Introduction to Relational IR

Jay Urbain

Credits: Ophir Frieder, Nazli Goharian
IIT

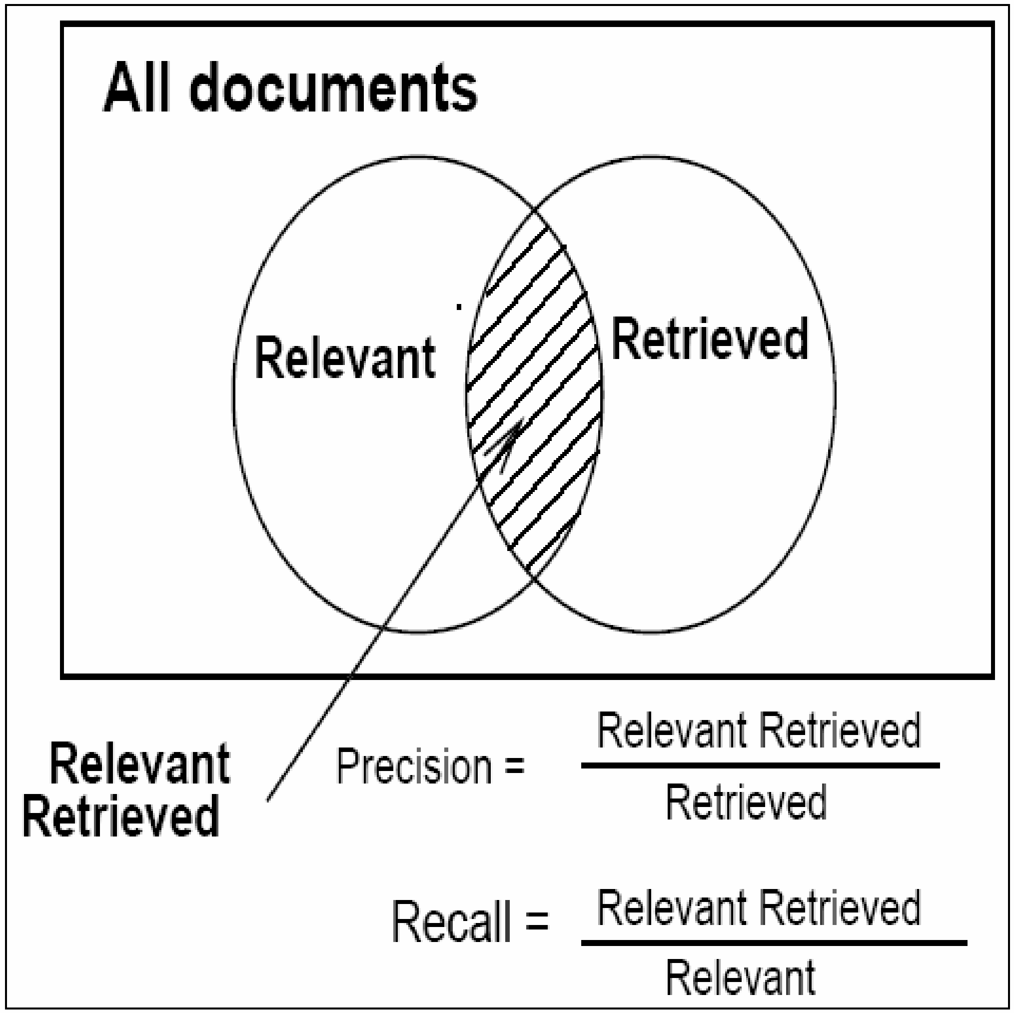
Topics

- Overview
- Performance Measurements
- Requirements
- Architecture
- Retrieval Strategies
- Relational Data Model

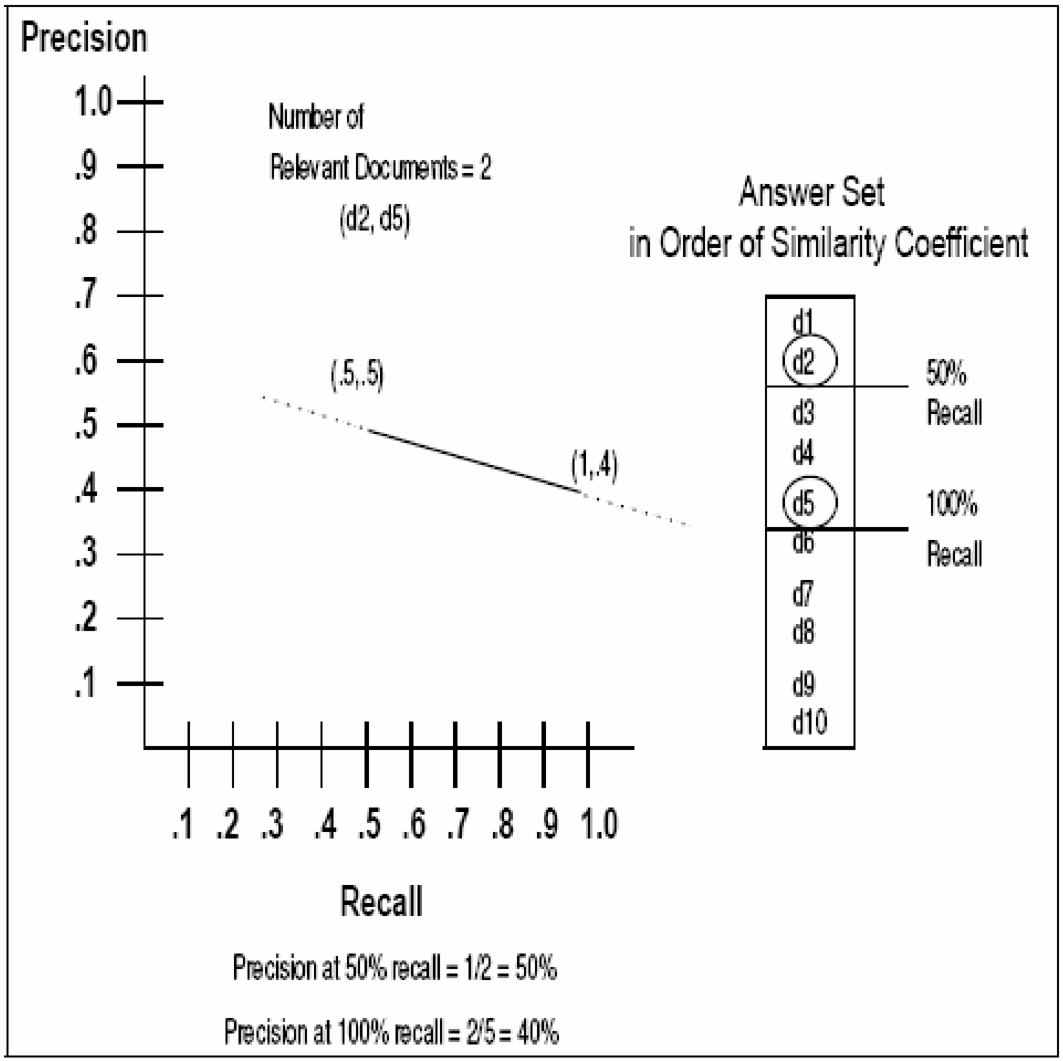
Overview

- Information Retrieval (IR) = Search
- *Information Retrieval* implies search covers any form of information:
  - structured relational data, text, video, image, sound, musical scores, DNA sequences, etc.
- Amount of structured data, e.g., gene microarrays, datawarehouses, XML, etc., and unstructured text growing rapidly.
- Need methods to integrate search of structured and unstructured data.

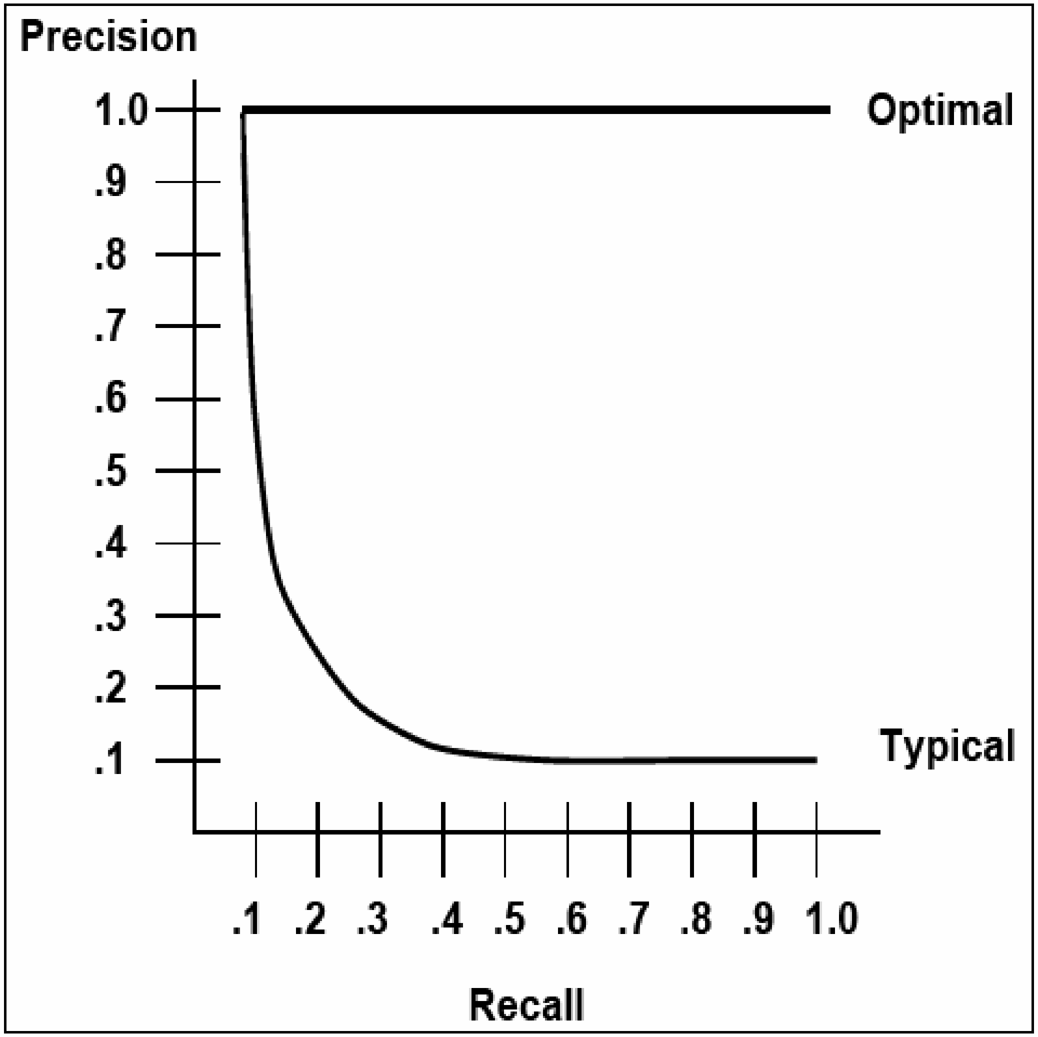
Performance



Precision measured at various points of recall



Precision/recall graph



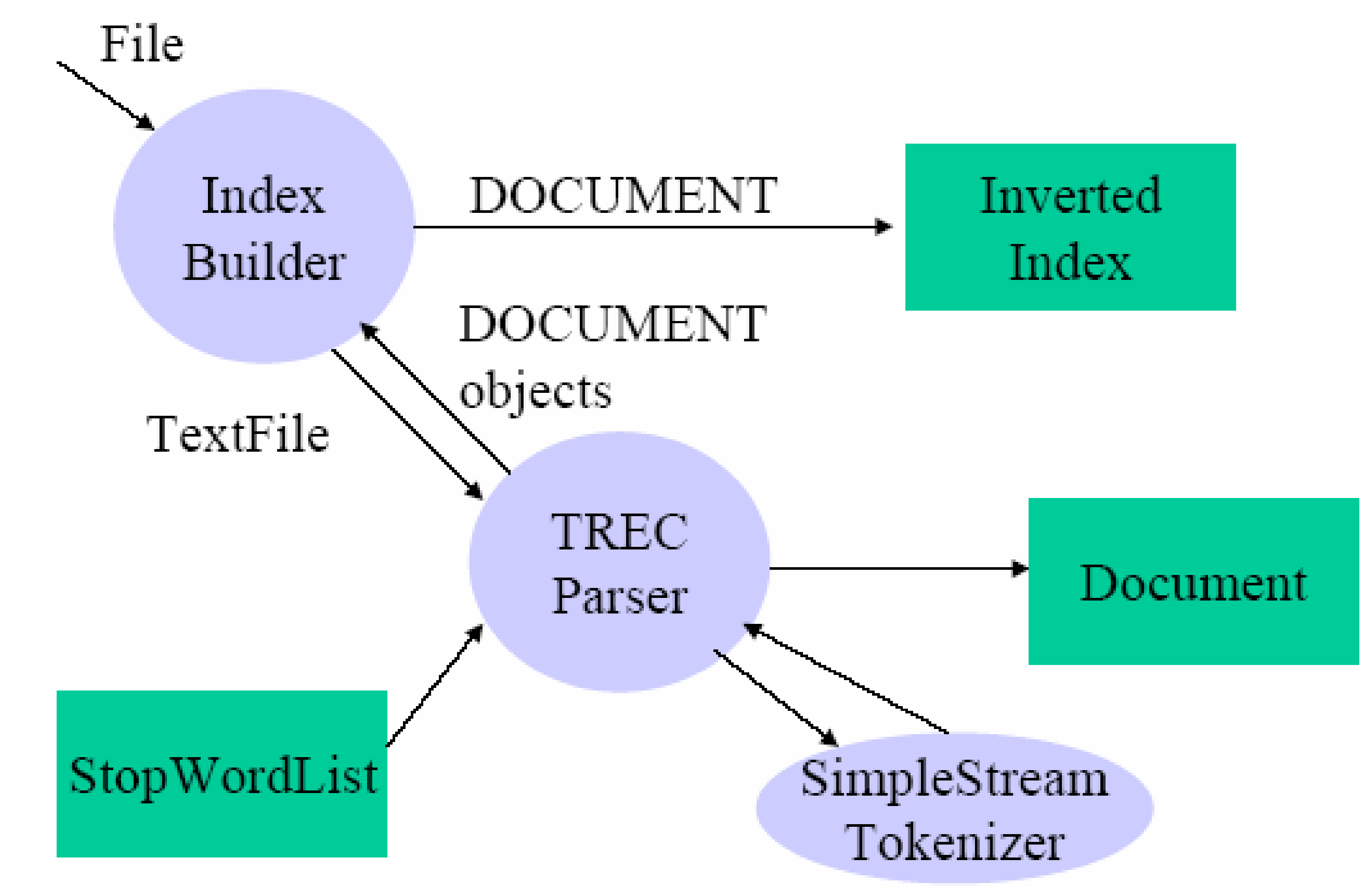
Requirements

- Scalability
  - Must handle large document/media collections
- Index Efficiency
  - Must build indexes in a reasonable amount of time
- Query Efficiency
  - Queries must run fast
- Query Effectiveness
  - Result set must be relevant

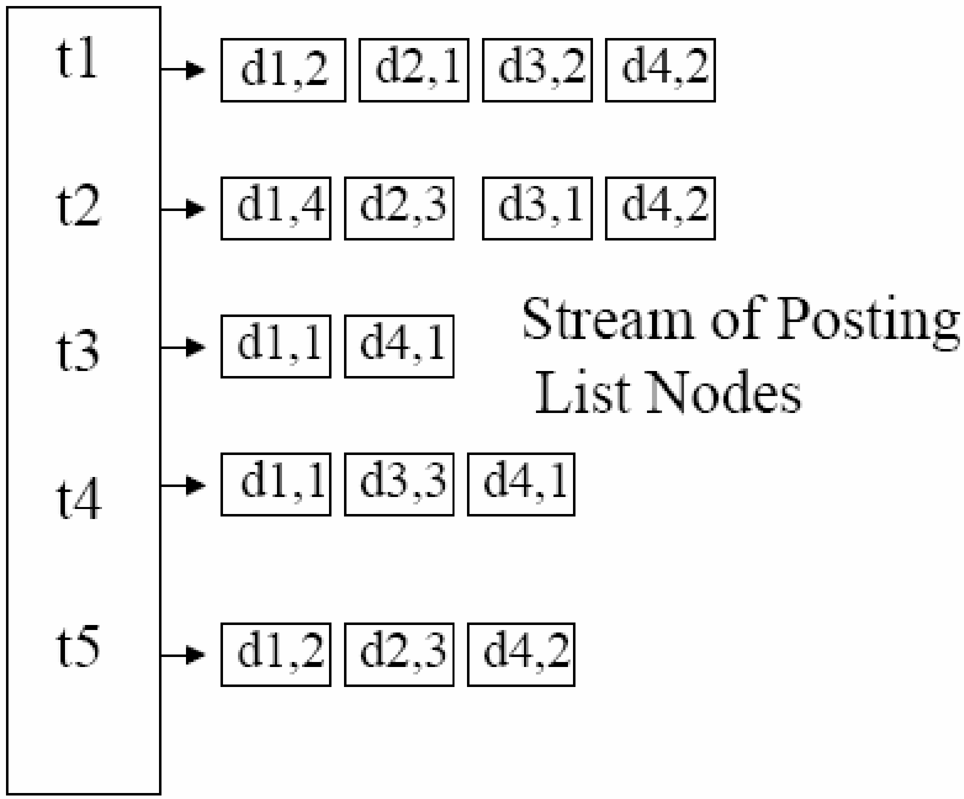
Architecture

- IR engine has two main components
  - Indexing:
    - Index documents/media
    - Inverted-index data structure
  - Query Processing:
    - Accept and process user query
    - Use *retrieval strategy* to identify relevant information.

Simple Indexing Architecture

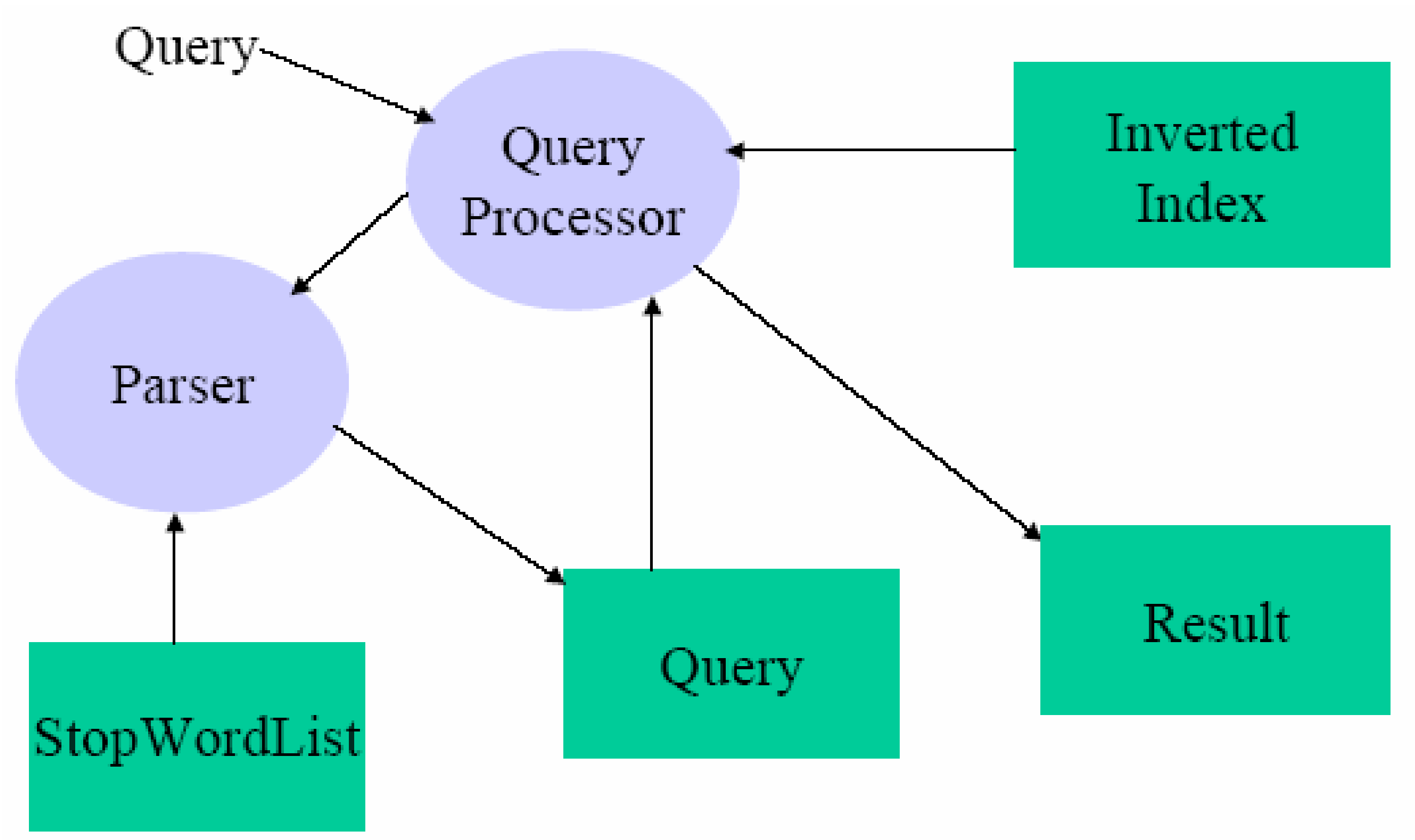


Simplified Inverted Index



Inverted Index

Query Processing



Retrieval Strategies

- Retrieval strategy
  - Algorithm that assigns a similarity coefficient *$SC(Q,D_i)$* to *each* document for a given user query.

- Challenges dealing with ambiguity in language
  - Same concept can be described by different terms
    - *New York = Big Apple?*
  - Different concepts can be described by same term

    River Bank != river bank

Retrieval Strategies

- Common retrieval strategies
  - Boolean
  - Vector Space Model
  - Probabilistic
  - Language Models
  - Page rank type methods

- Advanced techniques
  - *Inference networks, semantic models, mixed modal, clustering, hierarchical models, neural networks, genetic algorithms*

Constructing a retrieval function

First, some terminology:

- Term Frequency
  - $tf_{i,j}$ – # times term *i* occurs in doc *j*
- Document Frequency
  - df_i - # documents term *i* occurs
- Inverse Document Frequency $idf_i = \log\left(\frac{N}{df_i}\right)$

A simple similarity coefficient

- Take product of query and document vectors

$$SC(Q, D_i) = \sum_{j}^{|Q|} q_j \times d_{i,j}$$

- Weight terms by frequency and distinctiveness

$$d_{i,j} = tf_{i,j} \times idf_j \qquad q_j = 1$$

- In practice you should normalize for doc length:

$$SC(Q, D_i) = \frac{\sum_{j}^{|Q|} q_j \times d_{i,j}}{\sqrt{\sum_{j}^{|Q|} (q_j)^2 \sum_{j}^{|Q|} (d_{i,j})^2}}$$

Example SC calculation

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

Id	**Term**	**df**	**idf**
1	a	3	0.00
2	arrived	2	0.58
3	damaged	1	1.58
4	delivery	1	1.58
5	fire	1	1.58
6	gold	2	0.58
7	in	3	0.00
8	of	3	0.00
9	silver	1	1.58
10	shipment	2	0.58
11	truck	2	0.58

Retrieval Results

Docid	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**	**t7**	**t8**	**t9**	**t10**	**t11**	**SC**
D1	0.00	0.00	1.58	0.00	1.58	0.58	0.00	0.00	0.00	0.58	0.00	**0.58**
D2	0.00	0.58	0.00	1.58	0.00	0.00	0.00	0.00	1.58	0.00	0.58	**2.16**
D3	0.00	0.58	0.00	0.00	0.00	0.58	0.00	0.00	0.00	0.58	0.58	**1.16**
Q	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	1.00	

Ranked Results = D2, D3, D1

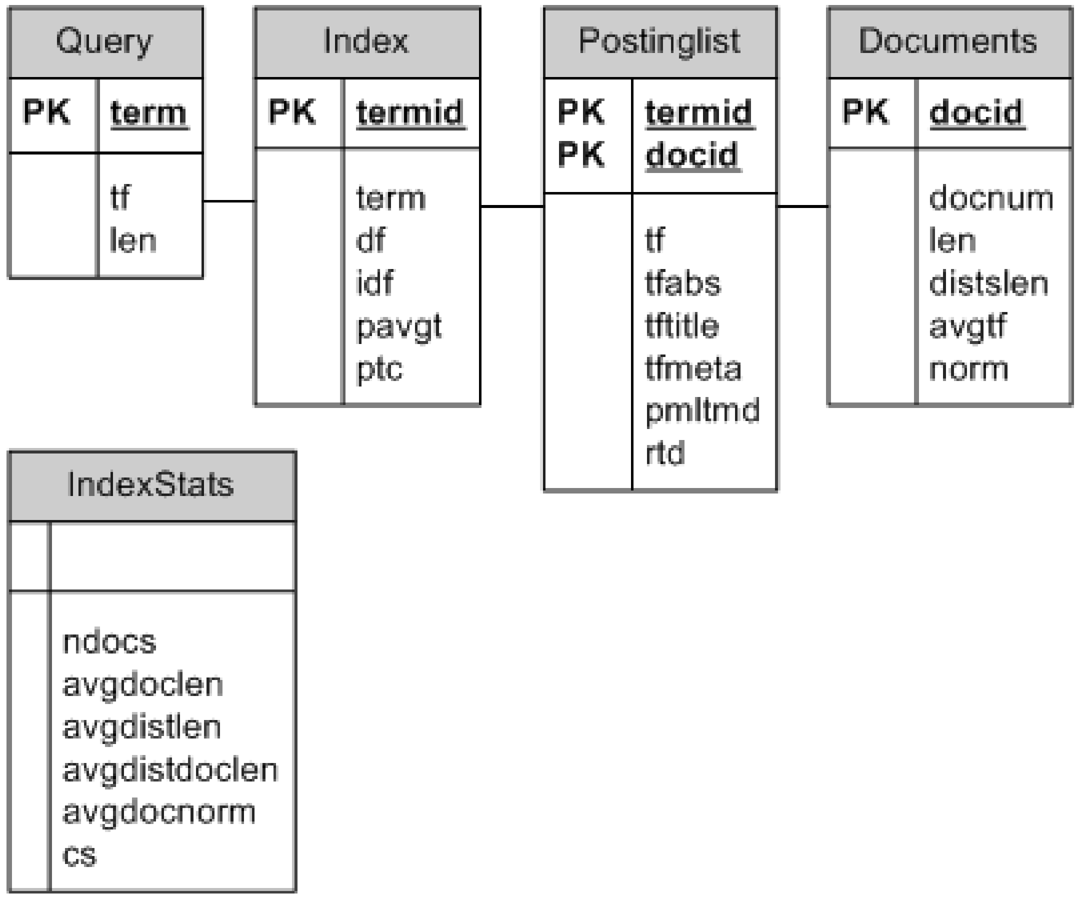
Back to the Relational Model

Integrate structured data with knowledge from text

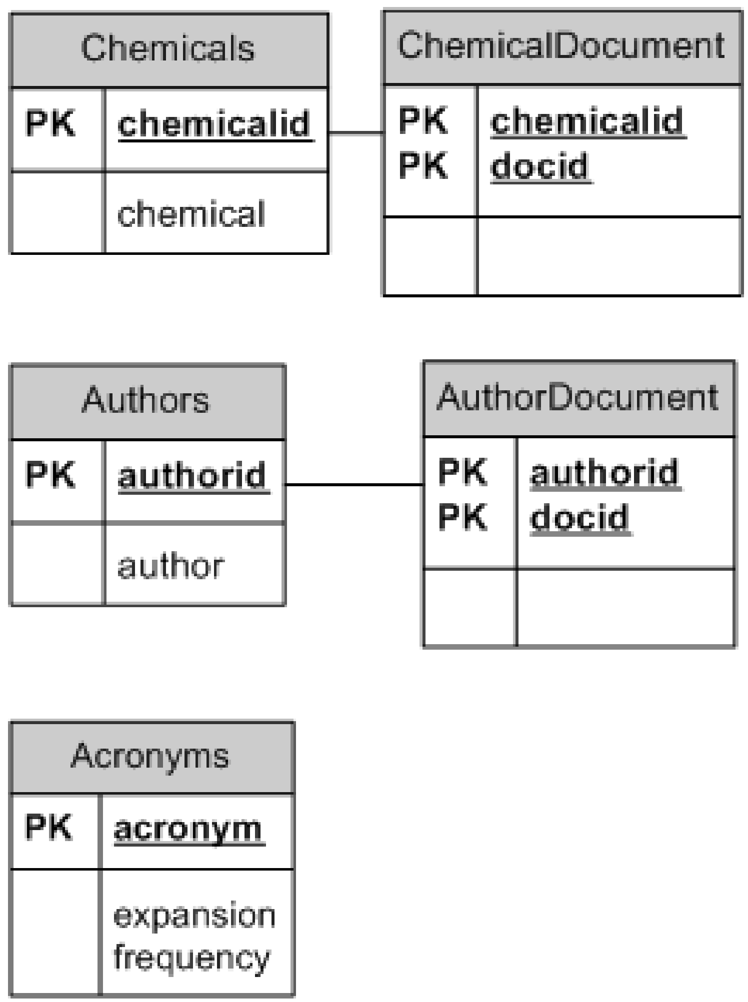
1. Add RDBMS functionality to information retrieval (IR) system
   - Mainstream approach in IR uses file-based inverted index.
   - Add software to integrate structured data with document index.
   - Support separate queries, then integrate.

2. Add IR functionality to RDBMS
   - Leverage off of investment by commercial database industry.
   - Build common index for structured and unstructured data.
   - Leverage data management tools.
   - Focus research efforts.

Relational Data Model



Sample auxiliary tables



Query Formulation

- Retrieval models implemented as aggregate SQL functions.
- Query table populated with topic terms.

Cosine:

$$\sum_{wq} \frac{idf * \ln(1+tf_q) * idf * \ln(1+tf_d)}{docLen}$$

PDLN:

$$\sum_{wq} \frac{idf * \ln(1+tf_q) * idf * \ln(1+tf_d)}{(1-s) * avgdoclen + s * doclen}$$

Query Formulation

Cosine with pivoted document length normalization (PDLN):

```
select p.docid, max(d.docnum) docnum,
    sum(i.idf*(1+ln(q.tf))*idf*(1+ln(p.tf))*d.NORM )) as sc
from index i, postinglist p, documents d, query q
where p.docid=d.docid
and   i.termid=p.termid
and   i.term=q.term
group by p.docid
order by sc desc;
```

Relevance Weighted Model

- Language model (LM) incorporating odds probability of relevance
- Relate LM members of the family of *tf*idf* weighted algorithms developed for vector space model.
- Only requires matching query terms in its computation.

$$P(t_1, t_2, ..., t_Q \mid d) = \prod_i^{|Q|} (\lambda_i P(t_i \mid d) + (1-\lambda_i) P(t_i)) \times P(d)$$

$$P(t_i \mid d) = \frac{tf(t_i, d)}{\sum_t tf(t, d)}$$

$$P(t_i) = \frac{df(t_i)}{\sum_t df(t)}$$

$$P(d) = \frac{\sum_t tf(t, d)}{\sum_d \sum_t tf(t, d)}$$

$$\sum_{wq} \ln\left(1 + \left(\frac{\lambda}{1-\lambda}\right) * \left(\frac{tf(t_i, d)}{df(t_i)}\right) * \left(\frac{\sum_t df(t)}{\sum_t tf(t, d)}\right)\right) + \ln\left(\frac{\sum_t tf(t, d)}{\sum_d \sum_t tf(t, d)}\right)$$

Relevance Weighted LM

- Relevance-weighted language model:

```
select p.docid, max(d.docnum) docnum,
    sum(ln(1+(lambda/(1-lambda))*
    (p.tf/d.len)*(s.cs/i.df)*(docPrior)  ) )  sc
from index i,postinglist p,documents d, query q, indexstats s
where p.docid=d.docid
and   i.termid=p.termid
and   i.term=q.term
group by p.docid
order by sc desc;
```

Evaluation

- Evaluated state-of-the art retrieval functions on relational model using OHSUMED Medline corpus.

- New methods for normalizing biomedical terms.

- Introduce *Relevance-based language model.*

Retrieval Model Results

- BM25 queries executed in the 0.5 to 1 second range.
- BM25 about twice as fast as LM-RW and the KL-Divergence formulations for LM-JM, LM-D, and LM-AD.
- LM-RW outperformed all other language models: LM-JM, LM-D, and LM-AD.

Retrieval	Parameters	MAP	% imp.	T*
PDLN	s=0.25	0.272	-	1.0
BM25	**k1=1.4,k3=7,b=0.75**	**0.307**	**13.2%**	**1.0**
LM-JM	$\delta = 0.8$	0.308	13.5%	12.0
LM-D	$\mu = 2000$	0.261	-4.1%	12.0
LM-AD	$\lambda = 0.1$	0.298	9.6%	12.0
LM-RW	$\lambda = 0.15$	**0.314**	**15.4%**	**4.5**

Problems addressed

- Developed a rapid-prototype, genomic-literature, retrieval engine using conventional relational technology.
- Captured the ability to integrate structured components into our search.
- Developed novel and effective term variation generation technique.
- Evaluated multiple retrieval models and demonstrated how these models can be implemented using standard SQL.
- Relevance language model
- Matched or exceeded state of the art results.

References

- *J. Urbain, N. Goharian, "A Relational Genomics Search Engine," BIOCOMP 2006: 69-74.*

- *Grossman D., Frieder, O., 2004. Information Retrieval: Algorithms and Heuristics, Second Edition; Springer Publishers, ISBN 1-4020-3003-7 (hardcover), 1-4020-3004-5.*

Probabilistic

- *BM25*:
- Best results with k1=1.4, k2=0, k3=7, and b=0.75

$$\sum_{wq} \ln\left(\frac{N - df + 0.5}{df + 0.5}\right)\left(\frac{(k_1 + 1) * tf_d}{k1 * (1 - b) + b * (\frac{docLen}{avgDocLen}) + tfd}\right)\left(\frac{(k_3 + 1) * tfq}{k_3 + tf_q}\right)$$

Language Models

- Jelinek Mercer

$$\sum_{wq} \ln((1-\lambda) * P_{ml}(w \mid d) + \lambda * P(w \mid C))$$

$$P_{ml}(w \mid d) = tf_{d} / doclen$$

- Dirichlet

$$\sum_{wq} \ln\left(\frac{tfd + \mu * P(w \mid C)}{docLen + \mu}\right)$$

- Absolute Discounting

$$\sum_{wq} \ln(\frac{\max(tfd - \delta, 0)}{docLen} + \frac{\delta * distDocLen}{docLen})$$