The Information Retrieval "data model"

Introduction: IR – XML - DBS
- Boolean Model
- Vector space Model
- (Probabilistic Model)
- Evaluation of Retrieval effectiveness

Data models
- Databases
  - Rigid data models: relational, object-oriented
  - Database conformant to schema
  - Semantics of query q: subset of database
  - No texts, images, ...  (originally)
- Semi structured DB / XML
  - Schema more flexible – if any
  - Many schema items
  - Text plays a big role
  - Semantics of queries: substructure of DB
- Information Retrieval
  - Data model: objects are sequences of terms
  - Semantics of query q: all database ordered by similarity to q (Ranking)
The Information Retrieval model

- **Document model**
  - $D = \text{"set of documents"}$
  - $K = \{k_1, \ldots, k_n\}$ set of index terms
    - $K \sim$ set of all words occurring in the database
    - Typically very large

For every $d_j \in D$, $k_i \in K$ there is a weight $w_{i,j} \geq 0$, $w_{i,j} \in \mathbb{R}$, $k_i$ does not occur in $d_j$ => $w_{i,j} = 0$

$d_j' = (w_{1,j}, \ldots, w_{n,j})$ is the document representation of $d_j$

identify $d_j'$ and $d_j$ in most cases, i.e.

$D = \{d_j | d_j = (w_{1,j}, \ldots, w_{n,j})\}$

i.e. a document is a high dimensional vector of real numbers, most of them are 0, each component represents a term $\in K$.

---

### Boolean retrieval

- **Model**
  - $wij = 1$ if term $k_i$ occurs in document $d_j$, else 0
  - Query language: boolean expression of $k_i \in K$
  - Evaluation of a query $q$:
    - let $d_j \in D$ a document vector of 0 and 1,
    - if $q = k_i$ then $d$ matches $q$ iff $d_{ij} = 1$
    - if $q = "q_1 \text{ AND } q_j"$ $q$ matches $d_j$ if $q_1$ matches $d_j$ and $q_2$ matches $d_j$
    - if $q = "q_1 \text{ OR } q_j"$ $q$ matches $d_j$ if $q_1$ matches $d_j$ or $q_2$ matches $d_j$
    - if $q = "\text{NOT } q_1"$ $q$ matches $d_j$ if $q_1$ does not match $d_j$

- **Implementation**
  - Conceptually simple
  - Efficient query evaluation
Boolean retrieval

- **Issues**
  - Very restrictive evaluation: binary decision
    Wanted: mapping \( s: Q \times D \to [0,1] \)
    \( Q \) is the set of all queries
  - Every term has the same influence on the result
    Wanted: weight should reflect "importance" of term
  - Example:
    term "system" occurs in many technical documents
    many times, term "recovery" only sometimes.
    In a search for "recovery AND system" both have the same significance.
  - For \( q = "k_1 OR…. k_j" \) a document matches if at least one term matches.
    No difference if one or all terms match.

---

Coordinate level match

- Let \( q \) be in disjunctive normal form:
  \( q' = \text{DISJ} (t_{i1} \text{ AND } t_{i2} \text{ AND}….\text{AND} t_{ik}), t_{ij} = 0 \) or 1

Example: \( q = (\text{XML or DTD}) \) and \( \text{parser} \)
\( q' = (111) \text{ OR } (101) \text{ OR } (011) \)
  - Extend each disjunctive term by 0's for all terms in \( K \) not occurring in \( q \)
  - \( q' = (0000100011000) \text{ OR } … = qSig_1 \text{ OR}…\text{OR} qSig_k \)

\[
s(q,d) = \max (qSig_i * d)
\]
\( (*) : \text{scalar product} \)

Means: the more query terms are in document, the better
Boolean Retrieval

- Discussion
  (+) Ranking
  (+) number of matching query terms in document d define rank of d
  (-) Rank dependent on number of query terms
  (-) Documents with many terms tend to be ranked higher
  (-) Terms which occur frequently in documents are treated in the same way as infrequent terms

- Requirement
  - More general term weights
  - Normalization of ranking s(q, d)

Vector space model

- Model
  - Documents: points in a |K| = n – dimensional vector space.
  - Weights normalized
    e.g. 0 <= w <= 1
  - Terms are independent of each other ("orthogonal")

- Queries ….
  … are (formally) documents(!): q = (q1, q2, ….qn)

- Needed: measure of similarity between document and query, e.g. vector difference.
Vector space: similarity function

- Heuristic similarity functions
  - Scalar product?
    \[ w_1j^*q_1 + w_2j^*q_2 + \ldots + w_nj^*q_n \]
    Not bounded, may become arbitrarily large
  - Cosine measure
    \[
    \cos \phi = \frac{d_j \cdot q}{|d_j| \cdot |q|} = \frac{\sum w_{ij}q_i}{\sqrt{\sum w_{ij}^2} \cdot \sqrt{\sum q_i^2}}
    \]
    Measures angle between query vector and document.

Weights

- How to assign weights to documents / queries
  - Manual weight? Impossible! (more or less)

- Document frequency
  - Remember: infrequent terms are typically more significant than frequent ones
    "the" compared to "car"
  - Hypothesis: importance of a term depends on number of documents it occurs in
  - Justification: Zipf's law
    "Frequency of an event is inversely proportional to its significance"\(^1\)
    (Human Behaviour and the Principle of Least effort (G. Zipf 1949))
    (see http://information-retrieval.de/irb)
  - Consistent to information theory (Shannon)

  \[ w \propto \frac{1}{DF(t)} \]
  i.e. number of documents, \( t \) occurs in

\(^1\)my formulation
Weights

- **Term frequency**
  - Hypothesis
    - Term frequency TF i.e. the frequency of a term t within a document d characterizes contents of d
  - Term frequency $f_{tj}$ is a function of term t and document $d_j$

- **Normalization**
  - TF should not be linear
    - normalization heuristics
      - e.g. $f_{tj} = 1 + \log f_{tj}$
      - or $f_{tj} = K + (1-K) \frac{f_{tj}}{\max_i f_{ij}}$
  - IDF should be independent from number of documents
    - normalization heuristics
      - e.g. $w_t = \log (1 + N/f_t)$, $N =$ number of documents
      - or $w_t = \log (1 + f_{\max} / f_t)$
    - many other heuristics...

Cumulative weight of term t in document j

$$w_{tj} = f\left(\frac{TF}{1/DF}\right) = f\left(TF, IDF\right)$$

IDF = inverse document frequency

- Weight of term t in document $d_j$ ("TF / IDF heuristics")

$$w_{tj} = r_{tj} \cdot w_t$$

- Weight of a query term

$$w_{tq} = q_t \cdot w_t$$

$q_t =$ weight relative to query.

Typical: $q_t = 1$ ("All terms equally important")
### Cosine measure

\[
\cos(d_j, q) = \frac{d_j \cdot q}{|d_j| \cdot |q|} = \frac{\sum_{t \in d_j \cap q} w_{tj} \cdot w_t}{\sqrt{\left(\sum_{t \in d_j} w_{tj}^2\right) \cdot \left(\sum_{t \in q} w_t^2\right)}}
\]

\[
= \frac{1}{(W_j \cdot W_q)} \cdot \sum_{t \in d_j \cap q} (1 + \log f_{t,j}) \cdot (1 + \log (N/f_t))^2
\]

Note: document frequency has double influence - counts in \(d_j\) as well as \(q\). Reasonable?

Most often used:

\[
\cos(d_j, q) = \frac{1}{(W_j \cdot W_q)} \cdot \sum_{t \in d_j \cap q} (1 + \log f_{t,j}) \cdot (1 + N/f_t)
\]

\[\Rightarrow\] Ranking of result set

**Issues:**
- Efficient implementation
- Evaluation of "retrieval effectiveness"
- Many more similarity measures.
- Specific measures for Web documents (e.g. Google: "page rank")

### Implementation of vector space model

- **Inverted file**
  - Terms \(t\)
  - \(df\)
  - \(D_o f_{t,j}\)
  - Index file, e.g. B+ tree, suffix tree
  - Postings lists, suited for boolean queries and similarity search

Most values may be calculated in advance and put into the posting list, e.g. \(1 + \log f_{t,j}\) but \(f_{t,j}\)
Evaluation

- Issues
  - Subjectiveness of judgement
    - How relevant is a document with respect to a query?
  - Elaborate, costly empirical tests required
    - many queries, many individual judgements
      - for each query, mean of judgements?

- Evaluation model
  - Ideal observer: knows relevant documents for each query
  - Check for each query q
    - how many relevant documents found
    - how many irrelevant documents found
  - Calculate mean over many queries

Recall and precision

Recall:
- fraction of relevant documents found of all relevant documents
- \( R = \frac{r}{r+v} \)

Precision:
- fraction of relevant documents found of all documents found
- \( P = \frac{r}{r+n} \)

How to evaluate ranking order?
### Evaluation

#### Recall-Precision Graph

<table>
<thead>
<tr>
<th>Recall level</th>
<th>Precision %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>18</td>
<td>10</td>
</tr>
<tr>
<td>19</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>21</td>
<td>10</td>
</tr>
<tr>
<td>22</td>
<td>10</td>
</tr>
<tr>
<td>23</td>
<td>10</td>
</tr>
<tr>
<td>24</td>
<td>10</td>
</tr>
<tr>
<td>25</td>
<td>10</td>
</tr>
</tbody>
</table>

**Recall level n:**
- n % of all relevant Documents have been found for a particular query and document set!

**RC curve:**
- Precision at recall level n

---

### Summary of first part

- **Data models**
  - **structured**
    - Logic based DM: RDM, ORDM, OODM
  - Semi-structured, XML
  - **unstructured**
    - String based:
      - Word sequences: IR
      - Bit strings: images
      - Bit string sequences: Video

- **Issue today:** not just one data model appropriate for an application
- **Solution today:** user defined types (txt, XML, image) in (object) relational systems
- **To come:** data in a network --> **Distributed Databases**