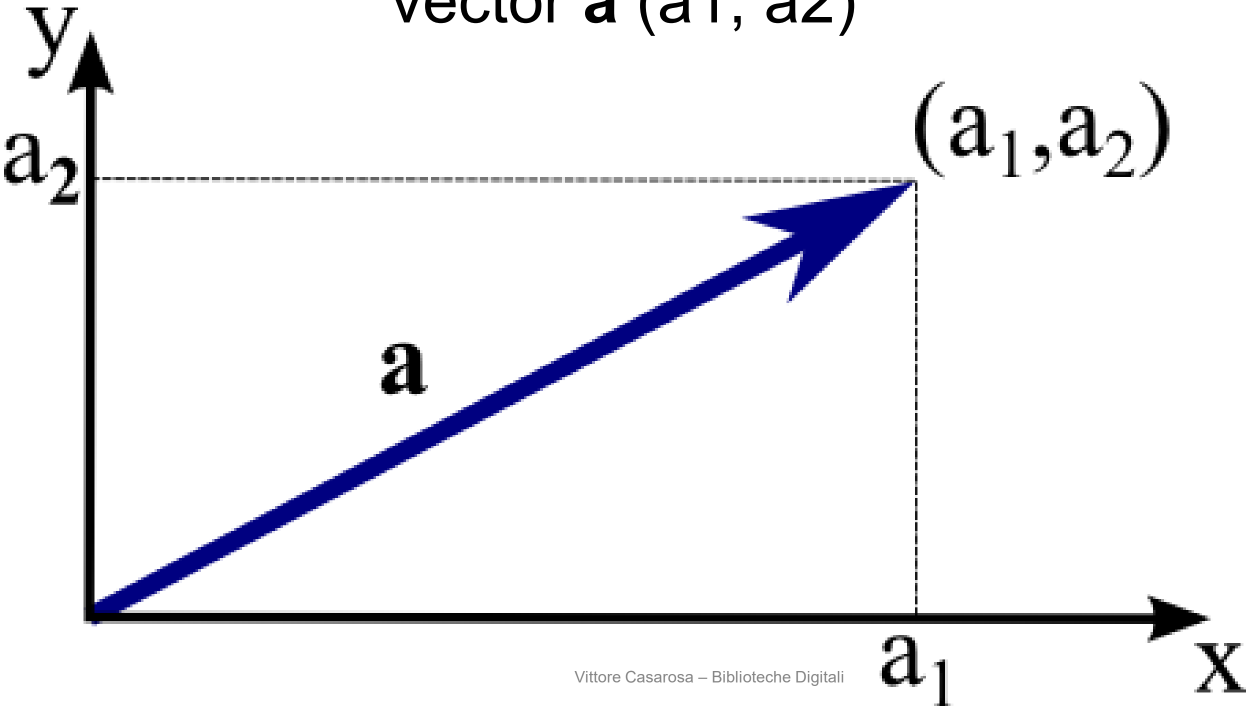
- Given the existence of the lexicon, the most “natural” (and easy) way to represent “the bag of words” in a mathematical form is a vector (a list of numbers), with as many elements as there are terms in the lexicon
- The elements of the vector are:
  - **zero** for those words of the lexicon that are not in the document
  - a number (**the weight**) for those words of the lexicon that are contained in the document

View of vectors in 2D

A vector is an **ordered list** of (n) numbers.

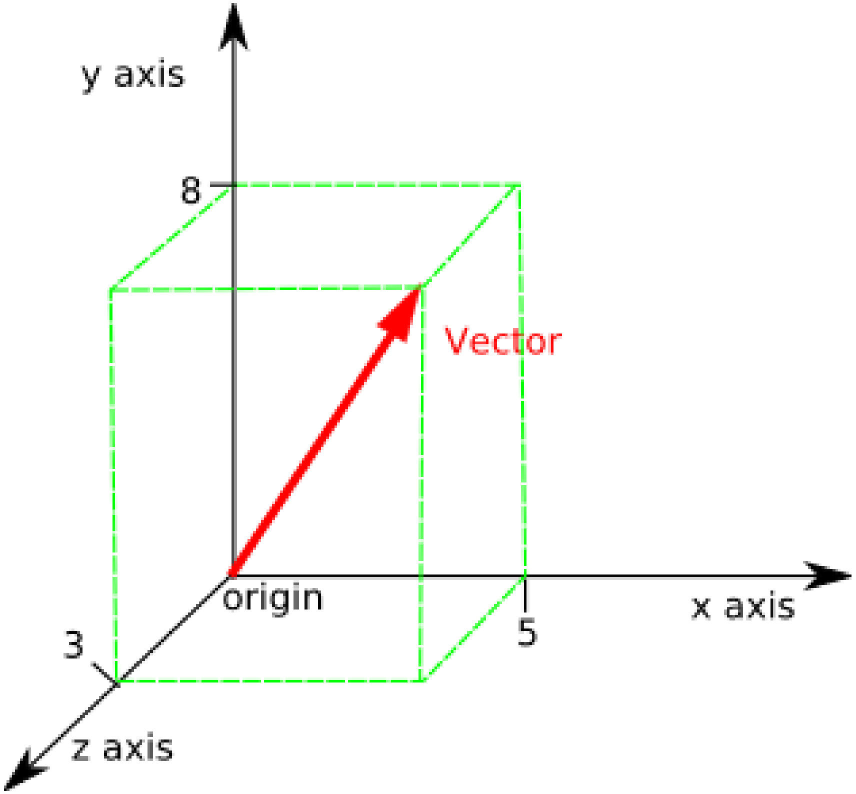
Thinking of that list of numbers as the coordinates of a point in (n-dimensional) space, a vector can be visually represented as a directed line from the origin to the point identified by the components of the vector

vector **a** (a1, a2)

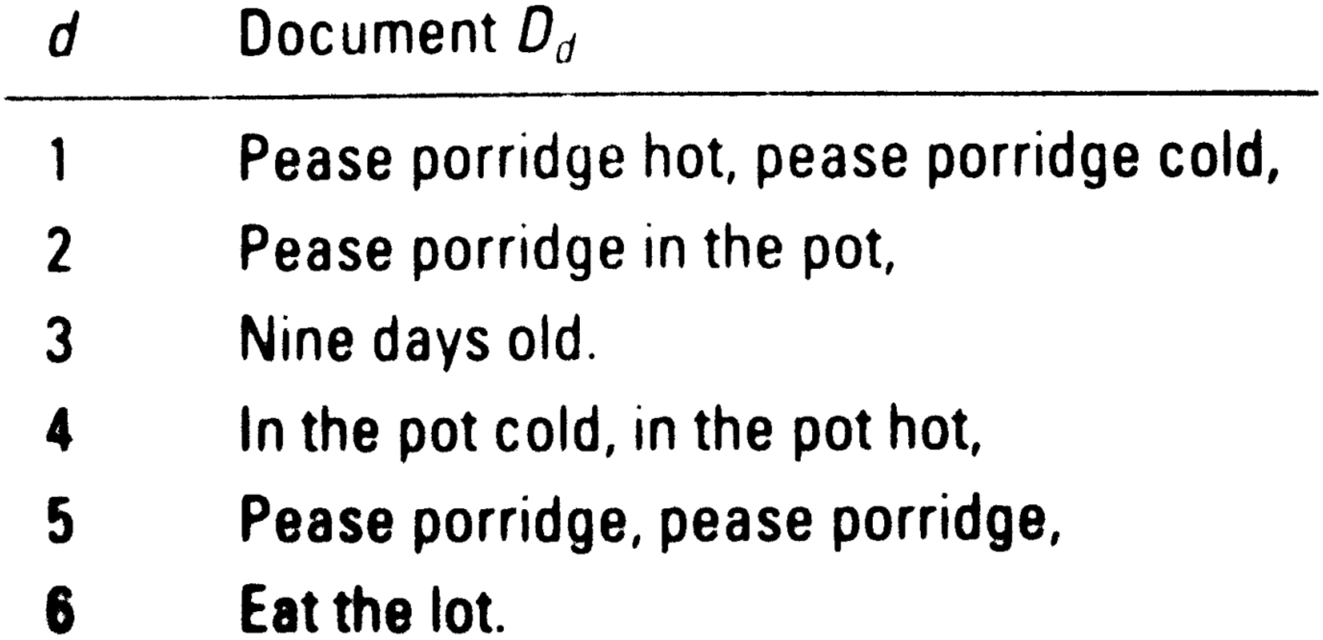


View of vectors in 3D

Vector (5,8,3)

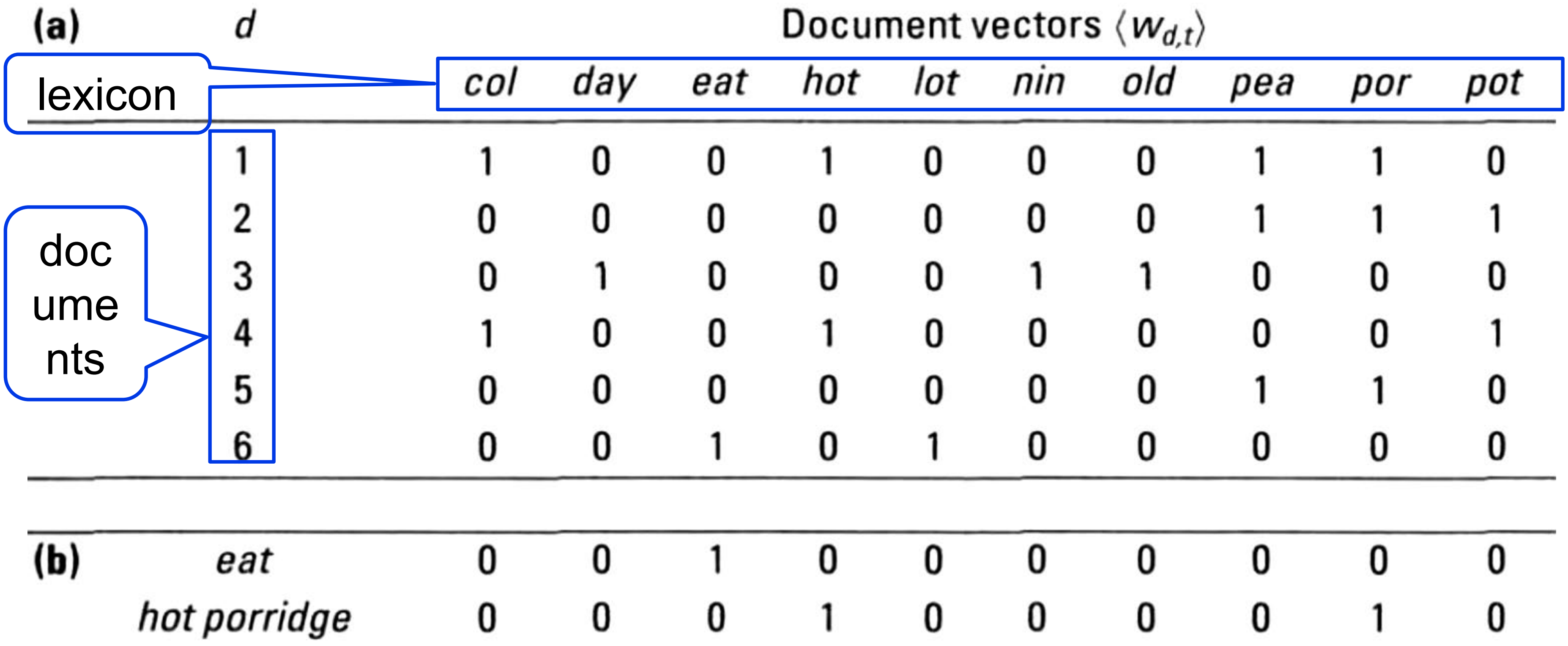


Simple example



d	Document D_d
1	Pease porridge hot, pease porridge cold,
2	Pease porridge in the pot,
3	Nine days old.
4	In the pot cold, in the pot hot,
5	Pease porridge, pease porridge,
6	Eat the lot.

Document as binary vectors



(a)	*d*	Document vectors $\langle w_{d,t} \rangle$									
		col	*day*	*eat*	*hot*	*lot*	*nin*	*old*	*pea*	*por*	*pot*
	1	1	0	0	1	0	0	0	1	1	0
	2	0	0	0	0	0	0	0	1	1	1
	3	0	1	0	0	0	1	1	0	0	0
	4	1	0	0	1	0	0	0	0	0	1
	5	0	0	0	0	0	0	0	1	1	0
	6	0	0	1	0	1	0	0	0	0	0
(b)	*eat*	0	0	1	0	0	0	0	0	0	0
	hot porridge	0	0	0	1	0	0	0	0	1	0

lexicon

doc
ume
nts

each document (and each query) is represented as a vector (a sequence) of 0's and 1's
the number of components of the vectors is equal to the size of the lexicon

Measuring relevance with vectors

- A possible (simple) measure of relevance is the inner product of the two (binary) vectors representing, respectively, the query and a given document
- The inner product of vectors is defined as “the sum of the products element by element of the components of the vector”
- X $(x_1,x_2,.....,x_n)$ • Y$(y_1,y_2,\quad ,y_n) = \sum x_i \bullet y_i$ (i = 1 to n)

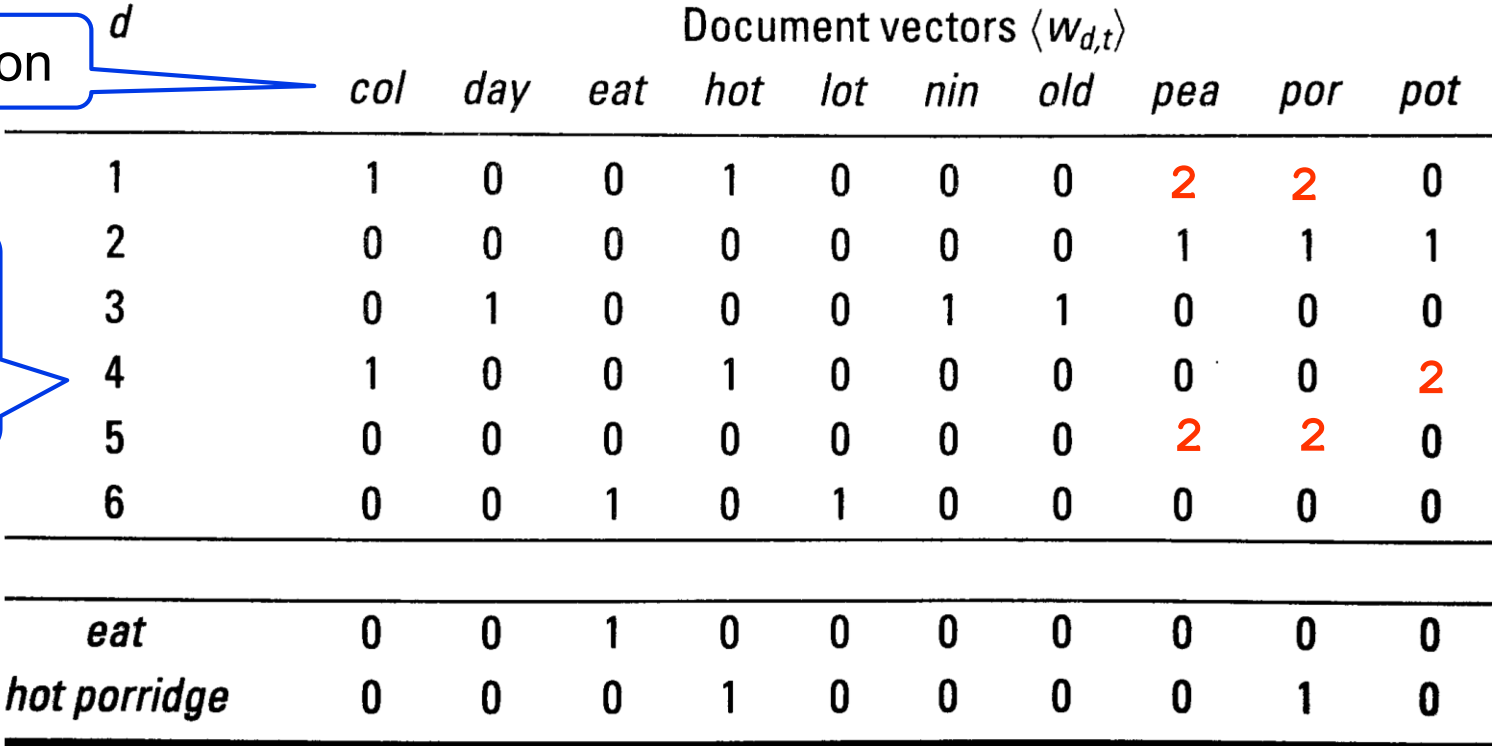
Relevance measured as inner product

- Match (eat, D_6) =
  $(0,0,1,0,0,0,0,0,0,0) \bullet (0,0,1,0,1,0,0,0,0,0) = 1$
- Match (hot porridge, D_1) =
  $(0,0,0,1,0,0,0,0,1,0) \bullet (1,0,0,1,0,0,0,1,1,0) = 2$
- In this “simple approach” the measure of relevance is just how many words of the query are present in a document
- Three drawbacks of this approach to compute relevance
  - No account of term frequency in the document (i.e. how many times a term appears in the document)
  - No account of term scarcity (in how many documents the term appears)
  - Long documents with many terms are favoured

Document as vectors of term frequency

lexicon

doc um ents



d	Document vectors $\langle w_{d,t} \rangle$									
	col	*day*	*eat*	*hot*	*lot*	*nin*	*old*	*pea*	*por*	*pot*
1	1	0	0	1	0	0	0	2	2	0
2	0	0	0	0	0	0	0	1	1	1
3	0	1	0	0	0	1	1	0	0	0
4	1	0	0	1	0	0	0	0	0	2
5	0	0	0	0	0	0	0	2	2	0
6	0	0	1	0	1	0	0	0	0	0
eat	0	0	1	0	0	0	0	0	0	0
hot porridge	0	0	0	1	0	0	0	0	1	0

each document (and each query) is now represented as a sequence (a vector) of zeros and numbers, i.e. each number is the number of occurrences of the term in the document

As before, the number of components of the vectors is equal to the size of the lexicon

Term frequency

- The first drawback can be solved using the term frequency within the document (i.e. the number of times that the term appears in the document)
- The binary (term, document) matrix becomes now a (term frequency, document) matrix
- Match (hot porridge, D_1) =

  $(0,0,0,1,0,0,0,0,1,0) \bullet (1,0,0,1,0,0,0,2,2,0) = 3$

Term weight

- More generally, we can assign to term *t* in document *d* a **weight,** that we can indicate with $w_{d,t}$
- We can also assign weights $w_{q,t}$ to the terms in the query
- In the vector representing the document (or the query) the weight of terms of the lexicon not in the document or in the query is zero
- The vectors describing the document and the query are now **vectors of weights** and their inner product gives a similarity measure of the two

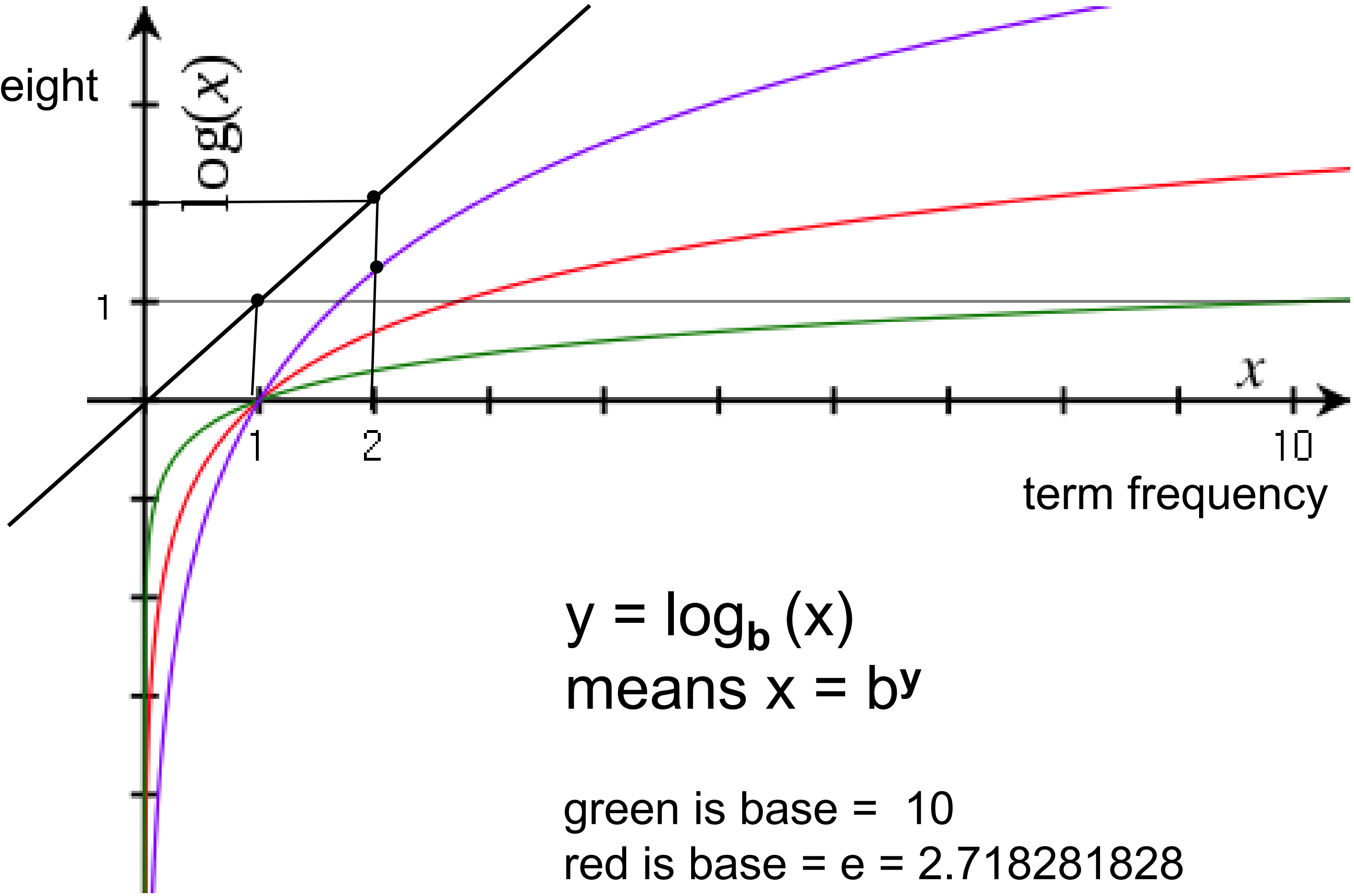
$$M(Q, D_d) = \sum_{t \in Q} w_{q,t} \cdot w_{d,t}$$

Assigning the weight to a term

- What is usually measured when building the index is the number of times that a term appear in the document (usually called term frequency, or *tf*)
- However, it is difficult to assess the relative importance of
  - 0 vs. 1 occurrence of a term in a doc
  - 1 vs. 2 occurrences
  - 2 vs. 3 occurrences …
- While it is clear that more is better, it does not necessarily mean that a lot is proportionally better than a few
- In practice, two common options for an initial term weight
  - use just the raw *tf*
  - use a logarithmic formula such as

$$wf_{t,d} = 0 \text{ if } tf_{t,d} = 0,\ 1 + \log tf_{t,d} \text{ otherwise}$$

The logarithmic curve



$y = \log_{b}(x)$

means $x = b^{y}$

green is base = 10

red is base = e = 2.718281828

purple is base = 1.7

Weighting term scarcity

- The way to take into account the scarcity (or abundance) of a term in a collection is to count the number of documents in which a term appears (usually called the term **document frequency**) and then to consider its inverse (i.e. the inverse document frequency, or *idf*)
  - measure of informativeness of a term: its rarity across the whole corpus
  - could just be the inverse of the raw count of number of documents the term occurs in ($idf_i = 1/df_i$)
  - the most commonly used version is:

$$idf_i = \log\left(\frac{n}{df_i}\right)$$

n is the number of documents in the collection

Final weight: tf x idf (or tf.idf)

- In conclusion, the weight of each term *i* in each document *d* ($w_{i,d}$) is usually given by the following formula (or very similar variations), called the *tf.idf* weight

$$w_{i,d} = tf_{i,d} \times \log(n / df_i)$$

$$tf_{i,d} = \text{frequency of term } i \text{ in document } d$$

$$n = \text{total number of documents}$$

$$df_i = \text{the number of documents that contain term } i$$

- Increases with the number of occurrences *within* a doc
- Increases with the rarity of the term *across* the whole corpus

Vectors of weights

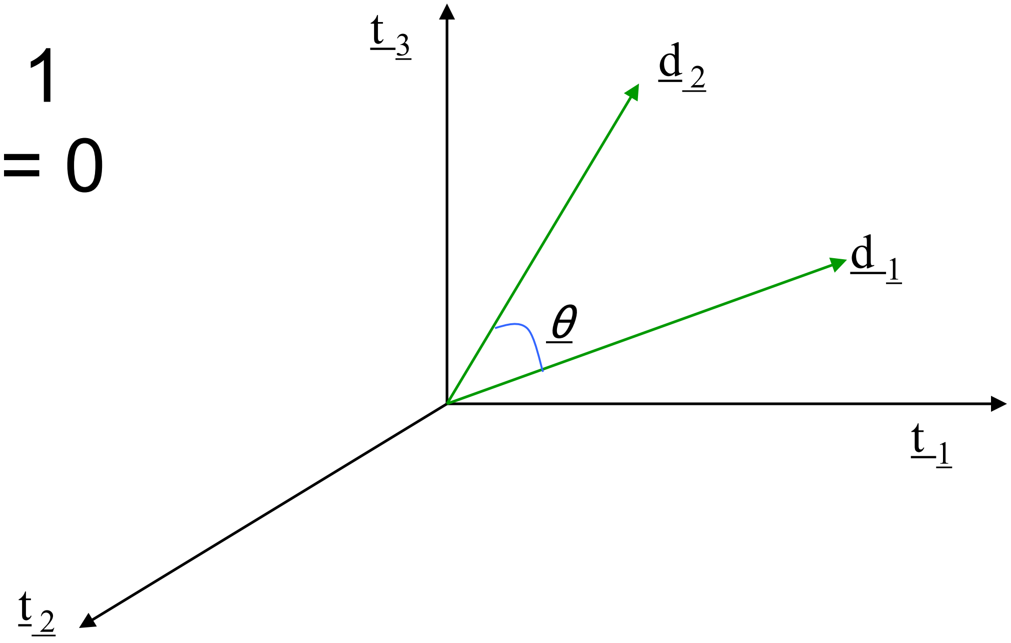
- We now have all documents represented as vectors of weights, which take care of both the **term frequency** (how many times a term appears in a document) and the **document frequency** (in how many documents a term appears)
- We can represent also the query as a vector of weights
- The simple inner product would still work, but we have not yet taken into account **the length of the documents**
- We can therefore change the formula measuring similarity to a different one, whose value does not depend on the length of a document

Similarity in vector space

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

cos 0° = 1

cos 90° = 0



Cosine similarity

- The cosine of the angle between two vectors can easily be computed with the formula

$$X \cdot Y = |X||Y| \cos\theta$$

$$\cos\theta = \frac{X \cdot Y}{|X||Y|}$$

- |X| is the modulus (the length) of X

$$|X| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

- The cosine is therefore

$$\cos\theta = \frac{X \cdot Y}{|X||Y|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

Similarity of two documents

- The cosine formula applied to documents

$$cosine(Q, D_d) = \frac{Q \cdot D_d}{|Q||D_d|} = \frac{1}{W_q W_d} \sum_{t=1}^{n} w_{q,t} \cdot w_{d,t}$$

- W_q and W_d represent the “length” of the vectors
- In the formula above the factor W_q can be ignored as it is just a multiplying factor that does not affect the ranking as it is the same for all documents

Summary of retrieval and ranking

- Build a “term-document matrix”, assigning a weight to each term in a document (instead of just a binary value as in the simple approach)
  - Usually the weight is *tf.idf*, i.e. the product of the “term frequency” (number of occurrences of the term in the document) and the “inverse of the “term document frequency” (number of documents in which the term appears)
- Consider each document as a vector in n-space (n is the number of distinct terms, i.e. the size of the lexicon)
  - The non-zero components of the vector are the weights of the terms appearing in the document
  - Normalize each vector to “unit length” (divide each component by the modulus – the “length” – of the vector)
- Consider also the query as a vector in n-space
  - The non-zero components are just the terms appearing in the query (possibly with a weight)
  - Normalize also the query vector
- Define the similarity measure between the query and a document as the cosine of the “angle” beteeen the two vectors
  - If both vectors are normalized, the computation is just the inner product of the two vectors

IR topics for next DL Course

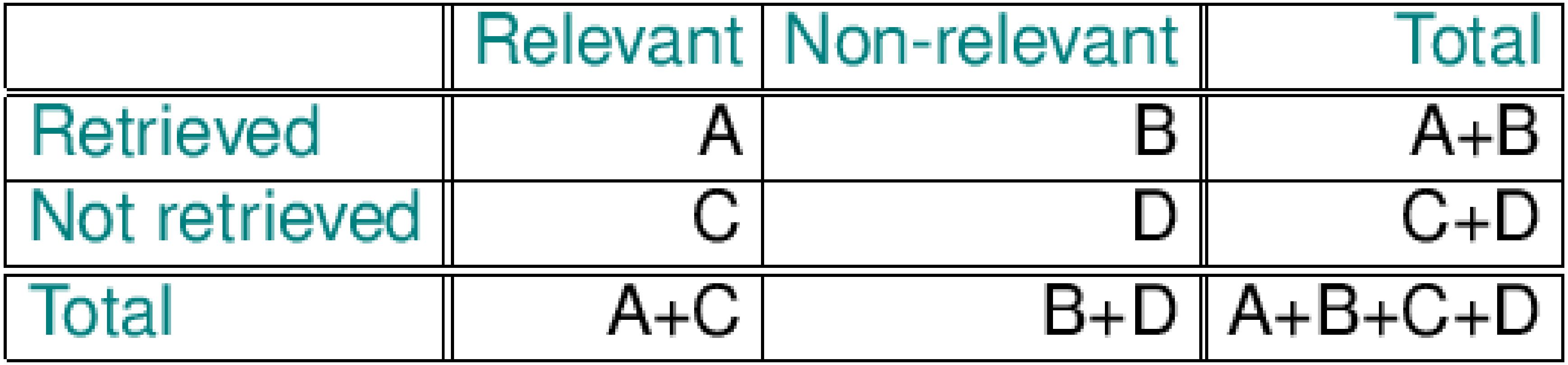
- Queries with a “jolly” character
  - lab* (easy)
  - *lab and *lab* (doable)
  - lab*r (difficult)
- Phrase searching
  - “Romeo and Juliet”
- Query expansion
  - Use of thesauri
  - User feedback
- Boosting
  - Index time
  - Query time
- Multilingual search

Evaluating relevance in IR

- The evaluation of an IR technique or system is in general done *experimentally* (rather than *analytically*).
- The user has an information need, which is translated into a query
- Information need: *I'm looking for information on whether drinking red wine is more effective at reducing your risk of heart attacks than white wine.*
- Query: *wine red white heart attack effective*
- Evaluation should be based on whether a doc addresses the information need, not whether it has those words

Relevance measurement

Given a query, we can divide the documents in the collection into four categories



	Relevant	Non-relevant	Total
Retrieved	A	B	A+B
Not retrieved	C	D	C+D
Total	A+C	B+D	A+B+C+D

Evaluation Metrics

- Recall
  - Proportion of retrieved relevant items out of all the relevant items
  - It measures the ability of the system to present “all” the relevant documents

$$\left(\frac{A}{A+C}\right)$$

- Precision
  - Proportion of retrieved relevant items out of all the retrieved items
  - It measures the ability of the system to present only those items that are relevant

$$\left(\frac{A}{A+B}\right)$$

Precision/Recall

- There will be a high recall (but low precision) by retrieving all docs for all queries
- Recall is a non-decreasing function of the number of docs retrieved
- In a good system, precision decreases as either the number of docs retrieved increases or recall increases
  - A fact with strong empirical confirmation
- Depending on the application, a user may prefer a high precision (e.g. a Google query) or a high recall (e.g. a query to a medical or legal digital library)