Chapter 11: Text Clustering

Overview

- Cluster analysis:
  - concept
  - distance and similarity functions between two objects
  - proximity functions between two clusters
  - cluster algorithms
  - number of clusters
- **Term clustering**:
  - algorithms
  - applications and their results:
    - query expansion
    - thesaurus construction

Overview

- **Document clustering**:
  - algorithms
  - applications and their results
    - cluster retrieval model
    - topic overviews
    - clustering of retrieval results
    - event detection

Cluster analysis: concept

- = a multivariate statistical technique that allows an automatic generation of groups in data
- Clustering supposes:
  1. an abstract representation of the object to be clustered containing the features for the grouping (e.g., feature vector)
  2. a function that calculates the relative importance of the features (e.g., weighting)
  3. a function that calculates a numerical distance or similarity between the representations of objects
  4. constraints w.r.t. cluster membership, cluster proximity, shape of the clusters, ...

Cluster analysis: concept

- In case of feature vectors, uses a multivariate *n* x *p* data matrix X :

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ . & & & \\ . & & & \\ . & & & \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}$$

where *n* = number of objects to be clustered

p = number of features (attributes, variables) measured

->The purpose of the cluster analysis is to group the objects that are represented by the rows of X

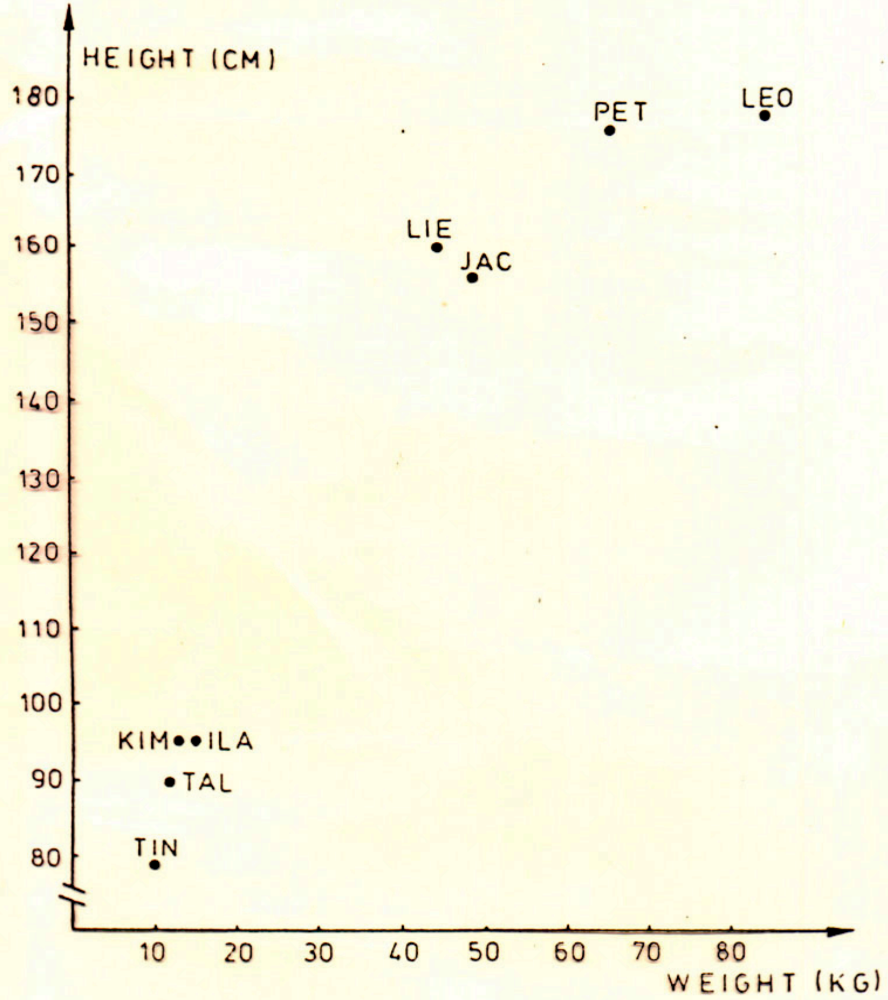


Figure 1 A plot of eight objects.

[Kaufman & Rousseeuw 1990]

Cluster analysis: concept

- **Feature selection** and **extraction**:
  - choice of the features and their weighting: how relevant a certain feature is found with respect to the grouping sought

Cluster analysis: concept

- Results:
  - a hierarchical grouping of the objects
  - partitioning of the collection in a number of groups or clusters
- A cluster can be represented by:
  - **centroid:** a kind of dummy object, that is computed based on the individual representations of the objects of the clusters, e.g., average vector, when individual objects are represented as a vector
  - **representative object**: e.g., medoid object that has the least average (or total) distance or largest average similarity with all other objects of its cluster

Distance and similarity functions

- Cluster methods often use a matrix that indicates the distance or the similarity between each pair of objects
- Example of a (dis)similarity matrix for the objects: $\mathbf{x}_1,\ldots,\mathbf{x}_n$ (n = 5) (case of a symmetric distance or similarity function)

$$
\begin{array}{cc}
 & \begin{array}{ccccc} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 & \mathbf{x}_5 \end{array} \\
\begin{array}{c} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \end{array} &
\begin{bmatrix}
 & & & & \\
0.1 & & & & \\
0.5 & 0.4 & & & \\
0.6 & 0.5 & 0.3 & & \\
0.8 & 0.7 & 0.5 & 0.4 &
\end{bmatrix}
\end{array}
$$

Distance and similarity functions

- Symmetric functions: e.g., Euclidean distance, cosine function, ...
- Asymmetric functions: e.g., Kullback-Leibler divergence
- Other application-dependent functions: e.g., for computing the similarity between two feature vectors, when the vectors have mixed values; kernel function that computes the similarity between structured objects (e.g., strings, trees)

  [see Retrieval models, Text categorization]

Proximity functions

- Proximity function between two clusters:
  - **maximum proximity**: defines proximity based on their most similar pair of objects
  - **minimum proximity**: defines proximity based on their least similar pair of objects
  - **average proximity**: defines proximity based on the average of the similarities between all pairs of objects
  - **mean proximity**: defines proximity based on the similarity of the representative (e.g., centroid, medoid) of each cluster

Cluster algorithms

- Sequential algorithms
- Hierarchical algorithms
- Algorithms based on cost function optimization

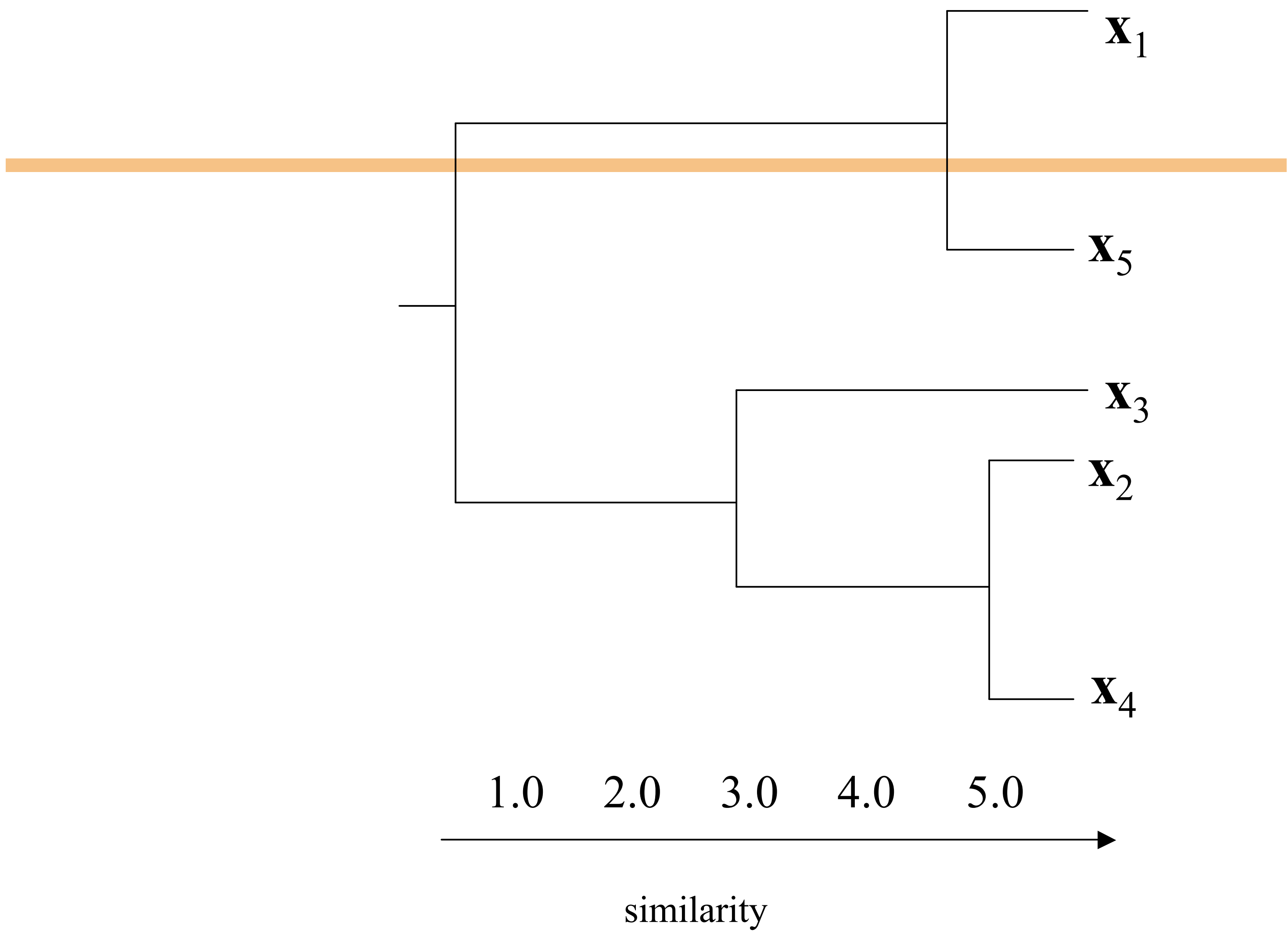
Sequential algorithms

- In one or maximum a few iterations, the clustering is built
- **Single pass algorithm**: in one pass all n objects are assigned to their closest cluster based on a threshold similarity value

Hierarchical clustering

- Two strategies:
  - **agglomerative clustering**:
    - starts from *n* individual objects which in consequent steps are grouped in more general clusters and finally into 1 cluster
  - **divisive clustering**:
    - a complete collection of *n* objects is divided in smaller and smaller groups until the *n* single objects are found

Dendrogram


$\mathbf{x}_1$
$\mathbf{x}_5$
$\mathbf{x}_3$
$\mathbf{x}_2$
$\mathbf{x}_4$
1.0 2.0 3.0 4.0 5.0
similarity

Agglomerative clustering

Initialization:

Choose $\Re_0 = \{C_i = \{\mathbf{x}_i\}, i = 1, \ldots, n\}$ as the initial clustering (i.e., set of singleton clusters)

$t = 0$

Repeat

among all possible pairs of clusters (C_r, C_s) in $\Re_t$ search the most similar pair of clusters (C_i, C_j) based on the chosen proximity function

define $C_{ij} = C_i \cup C_j$ and produce a new clustering:

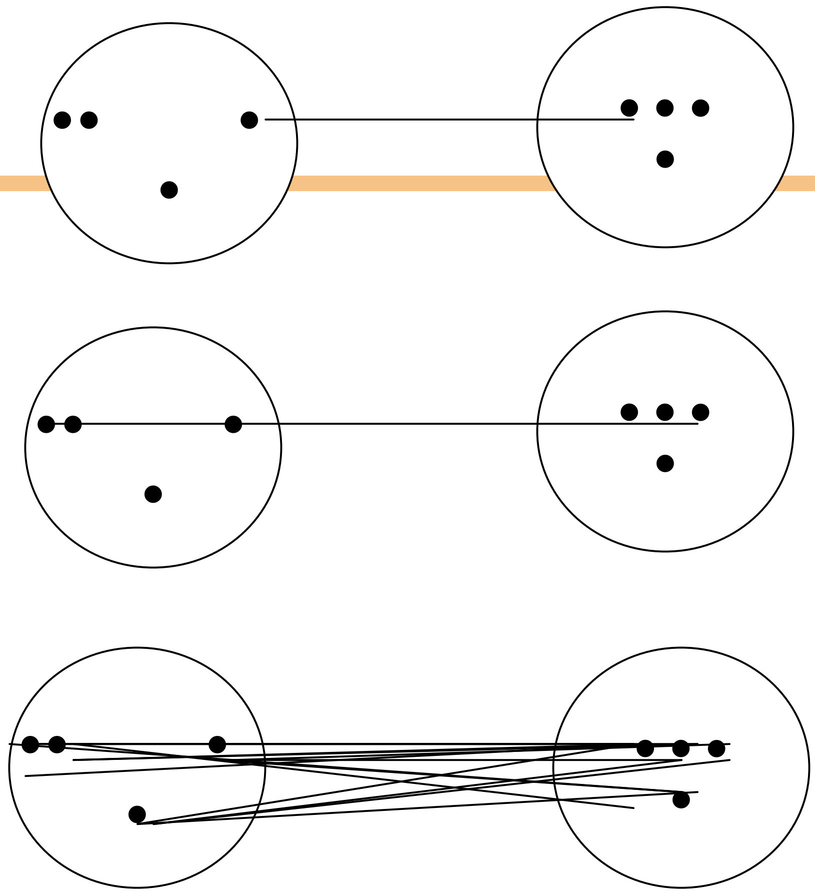
$$\Re_{t+1} = \{(\Re_t - \{C_i, C_j\}) \cup \{C_{ij}\}\}$$

t++;

Until $|\Re_t| = 1$

Agglomerative clustering

- Methods differ in their definition of proximity between clusters:
  - **single link(age)**(nearest neighbor) clustering:
    - use of the maximum proximity function
    - might generate drawn out clusters
  - **complete link(age)** (furthest neighbor) clustering:
    - use of the minimum proximity function tends to produce very compact clusters with small diameter
  - **group average link(age)**
    - use of the average proximity function
    - generates roughly ball shaped clusters
    - efficient variant: based on the mean proximity function



single linkage

complete linkage

group average linkage

- In the above matrix in a first step, a cluster is already formed with the objects x_1 and x_5. The similarity of this cluster with the remaining clusters is computed as sim $((x_1,x_5),x_i)$ where i = 2, 3 or 4.

- **Single link clustering:**

sim $((x_1,x_5),x_2)$ = max[sim(x_1,x_2), sim(x_5,x_2)] = sim(x_5,x_2)= 0.7
sim $((x_1,x_5),x_3)$ = max[(sim(x_1,x_3), sim(x_5,x_3)] = sim(x_5,x_3)= 0.5
sim $((x_1,x_5),x_4)$ = max[sim(x_1,x_4), sim(x_5,x_4)] = sim(x_1,x_4)= 0.6

- **Complete link clustering**:

  $sim((x_1,x_5),x_2) = min[sim(x_1,x_2), sim(x_5,x_2)] = sim(x_1,x_2) = 0.1$

  $sim((x_1,x_5),x_3) = min[sim(x_1,x_3), sim(x_5,x_3)] = sim(x_1,x_3) = 0.5$

  $sim((x_1,x_5),x_4) = min[sim(x_1,x_4), sim(x_5,x_4)] = sim(x_5,x_4) = 0.4$

- **Group average link clustering**:

  $sim((x_1,x_5),x_2) = (sim(x_1,x_2) + sim(x_5,x_2)) / 2 = 0.4$

  $sim((x_1,x_5),x_3) = (sim(x_1,x_3) + sim(x_5,x_3)) / 2 = 0.5$

  $sim((x_1,x_5),x_4) = (sim(x_1,x_4) + sim(x_5,x_4)) /2 = 0.5$

Divisive clustering

- Often iteratively each cluster is split in a few clusters by means of a partitioning algorithm
- Distinct advantage: possible to generate few large clusters early in the clustering process

Cost function optimization

- The clustering is evaluated based on a cost function *J*
- Usually the number of clusters (*k*) is fixed
- The algorithms start from an initial grouping into *k* clusters and iteratively other groupings into *k* clusters are tested in an attempt to optimize *J*
- Often used for **partitioning** *n* objects into *k* clusters

Examples

- **Hard clustering**:
  - for a chosen number of *k* clusters: e.g.,
    - popular method: *k*-means algorithm
    - *k*-medoid algorithm
- **Soft or fuzzy clustering**: object might belong to different clusters with degree of membership quantified by membership coefficients (not much used in text based information retrieval)
  - for a chosen number of *c* clusters: e.g.,
    - fuzzy *c*-means

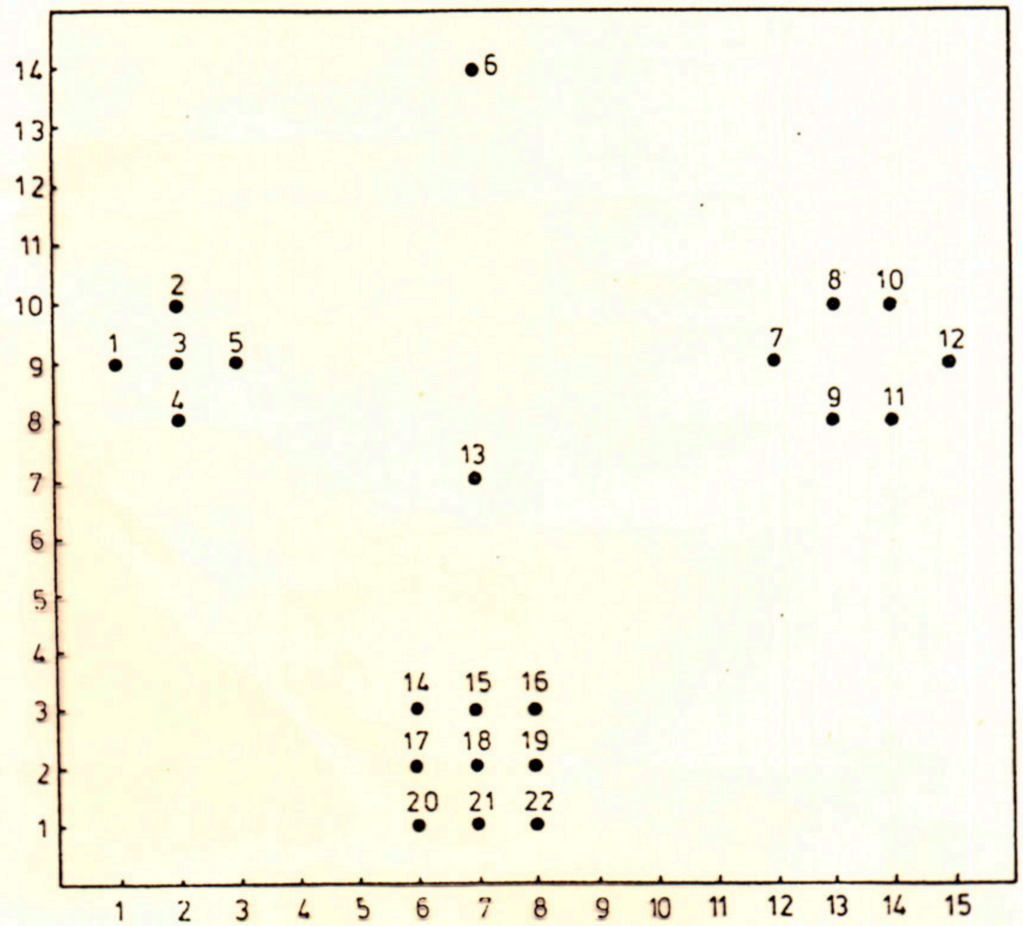


Figure 1 Data set with two intermediate objects.

[Kaufman & Rousseeuw 1990]

k-means algorithm

Let $\mathbf{x}_i \in \Re^p$, i=1,...,*n* denote the objects

Let $\mathbf{c}_j \in \Re^p$, j=1,...,*k* denote the centroid vectors of the *k* clusters C_j, with $2 <= k < n$

Objective:

$$maximize \sum_{j=1}^{k} \sum_{\substack{i=1 \\ \mathbf{x}_i \in C_j}}^{n} sim(\mathbf{x}_i, \mathbf{c}_j)$$

1. Take *k* objects (with $2 <= k < n$) as singleton clusters from a set of *n* objects randomly or such that the object points are mutually farthest apart
2. Assign each of the remaining *n* - *k* objects to the cluster with the nearest centroid; recompute the centroid of the gaining cluster after each assignment

k-means algorithm

3. REPEAT until no more changes are recorded

   in sequence: assign each object to the cluster with the nearest centroid

   each time an object x_i changes from cluster C_v to cluster C_w, compute the centroid $\mathbf{c}_v$ of C_v and the centroid $\mathbf{c}_w$ of C_w:

$$\mathbf{c}_v = \frac{1}{n_v - 1}\left(n_v \mathbf{c}_v - \mathbf{x}_i\right)$$

$$\mathbf{c}_w = \frac{1}{n_w + 1}\left(n_w \mathbf{c}_w + \mathbf{x}_i\right)$$

where n_v = number of objects in cluster C_v

n_w = number of objects in cluster C_w

k-medoid algorithm

Let $\mathbf{x}_i \in \Re^p$, i=1,...,*n* denote the objects

Let $\boldsymbol{m}_j \in \Re^p$, j=1,...,*k* denote the medoid vectors of the *k* clusters C_j , with 2 < = *k* < *n*

Objective:

$$maximize \sum_{j=1}^{k} \sum_{\substack{i=1 \\ \mathbf{x}_i \in C_j}}^{n} sim(\mathbf{x}_i, \mathbf{m}_j)$$

1. Take *k* objects (with 2< = k < *n*) as singleton clusters from a set of *n* objects randomly or such that the object points are mutually farthest apart: they form the medoids
2. Assign each of the remaining *n* - *k* objects to the cluster with the nearest medoid

k-medoid algorithm

3. Iteratively swap objects, i.e., by considering all pairs of objects ($\mathbf{x}_i$, $\mathbf{x}_h$) for which object $\boldsymbol{x}_i$ has been selected as medoid and $\mathbf{x}_h$ not, until the objective function cannot be improved anymore

Time and space complexities

- **Single pass**: time: close to $O(n)$ when n is large and number of clusters is small
- **Hierarchical** (agglomerative):
  - time: theoretically (cf. algorithm on slide 16) :
    - at each level t, there are n - t clusters, and $\binom{n-t}{2}$ or $\frac{(n-t)(n-t-1)}{2}$ pairs are considered
    - there are n -1 levels considered:

$$\sum_{t=1}^{n-1}\binom{n-t}{2} = \sum_{t=1}^{n-1}\left(\frac{(n-t)(n-t-1)}{2}\right) \approx \frac{n^3}{6}$$

    - time: $O(n^3)$

Time and space complexities

In practice (depending on proximity function):

- **single link**: time: $O(n^2)$ space: $O(n)$
- **complete link**: time: $O(n^2 \log n)$ space: $O(n^2)$
- **group average link**:

  cosine similarity time: $O(n^2)$ space: $O(n^2)$

  mean proximity time: $O(n^2)$ space: $O(n)$

  (when relatively few non-zero similarities: space less than $O(n^2)$)

- ***k*-means**: when n is large and numbers of clusters and iterations are small: time complexity close to $O(n)$
- Complement with the complexities of computing the similarity or distance function

Number of clusters

- Hierarchical and partitioning algorithms: how to find a good number of clusters (= k)?
- In the absence of ground-truth:
  - different heuristics that take into account intra- and inter-cluster similarity between objects of the clustering: give only an indication of a best clustering

Number of clusters

- The most simple approaches only consider intra-cluster similarities:

$$\exists! C_j \in \Re_c : sim(C_j) < \theta$$

where

θ = threshold for the similarity *sim*(*Cj*)

sim(*Cj*) = similarity between a pair of objects of cluster *Cj* or the average pair wise similarity of objects in cluster *Cj*

Number of clusters

- The inter-cluster similarity between two clusters C_i and C_j is also taken into account: a final clustering must satisfy the following criterion (only necessary condition):

$$sim(C_i,C_j) \leq min\{sim(C_i),sim(C_j)\} \quad \forall C_i,C_j \in \Re_c \text{ and } C_i \neq C_j$$

$$sim(C_i,C_j) = \max_{\mathbf{x}_i \in C_i, \mathbf{x}_j \in C_j} sim(\mathbf{x}_i,\mathbf{x}_j)$$

Number of clusters

- Alternatively, for each object x_i of the cluster structure $\Re_c$ the degree of fitness $f(\mathbf{x}_i)$ of x_i to its cluster C_i is computed:

$$f(\mathbf{x}_i) = \frac{a(\mathbf{x}_i) - b(\mathbf{x}_i)}{\max\{a(\mathbf{x}_i), b(\mathbf{x}_i)\}}$$

where $a(\mathbf{x}_i)$ = average similarity of $\mathbf{x}_i$ to all other objects of its cluster C_i:

$$\frac{1}{r-1}\sum_{\mathbf{x}_j \in C_i} sim(\mathbf{x}_i, \mathbf{x}_j) \quad C_i \in \Re_c,\ \mathbf{x}_i \neq \mathbf{x}_j, \mathbf{x}_i \in C_i \text{ and } r = |C_i|$$

Number of clusters

$$b(\mathbf{x}_i) = \underset{C_j}{argmax} \frac{1}{r} \sum_{\mathbf{x}_j \in C_j} sim(\mathbf{x}_i, \mathbf{x}_j) \quad C_j \in \Re_c, \; C_i \neq C_j \text{ and } r = |C_j|$$

$$-1 \leq f(\mathbf{x}_i) \leq 1$$

- When C_i to which $\mathbf{x}_i$ belongs is a singleton cluster, it is unclear how $a(x_i)$ should be defined and then simply $f(\mathbf{x}_i) = 0$; also, when the clustering contains only one cluster, $f(\mathbf{x}_i)$ cannot be defined
- $f(\mathbf{x}_i)$ is averaged over all objects
- This heuristic can be computed for different cluster structures (e.g., different k values), which gives a certain evaluation of goodness of the clustering: the best k can be chosen

Applications of cluster analysis

- Term clustering
- Document clustering
- ...

Algorithms for term clustering

- General process:

  1)**selection of the document set and the vocabulary**: term by document matrix

  2) computation of **term association or similarity matrix**: strength of the associations between terms

  3) **clustering** of related terms

D = document set

p = number of documents in D

n = number of distinct key terms or stems in D

s_{uv} = correlation factor

= the similarity (association, correlation) between term (or stem) u and term (or stem) v

$A_{n\text{x}n}$ = term-term correlation matrix with pair wise term similarities

Selection of the document set and the vocabulary

- Document set: selection of representative document corpus
- Term selection and normalization:
  - terms can be selected from titles, abstracts, or the full text
  - term selection cf. indexing with natural language index terms: stopword removal, selection of phrases, stemming, …
- Result: term by document matrix

Construction of the term association matrix

1. Based on the co-occurrence of terms (or stems) inside documents:
   - association between two terms (s_{uv}) computed with e.g., inner product, Dice coefficient, cosine function of the document vectors of the terms
   - result: A = **association matrix**
   - possible additional constraint for term co-occurrence: term must be present in same sentence, paragraph, or threshold on number of intermediate words

Construction of the term association matrix

2. Based on the co-occurrence of terms (or stems) inside documents and their distance (number of words between them):

   - example of computing the correlation factor:

$$s_{uv} = \frac{\sum_{k_i \in V(s_u)} \sum_{k_j \in V(s_v)} \frac{1}{r(k_i, k_j)}}{|V(s_u)| . |V(s_v)|}$$

$r(k_i,k_j)$ = distance (number of word positions) between two keywords k_i and k_j

(if k_i and k_j are in distinct contexts $r(k_i,k_j) = \infty$)

$V(s_u)$ and $V(s_v)$ = sets of keywords which have s_u and s_v as their respective stems

- result: A = **metric association matrix**

Construction of the term association matrix

3. Based on the occurrence of terms in similar neighborhoods:
   - idea: two terms with similar neighborhoods have some synonymy relationship
   - given:

$$\mathbf{s}_u = (s_{u1}, s_{u2}, \ldots, s_{un})$$

= the vector of all correlation factors of key term (or stem) *u*

$$\mathbf{s}_v = (s_{v1}, s_{v2}, \ldots, s_{vn})$$

= the vector of all correlation factors of key term (or stem) *v*

   - final correlation factor of *u* and *v* is computed as: inner product, cosine, ... of the vectors representing the correlation factors
   - result: *A* = **scalar association matrix**

Construction of the term association matrix

4) Based on **pointwise mutual information** (MI) statistic between two terms *u* an *v* :

$$\mathrm{MI}(u,v) = \log_2 \frac{P(u,v)}{P(u)P(v)}$$

where

$P(u)$ and $P(v)$ = probabilities of occurrence of respective *u* and *v* estimated from the document corpus

$P(u,v)$ = their probability of co-occurrence

if $\mathrm{MI}(u,v) >> 1$: *u* and v have a strong correlation

if $\mathrm{MI}(u,v) \approx 0$: *u* and v have no correlation

$s_{u,v} = \mathrm{MI}(u,v)$

Construction of the term association matrix

5) Based on chi-square value [see Text categorization]

6) Based on log likelihood ratio for a binomial distribution: originally developed for **collocation extraction** (collocation = compound term, usually element of meaning added to the collocation that can not be predicted from the meanings of the composing parts) [Dunning 1999]

Clustering of related terms

- Simple approach:
  - $S_u(n)$ = function which takes the u^{th} row of the association matrix A and returns the largest values s_{uv}, where v varies over the set of n key terms (or stems) and $v \neq u$
    - selection of k largest correlation values
    - selection of correlation value above threshold
- Cluster algorithms:
  - hierarchical algorithms
  - partitioning algorithms
- Related terms are often called **topic signatures**

Query expansion

- = identifying terms that are related to the query terms and add them to the query:
  - by the use of a (hand-built) thesaurus: adding of synonyms, stemming variations
  - expansion with terms that co-occur with the query terms in documents

Query expansion

- **local strategy**:
  - based on co-occurrence of terms in relevant retrieved document set
    - (pseudo-)relevance feedback needed
- **global strategy**:
  - based on co-occurrence of terms in document collection (cf. use of a machine-built thesaurus)
- combination of local and global strategies

Query expansion: results

- Local analysis: improves retrieval effectiveness with (pseudo-)relevance feedback:
  - e.g. metric correlation

- Global analysis:
  - results not conclusive: correlation valid for the corpus might not be valid for the current query

Thesaurus construction: results

- **Global analysis**:
  - acceptable results when learned from a large corpus of texts with a specialized vocabulary (e.g., in limited subject domains)
  - technique is questionable with heterogeneous text databases
- Is co-occurrence information useful?
  - synonyms do not often occur together in the same context, but, tend to share neighbors
  - potential of:
    - scalar association matrix
    - NLP techniques for detecting neighbors (such as syntactic dependencies in phrases, e.g. head-modifier relationships)

Document clustering

- = clustering of document texts based on the features (usually terms) they share:
  - identifying document relationships (e.g., texts that bear on the same topics)
- Applied in:
  - modeling retrieval, text classifications, event detection, text summarization, ...

Document clustering

- General process:
  - **selection of the document set and the vocabulary**: document by term matrix
  - computation of **association or similarity matrix**: strength of the associations between document texts
  - **clustering** of highly related documents

D = document set

n = number of documents in D

p = number of distinct features measured in D

$A_{n \times n}$ = document-document similarity matrix with pair wise document similarities

Selection of the document set and the vocabulary

- Document set: dependent upon application selection of :
  - complete documents
  - sentence, text blocks of fixed length, or paragraphs of individual documents
- Term selection and normalization:
  - terms can be selected from titles, abstracts, or the full text
  - term selection cf. indexing with natural language index terms: stopword removal, selection of phrases, stemming, term weighting, …
- Result: document by term matrix

Construction of the document similarity matrix

- Similarity between two documents computed with e.g., inner product, cosine function, ...

- Result: A = **similarity matrix**, association matrix

Clustering of related documents

- Hierarchical methods:
  - single link
  - complete link
  - group average link
- Partitioning methods:
  - e.g., *k*-means algorithm

Clustering of related documents

- Algorithms must often cope with huge collections:
  - e.g., by clustering a sample of texts and assigning the other texts to the cluster with the most similar centroid or representative object
  - e.g., splitting the collection, clustering each set, clustering all centroids or representative objects found in the complete collection; when in resulting clustering these "centers" occur in the same cluster, merging of corresponding clusters
- Interest in algorithms that:
  - generate a natural clustering without relying upon threshold similarity values or a fixed number of clusters
  - are computationally efficient

Other methods for document clustering

Spectral clustering: the document corpus is seen as an undirected graph, and the task of clustering is to find the best cuts in the graph optimizing certain criterion functions

Clustering based on non-negative matrix factorization of the term by document matrix: each axis captures the base topics of a particular document cluster, and each document is represented as a combination of the base topics, the most important base topic determines the cluster to which a document belongs

Non-negative matrix factorization

- **Non-negative matrix factorization (NMF): a (positive) matrix** is approximately factorized into a product of non-negative factors (i.e., all elements must be ≥ 0) [Lee & Seung 2001]:

$$\mathbf{A}_{pxn} \approx \mathbf{U}_{pxk}\mathbf{V}_{kxn}$$

- Factorization of matrices is generally non-unique:
  - different methods e.g., multiplicative update method, Kullback-Leibler divergence, gradient descent algorithms, alternating non-negative least squares, projected gradient, ...

e.g.,

Minimize $\left\| \mathbf{A} - \mathbf{UV} \right\|^2$ with respect to **U** and **V**, subject to the constraint **U**,**V** ≥ 0

Minimize $D(\mathbf{A} \| \mathbf{UV})$ with respect to **U** and **V**, subject to the constraint **U**,**V** ≥ 0

NMF in text mining

- NMF is used in text clustering:
  - the p x n term-document matrix **A**:
    - contains the term weights
    - is factored into:
      - a term-cluster matrix: $\mathbf{U}_{p \times k}$
      - a **cluster-document matrix** $\mathbf{V}_{k \times n}$
  - e.g., applied on medical publications, emails
- NMF extends beyond matrices to tensors of arbitrary order

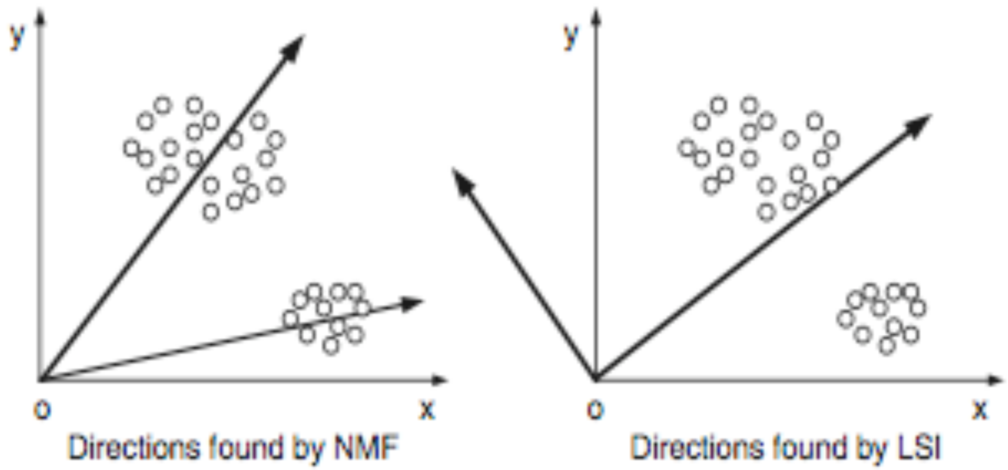


Figure 1: Illustration of the differences between NMF and LSI.

- **Analogy with the LSI**: in interpreting the meaning of the two non-negative matrices **U** and **V**:
  - each element u_{ij} of matrix **U** represents the degree to which term f_i belongs to cluster j
  - each element v_{ij} of matrix **V** indicates to which degree document d_j is associated with cluster i

- **Analogy with pLSA**: when NMF is obtained by minimizing the Kullback Leibler divergence, it is equivalent to pLSA, trained by maximum likelihood estimation [Gaussier & Goutte 2005]

Document clustering with NMF

- Given a document corpus, construct the term-document matrix **A** in which column *j* represents the weighted term-frequency vector of document d_j
- Perform the NMF on **A** to obtain the two non-negative matrices **U** and **V**
- Normalize **U** and **V**
- Use matrix **V** to determine the cluster label of each document: examine each row *i* of matrix **V** and assign document d_j to the cluster corresponding with row *x* where $x = \arg\max_i v_{ij}$

Document clustering: some applications and their results

- Cluster retrieval model
- Topic overviews
- Clustering retrieval output
- Event detection
- Text segmentation and summarization:
  - clustering of sentences, paragraphs and fixed text blocks [see Text summarization]

Cluster retrieval model

- Variant of the the vector space model:
  - similar documents are grouped in a cluster
  - for each cluster, a representation is made (e.g., centroid)
  - query is matched against cluster centroids:
    - in case of a **partition**: against the centroid of each cluster: when condition is satisfied (e.g. minimum similarity threshold)
      - all documents in the cluster are returned as the result
    - in case of a **hierarchy**: tree is processed downward, taking the highest scoring branch, until some stopping condition (e.g. minimum similarity threshold) is met:
      - subtree at that point is returned as the result

Cluster retrieval model

- Assumes **cluster hypothesis** [van Rijsbergen, 1979]: documents similar in content tend to be relevant to the same queries

- Results:
  - efficiency at retrieval time increases
  - best results with the hierarchical complete link (small collections) and group average (large collections) algorithms
  - increases recall

Topic overviews

- = generating topical overview of a complete document collection
  - clustering of the documents
  - Visualization of clusters: 2- (e.g., rings or glyphs) or 3- (e.g., spheres) dimensional displays
    - might indicate similarity between documents
  - clusters are labeled with:
    - document titles
    - main key terms: e.g.,
      - most frequent term(s) of the cluster
      - term with the highest weight in the centroid of the cluster, ...

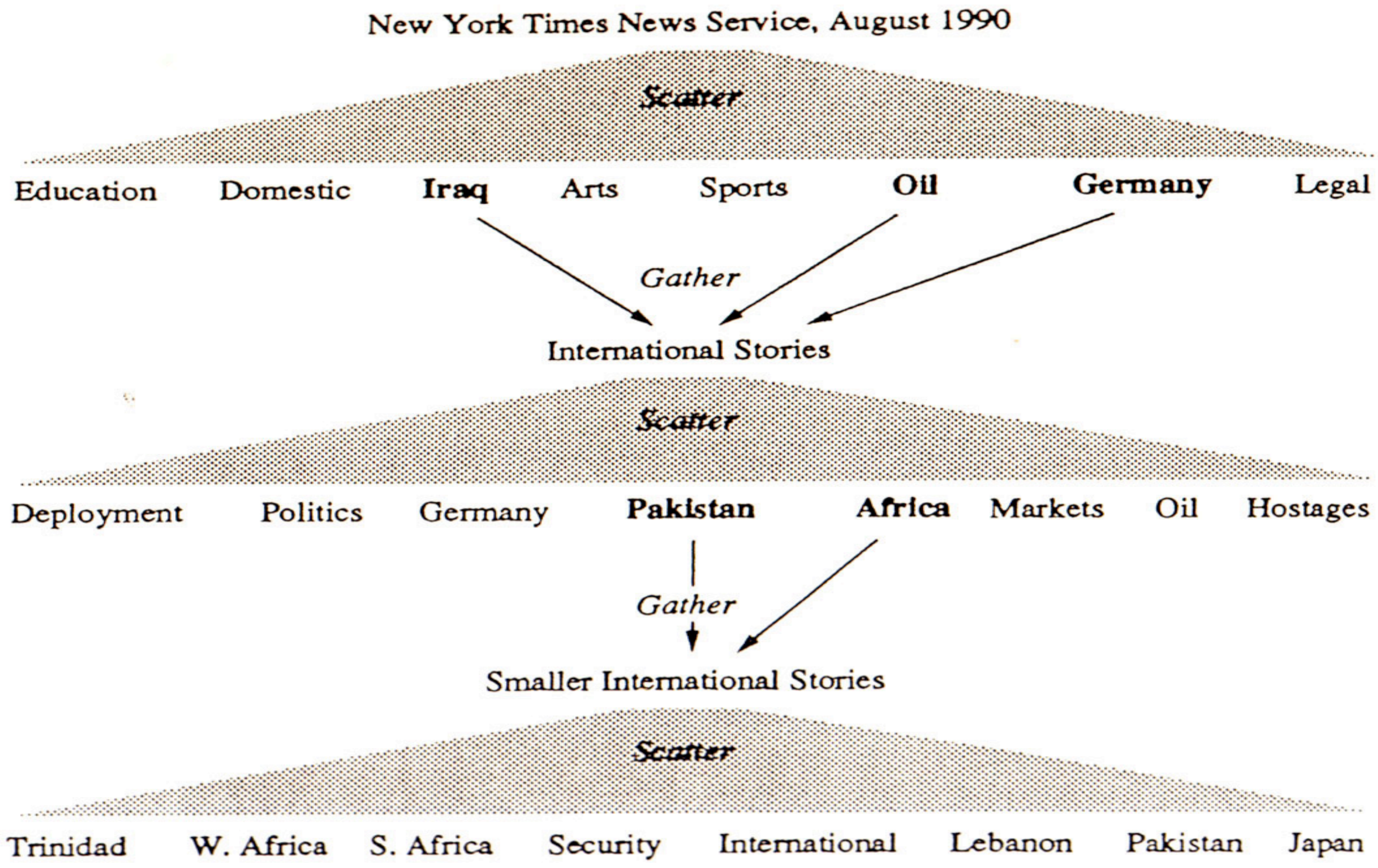


Figure 1: Illustration of Scatter/Gather

[Cutting et al. 1992]

Scatter/Gather system

- **Interface for browsing** a document collection (Cutting et al. 1992):
  - initially, the user is presented the content of *k* large clusters that represent the broad topics of the collection ("scatter" documents)
  - user can "gather" the contents of one or more clusters and system "rescatters" this document subset to form *k* new clusters
  - process can be iterated several times
  - generates: overview of topic and topic combinations at different levels

Scatter/Gather system: algorithms

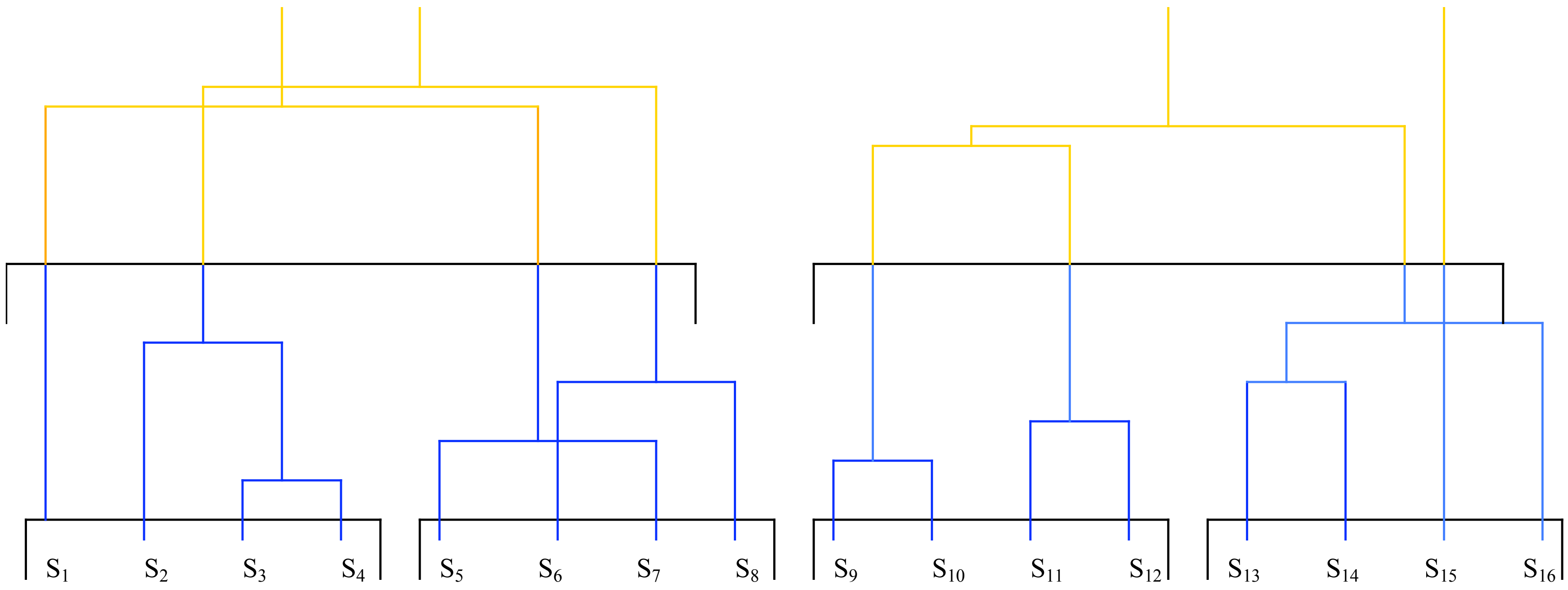
- **Initial clustering**: hierarchical group average cluster algorithm (GAC):
  1. start where each document = singleton cluster (initial partition)
  2. divide the current partition into non-overlapping and consecutive buckets of fixed size of *m* (*m* <= *n*) documents
  3. apply GAC to each bucket until the bucket size is reduced by a pre-determined factor ρ
  4. remove the bucket boundaries
  5. repeat steps 2-4 until *k* top-level clusters are obtained in the final partition

  time complexity : $O(mn)$

Scatter/Gather system: algorithms

- to improve the initial clustering: variation of the *k-means algorithm:* in a few iterative steps: documents are re-assigned to the most similar cluster centroid and cluster centroids are recomputed

- Subsequent clustering steps: **on-line**: must be very **efficient**:
  - random initial partitioning in *k* clusters of the selected subset of documents
  - improved by variation of *k*-means algorithm in few steps



$m = 4$

$\rho = 0.5$

$k = 4$

Topic overviews

- Results:
  - in not too heterogeneous or not too homogeneous document collection:
    - useful for identifying comprehensible themes at different levels of detail
  - but, not very accurate
  - maps are difficult to read for users:
    - labels (topic descriptions) are helpful
    - but, can not always be correctly extracted from the clusters

Clustering of retrieval results

- Cluster hypothesis also holds in retrieved set of documents
- Technique: cf. topic maps of document collection
  - cluster a long list of retrieved documents into clusters of related content
  - find relevant documents with minimum effort:
    - pick object of a cluster and discard rest of the cluster if object is not relevant
  - e.g., Scatter/Gather system
- Results: cf. topic maps of document collection
  - TREC conferences: retrieval results tend to cluster in 1 or 2 clusters with relevant documents, besides clusters of other documents

Event detection

- = automatically detect novel events from a temporally ordered stream of documents (e.g., news stories):
  - **retrospective detection**: discovery of previously not notified events in an existing collection
  - **on-line detection**: discovery of new events in document stream in real time (e.g., life news feeds)
- Document clustering:
  - based on content (lexical similarity) and temporal proximity:
    - detection of groups of related events within same time frame

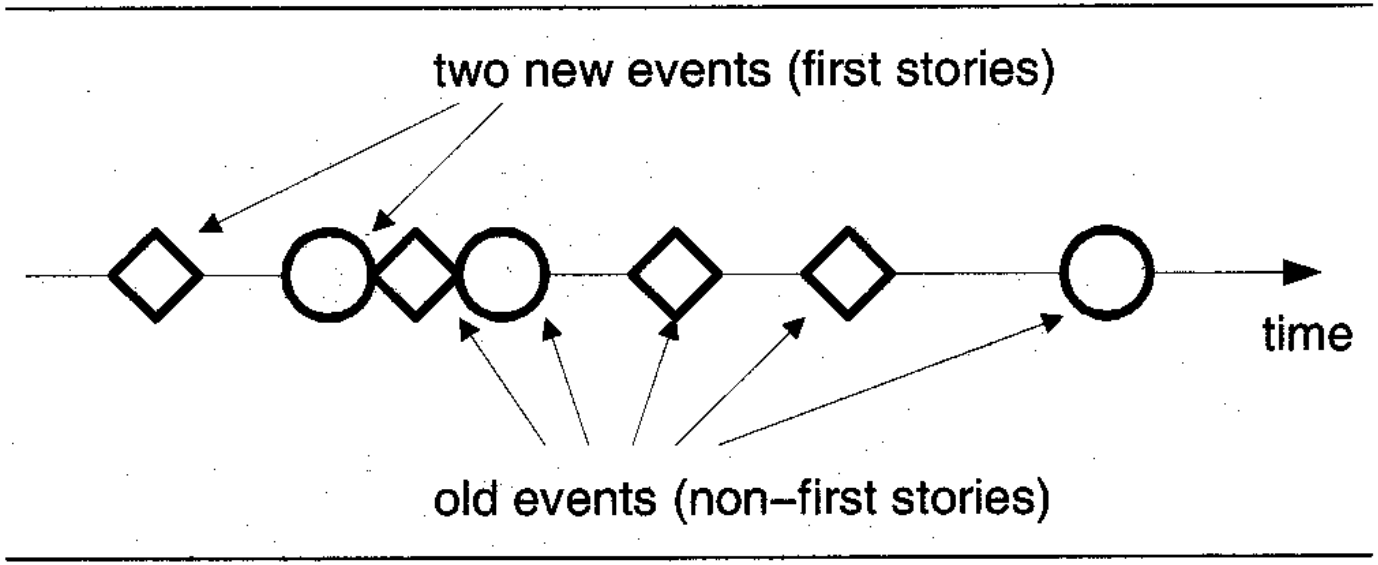


Figure 1: New Event Detection in a stream of news stories. Two different events are marked by diamonds and cirlces. The first story on each event is to be flagged.

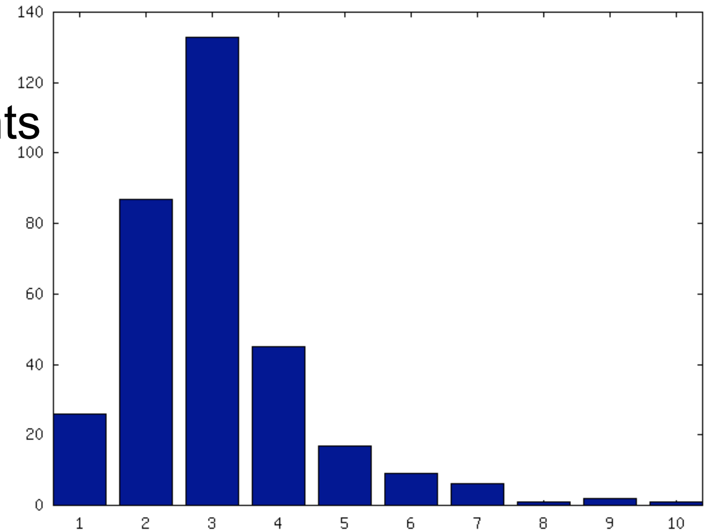
[Brants et al. 2003]

Event detection

- Extensive study on Wikinews (1000 documents covering 237 events): vector representations, probabilistic topic and named entity models, probabilistic event model, where an event generates named entities and other words, ...

Event distribution

Number of events

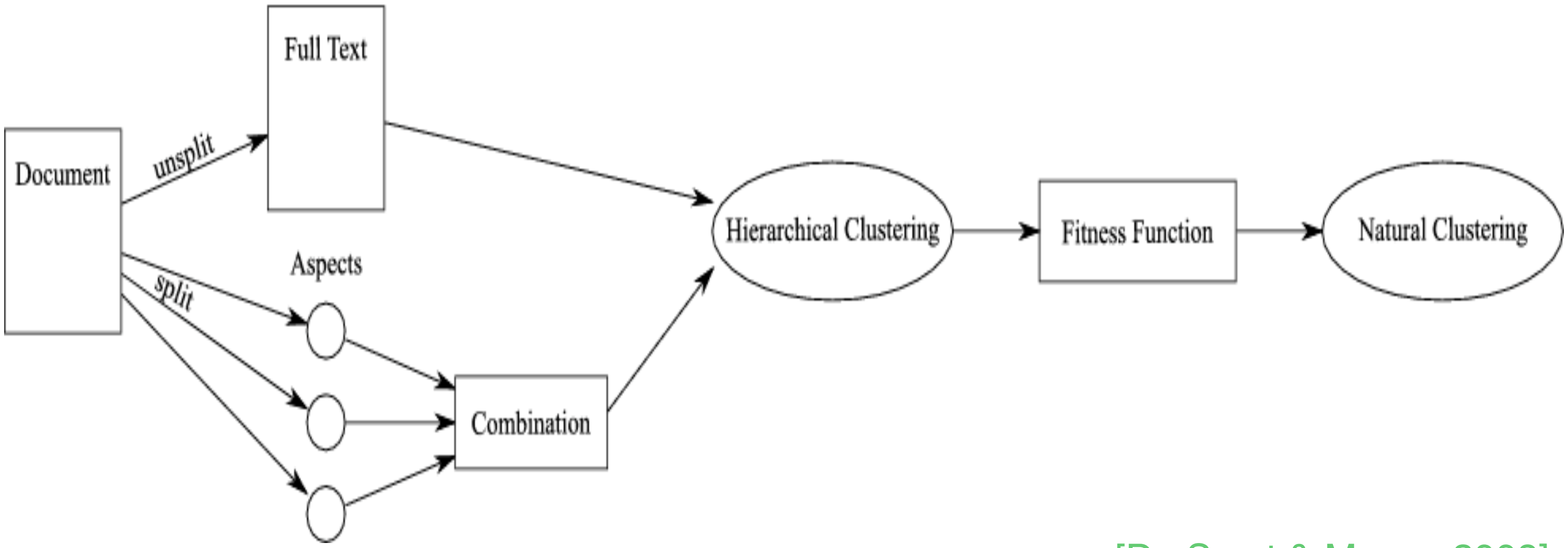


Number of documents

[De Smet & Moens 2008]

Event detection

- Best results:
  - split of content in named entities (persons, locations, organizations) and other words (named entity recognition [see Information extraction])
  - forming two separate vectors
  - cosine similarity based on named entity vectors and vectors with other words
  - fusion model of the similarities: by taking the maximum of the similarities



[De Smet & Moens 2008]

Event detection

[De Smet & Moens 2008]

	Natural clustering				Maximum result			
	P	R	F1	# events	P	R	F1	# events
Full text	64.5%	95.0%	76.8%	208	87.6%	88.7%	88.1%	335
Words	50.4%	93.8%	65.6%	162	88.2%	84.5%	86.3%	364
Entities	49.1%	84.8%	62.1%	173	72.7%	74.8%	73.7%	341
max	80.1%	92.2%	**85.7%**	271	90.2%	87.8%	89.0%	346
average	12.3%	94.5%	67.3%	164	87.1%	87.1%	87.1%	340

Table 1 2-way split for the vector space model

Hierarchical complete link clustering:
- Natural clustering: based on average fitness value (slides 34 and 35)
- Maximum result: search manually for a similarity threshold value, when used as stop criterion, yields best results in F_1 value

Results in terms of B-cubed precision and recall computed for each object document *i* and averaged given ground truth cluster M_i and machine-generated cluster C_i to which *i* belongs:

$$precision = \frac{|C_i \cap M_i|}{|C_i|}$$

$$recall = \frac{|C_i \cap M_i|}{|M_i|}$$

What have we learned?

- Clustering: unsupervised learning:
  - advantage: algorithms do not need a training corpus that is manually annotated or classified
  - this lack of information often leads to lesser quality results than a domain-savvy supervised approach: clusters obtained may not correspond to classifications that are a priori meaningful
  - many different approaches
- Term clustering: **acquisition of ontology**
- Document clustering:
  - in past and present many applications (e.g., **event detection** and **linking, clustering of retrieval results**)

Research questions to be solved

- Better methods for automatically recognizing term relationships: e.g., hypernymy, hyponymy, meronymy
- Similarity methods (e.g., kernels) for comparing advanced text representations
- Scalability when confronted with large data sets
- Methods to describe the obtained term or document clusters
- Clustering of heterogeneous data (e.g., text content, image content, link data)

Further reading

Bao, L., Tang, S., Li J., Zhang, Y. and Ye, W. (2008). Document clustering based on spectral clustering and non-negative matrix factorization. In *New Frontiers in Applied Artificial Intelligence* (pp. 149-158). Berlin: Springer.

Brants, T., Chen, F. & Farahat A. (2003). A system for new event detection. In *Proceedings of the Twenty-sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 330-337). New York: ACM.

Cutting, D.R., Karger, D.R., Pedersen, J.O. & Tukey, J.W. (1992). Scatter/Gather: a cluster-based approach to browsing large document collections. In N.J. Belkin, P. Ingwersen, & A.M. Pejtersen (Eds.), *Proceedings of the Fifteenth SIGIR Conference* (pp. 318-329). New York: ACM.

De Smet, W. & Moens, M.-F. (2008). A study on multi-document event clustering. Technical Report K.U.Leuven.

Gaussier, E. & Goutte, C. (2005). Relation between PLSA and NMF and its implications. *Proc. 28th international ACM SIGIR conference on Research and development in information retrieval (SIGIR-05)* (pp. 601−602). New York: ACM.

Kaufman, L. & Rousseeuw, P.J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. New York: John Wiley & Sons.

Lee, D.D. & Seung, H.S. (2001). Algorihms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13, 556-562.

Van Rijsbergen, C.J. (1979). *Information Retrieval* (2nd ed.). London: Butterworths.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing* 17(4): 395-416.

Voorhees, E.M. (1986). Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Information Processing & Management*, 22 (6), 465-476.

Xu, W. & Liu, X. (2003). Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th Annual InIternational ACM SIGIR Conference on Research and Development in Informaion Retrieval* (pp. 267-273). New York: ACM.

Vivisimo clustering engine: http://vivisimo.com used for the Clusty search engine: http://clusty.com/

Kartoo search engine: http://kartoo.com/