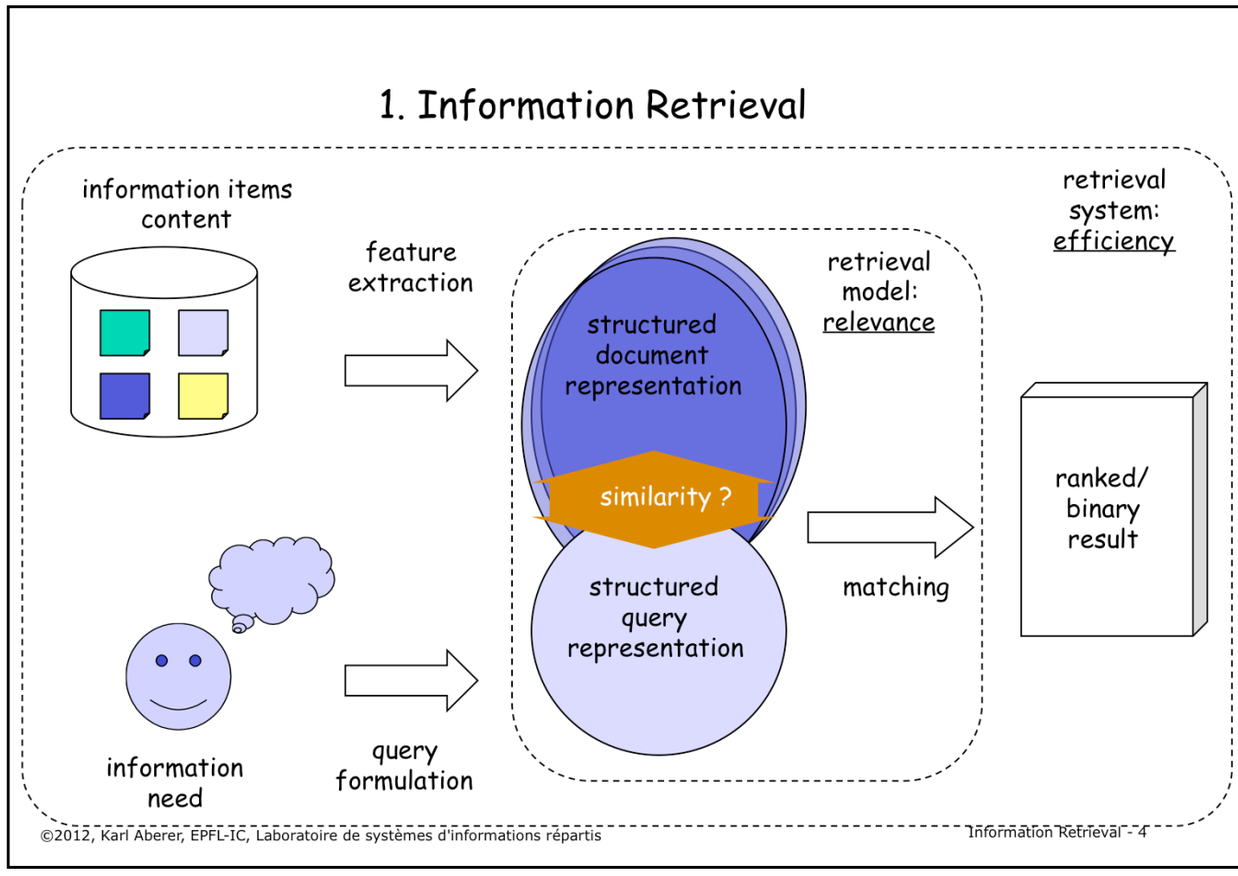
Information Retrieval and Data Mining
Part 1 – Information Retrieval

Today's Question

1. Information Retrieval
2. Text Retrieval Models
3. Latent Semantic Indexing
4. User Relevance Feedback
5. Inverted Files
6. Web Information Retrieval

What do you think ?

- How is a Web search engine working ?



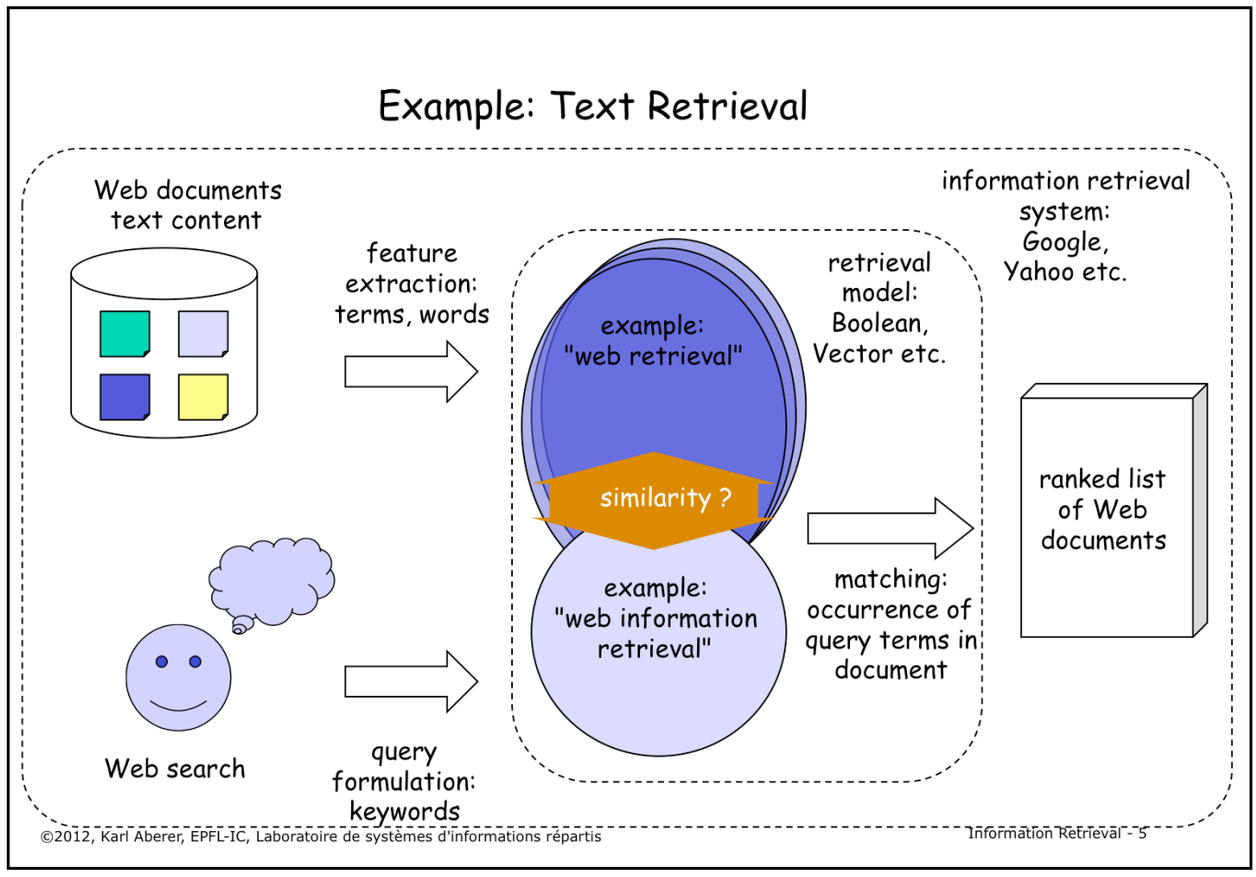
Information retrieval deals with the problem of matching information needs of human users with information provided in information collections. For achieving this an information retrieval system has to deal with the following tasks:

•Generating structured representations of information items: this process is called **feature extraction** and can include simple tasks, such as extracting words from a text as well as complex methods, e.g., for image or video analysis.

•Generating structured representations of information needs: often this task is solved by providing users with a query language and leave the formulation of structured queries to them. This is the case, for example, for simple keyword based query languages, as used in Web search engines. Some information retrieval systems also support the user in the **query formulation,** e.g., through visual interfaces.

•Matching of information needs with information items: this is the algorithmic task of computing similarity of information items and retrieval querie. T the heart of this step is the **information retrieval model**. Similarity measures on the structured representations of queries and documents are used to model **relevance** of information for users. As a result, a selection of relevant information items or a ranked result can be presented to the user.

Since information retrieval systems deal usually with large information collections and/or large user communities, the **efficiency** of an information retrieval system is crucial. This imposes fundamental constraints on the retrieval model. Retrieval models that would capture relevance very well, but are computationally prohibitively expensive, are not suitable for an information retrieval system.



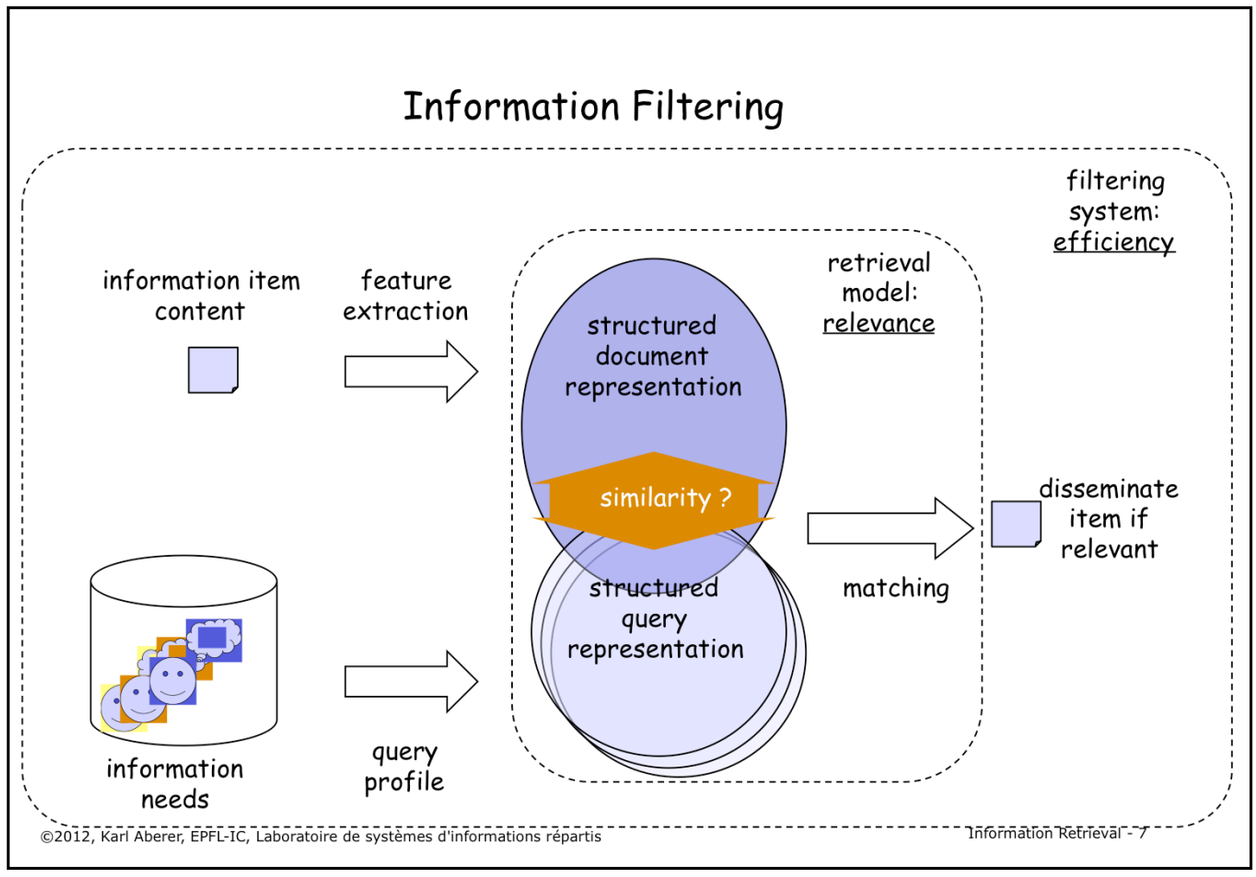
The currently most popular information retrieval systems are Web search engines. To a large degree, they are text retrieval system, since they exploit mainly the textual content of Web documents for retrieval. However, more recently Web search engines also start to exploit link information and even image information. The three tasks of a Web search engine for retrieval are:

1. extracting the textual features, which are the words or terms that occur in the documents. We assume that the web search engine has already collected the documents from the Web using a Web crawler.
2. support the formulation of textual queries. This is usually done by allowing the entry of keywords through Web forms.
3. computing the similarity of documents with the query and producing from that a ranked result. Here Web search engines use standard text retrieval methods, such as Boolean retrieval and vector space retrieval. We will introduce these methods in detail in this lecture later.

Retrieval Model

- Determines
  - the structure of the document representation
  - the structure of the query representation
  - the similarity matching function
- Relevance
  - determined by the similarity matching function
  - should reflect right topic, user needs, authority, recency
  - no objective measure
- Quality of a retrieval model depends on how well it matches user needs !
- Comparison to database querying
  - correct evaluation of a class of query language expressions
  - can be used to implement a retrieval model

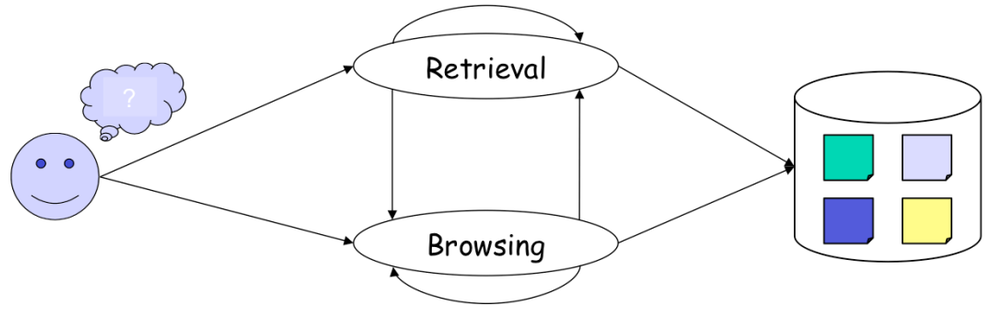
The heart of an information retrieval system is its retrieval model. The model is used to capture the meaning of documents and queries, and determine from that the relevance of documents with respect to queries. Although there exist a number of intuitive notions of what determines relevance one must keep clearly in mind that it is not an objective measure. The quality of a retrieval system can principally only be determined through the degree of satisfaction of its users. This is fundamentally different to database querying, where there exist criteria for correct query answering that can be formally verified, e.g., whether a result set retrieved from a database matches the logical conditions specified in a query.



Similarly as in a XML-based message filtering system, the roles of documents and queries can be swapped also in an information retrieval system, such that one obtains an information filtering system. Information filtering systems can be based on the same retrieval models as classical information retrieval systems for ad-hoc query access.

Information Retrieval and Browsing

- Retrieval
  - Produce a ranked result from a user request
  - Interpretation of the information by the system
- Browsing
  - Let the user navigate in the information set
  - Interpretation of the information by the human



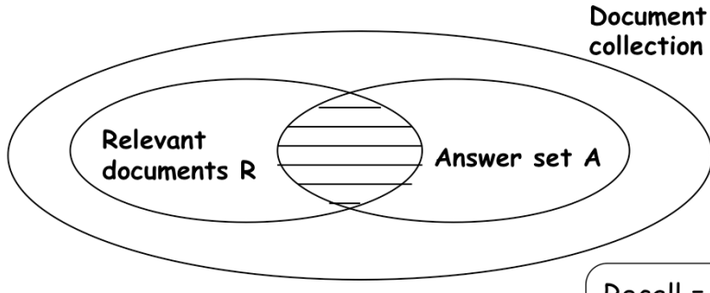
Information retrieval is usually closely connected to the task of browsing. Browsing is the explorative access by users to large document collections. By browsing a user implicitly specifies his/her information needs through selection of documents. This feedback can be used by an information retrieval system in order to improve its query representation and thus the retrieval result. One example of such an approach we will see when introducing relevance feedback. On the other hand, results returned by information retrieval systems are usually large, and therefore browsing is needed by users in order to explore the results. Both activities, retrieval and browsing thus can be combined into an iterative process.

Question

- A retrieval model attempts to model
  1. The process by which a users is accessing information
  2. The importance a user gives to a piece information
  3. The formal correctness of a query formulation by user
  4. All of the above

Evaluating Information Retrieval

- *Recall* is the fraction of relevant documents retrieved from the set of total relevant documents collection-wide
- *Precision* is the fraction of relevant documents retrieved from the total number retrieved (answer set)
- Test collections, where the relevant documents are identified manually are used to determine the quality of an IR system (e.g. TREC)



Recall = |A ∩ R| / |R|

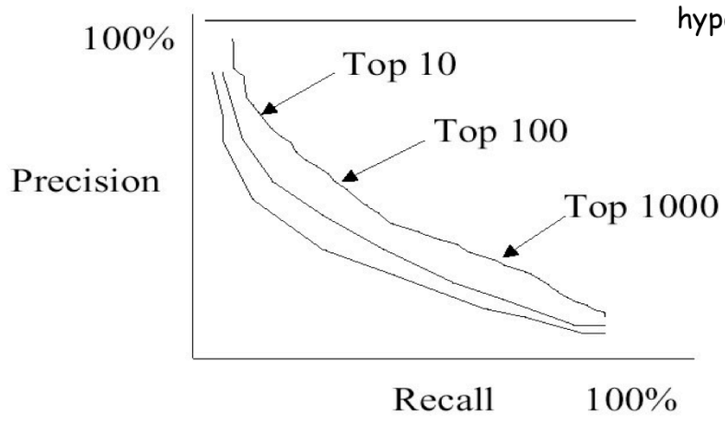
Precision = |A ∩ R| / |A|

Since there exists no objective criterion whether an information retrieval query is correctly answered, other means for evaluating the quality of an information retrieval system are required. The approach is to compare the performance of a specific system to human performance in retrieval. For that purpose test collections of documents, such as TREC (http://trec.nist.gov/), are created and for selected queries human experts select the relevant documents. Note that this approach assumes that humans have an agreed-upon, objective notion of relevance, an assumption that can be easily challenged of course. The results of IR systems are compared to the expected result in two ways:

1. **Recall** measures how large a fraction of the expected results is actually found.
2. **Precision** measures how many of the results returned are actually relevant.

Precision/Recall Tradeoff

- An IR system ranks documents by a similarity coefficient, allowing the user to trade off between precision and recall by choosing the cutoff level
- Precision depends on the number of results retrieved: P@k = precision for the top-k documents



hypothetical ideal IR system

realistic IR systems

One of the two measures of recall and precision can always be optimized. Recall can be optimized by simply returning the whole document collection, whereas precision can be optimized by returning only very few results. Important is the trade-off: the higher the precision for a specific recall, the better the information retrieval system. A hypothetical, optimal information retrieval system would return results with 100% percent precision always. If a system ranks the results according to relevance the user can control the relation between recall and precision by selecting a threshold of how many results he/she inspects.

Questions

- If the top 100 documents contain 50 relevant documents
  1. The precision of the system at 50 is 0.5
  2. The precision of the system at 100 is 0.5
  3. The recall of the system is 0.5
  4. None of the above

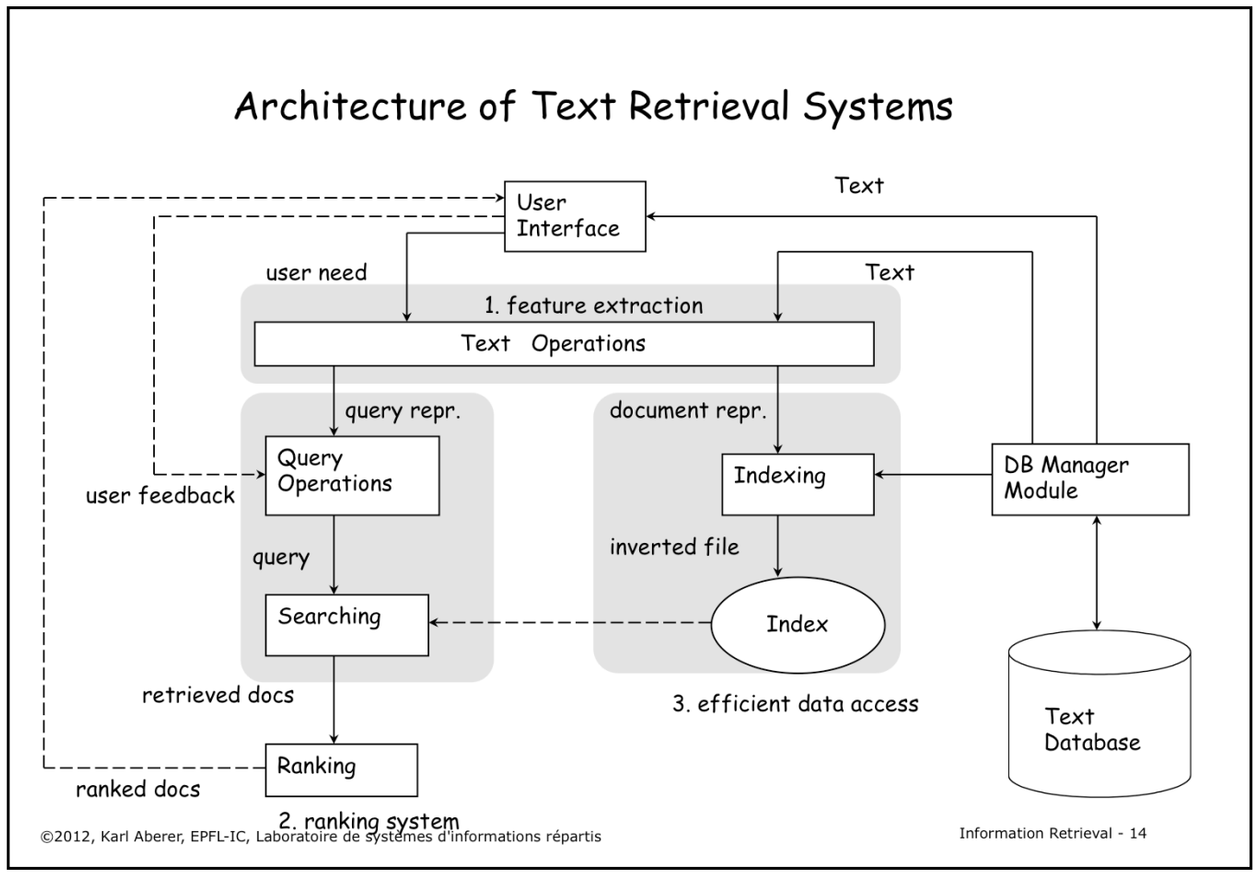
- If retrieval system A has higher precision than system B
  1. The top k documents of A will have higher similarity values than the top k documents of B
  2. The top k documents of A will contain more relevant documents than the top k documents of B
  3. A will recall more documents above a given similarity threshold than B
  4. Relevant documents in A will have higher similarity values than in B

2. Text-based Information Retrieval

- Most of the information needs and content are expressed in natural language
  - Library and document management systems
  - Web (Search Engines)
- Basic approach: use the words that occur in a text as *features* for the interpretation of the content
  - This is called the "full text" retrieval approach
  - Ignore grammar, meaning etc.
  - Simplification that has proven successful
  - Document structure may be taken into account additionally (e.g. PageRank/Google)

Classical information retrieval was concerned over many years primarily with the problem of retrieving information from large bodies of documents with mostly textual content, as they where typically found in library and document management systems. The problems addressed were classification and categorization of documents, systems and languages for retrieval, user interfaces and visualization. The area was perceived as being one of narrow interest for a highly specialized user community, mainly librarians. The advent of the WWW changed this perception completely, as the web is a universal repository of documents with universal access.

Since nowadays most of the information content is still available in textual form, text is an important basis for information retrieval. Natural language text carries a lot of meaning, which still cannot fully be captured computationally. Therefore information retrieval systems are based on strongly simplified models of text, ignoring most of the grammatical structure of text and reducing texts essentially to the terms they contain. This approach is called full text retrieval and is a simplification that has proven to be very successful. Nowadays, this approach is gradually extended by taking into account other features of documents, such as the document or link structure.

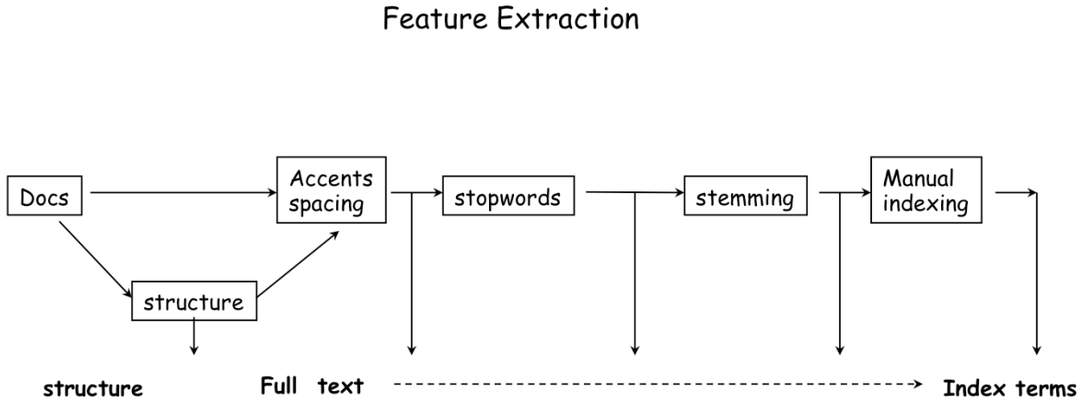


This figure illustrates the basic architecture with the different functional components of a text retrieval system. We can distinguish three main groups of components:

1. the feature extraction component: it performs text processing to turn queries and text documents into a keyword-based representation
2. the ranking system: it implements the retrieval model. In a first step user queries are potentially modified (in particular if user relevance feedback is used), then the documents required for producing the result are retrieved from the database and finally the similarity values are computed according to the retrieval model in order to compute the ranked result.
3. the data access system: it supports the ranking system by efficiently retrieving documents containing specific keywords from large document collections. The standard technique to implement this component is called **inverted files**.

In addition we recognize two components to interface the system to the user on the one hand, and to the data collection on the other hand.

Pre-Processing Text for Text Retrieval



In full text retrieval each document is represented by a set of representative keywords or index terms. An index term is a document word useful for capturing the document's main topics. Often, index terms are only nouns, because nouns carry meaning by themselves, whereas verbs express relationships between words. These relationships are more difficult to extract.

When using words as text features normally a stepwise processing approach is taken: in a first step, the document structure, e.g., from XML, is extracted and if required stored for further processing. The remaining text is stripped of special characters, producing the full text of the document. Then very frequent words which are not useful for retrieval, so-called "stopwords", are eliminated (e.g. "a", "and" etc.). As the same word can occur in natural language in different forms, usually stemming is used: Stemming eliminates grammatical variations of the same word by reducing it to a word root, e.g., the words connecting, connection, connections would be reduced to the same "stem" connect. This step can be followed by a manual intervention, where humans can select or add index terms based on their understanding of the semantics of the document. The result of the process is a set of index terms which represents the document.

Text Retrieval - Basic Concepts and Notations

Document d: expresses ideas about some topic in a natural language
Query q: expresses an information need for documents pertaining to some topic
Index term: a semantic unit, a word, short phrase, or potentially root of a word

Database DB: collection of n documents $d_j \in DB$, $j=1,\ldots,n$
Vocabulary T: collection of m index terms $k_i \in T$, $i=1,\ldots,m$

A document is represented by a set of index terms k_i

The importance of an index term k_i for the meaning of a document d_j is represented by a *weight* $w_{ij} \in [0,1]$; we write $d_j = (w_{1j}, \ldots, w_{mj})$

The IR system assigns a *similarity coefficient* $sim(q, d_j)$ as an estimate for the relevance of a document $d_j \in DB$ for a query q.

We introduce the precise terminology we will use in the following for text retrieval systems. Note that the way of how specific weights are assigned to an index term with respect to a document and of how similarity coefficients are computed are part of the definition of the text retrieval model.

Example: Documents

B1 A Course on Integral Equations
B2 Attractors for Semigroups and Evolution Equations
B3 Automatic Differentiation of Algorithms: Theory, Implementation, and Application
B4 Geometrical Aspects of Partial Differential Equations
B5 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 Knapsack Problems: Algorithms and Computer Implementations
B8 Methods of Solving Singular Systems of Ordinary Differential Equations
B9 Nonlinear Systems
B10 Ordinary Differential Equations
B11 Oscillation Theory for Neutral Differential Equations with Delay
B12 Oscillation Theory of Delay Differential Equations
B13 Pseudodifferential Operators and Nonlinear Partial Differential Equations
B14 Sinc Methods for Quadrature and Differential Equations
B15 Stability of Stochastic Differential Equations with Respect to Semi-Martingales
B16 The Boundary Integral Approach to Static and Dynamic Contact Problems
B17 The Double Mellin-Barnes Type Integrals and Their Applications to Convolution Theory

This is an example of a (simple) document collection that we will use in the following as running example.

Term-Document Matrix

Vocabulary (contains only terms that occur multiple times, no stop words)

Terms	Documents																
	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17
algorithms	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0
application	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
delay	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
differential	0	0	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0
equations	1	1	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0
implementation	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
integral	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
introduction	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
methods	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
nonlinear	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
ordinary	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
oscillation	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
partial	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
problem	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
systems	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0
theory	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1

all weights are set to 1 (equal importance)

In text retrieval we represent the relationship between the index terms and the documents in a term-document matrix. In this example only a selected vocabulary is used for retrieval, consisting of all index terms that occur in more than one document and only weights of 1 are assigned, indicating that the term occurs in the document.

Questions

- Full-text retrieval means that
  1. The document text is grammatically deeply analyzed for indexing
  2. The complete vocabulary of a language is used to extract index terms
  3. All words of a text are considered as potential index terms
  4. All grammatical variations of a word are indexed

- The term-document matrix indicates
  1. How many relevant terms a document contains
  2. How relevant a term is for a given document
  3. How often a relevant term occurs in a document collection
  4. Which relevant terms are occuring in a document collection

Boolean Retrieval

- Users specify which terms should be present in the documents
  - Simple, based on set-theory, precise meaning
  - Frequently used in old library systems
- Example query
  - "application" AND "theory"
  - answer: B3, B17

Retrieval Language

expr ::= term | (expr) | NOT expr | expr AND expr | expr OR expr

Weights for index terms appearing in documents

$w_{ij} = 1$ if $k_i \in d_j$ and 0 otherwise

Early information retrieval systems (as well as many systems today on the Web, such as amazon) use the Boolean retrieval model. This model is actually more similar to database querying, as requests are specified as first order (Boolean) expressions. Term weights are set to 1 when a term occurs in a document, just as in the term-document matrix on the previous slide.

"Similarity" Computation in Boolean Retrieval

- Step 1: Determine the disjunctive normal form of the query *q*
  - A disjunction of conjunctions
  - Using distributivity and Morgans laws, e.g. *NOT (s AND t) ≡ NOT s OR NOT t*
  - Thus $q = ct_1 \text{ OR } \ldots \text{ OR } ct_l$ where $ct = \underline{t}_1 \text{ AND } \ldots \text{ AND } \underline{t}_k$ and $\underline{t} \in \{t, \text{NOT } t\}$

- Step 2: For each conjunctive term *ct* create its weight vector *vec(ct)*
  - $vec(ct)=(w_1,\ldots,w_m)$:
    $w_i = 1$ if k_i occurs in *ct*
    $w_i = -1$ if *NOT* k_i occurs in *ct*
    $w_i = 0$ otherwise

- Step 3: If one weight vector of a conjunctive term *ct* in *q* matches the document weight vector $d_j = (w_{1j}, \ldots, w_{mj})$ of a document d_j , then the document d_j is relevant, i.e., $sim(d_j, q) = 1$
  - *vec(ct)* matches d_j if:
    $w_i = 1 \rightarrow w_{ij} = 1$
    $w_i = -1 \rightarrow w_{ij} = 0$

Computing the similarity of a document with a query reduces in Boolean retrieval to the problem of checking whether the term occurrences in the document satisfy the Boolean condition specified by the query. In order to do this in a systematic manner, a Boolean query is first normalized into disjunctive normal form. Using this equivalent representation, checking whether a document matches the query reduces to the problem of checking whether the document vector, i.e., the column of the term-document matrix corresponding to the document, matches one of the conjunctive terms of the query. A match is established if the document vector contains all the terms of the query vector in the correct form, i.e., if the term occurs positively in the query the term has to occur in the document, if the term occurs in the negated form in the query the term must not occur, and if the term does not occur in the query it may or may not occur in the document.

Example

- Index terms *{application, algorithm, theory}*
- Query *"application" AND ("algorithm" OR NOT "theory")*
- Disjunctive normal form of query

  ("application" AND "algorithm" AND "theory") OR
  ("application" AND "algorithm" AND NOT "theory") OR
  ("application" AND NOT "algorithm" AND NOT "theory")

- Query weight vectors *q={(1,1,1), (1,1,-1), (1,-1,-1)}*
- Documents

d_1={algorithm, theory, application}	(1,1,1)
d_2={algorithm, theory}	(0,1,1)
d_3={application, algorithm}	(1,1,0)

- Result *$sim(d_1, q) = sim(d_3, q) = 1$, $sim(d_2, q) = 0$*

This example illustrates a complete Boolean retrieval process for our sample document collection.

Question

- Let the query be represented by the following vectors: (1, 0, -1) (0, -1, 1); the document by the vector (1, 0, 1)
  1. Matches the query because it matches the first query vector
  2. Matches the query because it matches the second query vector
  3. Does not match the query because it does not match the first query vector
  4. Does not match the query because it does not match the second query vector

Vector Space Retrieval

- Limitations of Boolean Retrieval
  - No ranking: problems with handling large result sets
  - Queries are difficult to formulate
  - No tolerance for errors
- Key Idea of Vector Space Retrieval
  - represent both the document and the query by a weight vector in the m-dimensional keyword space assigning non-binary weights
  - determine their distance in the m-dimensional keyword space
- Properties
  - Ranking of documents according to similarity value
  - Documents can be retrieved even if they don't contain some query keyword
- Todays standard text retrieval technique
  - Web Search Engines
  - The vector model is usually as good as the known ranking alternatives
  - It is simple and fast to compute

The main limitation of the Boolean retrieval model is its incapability to rank the result and to match documents that do not contain all the keywords of the query. In addition, more complex requests become very difficult to formulate. The vector space retrieval model addresses these issues by supporting non-binary weights, i.e., real numbers in [0,1], both for documents and queries, and producing continuous similarity measures in [0,1]. The similarity measure is derived from the geometrical relationship of vectors in the m-dimensional space of document/query vectors. The vector space retrieval model is the standard retrieval technique used both on the Web and for classical text retrieval.

Similarity Computation in Vector Space Retrieval

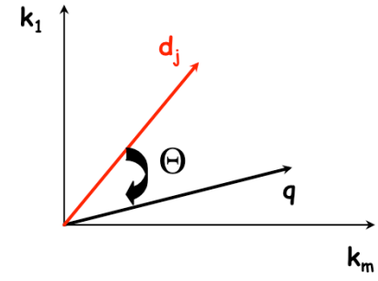
$$\vec{d}_j = \left(w_{1j}, w_{2j}, \ldots, w_{mj}\right), w_{ij} > 0 \quad if\ k_i \in d_j$$

$$\vec{q} = \left(w_{1q}, w_{2q}, \ldots, w_{mq}\right), w_{iq} \geq 0$$

$$sim(\vec{q}, \vec{d}_j) = \cos(\theta) = \frac{\vec{d}_j \bullet \vec{q}}{\left|\vec{d}_j\right| \left|\vec{q}\right|} = \frac{\sum_{i=1}^{m} w_{ij} w_{iq}}{\left|\vec{d}_j\right| \left|\vec{q}\right|}$$

$$|v| = \sqrt{\sum_{i=1}^{m} v_i^2}$$

Since $w_{ij} > 0$ and $w_{iq} \geq 0$, $0 \leq sim(q, d_j) \leq 1$



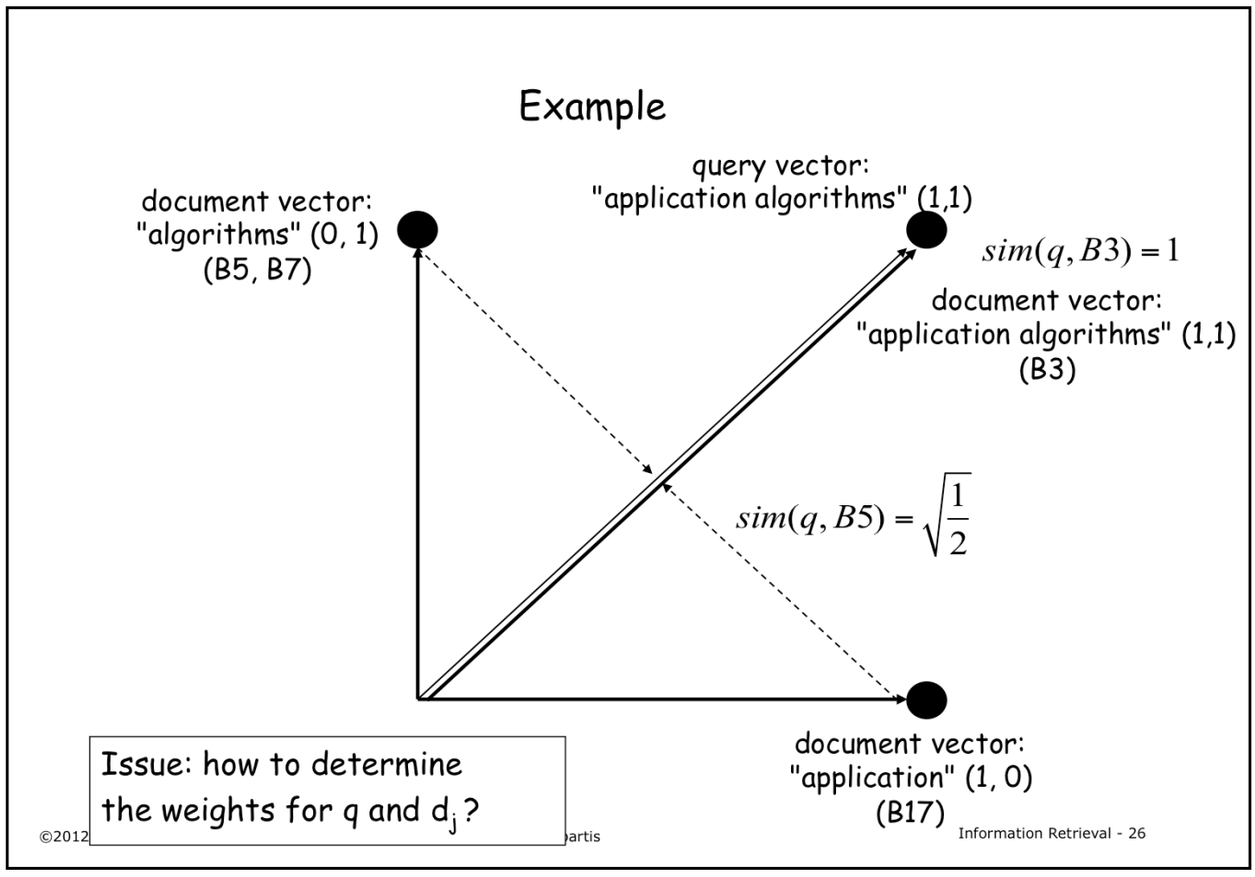
The distance measure for vectors has to satisfy the following properties:

•If two vectors coincide completely their similarity should be maximal, i.e., equal to 1.

•If two vectors have no keywords in common, i.e., if wherever the query vector has positive weights the document vector has weight 0, and vice versa – or in other words if the vectors are orthogonal – the similarity should be minimal, i.e., equal to 0.

•in all other cases the similarity should be between 0 and 1.

The scalar product (which is equivalent to the cosine of the angle of two vectors) has exactly these properties and is therefore (normally) used as similarity measure for vector space retrieval.



We apply a the same weighting scheme for the document and query vectors as in the previous example used to illustrate Boolean retrieval, and show the results vector space retrieval produces. We observe that also documents containing only one of the two keywords occurring in the query, would show up in the result, although with lower similarity value.

Since in vector space retrieval no longer exclusively binary weights are used, a central question is of how to determine weights that more precisely determine the importance of a term for the document. Obviously not all terms carry the same amount of information on the meaning of a document.This was for example one of the reasons to eliminate stop words, as they normally carry no meaning at all.

Weights of Document Vectors: Term Frequency

- Documents are similar if they contain the same keywords (frequently)
  - Therefore use the frequency *freq(i,j)* of the keyword k_i in the document d_j to determine the weight

(Normalized) term frequency of term k_i in Document d_j

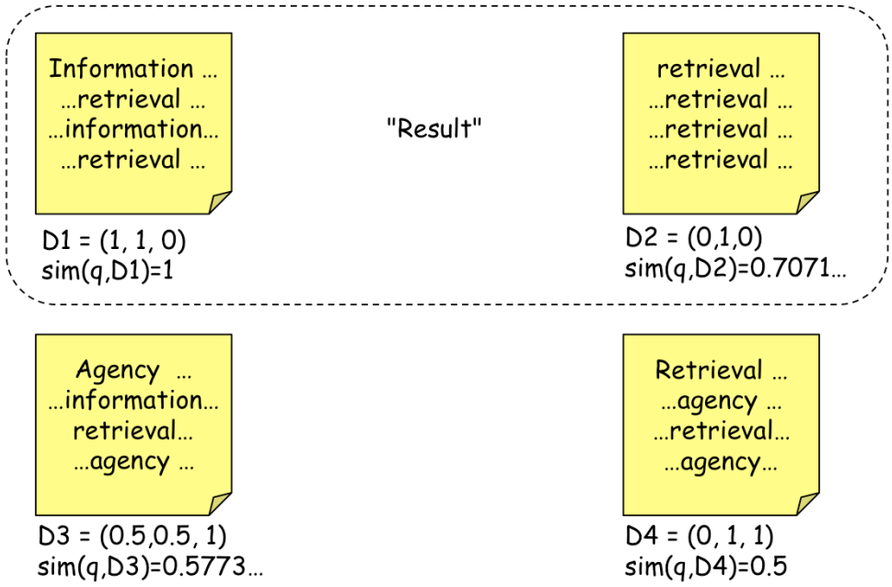
$$tf(i,j) = \frac{freq(i,j)}{\max_{k \in T} freq(k,j)}$$

An obvious difference that can be made among terms is with respect to their frequency of occurrence in a document. Thus a weighting scheme for documents can be defined by considering the (relative) frequency of terms within a document. The term frequency is normalized with respect to the maximal frequency of all terms occurring within the document.

Example

$$sim(\vec{q}, \vec{d}_j) = \frac{\sum_{i=1}^{m} w_{ij} w_{iq}}{\left|\vec{d}_j\right| \left|\vec{q}\right|}$$

Vocabulary T = {information, retrieval, agency}
Query q = (information, retrieval) = (1,1,0)



This example illustrates the use of term frequency. Assume we form the query vector by simply setting the weight to 1 if the keyword appears in the query. Then we would obtain D1 and D2 as result. Actually, this result appears to be non-intuitive, since we would expect that D3 is much more similar to q than D2. What has gone wrong?

The problem is that the term "retrieval", since it occurs very frequently in D2, leads to a high similarity value for D2. On the other hand the term retrieval has very little power to disambiguate meaning in this document collection, since every document contains this term. From an information-theoretic perspective one can state, that the term "retrieval" does not reduce the uncertainty about the result at all.

Question

- Which is right? The term frequency is normalized
  1. By the maximal frequency of a term in the document
  2. By the maximal frequency of a term in the document collection
  3. By the maximal frequency of a term in the vocabulary
  4. By the maximal term frequency of any document in the collection

Inverse Document Frequency

- We have not only to consider how frequent a term occurs within a document (measure for similarity), but also how frequent a term is in the document collection of size n (measure for distinctiveness)

Inverse document frequency of term k_i

$$idf(i) = \log(\frac{n}{n_i}) \in [0, \log(n)]$$

n_i number of documents in which term k_i occurs

- Inverse document frequency can be interpreted as the amount of information associated with the term k_i

Term weight $w_{ij} = tf(i,j)\ idf(i)$

Thus we have to take into account not only the frequency of a term within a document, when determining the importance of the term for characterizing the document, but also the discriminative power of the term with respect to the document collection as a whole. For that purpose, the inverse document frequency is computed and included into the term weight.

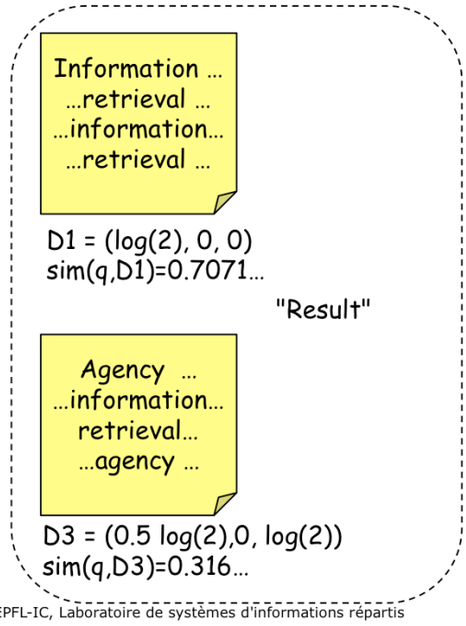
We can see now from this weighting scheme that eliminating stop words is actually an optimization of computing similarity measures in vector space retrieval. Since stop words normally occur in every document of a collection, their term weights will normally be 0 and thus the terms do not play a role in retrieval. Thus it is of advantage to exclude them already from the retrieval process at the very beginning.

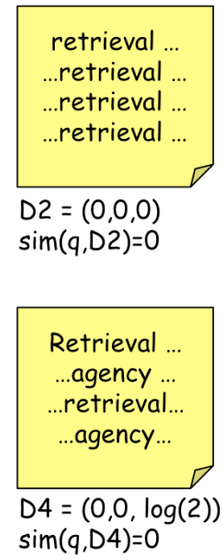
$$idf(i) = \log(\frac{n}{n_i}) \in [0, \log(n)]$$

Example

Vocabulary T = {information, retrieval, agency}
Query q = (information, retrieval) = (1,1,0)

idf(information)=idf(agency)=log(2)
idf(retrieval)=log(1)=0





We have now: n=4, $n_{information}$=2, $n_{retrieval}$=4, n_{agency}=2

The result corresponds much better to the "expectation" when using the inverse document frequencies.

Query Weights

- The same considerations as for document term weights apply also to query term weights

Query weight for query q

$$w_{iq} = \frac{freq(i,q)}{\max_{k \in T} freq(k,q)}) \log(\frac{n}{n_i})$$

- Example: Query q = (information, retrieval)
  - Query vector: (log(2), 0, 0)
  - Scores: sim(q, D1)= 0.569…
    sim(q, D2)=0
    sim(q, D3)=0.254
    sim(q, D4)=0

Finally, we have to look at the question of how to determine the weights for the query vector. One can apply the same principles as for determining the document vector, as is shown. In practice there exist a number of variations of this approach.

Example

- Query q = "application theory"
- Boolean retrieval result
  - application AND theory: B3, B17
  - application OR theory: B3, B11, B12, B17
- Vector retrieval result
  - Query vector (0, 2.14…, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.447…)
  - Ranked Result:

B17	0.770078…
B3	0.684042…
B12	0.232951…
B11	0.232951…

This examples provides an illustration of the differences of Boolean and vector space retrieval.

Discussion of Vector Retrieval Model

- The vector model with tf-idf weights is a good ranking strategy for general collections
  - many alternative weighting schemes exist, but are not fundamentally different
- Advantages
  - term-weighting improves quality of the answer set
  - partial matching allows retrieval of docs that approximate the query conditions
  - cosine ranking formula sorts documents according to degree of similarity to the query
- Disadvantages
  - assumes independence of index terms
  - not clear that this is a disadvantage

We summarize here the main advantages of the vector space retrieval model. It has proven to be a very successful model for general text collections, i.e., if there exists no additional (context) information on the documents that could be exploited, e.g., from a specific application domain. Providing a ranked result improves the usability of the approach, as users can more easily distinguish more relevant documents from less relevant documents. The model inherently assumes that there exist no mutual dependencies in the occurrences of the terms, i.e., that certain terms appear together more frequently than others. Studies have however shown that taking such co-occurrence probabilities additionally into account, can actually HURT the performance of the retrieval system. The reason is that co-occurrence probabilities are often related to specific application domains and thus do not easily transfer to general-purpose retrieval.

Questions

- The inverse document frequency of a term can increase
  1. By adding the term to a document that contains the term
  2. By adding a document to a document collection that does not contain the term
  3. By removing a document from the document collection that does not contain the term
  4. By adding a document to a document collection that contains the term

Summary

- What is the difference between data search and information retrieval ?
- What are the main processing steps in information retrieval ?
- How do browsing and filtering relate to information retrieval ?
- How is an information retrieval system evaluated ?
- How are the weights of document vectors and query vectors computed in Booelan retrieval ?
- How is the similarity coefficient computed in Boolean retrieval ?
- What is the basic abstraction the vector model uses to determine similarity of documents ?
- What are document frequency and inverse document frequency ?
- Why is inverse document frequency used in vector retrieval ?
- How are the weights of document vectors and query vectors computed in vector retrieval ?
- How is the similarity coefficient computed in vector retrieval ?
- Which documents receive similarity value of zero in vector retrieval ?

References

- Course material based on
  - Ricardo Baeza-Yates, Berthier Ribeiro-Neto, Modern Information Retrieval (ACM Press Series), Addison Wesley, 1999.
  - Michael W. Berry, Susan T. Dumais and Gavin W. O'Brien, Using Linear Algebra for Intelligent Information Retrieval, SIAM Review , Vol. 37, No. 4 (Dec., 1995), pp. 573-595