Automatic Data Analysis in Visual Analytics –
Selected Methods

Multimedia Information Systems 2 VU
(SS 2015, 707.025)

Vedran Sabol

Know-Center

March 15th, 2016
Lecture Overview

- Visual Analytics Overview
- Knowledge Discovery in Databases (KDD)
- Steps in the KDD chain
- Selected KDD methods for
  - Feature engineering
  - Clustering
  - Classification
  - Association Modelling
Visual Analytics Overview
Motivation

- In the Web we are dealing with:
  - Huge amounts of data (PBs and more)
  - Heterogeneous information (structures, content, semantic data, numeric data...)
  - Dynamic data sets (fast growth/change rates)
  - Uncertain, incomplete and conflicting information (quality)

→ Abundance of complex data which contains hidden knowledge

How understand and utilize our data?
  - Unveil implicitly present knowledge
  - Enable explorative analysis
Motivation

• Machines can crunch through huge amounts of data
  ▪ Getting better and faster (Moore’s law)
• Nevertheless, they are still behind humans in
  ▪ Identification of complex patterns and relationships
  ▪ Knowledge and experience
  ▪ Abstract thinking
  ▪ Intuition
  ▪ …
• Human visual system is a extremely efficient processing “machine”
  ▪ Still unbeatable in recognition of complex patterns
Visual Analytics

- A new interdisciplinary research area at the crossroads of
  - Data mining and knowledge discovery
  - Data, information and knowledge visualisation
  - Perceptual and cognitive sciences
- Human in the loop
Visual Analytics

- Combines automatic methods with interactive visualisation to get the best of both [Keim 2008]
  - interaction between humans and machines through visual interfaces to derive new knowledge
Visual Analytics

1. Machines perform the initial analysis
2. Visualization presents the data and analysis results
3. Humans are integrated in the analytical process through means for explorative analysis
   • User spots patterns and makes a hypothesis about the data
   • Further analysis steps - visual and/or automatic - to verify the hypothesis
   • Confirmed or rejected hypothesis: new knowledge!

Today’s lecture will focus on the first step
Knowledge Discovery
Knowledge Discovery Process

- Knowledge Discovery Process [Fayyad, 1996]
  - A chain of data processing and analysis steps
  - Goal: discovery of new, relevant, previously unknown patterns in data
Knowledge Discovery Process

• KDD is the non-trivial process of identifying valid, novel, potentially useful and understandable patterns in data.
• A set of various activities for making sense out of data
  ▪ Data is a set of facts
  ▪ Pattern discovery and data mining designates fitting a model to data, finding structure from data, finding a high-level description of data
  ▪ Quality of patterns depends on their validity, novelty, usefulness and simplicity
Knowledge Discovery Process

• Knowledge discovery refers to the entire process, of which **knowledge is the end-product**
  - Interactive (user interpretation, steering the process)
  - Iterative (provide feedback, refine results and reuse them for further analysis)
• All steps are necessary to ensure that the process produces useful knowledge
• Data mining is a crucial step in this process: applying data analysis algorithms that produce/identify patterns
Knowledge Discovery Process

Data Selection

- Gathering and selecting data which is to become the subject of further knowledge discovery steps
- Retrieving data from one or more databases or a digital libraries
  - Comparably simple: execute a query, retrieve a data subset
- Crawling: collect resources from the Web
Knowledge Discovery Process
Data Selection

• Complex: focused crawling
  ▪ Follow the Web link structure and retrieve resources
  ▪ Depending on specific properties
    • E.g. domains, timeliness, page rank, topics (complex!) etc.
  ▪ Prioritize links to follow first
    • depending on how well the resource satisfies the criteria

• Result of the data selection step: target data is available for analysis
Knowledge Discovery Process

Data Preprocessing

• Filtering, cleaning and normalising the selected data
• Filter out data which does not qualify for further processing
  ▪ Missing necessary information
  ▪ Duplicate data
  ▪ Unnecessary data (overhead)
  ▪ Identify and remove contradictory or obviously incorrect information
• Basic cleaning operations
  ▪ Handling missing data fields (e.g. meaningful defaults)
  ▪ Removal of noise (can be complex)
Knowledge Discovery Process
Data Preprocessing

• Normalizing data: bringing the data to a common denominator
  ▪ Convert different formats to a single one
    • Text (e.g. PDF, HTML, Word...)
    • Images (PNG, TIFF, JPEG...)
    • Audio/Video
    • ...
  ▪ Time information: convert different date formats
  ▪ Person data: name + surname or vice-versa
  ▪ Geo-spatial references: convert names to latitude and longitude
  ▪ Metadata harmonization
Knowledge Discovery Process
Data Transformation

• Raw data cannot be processed by data mining algorithms
• Transform the data into a form such that data mining algorithms can be applied
  ▪ Depends on the goal
  ▪ Depends on the applied algorithms
• Feature engineering: find useful features to represent the data
  • E.g. for text: meaning bearing words, such as nouns
  • But not stopwords (and, or, the...)
• Feature: individual measurable property of a phenomenon being observed
Knowledge Discovery Process

Data Transformation

• Feature examples
  ▪ Images: color histograms, textures, contours...
  ▪ Signals: amplitude, frequency, phase, distribution...
  ▪ Time series: ticks, intervals, trends...
  ▪ Graphs: neighboring nodes, weight and type of relationships
  ▪ Text: words, key terms and phrases, part-of-speech tags, named entities, grammatical dependencies, ...
Knowledge Discovery Process
Data Transformation

• Feature types
  • Numeric: continuous (e.g. time), discrete (e.g. count, occurrence)
  • Categorical: nominal (e.g. gender), ordinal (e.g. rating)
  • Linguistic (e.g. terms with POS tags)
  • Structural (e.g. parent-child)
Knowledge Discovery Process
Data Transformation

• Feature engineering
  ▪ Feature extraction: identify useful features to represent the data
  ▪ Feature transformation: reduce the number of variables under consideration (e.g. using dimensionality reduction)
  ▪ Feature selection: discard unnecessary features or features with low information content

• Feature engineering is crucial for data mining methods
  ▪ Garbage in – garbage out

• We will focus on text and graph data
Knowledge Discovery Process

Data Mining

- Data mining: discovering patterns of interest in a particular representational form
  - e.g. classification rules, cluster partition...
- Research area at the intersection of artificial intelligence, machine learning and statistics
- Represents the analytical step in the KDD chain
Knowledge Discovery Process

Data Mining

- Classes of data mining methods
  - Outlier detection (anomaly detection)
  - Summarization
  - Classification
  - Clustering
  - Association modelling (relationship extraction)
  - ...

Vedran Sabol (KTI/TU Graz, Know-Center)
Knowledge Discovery Process
Data Mining

• Outlier detection: identification of data elements which are not related to any other elements
  ▪ Out of range/erroneous measurements, topically unrelated text documents, unconnected graph elements...
  ▪ May be valuable: identify errors
• Summarization: computation of a compressed representation for one or multiple data elements
  ▪ Document: sentences with the highest information content in a document
  ▪ Document collections: most common words
Knowledge Discovery Process

Data Mining

• Classification: assign an example into a given set of categories
  ▪ Supervised machine learning technique
  ▪ Training (model fitting): learn a labeled set of training examples
    • Data elements belong to known classes
  ▪ Identify to which classes previously unseen examples belong to
    • Using the trained model
  ▪ Probabilistic and rule based approaches are common
  ▪ Applications: spam detection, sentiment analysis, topical categorization...
• Clustering: Identify groups of related (similar) data elements
  • Unsupervised learning technique: no pre-defined classes, no training
  • Criteria: maximize similarity within clusters, minimize similarity between clusters
  • Exclusive vs. inclusive clustering: each data element belongs to one vs. multiple clusters
  • Fuzzy clustering: assignment weights (instead of binary values)
  • Hierarchical vs. flat clustering: cluster hierarchy vs. one level of clusters
  • Incremental vs. non incremental: new elements incorporated to existing partition vs. computing partition from scratch
Knowledge Discovery Process
Association Modelling (Relation Extraction)

• Discovering of relations between variables in data
• For text: discovery of relationships between concepts (terms)
  ▪ E.g. depending on their co-occurrence
    • i.e. how often terms are mentioned together (in documents, in paragraphs or sentences)
    • Relationship has a weight but not a quality (relation type is undefined)
    • Example: person and company are often mentioned together → it is likely that they are associated in some way
  ▪ Extraction of relationship quality
    • Using natural language processing methods and pattern matching
      – E.g. <subject, verb, object> patterns
    • Lookup in WordNet lexical database
      – Synonyms, hyponyms/hypernyms, troponyms, meronyms, antonyms...
Knowledge Discovery Process
Presentation and Interpretation

• Interpretation of the discovered patterns
• Involves users in the process
  ▪ Intuition, knowledge, abstract thinking, visual pattern discovery…
• Use of visualisation
  ▪ Present discovered patterns in an easy to understand way
  ▪ Present data in a way that enables human visual system to discover additional patterns
  ▪ Interactive exploration of data and patterns
• Feedback
  ▪ Utilize human knowledge and abstract thinking capabilities
  ▪ Improve performance of the algorithms
• Iterative discovery process

Will be the topic of the next lectures
Feature Engineering
Feature Engineering
Text

- Identify features describing the content of some text
- Natural language processing (NLP) methods
  - Tokenisation: terms (words, bi-words, word n-grams)
  - Sentence detection and part-of-speech (POS) tagging: nouns, verbs, adjectives, prepositions...
  - Named entity recognition (NER): organizations, persons, locations, dates...
  - Stemming: reduce words to root form
  - Case folding
  - Stopword filtering
  - ...

“Organized by government, services of commemoration are being held in Germany to mark the end of World War I in 1918. ...”
Vector Space Model
Text - Bag of Words

- Each document represented as a feature vector
  - Features are dimensions of the vector space
    - For text these are the terms
  - Weight: frequency of the term in the document

- Examples:
  - \(d_1\): “Services of commemoration are being held around the world to mark the end of World War I in 1918. ...”
  - \(d_2\): “World War I (abbreviated as WW-I, WWI, or WW1), also known as the First World War ...”
  - \(d_3\): “We offer world wide service”

<table>
<thead>
<tr>
<th></th>
<th>service</th>
<th>commemor</th>
<th>world</th>
<th>end</th>
<th>war</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d_1)</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(d_2)</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>(d_3)</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Vector Space Model

Weighting

<table>
<thead>
<tr>
<th></th>
<th>servic</th>
<th>commemor</th>
<th>world</th>
<th>end</th>
<th>war</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>d2</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>d3</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TF/IDF term weighting

- Boost terms which are not common in the corpus $D$
- Reflects importance of a term $t$ for a document $d$
- Increases discrimination power of term vectors

TF/IDF Weighting

- Term Frequency (TF): frequency of term $t$ in document $d$
- Inverse Document Frequency (IDF) in corpus $D$:

$$idf(t, d) = \log \frac{|D|}{|\{d : t \in d\}|}$$

<table>
<thead>
<tr>
<th></th>
<th>servic</th>
<th>commemor</th>
<th>world</th>
<th>end</th>
<th>war</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>0,405</td>
<td>1,099</td>
<td>0</td>
<td>1,099</td>
<td>0,405</td>
</tr>
<tr>
<td>d2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0,81</td>
</tr>
<tr>
<td>d3</td>
<td>0,405</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, d, D)$$
Vector Space Model
Graph Vectorising

- Describe a node using its neighbourhood
  - Features: the IDs of a node’s neighbour nodes
  - Weights: close neighbours (with small amount of edges) get more weight
    - Weight = $1/(2^{\text{shortest connecting path length} - 1})$
      - Neighbour weight falls exponentially with its distance
    - Optional: divide weight by the neighbour’s edge count
      - Nodes connected to many other nodes have little discriminative power
  - Propagate only a fixed number of hops
    - e.g. threshold 3 – 5 hops

- Does not support weighted graph
  - Edge weights could be included in the computation
Vector Space Model

Graph Vectorising - Example

In this example: two hop Neighbourhood considered
A one hop neighbourhood closely an adjacency matrix

\[ \tilde{A} = [(B,1), (C,0.333), (D,0.25), (E,0.125)] \]
Similarity/Distance Computation
Similarity and Distance Metrics

- Computes similarity or distance between a pair of vectors
  - Needed by many data mining methods
- $k$ represents the index of the vector space dimensions
- $w_{n,k}$ is the weight of the $k$-th feature of the $n$-th data element
- Euclidean distance between high-dimensional vectors ([0,inf.])

$$\text{dist}(\vec{d}_i, \vec{d}_j) = \sqrt{\sum_{k} (w_{i,k} - w_{j,k})^2}$$

- Manhattan (city-block or taxi-cab) distance

$$\text{dist}(\vec{d}_i, \vec{d}_j) = \sum_{k} |w_{i,k} - w_{j,k}|$$
Similarity Metrics

- Cosine similarity - the angle between vectors \([0,1]\)

\[
sim(d_i, d_j) = \frac{d_j \cdot d_i}{|d_j| \cdot |d_i|} = \frac{\sum_{\forall k} w_{j,k} \cdot w_{i,k}}{\sqrt{\sum_{\forall k} w_{j,k}^2} \cdot \sqrt{\sum_{\forall k} w_{i,k}^2}}
\]

- Jaccard coefficient, Dice coefficient, ...
Distance Matrix

- **Distance matrix**
  - E.g. for cars represented by multiple dimensions
    - Engine displacement, power, weight, fuel consumption, dimensions, number of cylinders, price …

- Normalise the dimensions ([0,1] space)

- Compute pairwise distances

### Similarity or Distance Matrix

<table>
<thead>
<tr>
<th></th>
<th>Opel</th>
<th>VW</th>
<th>Suzuki</th>
<th>Toyota</th>
<th>Mercedes</th>
<th>BMW</th>
<th>Ferrari</th>
<th>Porsche</th>
<th>Lamborghini</th>
<th>Rolls Royce</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opel</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VW</td>
<td>3,1</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suzuki</td>
<td>5,0</td>
<td>4,4</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Toyota</td>
<td>3,8</td>
<td>3,3</td>
<td>3,7</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mercedes</td>
<td>5,9</td>
<td>5,8</td>
<td>7,0</td>
<td>5,3</td>
<td>2,7</td>
<td>6,8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMW</td>
<td>5,5</td>
<td>5,5</td>
<td>7,0</td>
<td>4,2</td>
<td>2,7</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ferrari</td>
<td>8,4</td>
<td>8,1</td>
<td>8,3</td>
<td>8,3</td>
<td>6,9</td>
<td>6,8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Porsche</td>
<td>8,4</td>
<td>8,1</td>
<td>8,4</td>
<td>8,3</td>
<td>6,4</td>
<td>6,4</td>
<td>3,0</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lamborghini</td>
<td>8,5</td>
<td>8,2</td>
<td>8,8</td>
<td>8,7</td>
<td>6,6</td>
<td>6,4</td>
<td>2,1</td>
<td>3,4</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Rolls Royce</td>
<td>8,5</td>
<td>8,6</td>
<td>8,9</td>
<td>8,2</td>
<td>5,8</td>
<td>7,0</td>
<td>6,6</td>
<td>6,8</td>
<td>6,3</td>
<td>-</td>
</tr>
</tbody>
</table>
Clustering
Clustering

- Aggregation: structure the data space into coarser entities – clusters
- Clustering algorithms
  - Partitional methods
    - **K-means**, k-medoids, fuzzy k-means
  - Hierarchical methods
    - Agglomerative (bottom-up), divisive (top-down)
  - Density-based clustering (DBSCAN)
  - ... (many others)
- Unsupervised learning
  - Applied only on unlabeled data
  - No pre-defined classes, no training
Clustering
Definition

- Grouping data elements by similarity
  - Data points in cluster are more similar to each other than to data points in other clusters
- Given a set of data points
  \[ X = \{< x_1 >, < x_2 >, \ldots, < x_{n-1} >, < x_n >\} \]
- Find groups \( C_1 \) to \( C_k \) \((k < n)\) which optimize criteria
  - Between Cluster Criterion: Minimize similarity of data elements from different clusters
  - Within Cluster Criterion: Maximize similarity within one cluster
K-means/medoids Clustering

- Given: \( n \) data elements, number of clusters \( k \)

- **Overview of the algorithm**
  1. Seeding: choose \( k \) data elements, use them as cluster representatives
  2. Compute similarity of data elements to cluster representatives
  3. Assign each data element to the most similar cluster
  4. Update cluster representatives for all clusters:
     - K-Means: compute centroids by adding cluster’s data element feature vectors
     - K-Medoids: choose a new medoid that minimises a cost function
  5. Go to point 2 unless
     - no data points move between the clusters or
     - iteration count has reached a predefined threshold

- **Converges to a local minimum**
  - A few iterations (e.g. 5) over data set usually sufficient
K-means Clustering

Disadvantages

- Sensitive to the choice of seeds
  - Heuristic: maximize the distance between initial seeds
  - Combine with other algorithms
    - Buckshot: apply hierarchical agglomerative clustering on a small sample of the data to compute seeds
K-means Clustering

Disadvantages

- $k$ must be known in advance
  - Guess $k$ through cluster splitting and merging
  - Given min and max values for $k$
    - Split a (large) cluster if cohesion (e.g. inner-cluster average similarity) of new clusters improves significantly
    - Merge a pair of (small) clusters if the resulting cluster still has high cohesion

- Creates hyperspherical clusters
  - May underperform in low-dim spaces
  - E.g. for elongated clusters
K-means Clustering

Complexity

- Similarity between a pair of vectors: $O(m)$
  - $m$ being dimensionality of the vector space
- Assigning $n$ documents to $k$ clusters: $O(kn)$ similarity computations
- Centroid computation: $O(nm)$
  - each data element added to one centroid
- When $I$ iterations necessary: $O(Iknm)$
- When $I$, $k$, $m$ constant: $O(n)$
  - Scales comparably well
Hierarchical Clustering

- Creates a tree structure
- Top-down hierarchical clustering
  - Example: recursive application of a partitional method (K-Means)
  - Balancing strategy to prevent hierarchy degeneration
    - Similarity penalty for large clusters
- Bottom up: Hierarchical Agglomerative Clustering
  - Assign each data element to one cluster \( c \)
  - Merge the most similar cluster pair
  - Keep merging until desired number of clusters is left
  - Hierarchical structure is useful
    - Coarse-grained view of the whole data space
    - Navigate top-down along the hierarchy to a finer-grained view
    - Useful for visualization: e.g. Level of Detail (LOD) rendering
Hierarchical Agglomerative Clustering
Linkage Strategies

- Strategies for merging clusters
  - Centroid: clusters with most similar centroids
  - Single Link: minimal distance between a pair of clusters
  - Complete Link: maximum distance between a pair of clusters
  - Average Link: average distance between a pair of clusters
Hierarchical Agglomerative Clustering

Single Link

\[ sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y) \]

- Maximum pair wise element similarity
  - Chaining Effect
  - Can find elongated clusters
Hierarchical Agglomerative Clustering

Complete Link

\[ \text{sim}(c_i, c_j) = \min_{x \in c_i, y \in c_j} \text{sim}(x, y) \]

- Minimum pairwise element similarity
  - Favours dense, spherical clusters
  - Sensitive to outliers
Hierarchical Agglomerative Clustering
Group Average Linking

- Average similarity between all pairs of data elements (including pairs from the same cluster)
- Compromise between Single & Complete Link
  - No chaining effects
  - No excessive outlier sensitivity

\[
sim(c_i, c_j) = \frac{1}{|c_i \cup c_j|(|c_i \cup c_j| - 1)} \sum_{\tilde{x} \in (c_i \cup c_j)} \sum_{\tilde{y} \in (c_i \cup c_j): \tilde{y} \neq \tilde{x}} \sim(\tilde{x}, \tilde{y})
\]
Hierarchical Agglomerative Clustering

Complexity

- Computation of pair-wise similarities: $O(n^2)$
- Up to $n-2$ merging steps: brute-force approach: $O(n^3)$
- Optimizations exist:
  - If similarity between a new cluster and all the other clusters be computed in constant time: $O(n^2)$
    - For single link (SLINK) and complete link (CLINK)
  - $O(n^2 \times \log n)$ for Group Average
- Do not scale well
  - Complete Link and Group Average viable for e.g. clustering of small graphs
Summarization
Summarization
Cluster Labeling

- Need cluster labels: interpretation by the users
- Textual description (title) of a distinct data element (medoid)
- Most important features of a cluster centroid - keywords
  - Centroid-Heuristic: 5-10 features with the highest weights
  - Discriminative vs. descriptive labels
    - Documents on computers: “computer” appears in each cluster label
      » Descriptive but useless for discriminating between clusters
    - Use features discriminating between data points
      » Appearing only in a fraction of data points (TFIDF)
  - Visualisation: tag clouds
    - Overview of most important keywords
    - Filtering by selecting keywords
Clustering and Cluster Summarization

Application

- Browsing data collections
  - Apply clustering recursively to compute a hierarchy
  - Labeled hierarchy as “virtual table of contents”
Classification
Classification

- Assigning data points to predefined classes (categories)
  - Supervised learning

- First phase: learning
  - Using labelled training data
  - Assignment of each data point to a category is known
  - A model is fitted to the training data

- Second phase: classification of previously unseen data
  - Using the trained model

- Classifier examples
  - Nearest centroid (Rocchio)
  - K nearest neighbours (knn)
  - Decision trees
  - ... (many others)
Classification

- K nearest neighbours
- Learning: adding data points to categories
  - Extremely lightweight (lazy learning): all computation differed to classification
  - Model consists only of class assignments
- Classification of a new data point
  - Find k (e.g. 4 or 5) nearest neighbours
  - Winner class contains most hits
- Disadvantage: problems with skewed class distribution ($|C_m| \gg |C_n|$)
  - Better chances for a larger class to contain more nearest neighbours
  - Can be addressed by considering distance/similarity to nearest neighbours

$\text{k = 3, red point classified to the white class}$
Classification

- Rocchio (nearest centroids) classifier
- Vectors weighted using TFIDF
- Learning: compute centroid vectors for each class
- Classification of a new data point
  - Compute similarity to each class
  - Winner class is the most similar one
Relationship Extraction
### Relationship Extraction

- **Term document matrix**
  
  $A = \begin{pmatrix} T_1 & T_2 & \ldots & T_t \\ D_1 & w_{11} & w_{12} & \ldots & w_{1t} \\ D_2 & w_{21} & w_{22} & \ldots & w_{2t} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ D_n & w_{n1} & w_{n2} & \ldots & w_{nt} \end{pmatrix}$

- **Term co-occurrence matrix**
  
  - Expresses the association between terms
  - Depending on their co-occurrence in documents

  \[ C = A^T A \]

- **Scalability problem: many documents, very many terms**
  
  - huge matrices
Relationship Extraction

- Matrix size reduction through feature selection
  - Remove terms occurring in
    - in a large proportion of documents
    - in a very small amount of documents
  - Consider only terms which are close to each other in the text
    - Weighting depending on distance between terms in the text
    - Cut-off threshold (e.g. 10 terms)

- Efficient implementation: double inverted index
  - Term-to-document + document-to-term
  - Retrieve weighted associations between any two terms/entities
Relationship Extraction
Application

- Navigation in association networks
  - Explore relationships between persons, organisations, places, topics...
Thank you!

Next lecture (12.04.2016): Practicals Tutorial and Project Presentation

!!! Attendance highly recommended !!!