1.4 EMBEDDING TECHNIQUES

1.4.1 Latent Semantic Indexing

Vector space retrieval is vague and noisy

- Based on index terms
- Unrelated documents might be included in the answer set
  - apple (company) vs. apple (fruit)
- Relevant documents that do not contain at least one index term are not retrieved
  - car vs. automobile

Observation

- The user information need is more related to concepts and ideas than to index terms

Despite its success and widespread use, the vector space retrieval model suffers from some problems. If the user misses relevant terms in the query, documents that might contain semantically related terms, i.e. terms with the same meaning, will be missed. Also, in the case of terms with multiple meanings irrelevant documents can be returned.

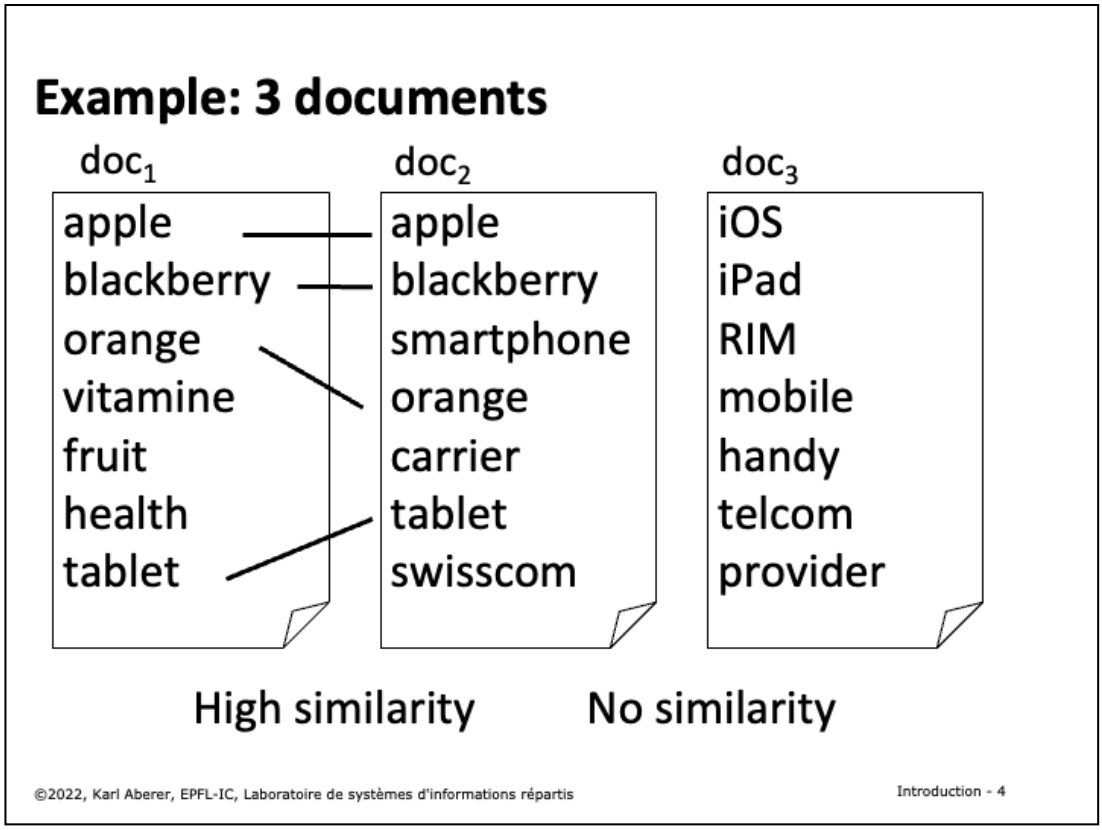
The important insight is that terms may indicate a concept a user is interested in, but there does not necessarily exist a one to one correspondence between terms and concepts. As a consequence, retrieval results may contain irrelevant documents, and relevant documents may be missed.

The Problem

Vector Space Retrieval handles poorly the following two situations

1. *Synonymy*: different terms refer to the same concept, e.g. car and automobile
   - Result: poor recall
2. *Homonymy*: the same term may have different meanings, e.g. apple, model, bank
   - Result: poor precision

These problems are related to the fact that the same concepts can be expressed through many different terms (synonyms) and that the same term may have multiple meanings (homonyms). Studies show that different users use the same keywords for expressing the same concepts only 20% of the time.



Let’s illustrate the problem resulting from synonyms and homonyms by an example. Among these three documents at the level of terms doc1 and doc2 are highly related, whereas doc3 has no similarity with the other two documents. Understanding the meaning of the words, it is however easy to see that in reality doc2 and doc3 are closely related, as they talk about mobile communications, whereas doc1 is completely unrelated to the others as it is about health and nutrition.

Key Idea

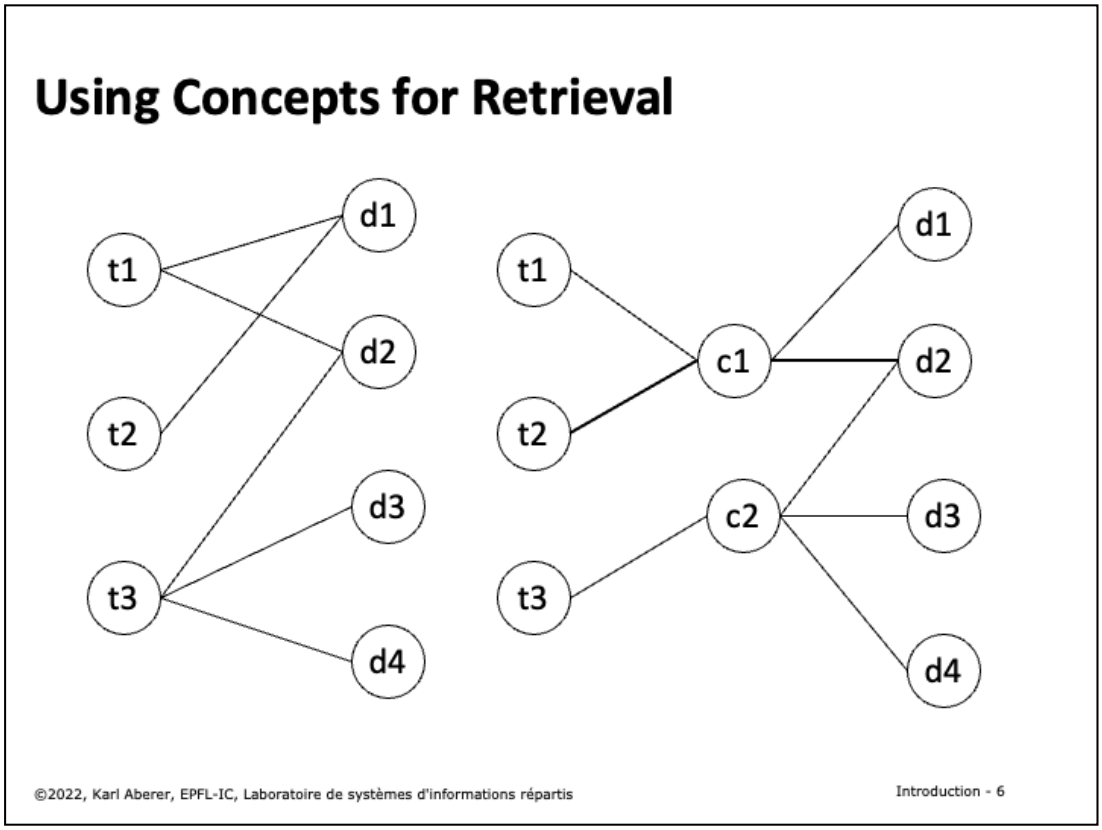
Map documents and queries into a lower-dimensional space composed of higher-level concepts

- Each concept represented by a combination of terms
- Fewer concepts than terms
- e.g. vehicle = {car, automobile, wheels, auto, sportscar}

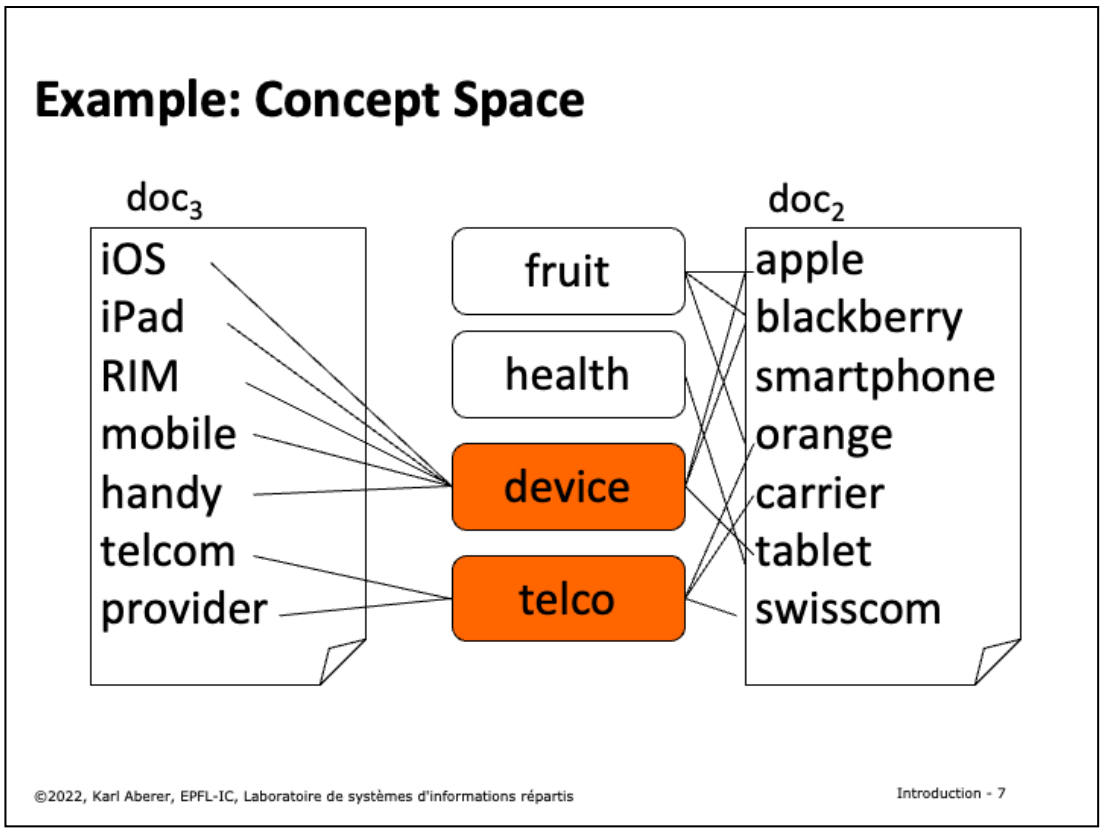
Dimensionality reduction

- Retrieval (and clustering) in a reduced concept space might be superior to retrieval in the high-dimensional space of index terms

Thus, it would be interesting to represent the meaning of documents and queries in information retrieval models on concepts, instead on terms. To that end, it is first necessary to define a "concept space" to which documents and queries are mapped, and compute similarity within that concept space. The concept space should ideally have much lower dimension than the term space, whose dimensionality is determined by the size of the vocabulary.



This figure illustrates the approach: rather than directly relating documents d and terms t, as in vector space retrieval, there exists an intermediate layer of concepts c to which both queries and documents are mapped. The concept space can be of a smaller dimension than the term space. In this small example we can imagine that the terms t1 and t2 are synonyms and thus related to the same concept c1. If now a query t2 is posed, in the standard vector space retrieval model only the document d1 would be returned, as it contains the term t2. By using the intermediate concept layer the query t2 would also return the document d2.



Applying this idea to our example before, we can consider a concept space consisting of four concepts, two related to health, two related to mobile communication. For doc2 and doc3 we recognize much better the close conceptual relationship the two documents have.

Similarity Computation in Concept Space

Concept represented by terms, e.g.

device = {iOS, iPad, RIM, mobile, handy, tablet, apple, blackberry}

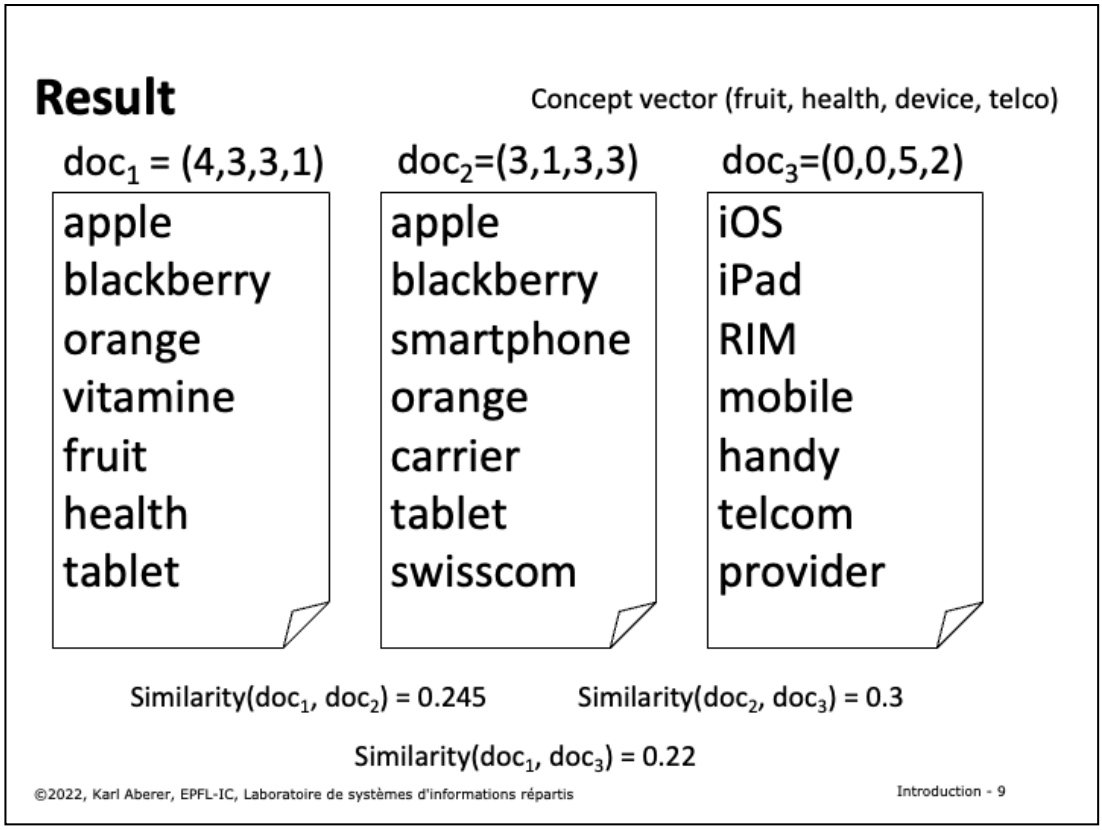
Document represented by concept vector, counting number of concept terms, e.g.

$doc_1 = (4, 3, 3, 1)$

$doc_3 = (0, 0, 5, 2)$

Similarity computed by scalar product of normalized concept vectors

We may consider concepts as being represented by sets of terms, and documents by concept vectors that count how many concept terms occur in the document. Using this approach we would obtain the non-normalized concept vectors doc1 = (4,3,3,1), doc2=(3,1,3,3) and doc3=(0,0,5,2).



After normalizing these vectors we compute the cosine similarities among the resulting concept vectors and obtain:

Sim (2,3) = 0.3

Sim (1,2) = 0.245

Sim (1,3) = 0.22

This result shows that indeed documents 2 and 3 are semantically closer, though still some confusion remains due to the large number of homonyms occurring in these documents.

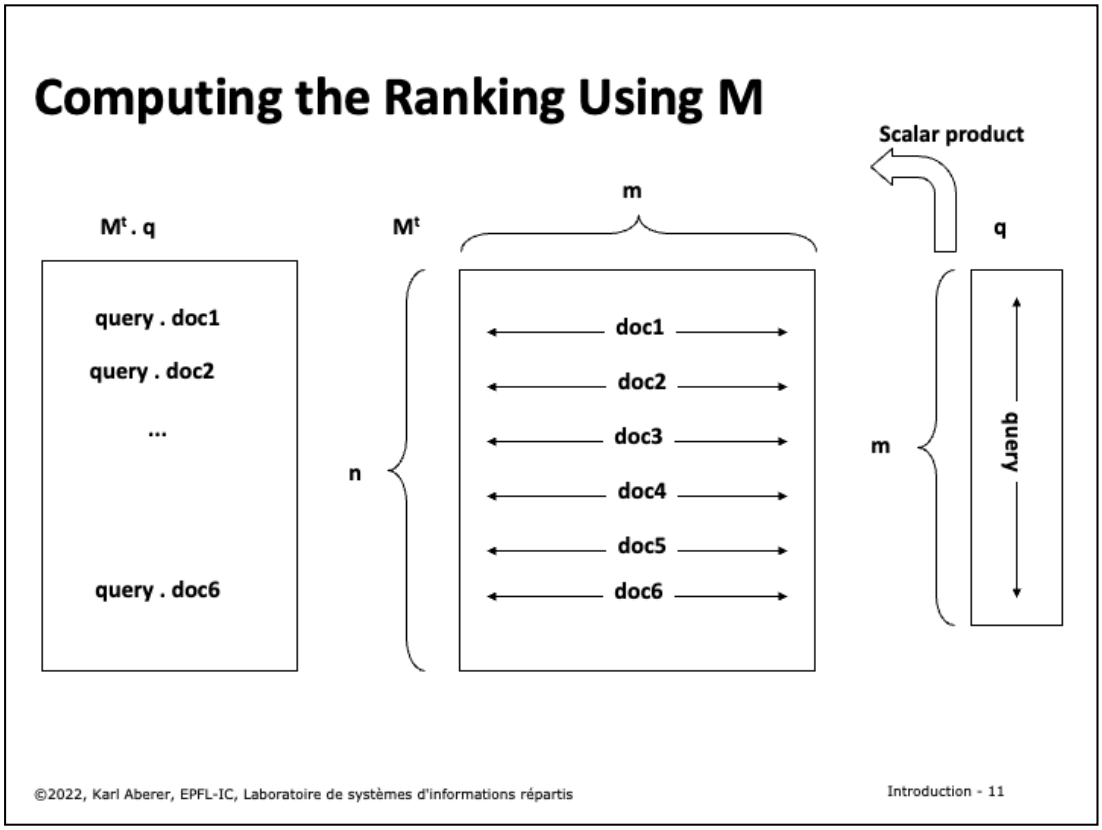
Basic Definitions

Problem: how to identify and compute "concepts" ?

Consider the term-document matrix

- Let M_{ij} be a term-document matrix with m rows (terms) and n columns (documents)
- To each element of this matrix is assigned a weight w_{ij} associated with t_i and d_j
- The weight w_{ij} can be based on a tf-idf weighting scheme

The challenge is to find a method that identifies and characterizes important concepts in document collections. One approach would be to perform this task manually, e.g. by using a predefined ontology and let users annotate documents using terms of the ontology. This is an approach that his been used in libraries, but is labor intensive. Thus we will now present a method that performs the task of concept identification and document classification by concepts automatically. Starting point for the method is the term-document matrix that we have introduced for vector space retrieval with weigths based on a tf-idf weighting scheme.



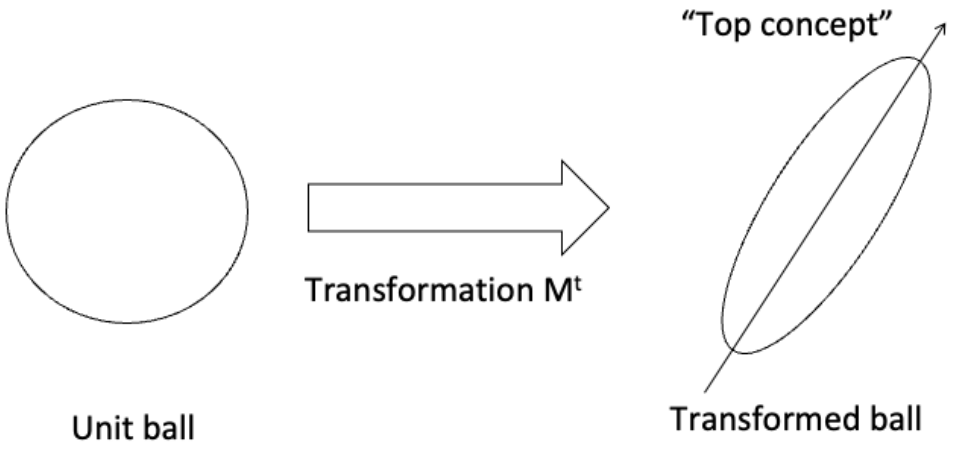
We can interpret the process of producing a retrieval result in vector space retrieval as a matrix operation. This is illustrated in this figure. The ranking results from computing the product of a query vector q with the term-document matrix M. We assume that all columns in M and q are normalized to 1.

In vector space retrieval each row of the matrix M corresponds to

A. A document
B. A concept
C. A query
D. A term

Identifying Top Concepts

Key Idea: extract the essential features of M^t and approximate it by the most important ones



One way to understand of how concepts can be extracted from a document collection is to consider the effect of the term-document matrix on data. If we apply this matrix to a (high-dimensional) unit ball it will distort this ball into an ellipsoid. This ellipsoid will have one direction with the strongest distortion. We may think of this direction as corresponding to a particularly important concept in the document collection.

Singular Value Decomposition (SVD)

Represent Matrix M as $M = K.S.D^t$

- K and D are matrices with orthonormal columns

$$K.K^t = I = D.D^t$$

- S is an r x r diagonal matrix of the singular values sorted in decreasing order where r = min(m, n), i.e. the rank of M
- Such a decomposition always exists and is unique (up to sign)

To identify the principal direction of distortion induced by the matrix transformation, a standard tool from linear algebra can be used, the singular value decomposition (SVD). SVD decomposes a matrix into the product of three matrices. The middle matrix S is a diagonal matrix, where the elements of this matrix are the singular values of the matrix M.

Construction of SVD

K is the matrix of eigenvectors derived from $M.M^t$

D is the matrix of eigenvectors derived from $M^t.M$

Algorithms for constructing the SVD of a m x n matrix have complexity $O(n^3)$ if $m \leq n$

Formally the SVD can be computed by constructing Eigenvectors of matrices derived from the original matrix M. This computation can be performed in time O(n^3). Note that the time complexity is considerable for large matrices, which makes the approach computationally expensive. There exist however also approximation and randomized techniques to perform this decomposition more efficiently (e.g. https://research.fb.com/blog/2014/09/fast-randomized-svd/).

Interpretation of SVD

We can write M = K.S.D^t as sum of outer vector products

$$M = \sum_{i=1}^{r} s_i k_i \otimes d_i^t$$

The s_i are ordered in decreasing size

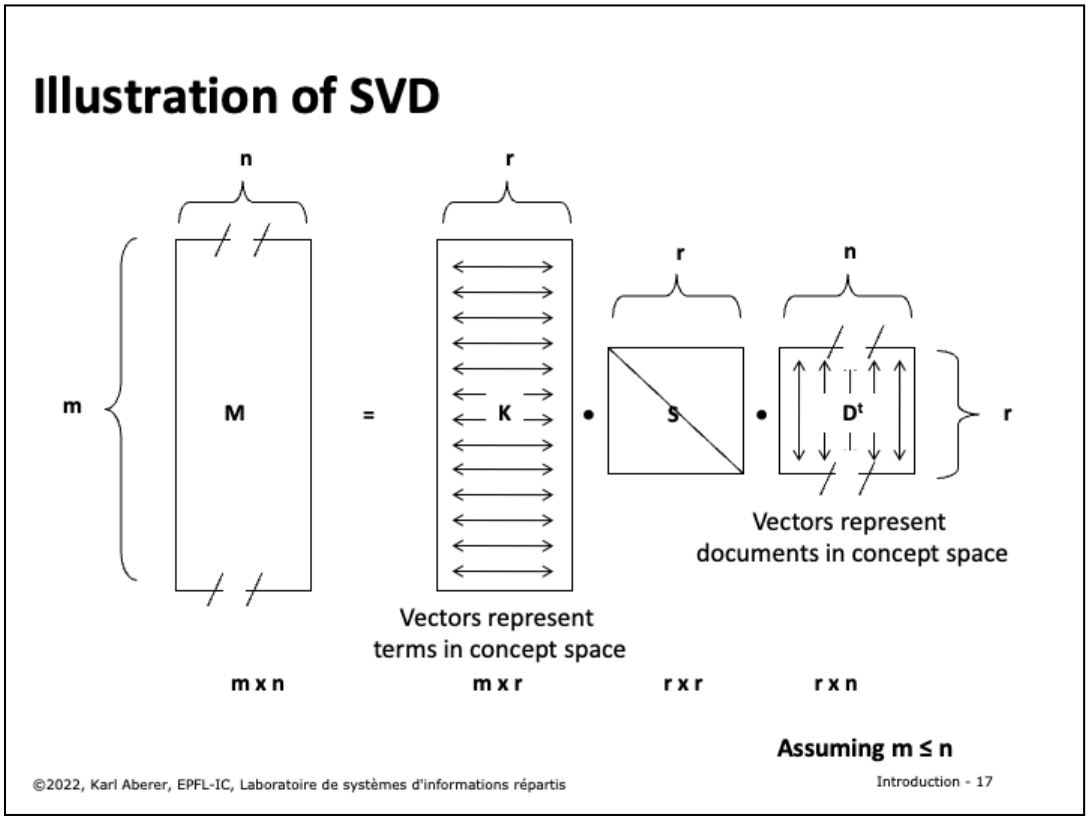
By taking only the largest ones we obtain a «good» approximation of M (least square approximation)

The singular values s_i are the lengths of the semi-axes of the hyperellipsoid E defined by

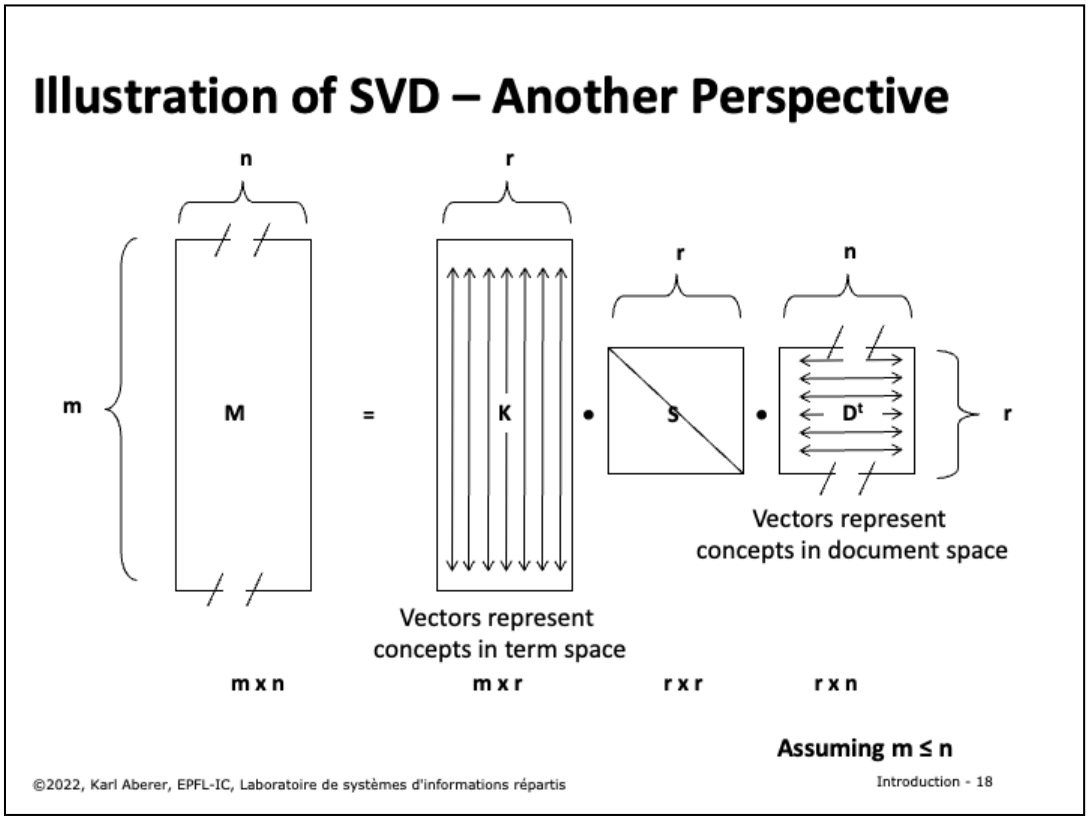
$$E = \left\{ Mx \mid \|x\|_2 = 1 \right\}$$

One way to understand of how the SVD extracts the important concepts from the term-document matrix is the following: the decomposition can be used to rewrite the original matrix as the sum of components that are weighted by the singular values. Thus we can obtain approximations of the matrix by only considering the larger singular values. The SVD after eliminating less important dimensions (smaller singular values) can be interpreted as a least square approximation to the original matrix. The symbol ⊗ denotes the **outer product** of two vectors, d_i is the i-th row of D. The outer product of vectors u =. (u_1,…,u_m) and v = (v_1,…,v_n) is given by the mxn matrix m with entries m_ij = u_i v_j.

The singular values have also a geometric interpretation, as they tell us how a unit ball (||x||=1) is distorted when the linear transformation defined by the matrix M is applied to it. We can interpret the axes of the hyperellipsoid E as the dimensions of the concept space.



This figure illustrates the structure of the matrices generated by the SVD. In general, m <= n, i.e., the number of documents is larger than the size of the vocabulary. The rows in K can then be interpreted as the representation of terms in the concept space, and the rows in D as the representation of documents in the concept space.



We can also see of how the concepts are represented by terms respectively documents. In particular, each concepts is now described as a vector of weighted terms.

Latent Semantic Indexing (LSI)

In the matrix S, select only the s largest singular values

- Keep the corresponding columns in K and D

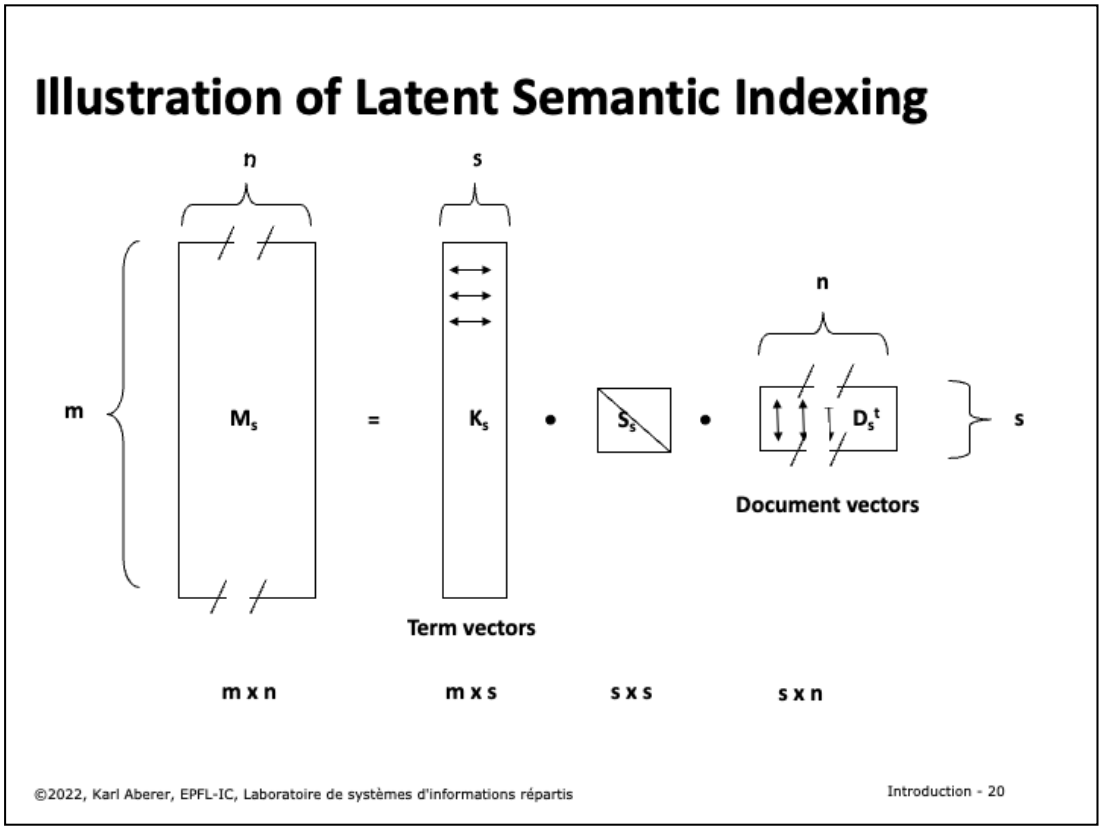
The resultant matrix is called M_s and is given by

- $M_s = K_s.S_s.D_s^t$ where s, $s < r$, is the dimensionality of the concept space

The parameter s should be

- large enough to allow fitting the characteristics of the data
- small enough to filter out the non-relevant representational details

Using the singular value decomposition, we can now derive an "approximation" of M by retaining only the s largest singular values in matrix S. The choice of s determines on how many of the "important concepts" the ranking will be based on. The assumption is that concepts with small singular value in S are rather to be considered as "noise" and thus can be dismissed. The resulting method is called Latent Semantic Indexing (LSI).



This figure illustrates the structure of the matrices after reducing the dimensionality of the concept space to s, when only the first s singular values are kept for the computation of the ranking. The rows in matrix K_s correspond to term vectors, whereas the columns in matrix D_s^t correspond to document vectors. By using the cosine similarity measure between columns of matrix D_s^t a similarity measure of documents can be computed.

Answering Queries

Documents can be compared by computing cosine similarity in the concept space, i.e., comparing their columns $(D_s^t)_i$ and $(D_s^t)_j$ in matrix D_s^t

A query q is treated like one further document

- it is added as an additional column to matrix M
- the same transformation is applied to this column as for mapping M to D

After performing the SVD, the similarity of different documents can be determined by computing the cosine similarity measure among their representation in the concept space (the columns of matrix D_s^t). Queries are considered like documents that are added to the document collection. For answering queries, the query is treated like a document, and its representation in the concept space is used to compute the similarity to documents.

Mapping Queries

Mapping of M to D

$M = K.S.D^t$

$S^{-1}.K^t.M = D^t$ (since $K.K^t = 1$)

$D = M^t.K.S^{-1}$

Apply same transformation to q:

$q^* = q^t.K_s.S_s^{-1}$

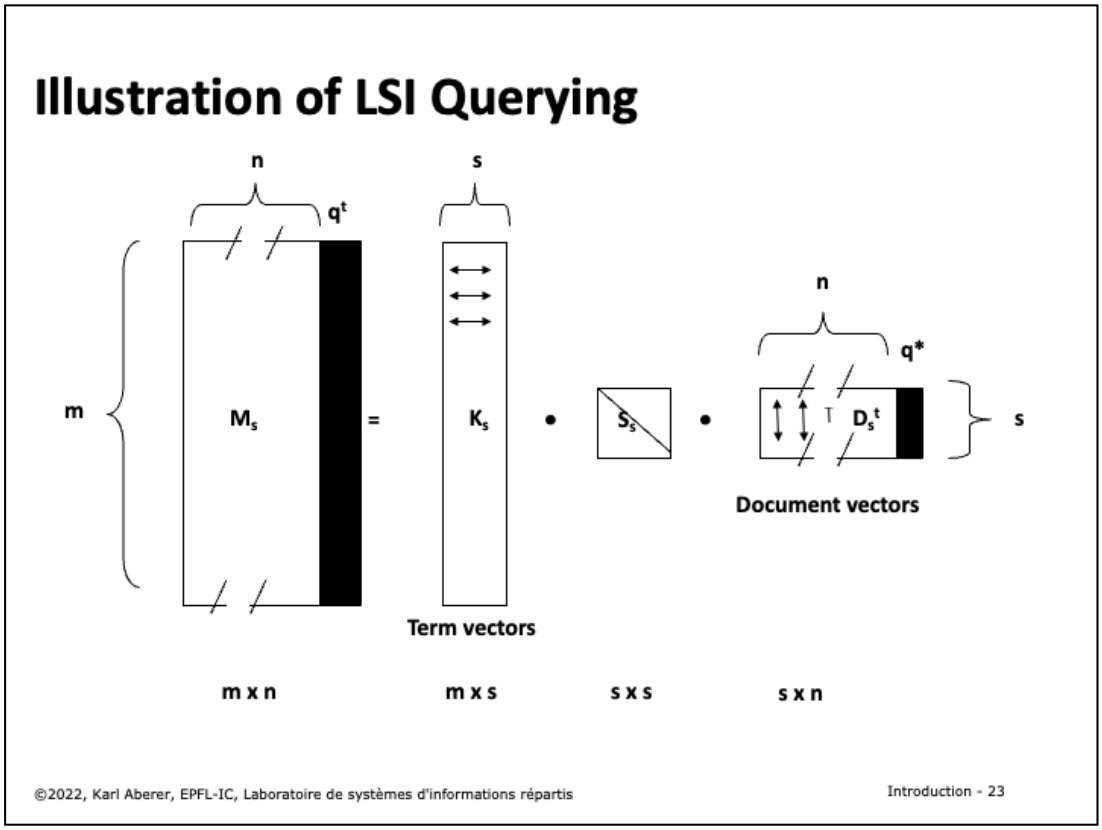
Then compare transformed vector by using the standard cosine measure

$$sim(q^*, d_i) = \frac{q^* \bullet (D_s^t)_i}{|q^*| |(D_s^t)_i|}$$

The transformation of queries into concept vectors works as follows: when a new column (the query) is added to M, we have to apply the same transformation to this new column, as to the other columns of M, in order to produce the corresponding column in the matrix D_s^t, representing documents in the concept space. We exploit the fact that $K_s^t.K_s = 1$.

Since $M_s = K_s.S_s.D_s^t$ we obtain $S_s^{-1}.K_s^t.M_s = D_s^t$ or $D_s = M_s^t.K_s.S_s^{-1}$.

This transformation is applied to the query vector q to obtain a query vector q* in the concept space. After that step, the similarity of the query to the documents in the concept space can be computed. ($(D_s^t)_i$ denotes the i-th column of matrix D_s^t)



This figure illustrates of how a query vector is treated like an additional document vector.

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.

Implementation in Python

```
from sklearn.feature_extraction.text import TfidfVectorizer

tf = TfidfVectorizer(analyzer='word', ngram_range=(1,1), min_df = 2, stop_words = 'english')
features = tf.fit_transform(titles)
M = np.transpose(np.array(features.todense()))
```

```
# compute SVD
K, S, Dt = np.linalg.svd(M, full_matrices=False)

# LSI select dimensions
K_sel = K[:,0:2]
S_sel = np.diag(S)[0:2,0:2]
Dt_sel = Dt[0:2,:]
```


Given Python libraries that compute the SVD, we can implement the LSI method directly using the definitions and based on the term-document matrix that we have already computed for the vector space retrieval model.

Results (s=2)

```
K_sel
```

```
array([[ 0.01781272, -0.4729881 ],
       [ 0.03264057, -0.43230378],
       [ 0.15088442, -0.17568951],
       [ 0.55589867,  0.07082109],
       [ 0.6843092 ,  0.1075997 ],
       [ 0.01570413, -0.37133288],
       [ 0.09073864, -0.07173948],
       [ 0.01775573, -0.20943739],
       [ 0.19758761,  0.08201858],
       [ 0.11060875,  0.05205271],
       [ 0.19758761,  0.08201858],
       [ 0.15088442, -0.17568951],
       [ 0.20802226,  0.09313466],
       [ 0.01555703, -0.22913745],
       [ 0.10330872, -0.00853892],
       [ 0.14994428, -0.49674497]])
```

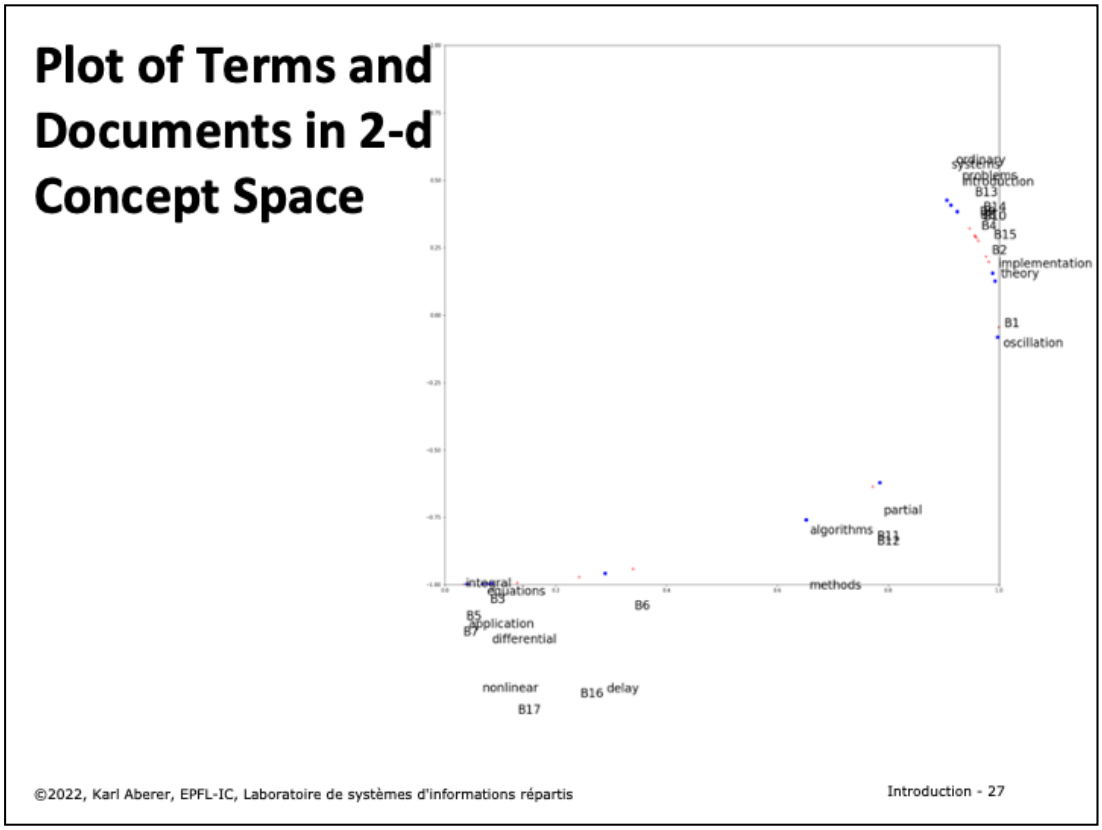
```
S_sel
```

```
array([[2.12109044, 0.        ],
       [0.        , 1.50037763]])
```

```
np.transpose(Dt_sel)
```

```
array([[ 0.18982901, -0.0083562 ],
       [ 0.32262142,  0.07171508],
       [ 0.04727092, -0.58437076],
       [ 0.33399497,  0.1003478 ],
       [ 0.01183895, -0.31440669],
       [ 0.03875274, -0.10771362],
       [ 0.01329859, -0.40782546],
       [ 0.3018997 ,  0.09205811],
       [ 0.07134057,  0.02202618],
       [ 0.33016936,  0.09458633],
       [ 0.29162009, -0.24019296],
       [ 0.29162009, -0.24019296],
       [ 0.29566823,  0.10051203],
       [ 0.33016936,  0.09458633],
       [ 0.40993831,  0.08283115],
       [ 0.03543573, -0.14179904],
       [ 0.05664377, -0.43260133]])
```


Her we inspect the resulting matrices, when selecting s=2 dimensions.



Since in our example the concept space has two dimensions, we can plot both the documents and the terms in this 2-dimensional space. It is interesting to observe of how semantically "close" terms and documents cluster in the same regions. This illustrates very well the potential of latent semantic indexing in revealing the « essential concepts » in document collections.

Applying SVD to a term-document matrix M. Each concept is represented in K

A. as a singular value
B. as a linear combination of terms of the vocabulary
C. as a linear combination of documents in the document collection
D. as a least squares approximation of the matrix M

The number of term vectors in the matrix K_s used for LSI

A. Is smaller than the number of rows in the matrix M
B. Is the same as the number of rows in the matrix M
C. Is larger than the number of rows in the matrix M

A query transformed into the concept space for LSI has …

A. s components (number of singular values)
B. m components (size of vocabulary)
C. n components (number of documents)

Discussion of Latent Semantic Indexing

Latent semantic indexing provides an interesting conceptualization of the IR problem

Advantages

- It allows reducing the complexity of the underlying concept representation
- Facilitates interfacing with the user

Disadvantages

- Computationally expensive
- Poor statistical explanation

LSI has been a fundamental advance in the development of informational retrieval. However, it also suffers from conceptual problems.

For example, least squares approximation is based on the assumption that term frequencies are normally distributed. This is not consistent with the observation that term frequencies are power-law distributed.

Alternative Techniques

Probabilistic Latent Semantic Analysis

- Based on Bayesian Networks

Latent Dirichlet Allocation

- Based on Dirichlet Distribution
- State-of-the-art method for concept extraction

Same objective of creating a lower-dimensional concept space based on the term-document matrix

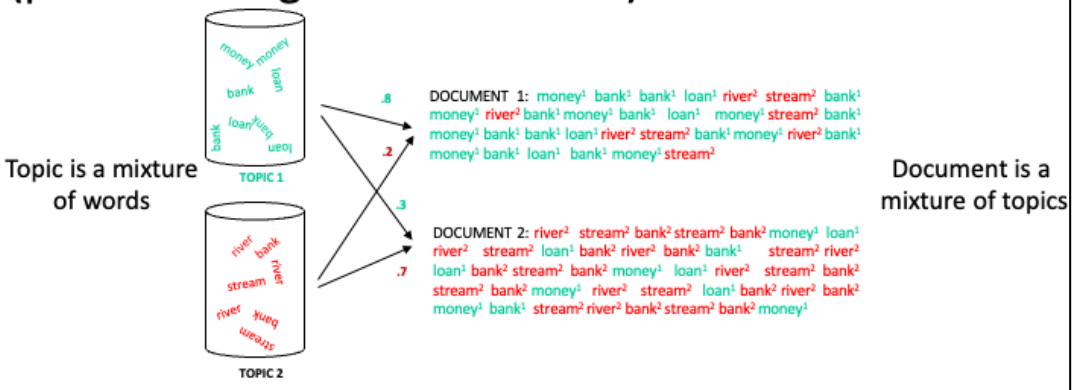
- Better explained mathematical foundation
- Better experimental results

LSI has triggered the search for different alternative approaches for better capturing the semantic properties of text in information retrieval.

One recent successful approach is LDA, based on the use of Dirichlet distributions. Like LSI it generates a concept space, where concepts are represented as term vectors. The method has better theoretical foundations and empirically it produces better results. It is nowadays considered as a golden standard. The approach is mathematically more involved than LSI, and therefore we will not be able to develop this method in this lecture.

1.4.2 Latent Dirichlet Allocation (LDA)

Idea: assume a document collection is (randomly) generated from a known set of topics (probabilistic generative model)



This illustration shows the basic idea underlying LDA: it is assumed that there exist topics that are represented as mixtures of words. In the example we see words that are related to two meanings of “bank”, the financial institution and the river bank, and are represented by different related works. Then, documents are assumed to be generated from a mixture of topics, where the mixture is determined by weights, as indicated in the figure. As a result, each document contains terms form its associated topics in the right proportion.

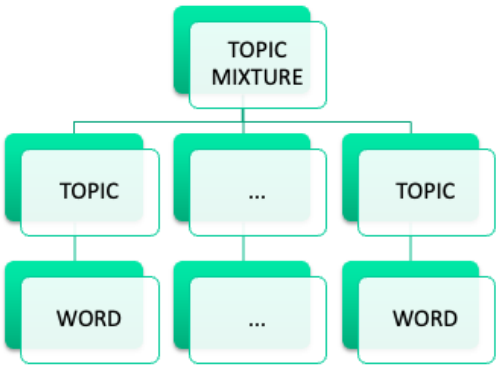
Like the probabilistic language model used in information retrieval, this approach us an example of a probabilistic generative model.

Document Generation using a Probabilistic Process

For each document, choose a mixture of topics

For every word position, sample a topic from the topic mixture

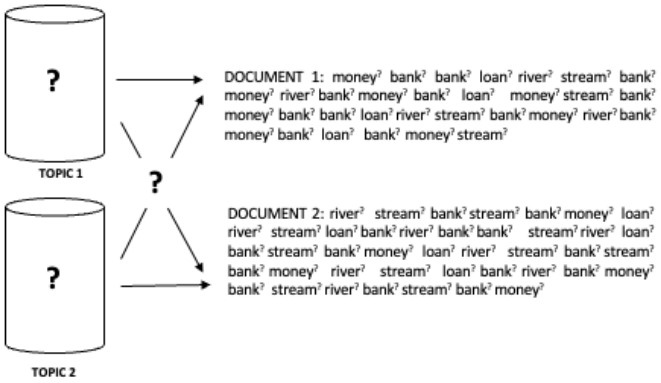
For every word position, sample a word from the chosen topic



Given the topic model, we obtain a probabilistic process for generating documents.

LDA: Topic Identification

Approach: Inverting the process: given a document collection, reconstruct the topic model



While it is straightforward to generate documents, once the probabilistic topic model is given, deriving a topic model for a given document collection is a hard problem. To solve this problem, a probabilistic topic model needs to be derived, that is likely to generate the document collection under consideration.

Latent Dirichlet Allocation

Topics are **interpretable** unlike the arbitrary dimensions of LSI

- Distributions follow a Dirichlet distribution
- Construction of topic model is mathematically involved, but computationally feasible
- Considered as the state-of-the art method for topic identification

We do not present the details of LDA here, as the method is mathematically fairly involved. However, it is important to note that it is considered today as the state-of-the-art method of topic detection.

Use of Topic Models

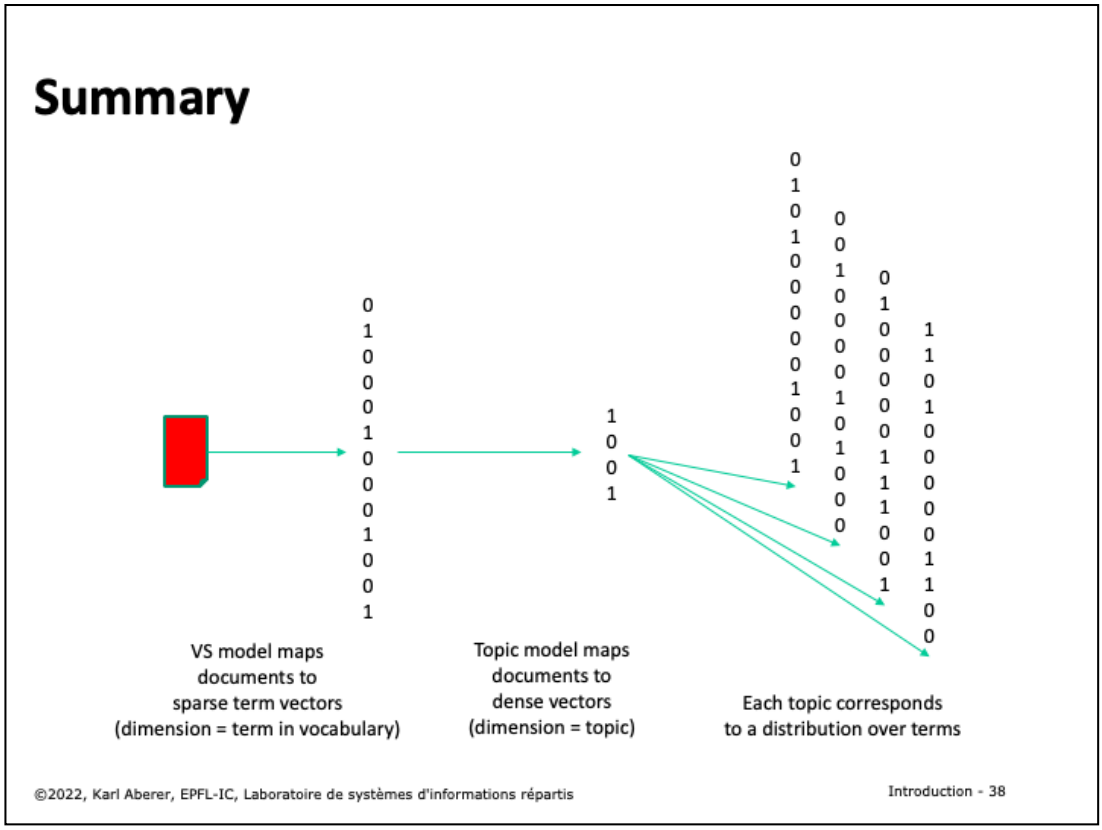
Unsupervised Learning of topics

- Understanding main topics of a topic collection
- Organizing the document collection

Use for document retrieval: use topic vectors instead of term vectors to represent documents and queries

Document classification (Supervised Learning): use topics as features

Topic models provide a representation of documents at a conceptual level. Therefore, they are applied in many different contexts, including document retrieval and classification.



This illustration summarizes the connection between the vector space model, which we introduced for information retrieval and topic models that produce low-dimensional (dense) representation of documents.