Specifying Model

- Structure Quadruple   [D, Q, F, $R(q_i, d_j)$]
  - D = Representation of documents
  - Q = Representation of queries
  - F = Framework for modeling representations and their relationships
    - Standard language/algebra/impl. type for translation to provide semantics
    - Evaluation w.r.t. “direct” semantics through benchmarks
  - R = Ranking function that associates a real number with a query-doc pair

About index terms

- Each document represented by a set of representative keywords or index terms
  - Index terms meant to capture document’s main themes or semantics.
  - Usually, index terms are nouns because nouns have meaning by themselves.
  - However, search engines assume that all words are index terms (full text representation)

  - T1 = “conference”
  - T2 = “crime”
  - Adjectives, adverbs, conjunction, etc not useful.

Notations/Conventions

- Ki is an index term
- dj is a document
- t is the total number of docs
- $K = (k1, k2, \ldots, kt)$ is the set of all index terms
- $wij >= 0$ is the weight associated with (ki,dj)
  - $wij = 0$ if the term is not in the doc
- $vec(dj) = (w1j, w2j, \ldots, wtj)$ is the weight vector associated with the document dj
- $gi(vec(dj)) = wij$ is the function which returns the weight associated with the pair (ki,dj)

The Boolean Model

- Simple model based on set theory
- Queries and documents specified as boolean expressions
  - precise semantics
  - *E.g.,* $q = ka \wedge (kb \vee \neg kc)$
- Terms are either present or absent. Thus, $wij \; \varepsilon \; \{0,1\}$

Example

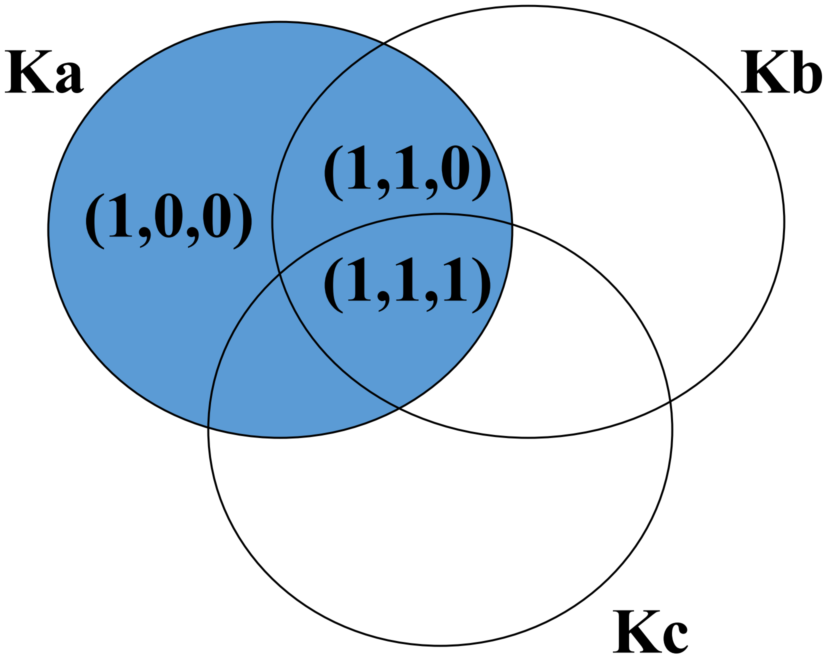
- $q = ka \wedge (kb \vee \neg kc)$
- $vec(qdnf) = (1,1,1) \vee (1,1,0) \vee (1,0,0)$
    - *Disjunctive Normal Form*
- $vec(qcc) = (1,1,0)$
    - Conjunctive component
- Similar/Matching documents
    - *md1 = [ka ka d e] => (1,0,0)*
    - *md2 = [ka kb kc] => (1,1,1)*
- Unmatched documents
    - *ud1 = [ka kc] => (1,0,1)*
    - *ud2 = [d] => (0,0,0)*

Similarity/Matching function

$$sim(q,dj) = \begin{cases} 1 & \text{if } vec(dj) \ \varepsilon \ vec(qdnf)) \\ 0 & \text{otherwise} \end{cases}$$

- *Requires coercion for accuracy*

Venn Diagram



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

Drawback of Boolean model

- Expressive power of boolean expressions to capture information need and document semantics inadequate
- Retrieval based on binary decision criteria (with no partial match) does not reflect our intuitions behind relevance adequately

- As a result
    - Answer set contains either too few or too many documents in response to a user query
    - No ranking of documents

Vector Model

- Task:
  - Document collection
  - Query specifies information need: free text
  - Relevance judgments: depends upon the weighting scheme for all docs

- Word evidence: Bag of words
  - No ordering information

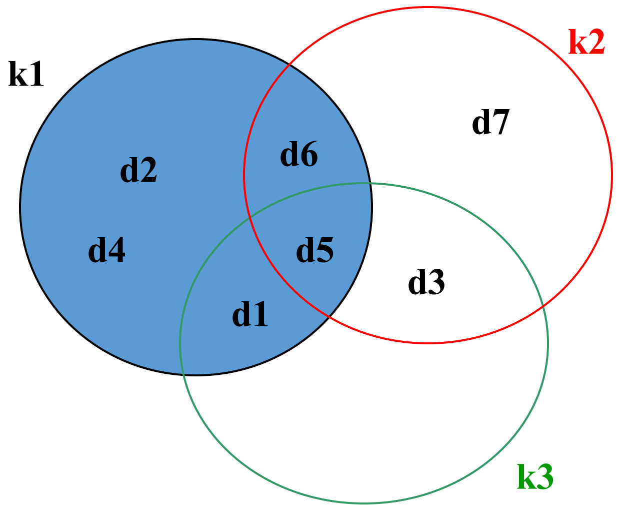
Vector Space Model

- Represent documents and queries as
  - Vectors of term-based features
    - Features: tied to occurrence of terms in collection
  - E.g. $\vec{d}_j = (t_{1,j}, t_{2,j}, ..., t_{N,j}); \vec{q}_k = (t_{1,k}, t_{2,k}, ..., t_{N,k})$

- Solution 1: Binary features: t=1 if presence, 0 otherwise
  - Similarity: number of terms in common
    - Dot product

$$sim(\vec{q}_k, \vec{d}_j) = \sum_{i=1}^{N} t_{i,k} t_{i,j}$$

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	

Vector Space Model II

- Problem: Not all terms equally interesting
  - E.g. “accuracy” vs “crime”

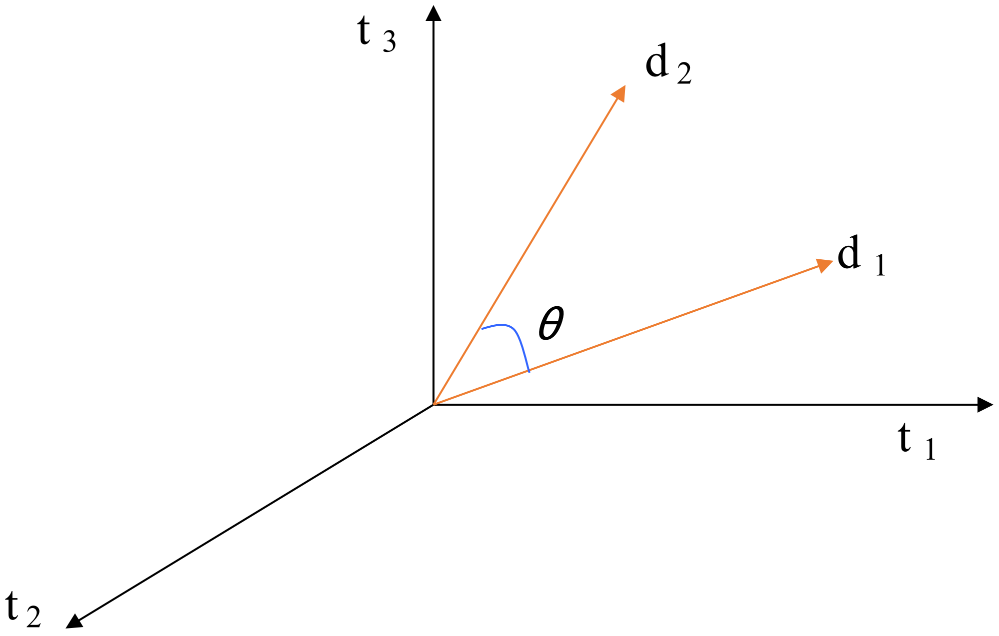
$$\vec{d}_j = (w_{1,j}, w_{2,j}, \ldots, w_{N,j}); \vec{q}_k = (w_{1,k}, w_{2,k}, \ldots, w_{N,k})$$

- Solution: Replace binary term features with weights
  - Document collection: term-by-document matrix

  - View as vector in multidimensional space
    - Nearby vectors are related
  - Normalize for vector length

Cosine similarity

- Distance between vectors d_1 and d_2 *captured* by the cosine of the angle x between them.



Queries in the vector space model

Central idea: the query as a vector:

- We regard the query as short document
  - Note that d_q is very sparse!
- We return the documents ranked by the closeness of their vectors to the query, also represented as a vector.

$$sim(d_j, d_q) = \frac{\vec{d}_j \cdot \vec{d}_q}{\left|\vec{d}_j\right|\left|\vec{d}_q\right|} = \frac{\sum_{i=1}^{n} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \sqrt{\sum_{i=1}^{n} w_{i,q}^2}}$$

Vector Similarity Computation

- Similarity = Dot product

$$sim(\vec{q}_k, \vec{d}_j) = \vec{q}_k \bullet \vec{d}_j = \sum_{i=1}^{N} w_{i,k} w_{i,j}$$

- Normalization:
  - Normalize weights in advance
  - Normalize post-hoc

$$sim(\vec{q}_k, \vec{d}_j) = \frac{\sum_{i=1}^{N} w_{i,k} w_{i,j}}{\sqrt{\sum_{i=1}^{N} w_{i,k}^2} \sqrt{\sum_{i=1}^{N} w_{i,j}^2}}$$

- Cosine of angle between two vectors
- The denominator involves the lengths of the vectors.

Computation of weights wij and wiq

- How to compute the weights wij and wiq ?
  - quantification of intra-document content (similarity/semantic emphasis)
    - *tf* factor, the *term frequency* within a document
  - quantification of inter-document separation (dis-similarity/significant discriminant)
    - *idf* factor, the *inverse document frequency*
  - *wij = tf(i,j) * idf(i)*

Weighting scheme

- 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* makes the values of *tf* and *idf* comparable.

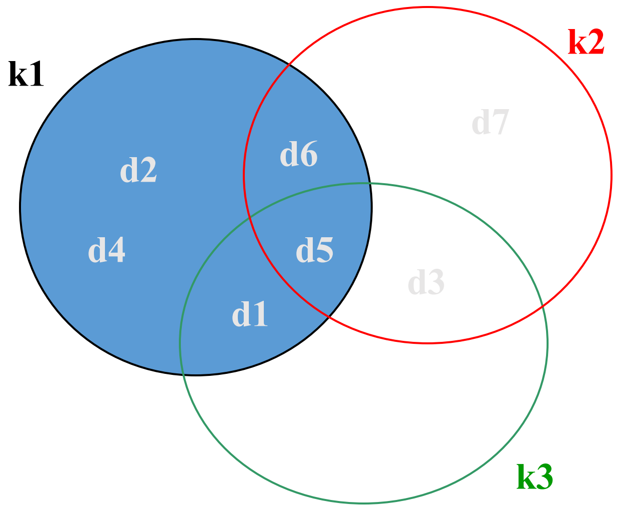
Rules:

- WARNING: In a lot of IR literature, "frequency" is used to mean "count"
  - Thus *term frequency* in IR literature is used to mean *number of occurrences* in a doc
  - Not divided by document length (which would actually make it a frequency)

Best weighting scheme

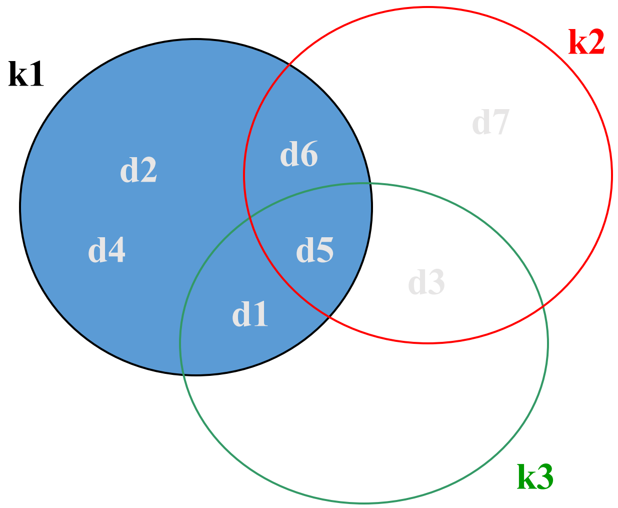
- The best term-weighting schemes use weights which are given by
  - *wij = f(i,j) * log(N/ni)*
  - the strategy is called a *tf-idf* weighting scheme
- For the query term weights, use
  - *wiq = (0.5 + [0.5 * freq(i,q) / max(freq(l,q)]) * log(N/ni)*
- The vector model with *tf-idf* weights is a good ranking strategy for general collections.
  - It is also simple and fast to compute.

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	

Example 2(Boolean model)

- Consider these documents:
- **Doc 1** breakthrough drug for schizophrenia
- **Doc 2** new schizophrenia drug
- **Doc 3** new approach for treatment of schizophrenia
- **Doc 4** new hopes for schizophrenia patients

- For the document collection, Use and depict the Boolean model and what are the Returned results for these queries:

a. schizophrenia AND drug

Example (vector model)

- Q : “gold silver truck”
- D1 : “shipment of gold damaged in a fire”
- D2 : “delivery of silver arrived in a silver truck”
- D3 : “Shipment of gold in a truck”

- Find the ranking of the document using vector space model.

Example 2

Q 1: "About modi interview in politics" (5)

D1 : "In the biggest *interview* of 2014 Arnab asks all the questions that India wanted answers from the Gandhi"

D2 : "Interview with BJP leader Narendra Modi | India Insight – Reuters"

D3 : "among all politicians *Modi* is the most polarizing *politician* in India"