Modern Information Retrieval

Chapter 2

Modeling

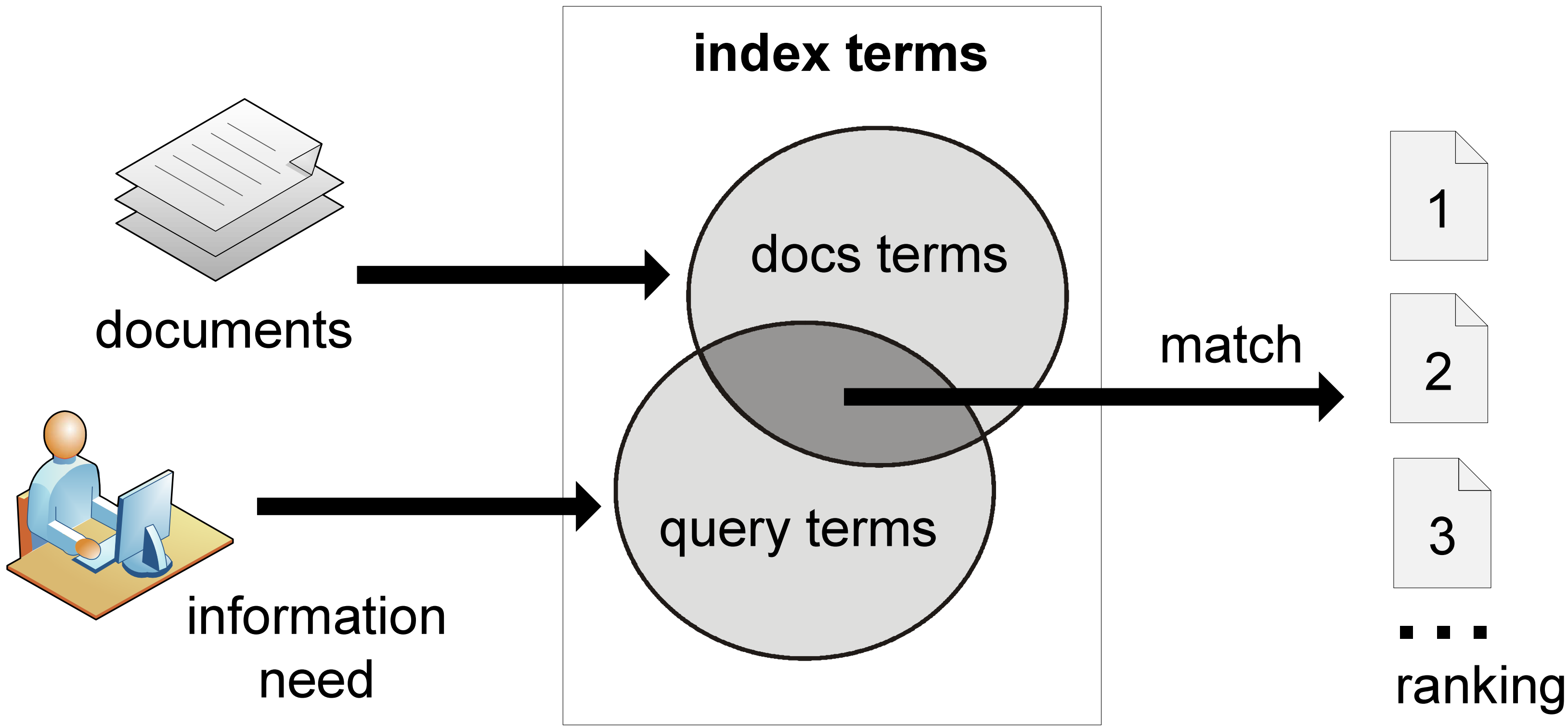
Introduction to IR Models
Retrieval: Ad Hoc x Filtering
Classic IR Models

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

- Information retrieval process



Introduction

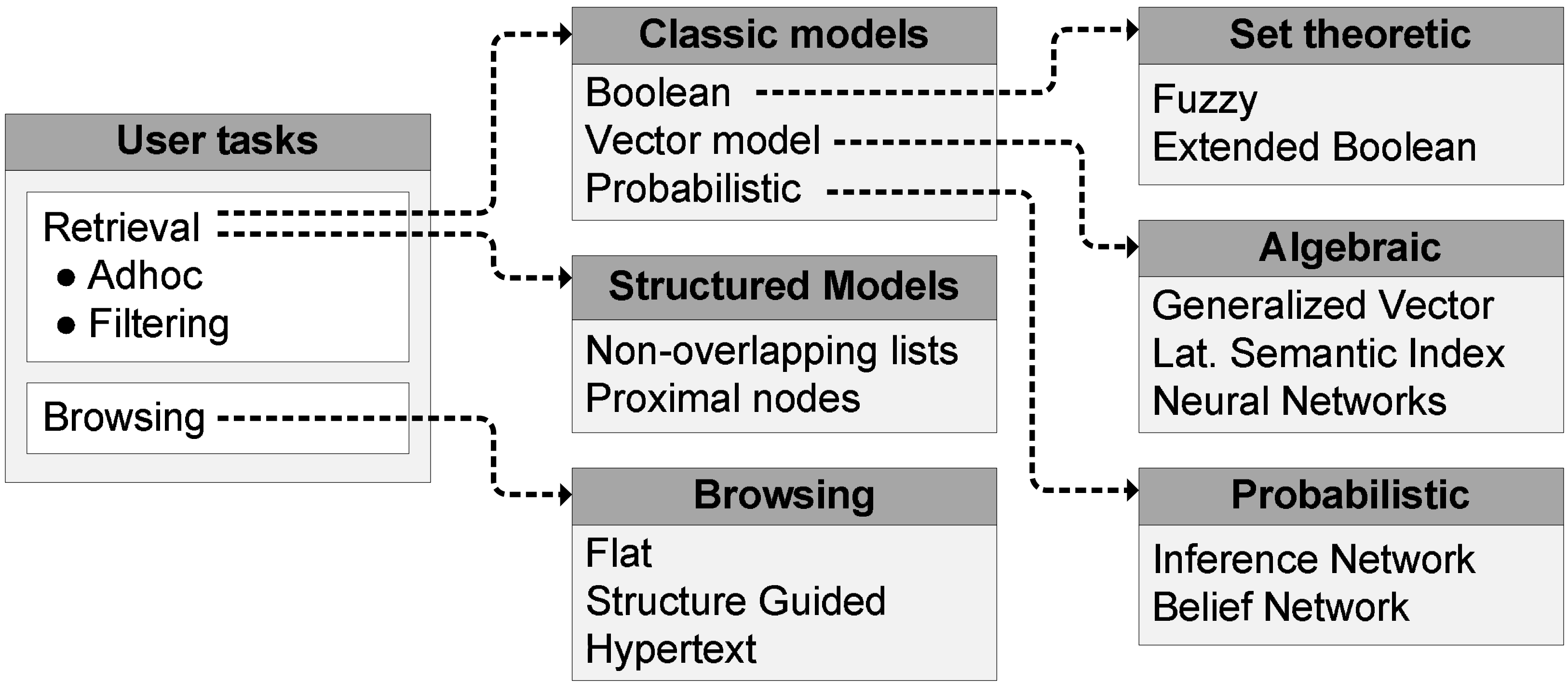
- Matching at index term level is quite imprecise
- No surprise that users get frequently unsatisfied
- Since most users have no training in query formation, problem is even worst
- Frequent dissatisfaction of Web users
- Issue of deciding relevance is critical for IR systems: ranking

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

- A taxonomy of information retrieval models



IR Models

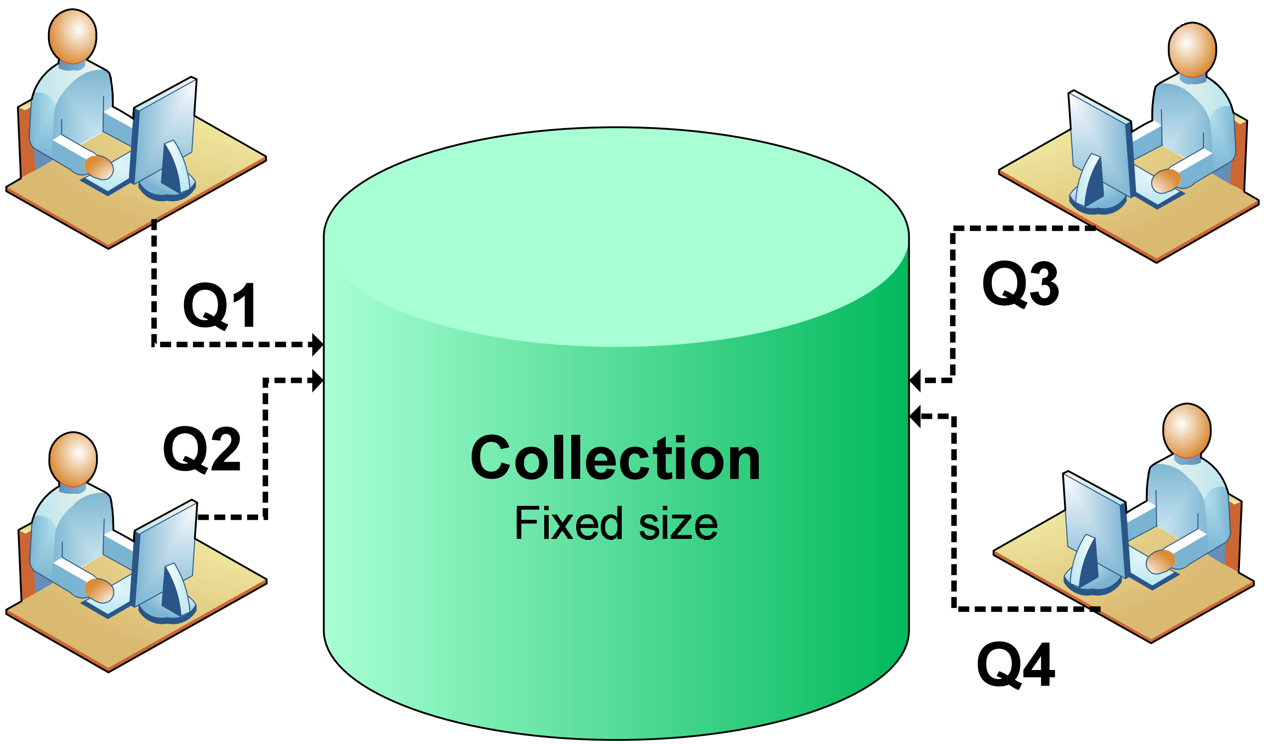
- The IR model, the logical view of the docs, and the retrieval task are distinct aspects of the system

Logical View of Documents

User Tasks		Index terms	Full Text	Full Text + Structure
	Retrieval	Classic Set theoretic Algebraic Probabilistic	Classic Set theoretic Algebraic Probabilistic	Structured
	Browsing	Flat	Flat Hypertext	Structure guided Hypertext

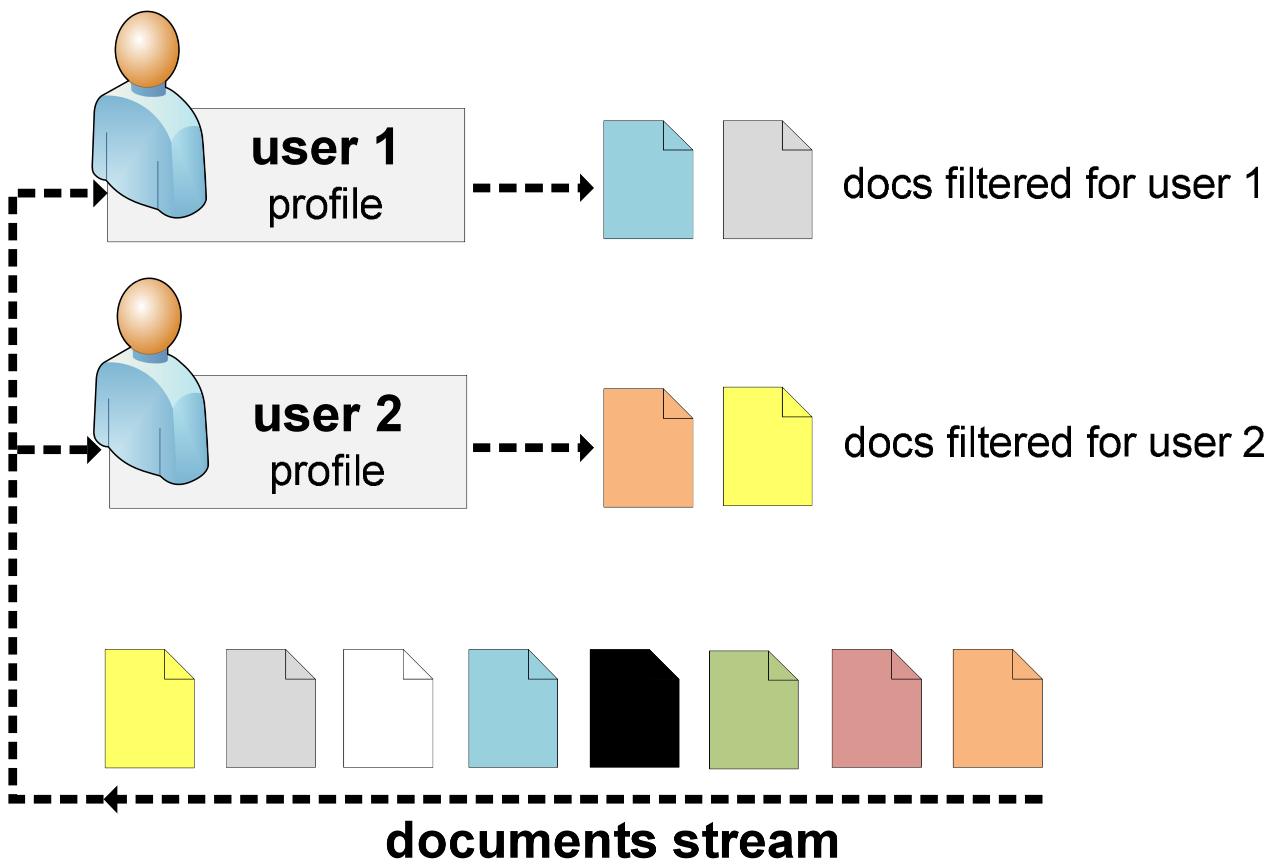
Retrieval: Ad Hoc x Filtering

- Ad Hoc Retrieval:



Retrieval: Ad Hoc x Filtering

- Filtering



Classic IR Models

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
- To quantify the importance of an index term, we associate a weight with it
- Let
  - k_i be an index term
  - d_j be a document
  - w_{ij} be a weight associated with (k_i , d_j), which quantifies the importance of k_i for describing the contents of d_j

Classic IR Models: Basic Concepts

- Let
  - k_i : i^{th} index term
  - d_j : j^{th} document
  - t : total number of terms in the vocabulary
  - $K = \{k_1, k_2, ..., k_t\}$: the set of all index terms
  - $w_{ij} \geqslant 0$: weight associated with (k_i, d_j)
    - if $w_{ij} = 0$ then term k_i does not occur within d_j
  - $\vec{d}_j = (w_{1j}, w_{2j}, ..., w_{tj})$: weighted vector associated with d_j
  - $g(\vec{d}_j)$: a reference to the weight w_{ij}

The Boolean Model

The Boolean Model

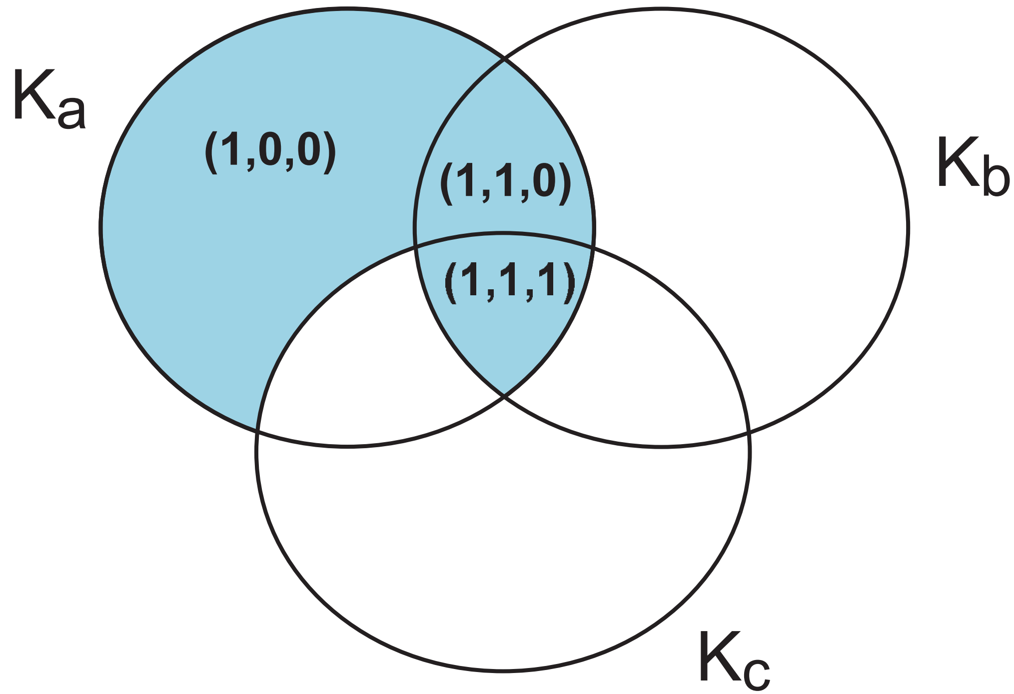
- Simple model based on set theory
- Queries specified as Boolean expressions
  - precise semantics
  - neat formalism
  - $q = k_a \wedge (k_b \vee \neg k_c)$
- Let
  - w_{iq} : weight associated with pair (k_i, q)
  - $w_{iq} \in \{0, 1\}$: terms either present or absent (Boolean)
  - $\vec{d}_q = (w_{1q}, w_{2q}, ..., w_{tq})$: weighted vector associated with q

The Boolean Model

- Let,
  - $q = k_a \wedge (k_b \vee \neg k_c)$
  - $\vec{d}_q$: weighted vector associated with q
  - $dnf(\vec{d}_q)$: distinct normal form for vector $\vec{D}_q$
- Then,
  - $dnf(\vec{d}_q) = (1,1,1) \vee (1,1,0) \vee (1,0,0)$
    - (1,1,1) : conjunctive component for (k_a, k_b, k_c)
    - (1,1,0) : conjunctive component for $(k_a, k_b, \neg k_c)$
    - (1,0,0) : conjunctive component for $(k_a, \neg k_b, \neg k_c)$
  - cc_q : a conjunctive component for $q, cc_q \in dnf(\vec{d}_q)$

The Boolean Model

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



- $sim(q, d_j) = 1$, if $\exists cc_q | \forall k_i, g_i(\vec{d_j}) = g_i(cc_q)$
- $sim(q, d_j) = 0$, 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

The Vector Model

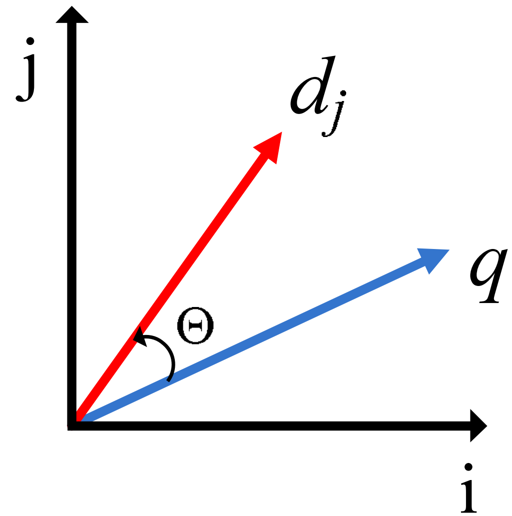
- Use of binary weights is too limiting
- Non-binary weights provide consideration for partial matches
- These term weights are used to compute a degree of similarity between a query and each document
- Ranked set of documents provides for better matching

The Vector Model

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

- Similarity



$$sim(d_j, q) = cos(\theta) = \frac{\vec{d_j} \bullet \vec{q}}{|\vec{d_j}| \times |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{t} w_{i,q}^2}}$$

- Since $w_{ij} > 0$ and $w_{iq} > 0$ then $0 \leqslant sim(d_j, q) \leqslant 1$
- A document is retrieved even if it matches the query terms only partially

The Vector Model

- Similarity

$$sim(d_j, q) = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{t} w_{i,q}^2}}$$

- 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 (dissimilarity)
    - *idf factor*, the inverse document frequency
  - $w_{i,j} = tf_{i,j} \times idf_i$

The Vector Model

- Let, $k_i \in d_j$
  - 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
  - $f_{i,j} = \frac{freq_{i,j}}{max_l \;\; freq_{l,j}}$
  - where the maximum is computed over all terms which occur within the document d_j
- The *idf factor* is computed as
  - $idf_i = \log \frac{N}{n_i}$
  - 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

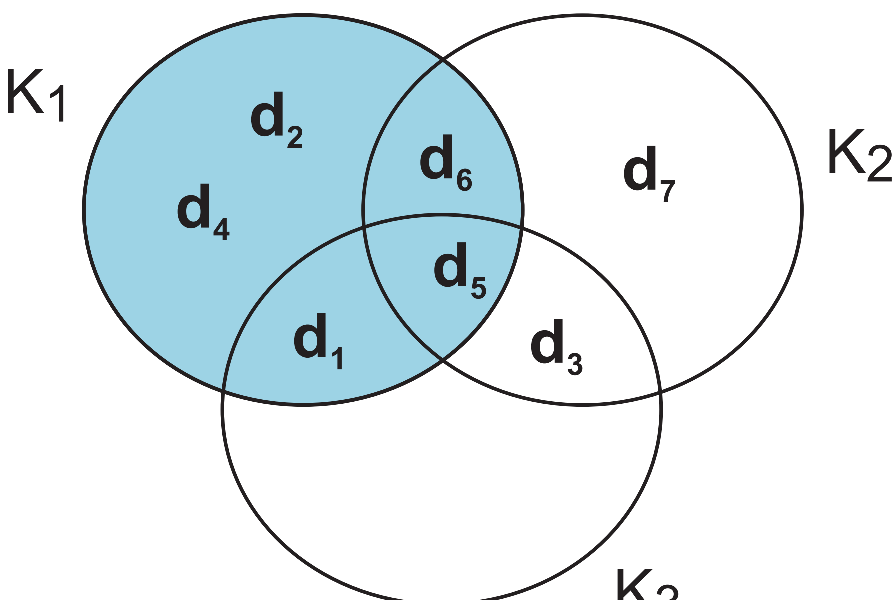
- The best term-weighting schemes use weights which are give by
  - $w_{i,j} = f_{i,j} \times \log \frac{N}{n_i}$
  - the strategy is called a tf-idf weighting scheme
- For the query term weights, a suggestion is
  - $w_{i,q} = \left(0.5 + \frac{0.5 \ freq_{i,q}}{max_l \ \ freq_{l,q}}\right) \times \log \frac{N}{n_i}$
- 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

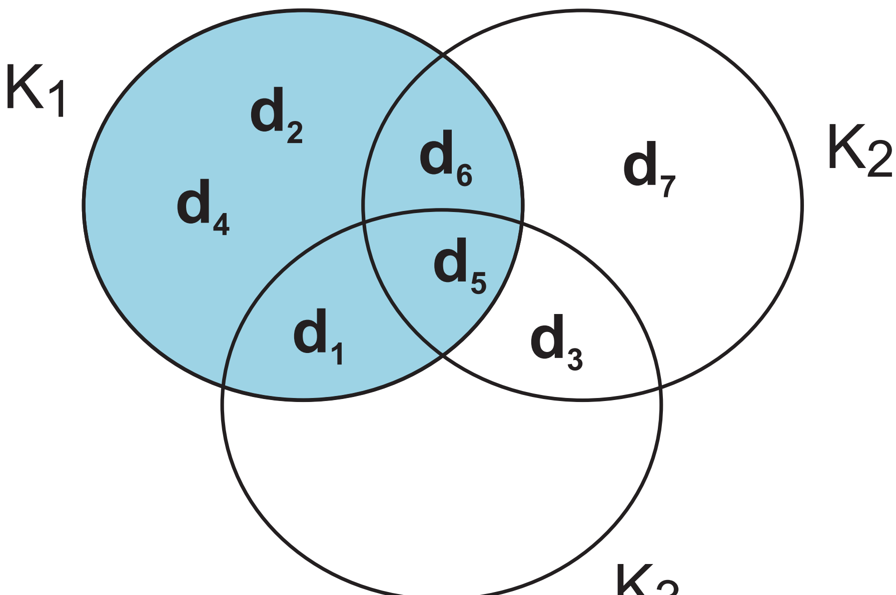
- Example 1



	K_1	K_2	K_3	$q \bullet d_j$
d_1	1	0	1	2
d_2	1	0	0	1
d_3	0	1	1	2
d_4	1	0	0	1
d_5	1	1	1	3
d_6	1	1	0	2
d_7	0	1	0	1
q	1	1	1	

The Vector Model

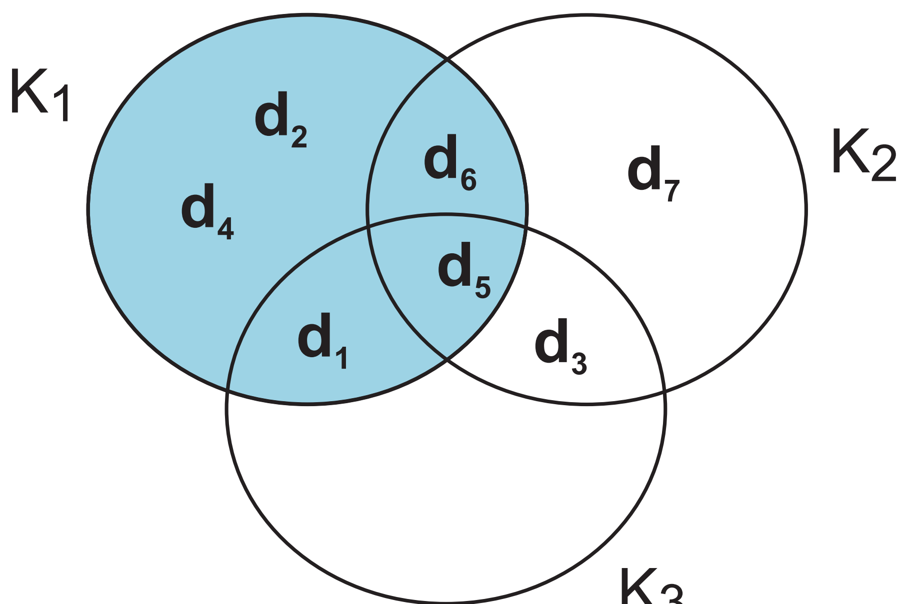
- Example 2



	K_1	K_2	K_3	$q \bullet d_j$
d_1	1	0	1	4
d_2	1	0	0	1
d_3	0	1	1	5
d_4	1	0	0	1
d_5	1	1	1	6
d_6	1	1	0	3
d_7	0	1	0	2
q	1	2	3	

The Vector Model

- Example 3



	K_1	K_2	K_3	$q \bullet d_j$
d_1	2	0	1	5
d_2	1	0	0	1
d_3	0	1	3	11
d_4	2	0	0	2
d_5	1	2	4	17
d_6	1	2	0	5
d_7	0	5	0	10
q	1	2	3	

Probabilistic Model

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)
- But, what are these properties?
- 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 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.
- But,
  - how to compute probabilities?
  - what is the sample space?

The Ranking

- Similarity

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

- $P(\vec{d}_j|R)$: probability of randomly selecting the document d_j from the set R of relevant documents
- $P(R)$: probability that a document randomly selected from the entire collection is relevant
- $P(\vec{d}_j|\overline{R})$ and $P(\overline{R})$: analogous and complementary

The Ranking

- Assuming independence of index terms

$$sim(d_j, q) \sim \frac{(\prod_{g_i(\vec{d_j})=1} P(k_i|R)) \times (\prod_{g_i(\vec{d_j})=0} P(\overline{k}_i|R))}{(\prod_{g_i(\vec{d_j})=1} P(k_i|\overline{R})) \times (\prod_{g_i(\vec{d_j})=0} P(\overline{k}_i|\overline{R}))}$$

- $P(k_i|R)$: probability that the index term k_i is present in a document randomly selected from the set R of relevant documents
- $P(\overline{k}_i|R)$: probability that the index term k_i is not present in a document randomly selected from the set R
- The probabilities associated with the $\overline{R}$ have meanings which are analogous to the ones just described

The Ranking

- Finally

$$sim(d_j, q) \sim$$

$$\sim \sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \left(\log \frac{P(k_i|R)}{1 - P(k_i|R)} + \log \frac{1 - P(k_i|\overline{R})}{P(k_i|\overline{R})} \right)$$

- Where
  - $P(\overline{k}_i|R) = 1 - P(k_i|R)$
  - $P(\overline{k}_i|\overline{R}) = 1 - P(k_i|\overline{R})$

The Initial Ranking

- Similarity $sim(d_j, q)$

$$\sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \left(\log \frac{P(k_i|R)}{1 - P(k_i|R)} + \log \frac{1 - P(k_i|\overline{R})}{P(k_i|\overline{R})} \right)$$

- Probabilities $P(k_i|R)$ and $P(k_i|\overline{R})$?
- Estimates based on assumptions:
  - $P(k_i|R) = 0.5$
  - $P(k_i|\overline{R}) = \frac{n_i}{N}$ where n_i is the number of docs that contain k_i
  - Use this initial guess to retrieve an initial ranking
  - Improve upon this initial ranking

Improving the Initial Ranking

$$\sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \left(\log \frac{P(k_i|R)}{1 - P(k_i|R)} + \log \frac{1 - P(k_i|\overline{R})}{P(k_i|\overline{R})} \right)$$

- Let
  - V : set of docs initially retrieved
  - V_i : subset of docs retrieved that contain k_i
- Reevaluate estimates:
  - $P(k_i|R) = \frac{V_i}{V}$
  - $P(k_i|\overline{R}) = \frac{n_i - V_i}{N - V}$
- Repeat recursively

Improving the Initial Ranking

$$\sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \left(\log \frac{P(k_i|R)}{1 - P(k_i|R)} + \log \frac{1 - P(k_i|\overline{R})}{P(k_i|\overline{R})} \right)$$

- To avoid problems with $V = 1$ and $V_i = 0$:
  - $P(k_i|R) = \frac{V_i + 0.5}{V+1}$
  - $P(k_i|\overline{R}) = \frac{n_i - V_i + 0.5}{N - V + 1}$
- Also,
  - $P(k_i|R) = \frac{V_i + \frac{n_i}{N}}{V+1}$
  - $P(k_i|\overline{R}) = \frac{n_i - V_i + \frac{n_i}{N}}{N - V + 1}$

Pluses and Minuses

- Advantages:
  - Docs ranked in decreasing order of probability of relevance
- Disadvantages:
  - need to guess initial estimates for $P(k_i|R)$
  - method does not take into account *tf* and *idf* factors

Comparison of Classic Models

- Boolean model does not provide for partial matches and is considered to be the weakest classic model
- Salton and Buckley did a series of experiments that indicate that, in general, the vector model outperforms the probabilistic model with general collections
- This seems also to be the view of the research community