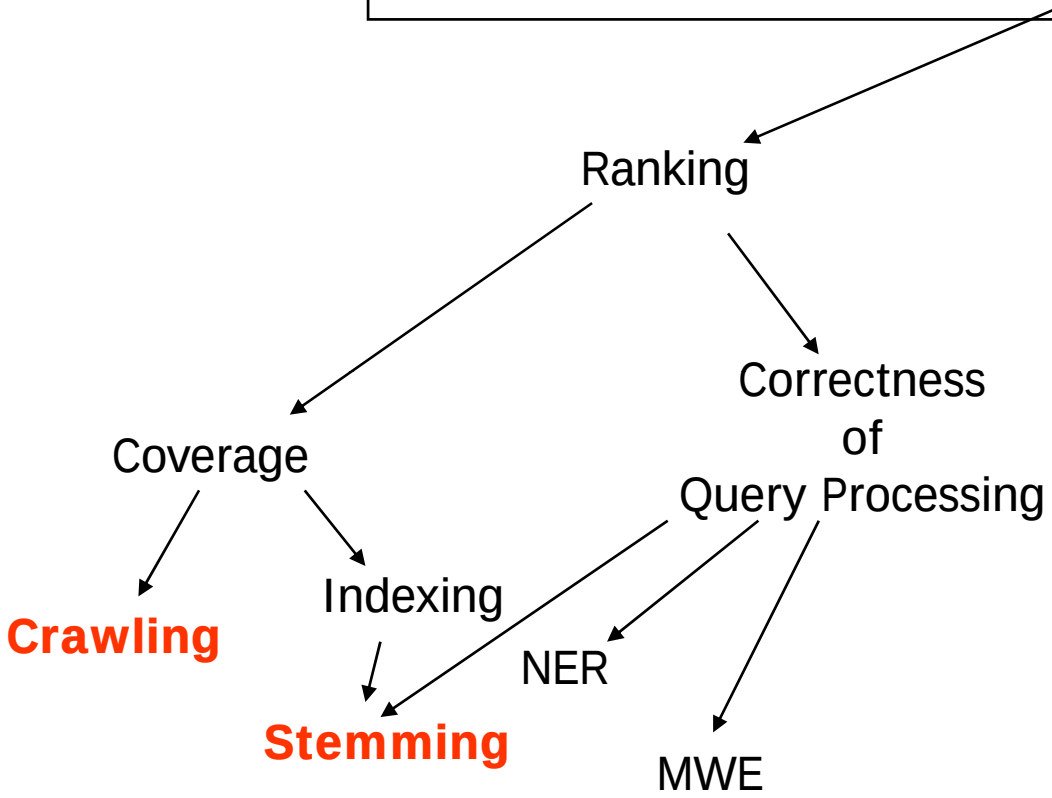
# CS344: Introduction to Artificial Intelligence

Pushpak Bhattacharyya
CSE Dept.,
IIT Bombay

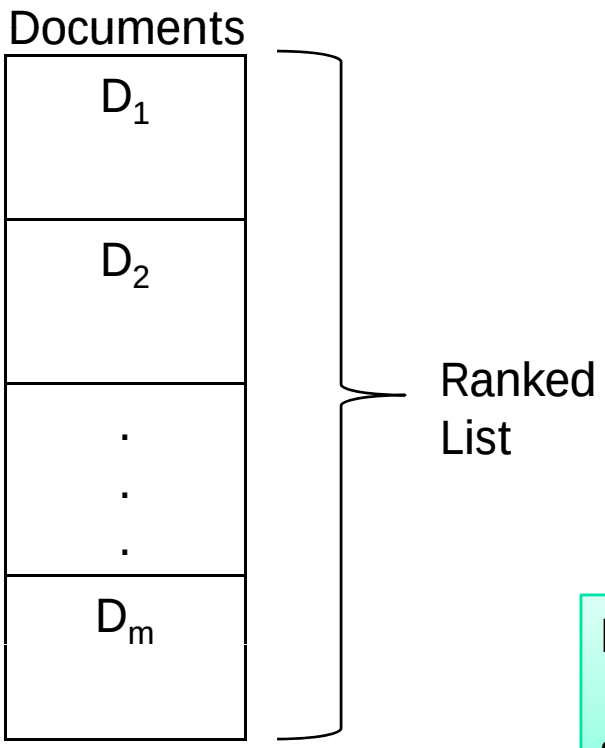
Lecture 32-33: Information Retrieval: Basic concepts and Model

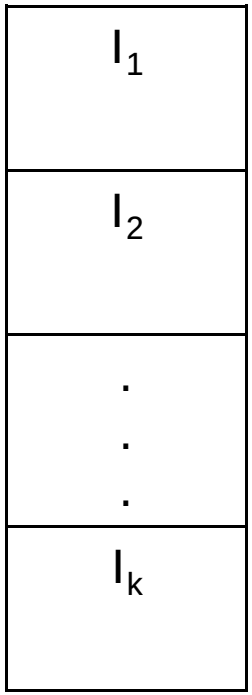
# The elusive *user satisfaction*



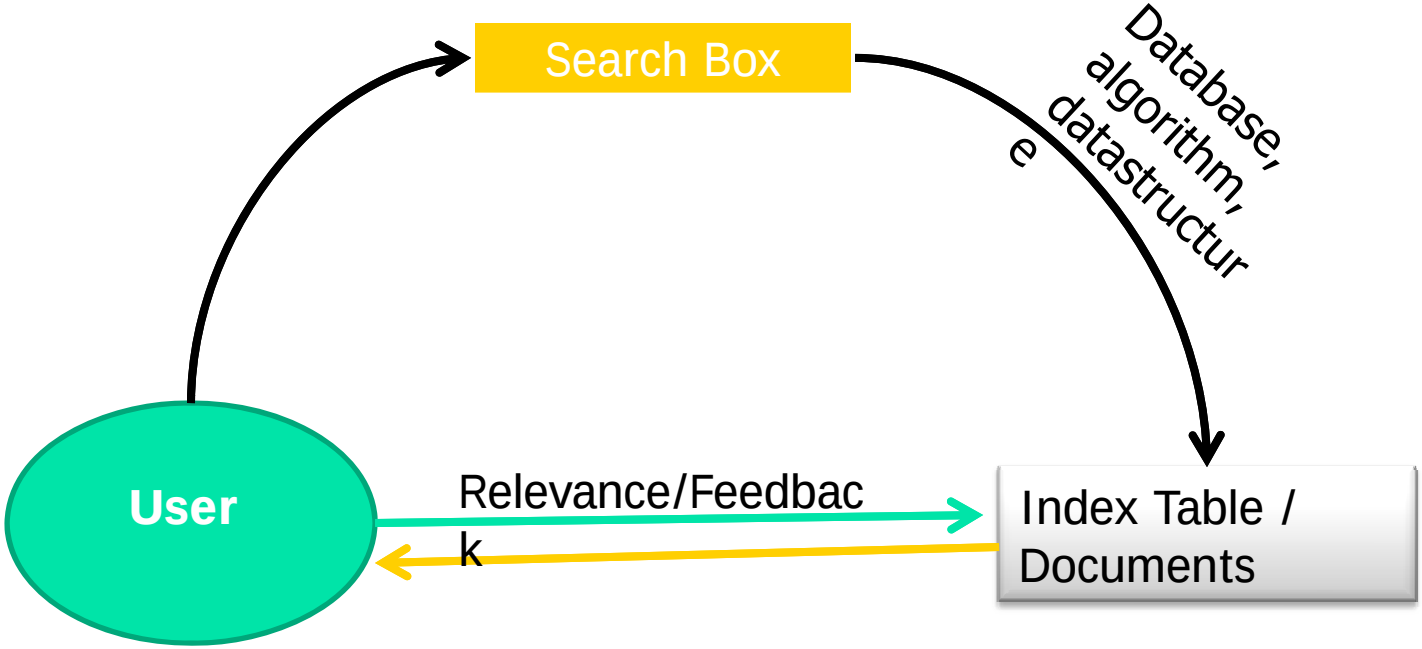
# What happens in IR

$q_1$ $q_2$ ... $q_n$ // Search Box, $q_i$ are query terms



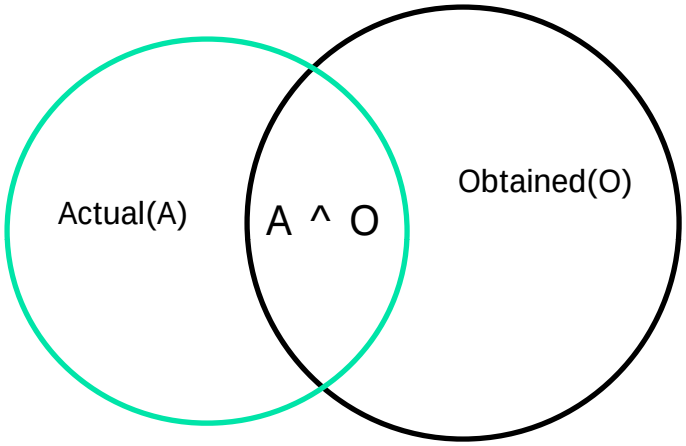




Note: High ranked relevant document = user information need getting satisfied !


Search Box
Database, algorithm, datastructur e
User
Relevance/Feedbac k
Index Table / Documents

# How to check quality of retrieval (P, R, F)

- Three parameters
  - Precision P = |A ^ O|/|O|
  - Recall R = |A ^ O| / |A|
  - F-score = 2PR/(P+R)
    - Harmonic mean



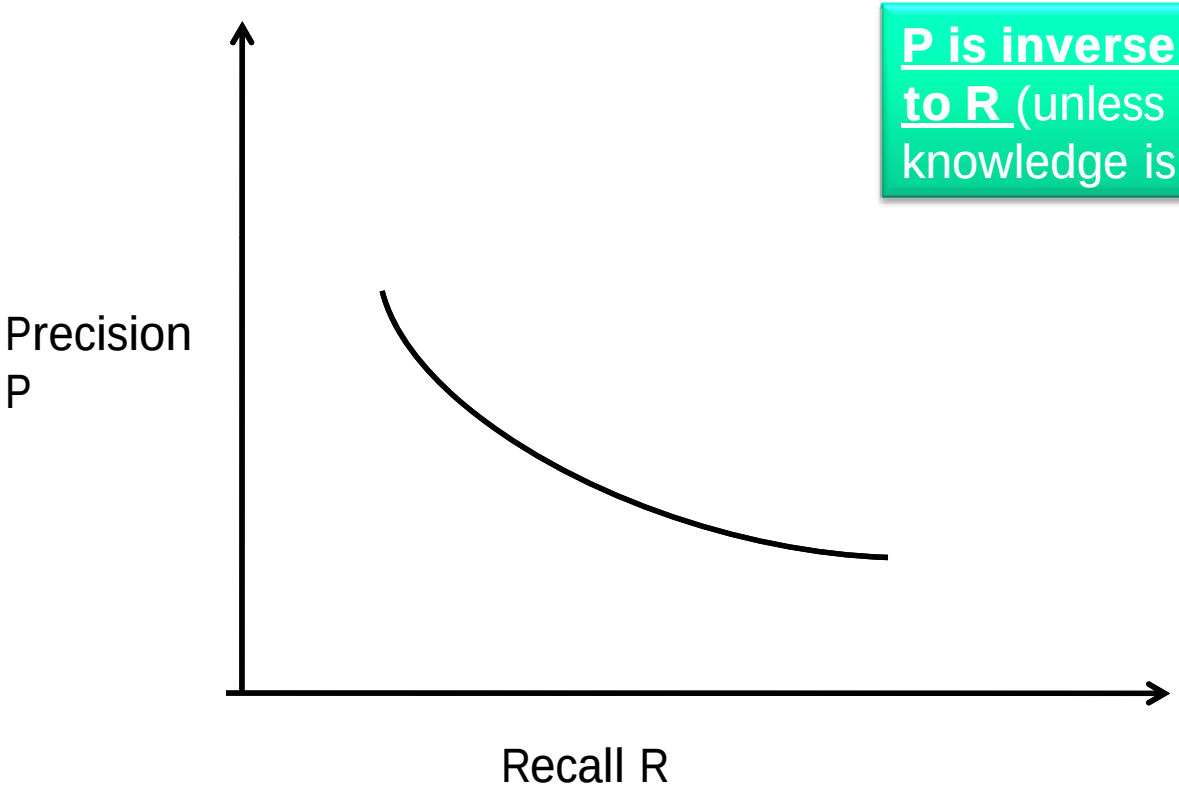

All the above formula are very general. We haven't considered that the documents retrieved are ranked and thus the above expressions need to be

# P, R, F (contd.)

- Precision is easy to calculate, Recall is not.
- Given a known set of pair of <q, D>
- Relevance judgement <q,D> $\rightarrow \{0, 1\}$ (Human evaluation)

# Relation between P & R

**P is inversely related to R** (unless additional knowledge is given)



# Precision at rank k

- Choose the top k documents, see how many of them are relevant out of them.
- $P_k$ = (# of relevant documents)/k
- Mean Average Precision (MAP)

  = $\frac{1}{L}\sum P_K$

| Documents |
| --- |
| $D_1$ |
| $D_2$ |
| . . . |
| $D_k$ |
| . . . |
| $D_m$ |

# Sample Exercise

- $D_1$: Delhi is the capital of India. It is a large city.
- $D_2$: Mumbai, however is the commercial capital with million dollars inflow & outflow.

**The words in red constitute the useful words from each sentence. The other words (those in black) are very common and thus do not add to the information content of the sentence.**

Vocabulary: unique red words, 11 in number; each doc will be represented by a 11-tuple vector: each component 1 or 0

# IR Basics

(mainly from *R. Baeza-Yates and B. Ribeiro-Neto. Modern Information RetrievalAddison-Wesley, Wokingham, UK, 1999.*
and
Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press. 2008. )

# Definition of IR Model

An IR model is a quadrupul

$$[D, Q, F, R(q_i, d_j)]$$

*Where,*

*D:* documents

*Q*: Queries

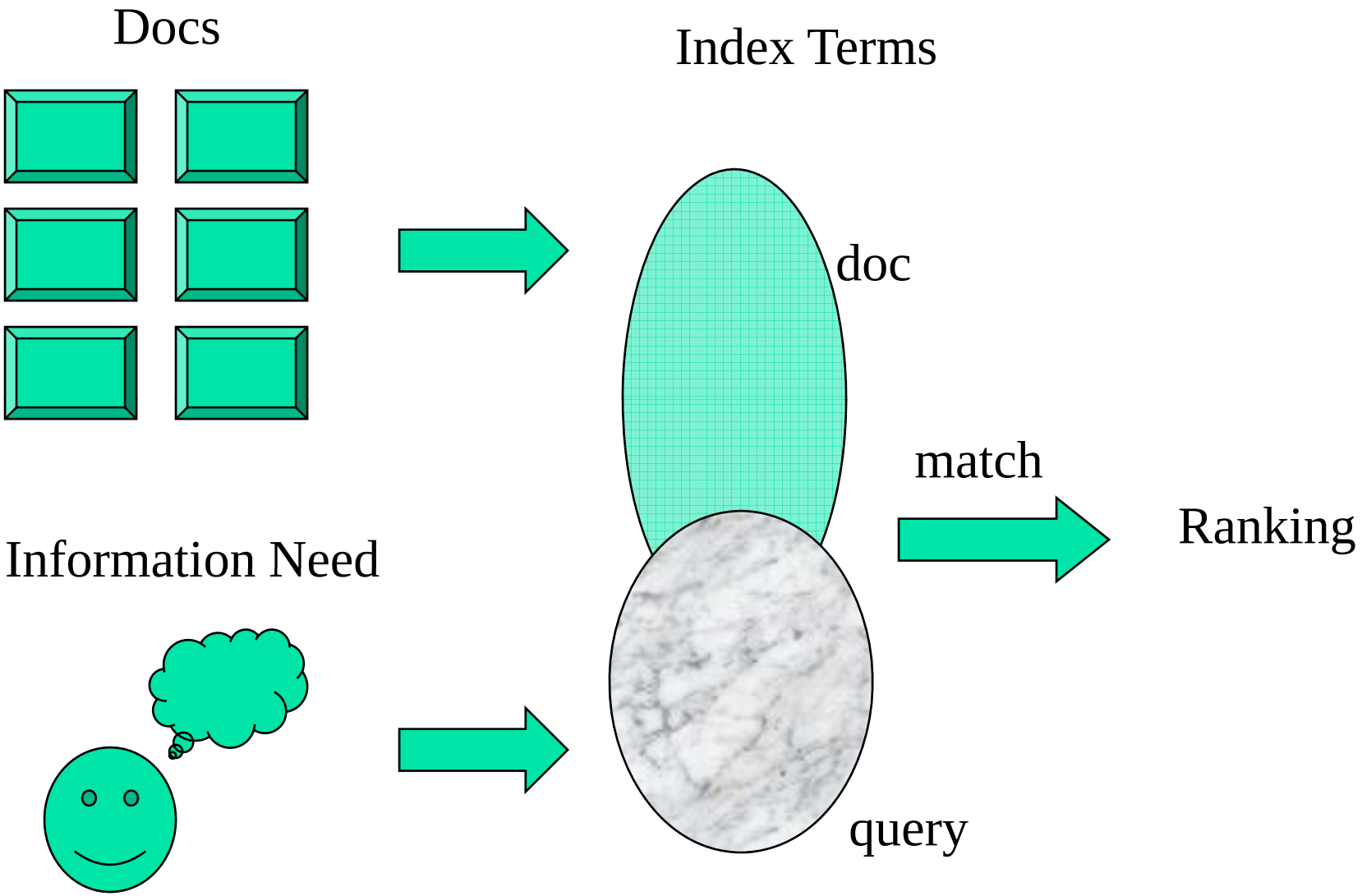
*F*: Framework for modeling document, query and their relationships

*R(.,.)*: Ranking function returning a real no. expressing the relevance of $d_j$ with $q_i$

# Index Terms

- Keywords representing a document
- Semantics of the word helps remember the main theme of the document
- Generally nouns
- Assign numerical weights to index terms to indicate their importance

# Introduction



# Classic IR Models - Basic Concepts

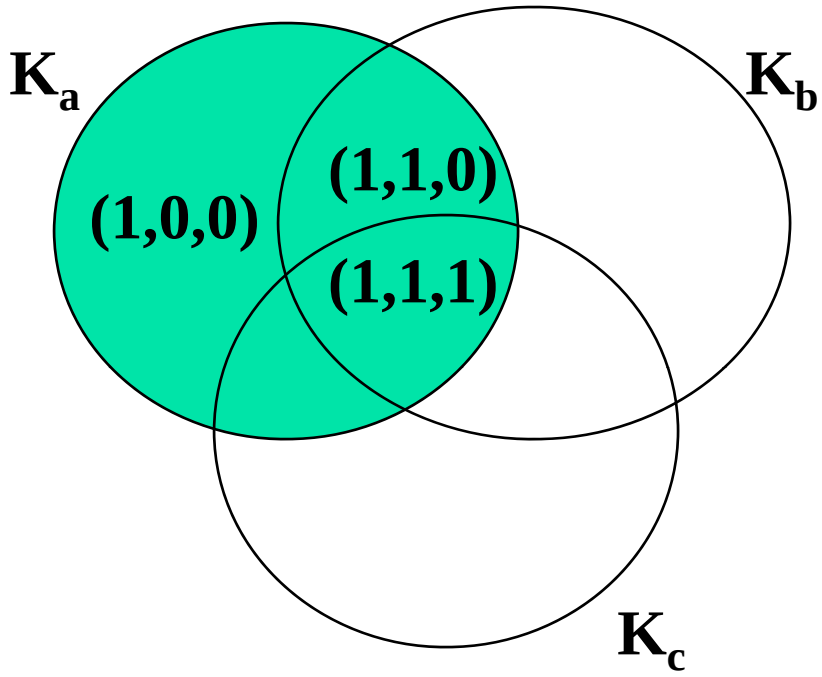
- The *importance* of the index terms is represented by weights associated to them
- Let
  - $t$ be the number of index terms in the system
  - $K = \{k_1, k_2, k_3, \ldots k_t\}$ set of all index terms
  - $k_i$ be an index term
  - $d_j$ be a document
  - $w_{ij}$ is a weight associated with $(k_i, d_j)$
  - $w_{ij} = 0$ indicates that term does not belong to doc
  - $vec(d_j) = (w_{1j}, w_{2j}, \ldots, w_{tj})$ is a weighted vector associated with the document *dj*
  - $g_i(vec(dj)) = w_{ij}$ is a function which returns the weight associated with pair $(k_i, d_j)$

# The Boolean Model

- Simple model based on set theory
- Only *AND, OR* and *NOT* are used
- Queries specified as boolean expressions
  - precise semantics
  - neat formalism
  - $q = k_a \wedge (k_b \vee \neg k_c)$
- Terms are either present or absent. Thus, $w_{ij} \varepsilon \{0,1\}$
- Consider
  - $q = k_a \wedge (k_b \vee \neg k_c)$
  - $vec(q_{dnf}) = (1,1,1) \vee (1,1,0) \vee (1,0,0)$
  - $vec(q_{cc}) = (1,1,0)$ is a conjunctive component

# The Boolean Model

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




- $sim(q,d_j) = 1$ if $\exists\ vec(q_{cc}) \mid (vec(q_{cc}) \in vec(q_{dnf})) \wedge (\forall k_i,\ g_i(vec(d_j)) = g_i(vec(q_{cc})))$

  0 otherwise

# Drawbacks of the Boolean Model

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

# The Vector Model

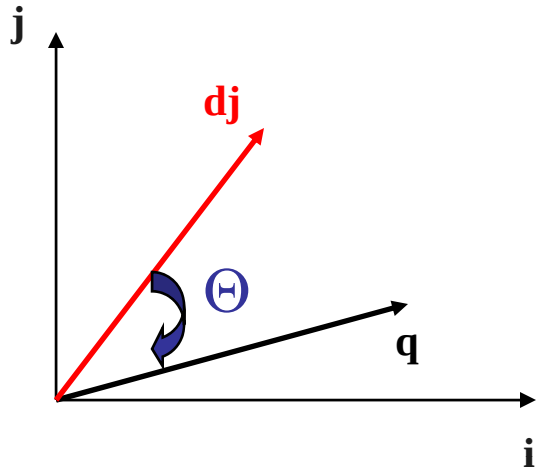
- Use of binary weights is too limiting
- Non-binary weights provide consideration for partial matches
- These term weights are used to compute a *degree of similarity* between a query and each document
- Ranked set of documents provides for better matching

# The Vector Model

- Define:
  - $w_{ij} > 0$ whenever $k_i \in d_j$
  - $w_{iq} >= 0$ associated with the pair $(k_i, q)$
  - $vec(d_j) = (w_{1j}, w_{2j}, \ldots, w_{tj})$
    $vec(q) = (w_{1q}, w_{2q}, \ldots, w_{tq})$

- In this space, queries and documents are represented as weighted vectors

# The Vector Model




- $Sim(q,d_j) = cos(\Theta)$
  $= [vec(d_j) \bullet vec(q)] / |d_j| * |q| = [\Sigma\ w_{ij} * w_{iq}] / |d_j| * |q|$
- Since $w_{ij} > 0$ and $w_{iq} > 0$, $0 <= sim(q,d_j) <= 1$
- A document is retrieved even if it matches the query terms only partially

# The Vector Model

- $Sim(q,dj) = [\Sigma\ w_{ij} * w_{iq}] / |d_j| * |q|$
- How to compute the weights $w_{ij}$ and $w_{iq}$ ?
- A good weight must take into account two effects:
  - quantification of intra-document contents (similarity)
    - *tf* factor, the *term frequency* within a document
  - quantification of inter-documents separation (dissi-milarity)
    - *idf* factor, the *inverse document frequency*
  - $wij = tf(i,j) * idf(i)$

# The Vector Model

- Let,
  - $N$ be the total number of docs in the collection
  - $n_i$ be the number of docs which contain $k_i$
  - $freq(i,j)$ raw frequency of $k_i$ within $d_j$
- A normalized $tf$ factor is given by
  - $f(i,j) = freq(i,j) / max_l(freq(l,j))$
  - where the maximum is computed over all terms which occur within the document $dj$
- The $idf$ factor is computed as
  - $idf(i) = log (N/n_i)$
  - the *log* is used to make the values of $tf$ and $idf$ comparable. It can also be interpreted as the *amount of information* associated with the term $k_i$.

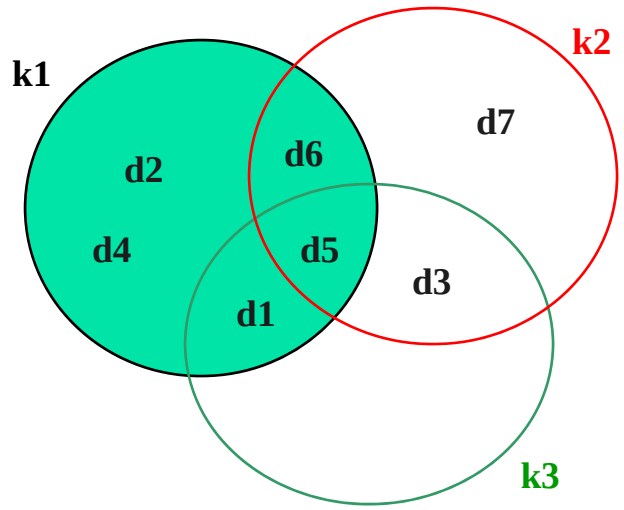
# The Vector Model

- The best term-weighting schemes use weights which are give by
  - $w_{ij} = f(i,j) * log(N/n_i)$
  - the strategy is called a *tf-idf* weighting scheme
- For the query term weights, a suggestion is
  - $w_{iq} = (0.5 + [0.5 * freq(i,q) / max_l(freq(l,q)]) * log(N/ni)$
- The vector model with *tf-idf* weights is a good ranking strategy with general collections
- The vector model is usually as good as the known ranking alternatives. It is also simple and fast to compute.

# The Vector Model

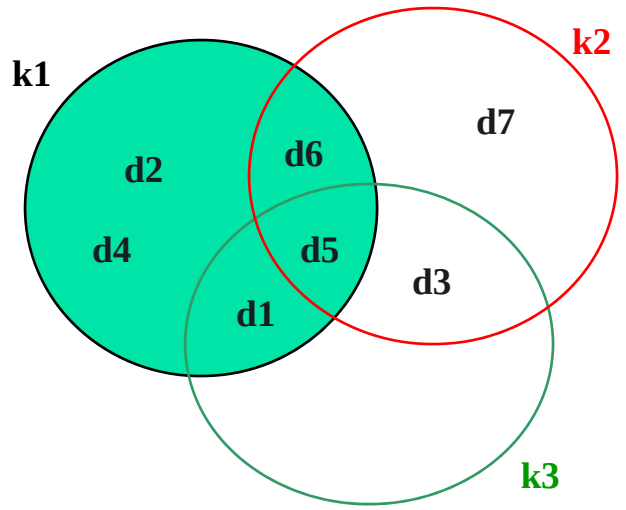
- Advantages:
  - term-weighting improves quality of the answer set
  - partial matching allows retrieval of docs that approximate the query conditions
  - cosine ranking formula sorts documents according to degree of similarity to the query
- Disadvantages:
  - assumes independence of index terms; not clear that this is bad though

# The Vector Model: Example I



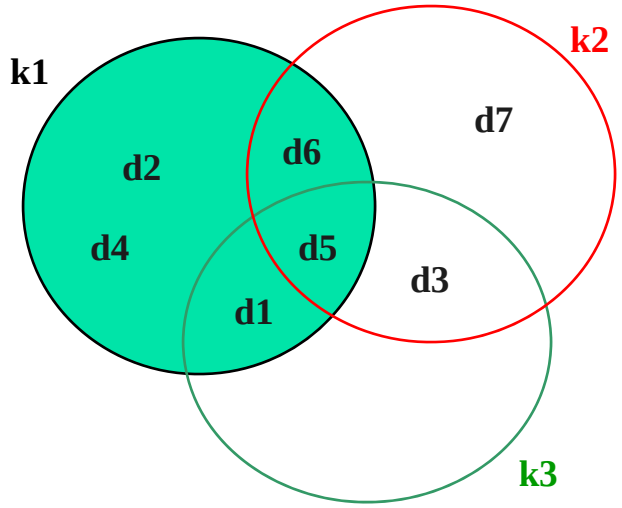

| | k1 | k2 | k3 | q • dj |
|---|---|---|---|---|
| d1 | 1 | 0 | 1 | 2 |
| d2 | 1 | 0 | 0 | 1 |
| d3 | 0 | 1 | 1 | 2 |
| d4 | 1 | 0 | 0 | 1 |
| d5 | 1 | 1 | 1 | 3 |
| d6 | 1 | 1 | 0 | 2 |
| d7 | 0 | 1 | 0 | 1 |
| | | | | |
| q | 1 | 1 | 1 | |

# The Vector Model: Example II




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

# The Vector Model: Example III




| | k1 | k2 | k3 | q • dj |
|---|---|---|---|---|
| d1 | 2 | 0 | 1 | 5 |
| d2 | 1 | 0 | 0 | 1 |
| d3 | 0 | 1 | 3 | 11 |
| d4 | 2 | 0 | 0 | 2 |
| d5 | 1 | 2 | 4 | 17 |
| d6 | 1 | 2 | 0 | 5 |
| d7 | 0 | 5 | 0 | 10 |
| | | | | |
| q | 1 | 2 | 3 | |