The Fuzzy Set Theory

The Fuzzy Set Theory deals with the representation of classes whose boundaries are not well defined

A fuzzy subset of a u.o.d. U is characterized by a membership function $\mu_A : U \rightarrow [0,1]$ which associates with each element $u \in U$ a number $\mu_A(u)$ in the interval [0,1]

- 0 means no membership
- 1 means full membership
- 0<x<1 something in between

Commonly used operations on fuzzy sets are:

- the complement $\mu_{\bar{A}}(u) = 1 - \mu_A(u)$
- the union $\mu_{A \cup B}(u) = \max(\mu_A(u), \mu_B(u))$
- the intersection $\mu_{A \cap B}(u) = \min(\mu_A(u), \mu_B(u))$

The Fuzzy Logic Model (FLM) [Tahani,1976]

The Fuzzy Logic Model (FLM), proposed by Tahani (1976) is an extension, in a quantitative direction, of the boolean model:

- A document is represented by an and of weighted index terms;
  e.g. d_1 = (and <jazz,0.4> <sax,0.3> <Coltrane,0.5>);
- A query q_j is represented as in the Boolean Model
- The matching function RSV computes the degree to which document d_i satisfies q_j
  - $RSV(d_i,t_k) = w_{ki}$
  - $RSV(d_i,(\text{not } q_j)) = 1 - RSV(d_i,q_j)$
  - $RSV(d_i,(\text{and } q_1 .. q_n)) = \min\{RSV(d_1,q_1),..,RSV(d_i,q_n))$
  - $RSV(d_i,(\text{or } q_1 .. q_n)) = \max\{RSV(d_1,q_1),..,RSV(d_i,q_n))$

Here too there is an implicit closed word assumption: the absence of a term t in the representation of d is only a notational abbreviation for the presence of <t,0> in the same representation

Ogawa-Morita-Kobayashi implementation [1991]

- The basic idea is to expand the set of index terms in the query with related terms such that additional relevant documents can be retrieved by the user query
- A thesaurus is constructed by defining a term-term correlation matrix (keyword connection matrix)

$$c_{il} = \frac{n_{il}}{n_i + n_l - n_{il}}$$

n_i :#docs containing term k_i n_l :#docs containing term k_l

n_{il} :#docs containing both of them

- We can use this matrix to define a fuzzy set associated to each index term k_i
- In this fuzzy set, a document d_j has degree of membership

$$\mu_{ij} = 1 - \prod_{k_l \in d_j} (1 - c_{il})$$

- Given a query as a DNF $q = cc_1 \vee cc_2 \ldots \vee cc_p$

$$\mu_{qj} = \mu_{(\sum_i cc_i)j} = 1 - \prod_{i=1}^{p} (1 - \mu_{cc_i j}) = 1 - \prod_i (1 - \prod_k \mu_{cc_i^k j})$$

N.B. We have used algebraic sum (implemented as the complement of negated algebraic product) in place of the max function, and algebraic product in place of the min function, respectively.

Advantages of FLM

1. To a user familiar with Boolean systems, a fuzzy system is deifferent only in that it returns ranked output
2. If the weights $w_{ki} \in \{0,1\}$, the FLM behaves like the BM; a FLM-based IR system is thus compatible also with document representations produced for Boolean systems
3. Possibility to associate weights to index terms in the IREPs of documents
4. Output magnitude control:
   - No more under-dimensioned output, as the terms that represent a document will be many more than in a Boolean representation, even if they have weights 0. As a consequence, almost always at least one document satisfies a query with degree >0
   - No more over-dimensioned output as, thanks to ranked retrieval, the user may scan the ranked list until desired

Disadvantages of FLM

1. As in the BM, no automatic query acquisition;
2. As in the BM, no possibility to associate weights to index terms in the IREPs of queries. Extensions of the FLM to allow this are mathematically complicated
3. As in the BM, flattened, hence unintuitive, results (lack of output discrimination)! The problem derives from the fact that documents are ranked based only on the weight of a single term
   - Following query (and jazz flute Coltrane) the same RSV of 0.1 is given to document (and <jazz,0.1><flute,0.1><Coltrane,0.1><Dolphy,0.4>) and to document (and <jazz,0.1><flute,0.9>,<Coltrane,0.9><Parker,0.3>) alike;
   - Following query (or jazz flute Coltrane Dolphy) the same RSV of 0.8 is given to document (and <jazz,0.5><flute,0.5><Coltrane,0.5>,<Dolphy,0.8>) and to document (and <classical,0.1><flute,0.8><Gazzelloni,0.1>) alike.

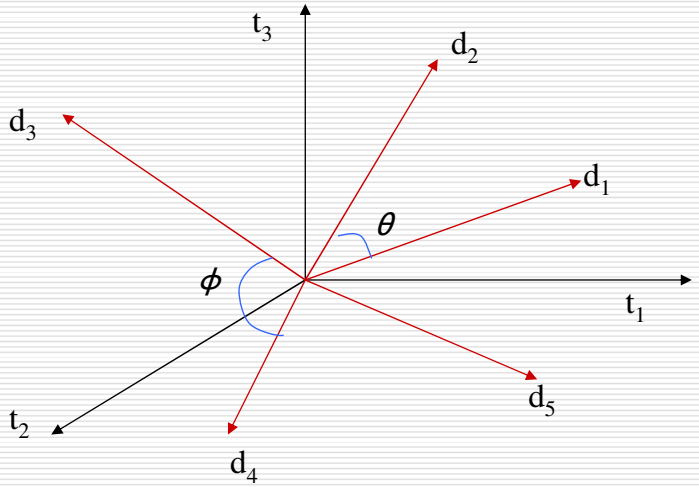
The Vector Space Model
Salton 1965

- Document d_i and query q_j are viewed as vectors in a n-dimensional space, where n is the dictionary size

- Why turn docs into vectors?

  First application: Query-by-example
  - Given a doc D, find others "like" it.
  - Now that D is a vector, find vectors (docs) "near" it.

Intuition



Postulate: Documents that are "close together" in the vector space talk about the same things.

The vector space model

Query as vector:

- We regard query as short document
- We return the documents ranked by the closeness of their vectors to the query, also represented as a vector.

Desiderata for proximity

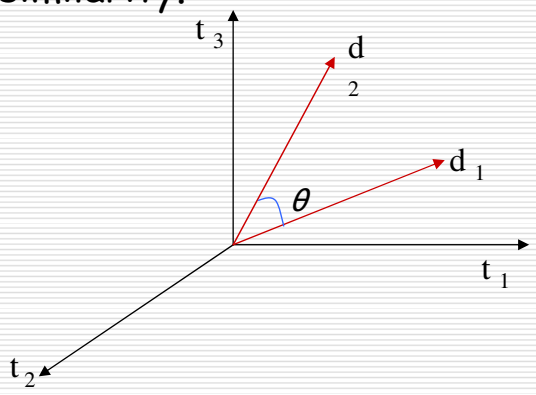
- If d_1 is near d_2, then d_2 is near d_1.
- If d_1 near d_2, and d_2 near d_3, then d_1 is not far from d_3.
- No doc is closer to d than d itself.

First cut

- Distance between d_1 and d_2 is the length of the vector $|d_1 - d_2|$.
  - Euclidean distance
- Why is this not a great idea?
- We still haven't dealt with the issue of length normalization
  - Long documents would be more similar to each other by virtue of length, not topic
- However, we can implicitly normalize by looking at *angles* instead

Cosine similarity

- Distance between vectors d_1 and d_2 *captured* by the cosine of the angle *x* between them.
- Note – this is *similarity*, not distance
  - No triangle inequality for similarity.



Cosine similarity

- A vector can be *normalized* (given a length of 1) by dividing each of its components by its length – here we use the L_2 norm $\|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2}$

$$\left|\vec{d}_j\right| = \sqrt{\sum_{i=1}^{n} w_{i,j}} = 1$$

- This maps vectors onto the unit sphere: longer documents don't get more weight