

## Text Mining

*Amin Mantrach & Nicolas van Zeebroeck (IRIDIA)*

### Overview

- Technological and algorithmic foundations
  - Pre-processing
  - Representation
  - Text Analysis
  - Evaluation

## Text Mining Defined

- The process of deriving high quality information from text (*Wikipedia*)
- Usually involves
  - structuring the input text (pre-processing & representation)
  - deriving patterns within the structured data (analysis)
  - evaluation and interpretation of the output (evaluation)
- Typical tasks:
  - Categorization, clustering, concept/entity extraction, production of taxonomies, sentiment analysis, document summarization, and entity relation modeling

ULB

ULB

# Text Mining

## Technological & Algorithmic Foundations

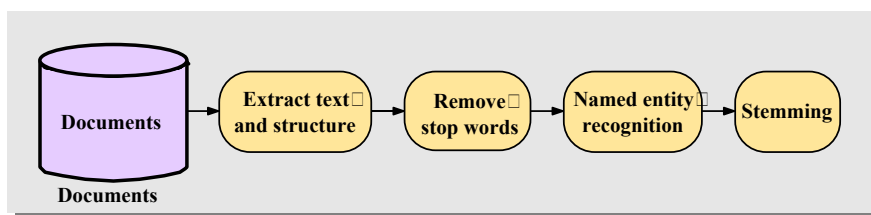
## 1. Pre-processing

- We have a collection of documents
- Here are the standard pre-processing steps
  - Extract text and structure  
(eg. from Microsoft Word, HTML pages or LaTeX to XML)
  - Clean characters and encoding
  - Remove stop words (eg. remove "the", "at", "all", etc)
  - Named entity recognition (eg. find proper names)
  - Stemming (eg. extract "process" from "processing")
  - Part-of-Speech Tagging

ULB

## 1. Pre-processing

- This is a tedious job!



- But tools are readily available, eg.:
  - Galilei (ULB)
  - Weka (Wakaiki)
  - Text to Matrix Generator (Matlab toolbox)

ULB

## 1. Pre-processing

- Stemming aims to extract the « root » of the words
- Stemming can be based on
  - A dictionary
    - Eg: Mmorph developed at the University of Geneva
  - A set of rules developed by linguists
    - Eg: Porter's stemming algorithm for English (translations exist in different languages)
- Example: what is the stem (or root) of “(I) was”
  - With Porter: “WA”
  - With dictionary: “TO BE”

ULB

## 1. Pre-processing

- Part of Speech Tagging (POS or POST)
  - The process of marking up the words in a text as corresponding to a particular part of speech
    - Eg: the identification of words as nouns, verbs, adjectives, adverbs, etc.
  - Based on both its definition, as well as its context
    - i.e., relationship with adjacent and related words in a phrase, sentence, or paragraph.
  - Performed using algorithms which associate discrete terms, as well as hidden parts of speech, in accordance with a set of descriptive tags
  - Available toolboxes include
    - TreeTagger

ULB

## 2. Representation

- In its basic form, each document is represented by a vector → **Vector Space Model**
- The coordinates of the vector are words
  - Each element of the vector represents the frequency of the word in the document or the query
  - In the space of words

ULB

## 2. Representation

- Thus a document is represented by a vector
  - Document  $j$  is characterized  $\mathbf{d}_j$
  - $f_{ij}$  is the frequency of word  $w_i$  in document  $j$
  - The total number of words is  $n_w$
- The dimension of the vector is  $n_w$

ULB

## 2. Representation

- Thus each document is represented by

$$\mathbf{d}_j \triangleq \begin{bmatrix} f_{1j} \\ f_{2j} \\ \vdots \\ f_{n_w j} \end{bmatrix}$$

- This is called the « **bag of words** » representation in the words space
  - The order of the words is not taken into account
  - This vector is usually sparse
  - This vector is very large

ULB

## 2. Representation

- The total number of documents is  $n_d$
- The terms-documents matrix is

$$\mathbf{D} \triangleq \left[ \begin{array}{cccc} & \text{documents} & & \\ f_{11} & f_{12} & \cdots & f_{1n_d} \\ f_{21} & f_{22} & \cdots & f_{2n_d} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n_w 1} & f_{n_w 2} & \cdots & f_{n_w n_d} \end{array} \right] \left. \vphantom{\begin{array}{c} \\ \\ \\ \\ \end{array}} \right\} \text{words}$$

ULB

## 2. Representation

- Term weighting
  - Of course, each word does not have the same « weight »
  - We would like to take account of the "discriminative power" of every word
  - For instance, if a word is present in every document, it is useless
  - $P(w_i)$  is the a priori probability that word  $w_i$  appears in a document

ULB

## 2. Representation

- This quantity is often called the inverse document frequency (idf) associated to word  $w_i$ :  $idf_i = -\log_2[P(w_i)]$ 
  - It is a measure of the general importance of the word (or term)  $w_i$
- It is estimated by taking the logarithm of
  - the number of documents in which  $w_i$  appears, divided by the total number of documents

ULB

## 2. Representation

- Another quantity of interest is the term frequency,  $tf_{ij}$

$$tf_{ij} = \frac{f_{ij}}{\sum_{i=1}^{n_w} f_{ij}}$$

- It measures the importance of the term  $w_i$  within the particular document  $d_j$
- It is normalized to prevent a bias towards longer documents

ULB

## 2. Representation

- The tf-idf score is simply the product of the tf and the idf scores,  $tf.idf_{ij} = idf_i . tf_{ij}$ 
  - The tf-idf weighting scheme is often used in the vector space model
- By replacing the term-frequency elements of the terms-documents matrix by the tf-idf scores

ULB

## 2. Representation

- Document – Term Matrix

- N Documents x K Terms

- Different definitions for  $d_{ij}$  :

- Binary: doc i contains term j or not
- Frequency:  $f_{ij}$  = number of occurrences of term j in doc i
- Generalized approach:  $d_{ij} = t_{ij} \cdot g_j \cdot s_i$  where
  - $t_{ij}$  = term weighting component, which only depends on  $f_{ij}$
  - $g_j$  = global weighting component, which depends on  $d_j$
  - $d_j$  = number of documents containing term j
  - $s_i$  = normalization component for  $d_i$

- Particular case:  $TFIDF_{ij} = f_{ij} \times \log\left(\frac{N}{d_j}\right)$

- Normalized TFN:

- Each doc vector lies on surface unit sphere
- Retains only proportion of words in documents

→ Documents with similar words but different lengths lead to similar vectors

$$D = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1K} \\ d_{21} & d_{22} & \cdots & d_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N1} & d_{N2} & \cdots & d_{NK} \end{bmatrix}$$

$$TFN_{ij} = \frac{f_{ij} \times \log\left(\frac{N}{d_j}\right)}{\sqrt{\sum_{j=1}^K \left(f_{ij} \times \log\left(\frac{N}{d_j}\right)\right)^2}}$$

ULB

## 2. Representation

- How to reduce the size of the matrix?

- Feature selection:

- Eliminate non content-bearing “high-frequency” & “low-frequency” terms
- In practice:
  - Keep only TOP x% of most frequent terms after dropping stop words
  - Local (document by document) v. Global (over entire collection)

- Information bottleneck

- Use training set to compute Information Gain indexes

- Dependence between terms and categories

- Term clustering: replace words by coherent clusters of words

- Less sensitive to synonymy, homonymy and polysemy
- But requires background information on categories  
Lewis, 1992; Baker & McCallum, 1998

- Latent Semantic Indexing (LSI)

- Also improves when categories information is used
- Ideally one LSI space for each category

ULB

## 2. Representation

- Latent semantic models
  - These models try to capture some semantic information
  - For instance, if we introduce a query with "newborn", it would be nice if documents containing "baby" but not "newborn" are also retrieved
  - We say that words are semantically related when they are used in the same context
  - This way, we can capture some « semantic similarity » between words
  - In the present case, we will say that two words are semantically related when they often occur in the same document

ULB

## 2. Representation

- One solution to this problem is to use "sub-space projection methods" like
  - "Singular Value Decomposition" (SVD) or
  - "factor analysis"
- The rank  $m$  SVD of a matrix of rank  $n$  is the « best approximation » to this matrix having rank  $m < n$ 
  - In the present case, we use a SVD in order to reduce the rank of the term-document matrix

ULB

## 2. Representation

- This allows to reduce the dimensionality of the space by clustering the words that are semantically "similar",
- That is, used in the same documents
- This allows us to build a kind of concept space

ULB

## 2. Representation

- Every matrix has a "singular value decomposition":

$$\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}$$

where  $\mathbf{U}^T\mathbf{U} = \mathbf{I}$  and  $\mathbf{V}^T\mathbf{V} = \mathbf{I}$

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n \end{bmatrix}$$

with  $\sigma_1 > \sigma_2 > \dots > \sigma_n > 0$

ULB

## 2. Representation

- If we want the best rank- $m$  approximation to  $D$ , we put

$$\sigma_{m+1} = 0, \sigma_{m+2} = 0, \dots, \sigma_n = 0$$

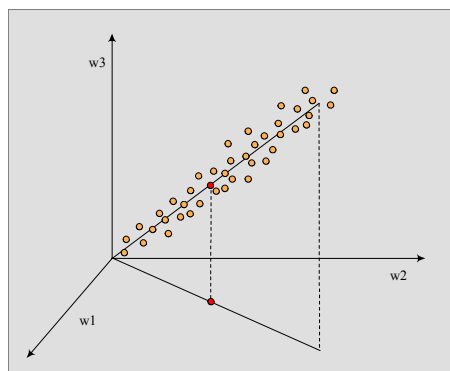
- So that we obtain

$$\tilde{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & \sigma_2 & 0 & & & & 0 \\ \vdots & 0 & \ddots & \ddots & & & \vdots \\ \vdots & & \ddots & \sigma_m & 0 & & \vdots \\ \vdots & & & 0 & 0 & \ddots & \vdots \\ \vdots & & & & & \ddots & \ddots & 0 \\ 0 & 0 & \dots & \dots & \dots & 0 & 0 \end{bmatrix}$$

ULB

## 2. Representation

- $\tilde{D}$  is the best rank- $m$  approximation to  $D$ 
  - That is, there is no rank- $m$  matrix closer to  $D$  in terms of the Frobenius norm  $\tilde{D} = U\tilde{\Sigma}V$



ULB

## 2. Representation

- The vector-space method relies on linear algebra concepts
- The SVD approach allows to work in a latent space representing concepts
- The main problem:
  - How many dimensions of the subspace do we keep?

ULB

## 3. Analysis

- Overview
  - Information retrieval
  - Classification
  - Clustering

ULB

### 3.1. Analysis > Information Retrieval

- We have a collection of documents
  - Mainly text or html-based
- We have a set of users
- A user wants to retrieve the documents related to a given concept
- He consequently submits a query expressed through words or terms
- An information retrieval system returns the documents most related to this concept

ULB

### 3.1. Analysis > Information Retrieval

- A query is also represented by a vector
  - Here is a query  $q$
  - Each element is 0 or 1 (presence or absence of a word)

$$q \triangleq \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad i \rightarrow \text{word } w_i \text{ is present in the query}$$

ULB

### 3.1. Analysis > Information Retrieval

- In case of a weighted document-term matrix
  - We redefine the query vector  $q$  as

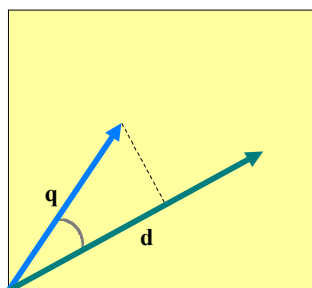
$$\mathbf{q} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -\log_2 [P(w_i)] \\ 0 \\ \vdots \\ 0 \end{bmatrix} i$$

- Each word  $w_i$  is weighted by the information provided by knowing the presence of the word

ULB

### 3.1. Analysis > Information Retrieval

- The purpose is of course to retrieve documents  $d_i$  based on a query  $q$
- We have to define a notion of similarity between a query and a document



ULB

### 3.1. Analysis > Information Retrieval

- The similarity between a query  $q$  and a document  $d_i$  can be defined as

- The cosinus of the angle between these two vectors:

$$\text{sim}(\mathbf{q}, \mathbf{d}_i) \triangleq \cos(\mathbf{q}, \mathbf{d}_i) = \frac{\mathbf{q}^T \mathbf{d}_i}{\|\mathbf{q}\| \|\mathbf{d}_i\|}$$

- Euclidean distance does not work well because queries contain much lesser words than documents
- It is called the cosine similarity

ULB

### 3.1. Analysis > Information Retrieval

- The similarity between the query and all documents can be computed by using the term-document matrix

$$\begin{aligned} \cos(\mathbf{q}, \mathbf{D}) &= \frac{\mathbf{q}^T}{\|\mathbf{q}\|} \mathbf{D} \text{diag} \left[ \frac{1}{\|\mathbf{d}_i\|} \right] \\ &= \frac{\mathbf{q}^T}{\|\mathbf{q}\|} [ \mathbf{d}_1 \quad \dots \quad \mathbf{d}_i \quad \dots \quad \mathbf{d}_{n_d} ] \text{diag} \left[ \frac{1}{\|\mathbf{d}_i\|} \right] \\ &= \left[ \frac{\mathbf{q}^T \mathbf{d}_1}{\|\mathbf{q}\|} \quad \dots \quad \frac{\mathbf{q}^T \mathbf{d}_i}{\|\mathbf{q}\|} \quad \dots \quad \frac{\mathbf{q}^T \mathbf{d}_{n_d}}{\|\mathbf{q}\|} \right] \text{diag} \left[ \frac{1}{\|\mathbf{d}_i\|} \right] \\ &= \left[ \frac{\mathbf{q}^T \mathbf{d}_1}{\|\mathbf{q}\| \|\mathbf{d}_1\|} \quad \dots \quad \frac{\mathbf{q}^T \mathbf{d}_i}{\|\mathbf{q}\| \|\mathbf{d}_i\|} \quad \dots \quad \frac{\mathbf{q}^T \mathbf{d}_{n_d}}{\|\mathbf{q}\| \|\mathbf{d}_{n_d}\|} \right] \end{aligned}$$

ULB

## 3.2. Analysis > Classification

- Given a collection of records (documents or terms) (training set)
  - Each record contains a set of features, one of the attributes is the class
- Find a model for predicting the class feature
  - as a function of the values of other features
- Previously unseen records should be assigned a class as accurately as possible
- Text Mining applications :
  - Automatic Classification of Documents
  - Named Entity Recognition

ULB

## 3.2. Analysis > Classification

### Four different steps:

1. Building the classification models from a sample or data set for which the values of both the features and the dependent variable are known
  - This is the training set
2. Comparison of the performance of the different models on an independent data set: a validation set that was not used for building the models
  - Cross-validation or bootstrapping are usually used

ULB

### 3.2. Analysis > Classification

3. Assessment of the performance of the best model, on an independent data set: a test or validation set that was not used for building or comparing the models
  - evaluate the accuracy of the model (prediction error, precision/recall,...)
  - Allow to take a decision for putting the model into production
4. Application of the best model to new instances during the production phase

ULB

### 3.2. Analysis > Classification

- The Naïve Bayes model is a statistical classifier, which performs probabilistic prediction, i.e. predicts:
  - Class membership probabilities
  - Then a posteriori probabilities
- Based on Bayes' theorem
- Generative model
- Incremental
  - Each training example can incrementally increase/decrease the probability that a hypothesis is correct
  - Prior knowledge can be combined with observed data

ULB

## 3.2. Analysis > Classification

- Bayes' Theorem: Basics
  - Let  $x$  be a feature vector (“evidence”)
    - Class label is unknown
  - Let  $Y = W_k$  be the hypothesis that  $x$  belongs to class  $W_k$
  - We must determine  $P(W_k|x)$ : probability that record belongs to class  $W_k$  given the observed data sample  $x$
  - $P(W_k)$  = Prior Probability = the initial probability
    - E.g.:  $x$  will buy computer, regardless of age, income, ...
  - $P(x)$ : Probability that sample data  $x$  is observed
  - $P(x|W_k)$  = within-class density = likelihood: Probability of observing sample  $x$ , given that the sample belongs to class  $W_k$ 
    - E.g.: given that the individual will buy a computer, the probability that  $x$  has a medium income

ULB

## 3.2. Analysis > Classification

- Bayes' Theorem: Basics
  - Given training data  $x$ , the a posteriori probability of a class  $W_k$ ,  $P(W_k|x)$ , follows from Bayes' theorem:
 
$$P(W_k | x) = \frac{P(x | W_k)P(W_k)}{P(x)}$$
  - Informally, this can be written as
    - A Posteriori = Likelihood\_Ratio w Prio\_Evidence
  - Predicts  $w$  belongs to  $W_k$  iff the probability  $P(W_k|x)$  is the highest among all the  $P(W_i|w)$  for all classes
    - This is the Bayes decision rule (does not include costs)
  - Practical difficulty: requires initial knowledge of many multivariate probability distributions

ULB

## 3.2. Analysis > Classification

- Towards Naïve Bayes classifier
  - Let T be a training set of feature vectors and their associated class labels
    - Each feature vector is represented by  $\{x_1, x_2, \dots, x_p\}^T$
  - Suppose there are q classes  $\{W_1, W_2, \dots, W_q\}$
  - Classification is based on maximum a posteriori =  $P(W_k | x)$
  - From Bayes' Theorem:

$$P(W_k | x) = \frac{P(x | W_k)P(W_k)}{P(x)}$$

- Since  $P(x)$  is constant for all classes, only

$$P(W_k | x) \propto P(x | W_k)P(W_k)$$

needs to be maximized

ULB

## 3.2. Analysis > Classification

- Derivation of Naïve Bayes classifier
  - An assumption: features are class-conditionally independent
    - I.e.: no dependence relation between the p features

$$P(\mathbf{x} | \omega_k) = \prod_{i=1}^p P(x_i | \omega_k) = P(x_1 | \omega_k) \times P(x_2 | \omega_k) \times \dots \times P(x_p | \omega_k)$$

$$\text{Log}[P(\mathbf{x} | \omega_k)] = \sum_{i=1}^p \text{Log}[P(x_i | \omega_k)] = \text{Log}[P(x_1 | \omega_k)] + \dots + \text{Log}[P(x_p | \omega_k)]$$

- This greatly simplifies the problem
  - We only have to estimate the univariate within-class distributions:  $P(x_i | W_k)$

ULB

## 3.2. Analysis > Classification

- Estimation of within-class probabilities
  - If  $X_i$  is categorical,  $P(X_i=x|W_k)$  can be estimated by
    - The # samples in  $W_k$  having value  $x$  for  $X_i$  divided by # samples in  $W_k$
  - If  $X_i$  is continuous-valued,  $P(X_i=x|W_k)$  can be computed based on a parametric density, e.g. Gaussian distribution:

$$P(x_i | \omega_k) = g(x_i, \mu_k, \sigma_k) \quad g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

ULB

## 3.2. Analysis > Classification

- Training dataset (sample)

Class:

C1:buys\_computer = 'yes'

C2:buys\_computer = 'no'

Data sample:

$x = [\text{age} \leq 30,$

Income = medium,

Student = yes

Credit\_rating = Fair]<sup>T</sup>

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

ULB

## 3.2. Analysis > Classification

- Classifier example

- $P(C_k)$ :  $P(\text{buys\_computer} = \text{"yes"}) = 9/14 = 0.643$   
 $P(\text{buys\_computer} = \text{"no"}) = 5/14 = 0.357$
    - Compute  $P(\mathbf{x}|C_k)$  for each class; thus, first compute the  $P(x_i|C_k)$ 
      - $P(\text{age} = \text{"<=30"} | \text{buys\_computer} = \text{"yes"}) = 2/9 = 0.222$
      - $P(\text{age} = \text{"<= 30"} | \text{buys\_computer} = \text{"no"}) = 3/5 = 0.6$
      - $P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"yes"}) = 4/9 = 0.444$
      - $P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$
      - $P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$
      - $P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"no"}) = 1/5 = 0.2$
      - $P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$
      - $P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$
    - $\mathbf{x} = [\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit\_rating} = \text{fair}]^T$
    - $P(\mathbf{x}|C_k)$  :
      - $P(\mathbf{x}|\text{buys\_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$
      - $P(\mathbf{x}|\text{buys\_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$
    - $P(\mathbf{x}|C_k) \times P(C_k)$  :
      - $P(\mathbf{x}|\text{buys\_computer} = \text{"yes"}) \times P(\text{buys\_computer} = \text{"yes"}) = 0.028$
      - $P(\mathbf{x}|\text{buys\_computer} = \text{"no"}) \times P(\text{buys\_computer} = \text{"no"}) = 0.007$
- ⇒ Therefore,  $\mathbf{x}$  belongs to class ("buys\_computer = yes")

ULB

## 3.3. Analysis > Clustering

- Aim
  - Non supervised classification
  - Groups documents according to similarity measures
  - Can structure groups into hierarchy
  - Essentially: an optimization problem

ULB

### 3.3. Analysis > Clustering

#### Similarity measures

- Aim
  - Provide measures of distance between documents
- Approaches
  - Minkowski metric:  $D_p(d_i, d_j) = \left( \sum_k (d_{ik} - d_{jk})^p \right)^{1/p}$
  - Euclidian distance: special case with  $p=2$
  - Cosine similarity:  $Sim(d_i, d_j) = (d_i \cdot d_j) = \sum_k d'_{ik} \cdot d'_{jk}$ 
    - where  $d'$  is the normalized vector  $d = d / |d|$
  - Coherence (goodness) of cluster  $C_j$ :  $\sum_{d_i \in C_j} d_i^T \bar{c}_j$ 
    - Where centroid of cluster  $C_j$ :  $\bar{c}_j = \frac{\sum_{d_i \in C_j} d_i}{\left\| \sum_{d_i \in C_j} d_i \right\|}$

ULB

### 3.3. Analysis > Clustering

- Approaches
  - Partitional v. hierarchical: groups are disjoint or nested
  - Hard v. soft: Docs belong to one (hard) or more (soft) groups (with fractional degree of belonging)
  - But unrealistic from computation point of view → Approximate
    - Agglomerative:
      - Start with 1 doc = 1 cluster
      - Merge clusters until stopping criterion is reached
    - Divisive:
      - Start with all docs = 1 cluster
      - Split clusters until stopping criterion is reached
    - Shuffling:
      - Iteratively redistribute objects into clusters
    - Expectation maximization (EM):
      - Estimate probability distributions (mixture-resolving)

ULB

### 3.3. Analysis > Clustering

		FLAT/PARTITIONAL	HIERARCHICAL
HARD	Agglomerative		
	Divisive		
	Shuffling	k-means Spherical k-means Co-clustering Nearest neighbour	
	Probabilistic		
SOFT (FUZZY)	Agglomerative		HAC
	Divisive		
	Shuffling	Fuzzy Co-clustering	
	Probabilistic		EM-based mixture resolving PLSA

ULB

### 3.3. Analysis > Clustering

- k-Means

- Maximizes quality function:  $Q(C_1, C_2, \dots, C_k) = \sum_{j=1}^k \sum_{d_i \in C_j} d_i^T \bar{c}_j$

- Initialization

- Provide  $k$  seeds (docs) as core of  $k$  initial cluster
  - Randomly or externally
- Assign each other document to closest cluster (seed)

- Iteration

- Compute centroid of each cluster  $C_j$   $\bar{c}_j = \frac{\sum_{d_i \in C_j} d_i}{\left\| \sum_{d_i \in C_j} d_i \right\|}$

- Reassign each document to cluster with closest centroid

- Stopping condition

- Stable solution is reached (no more changes occur)

- Pitfalls

- Strongly depends on initial seeds → Several runs or external seeds

ULB

### 3.3. Analysis > Clustering

- EM-based probabilistic clustering (mixture-resolving)
  - Approach:
    - Clusters are drawn from k-distributions
    - Identify parameters of each distribution to compute  $P(C_i/d)$
  - Initialization
    - Select initial parameters of each distribution
      - Randomly or externally
  - Iteration
    - Expectation-Step: Compute  $P(C_i/d)$  for each document using current parameters and relabel documents according to computed probabilities
    - Maximization-Step: Reestimate all parameters to maximize likelihood while assuming current labels are correct
  - Stopping condition
    - Convergence in Maximum Likelihood estimation

ULB

### 3.3. Analysis > Clustering

- Probabilistic Latent Semantic Indexing (PLSI)
  - Characteristics:
    - Based on Factor Analysis
    - Stronger statistical foundations than traditional SVD
    - Factor representation can deal with polysemous words and explicitly distinguish between different meanings and types of word usage
  - Approach:
    - Construct Aspect model: unobserved class variable for each observation
    - Estimate E-M model
    - Documents are not assigned to clusters but characterized by specific mixture of factors with weights  $P(z/d)$
    - Generates a reduced vector space based on this mixture decomposition

ULB

### 3.3. Analysis > Clustering

- Probabilistic Latent Semantic Indexing (PLSI)
  - Aspect model: Latent variable model which associates unobserved class variable  $z$  with each observation (=occurrence of a word  $w$  in a document  $d$ )
    - Select document  $d$  with probability  $P(d)$
    - Pick latent class  $z$  with probability  $P(z|d)$
    - Generate word  $w$  with probability  $P(w|z)$
  - Obtain observed pair  $(d,w)$  while latent class variable  $z$  is discarded

$$P(d, w) = P(d)P(w|d) = P(d)\sum_{z \in Z} P(w|z)P(z|d)$$

ULB

### 3.3. Analysis > Clustering

- Probabilistic Latent Semantic Indexing (PLSI)
  - E-Step:
 
$$P(z|d, w) = \frac{P(z)P(d|z)P(w|z)}{\sum_{z' \in Z} P(z')P(d|z')P(w|z')}$$
    - Probability that word  $w$  in document  $d$  is explained by factor  $z$
  - M-Step: Maximize
 
$$\zeta = \sum_{d \in D} \sum_{w \in W} n(d, w) \log P(d, w)$$
    - Where  $n(d,w)$  is the frequency of term  $w$  in document  $d$

ULB

### 3.3. Analysis > Clustering

- Probabilistic Latent Semantic Indexing (PLSI)

- Is equivalent to reduction of vector space with

$$P = \hat{U}\hat{D}\hat{V}^T$$

$$\hat{U} = (P(d_i | z_k))_{i,k}$$

$$\hat{D} = \text{diag}(P(z_k))_k$$

$$\hat{V} = (P(w_j | z_k))_{j,k}$$

- The eigenvectors in SVD correspond to the factors  $P(w|z)$  and the component distributions  $P(d|z)$  of the aspect model
- The singular values of SVD corresponds to the mixing proportions  $P(z)$

ULB

### 3.3. Analysis > Clustering

- Spherical k-means

- Based on cosine similarity and normalized TFN

- Doc and concept vectors lie on surface of high-dimensional sphere

- Maximizes quality function:  $Q(C_1, C_2, \dots, C_k) = \sum_{j=1}^k \sum_{d_i \in C_j} d_i^T \bar{c}_j$

- Initialization

- Provide  $k$  seeds (docs) as core of  $k$  initial cluster
  - Randomly or externally
- Assign each other document to cluster with closest centroid (seed)

- Iteration

- Recompute centroid of each cluster  $C_j$
- Reassign each document to cluster with closest centroid
- Renormalize concept vectors based on new partitioning:  $s_j = \sum_{d_i \in C_j} d_i$

- Stopping condition

- Stable solution is reached (no more changes occur)

- Pitfalls

- Strongly depends on initial seeds → Several runs or external seeds

ULB

### 3.3. Analysis > Clustering

- Co-clustering / Bi-clustering

- Find sub-matrices of D whose elements are highly coherent
- Documents are clustered wrt only some words that belong to same co-cluster

- ideally, these words are most relevant

$$D = \begin{bmatrix} 1 & 1 & 0.2 & 0.4 \\ 1.1 & 1.2 & 0 & 0.1 \\ 0 & 0.1 & 2.3 & 2.2 \\ 0 & 0 & 2.3 & 2.3 \end{bmatrix}$$

- Fuzzy co-clustering

- Boundary between 2 co-clusters is fuzzy
- Allows any doc or word to belong to more than one cluster
- 3 components:
  - Degree of aggregation
  - Constraints
  - Fuzzifier

ULB

### 3.3. Analysis > Clustering

- Co-clustering / Bi-clustering

- Degree of aggregation  $\sum_{c=1}^C \sum_{i=1}^N \sum_{j=1}^K u_{ci} v_{cj} d_{ij}$  where

- $u_{ci}$  = membership of document i in cluster c
- $v_{cj}$  = membership of word j in cluster c
- $d_{ij}$  = correlation between document i and word j

- Constraints  $\begin{cases} \sum_{c=1}^C u_{ci} = 1 \\ \sum_{j=1}^K v_{cj} = 1 \end{cases}$  for  $1 \leq i \leq N$  and  $1 \leq c \leq C$

- Fuzzifier

- Fuzzy Entropy  $-T_u \sum_{c=1}^C \sum_{i=1}^N u_{ci} \ln u_{ci} - T_v \sum_{c=1}^C \sum_{j=1}^K v_{cj} \ln v_{cj}$

- Fuzzy Gini  $-T_u \sum_{c=1}^C \sum_{i=1}^N u_{ci}^2 - T_v \sum_{c=1}^C \sum_{j=1}^K v_{cj}^2$

- Single Term Fuzzifier  $T \sum_{c=1}^C \sum_{i=1}^N \sum_{j=1}^K [(u_{ci} + v_{cj}) - u_{ci} v_{cj}]^2$

ULB

### 3.3. Analysis > Clustering

- Co-clustering

- Objective function to maximize

$$\sum_{c=1}^C \sum_{i=1}^N \sum_{j=1}^K u_{ci} v_{cj} d_{ij} + \text{fuzzifier} + \sum_{i=1}^N \lambda_i \left( \sum_{c=1}^C u_{ci} - 1 \right) + \sum_{c=1}^C \lambda_c \left( \sum_{j=1}^K v_{cj} - 1 \right)$$

- First partial derivatives of objective function provide update equations for membership functions  $u$  and  $v$
- Initialization
  - Set parameters  $C, T_u, T_v$
  - Randomly initialize  $u_{ci}$
  - Compute  $v_{cj}$
- Iteration
  - Update  $u$  and  $v$  alternatively
- Stopping condition
  - Convergence is achieved
  - Final values of  $u_{ci}$  and  $v_{cj}$  represent membership of document and words in each co-cluster

ULB

### 3.3. Analysis > Clustering

- Hierarchical Agglomerative Clustering (HAC)

- Approach:
  - Start with each document = 1 cluster and merge closest clusters
- Initialization
  - Consider each document is in a separate cluster
- Iteration
  - Find the pair of most similar clusters and merge them together
  - Similarity approaches
    - Single link: Maximum similarity between pairs of docs from 2 clusters
    - Complete link: Minimum similarity between pairs of docs from 2 clusters
    - Average link: Average similarity between pairs of docs from 2 clusters
    - Centre of gravity: distance between centroids of clusters
    - Ward Algorithm: Increase in intra-class inertia when 2 clusters are merged
- Stopping condition
  - Everything is merged into one single cluster
  - History of mergers provides binary tree of the clusters hierarchy

ULB

### 3.3. Analysis > Clustering

- Additional approaches
  - HAC + k-Means
    - HAC used to cluster  $\sqrt{kn}$  documents
    - Serve as seeds for k-Means algorithm
  - Nearest neighbour
    - Assigns each document to cluster of nearest (labelled) neighbour (provided distance with neighbour is below predefined threshold)
  - Graph-theoretic clustering
    - Construct Minimal Spanning Tree (MST) of documents
    - Delete edges with largest lengths
  - To organize clusters into hierarchy
    - Genetic Algorithms

ULB

### 3.3. Analysis > Clustering

#### Data Abstraction

- Aim:
  - Find meaningful and concise descriptor (label) for each cluster
  - Very small number of terms distinguishing clusters
- Approaches:
  - Centroid / Medoid document(s) title
  - Most frequent terms in cluster
    - Typically 5 or 10 most frequent terms in centroid vector of each cluster
  - Ideally: distinctive noun phrase

ULB

## 4. Evaluation

- In general, we compute two measures:
  - The precision
  - The recall
- As well as the F-measure

ULB

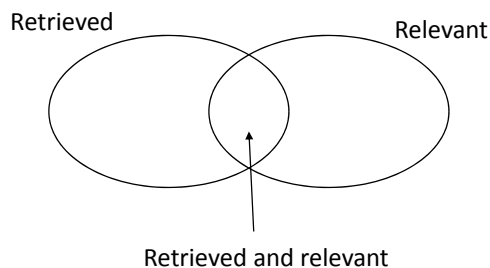
## 4. Evaluation

- The precision measure estimates the percentage of relevant retrieved documents in the set of all retrieved documents
  - Precision indicates to which extent the retrieved documents are indeed relevant
- The recall measure estimates the percentage of relevant retrieved documents in the set of all relevant documents
  - Recall indicates to which extent the relevant documents are indeed retrieved

ULB

## 4. Evaluation

- Assessment of documents retrieval systems



- There is a trade-off between precision and recall
- The F-measure, taking both precision and recall is
  - $F = 2 (\text{precision} \times \text{recall}) / (\text{precision} + \text{recall})$

ULB

## 4. Evaluation

- There is a trade-off between precision and recall
- The F-measure, taking both precision and recall is

$$F = 2 (\text{precision} \times \text{recall}) / (\text{precision} + \text{recall})$$

ULB

64