Chapter 2

Modeling

Set-Based Model
Extended Boolean Model
Fuzzy Set Model
The Generalized Vector Model
Latent Semantic Indexing
Neural Network for IR

Set Theoretic Models

Set Theoretic Models

- The Boolean model imposes a binary criterion for deciding relevance
- The question of how to extend the Boolean model to accomodate partial matching, i.e., a ranking for the documents retrieved has attracted considerable attention in the past
- We now discuss three alternative set theoretic models:
  - Set-Based Model
  - Extended Boolean Model
  - Fuzzy Set Model

Set Theoretic Models

Set-Based Model

Set-Based Model

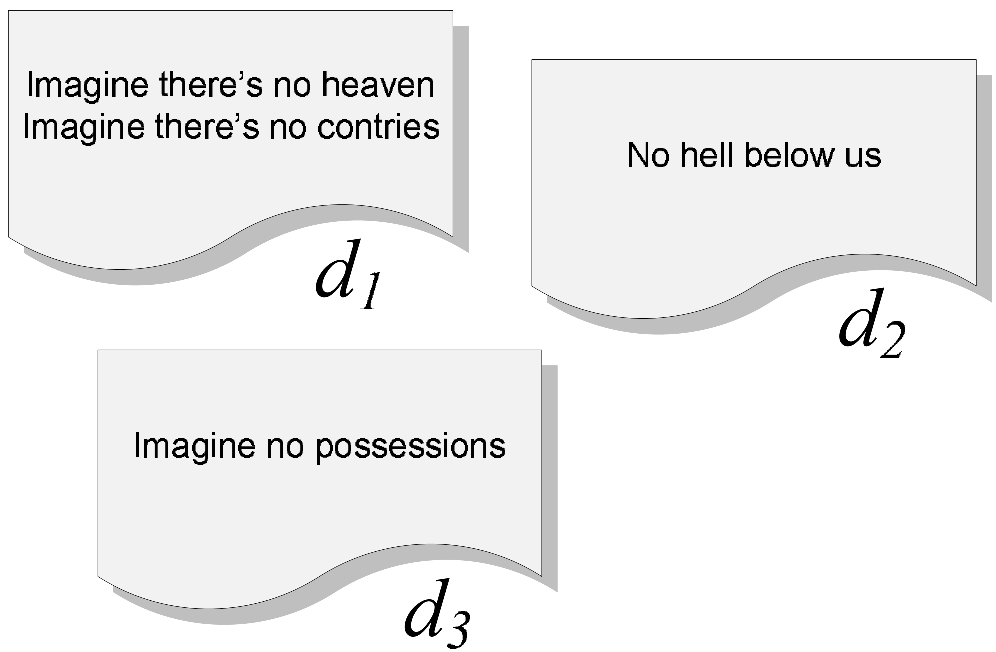
- A more recent approach (2005) that combines set theory with vectorial ranking
- The fundamental idea is to use mutual dependencies among index terms to improve results
  - Term dependencies is captured through the introduction of **termsets**, which are sets of correlated terms
- The approach leads to improved results with various collections
- The first IR model that effectively took advantage of term dependence, while keeping computational costs low

Termsets

- **Termset** is a concept used in place of the index terms
- A termset $S_i = \{k_a, k_b, ..., k_n\}$ is a subset of the terms in the collection
- If all index terms in S_i occur in a document d_j then we say that the termset S_i occurs in d_j
- We call N_i the number of documents in which S_i occurs
- There are 2^t termsets that might occur in the documents of the collection
  - However, it is common that the actual number of termsets in the collection is far smaller than 2^t, because most combinations of terms have no semantic meaning

Termsets

- Let t be the number of terms of the collection
- Then, the set $V_S = \{S_1, S_2, ..., S_{2^t}\}$, formed by all termsets in the collection, is the **vocabulary-set** of the collection
- To illustrate, consider the document collection below

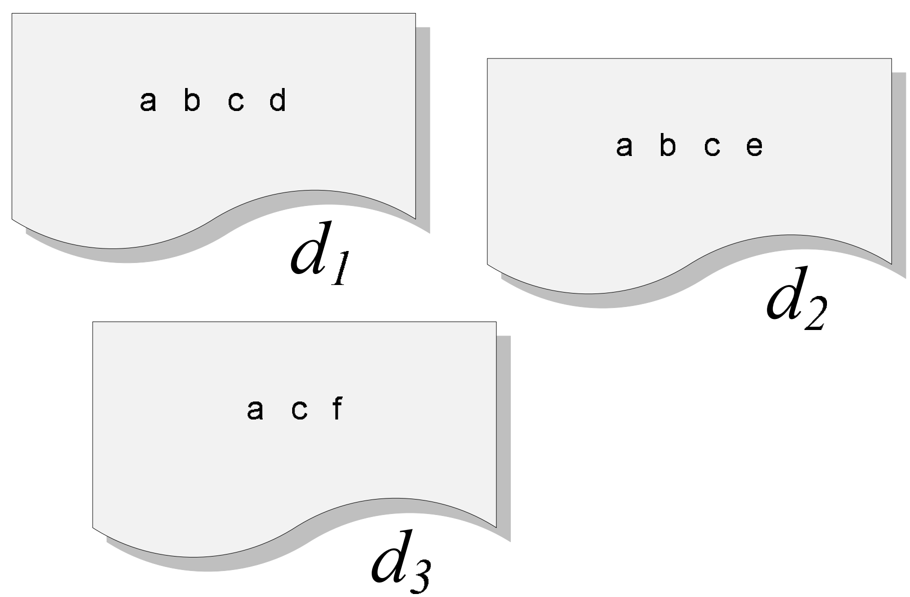


Termsets

- To simplify notation, let us define

k_a = imagine	k_d = heaven
k_b = there's	k_e = countries
k_c = no	k_f = possessions

- Further, let the letters $a...f$ refer to the index terms $k_a...k_f$, respectively



Termsets

- Consider the query q as *imagine there's no possessions*, i.e. $q = \{a, b, c, f\}$. For this query, the vocabulary-set is as below

Termset	Set of Terms	Documents
S_a	$\{a\}$	$\{d_1, d_2, d_3\}$
S_b	$\{b\}$	$\{d_1, d_2\}$
S_c	$\{c\}$	$\{d_1, d_2, d_3\}$
S_f	$\{f\}$	$\{d_3\}$
S_{ab}	$\{a, b\}$	$\{d_1, d_2\}$
S_{ac}	$\{a, c\}$	$\{d_1, d_2, d_3\}$
S_{af}	$\{a, f\}$	$\{d_3\}$
S_{bc}	$\{b, c\}$	$\{d_1, d_2\}$
S_{cf}	$\{c, f\}$	$\{d_3\}$
S_{abc}	$\{a, b, c\}$	$\{d_1, d_2\}$
S_{acf}	$\{a, c, f\}$	$\{d_3\}$

Notice that there are 11 termsets that occur in our collection, out of the maximum of 15 termsets that can be formed with the terms in q

Termsets

- At query processing time, only the termsets generated by the query need to be considered
  - For short queries, this is reasonably efficient
  - For long queries, many more termsets might need to be computed and taken into account
- To reducing the number of termsets, we can consider only those with a certain frequency in the collection
- A termset composed of n terms is called an n-termset
- An n-termset S_i is said to be frequent if its N_i value is greater than or equal to a given threshold
  - This implies that an n-termset is frequent if and only if all of its $(n-1)$-termsets are also frequent

Ranking Computation

- Let the threshold on the frequency of termsets be 2
- To compute all frequent termsets for the query $q = \{a, b, c, f\}$ we proceed as follows.

1. Compute the frequent 1-termsets and their inverted lists:
   - $S_a = \{d_1, d_2, d_3\}$
   - $S_b = \{d_1, d_2\}$
   - $S_c = \{d_1, d_2, d_3\}$
2. Combine the inverted lists to compute frequent 2-termsets:
   - $S_{ab} = \{d_1, d_2\}$
   - $S_{ac} = \{d_1, d_2\}$
   - $S_{bc} = \{d_1, d_2\}$
3. Combine the inverted lists to compute frequent 3-termsets:
   - $S_{abc} = \{d_1, d_2\}$

Ranking Computation

- Notice that there are only 7 frequent termsets in our collection
- We can efficiently compute the inverted lists for frequent n-termsets by starting with the inverted lists of frequent 1-termsets
- This is reasonably fast for short queries up to 4-5 terms

Ranking Computation

- The ranking computation is based on the Vector Space model, using termsets instead of index terms
- Given a query q, specified as a set of index terms, let
  - $\{S_1, S_2, \ldots\}$ be the set of all termsets originated from q
  - $\mathcal{N}_i$ be the number of documents in which termset S_i occurs
  - N be the total number of documents in the collection
  - $\mathcal{F}_{i,j}$ be the frequency of termset S_i in document d_j
- For each pair $[S_i, d_j]$ we compute a weight $\mathcal{W}_{i,j}$ given by

$$\mathcal{W}_{i,j} = (1 + \log \mathcal{F}_{i,j}) \log(1 + \frac{N}{\mathcal{N}_i})$$

- We also compute a $\mathcal{W}_{i,q}$ value for each pair $[S_i, q]$

Ranking Computation

- The weights of interest considering the query $q = \{a, b, c, f\}$ and the document d_3 are (assuming minimum threshold frequency of 1)

Termset	Weight	
S_a	$\mathcal{W}_{a,3}$	$1 * \log 3/3 = 0$
S_b	$\mathcal{W}_{b,3}$	$0 * \log 3/2 = 0$
S_c	$\mathcal{W}_{c,3}$	$1 * \log 3/3 = 0$
S_f	$\mathcal{W}_{f,3}$	$1 * \log 3/1 = 1.58$
S_{ab}	$\mathcal{W}_{ab,3}$	$0 * \log 3/2 = 0$
S_{ac}	$\mathcal{W}_{ac,3}$	$1 * \log 3/3 = 0$
S_{af}	$\mathcal{W}_{af,3}$	$1 * \log 3/1 = 1.58$
S_{bc}	$\mathcal{W}_{bc,3}$	$0 * \log 3/2 = 0$
S_{cf}	$\mathcal{W}_{cf,3}$	$1 * \log 3/1 = 1.58$
S_{abc}	$\mathcal{W}_{abc,3}$	$0 * \log 3/2 = 0$
S_{acf}	$\mathcal{W}_{acf,3}$	$1 * \log 3/1 = 1.58$

There are only 4 termsets that need to be taken into account for ranking d_3 in this case

Ranking Computation

- A document d_j and a query q are represented as vectors in a 2^t-dimensional space of termsets, as follows

$$
\begin{aligned}
\vec{d_j} &= (\mathcal{W}_{1,j}, \mathcal{W}_{2,j}, \ldots, \mathcal{W}_{2^t,j}) \\
\vec{q} &= (\mathcal{W}_{1,q}, \mathcal{W}_{2,q}, \ldots, \mathcal{W}_{2^t,q})
\end{aligned}
$$

- The rank of d_j to the query q is computed as follows

$$
sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q}}{|\vec{d_j}| \times |\vec{q}|} = \frac{\sum_{S_i} \mathcal{W}_{i,j} \times \mathcal{W}_{i,q}}{|\vec{d_j}| \times |\vec{q}|}
$$

- For termsets that are not in the query q, $\mathcal{W}_{i,q} = 0$

Ranking Computation

- The document norm $|\vec{d_j}|$ is hard to compute in the space of termsets
- Then, its computation is restricted to 1-termsets
- Let again $q = \{a, b, c, f\}$ and d_3
  - We need to consider four termsets: S_f, S_{af}, S_{cf}, and S_{acf}
  - The document norm in terms of 1-termsets is given by

$$|\vec{d_3}| = \sqrt{\mathcal{W}_{a,3}^2 + \mathcal{W}_{b,3}^2 + \mathcal{W}_{c,3}^2} = \sqrt{0 + 0 + 1.58^2} = 1.58$$

  - The rank of d_3 ($sim(d_3, q)$) is then given by

$$\frac{\mathcal{W}_{f,3} * \mathcal{W}_{f,q} + \mathcal{W}_{af,3} * \mathcal{W}_{af,q} + \mathcal{W}_{cf,3} * \mathcal{W}_{cf,q} + \mathcal{W}_{acf,3} * \mathcal{W}_{acf,q}}{1.58}$$

$$\frac{1.58 * 1.58 + 1.58 * 1.58 + 1.58 * 1.58 + 1.58 * 1.58}{1.58} = 6.32$$

Closed Termsets

- The concept of frequent termsets allows restricting the ranking computation to those termsets that occur in the collection with a minimum threshold frequency
- Yet, there are many frequent termsets in a large collection.
  - This is troublesome because the number of termsets to consider might be prohibitively high with large queries.
- To resolv this problem, we can further restrict the ranking computation to a smaller number of termsets
  - This can be accomplished by observing some properties of termsets such as the notion of closure

Closed Termsets

- The **closure of a termset** S_i is the set of all frequent termsets that co-occur with S_i in the same set of documents.
- Given the closure of S_i, the largest termset in it is called a **closed termset** and is referred to as Φ_i.
- To formalize, let
  - $D \subseteq C$ be a subset of all documents in the collection C;
  - $S(D)$ be a set composed of the termsets that occur in all documents in D;
  - $l(S_i)$ be the set of documents in which S_i occurs, i.e., the (document) inverted list of termset S_i.

Closed Termsets

- Then, the closed termset Φ_i satisfies the following property

$$\nexists S_j \in S(D) \mid \Phi_i \subset S_j \wedge l(\Phi_i) = l(S_j)$$

- Closed termsets encapsulate smaller termsets occuring in the same set of documents
- As a result, their number is at most equal to that of frequent termsets
- Thus, if we restrict the ranking computation to closed termsets, we can frequently expect a reduction in query processing time

Fuzzy Set Model

Fuzzy Set Model

- Queries and docs represented by sets of index terms: matching is *approximate* from the start
- This *vagueness* can be modeled using a fuzzy framework, as follows:
  - with each term is associated a *fuzzy* set
  - each doc has a degree of membership in this fuzzy set
- This interpretation provides the foundation for many IR models based on fuzzy theory
- In here, we discuss the model proposed by Ogawa, Morita, and Kobayashi (1991)

Fuzzy Set Theory

- Framework for representing classes whose boundaries are not well defined
- Key idea is to introduce the notion of a *degree of membership* associated with the elements of a set
- This degree of membership varies from 0 to 1 and allows modeling the notion of *marginal* membership
- Thus, membership is now a *gradual* notion, contrary to the crispy notion enforced by classic Boolean logic

Fuzzy Set Theory

- Definition
  - A fuzzy subset A of a universe of discourse U is characterized by a membership function $\mu_A : U \rightarrow [0,1]$ which associates with each element u of U a number $\mu_A(u)$ in the interval [0,1].
- Definition
  - Let U be the universe of discourse, A and B be two fuzzy subsets of U, and $\overline{A}$ be the complement of A relative to U. Also, let u be an element of U. Then,

$$\begin{aligned}
\mu_{\overline{A}}(u) &= 1 - \mu_A(u) \\
\mu_{A \cup B}(u) &= max(\mu_A(u), \mu_B(u)) \\
\mu_{A \cap B}(u) &= min(\mu_A(u), \mu_B(u))
\end{aligned}$$

Fuzzy Information Retrieval

- Fuzzy sets are modeled based on a thesaurus
- This thesaurus is built as follows:
  - Let $\vec{c}$ be a term-term correlation matrix
  - Let $c_{i,l}$ be a normalized correlation factor between two terms k_i and k_l:

$$c_{i,l} = \frac{n_{i,l}}{n_i + n_l - n_{i,l}}$$

n_i: number of docs which contain k_i
n_l: number of docs which contain k_l
$n_{i,l}$: number of docs which contain both k_i and k_l

- We now have the notion of *proximity* among index terms.

Fuzzy Information Retrieval

- The correlation factor $c_{i,l}$ can be used to define fuzzy set membership for a document d_j as follows:

$$\mu_{i,j} = 1 - \prod_{k_l \,\in\, d_j} (1 - c_{i,l})$$

  $\mu_{i,j}$: membership of doc d_j in fuzzy subset associated with k_i

- The above expression computes an algebraic sum over all terms in d_j
- A document d_j belongs to the fuzzy set associated with k_i, if its own terms are associated with k_i

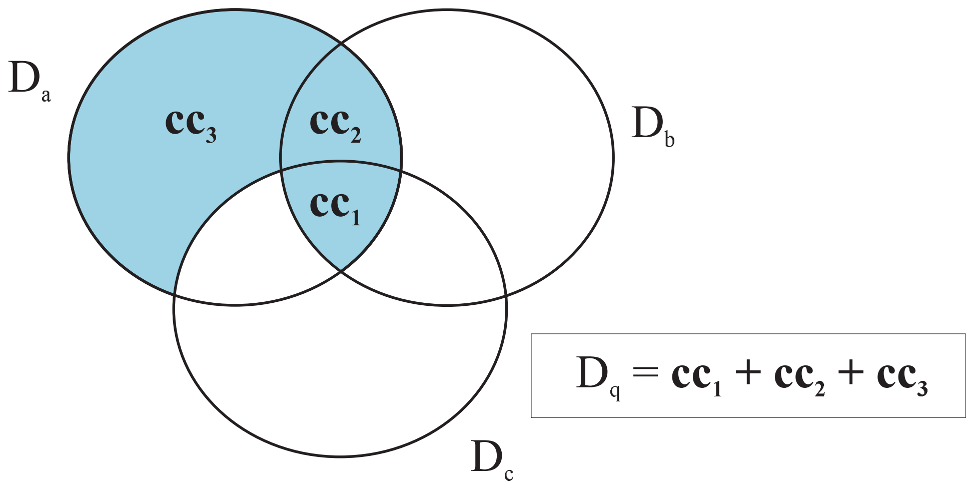
Fuzzy Information Retrieval

- If d_j contains a term k_l which is closely related to k_i, we have
  - $c_{i,l} \sim 1$
  - $\mu_{i,j} \sim 1$
  - and k_i is a good fuzzy index for d_j

$$\mu_{i,j} = 1 - \prod_{k_l \in d_j} (1 - c_{i,l})$$

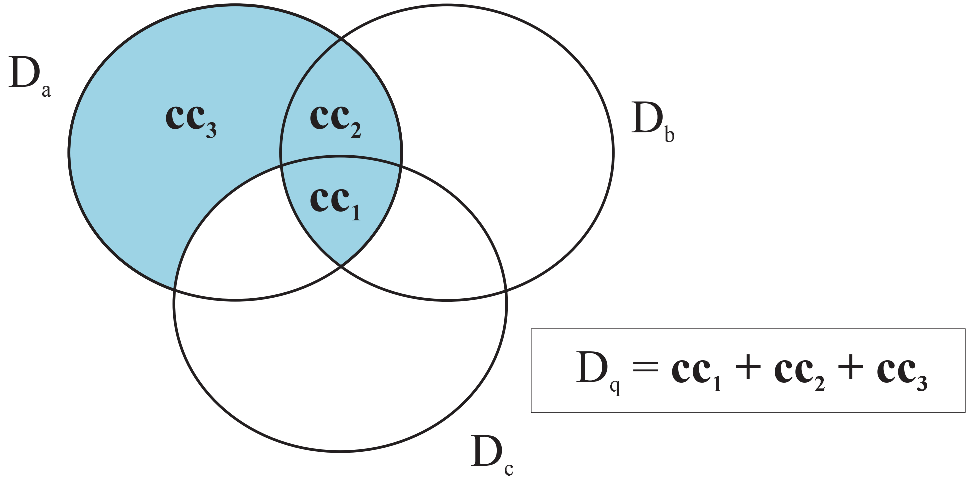
$\mu_{i,j}$: membership of doc d_j in fuzzy subset associated with k_i

Fuzzy IR: An Example



- $q = k_a \wedge (k_b \vee \neg k_c)$
- Disjunct normal form is given by
  $\vec{q}_{dnf} = (1,1,1) + (1,1,0) + (1,0,0) = cc_1 + cc_2 + cc_3$

Fuzzy IR: An Example



$$
\begin{aligned}
\mu_{q,j} &= \mu_{cc_1+cc_2+cc_3,j} \\
&= 1 - \prod_{i=1}^{3}(1 - \mu_{cc_i,j}) \\
&= 1 - (1 - \mu_{a,j}\mu_{b,j}\mu_{c,j}) \times \\
&\quad (1 - \mu_{a,j}\mu_{b,j}(1 - \mu_{c,j})) \times (1 - \mu_{a,j}(1 - \mu_{b,j})(1 - \mu_{c,j}))
\end{aligned}
$$

Fuzzy Information Retrieval

- Fuzzy IR models have been discussed mainly in the literature associated with fuzzy theory
- Experiments with standard test collections are not available
- Difficult to compare at this time

Extended Boolean Model

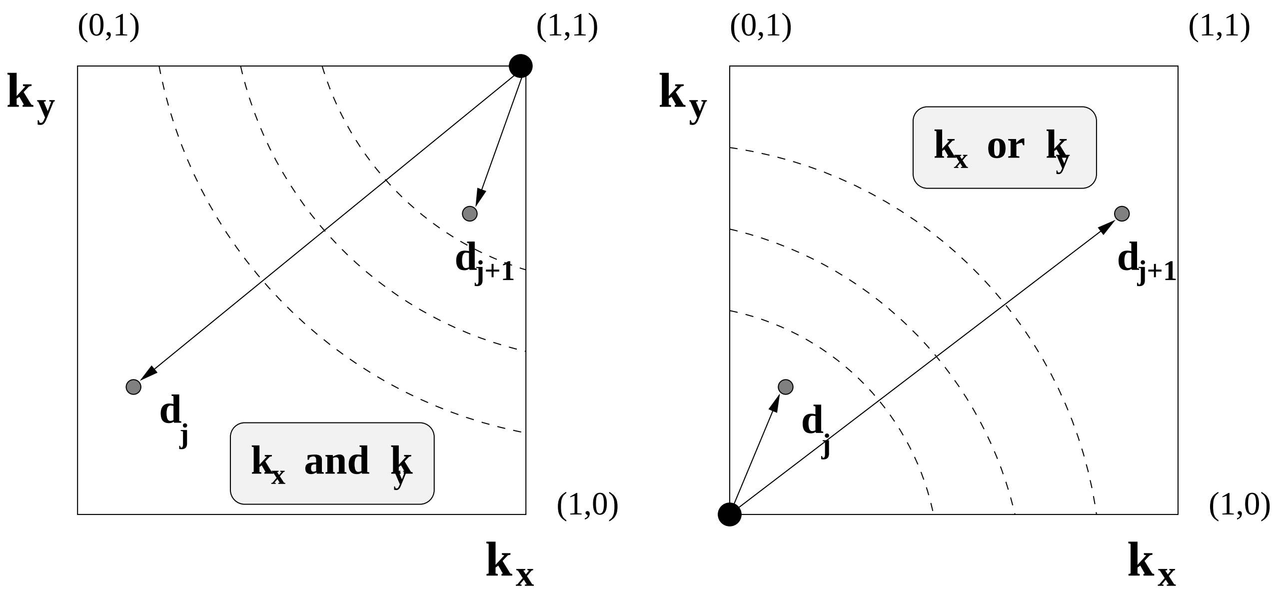
Extended Boolean Model

- Boolean model is simple and elegant
- But, no provision for a ranking
- As with the fuzzy model, a ranking can be obtained by relaxing the condition on set membership
- Extend the Boolean model with the notions of partial matching and term weighting
- Combine characteristics of the Vector model with properties of Boolean algebra

The Idea

- The extended Boolean model (introduced by Salton, Fox, and Wu, 1983) is based on a critique of a basic assumption in Boolean algebra
- Let,
  - $q = k_x \wedge k_y$
  - $w_{x,j} = f_{x,j} \times \frac{idf_x}{max_i \; idf_i}$
  - $w_{x,j}$ is a weight associated with the pair $[k_x, d_j]$
- To simplify notation, let
  - $w_{x,j} = x$ and $w_{y,j} = y$

The Idea



$$
\begin{aligned}
sim(q_{or}, d) &= \sqrt{\frac{x^2 + y^2}{2}} \\
sim(q_{and}, d) &= 1 - \sqrt{\frac{(1-x)^2 + (1-y)^2}{2}}
\end{aligned}
$$

Generalizing the Idea

- We can extend the previous model to consider Euclidean distances in a t-dimensional space
- This can be done using *p-norms* which extend the notion of distance to include p-distances, where $1 \leq p \leq \infty$ is a new parameter
- A generalized conjunctive query is given by
  - $q_{and} = k_1 \ \wedge^p \ k_2 \ \wedge^p \ \ldots \wedge^p k_m$
- A generalized disjunctive query is given by
  - $q_{or} = k_1 \ \vee^p \ k_2 \ \vee^p \ \ldots \vee^p k_m$

Generalizing the Idea

- The query-document similarities are now given by

$$sim(q_{or}, d_j) = \left(\frac{x_1^p + x_2^p + \ldots + x_m^p}{m} \right)^{\frac{1}{p}}$$

$$sim(q_{and}, d_j) = 1 - \left(\frac{(1 - x_1)^p + (1 - x_2)^p + \ldots + (1 - x_m)^p}{m} \right)^{\frac{1}{p}}$$

- where each x_i stands for the weight $w_{i,d}$ associated to the pair $[k_i, d_j]$

Properties

- $sim(q_{or}, d_j) = \left(\frac{x_1^p + x_2^p + ... + x_m^p}{m}\right)^{\frac{1}{p}}$
- $sim(q_{and}, d_j) = 1 - \left(\frac{(1-x_1)^p + (1-x_2)^p + ... + (1-x_m)^p}{m}\right)^{\frac{1}{p}}$
- If $p = 1$ then (vector-like)
  - $sim(q_{or}, d_j) = sim(q_{and}, d_j) = \frac{x_1 + ... + x_m}{m}$
- If $p = \infty$ then (Fuzzy like)
  - $sim(q_{or}, d_j) = max(x_i)$
  - $sim(q_{and}, d_j) = min(x_i)$

Properties

- By varying p, we can make the model behave as a vector, as a fuzzy, or as an intermediary model
- This is quite powerful and is a good argument in favor of the extended Boolean model
- $q = (k_1 \wedge^p k_2) \vee^p k_3$
  - k_1 and k_2 are to be used as in a vectorial retrieval while the presence of k_3 is required
- $sim(q, d) = \left(\frac{\left(1 - \left(\frac{(1-x_1)^p + (1-x_2)^p}{2}\right)^{\frac{1}{p}}\right)^p + x_3^p}{2} \right)^{\frac{1}{p}}$

Conclusions

- Model is quite powerful
- Properties are interesting and might be useful
- Computation is somewhat complex
- However, distributivity operation does not hold for ranking computation:
  - $q_1 = (k_1 \ \vee \ k_2) \ \wedge \ k_3$
  - $q_2 = (k_1 \ \wedge \ k_3) \ \vee \ (k_2 \ \wedge \ k_3)$
  - $sim(q_1, d_j) \neq sim(q_2, d_j)$

Algebraic Models

Generalized Vector Model

Generalized Vector Model

- Classic models enforce independence of index terms
- For the Vector model:
  - Set of term vectors $\{\vec{k}_1, \vec{k}_2, \ldots, \vec{k}_t\}$ are linearly independent and form a basis for the subspace of interest
- Frequently, this is interpreted as:
  - $\forall_{i,j} \Rightarrow \vec{k}_i \bullet \vec{k}_j = 0$
- In 1985, Wong, Ziarko, and Wong proposed an interpretation in which the set of terms is linearly independent, but not pairwise orthogonal

Key Idea

- In the generalized vector model, two index terms might be non-orthogonal and are represented in terms of smaller components (minterms)
- As before let,
  - $w_{i,j}$ be the weight associated with $[k_i, d_j]$
  - $\{k_1, k_2, \ldots, k_t\}$ be the set of all terms
- If these weights are all binary, all patterns of occurrence of terms within documents can be represented by the minterms:
  - $m_1 = (0, 0, \ldots, 0)$, $m_2 = (1, 0, \ldots, 0)$, $\ldots$, $m_{2^t} = (1, 1, \ldots, 1)$
  - In here, m_2 indicates documents in which solely the term k_1 occurs

Key Idea

- The basis for the generalized vector model is formed by a set of 2^t vectors defined over the set of minterms, as follows:

$$\begin{aligned}
\vec{m}_1 &= (1, 0, \ldots, 0, 0) \\
\vec{m}_2 &= (0, 1, \ldots, 0, 0) \\
&\vdots \\
\vec{m}_{2^t} &= (0, 0, \ldots, 0, 1)
\end{aligned}$$

- Notice that,
  - $\forall_{i,j} \Rightarrow \vec{m}_i \bullet \vec{m}_j = 0$ i.e., pairwise orthogonal

Key Idea

- Minterm vectors are pairwise orthogonal. But, this does not mean that the index terms are independent:
  - The minterm m_4 is given by:
    $m_4 = (1, 1, 0, \ldots, 0)$
  - This minterm indicates the occurrence of the terms k_1 and k_2 within a same document. If such document exists in a collection, we say that the minterm m_4 is active and that a dependency between these two terms is induced
  - The generalized vector model adopts as a basic foundation the notion that co-occurence of terms within documents induces dependencies among them

Forming the Term Vectors

- Let $g_i(m_j)$ return the weight $\{0,1\}$ of the index term k_i in the minterm m_j. The vector associated with the term k_i is computed as:

$$\vec{k}_i = \frac{\sum_{\forall r,\ g_i(m_r)=1} c_{i,r}\ \vec{m}_r}{\sqrt{\sum_{\forall r,\ g_i(m_r)=1}\ c_{i,r}^2}}$$

$$c_{i,r} = \sum_{d_j \mid g_l(\vec{d}_j)=g_l(m_r)\ for\ all\ l} w_{i,j}$$

- The weight $c_{i,r}$ associated with the pair $[k_i, m_r]$ sums up the weights of the term k_i in all the documents which have a term occurrence pattern given by m_r.
- Notice that for a collection of size N, only N minterms affect the ranking (and not 2^t)

Dependency between Index Terms

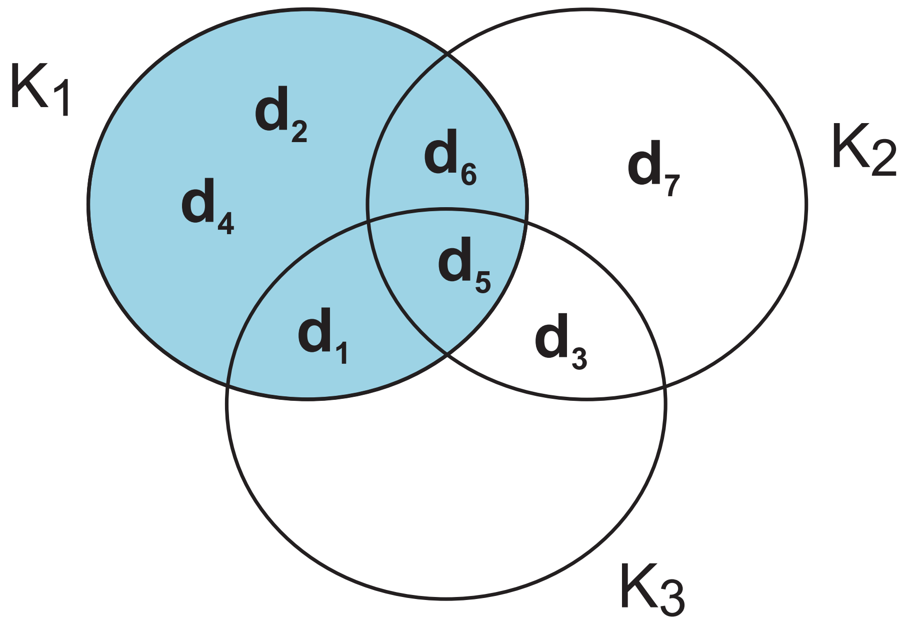
- A degree of correlation between the terms k_i and k_j can now be computed as:

$$\vec{k}_i \bullet \vec{k}_j = \sum_{\forall r \mid g_i(m_r)=1 \wedge g_j(m_r)=1} c_{i,r} \times c_{j,r}$$

- This degree of correlation sums up (in a weighted form) the dependencies between k_i and k_j induced by the documents in the collection (represented by the m_r minterms).

The Generalized Vector Model

- An Example



	K_1	K_2	K_3
d_1	2	0	1
d_2	1	0	0
d_3	0	1	3
d_4	2	0	0
d_5	1	2	4
d_6	1	2	0
d_7	0	5	0
q	1	2	3

Computation of $c_{i,r}$

	K_1	K_2	K_3
d_1	2	0	1
d_2	1	0	0
d_3	0	1	3
d_4	2	0	0
d_5	1	2	4
d_6	0	2	2
d_7	0	5	0
q	1	2	3

	K_1	K_2	K_3
$d_1 = m_6$	1	0	1
$d_2 = m_2$	1	0	0
$d_3 = m_7$	0	1	1
$d_4 = m_2$	1	0	0
$d_5 = m_8$	1	1	1
$d_6 = m_7$	0	1	1
$d_7 = m_3$	0	1	0
$q = m_8$	1	1	1

	$c_{1,r}$	$c_{2,r}$	$c_{3,r}$
m_1	0	0	0
m_2	3	0	0
m_3	0	5	0
m_4	0	0	0
m_5	0	0	0
m_6	2	0	1
m_7	0	3	5
m_8	1	2	4

Computation of $\overrightarrow{k_i}$

- $\overrightarrow{k_1} = \frac{(3m_2+2m_6+m_8)}{\sqrt{3^2+2^2+1^2}}$
- $\overrightarrow{k_2} = \frac{(5m_3+3m_7+2m_8)}{\sqrt{5+3+2}}$
- $\overrightarrow{k_3} = \frac{(1m_6+5m_7+4m_8)}{\sqrt{1+5+4}}$

	$c_{1,r}$	$c_{2,r}$	$c_{3,r}$
m_1	0	0	0
m_2	3	0	0
m_3	0	5	0
m_4	0	0	0
m_5	0	0	0
m_6	2	0	1
m_7	0	3	5
m_8	1	2	4

Computation of Document Vectors

- $\overrightarrow{d_1} = 2\overrightarrow{k_1} + \overrightarrow{k_3}$
- $\overrightarrow{d_2} = \overrightarrow{k_1}$
- $\overrightarrow{d_3} = \overrightarrow{k_2} + 3\overrightarrow{k_3}$
- $\overrightarrow{d_4} = 2\overrightarrow{k_1}$
- $\overrightarrow{d_5} = \overrightarrow{k_1} + 2\overrightarrow{k_2} + 4\overrightarrow{k_3}$
- $\overrightarrow{d_6} = 2\overrightarrow{k_2} + 2\overrightarrow{k_3}$
- $\overrightarrow{d_7} = 5\overrightarrow{k_2}$
- $\overrightarrow{q} = \overrightarrow{k_1} + 2\overrightarrow{k_2} + 3\overrightarrow{k_3}$

	K_1	K_2	K_3
d_1	2	0	1
d_2	1	0	0
d_3	0	1	3
d_4	2	0	0
d_5	1	2	4
d_6	0	2	2
d_7	0	5	0
q	1	2	3

Conclusions

- Model considers correlations among index terms
- Not clear in which situations it is superior to the standard Vector model
- Computation costs are higher
- Model does introduce interesting new ideas

Latent Semantic Indexing

Latent Semantic Indexing

- Classic IR might lead to poor retrieval due to:
  - unrelated documents might be included in the answer set
  - relevant documents that do not contain at least one index term are not retrieved
  - **Reasoning**: retrieval based on index terms is vague and noisy
- The user information need is more related to concepts and ideas than to index terms
- A document that shares concepts with another document known to be relevant might be of interest

Latent Semantic Indexing

- The key idea is to map documents and queries into a lower dimensional space (i.e., composed of higher level concepts which are in fewer number than the index terms)
- Retrieval in this reduced concept space might be superior to retrieval in the space of index terms

Latent Semantic Indexing

- Definitions
  - Let t be the total number of index terms
  - Let N be the number of documents
  - Let $\vec{M}$=(M_{ij}) be a term-document matrix with t rows and N columns
  - To each element of this matrix is assigned a weight $w_{i,j}$ associated with the pair $[k_i, d_j]$
  - The weight $w_{i,j}$ can be based on a *tf-idf* weighting scheme

Latent Semantic Indexing

- The matrix $\vec{M}$=(M_{ij}) can be decomposed into 3 matrices (singular value decomposition) as follows:
  -
$$\vec{M} = \vec{K}\vec{S}\vec{D}^t$$
  - $\vec{K}$ is the matrix of eigenvectors derived from $\vec{M}\vec{M}^t$
  - $\vec{D}^t$ is the matrix of eigenvectors derived from $\vec{M}^t\vec{M}$
  - $\vec{S}$ is an $r \times r$ diagonal matrix of singular values where
    - $r = min(t, N)$ that is, the rank of $\vec{M}$

Computing an Example

- Let $\vec{M}=(M_{ij})$ be given by the matrix

	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	

- Compute the matrices $\vec{K}$, $\vec{S}$, and $\vec{D}^t$

Latent Semantic Indexing

- In the matrix $\vec{S}$, select only the s largest singular values
- Keep the corresponding columns in $\vec{k}$ and $\vec{D}^t$
- The resultant matrix is called $\vec{M}_s$ and is given by
  - $\vec{M}_s = \vec{K}_s \vec{S}_s \vec{D}_s^t$
  - where s, $s < r$, is the dimensionality of the concept space
- The parameter s should be
  - large enough to allow fitting the characteristics of the data
  - small enough to filter out the non-relevant representational details

Latent Ranking

- The user query can be modelled as a pseudo-document in the original $\vec{M}$ matrix
- Assume the query is modelled as the document numbered 0 in the $\vec{M}$ matrix
- The matrix $\vec{M}_s^t \vec{M}_s$ quantifies the relantionship between any two documents in the reduced concept space
- The first row of this matrix provides the rank of all the documents with regard to the user query (represented as the document numbered 0)

Conclusions

- Latent semantic indexing provides an interesting conceptualization of the IR problem
- It allows reducing the complexity of the underline representational framework which might be explored, for instance, with the purpose of interfacing with the user

Neural Network Model

Neural Network Model

- Classic IR:
  - Terms are used to index documents and queries
  - Retrieval is based on index term matching
- Motivation:
  - Neural networks are known to be good pattern matchers

Neural Network Model

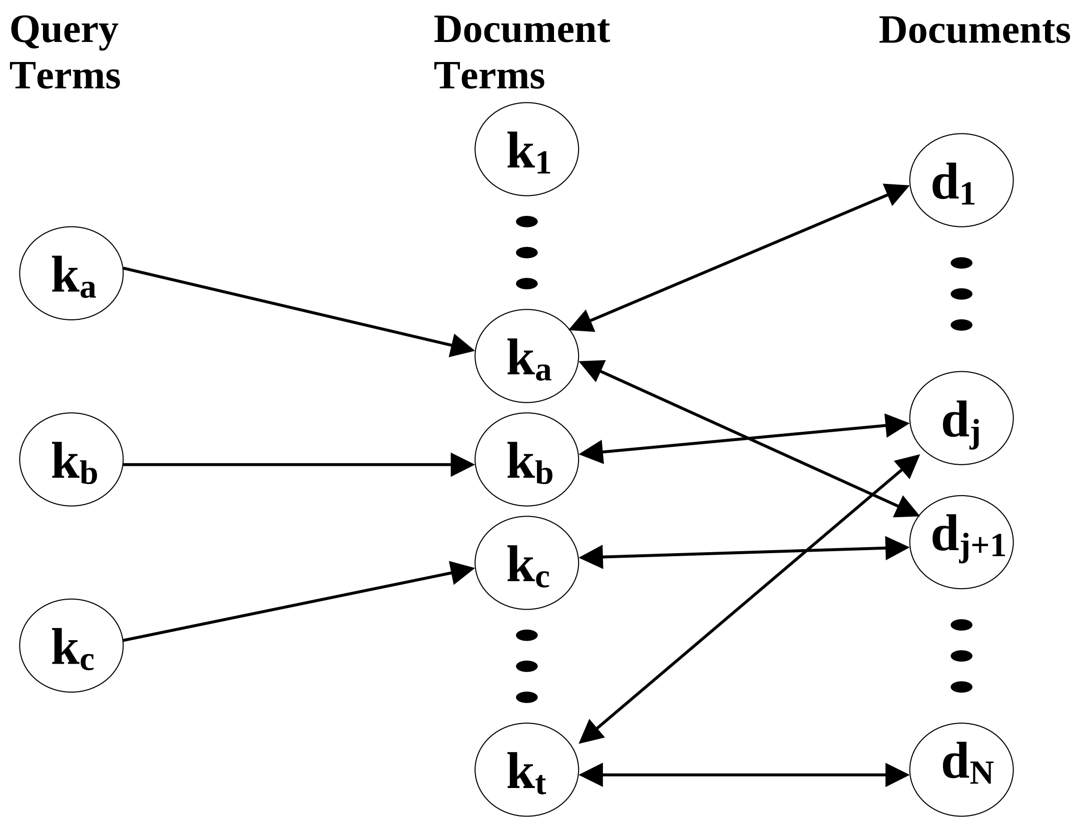
- Neural Networks:
  - The human brain is composed of billions of neurons
  - Each neuron can be viewed as a small processing unit
  - A neuron is stimulated by input signals and emits output signals in reaction
  - A chain reaction of propagating signals is called a spread *activation process*
  - As a result of spread activation, the brain might command the body to take physical reactions

Neural Network Model

- A neural network is an oversimplified representation of the neuron interconnections in the human brain:
  - nodes are processing units
  - edges are synaptic connections
  - the strength of a propagating signal is modelled by a weight assigned to each edge
  - the state of a node is defined by its *activation level*
  - depending on its activation level, a node might issue an output signal

Neural Network for IR

- From the work by Wilkinson & Hingston, SIGIR'91



Neural Network for IR

- Three layers network
- Signals propagate across the network
- First level of propagation:
  - Query terms issue the first signals
  - These signals propagate accross the network to reach the document nodes
- Second level of propagation:
  - Document nodes might themselves generate new signals which affect the document term nodes
  - Document term nodes might respond with new signals of their own

Quantifying Signal Propagation

- Normalize signal strength (MAX = 1)
- Query terms emit initial signal equal to 1
- Weight associated with an edge from a query term node k_i to a document term node k_i:

$$\overline{w}_{i,q} = \frac{w_{i,q}}{\sqrt{\sum_{i=1}^{t} \; w_{i,q}^2}}$$

- Weight associated with an edge from a document term node k_i to a document node d_j:

$$\overline{w}_{i,j} = \frac{w_{i,j}}{\sqrt{\sum_{i=1}^{t} \; w_{i,j}^2}}$$

Quantifying Signal Propagation

- After the first level of signal propagation, the activation level of a document node d_j is given by:

$$\sum_{i=1}^{t} \overline{w}_{i,q}\, \overline{w}_{i,j} = \frac{\sum_{i=1}^{t} \; w_{i,q}\, w_{i,j}}{\sqrt{\sum_{i=1}^{t} \; w_{i,q}^2} \times \sqrt{\sum_{i=1}^{t} \; w_{i,j}^2}}$$

  which is exactly the ranking of the Vector model

- New signals might be exchanged among document term nodes and document nodes in a process analogous to a feedback cycle

- A minimum threshold should be enforced to avoid spurious signal generation

Conclusions

- Model provides an interesting formulation of the IR problem
- Model has not been tested extensively
- It is not clear the improvements that the model might provide