Recuperação de Informação B

Cap. 02: Modeling

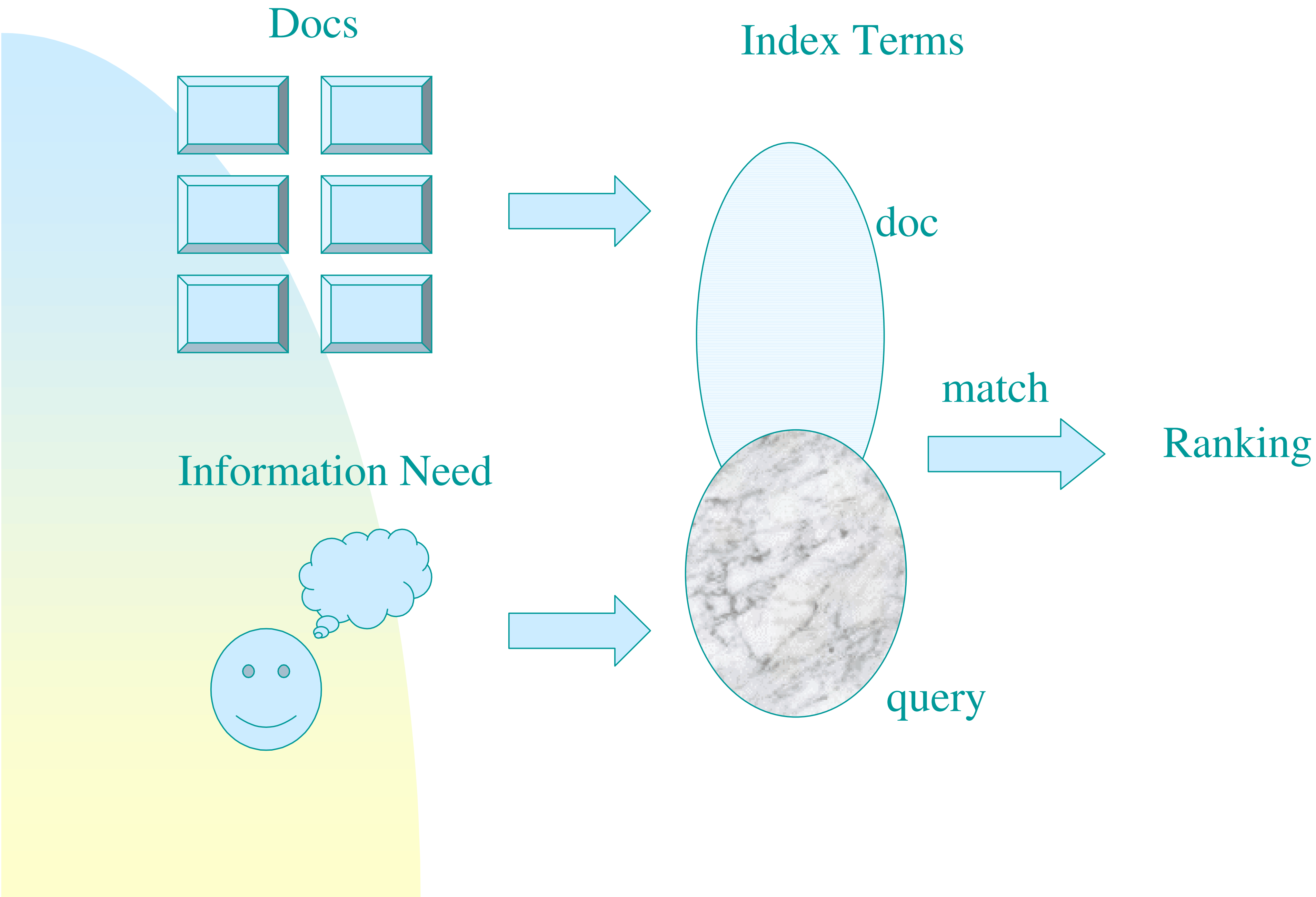
2.1, 2.2, 2.3, 2.4, 2.5.1, 2.5.2, 2.5.3

September 01, 1999

Introduction

- IR systems usually adopt index terms to process queries
- Index term:
  - a keyword or group of selected words
  - any word (more general)
- Stemming might be used:
  - connect: connecting, connection, connections
- An inverted file is built for the chosen index terms

Introduction



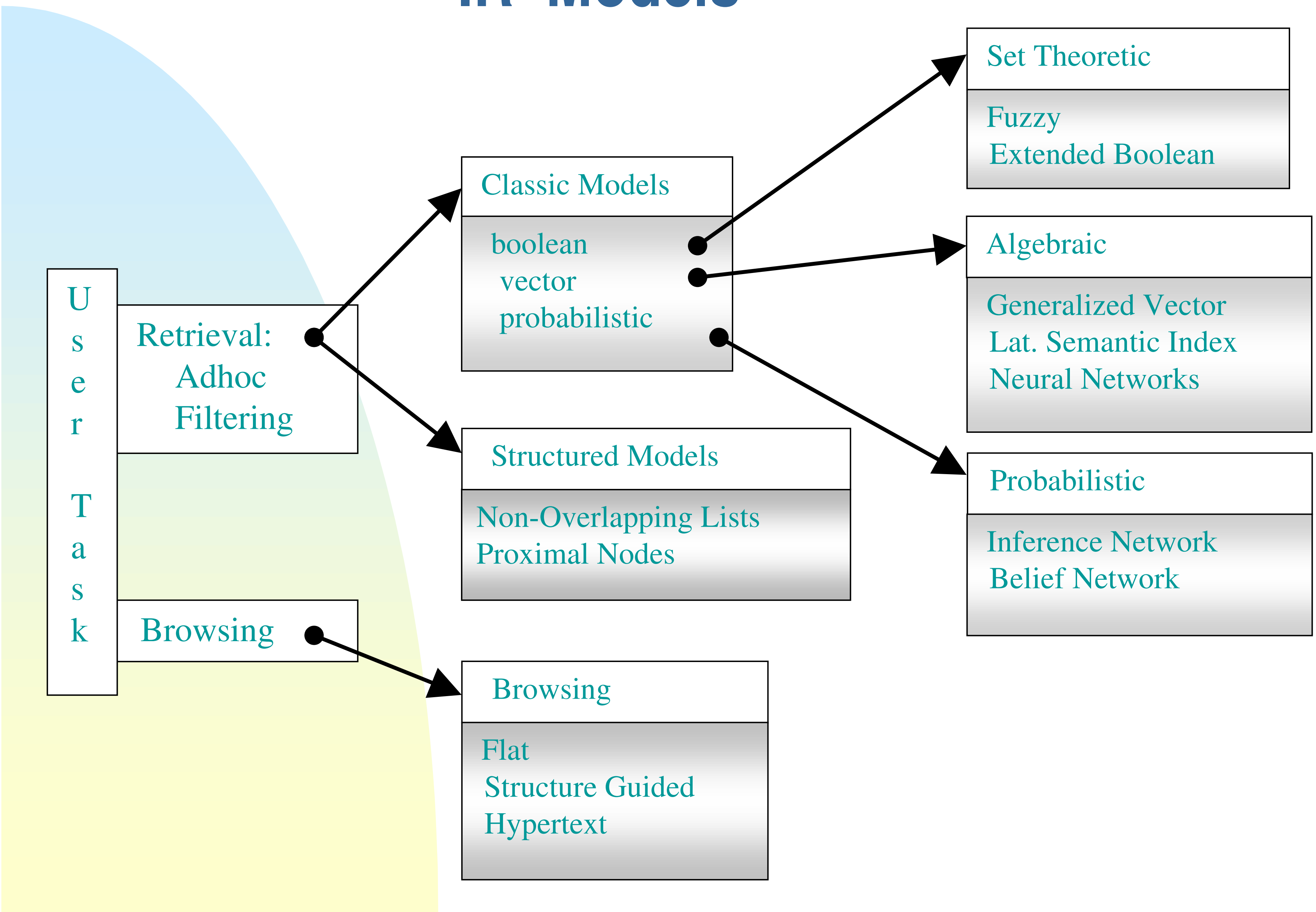
Introduction

- Matching at index term level is quite imprecise
- No surprise that users get frequently unsatisfied
- Since most users have no training in query formation, problem is even worst
- Frequent dissatisfaction of Web users
- Issue of deciding relevance is critical for IR systems: ranking

Introduction

- A *ranking* is an ordering of the documents retrieved that (hopefully) reflects the relevance of the documents to the user query
- A ranking is based on fundamental premisses regarding the notion of relevance, such as:
  - common sets of index terms
  - sharing of weighted terms
  - likelihood of relevance
- Each set of premisses leads to a distinct *IR model*

IR Models



U s e r T a s k
Retrieval:
Adhoc
Filtering
Browsing
Classic Models
boolean
vector
probabilistic
Structured Models
Non-Overlapping Lists
Proximal Nodes
Browsing
Flat
Structure Guided
Hypertext
Set Theoretic
Fuzzy
Extended Boolean
Algebraic
Generalized Vector
Lat. Semantic Index
Neural Networks
Probabilistic
Inference Network
Belief Network

IR Models

- The IR model, the logical view of the docs, and the retrieval task are distinct aspects of the system

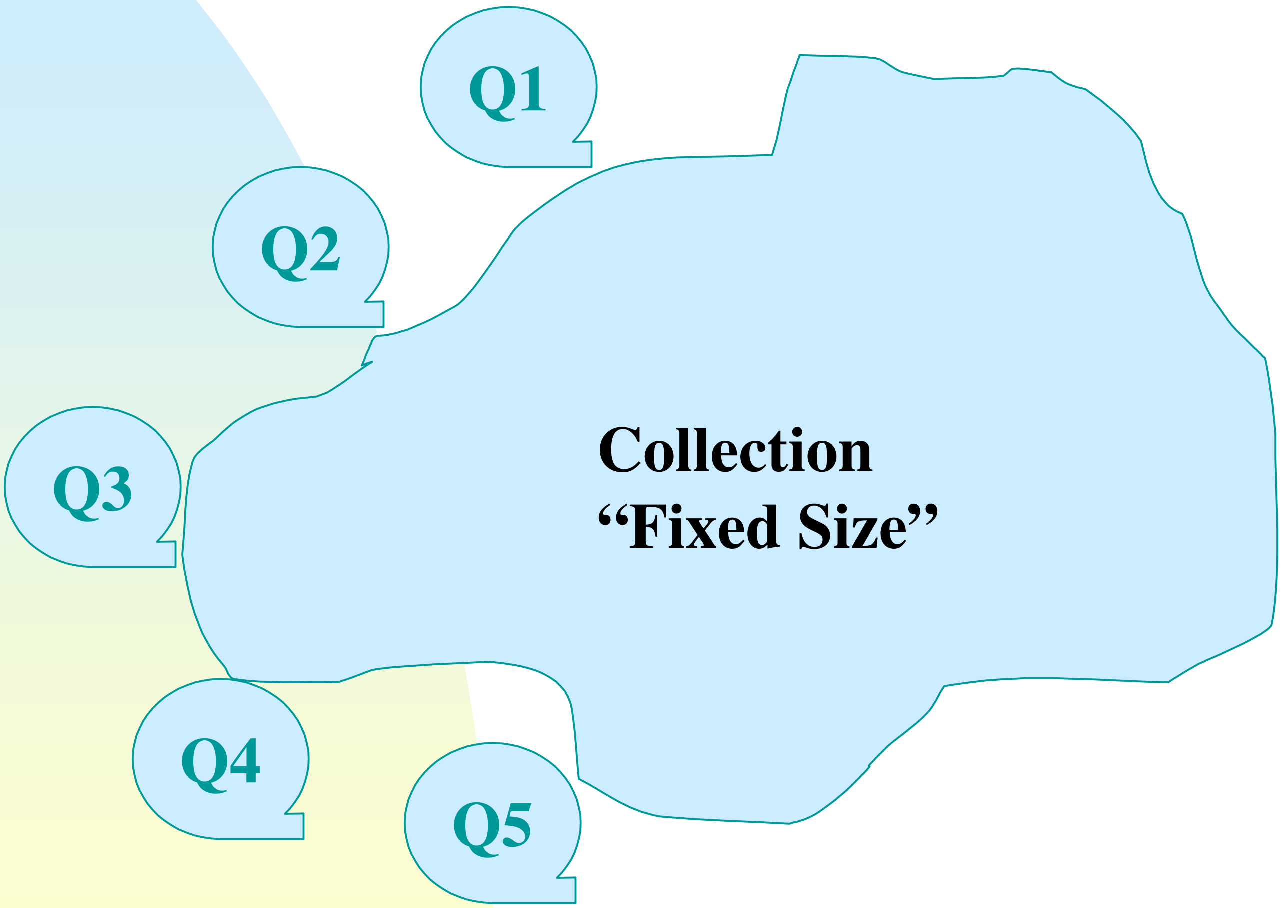
LOGICAL VIEW OF DOCUMENTS

U S E R T A S K

	Index Terms	**Full Text**	**Full Text + Structure**
Retrieval	Classic Set Theoretic Algebraic Probabilistic	Classic Set Theoretic Algebraic Probabilistic	Structured
Browsing	Flat	Flat Hypertext	Structure Guid Hypertext

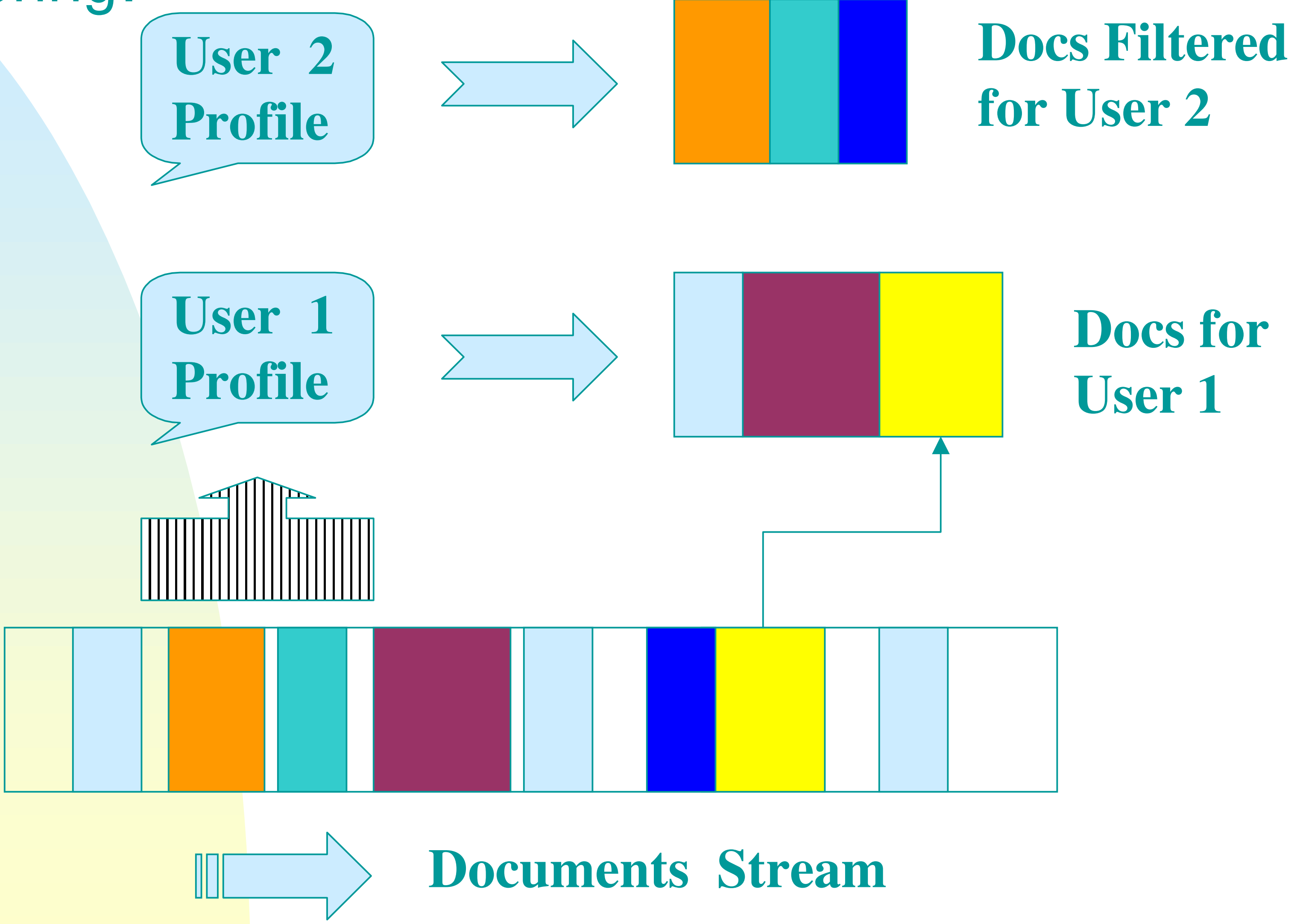
Retrieval: Ad Hoc x Filtering

- Ad hoc retrieval:



Retrieval: Ad Hoc x Filtering

- Filtering:



Classic IR Models - Basic Concepts

- Each document represented by a set of representative keywords or index terms
- An index term is a document word useful for remembering the document main themes
- Usually, index terms are nouns because nouns have meaning by themselves
- However, search engines assume that all words are index terms (full text representation)

Classic IR Models - Basic Concepts

- Not all terms are equally useful for representing the document contents: less frequent terms allow identifying a narrower set of documents
- The *importance* of the index terms is represented by weights associated to them
- Let
  - *ki* be an index term
  - *dj* be a document
  - *wij* is a weight associated with *(ki,dj)*
- The weight *wij* quantifies the importance of the index term for describing the document contents

Classic IR Models - Basic Concepts

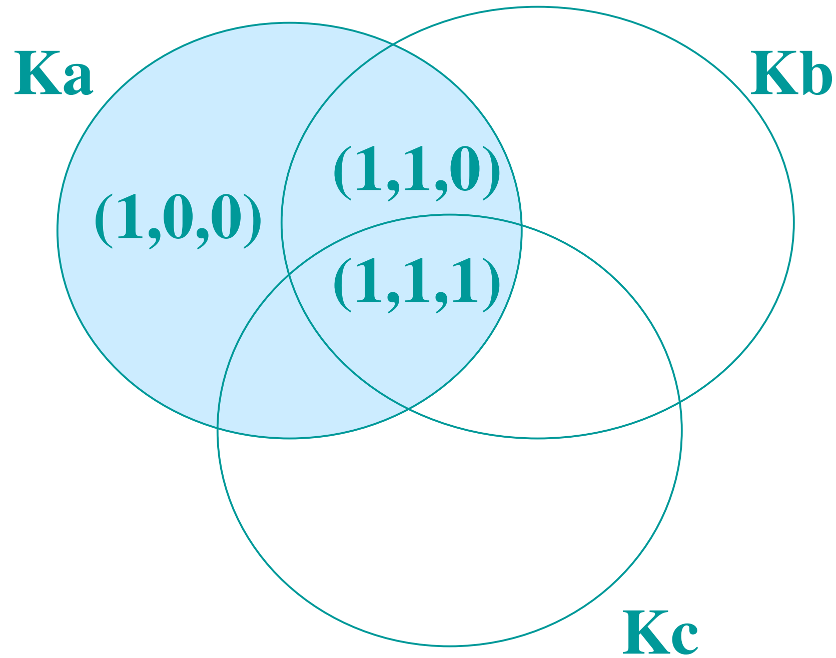
- *Ki* is an index term
- *dj* is a document
- *t* is the total number of docs
- *K = (k1, k2, …, kt)* is the set of all index terms
- *wij >= 0* is a weight associated with *(ki,dj)*
- *wij = 0* indicates that term does not belong to doc
- *vec(dj) = (w1j, w2j, …, wtj)* is a weighted vector associated with the document *dj*
- *gi(vec(dj)) = wij* is a function which returns the weight associated with pair *(ki,dj)*

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, $wij \; \varepsilon \; \{0,1\}$
- 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

The Boolean Model

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



- $sim(q,dj) = 1$ if $\exists\ vec(qcc)\ |$
  $(vec(qcc)\ \varepsilon\ vec(qdnf)) \wedge$
  $(\forall ki,\ gi(vec(dj)) = gi(vec(qcc)))$
  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

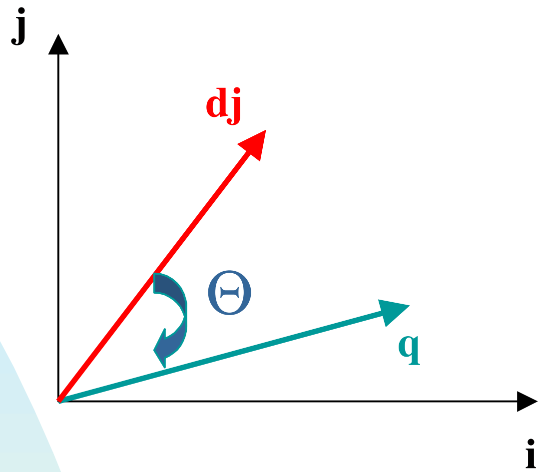
The Vector Model

- 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:
  - *wij > 0* whenever *ki* $\in$ *dj*
  - *wiq >= 0* associated with the pair *(ki,q)*
  - *vec(dj) = (w1j, w2j, ..., wtj)*
    vec(q) = (w1q, w2q, ..., wtq)
  - To each term *ki* is associated a unitary vector *vec(i)*
  - The unitary vectors *vec(i)* and *vec(j)* are assumed to be orthonormal (i.e., index terms are assumed to occur independently within the documents)
- The *t* unitary vectors *vec(i)* form an orthonormal basis for a t-dimensional space
- In this space, queries and documents are represented as weighted vectors

The Vector Model



- *Sim(q,dj) = cos(Θ)*

  *= [vec(dj) • vec(q)] / |dj| * |q|*

  *= [Σ wij * wiq] / |dj| * |q|*

- Since *wij > 0* and *wiq > 0,*

  0 <= sim(q,dj) <=1

- A document is retrieved even if it matches the query terms only partially

The Vector Model

- $Sim(q,dj) = [\Sigma\ wij * wiq] / |dj| * |q|$
- How to compute the weights *wij* and *wiq* ?
- 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
  - ni be the number of docs which contain ki
  - $freq(i,j)$ raw frequency of ki within dj
- A normalized *tf* factor is given by
  - $f(i,j) = freq(i,j) / max(freq(l,j))$
  - where the maximum is computed over all terms which occur within the document dj
- The *idf* factor is computed as
  - $idf(i) = log(N/ni)$
  - 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 ki.

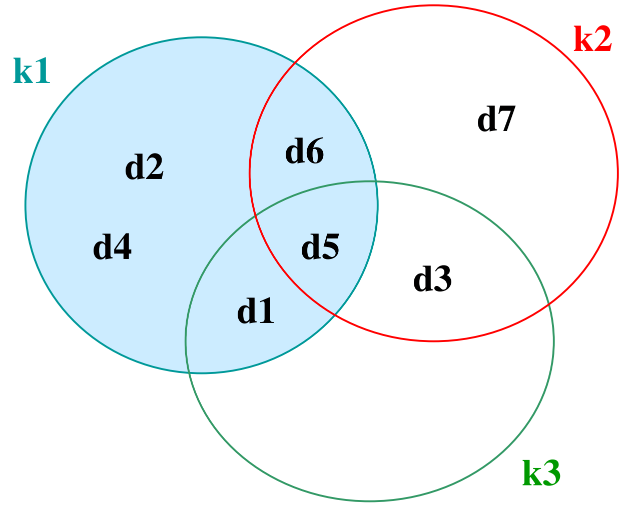
The Vector Model

- The best term-weighting schemes use weights which are give by
  - *wij = f(i,j) * log(N/ni)*
  - the strategy is called a *tf-idf* weighting scheme
- For the query term weights, a suggestion is
  - *wiq = (0.5 + [0.5 * freq(i,q) / max(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

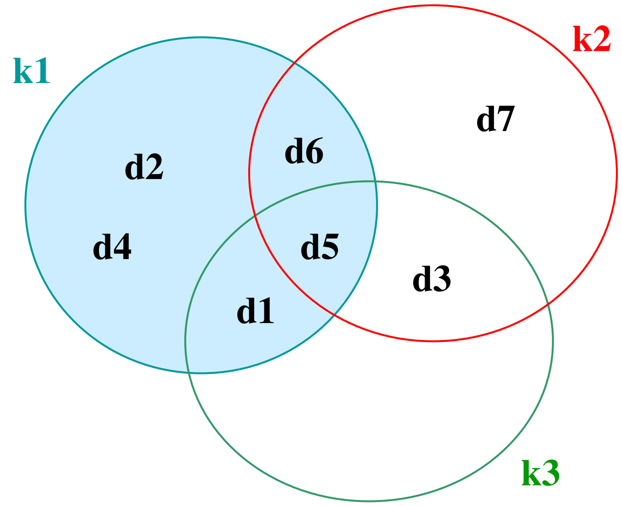
- Advantages:
  - term-weighting improves quality of the answer set
  - partial matching allows retrieval of docs that approximate the query conditions
  - cosine ranking formula sorts documents according to degree of similarity to the query
- Disadvantages:
  - assumes independence of index terms (??);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

The Vector Model: Example I



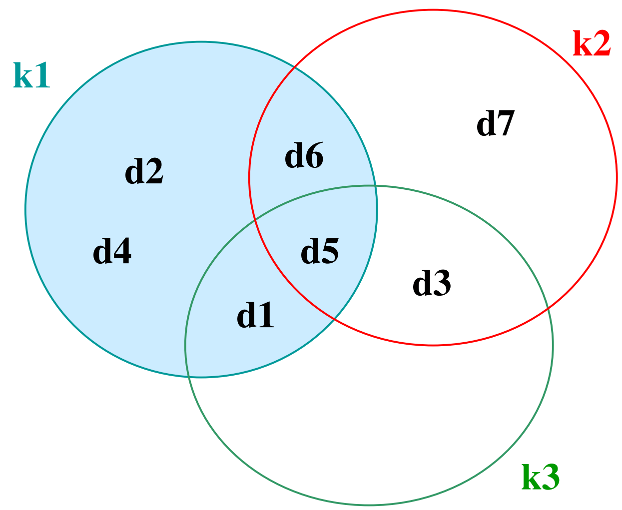
	k1	k2	k3	q • dj
d1	1	0	1	2
d2	1	0	0	1
d3	0	1	1	2
d4	1	0	0	1
d5	1	1	1	3
d6	1	1	0	2
d7	0	1	0	1
q	1	1	1	

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	