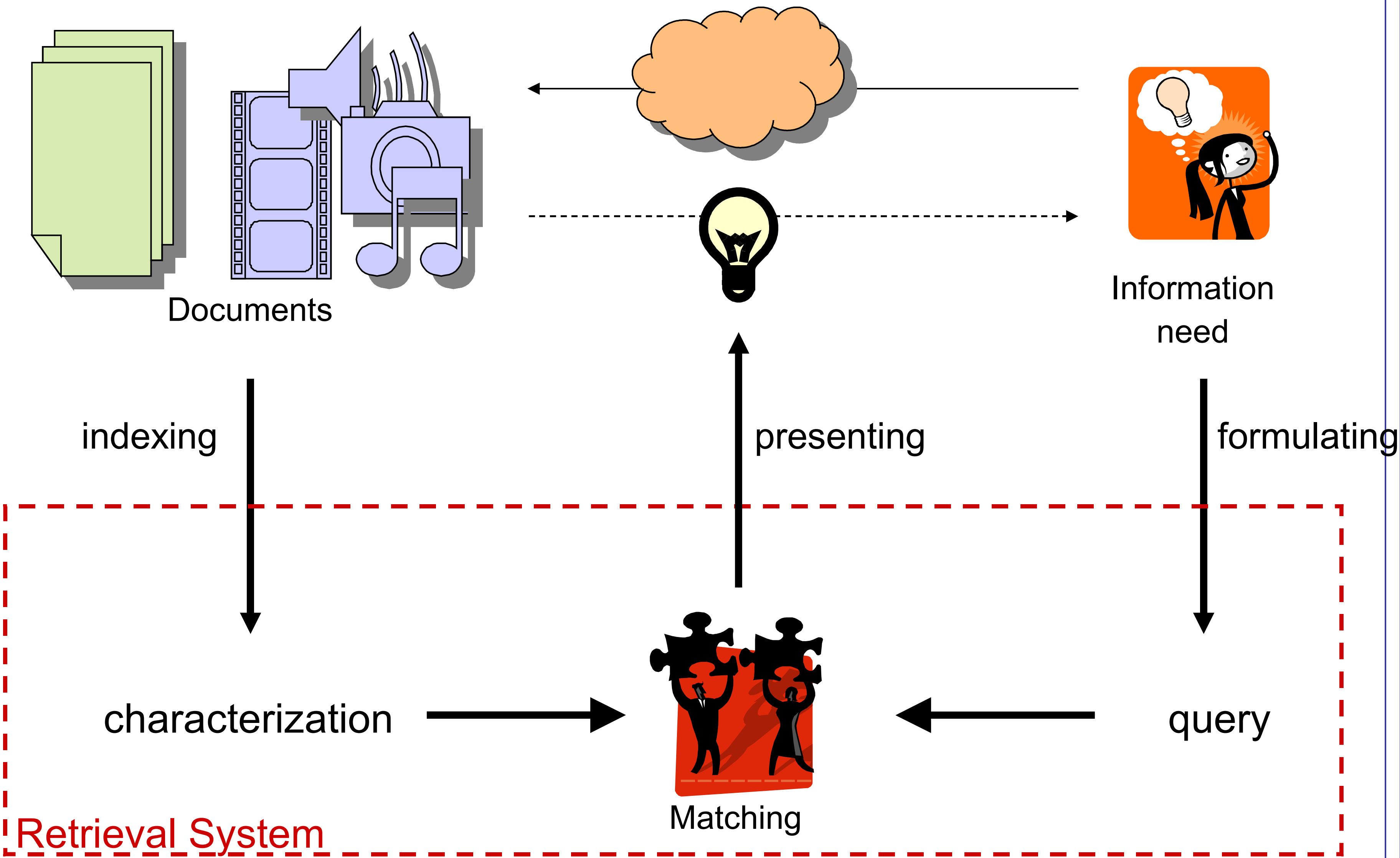
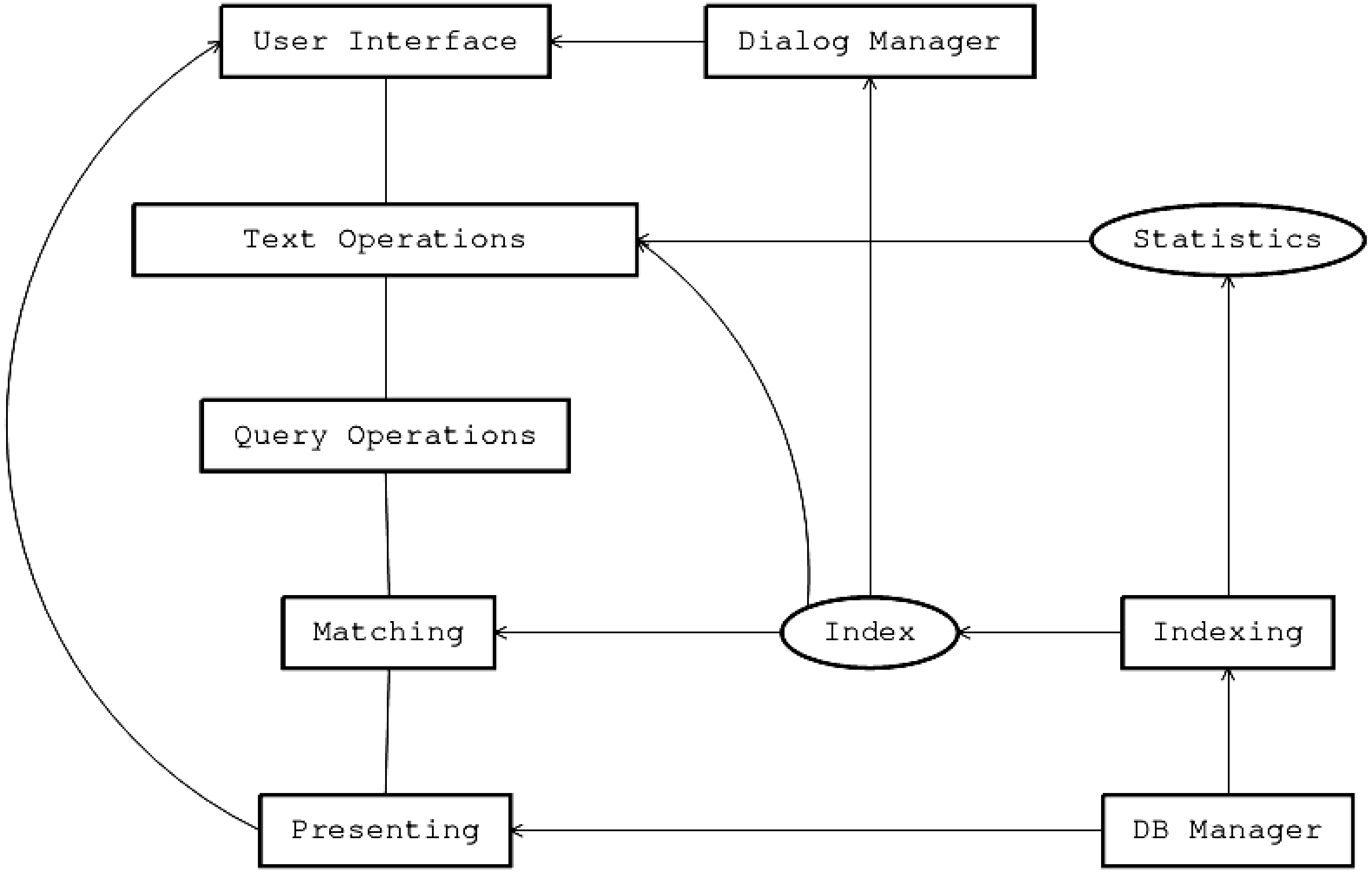
Query Modification

Contents

IR paradigm


Documents
Information need
indexing
presenting
formulating
characterization
query
Matching
Retrieval System

General Architecture



A nasty problem

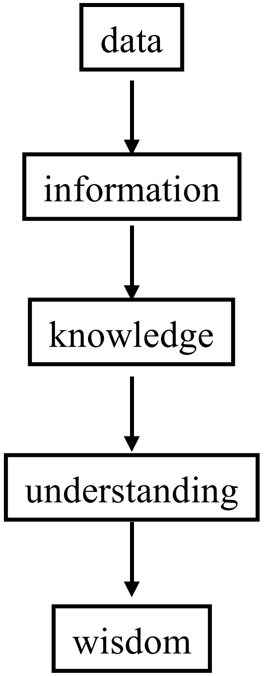
- Problem:
  - A collection of data is not information.
  - A collection of information is not knowledge.
  - A collection of knowledge is not wisdom.
  - A collection of wisdom is not truth.

(Fleming, Toffler)

- But what we have is: the data
  - both on the offering and asking side

Levels of understanding

- According to Russell Ackoff, the content of the human mind can be classified into five categories:
  - **Data**: symbols
  - **Information**: data that are processed to be useful; provides answers to "who", "what", "where", and "when" questions
  - **Knowledge**: application of data and information; answers "how" questions
  - **Understanding**: appreciation of "why"
  - **Wisdom**: evaluated understanding.



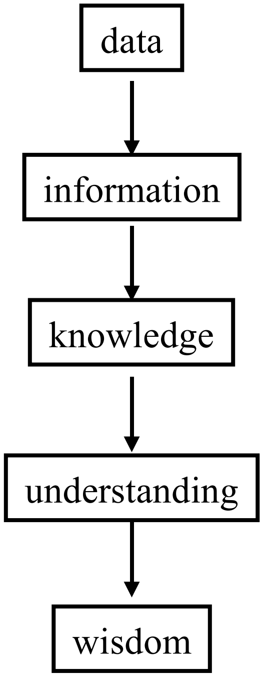
Data and Information

- **Data** is raw.
  - It simply exists and has no significance beyond its existence (in and of itself).
  - It can exist in any form, usable or not.
  - It does not have meaning of itself.
  - In computer parlance, a spreadsheet generally starts out by holding data.

- **Information** is data that has been given meaning by way of relational connection.
  - This "meaning" can be useful, but does not have to be.
  - In computer parlance, a relational database makes information from the data stored within it.

What the searcher wants

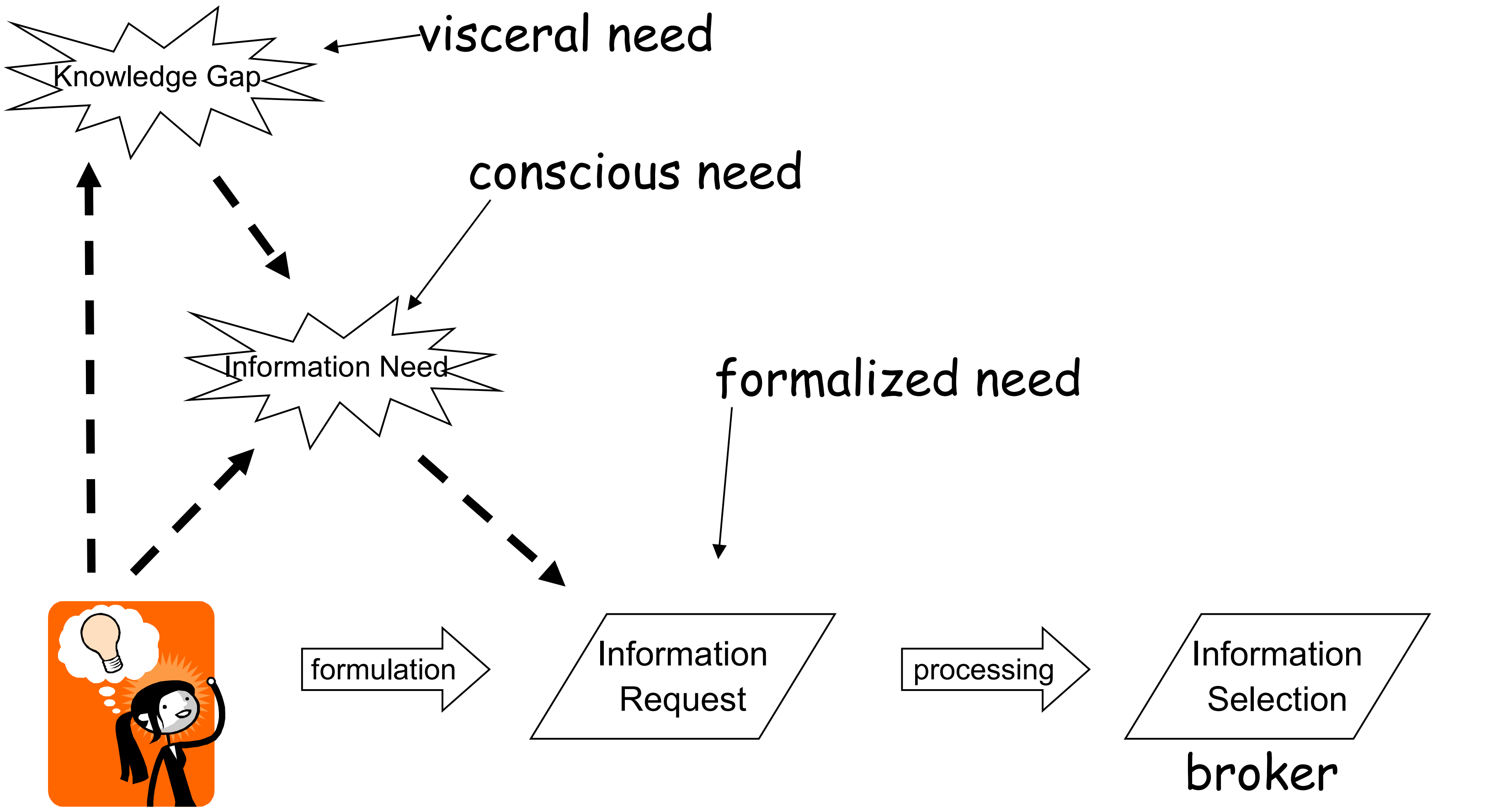
- Searcher has a knowledge gap
- Needs information to fill the gap
- But how is the starting state of a searcher?



Levels of information need

- visceral need (unconscious awareness):
  searcher can recognize some characteristics
- conscious need: searcher can judge relevance
- formalized need:
  - searcher has implicit or explicit formulation of need;
  - if implicit: can judge relevancy of description
- compromised need:
  searcher can compare different solutions.

Levels in action



Contents

1. General Architecture
2. The Information Retrieval Problem
3. Classic Models
4. Quality Measures
5. Query Modification
6. Conceptual Decomposition

What the searcher wants

- knowledge gap (visceral)

  and thus: information need (conscious)

- interpreted as document need (formalized)

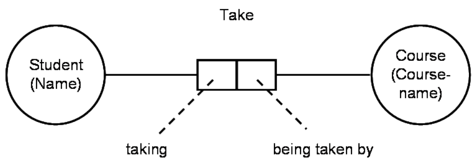
  - comparative model: partial ordering of documents
    $N = (D, <)$
  - weighted model: each document has need weight
    $N: D \rightarrow [0,1]$
  - incremental model: conditional need
    $N: P(D) \times D \rightarrow [0,1]$

How this can be formulated

- syntax-oriented retrieval languages: SQL

```
SELECT course
FROM Taken
WHERE student = 'Jansen'
```

- semi-natural retrieval languages: Lisa-D



Course being taken by Student 'Jansen'

Course being taken by Student taking Course 'Information Retrieval'

- semantics-oriented retrieval languages:
  - keywords/terms
  - "phrase"
  - +/- keyword
  - query, query

Course Taken Student Jansen

- dialogue (guided tour): Query by Navigation

Search Engine Features Chart

Last updated Sep. 17, 2006.
by Greg R. Notess

* See also Search Engines by Search Features.
* Search engines grouped by size; all words link to more detailed reviews.

Search Engines	Boolean	Default	Proximity	Truncation	Fields	Limits	Stop	Sorting
Google Review	-, OR	and	Phrase	No (stems) word in phrase	intitle, inurl, link, site, more	Language, filetype, date, domain	Few, + searches	Relevance, site
Yahoo! Review	AND, OR, NOT, ( ), -	and	Phrase	No word in phrase	intitle, inurl, link, site, more	Language, file type, date, domain	No	Relevance, site
Ask Review	-, OR	and	Phrase	No	intitle, inurl, site	Language, site, date	Yes, + searches	Relevance, metasites
Live Search Review	AND, OR, NOT, ( ), -	and	Phrase	No	intitle, link, site, loc, url	Language, site	Varies, + searches	Relevance,site, sliders
Gigablast Review	AND, OR, AND NOT, ( ), +, -	and	Phrase	No	title, site, ip, more	Domain, type	Varies, + searches	Relevance
Exalead Review	AND, OR, NOT, ( ),-	and	Phrase, NEAR	Yes and stems	intitle, inurl, link, site	Language, file type, date, domain	Varies, + searches	Relevance, date
WiseNut Review	- only	and	Phrase	No	No	Language	Yes, + searches	Relevance, site

A Notess.com Web Site

Requirements Retrieval Languages

- sufficiently expressive:

$$\forall_{A \subseteq D} \exists_{q \in Q} [q \text{ describes } A]$$

  - weighted model: q describes A if top |A| documents form A

- sufficiently convenient:
  - how efficiently can searcher find q given A

- efficiently computable

What does this query mean?

- intuitive semantics

- formal semantics: (assuming weighted model)

$$Norm : Q \rightarrow (D \rightarrow [0,1])$$

Golden Standard

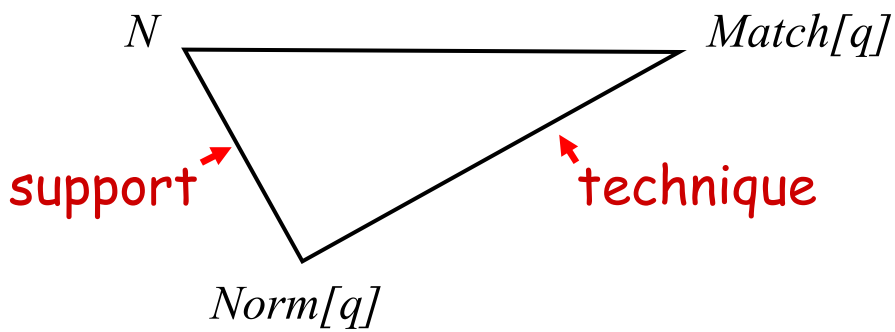
- operational semantics

$$Match : Q \rightarrow (D \rightarrow [0,1])$$

- IR system should try to minimize overall difference

$$\Delta\,(Norm, Match)$$

What the searcher should know



$$\underbrace{\Delta(N, Match[q])}_{hopefully\ \ small}$$

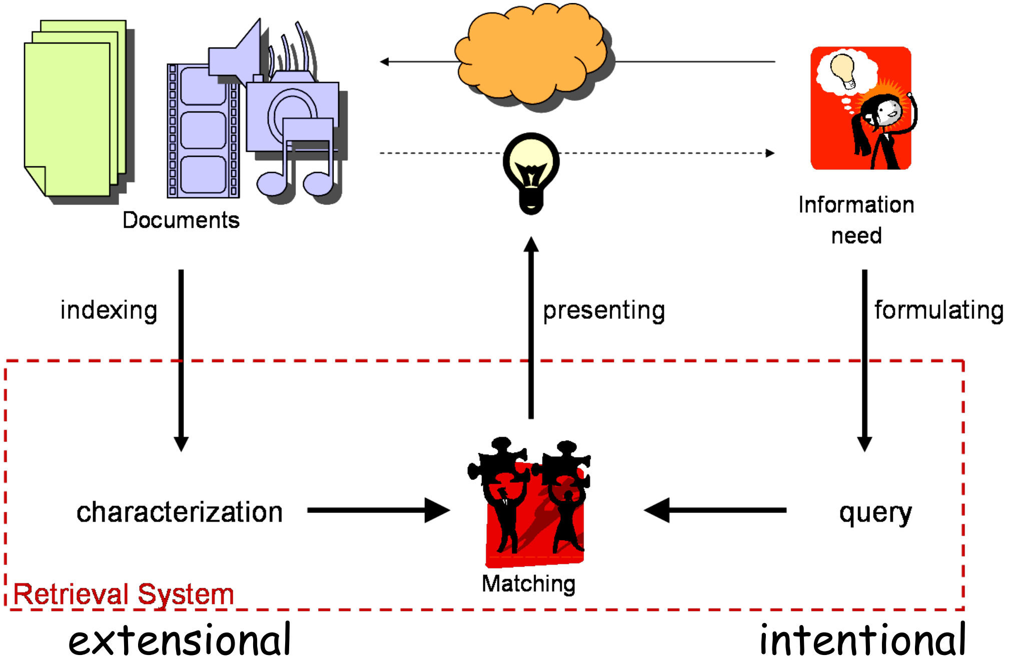
$$\leq \underbrace{\Delta(N, Norm[q])}_{quality\ \ IR\ model} + \underbrace{\Delta(Norm[q], Match[q])}_{quality\ \ IR\ system}$$

(Triangular Inequality)

- *A-priori support*: learning syntax, semantics and pragmatics of *O*
- *A-posteriori support*: dialog manager supporting process of formulation

Man-in-the-middle support

- document impression:
  - extensional aspects
- need impression (query)
  - intentional aspects



- characterization language C
  - base semantic units
  - construction rules for compound semantic units
- indexing
  - to what extent is document about the semantic units

Contents

Boolean Model

The formal model: indexing

- Let K be a set of terms (keywords)
  - terms are base concepts
    - verbs, nouns, adjectives
  - sets of terms are semantic units
- Documents are indexed by a set of terms:
  - The indexing process produces a set $\chi\,(d)$ of terms for each document d.
- Example:

  $\chi\,(d) =$ {computing science, information retrieval, archiving, hypertext, hypermedia}

Main assumptions

- If a document d contains a term t,

  then d ***is about*** t

- If a term t is requested,
  and d ***is about*** t

  then document d is relevant.

The formal model: query language

- Query language inductively defined as proposition calculus over K:
  - each term in K is seen as a variable, and therefore an elementary proposition
  - if p and q are propositions, then also propositions are:

    $p \wedge q$

    $p \vee q$

    $\neg p$
  - there are not other ways to construct propositions
- The boolean operators are also denoted as AND, OR and NOT.
- Example:

  archiving AND hypertext AND NOT implementation

The formal model: valuation

- The validity of a query (boolean proposition) depends on the value of the variables.

- Example:
  - suppose: archiving has value TRUE,
    hypertext has value FALSE
    implementation has value TRUE
  - then: archiving AND hypertext AND NOT implementation has value FALSE

- A value assignment is a function that assigns a truth value to each boolean variable.

- Let val be a value assignment, then we write $val \models q$

  to denote that query q is true for this value assignment.

The formal model: semantics

- We may see χ (d) as the value assignment that
  - assigns to term k the value TRUE if $k \in \chi(d)$
  - and FALSE otherwise.

- The result of query q then is defined as:

$$Match[q] = \left\{ d \in D \middle| \chi(d) \models q \right\}$$

- Note: this set has also been referred to as the support for q

Probabilistic Model

has been discussed before

Vector Model

The formal model

- Let K be a set of terms (keywords)
  - terms are base concepts
  - term weighting schemes (function $K \rightarrow [0,1]$) are semantic units
- Documents are indexed by stating for each term the degree in which the document is about that term.
  - The indexing process produces a function $K \rightarrow [0,1]$
- Example:

$$\chi(d) = \{\text{computing science:0.1, information retrieval:0.9, archiving:0.3, hypertext:0.5, hypermedia:0.6}\}$$

Improving the base assumption

- Next we relax our assumption:
  - If a document d contains a term t,
    then d is about t
  - If a term t is requested, and d is about t
    then document d is relevant.
- New assumption:
  - If a document d contains term t with intensity f
    then d is about t with weight f
  - If a term t is requested
    and document d provides t with weight f
    then document d has relevancy f for this searcher
  - If a term t is requested with necessity n
    and document d provides t with weight f
    then document d has relevancy n * f

- Document d specifies for each term t the degree $\chi(d)$ (t) in which it is about that feature:
  - Query q specifies the need for term t analogously: q (t)
  - Outcome: document qualifies to some extent
- Each term provides some evidence for relevancy:

$$\text{demand} * \text{supply} = \text{need} * \text{weight} = q(t) * \chi(d)\,(t)$$

- We assume the terms to be sufficiently independent to express the overall evidence for relevancy as:

$$\Sigma_{\text{term } t}\ \text{evidence term } t = \Sigma_{\text{term } t}\ q(t) * \chi(d)\,(t) \ = \ q \bullet \chi(d)$$

(dot-product)

Vector representation

- Assume the elements of K are numbered:

  $K = \{k_1, \ldots, k_m\}$,

  then we may see the function $\chi(d)\colon K \rightarrow [0,1]$ as a vector

  $(d_1, d_2, \ldots, d_m)$

  where $d_i = \chi(d)\,(k_j)$

- Usually, d and its vector representation $(d_1, d_2, \ldots, d_m)$ are identified

- The inner product for vectors also is a matrix multiplication:

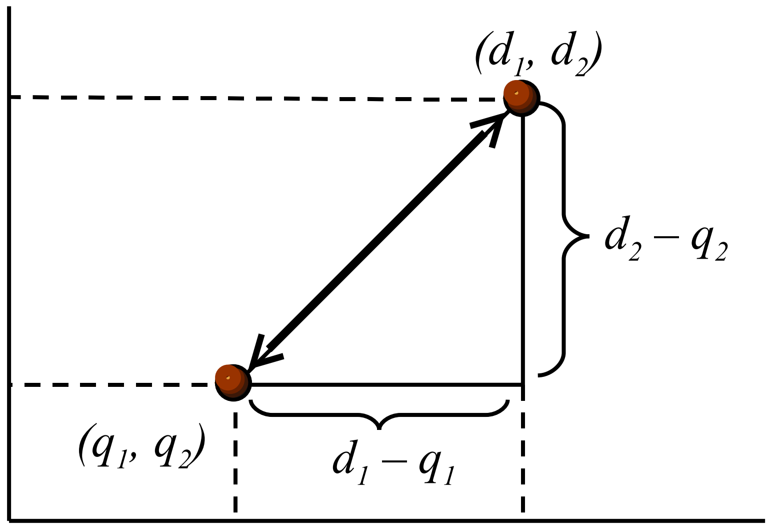
  $q \bullet \chi(d) = q^T \chi(d)$

Euclidian distance

- Given two document vectors d and q, then their distance is computed by

$$Dist(d,q) = \sqrt{\sum_{i=1}^{m} (d_i - q_i)^2}$$
$$= \sqrt{(d - q) \bullet (d - q)}$$

Pythagoras Theorem



- Length of vector d is distance to origin: information quantity

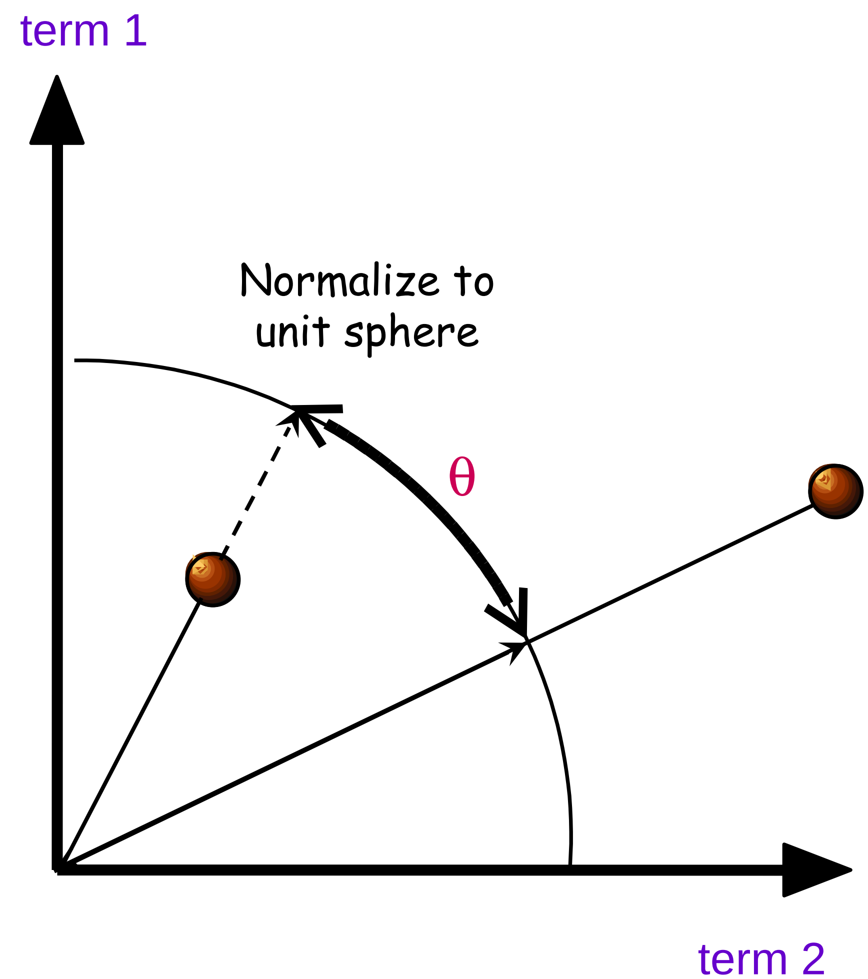
$$\|d\| = Dist(d,0) = \sqrt{\sum_{i=1}^{m} d_i^2} = \sqrt{d \bullet d}$$

Normalizing vectors

Documents are normalized by projecting them on the unit sphere.

This is done by dividing a vector by its length:

$$\frac{v}{\|v\|} = \frac{1}{\|v\|} v$$



Rather than Euclidian distance, take arc distance.

The arc distance may vary from 0 to $\pi/2$.

Will be normalized to interval [0,1]

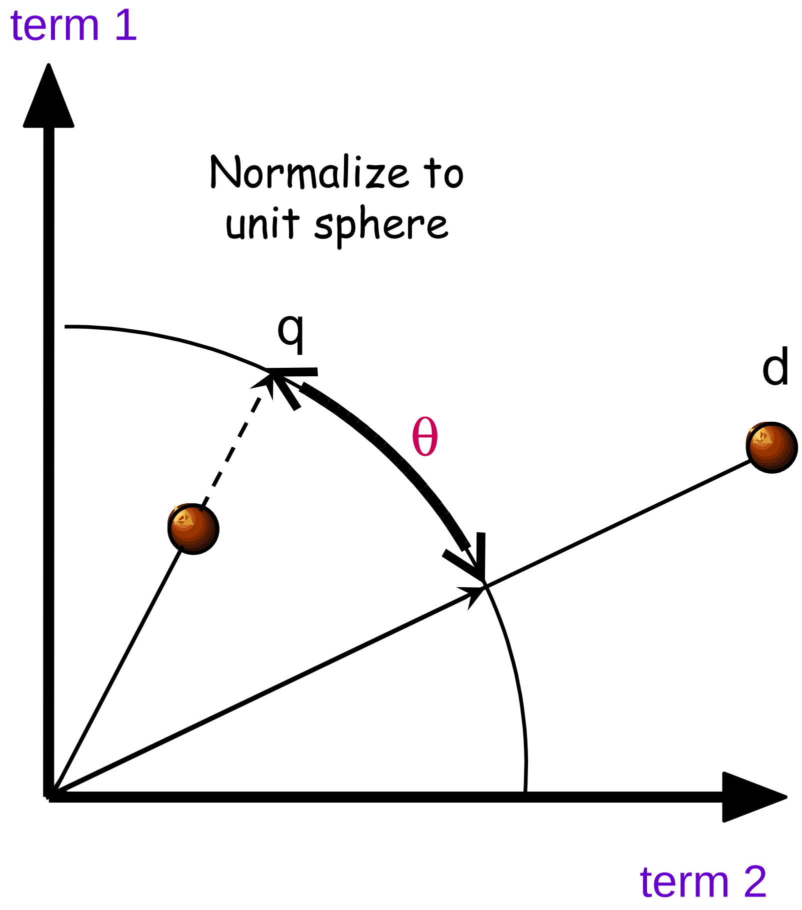
Matching

- Normalized arc distance d and q:

$$Sim(d,q) = \frac{d \bullet q}{\|d\| \cdot \|q\|}$$

$$= \frac{d \bullet q}{\sqrt{(d \bullet d) \cdot (q \bullet q)}}$$

$$= \frac{\sum_{i=1}^{m} d_i \cdot q_i}{\sqrt{\left(\sum_{i=1}^{m} d_i^{\,2}\right)\left(\sum_{i=1}^{m} q_i^{\,2}\right)}}$$



Total evidence

- So if we assume both document and query vector to have length 1, then we have:

$$\text{Sim}(d, q) = d \bullet q$$

- This is in line with the total evidence approach.

Cosine measure

- The inner vector product has the following property:

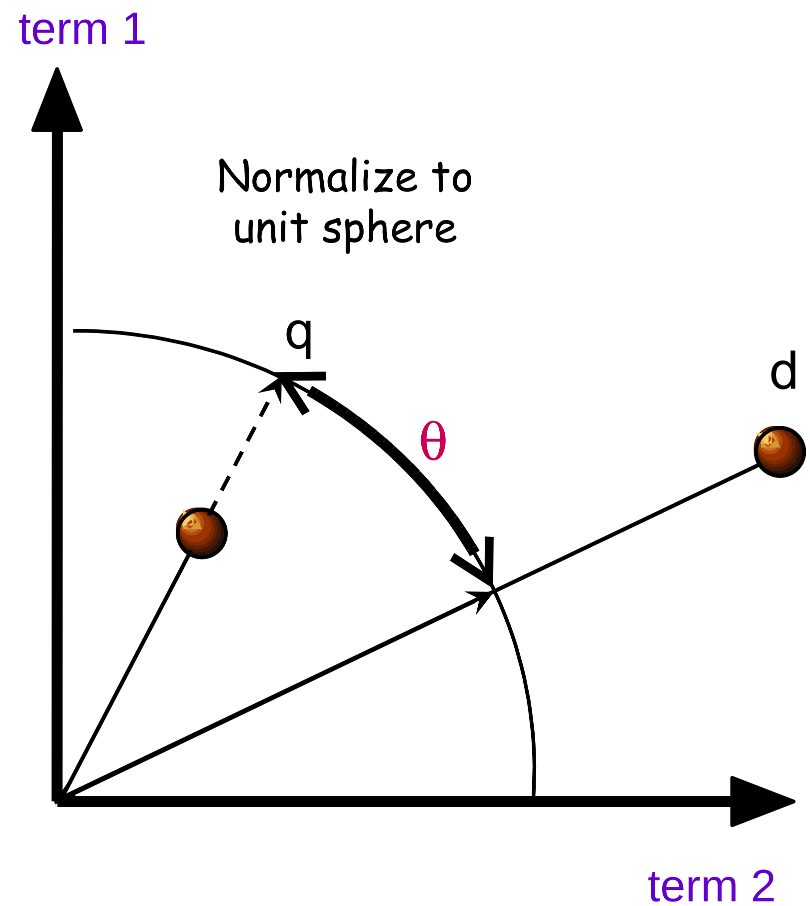
$$x \bullet y = \|x\| \cdot \|y\| \cdot \cos(\angle(x, y))$$

- Consequently we have:

$$Sim(d, q) = \frac{d \bullet q}{\|d\| \cdot \|q\|}$$

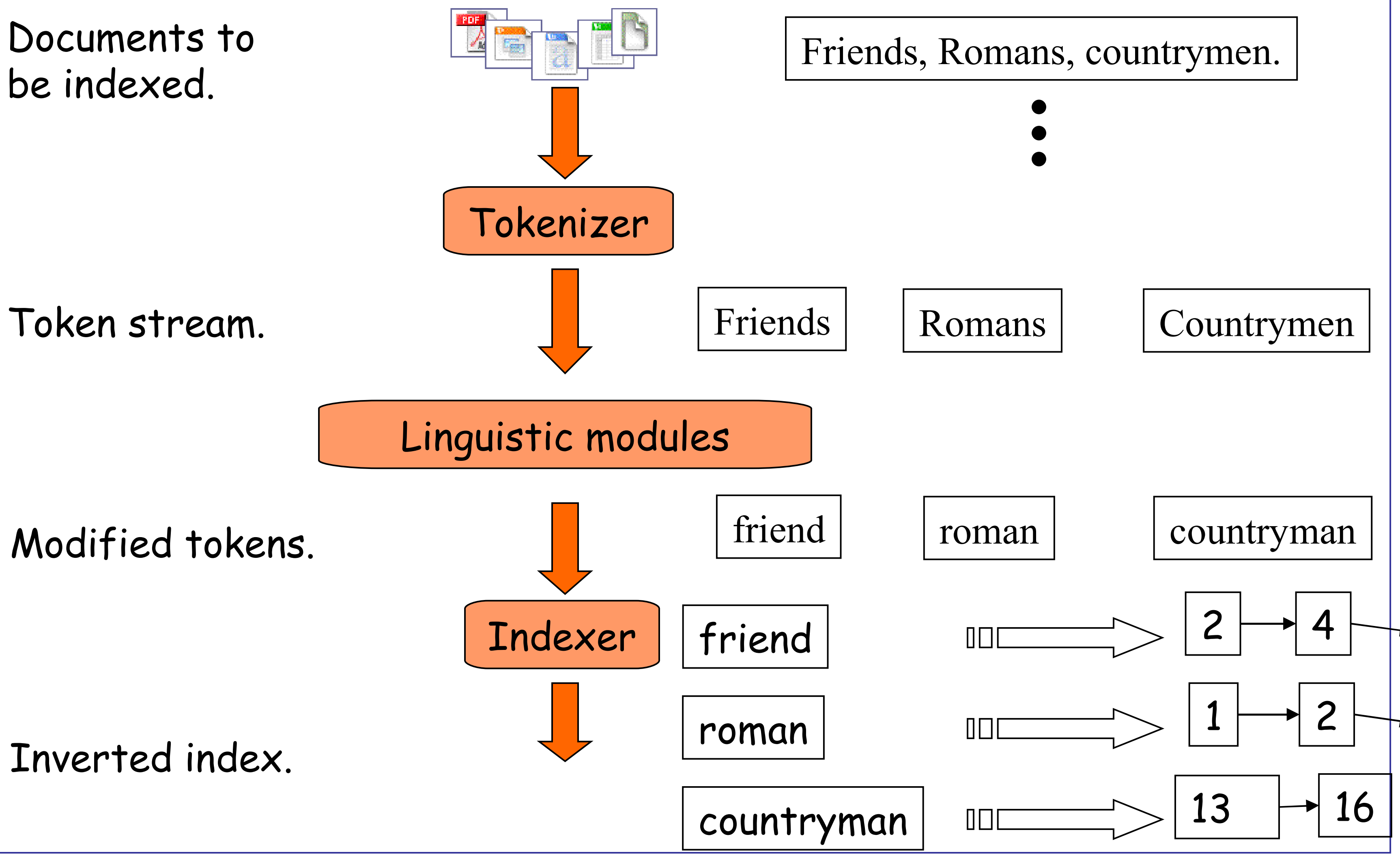
$$= \frac{\|d\| \cdot \|q\| \cdot \cos(\vartheta)}{\|d\| \cdot \|q\|}$$

$$= \cos(\vartheta)$$

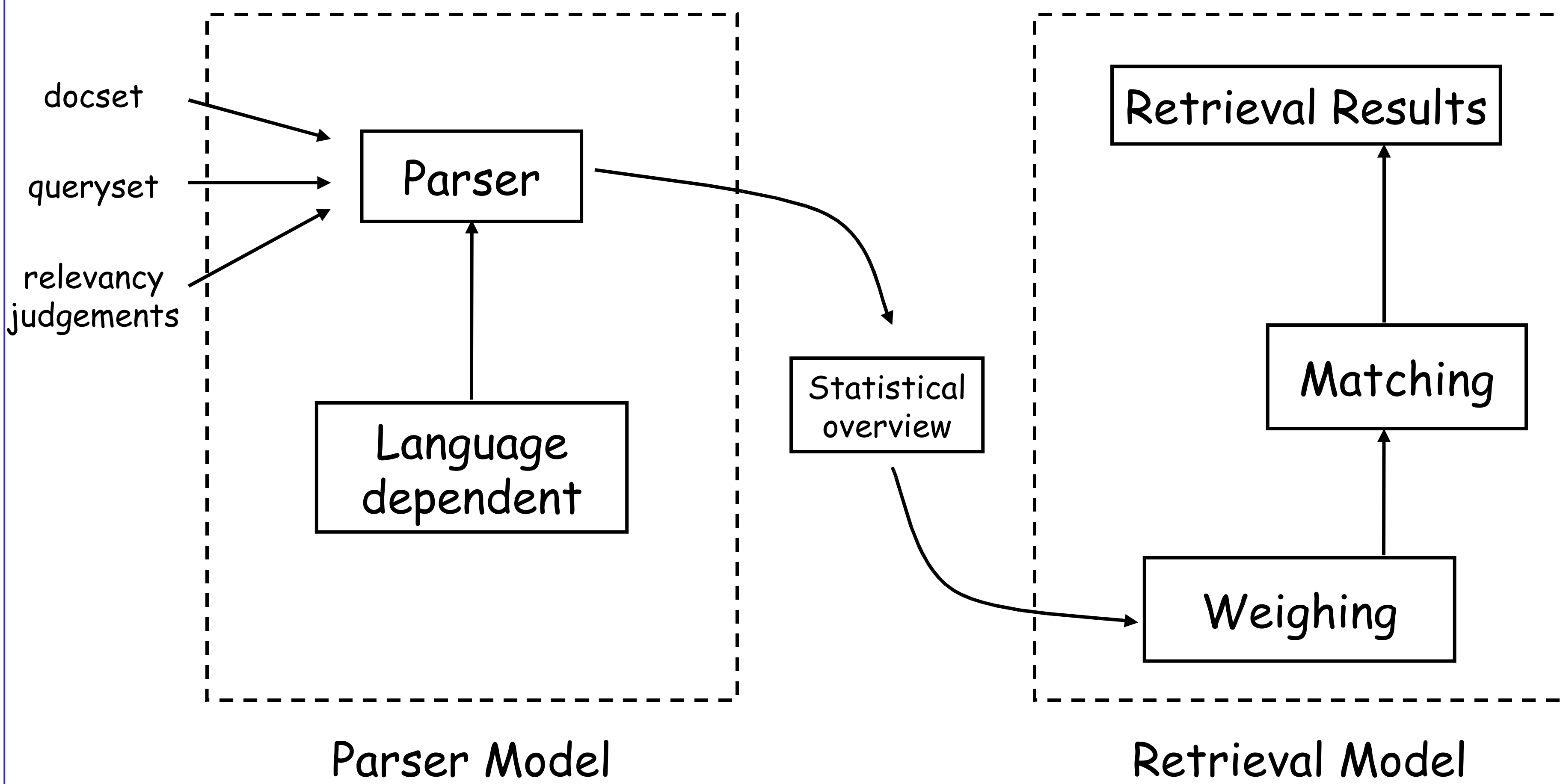


Indexing

Inverted index construction

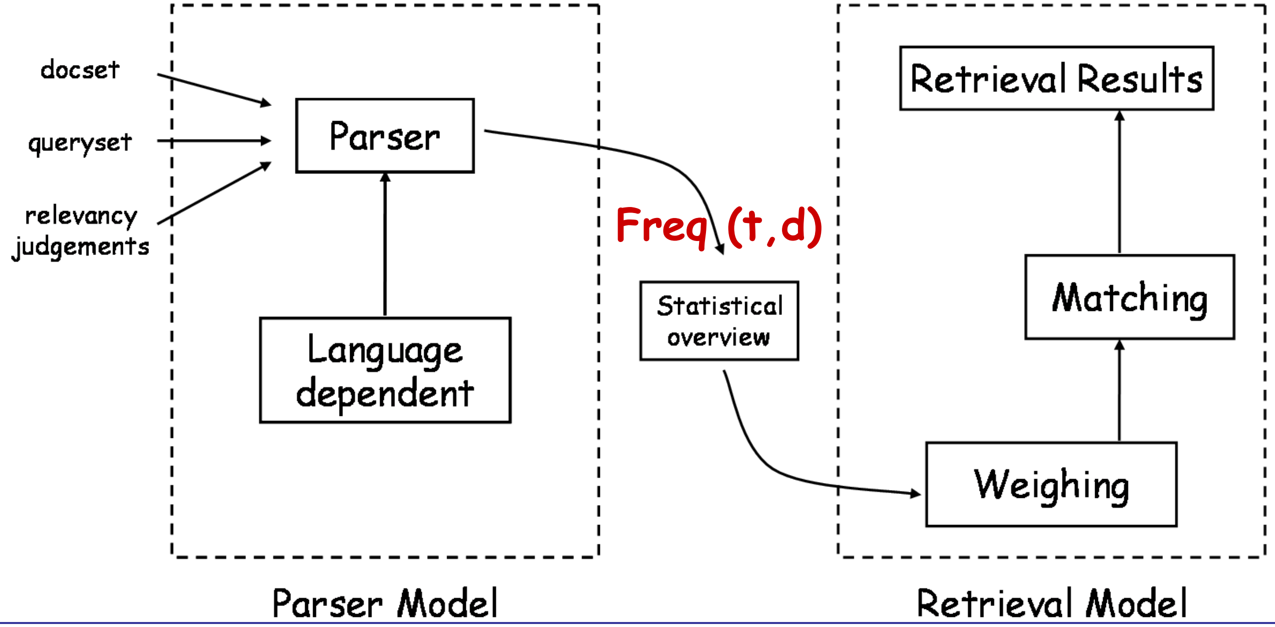


Advanced experimental architecture



Assigning weights

- We assume (simple) terms in the vector model.
- Statistical overview:
  - let Freq(t,d) be the frequency of term t in document d.
- Typical way of assigning weights to terms: TF-IDF.



TF-IDF weighting

- Normalized frequencies:

$$f(t,d) = \frac{Freq(t,d)}{\max_s Freq(s,d)}$$

- Inverse document frequency:

$$Idf(t) = {}^{2}\log \frac{N}{n(t)}$$

where n(t) is the number of documents containing term t

- Then:

$$a(d,t) = \underbrace{f(t,d)}_{\text{internal}} \cdot \underbrace{Idf(t)}_{\text{external}} = \frac{Freq(t,d)}{\max_s Freq(s,d)} \cdot \log \frac{N}{n(t)}$$

- Stopword t: n(t) = N. Then Idf (t) = 0, and thus a (d,t) = 0
- Noise word t: n(t) = 1. Then Freq(t,d_0) = 1 for document d_0 only, and Freq (t,d)=0 for the other documents. So:

$$a(d,t) = \begin{cases} \dfrac{1}{f_0}\log(N) & \text{if } d = d_0,\, f_0 \text{ max freq in } d_0 \\ 0 & \text{otherwise} \end{cases}$$

Normalization

- Each document gets assigned a vector this way.
- The document vectors are normalized to length 1.
- Assume the documents of D are numbered:

$$D = \{D_1, \ldots, D_n\},$$

- then $d_{i,j}$ is weight of term k_i in document D_j
- So: $D_i = (d_{i,1}. d_{i,2}, \ldots, d_{i,m})^T$

The query vector

- If the query vector is obtained from a description, then:

$$a(q,t) = \underbrace{\overline{f}(t,d)}_{\text{internal}} \cdot \underbrace{Idf(t)}_{\text{external}}$$

normalized to length 1,
where

$$\overline{f}(t,d) = avg\left(1, \frac{Freq(t,q)}{\max_s Freq(s,q)}\right)$$

$$= 0.5 + 0.5 \frac{Freq(t,q)}{\max_s Freq(s,q)}$$

The association matrix

Association Matrix

- The matching result is a vector that contains all similarities. Assuming vectors have unit length:

$$\begin{pmatrix} Sim(d_1, q) \\ \vdots \\ Sim(d_n, q) \end{pmatrix} = \begin{pmatrix} D_1 \bullet q \\ \vdots \\ D_n \bullet q \end{pmatrix} = \begin{pmatrix} D_1^T q \\ \vdots \\ D_n^T q \end{pmatrix} = \underbrace{\begin{bmatrix} D_1^T \\ \vdots \\ D_n^T \end{bmatrix}}_{\text{association matrix}} q$$

$$\begin{array}{c} \\ D_1 \\ D_2 \\ \vdots \\ \vdots \\ D_n \end{array} \begin{array}{c} \begin{array}{cccc} T_1 & T_2 & \cdots & T_m \end{array} \\ \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1m} \\ d_{21} & d_{22} & \cdots & d_{2m} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nm} \end{pmatrix} \end{array}$$

Dual view on Association Matrix

- The association matrix may be viewed from
  - the document view, where D_i is the document vector for document d_i

$$A = \begin{bmatrix} D_1^T \\ \vdots \\ D_n^T \end{bmatrix} \qquad D_i = \begin{bmatrix} d_{i1} \\ \vdots \\ d_{im} \end{bmatrix}$$

$$\begin{array}{c|cccc} & T_1 & T_2 & \dots & T_m \\ \hline D_1 & d_{11} & d_{12} & \dots & d_{1m} \\ D_2 & d_{21} & d_{22} & \dots & d_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ D_n & d_{n1} & d_{n2} & \dots & d_{nm} \end{array}$$

  - the term view, where T_j is a term vector for term t_j:

$$A = \begin{bmatrix} T_1 & \dots & T_m \end{bmatrix} \qquad T_j = \begin{bmatrix} d_{1j} \\ \vdots \\ d_{nj} \end{bmatrix}$$

$$\begin{array}{c|cccc} & T_1 & T_2 & \dots & T_m \\ \hline D_1 & d_{11} & d_{12} & \dots & d_{1m} \\ D_2 & d_{21} & d_{22} & \dots & d_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ D_n & d_{n1} & d_{n2} & \dots & d_{nm} \end{array}$$

Dual interpretation of query result

- The evaluation of query q consists of the evaluation of Aq
- document view: query result as document similarities

$$Aq = \begin{bmatrix} D_1^T \\ \vdots \\ D_n^T \end{bmatrix} q = \begin{pmatrix} D_1^T q \\ \vdots \\ D_n^T q \end{pmatrix} = \begin{pmatrix} Sim(D_1, q) \\ \vdots \\ Sim(D_n, q) \end{pmatrix}$$

- term view: query result as linear combination of term vectors

$$Aq = \begin{bmatrix} T_1 & \dots & T_m \end{bmatrix} q = \sum_{j=1}^{m} q_j T_j$$

- Matrix A
  - transforms meaning,
  - transforms an intentional description into an extensional description

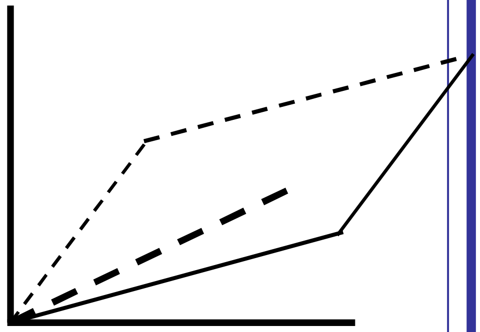
Term vector interpretation

- Each term vector T_j describes the meaning of term t_j as a weighted collection of documents, assuming a document represents a (materialized) elementary meaning unit.

- A query q then represents a compound meaning unit that can be obtained from the collection.

- This meaning is described as a linear combination of elementary meaning units:

$$Aq = \begin{bmatrix} T_1 & \ldots & T_m \end{bmatrix} q = \sum_{j=1}^{m} q_j T_j$$



- In terms of matrices: the image space of A

Not supported information need

- A query q (≠ 0) is not supported if $Aq = 0$
  - i.e. the meaning of the query is not present in the collection
  - in that case, for each i we have: Sim $(d_i, q) = 0$
- This is also referred to as the null space of A, defined as the set of solutions of the equation:

$$Aq = \sum_{j=1}^{m} q_j T_j = 0$$

- Example: $q = (3\ {-2})^T$ is not supported as it has no result!

$$\begin{pmatrix} 2 & 3 \\ 4 & 6 \end{pmatrix} \begin{pmatrix} 3 \\ -2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Pure support

- (In case of a square matrix) An information need q for which

  $$Aq = \lambda q$$

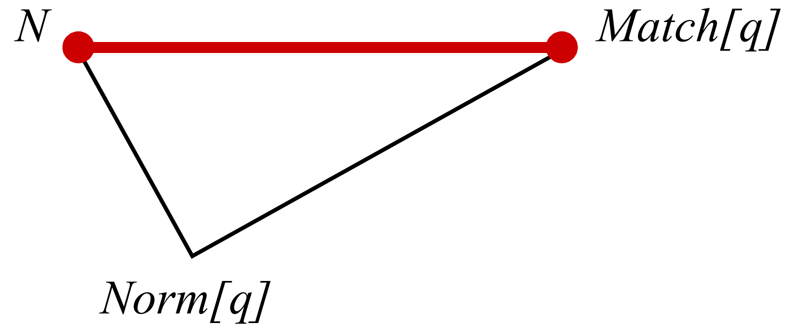
  has a pure support from the document collection described by association matrix *A*.

- q is a main topic from this collection with reinforcement factor λ

- Formal terms: eigenvalue and eigenvector of *A*

Contents

The ultimate judgment



$$\underbrace{\Delta(N, Match[q])}_{hopefully\ \ small}$$

$$\leq \underbrace{\Delta(N, Norm[q])}_{formulate} + \underbrace{\Delta(Norm[q], Match[q])}_{compute}$$

- IR is about:
  - satisfying vague information needs provided by users (imprecisely specified in ambiguous natural language)
  - by satisfying them approximately against information provided by authors (specified in the same ambiguous natural language)

(Smeaton)

Exact science?

- In what ways can a document be relevant to a query (have value)?
  - Answer precise question precisely
  - Partially answer question
  - Suggest a source for more information
  - Give background information
  - Remind the user of other knowledge
  - Others ...
- How relevant is the document
  - (subjective) for this particular searcher
  - (cognitive) for this particular information need
  - (situational) in this particular situation
  - (dynamic) at this particular moment
- Subjective, but measurable to some extent
  - How often do people agree a document is relevant to a query

What is value?

- Our value mechanism bares similarity to the three aspects of architecture as formulated by the Roman architect Vitruvius;
  - utilitas corresponds to our informational aspect of value,
  - firmitas corresponds to our structural aspect of value,
  - venustas corresponds to the emotional aspect of value.

This complex value domain can be used to study transactors

Marcus Vitruvius Poll(i)o
(±85—20BC

No ‘exact’ science!

- Evaluation is not done analytically, but experimentally
  - real users (specifying requests)
  - test collections (real document collections)
  - benchmarks (TREC: text retrieval conference)

because:

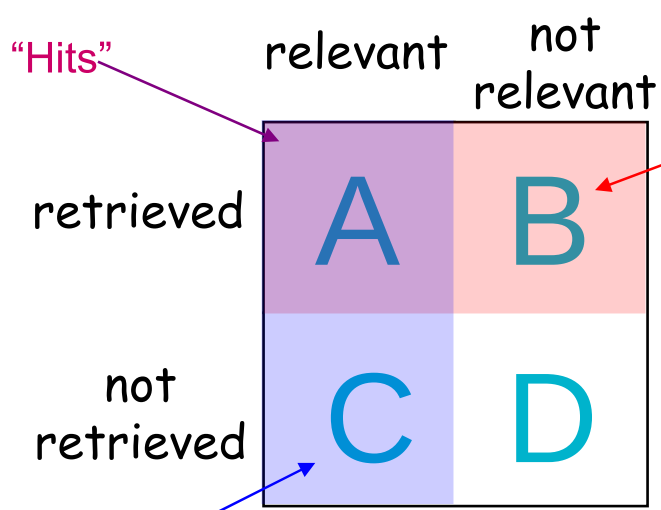
"In theory is there is no difference between theory and practice.
In practice there is."
(Jan LA van de Snepscheut)

Evaluation of retrieval system

- What can be measured that reflects the searcher's ability to use a system? (Cleverdon, 1966)
  - Coverage of Information: Extent to which any/all relevant items are included in the document corpus.
  - Form of Presentation: Influence of search output format on the user's ability to utilize the retrieved materials.
  - Effort required/Ease of Use: Work required from the user in formulating queries, conducting the search, and screening the output.
  - Time and Space Efficiency (response time): Time interval between receipt of a user query and the presentation of system responses.
  - Recall
  - Precision

Effectiveness

Precision and Recall

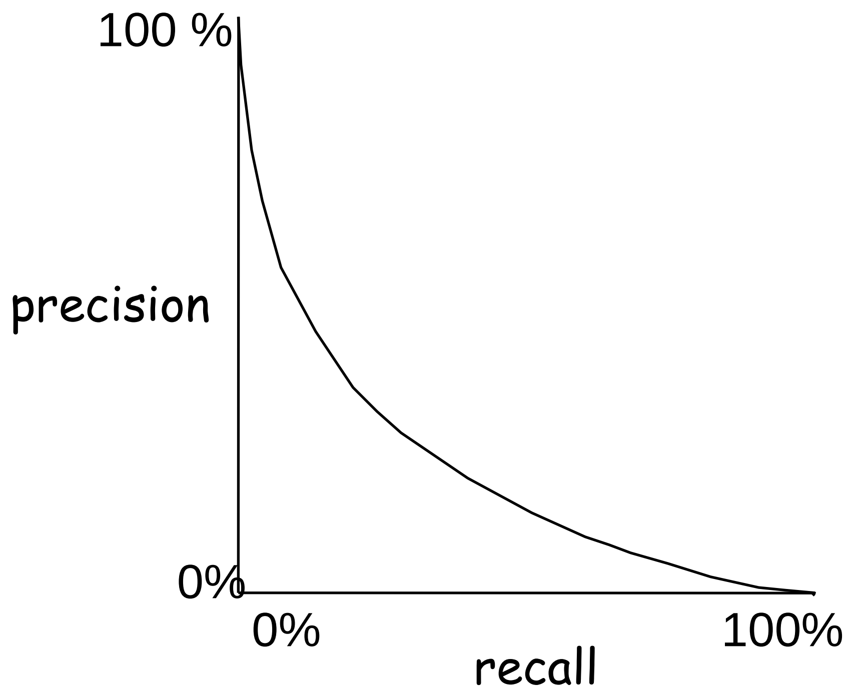


"Type one errors" "Errors of commission" "False positives"

"Type two errors"
"Errors of omission"
"False negatives"

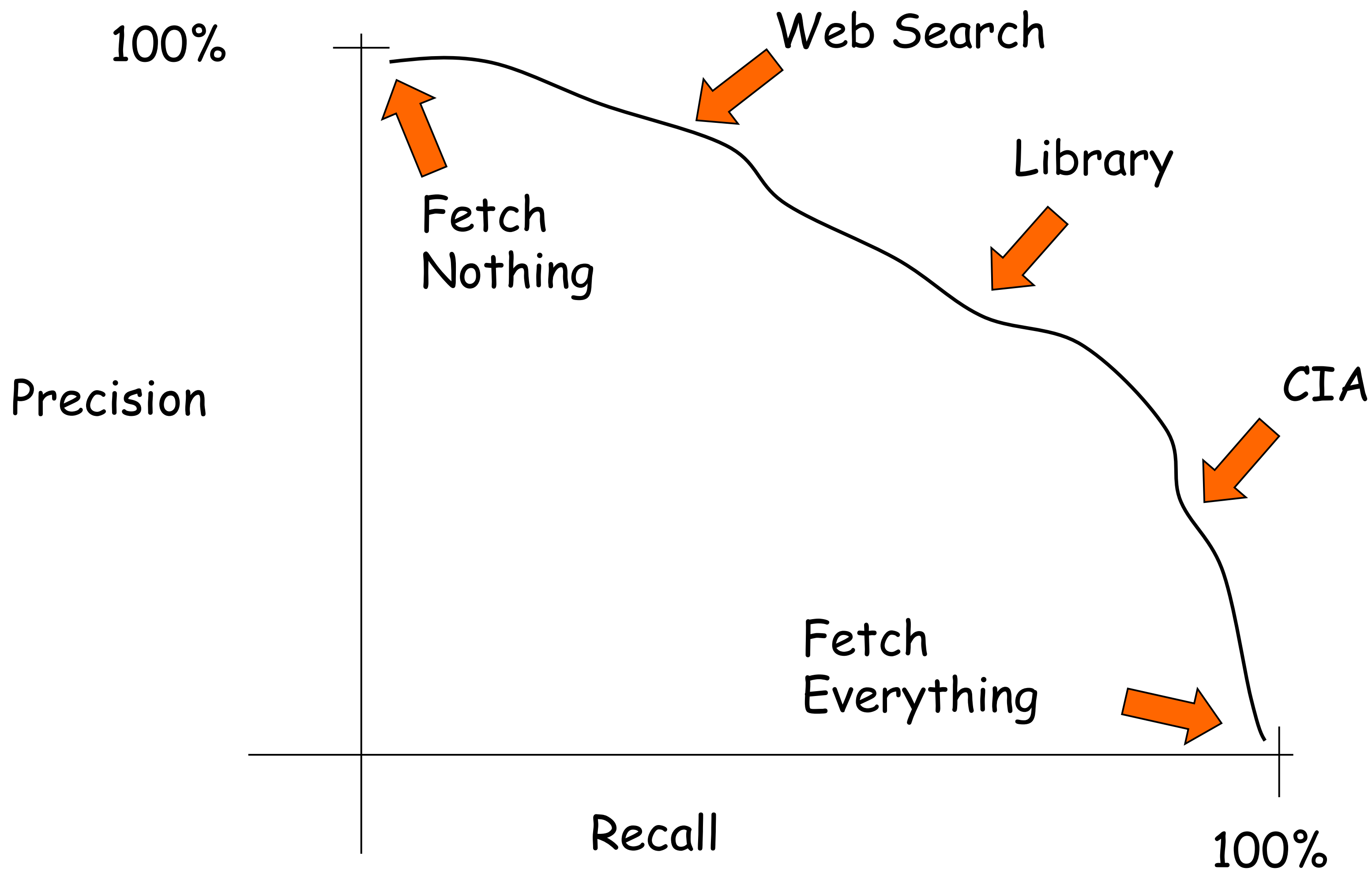
$$precision = \frac{A}{A+B} = \frac{|relevant \cap retrieved|}{|retrieved|}$$

$$recall = \frac{A}{A+C} = \frac{|relevant \cap retrieved|}{|relevant|}$$



Average precision = area under curve

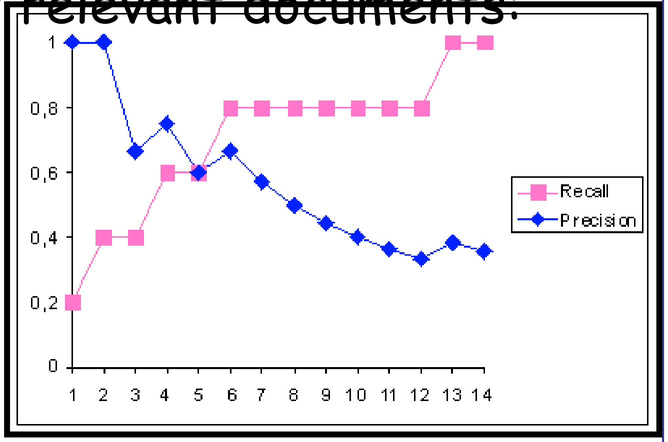
Precision-Recall Tradeoff



Average precision

- Let query q lead to a result list of documents, and let res(q) = $\{r_1, r_2, .., r_g\}$ be the positions where the relevant documents are found in this list.
  - Example: (red relevant)
    matching[q] = $d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}, d_{11}, d_{12}, d_{13}, d_{14}$
    then res(q) = {1, 2, 4, 5, 13}

- Evaluate recall and precision at positions of ~~relevant documents:~~



- at position r_i of i-th relevant document:
  - recall: i / g
  - precision: i / r_i

- The average precision is taken over these positions

Average precision

- Let query q lead to a result list of documents, and let res(q) = <r_1, r_2, ..r_g> be the subsequent positions where the relevant documents are found in this list.

- Then the average precision is defined as:

$$AP(q) = \frac{1}{g}\sum_{i=1}^{g}\frac{i}{r_i}$$

Examples:
- {1,2,3} AP = 1.00
- {1,2,4} AP = 0.92
- {1,2,5} AP = 0.87
- {2,3,4} AP = 0.64
- {3,4,5} AP = 0.48

Conclusion: high positions are highly rewarded!

MAP (Mean Average Precision)

Assume queries Q = {q_1, …, q_n}

The mean average precision for this collection of queries is defined as:

$$MAP(Q) = \frac{1}{|Q|} \sum_{q \in Q} AP(q) \qquad AP(q) = \frac{1}{g} \sum_{i=1}^{g} \frac{i}{r_i}$$

E.g. Rank:

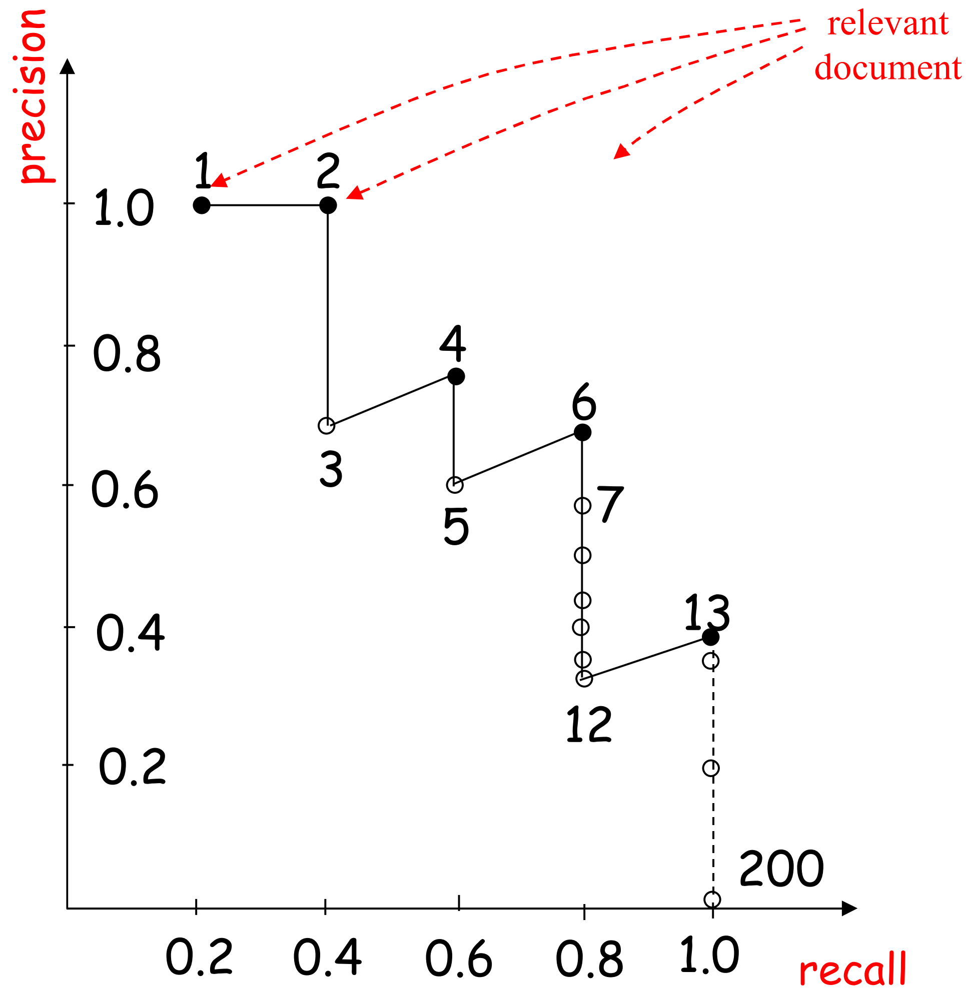
1	4
5	8
10	

1st rel. doc.
2nd rel. doc.
3rd rel. doc.

$$MAP = \frac{1}{2}\left[\frac{1}{3}\left(\frac{1}{1}+\frac{2}{5}+\frac{3}{10}\right) + \frac{1}{2}\left(\frac{1}{4}+\frac{2}{8}\right)\right] = 0.41$$

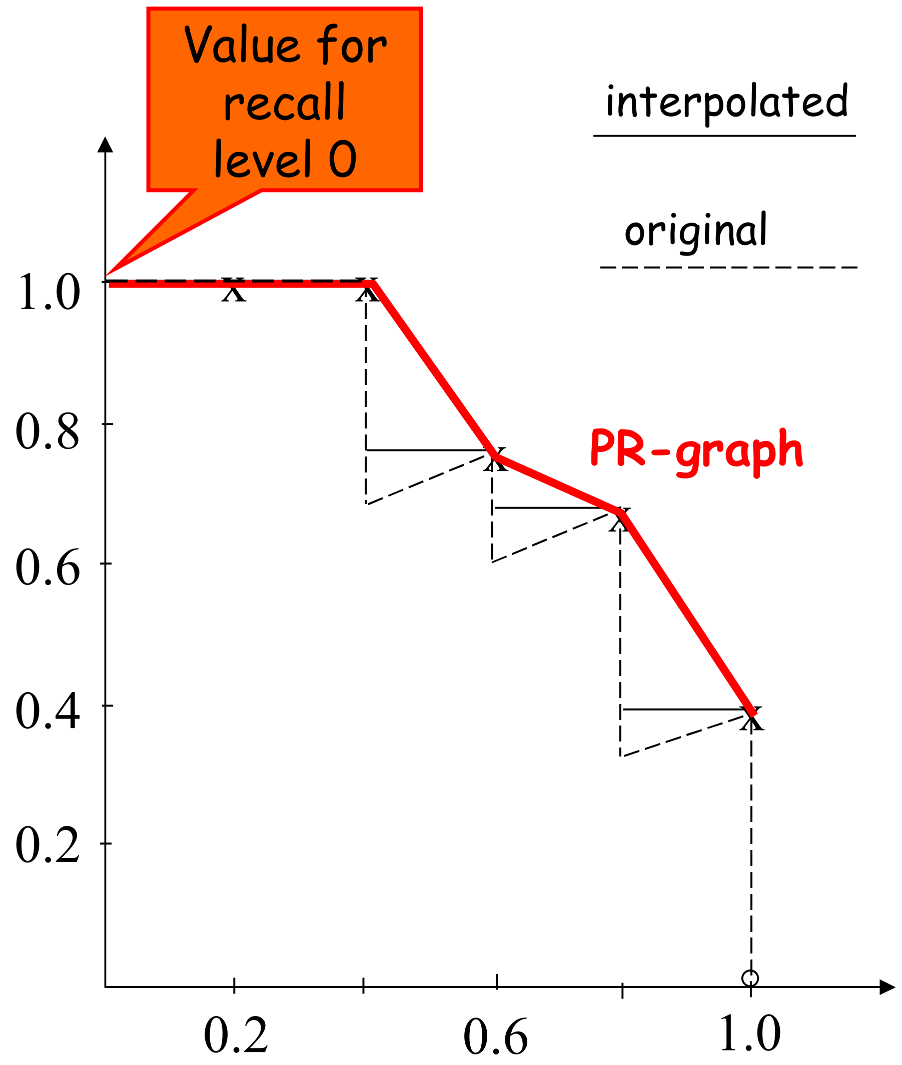
Computation of Recall and Precision

n	Recall	Precision
1	0.2	1.00
2	0.4	1.00
3	0.4	0.67
4	0.6	0.76
5	0.6	0.60
6	0.8	0.67
7	0.8	0.57
8	0.8	0.50
9	0.8	0.44
10	0.8	0.40
11	0.8	0.36
12	0.8	0.33
13	1.0	0.38
14	1.0	0.36



Interpolated Recall-Precision Graph

- For certain recall (precision) precision (recall) is not specified
- For more than one curve (query), how to get the average one?
- Using interpolated curve -- the best performance a user can achieve
- Special point: recall level 0



11-point average computation

- Query q seen as sequence of (r, p) pairs.
- P[q](r) = interpolated p value

- Micro average the precision figures at each recall level

$$\overline{P}(r) = \frac{1}{|Q|}\sum_{q\in Q} P[q](r)$$

- Compute the 11-point average:

$$Avg_{11} = \frac{1}{11}\sum_{i=0}^{10} \overline{P}(\tfrac{1}{10} i)$$

Contents

Query Modification

- Improving initial query formulation
  - Relevance feedback

    approaches based on feedback information from searchers
  - Local analysis

    approaches based on information derived from the set of documents initially retrieved (called the local set of documents)
  - Global analysis

    approaches based on global information derived from the document collection

Relevance Feedback

- Relevance feedback process
  - it shields the user from the details of the query reformulation process
  - it breaks down the whole searching task into a sequence of small steps which are easier to grasp
  - it provides a controlled process designed to emphasize some terms and de-emphasize others

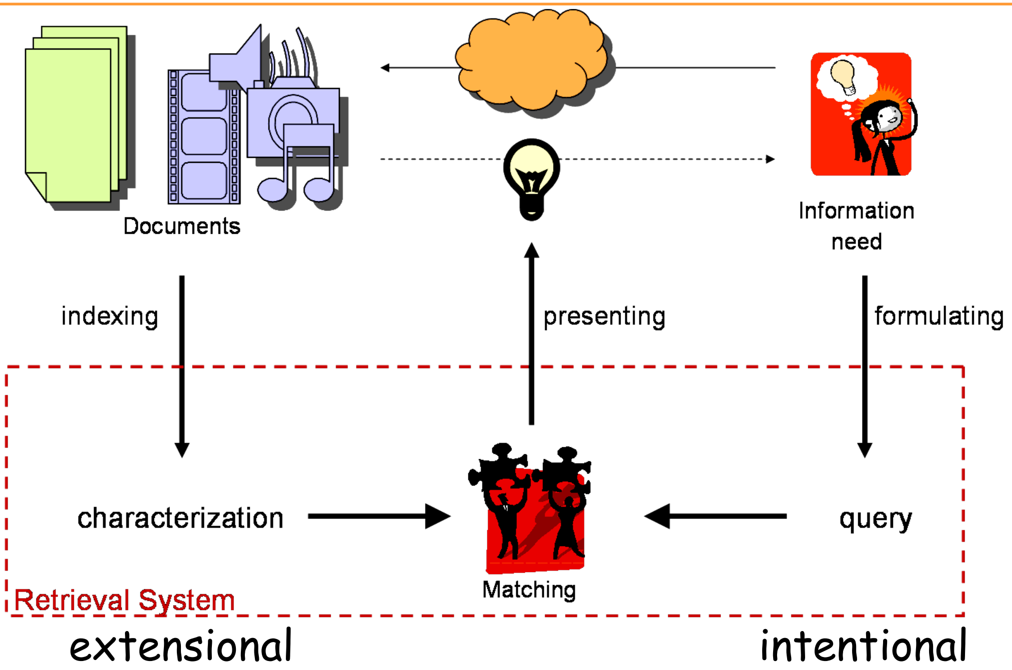
- Move toward relevant documents

- Move away from irrelevant documents

Basic techniques

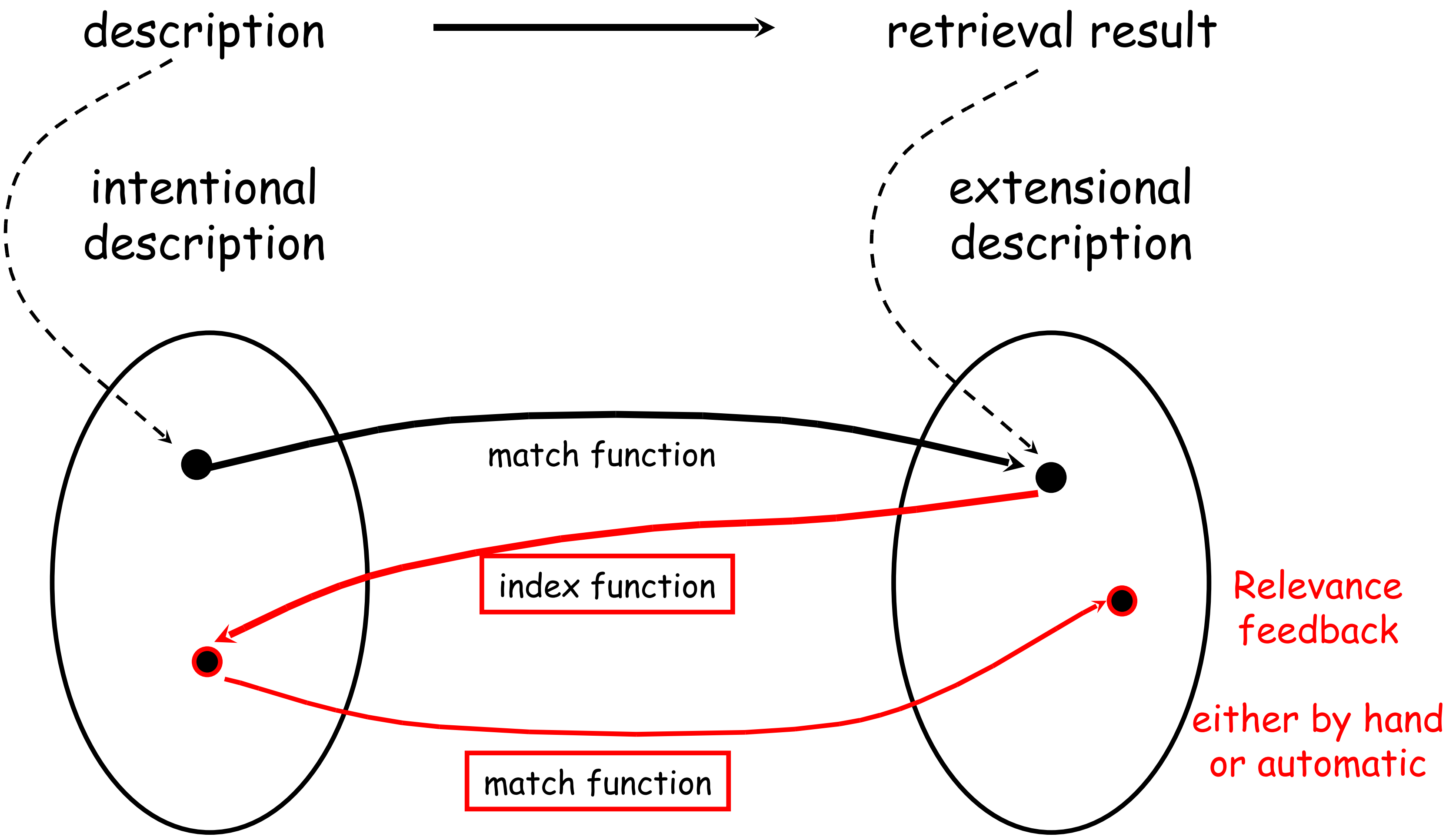
- Query expansion
  - addition of new terms from relevant documents
- Term reweighing
  - modification of term weights based on the user relevance judgment:
  - Increase weight of terms in relevant documents
  - decrease weight of terms in irrelevant documents.

Transformation of meaning description



- Information Retrieval may be seen as a transformational problem:
  - matching: transform an intentional description of meaning into an extensional description
  - indexing: transform an extensional description into an intentional desctiption

A dualistic view



Feedback algorithm

- The algorithm:

  Evaluate query q
  repeat
    Offer k most relevant documents: T
    Ask feedback, splitting T into
      set R of relevant documents and
      S of nonrelevant documents.
    Compute modified query q_m
    Evaluate modified query q_m
  until satisfied

Optimal query

- Problem:
  - given
    - set R of relevant documents
    - set S of irrelevant documents
  - find a query q that best generalizes R and S

- Solution: use bonus-malus strategy
  - bonus: similarity with relevant document
  - malus: similarity with irrelevant document

Notation

- Notations:

$$Sum(S) = \sum_{d \in S} d$$

$$Avg(S) = \begin{cases} \frac{1}{|S|} \sum_{d \in S} d & if \;\; S \neq \phi \\ 0 & otherwise \end{cases}$$

- $Avg(S)$ is called the centroid of S

Optimal query

- bonus: average similarity with relevant document

$$\frac{1}{|R|}\sum_{d\in R}(d \bullet q)$$

- malus: average similarity with irrelevant document

$$\frac{1}{|S|}\sum_{d\in S}(d \bullet q)$$

- Total score: bonus - malus

Optimization problem

- Find query q, ||q || = 1, that maximizes the bonus - malus score:

$$score(q) = \frac{1}{|R|}\sum_{d\in R}(d \bullet q) - \frac{1}{|S|}\sum_{d\in S}(d \bullet q)$$

$$= \left(\frac{1}{|R|}\sum_{d\in R} d\right) \bullet q - \left(\frac{1}{|S|}\sum_{d\in S} d\right) \bullet q$$

$$= \left(Avg(R) - Avg(S)\right) \bullet q$$

- Optimal for $q = \frac{Avg(R) - Avg(S)}{|Avg(R) - Avg(S)|}$

cos (α) maximal when α = 0

Rocchio method

- Rocchio (1965, 1971)
  - R: set of relevant documents, as identified by the user among the retrieved documents
  - S: set of non-relevant documents among the retrieved documents
  - q: the initial query

$$
\begin{aligned}
q_m &= Mix(q, Avg(R), Avg(S)) \\
&= \alpha \cdot q + \beta \cdot Avg(R) - \gamma \cdot Avg(S)
\end{aligned}
$$

SVD

Interpretation of meaning

- The matrix A^TA gives an impression of term similarities

$$A = \begin{bmatrix} T_1 & \dots & T_m \end{bmatrix}$$

  - Let q be some term vector, then the i-th component of $A^TA\ q$ is the cumulative contribution from the components q_k via the similarity between T_k and T_i:
    - $(A^TA\ q)_i = \Sigma_k\ (A^TA)_{i,k}\ q_k = \Sigma_k\ (T_i^T\ T_k)\ q_k$
    - Contribution thus is:
      raw similarity of T_k with $T_i \times$ provision of T_k in q

- So A^TA q is an interpretation of q
  - in terms of the collection
  - as its validating effect on all terms.

- So $A^T A$ q is an interpretation of q
  - in terms of the collection
  - as its validating effect on all terms.

- Conclusion:

  Evaluate: $A\,(A^T A\, q)$

- The singular value decomposition provides a more fundamental approach.

Stability of meaning

- So $A^T A$ q is an interpretation of q
  - in terms of the collection
  - as its validating effect on all terms.
- An interesting question is:

  which terms are stable under this interpretation

- In terms of matrices:

  what are eigenvalues and eigenvectors of $A^T A$:

  $A^T A \; t = \lambda \; t$

Relation terms and documents

- Let $A^T A\, t = \lambda\, t$,

  then $A\, A^T A\, t = \lambda\, A\, t$

  which can be rewritten as: $A\, A^T d = \lambda\, d$
  where $d = A\, t$

  and thus $d = A\, t$ is an eigenvector of the document-document association matrix $A\, A^T$ with eigenvalue λ

- The combination $(t, A\, t)$ may be seen as a concept of strength λ

- So the term-term association matrix $A^T A$ and the document-document association matrix $A\, A^T$
  - have the same eigenvalues
  - the eigenvectors can be transformed into each other.

Symmetric Matrices

- If A is a symmetric matrix, then A can be decomposed according to its eigenvalues and eigenvectors. That is,

$$AV = V\Lambda \qquad (1)$$

where V is a matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues.

- Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of A, and $v_1, \ldots, v_n$ be the corresponding set of normalized eigenvectors, then:

$$\begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix}$$

$$AV = A[v_1 \ldots v_n] = [Av_1 \ldots Av_n] = [\lambda_1 v_1 \ldots \lambda_n v_n] = V\Delta(\lambda_1, \ldots \lambda_n)$$

What is Latent Semantic Indexing

- In the vector model of documents, terms are considered being independent.
  - It is a simplifying assumption that is not true.
  - In reality the terms have varying degrees of correlation or dependencies or associations.

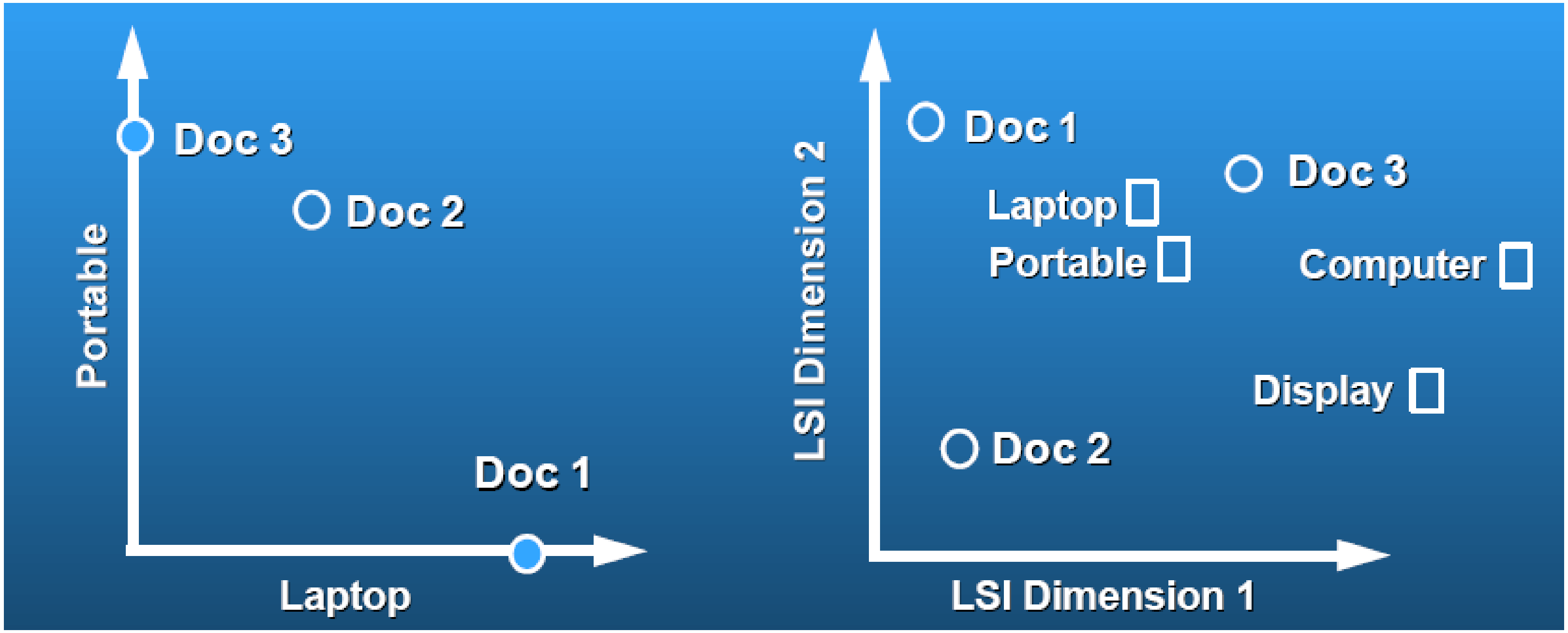
- Synonymy
  - widespread synonym occurances
  - decrease recall.
- Polysemy
  - retrieval of irrelevant documents
  - poor precision
- Noise
  - Boolean search on specific words
  - Retrieval of contently unrelated documents

What is Latent Semantic Indexing

- The latent semantic indexing (LSI) approach takes into account these associations between the terms by deriving a new set of indexing terms through a statistical method, known as singular value decomposition (SVD).
  - To find and fit a useful model of the relationships between terms and documents.
  - To find out what terms "really" are implied by a query .
- LSI
  - allows the user to search for concepts rather than specific words.
  - can retrieve documents related to a user's query even when the query and the documents do not share any common terms.
- The approach is termed LSI since
  - the new terms are "hidden", they are not directly found in the documents
  - and carry semantic information

Latent Semantic Analysis

- Latent semantic space: illustrating example



courtesy of Susan Dumais

How LSI Works?

- Uses a multidimensional vector space (the conceptual space) to place all documents and terms.
- Each dimension in that space corresponds to a concept existing in the collection.
- Thus underlying topics of the document are encoded in a concept vector.
- Common related terms in a document and query will pull document and query vector close to each other.

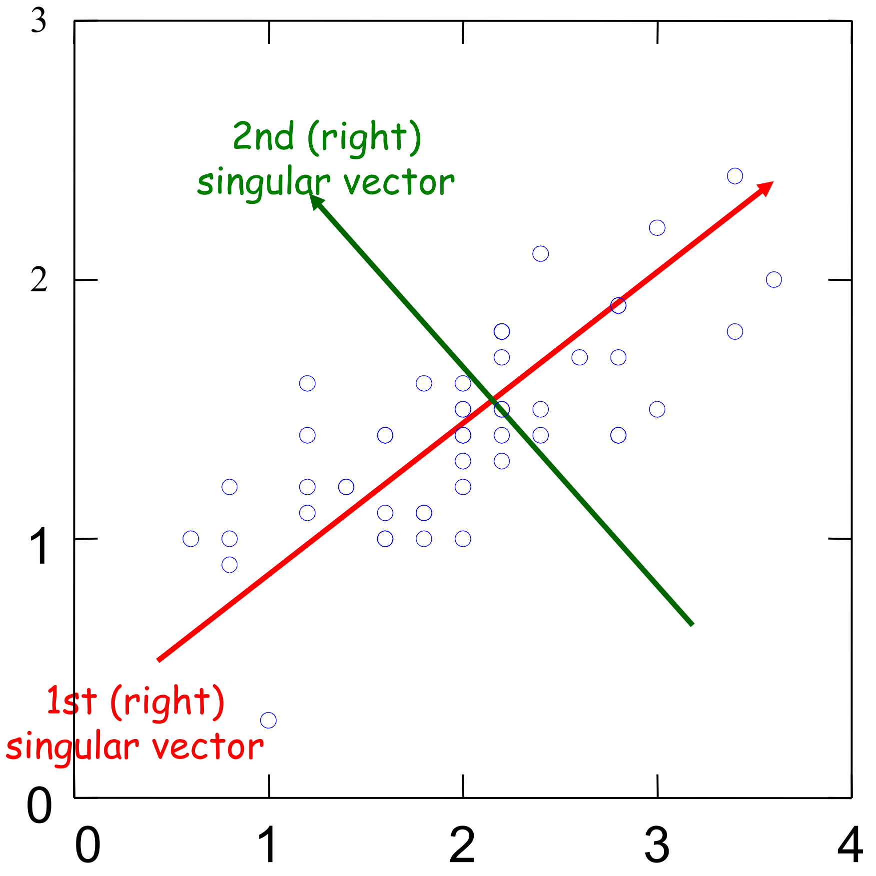
Advantages

- LSI analysis effectively does
  - Dimensionality reduction
  - Noise reduction
  - Exploitation of redundant data
  - Correlation analysis and Query expansion (with related words)
- Any one of the individual effects can be achieved with simpler techniques (see thesaurus construction).
- But LSI does all of them together

Drawback!

- The complexity of the LSI model obtained from truncated SVD is costly.
  - Storage
    LSI loses sparse nature of the term by document matrix.
  - Efficiency
    With LSI, the query must be compared to every document in the collection
- Its execution efficiency lag far behind the execution efficiency of the simpler, Boolean models, especially on large data sets.

SVD, intuition



Let the blue circles represent n documents. We have 2 terms.

Then, the SVD of the n x 2 matrix of the data will return ...

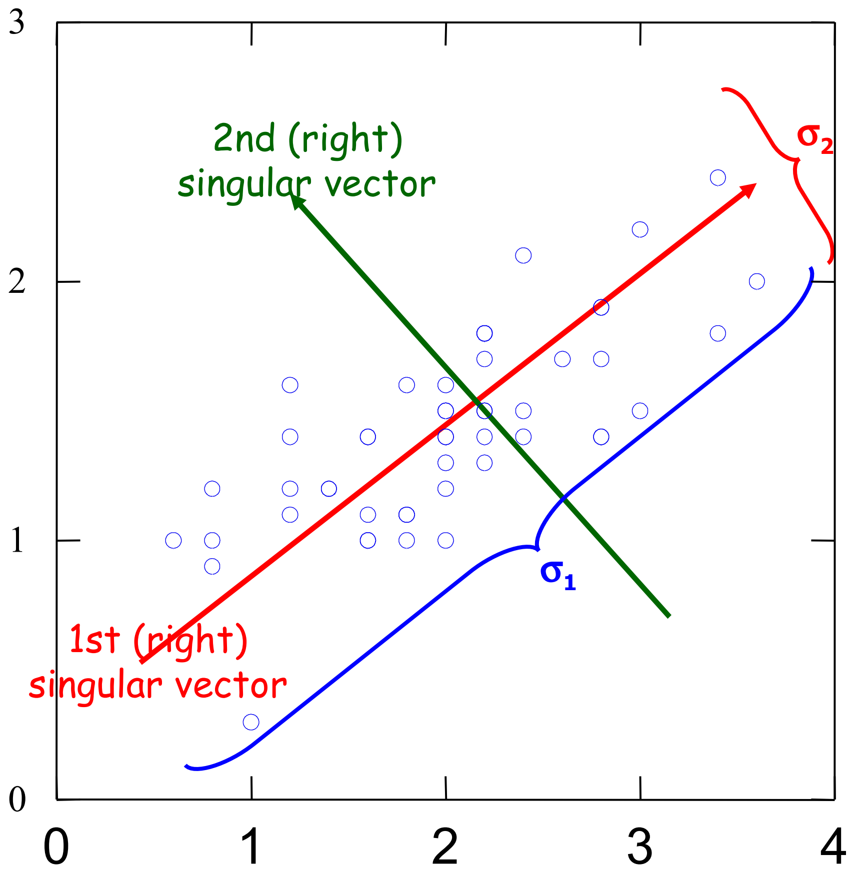
1st (right) singular vector:

direction of maximal variance,

2nd (right) singular vector:

direction of maximal variance, after removing the projection of the data along the first singular vector.

Singular Values



σ_1: measures how much of the data variance is explained by the first singular vector.

σ_2: measures how much of the data variance is explained by the second singular vector.

The decomposition

- According to SVD, an arbitrary matrix A of size $n \times m$ can be expressed as follows

$$A\ V = U\ \Sigma$$

> Remember:
> if A is a symmetric matrix,
> then $AP = P\Lambda$

- where
  - U and V are unitary matrices of size $n \times n$ and $m \times m$, respectively,
    - $U^TU = UU^T = I$ and $V^TV = VV^T = I$
  - Σ is a nxm matrix with a general diagonal entry σ_i, called a singular value of A.

- Since U and V are unitary matrices, we can also write

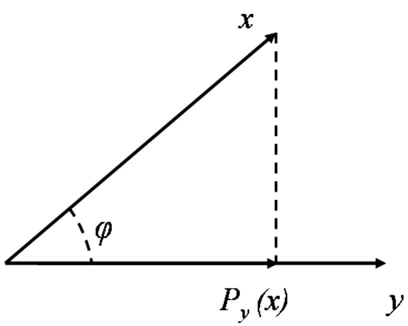
$$\underset{n\times m}{A} = \underset{n\times n}{U}\ \underset{n\times m}{\Sigma}\ \underset{m\times m}{V^T}$$

Geometric interpretation

term space

$$Ax = U\Lambda V^T x = U\Lambda \begin{pmatrix} v_1^T \\ \vdots \\ v_m^T \end{pmatrix} x = U\Lambda \begin{pmatrix} v_1^T x \\ \vdots \\ v_m^T x \end{pmatrix}$$

$$x^T y = x \bullet y = \left\| P_y(x) \right\| \cdot \left\| y \right\|$$



concept space

concept transformation

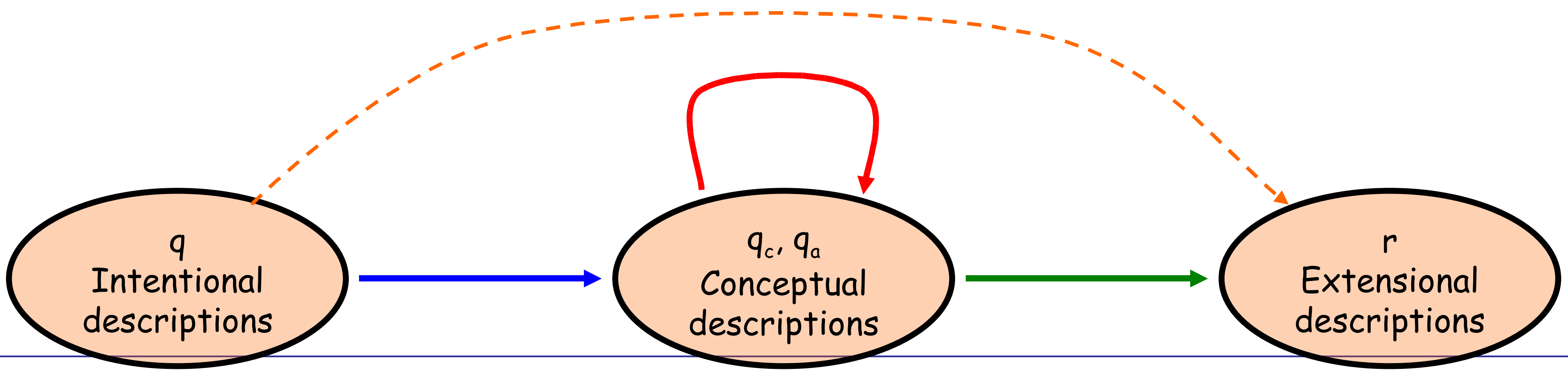
$$= U\Lambda \begin{pmatrix} \left\| P_{v_1}(x) \right\| \\ \vdots \\ \left\| P_{v_m}(x) \right\| \end{pmatrix} = U \begin{pmatrix} \sigma_1 \left\| P_{v_1}(x) \right\| \\ \vdots \\ \sigma_m \left\| P_{v_m}(x) \right\| \end{pmatrix}$$

$$= \sum_{j=1} \sigma_j \left\| P_{v_j}(x) \right\| \cdot u_j$$

document space

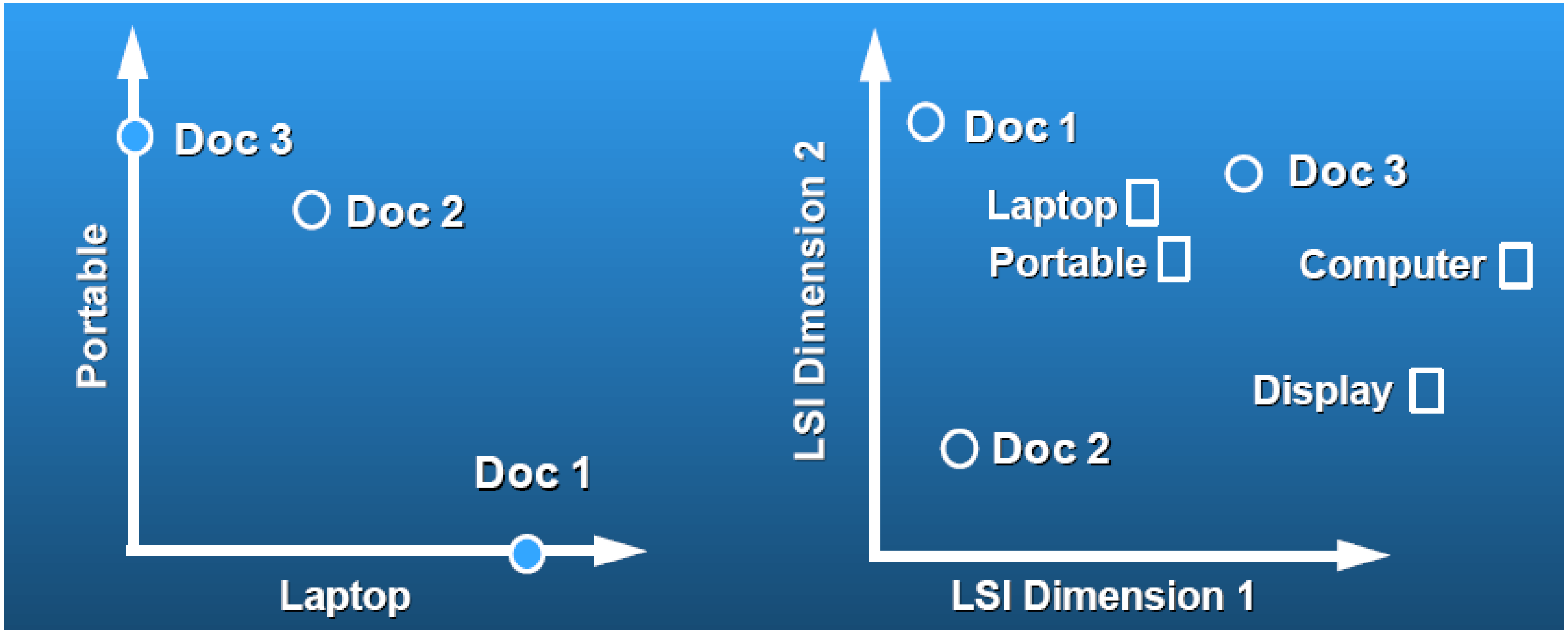
Query Evaluation

- $A\, q = (U\, \Sigma\, V^T)\, q$
- first transform query to concept space
  $= (U\, \Sigma)\, V^T q$ // conceptual query
  $= (U\, \Sigma)\, q_c$
- get concept amplification
  $= U\, (\Sigma\, q_c)$ // conceptual answer
- $= U\, q_a$
  transform to document space
  $= r$



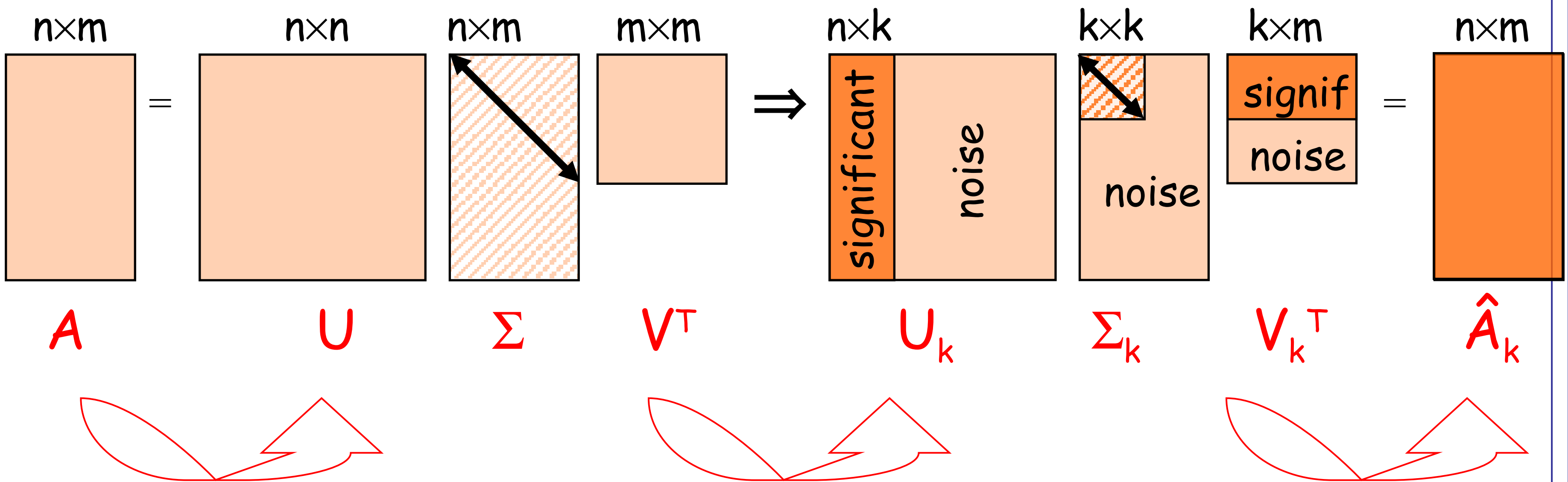
Latent Semantic Analysis

- Latent semantic space: illustrating example



courtesy of Susan Dumais

Summary of the approach



Singular Value Decomposition (SVD): Convert term-document matrix into 3 matrices U, Σ and V

Reduce Dimensionality: Throw out low-order rows and columns

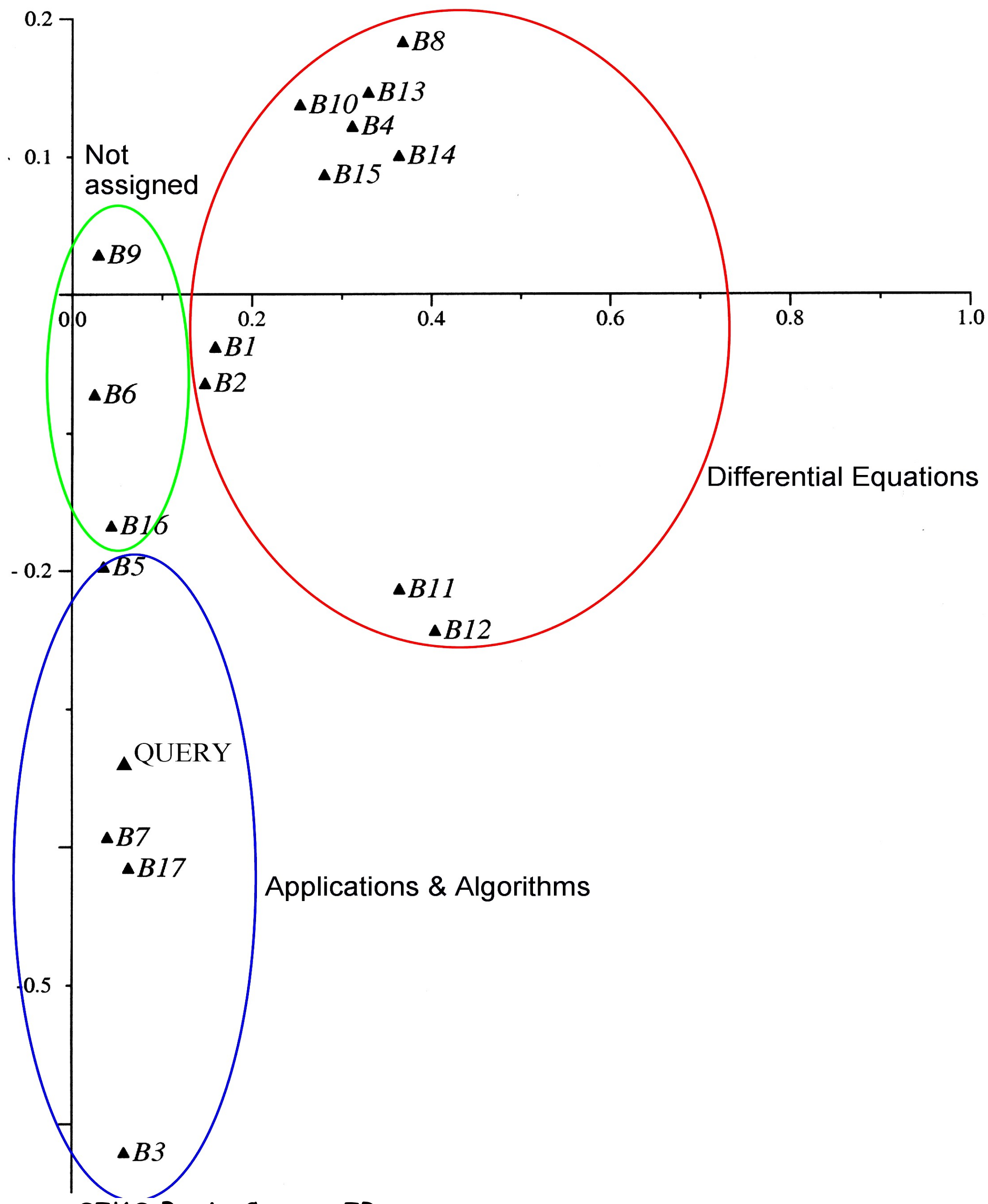
Recreate Matrix: Multiply to produce approximate term-document matrix. Use new matrix to process queries

Example Berry/Dumais/O'Brien

(Themen: AA = "Applications & Algorithms", DE = "Differential Equations", ? = Nicht zuweisbar)

Name	Thema	Titel
B1	DE	A Course on Integral Equations
B2	DE	Attractors for Semigroups and Evolution Equations
B3	AA	Automatic Differentiation of Algorithms: Theory, Implementation, and Applications
B4	DE	Geometrical Aspects of Partial Differential Equations
B5	AA	Ideals, Varieties, and Algorithms – An Introduction to Computational Algebraic Geometry and Commutative Algebra
B6	?	Introduction to Hamiltonian Dynamical Systems and the N-Body Problem
B7	AA	Knapsack Problems: Algorithms and Computer Implementations
B8	DE	Methods of Solving Singular Systems of Ordinary Differential Equations
B9	?	Nonlinear Systems
B10	DE	Ordinary Differential Equations
B11	DE	Oscillation Theory for Neutral Differential Equations with Delay
B12	DE	Oscillation Theory of Delay Differential Equations
B13	DE	Pseudodifferential Operators and Nonlinear Partial Differential Equations
B14	DE	Sinc Methods for Quadrature and Differential Equations
B15	DE	Stability of Stochastic Differential Equations
B16	?	The Boundary Integral Approach to Static and Dynamic Contact Problems
B17	AA	The Double Mellin-Barnes Type Integrals and their Applications to Convolution Theory

The semantic space for k = 2



reweighing

Reweigh Algorithm

- Evaluate query e
- Take randomly k documents ΔT, $T = \Delta T$
- repeat
  - Ask feedback about ΔT:

    $\Delta T = \Delta R \cup \Delta S$

    $R = R \cup \Delta R$, $S = S \cup \Delta S$, $T = R \cup S$

  - Compute p_i and q_i
  - Re-evaluate query: $d_1, d_2, d_3, ..$
  - Determine i such that:

    $\#(\{d_1, \ldots, d_i\} - T) = k$

  - $\Delta T = \{d_1, \ldots, d_i\} - T$

Retrieval status value

- Isolate document dependent part:

$$RSV(d) = \sum_t d_t \log\left(\frac{p_t}{1-p_t} \cdot \frac{1-q_t}{q_t}\right)$$

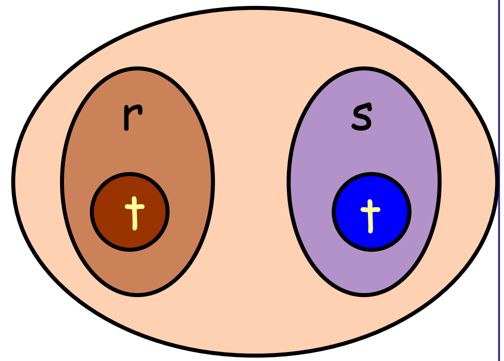
$$= \sum_{t \in d} \log\left(\frac{Odds(p_t)}{Odds(q_t)}\right)$$

- Remark: this may be interpreted as the inner vector product d•s where s is the newly constructed term weight vector!

$$s_t = \log\left(\frac{Odds(p_t)}{Odds(q_t)}\right)$$

Robertson-Sparck Jones Model

- Let
  - r: number of documents in R
  - s: number of documents in S
  - r_t: number of documents in R having term t
  - s_t: number of documents in S having term t



Then:

$$p_t = \frac{r_t}{r} \qquad q_t = \frac{s_t}{s}$$

Stephen Robertson

- (Robertson & Sparck Jones 76)

when r,s = 0

$$p_t = \frac{r_t + 0.5}{r + 1} \qquad q_t = \frac{s_t + 0.5}{s + 1}$$

- Instead of 0.5, alternative adjustments have been proposed

overall term prob

$$p_t = \frac{r_t + \frac{n_t}{N}}{r + 1} \qquad q_t = \frac{\boxed{n_t - r_t} + \frac{n_t}{N}}{n - r + 1}$$

s_t

Karen Sparck Jones

No Relevance Info

- We will assume p_i to be a constant (typically 0.5)
- Estimate q_i by assuming all documents to be non-relevant

(Croft & Harper 79)

$$p_t = \text{constant} \qquad q_t = \frac{N - r_t}{r_t}$$

Bruce Croft

point-5 formula as extension

$$q_t = \frac{N - r_t + 0.5}{r_t + 1}$$

David Harper

Probabilistic Model

- Definition
  - p_i: the probability of observing term t_i in the set of relevant documents
  - q_i: the probability of observing term t_i in the set of nonrelevant documents

$$sim(d,e) = \sum_{i=1}^{t} d_i \cdot e_i \cdot \log\left(\frac{p_i}{1-p_i} \frac{1-q_i}{q_i} \right)$$

Comparing the models

- The formula

$$sim(d,e) = \sum_{i=1}^{t} d_i \cdot e_i \cdot \log\left(\frac{p_i}{1-p_i}\frac{1-q_i}{q_i}\right)$$

- Could also be seen as evaluating modified query e'

$$sim(d,e')$$

$$where\ e'_i = e_i \cdot \log\left(\frac{p_i}{1-p_i}\frac{1-q_i}{q_i}\right)$$

Why is Feedback Not Widely Used

- Users sometimes reluctant to provide explicit feedback.
- Results in long queries that require more computation to retrieve, and search engines process lots of queries and allow little time for each one.
- Makes it harder to understand why a particular document was retrieved.

Pseudo feedback

- Use relevance feedback methods without explicit user input.
- Just assume the top m retrieved documents are relevant, and use them to reformulate the query.
- Allows for query expansion that includes terms that are correlated with the query

Contents

1. General Architecture
2. The Information Retrieval Problem
3. Classic Models
4. Quality Measures
5. Query Modification
6. Conceptual Decomposition

Grasping natural language

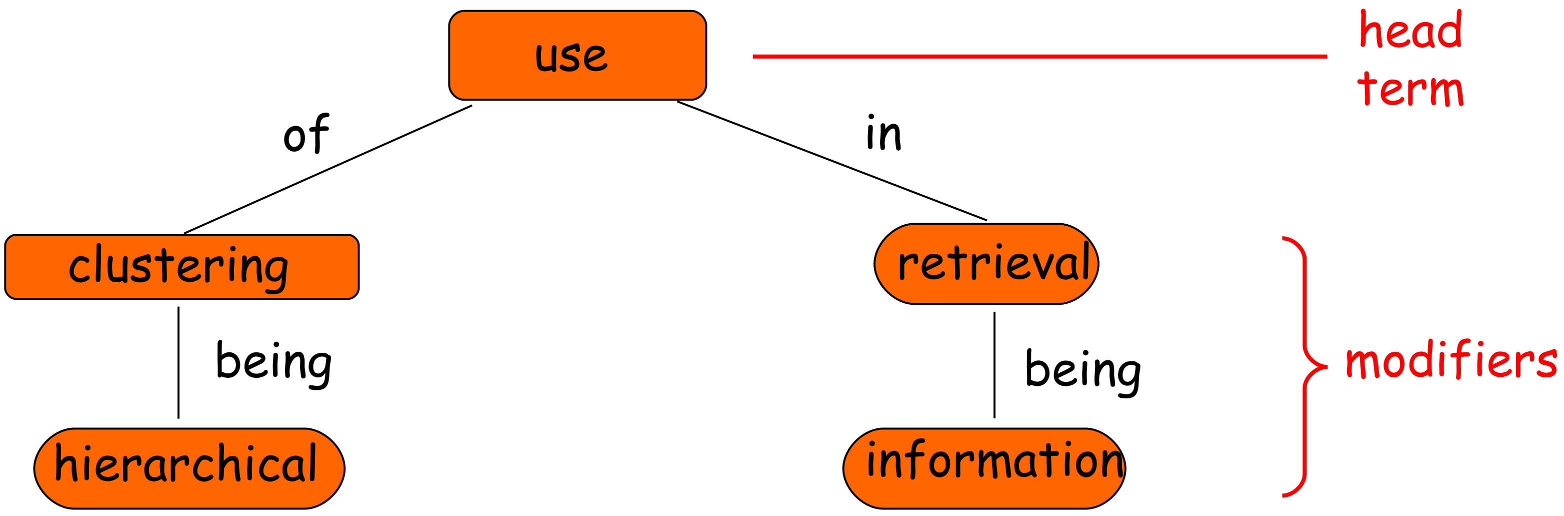
- Idea:
  - natural language is closely related to human cognition
  - concepts of natural language are meaningful for human beings
  - they reflect their common view on the real world
  - and are a way to exchange and share knowledge

- Main concepts of natural language:
  - verb phrase
  - noun phrase

- We summarize index expressions and query by navigation

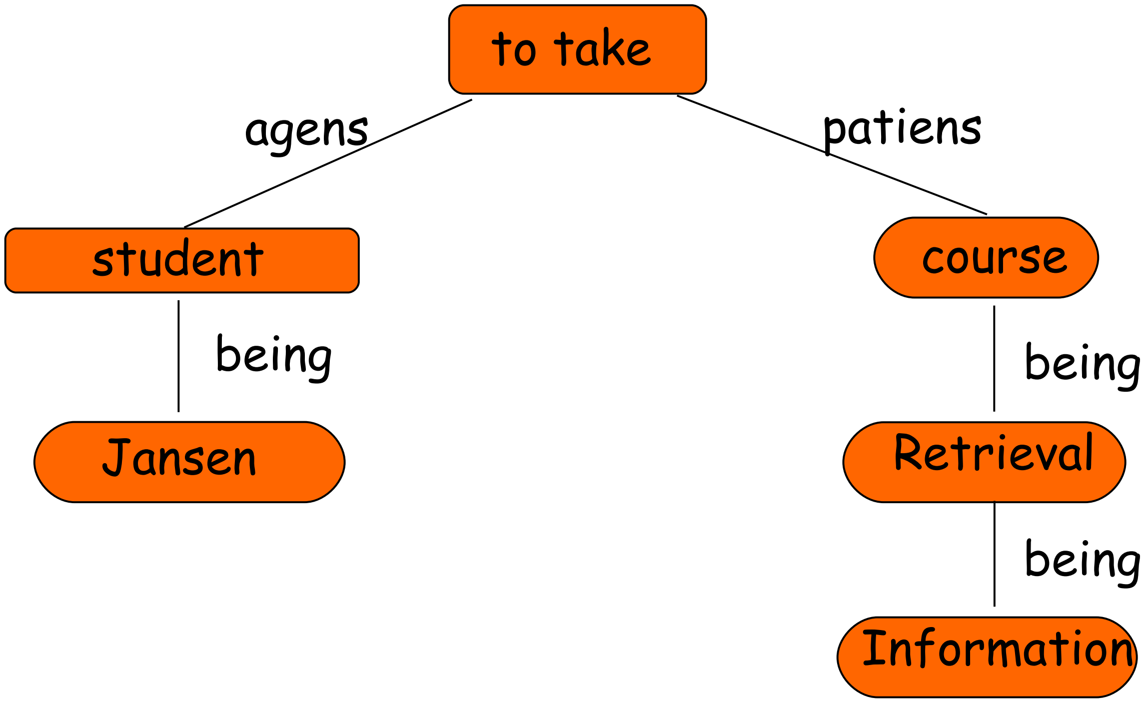
Approximation natural language

- Approximation noun phrase by index expressions
  - the use of hierarchical clustering in information retrieval



Approximation natural language

- Approximation verb phrase by index expressions
  - student Jansen takes course Information Retrieval

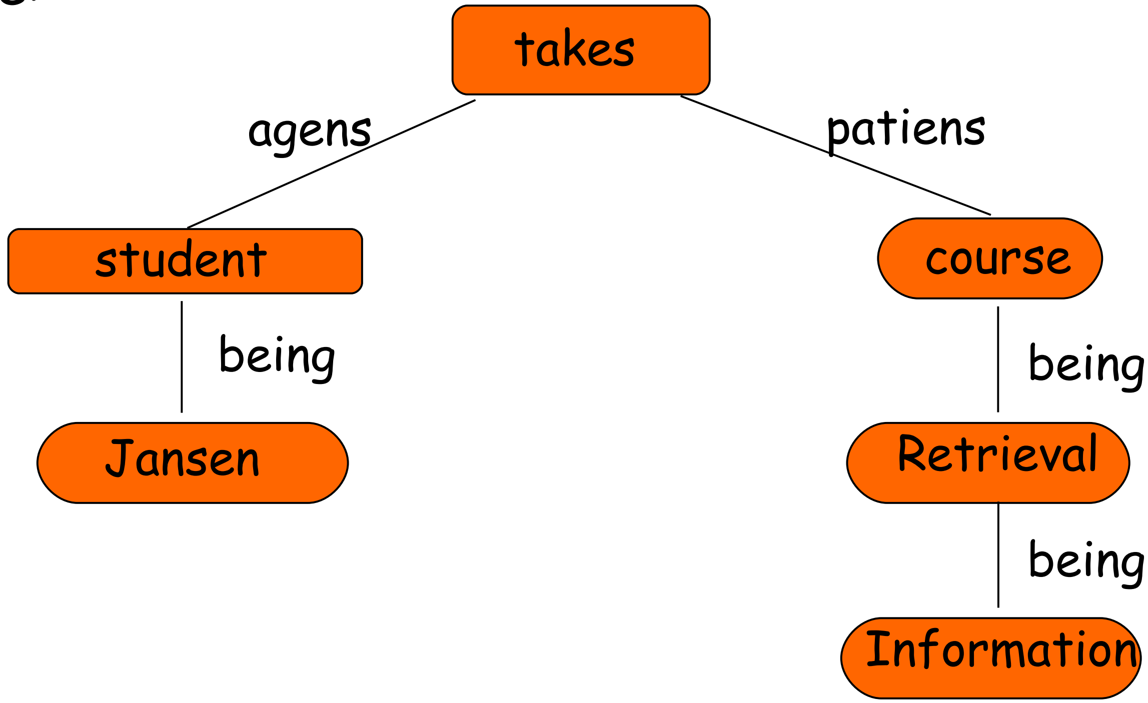


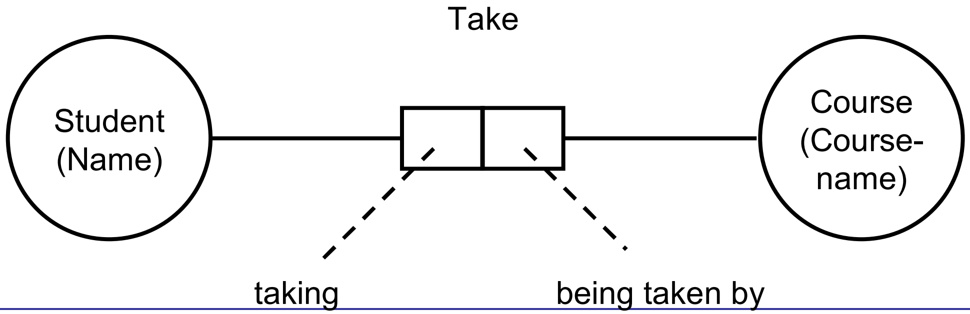
ORM Normalform

- Note that the conversion of a sentence into an index expression is also a method to bring a sample sentence in normalform
- From this normalform the sentence type is derived.
- Form ORM the instances are omitted from the index expression
  - structure is dominant for information analysis
  - instances are important for information need analysis
- The resulting structure is the sentence structure
- This sentence structure may be seen as a grammar rule to generate sentences of this particular sentence type

The resulting ORM schema fragment

- Example:

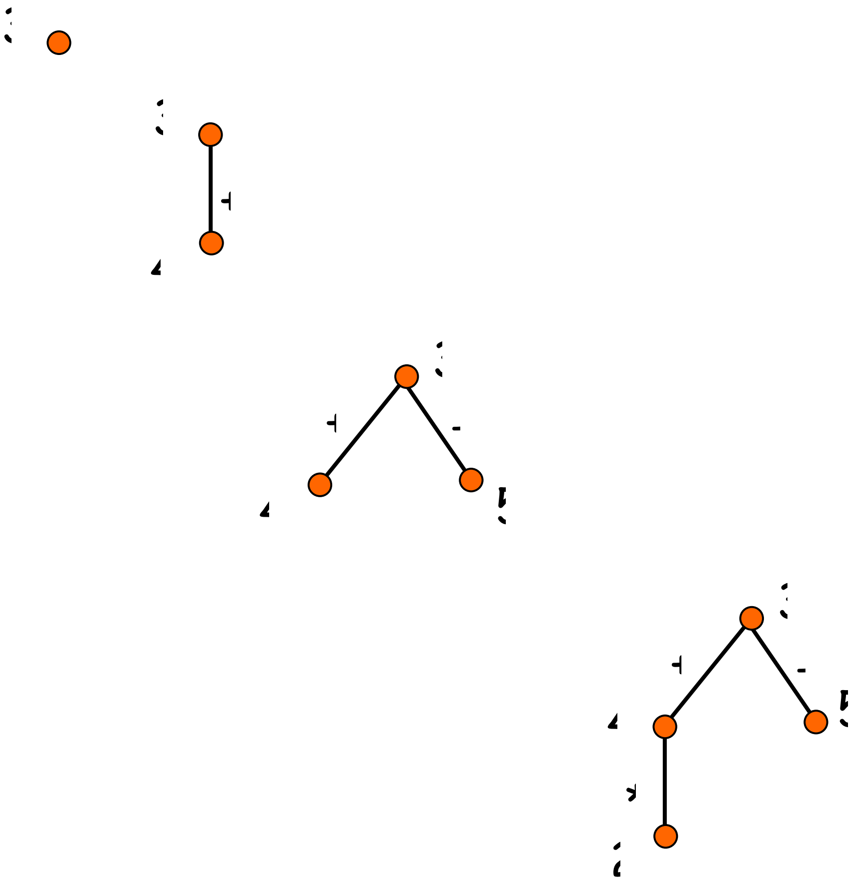




Relation computational expressions

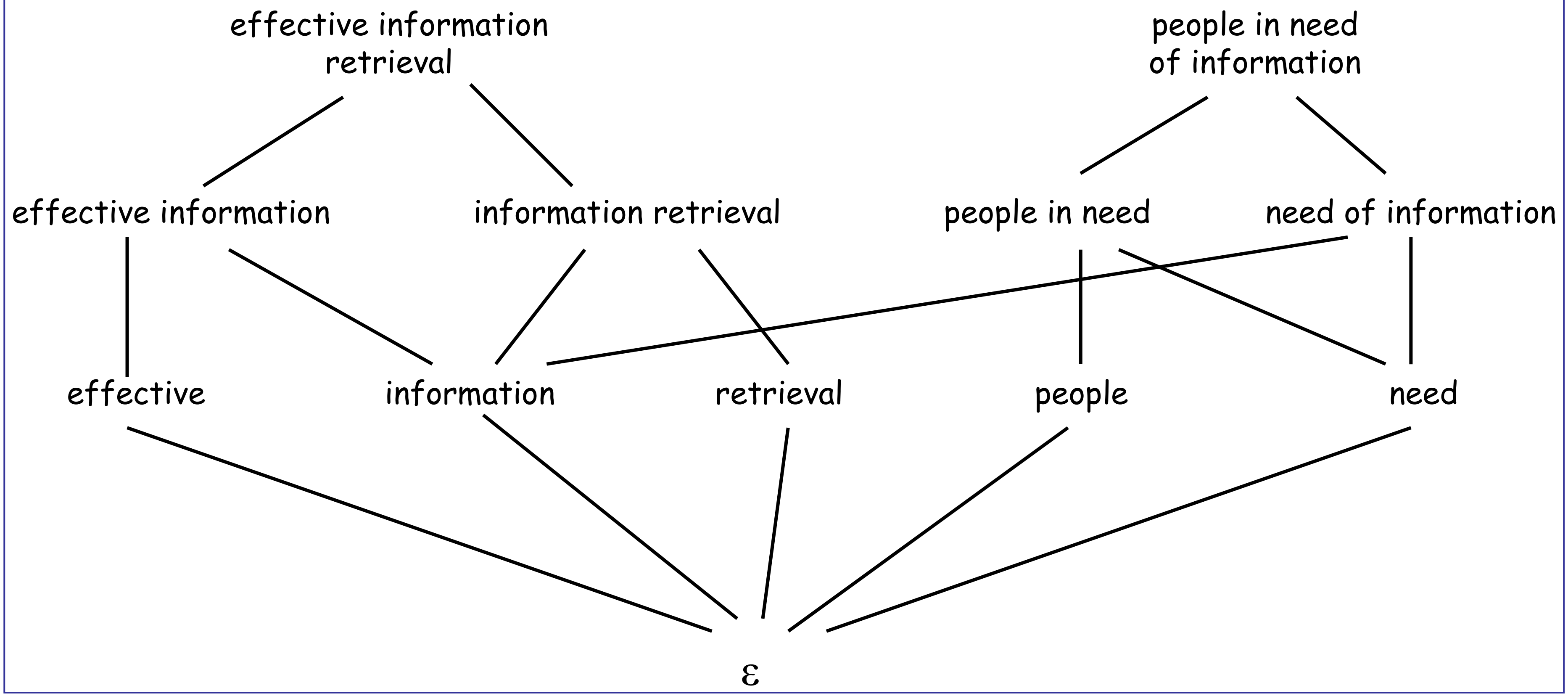
- Expression
- 3
- 3 + 4
- 3 + 4 - 5
- 3 + 4 * 2 - 5

Index expression

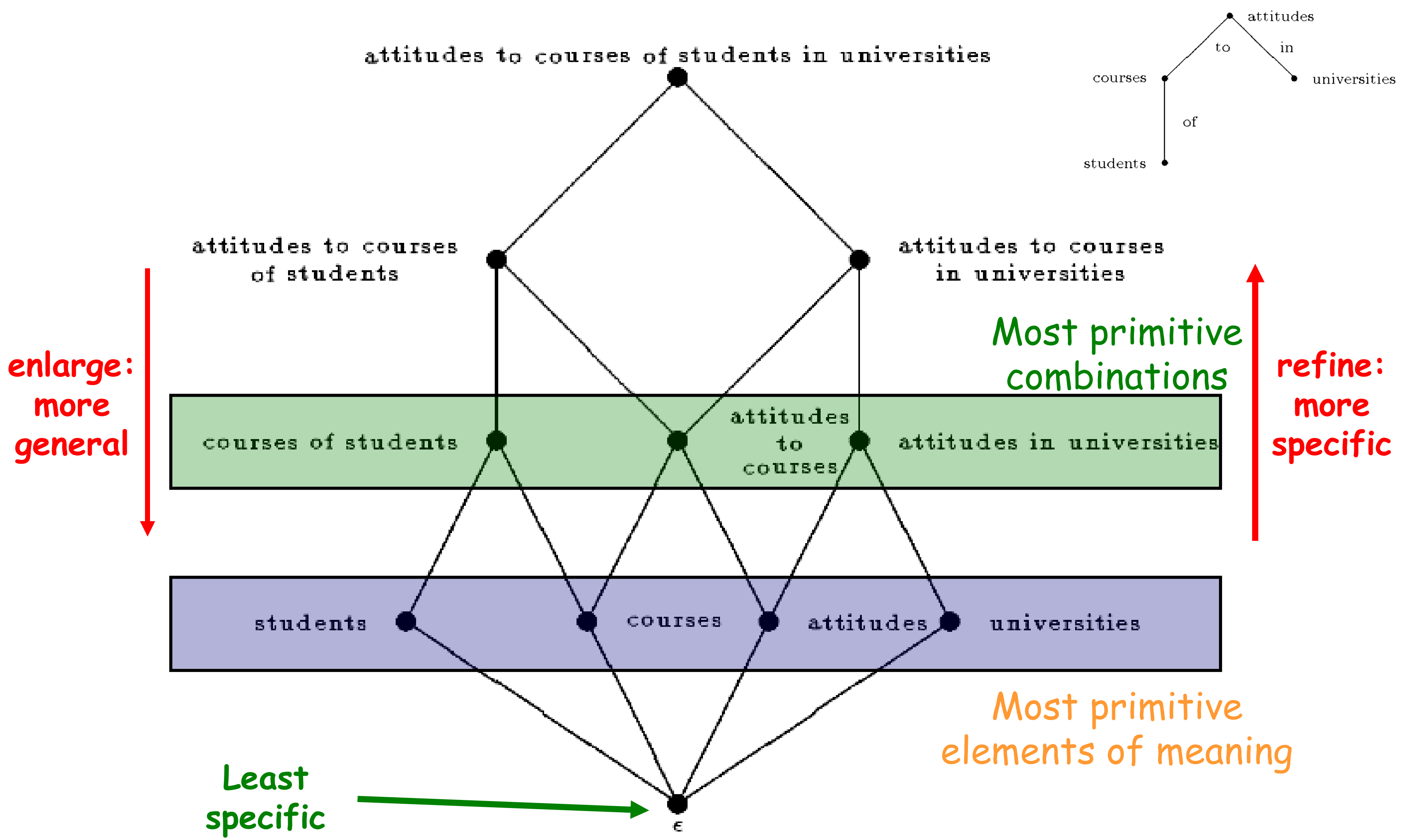


Merging index expressions into lithoid

- Consider:
  - effective information retrieval
  - people in need of information

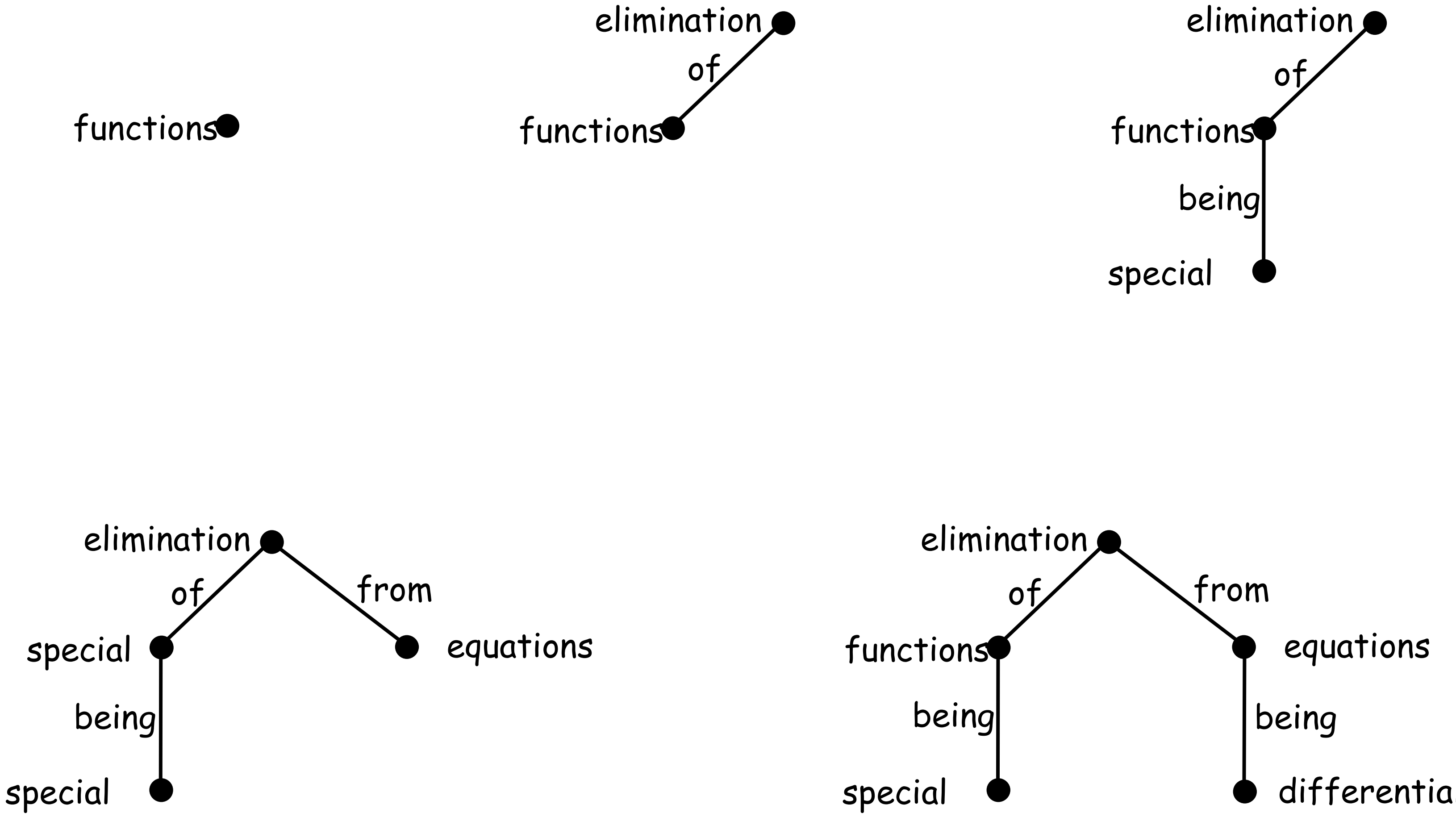


The lithoid



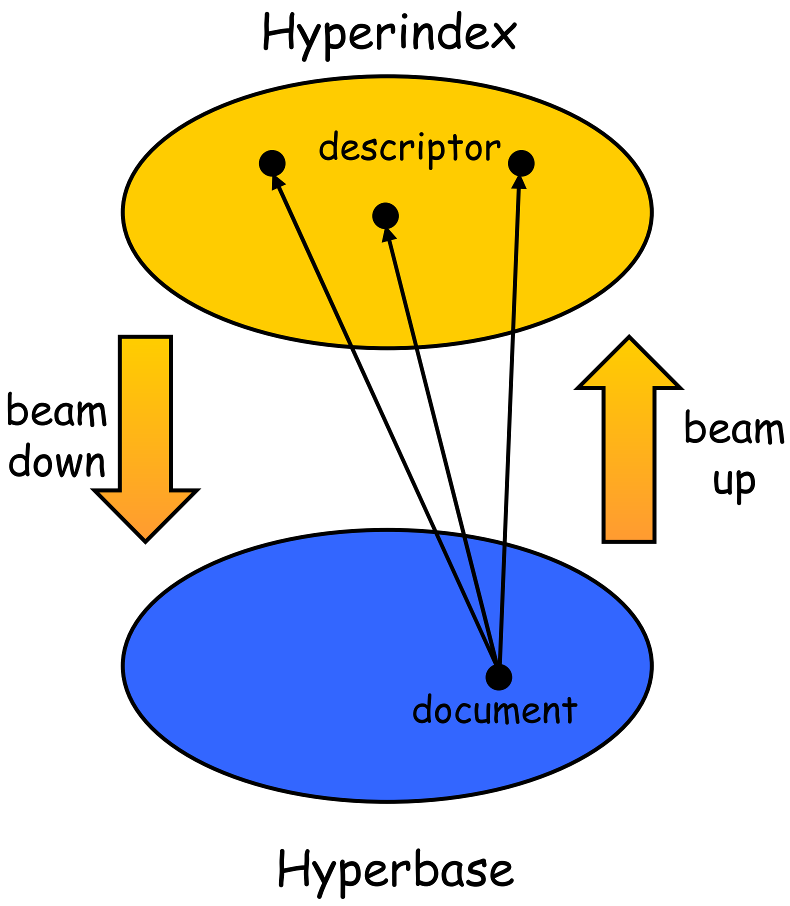
attitudes to courses of students in universities
attitudes to courses of students
attitudes to courses in universities
courses of students
attitudes to courses
attitudes in universities
students
courses
attitudes
universities
ε
attitudes
to
in
courses
universities
of
students
enlarge: more general
refine: more specific
Most primitive combinations
Most primitive elements of meaning
Least specific

A sample navigation



The query dialog

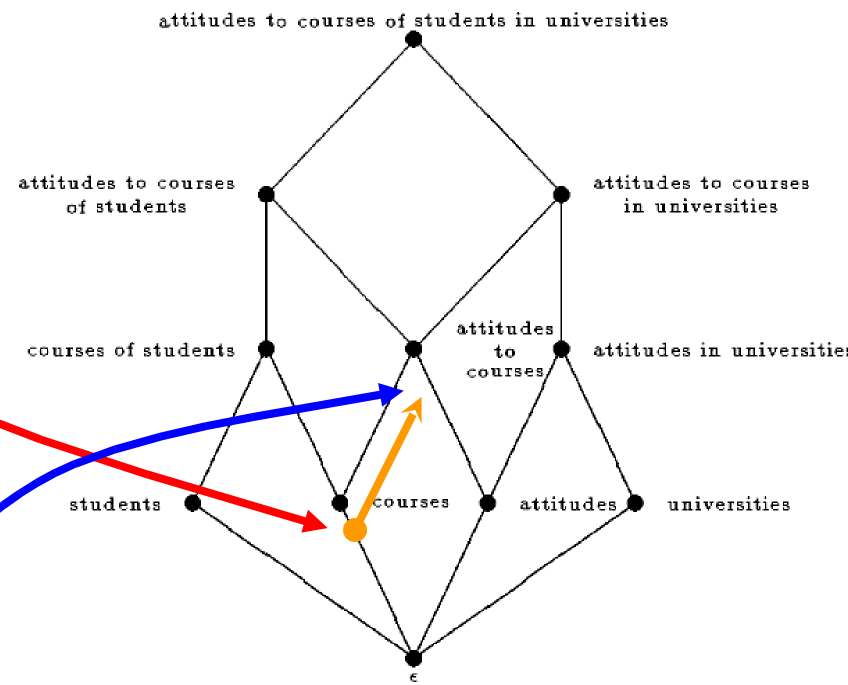
- Query by navigation:
  - First possible aspects are suggested to the searcher
  - By recognizing aspects, the dialog will start to build a query expression, using
    - refinement
    - enlargement
- Query by example:
  - By beaming down the searcher can inspect “relevant” documents, and find relevant examples
  - By beaming up, the searcher can continue query by navigation



The user interface

- The current node in the lithoid is called the focus.
- The system displays the direct environment of the focus

courses
⇑ courses of students
⇑ attitudes to courses
⇓ Start



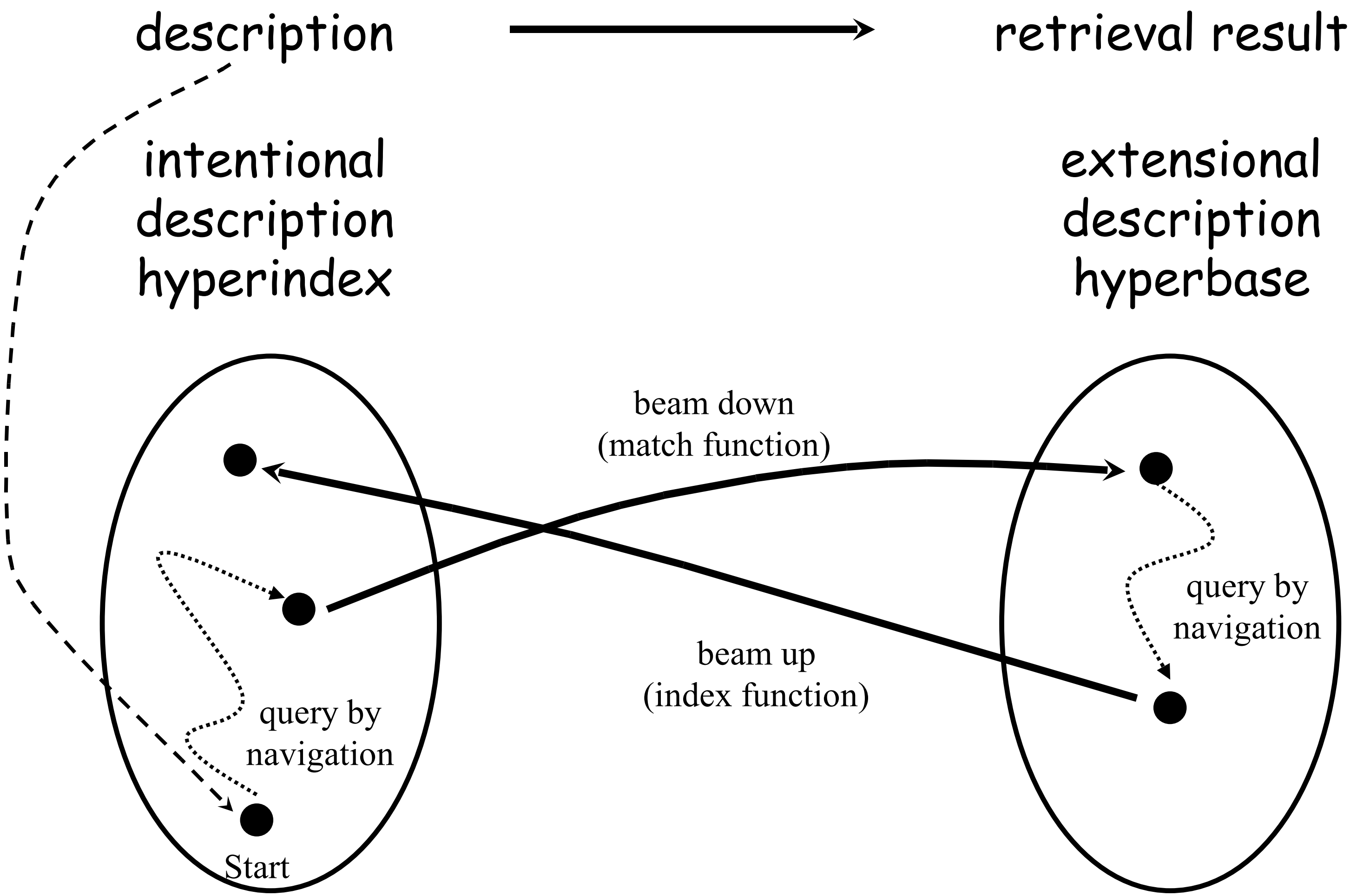
- Selecting attitudes to courses:

attitudes to courses
⇑ attitudes to courses of students
⇑ attitudes to courses in universities
⇓ courses
⇓ attitudes

Motivation

- The navigation starts in the least specific element:
  - at this point the searcher has not yet revealed any detail of the information need
- Assuming a visceral information need, the most primitive elements of meaning are offered for recognition:
  - verbs
  - nouns
  - adjectives
- Upon beaming down, the searcher level of information need will be the conscious need level: the searcher now can judge relevance of documents.
- By iterating a formalized need will result.

The dualistic view



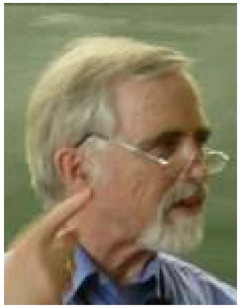
Involving Semantics

Involving (situational) semantics

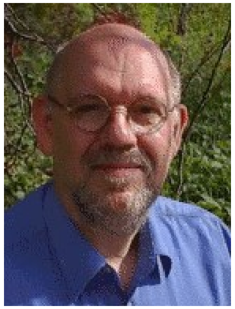
- So far, syntactic structure has driven the construction of the lithoid.
- If a special collection is assumed, then semantic knowledge from this collection may be employed.
- Syntactic steps may be too detailed in terms of these semantics, as there is no real difference in retrieval result.
- It would be helpful to classify the subexpressions using a similarity relation that is an equivalence relation.
- An answer: formal concept analysis

Formal Concept Analysis

- Foundations
  - notion of "concept" in logic(19th Century)
  - Lattice Theory (~1940's)
- Introduced by Rudolf Wille (1979), later advocated by Bernhard Ganter
- A discrete technique for data analysis and knowledge processing
  - more suited to problems in our discipline (until law of large numbers takes over)?



Rudolf Wille



Bernhard Ganter

Example: beverages

- Suppose we have the following:
  - Objects
    - Tea, Coffee, Mineral Water, Wine, Beer, Cola, Champagne
  - Attributes
    - non-alcoholic, hot, alcoholic, caffeine, sparkling
- Objects are characterized by the attributes they possess, for example:
  - Cola: non-alcoholic, caffeine, sparkling

Cross Table

Attributes / Objects	non-alcoholic	hot	alcoholic	caffeine	sparkling
Tea	X	X			
Coffee	X	X		X	
Mineral Water	X				X
Wine			X		
Beer			X		X
Cola	X			X	X
Champagne			X		X

So coffee is a specialization of tea!

Mutual meaning assignment

- By our view (and resulting characterization):
  - An object is a set of attributes
- But also (!):
  - An attribute is a set of objects
- This is called a dualistic view
- Question: how doe these views relate to each other?

Commonality

- Consider for example
  - tea and coffee

Attributes / Objects	non-alcoholic	hot	alcoholic	caffeine	sparkling
Tea	X	X			
Coffee	X	X		X	
Mineral Water	X				X
Wine			X		
Beer			X		X
Cola	X			X	X
Champagne			X		X

- What do they have in common ➔ what attributes do they share?
- Answer: both are nonalcoholic, hot
- Next question: are there more objects with this 'meaning'?
- Answer: no
- Conclusion: the combinations {tea, coffee} and {nonalcoholic, hot} represent the same meaning!

Formal concept

- So: representing the same meaning by a set A of objects and a set B of attributes is described as follows:
  - the common attributes of the objects in A are B:

    ComAttr (A) = B

  - A is the set of all objects characterized by B:

    ComDocs (B) = A

Another example

- Consider for example
  - {wine}

Objects \ Attributes	non-alcoholic	hot	alcoholic	caffeine	sparkling
Tea	X	X			
Coffee	X	X		X	
Mineral Water	X				X
Wine			X		
Beer			X		X
Cola	X			X	X
Champagne			X		X

- Common attributes are
  {alcoholic}
- Next question:
  are there more objects with this 'meaning'?
- Answer: yes!
  - ComDocs ({alcoholic}) = {wine, beer, champagne}
- Note however: ComAttr ({wine, beer, champagne}) = {alcoholic}
- So {wine, beer, champagne} and {alcoholic} form a concept!

Formal Context

- A formal context is:

  a triple $(\mathscr{D}, \mathscr{T}, \sim)$

  where:
  - $\mathscr{D}$ is the set of objects
  - $\mathscr{T}$ is the set of attributes
  - $\sim \subseteq \mathscr{D} \times \mathscr{T}$ is a relation between $\mathscr{D}$ and $\mathscr{T}$.

- To represent an object d is in a relation with attribute t, we write

  $d \sim t$

Formal Context

- A formal context relates objects to attributes.
- For example, the document-term incidence matrix A.
- Notation:
  - d ~ t means: document d contains term t
  - D ~ t means: for each document d from D we have d ~ t
  - d ~ T means: for each term t from T we have: d ~ t
  - D ~ T means: for each document d from D and term t from T: d ~ t

Common attributes and objects

- The common attributes of a set D of objects:

  $$\text{ComAttr}(D) = \{ t \mid D \sim t \}$$

  - example: ComAttr ({Tea, Coffee}) = {non-alcoholic, hot}
- The common documents of a set T of attributes:

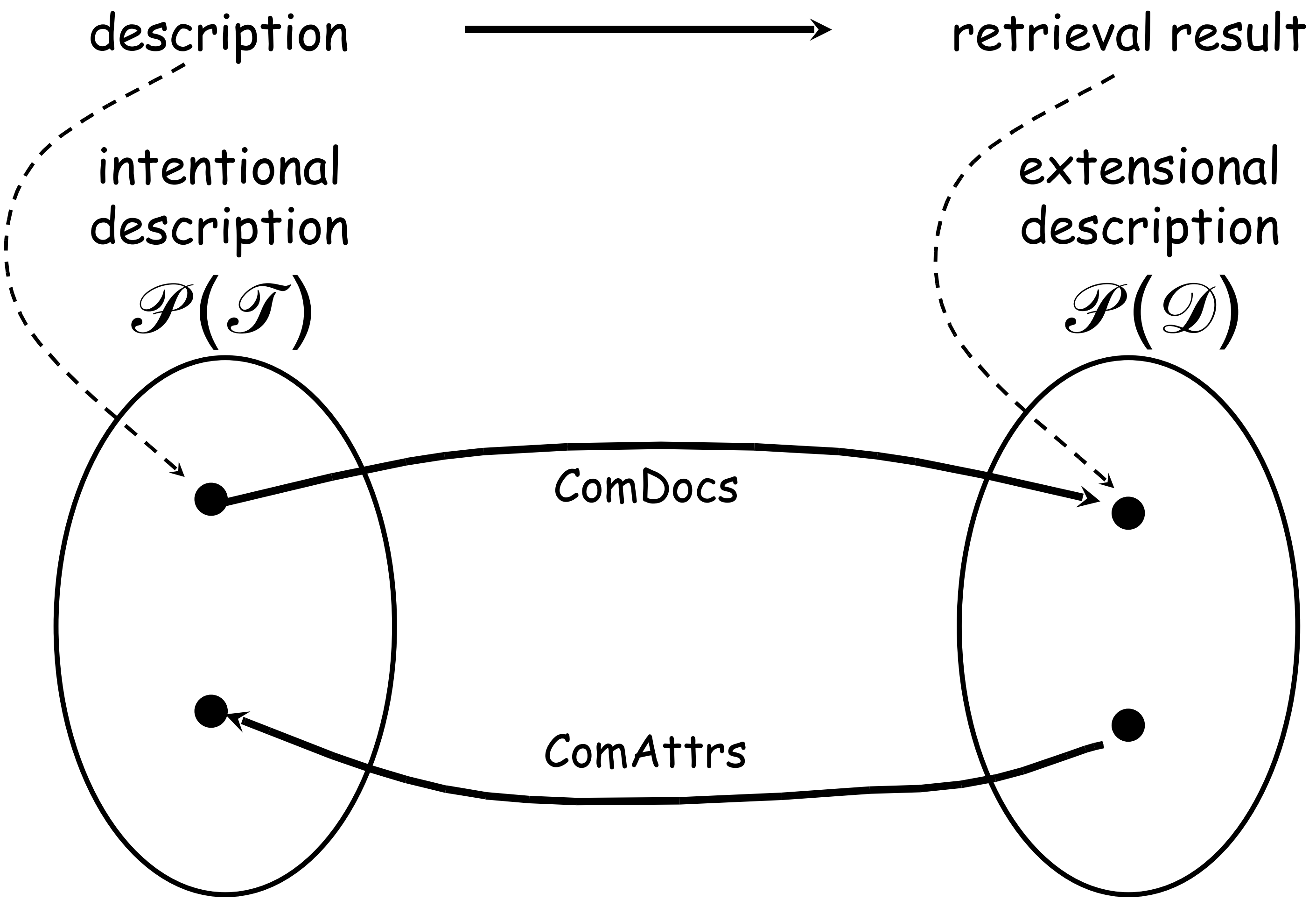
  $$\text{ComDocs}(T) = \{ d \mid d \sim T \}$$

  - example: ComObj ({non-alcoholic, hot}) = {Tea, Coffee}
- ComAttr ($\mathscr{D}$) = set of stopwords
- ComDocs ($\mathscr{T}$) = set of documents containing all terms (!)

Assignment of meaning

- The common attributes of a set D of objects express the meaning of this collection D in terms of attributes they share.
  - example: The meaning of {Tea, Coffee} is: {non-alcoholic, hot}
- The common documents of a set T of attributes express the meaning of T as a set of documents:
  - example: The meaning of {non-alcoholic, hot} is: {Tea, Coffee}

The dualistic view



Formal concepts

- A formal concept is a pair (D,A) with mutual assignment of meaning:

  ComDocs (A) = D
  ComAttr (D) = A

  i.e.: an agreement on meaning

- We call D the extension of the concept, and A its intention:

  ext ((D,A)) = D
  int ((D,A)) = A

Ordering of concepts

- Concepts may be ordered according to their extensionality:

$$c_1 \leq c_2 \equiv ext\ (c_1) \subseteq ext\ (c_2)$$

({Coffee}, {non-alcoholic, hot, caffeine}) $\leq$ ({Tea, Coffee}, {non-alcoholic, hot})

- This implies an intentional ordering:

$$c_1 \leq c_2 \Leftrightarrow int\ (c_1) \supseteq int\ (c_2)$$

- The resulting structure is called the formal lattice.

How to find concepts?

- lemma: ComAttr (ComDocs (ComAttr (D))) = ComAttr (D)

- Conclusion:

  (ComDocs (ComAttr (D)), ComAttr (D)) is a concept.

  If D = {d}, then this concept is called the base concept of d

- Base concept of Tea:
  - ComAttr ({Tea}) = {non-alcoholic, hot}
  - ComDocs ({non-alcoholic, hot}) = {Tea, Coffee}

  → Base concept of Tea is: ({Tea, Coffee}, {non-alcoholic, hot})

How to find concepts?

- lemma: ComDocs (ComAttr (ComDocs (T))) = ComDocs (T)

- Conclusion:

  (ComDocs (T), ComAttr (ComDocs (T))) is a concept.

  If T = {t}, then this concept is called the base concept of t

- Base concept of hot:
  - ComDocs ({hot}) = {Tea, Coffee}
  - ComAttr ({Tea, Coffee}) = {non-alcoholic, hot}

  → Base concept of hot is: ({Tea, Coffee}, {non-alcoholic, hot})

Combining concepts

- We have concepts:
  - ({Cola, Coffee}, {non-alcoholic, caffeine})
  - ({Coffee, Tea}, {non-alcoholic, hot})
- Two ways to combine them:
  - by intersection of extensions:

    ({Coffee}, {non-alcoholic, caffeine, hot}

  - by intersection of intensions

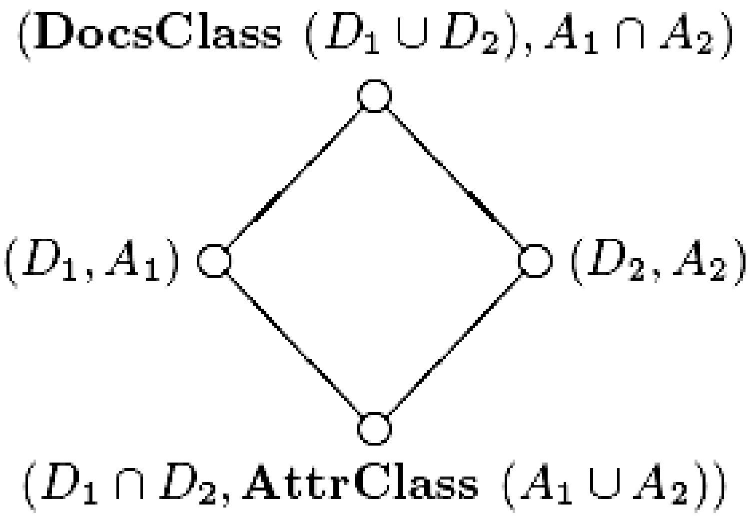
    not as easy

Combining concepts

- lemma:

  Let (D_1, A_1) and (D_2, A_2) concepts,

  Then also:



Binary join and meet

Generation algorithm

- Start with the base concepts.
- Repeat joining concepts already generated, until no new concepts are found

Titles from books reviewed in SIAM

d1: A Course on Integral Equations

d2: Attractors for Semigroups and Evolution Equations

d3: Automatic Differentiation of Algorithms: Theory, Implementation, and Application

d4: Geometrical Aspects of Partial Differential Equations

d5: Ideals, Varieties, and Algorithms - An Introduction to Computational Algebraic Geometry and Commutative Algebra

d6: Introduction to Hamiltonian Dynamical Systems and the N -Body Problem

d7: Knapsack Problems: Algorithms and Computer Implementations

d8: Methods of Solving Singular Systems of Ordinary Differential Equations

d9: Nonlinear Systems

d10: Ordinary Differential Equations

d11: Oscillation Theory for Neutral Differential Equations with Delay

d12: Oscillation Theory of Delay Differential Equations

d13: Pseudodifferential Operators and Nonlinear Partial Differential Equations

d14: Sinc Methods for Quadrature and Differential Equations

d15: Stability of Stochastic Differential Equations with Respect to Semi-Martingales

d16:The Boundary Integral Approach to Static and Dynamic Contact Problems

d17:The Double Mellin-Barnes Type Integrals and Their Application to Convolution Theory

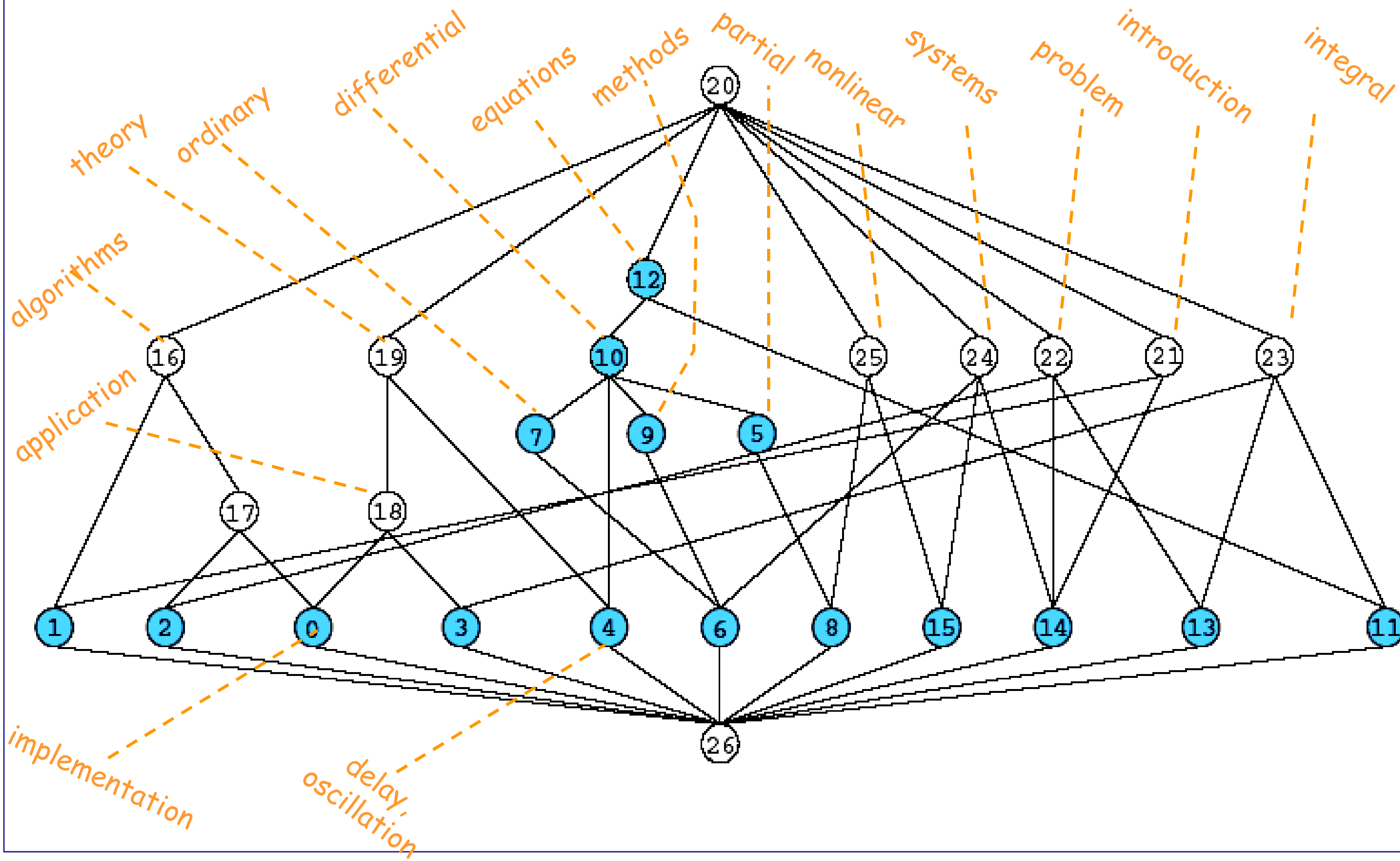
The context

	algorithms	application	delay	differential	equations	implementation	integral	introduction	methods	nonlinear	ordinary	oscillation	partial	problem	systems	theory
d_1					×		×									
d_2					×											
d_3	×	×				×										×
d_4				×	×								×			
d_5	×							×								
d_6								×						×	×	
d_7	×					×								×		
d_8				×	×				×		×				×	
d_9										×					×	
d_{10}				×	×						×					
d_{11}			×	×	×							×				×
d_{12}			×	×	×							×				×
d_{13}				×	×					×			×			
d_{14}				×	×				×							
d_{15}				×	×											
d_{16}							×							×		
d_{17}		×					×									×

The concepts

concept	documents	attributes
c0	d3	algorithms, application, implementation, theory
c1	d5	algorithms, introduction
c2	d7	algorithms, implementation, problem
c3	d17	application, integral, theory
c4	d11; d12	delay, differential, equations, oscillation, theory
c5	d4; d13	differential, equations, partial
c6	d8	differential, equations, methods, ordinary, systems
c7	d8; d10	differential, equations, ordinary
c8	d13	differential, equations, nonlinear, partial
c9	d8; d14	differential, equations, methods
c10	d4; d8; d10; d11; d12; d13; d14; d15	differential, equations
c11	d1	equations, integral
c12	d1; d2; d4; d8; d10; d11; d12; d13; d14; d15	equations
c13	d16	integral, problem
c14	d6	introduction, problem, systems
c15	d9	nonlinear, systems
c16	d3; d5; d7	algorithms
c17	d3; d7	algorithms, implementation
c18	d3; d17	application, theory
c19	d3; d11; d12; d17	theory
c20	all documents	no attributes
c21	d5; d6	introduction
c22	d6; d7; d16	problem
c23	d1; d16; d17	integral
c24	d6; d8; d9	systems
c25	d9; d13	nonlinear
c26	no documents	all attributes

The lattice



theory
ordinary
differential
equations
methods
partial
nonlinear
systems
problem
introduction
integral
algorithms
application
implementation
delay,
oscillation
20
12
16
19
10
25
24
22
21
23
7
9
5
17
18
1
2
0
3
4
6
8
15
14
13
11
26

Pseudo-relevance feedback

- Take the first elements from the initial query result.
- Extract the index expressions.
- Restrict initial context to these index expressions.
- Build the lithoid.
- Navigate top-down, looking for most relevant concept.
- Use its intention to enrich the original query.

End of presentation