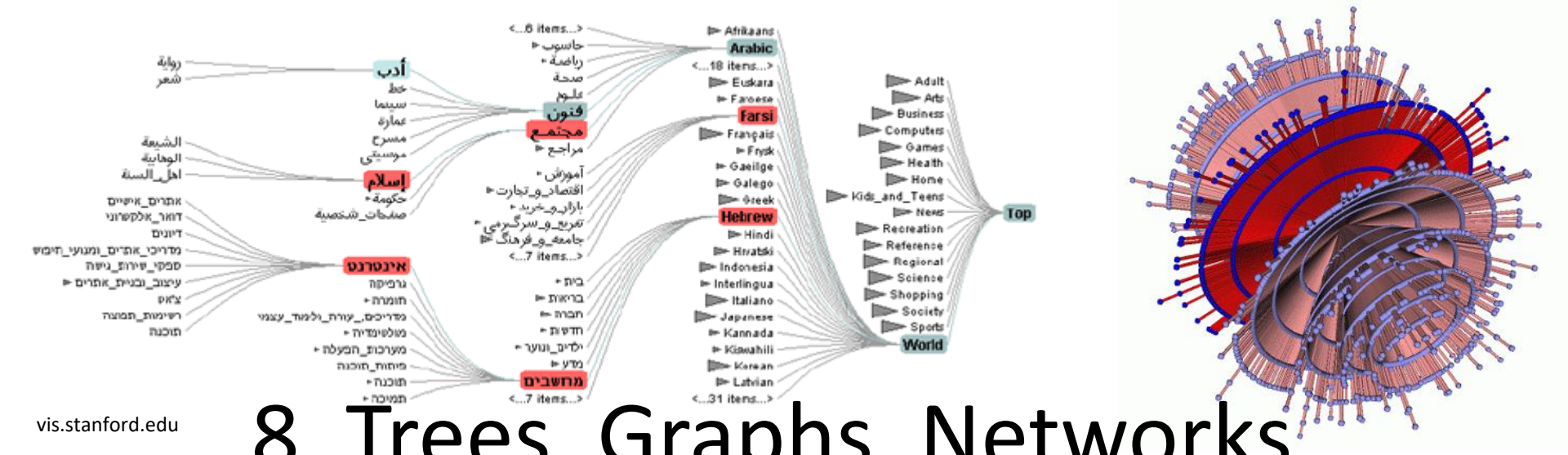
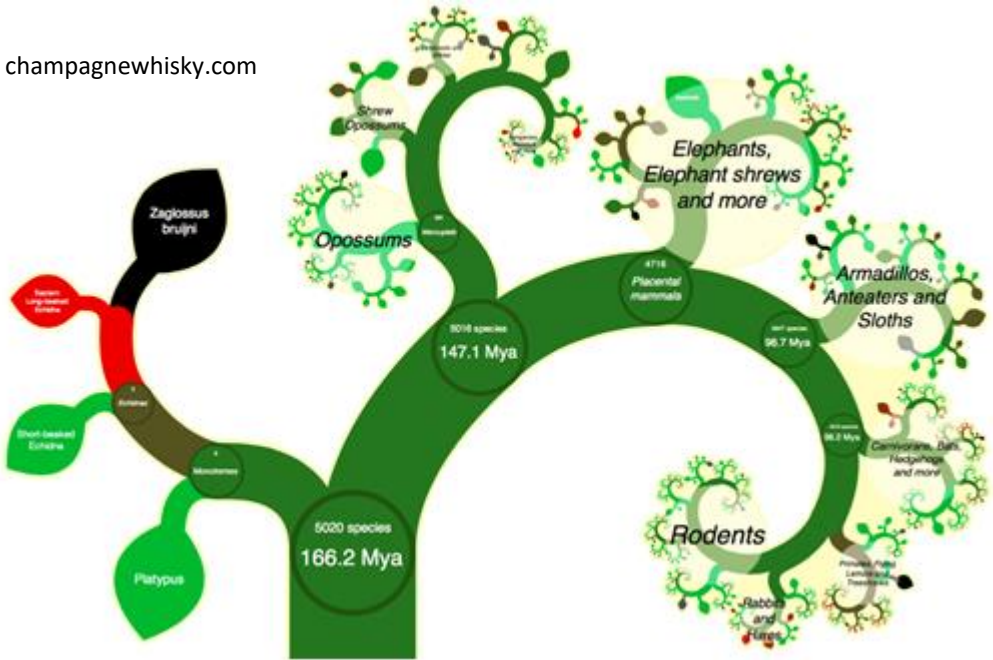


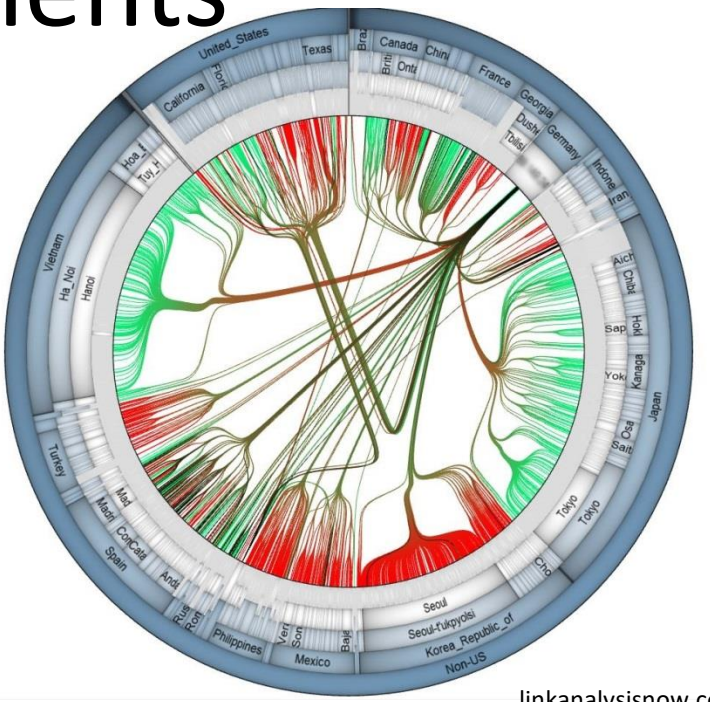
vis.stanford.edu

8. Trees, Graphs, Networks, Texts, Documents

www.ibiblio.org

champagnewhisky.com

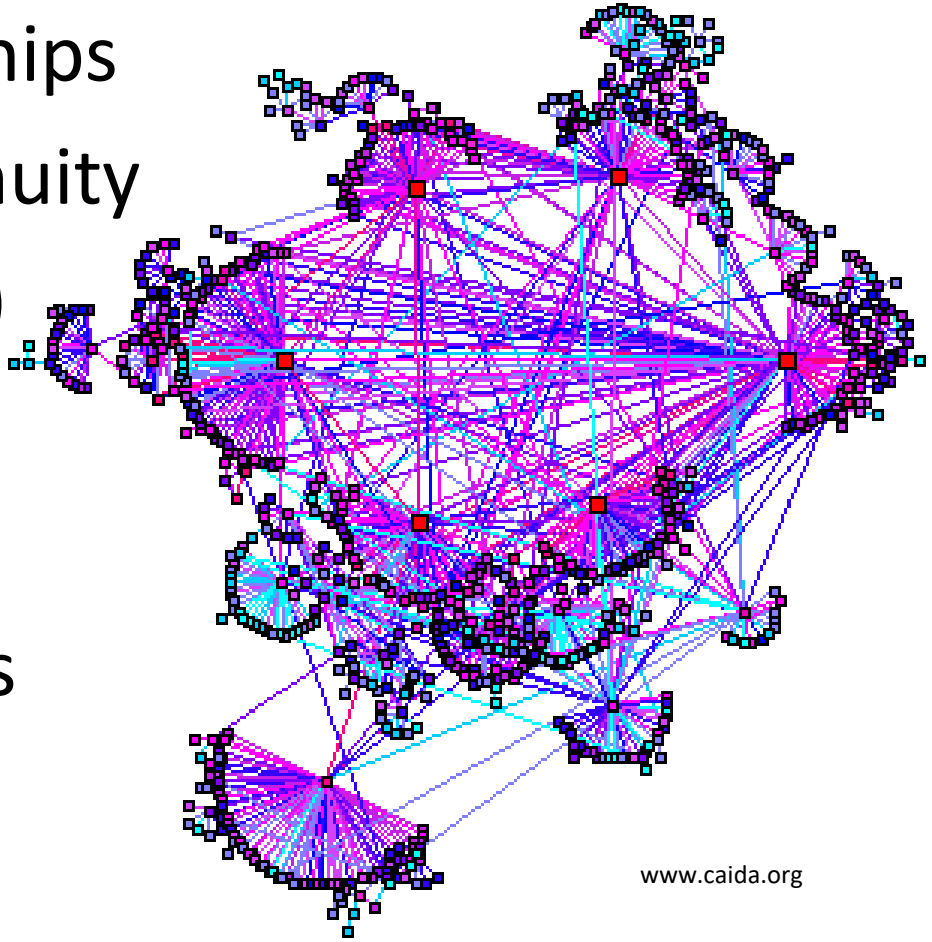




linkanalysisnow.com

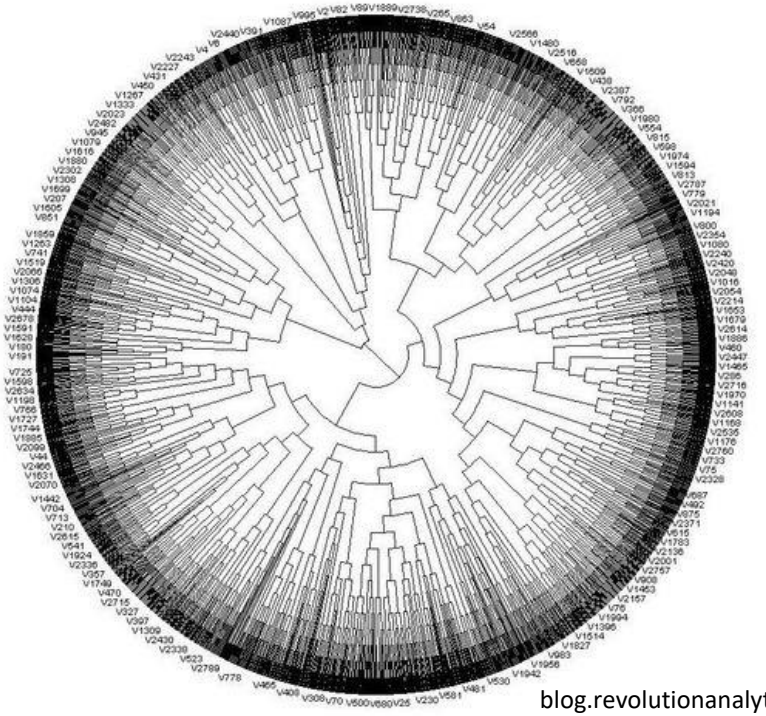
Relationships Between Data

- Hierarchical relationships
- Interconection, continuity
- Derivation (sequence)
- Shared classification
- Similarity of values
- Similarity of attributes



Visualization of Hierarchical Structures

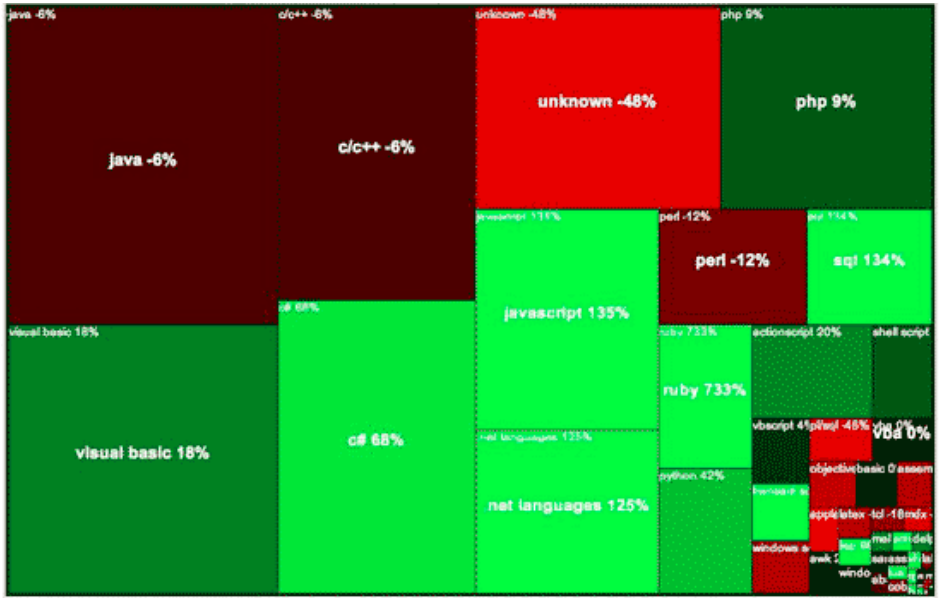
- Hierarchical structures = trees
- Algorithms for visualization:
  - Space-filling – maximal usage of screen space
  - Non-space-filling

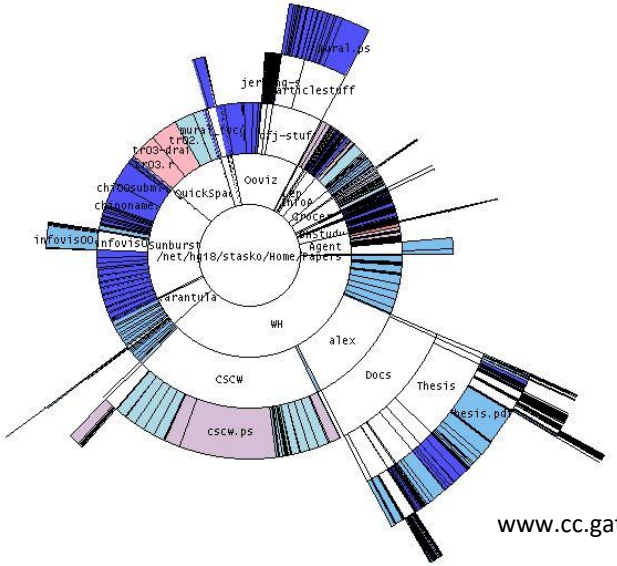


blog.revolutionanalytics.com

Space-filling Methods

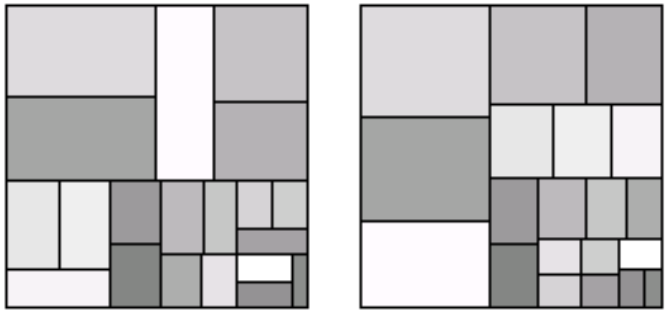
- Two most common types:
  - Rectangular layout
    - Treemaps
  - Radial layout
    - Sunburst view



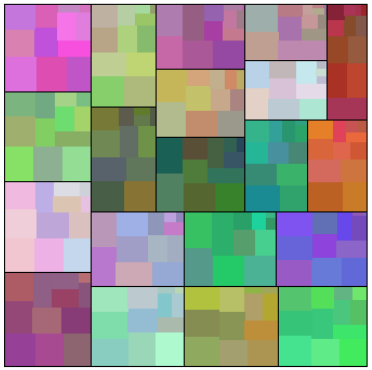


Rectangular Layout

- Treemaps
  - Basic form - recursive splitting of rectangle alternatively by horizontal and vertical lines
  - Other variants, e.g.:
    - Squarified treemaps
    - Nested treemaps



hcil2.cs.umd.edu



www.mathworks.com

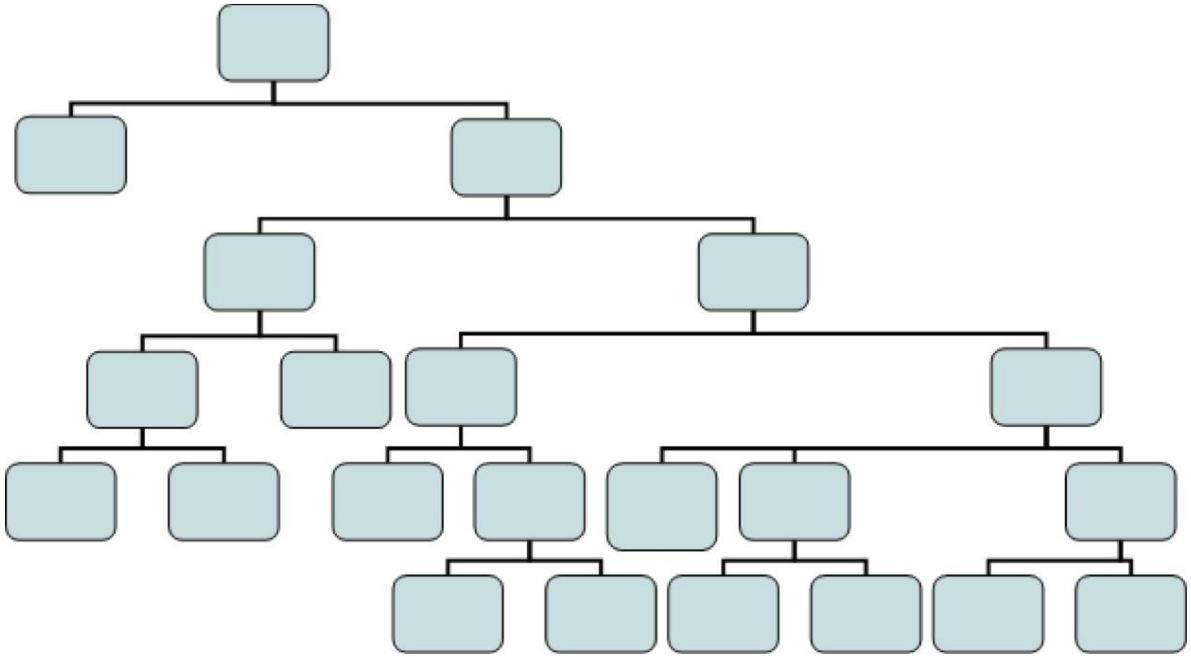
Treemap - Pseudocode

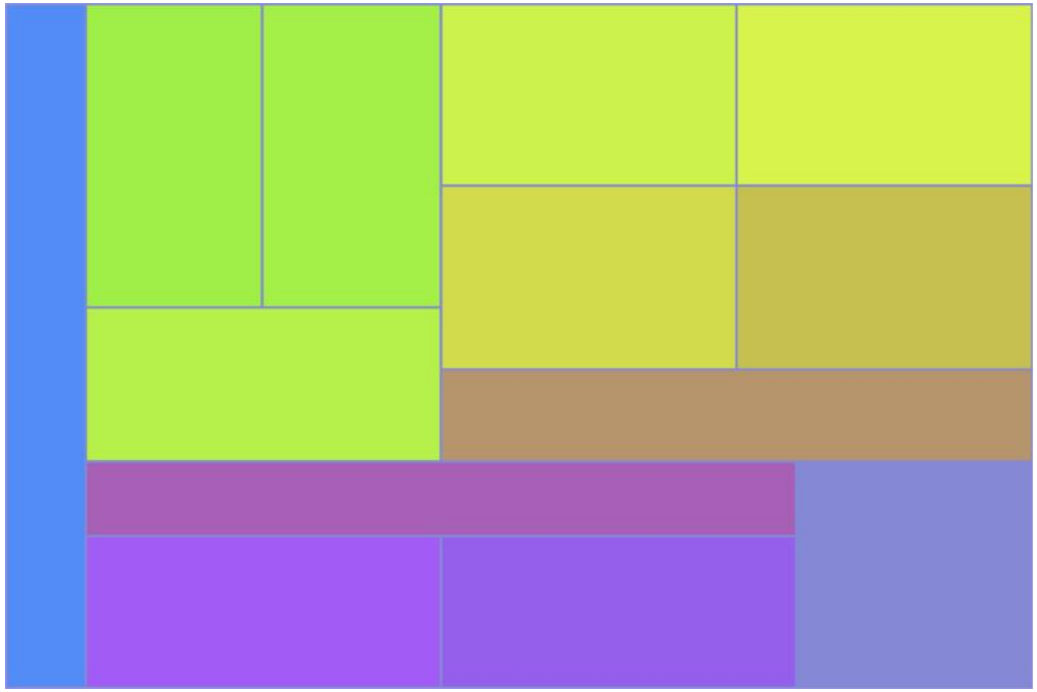
Naming:

```
Width = width of the rectangle
Height = height of the rectangle
Node = root node of the tree
Position = position of rectangle (e.g., [0, 0])
Orientation = orientation of the split – alternating
horizontal and vertical split
```

`treemap(Node n, Orientation o, Position orig, Width w, Height h)`

```
treemap(Node n, Orientation o, Position orig, Width w,
Height h)
   if n is the leaf node (has no child nodes)
       draw_rectangle(orig, w, h);
       return;
   for each child node of the n (child_i)
       get number of leaf nodes in the subtree;
   sum up the total number of the leaf nodes;
   compute the percentage of leaf nodes in each subtree
   (percent_i);
   if orientation is horizontal
       for each subtree
          compute the offset of the origin based on the
          origin and the width (offset_i);
          treemap(child_i, vertical, orig+offset_i,
          w*percent_i, h);
   else
       for each subtree
          compute the offset of the origin based on the
          origin and the height (offset_i);
          treemap(child_i, horizontal, orig+offset_i, w,
          h*percent_i);
```





Radial Layout

- Sunburst displays
- Root of the hierarchy placed in the center of the radial view, individual layers of hierarchy represented by concentric rings
- Rings are split based on the number of nodes at the given level
- Radial techniques display inner nodes as well as leaves

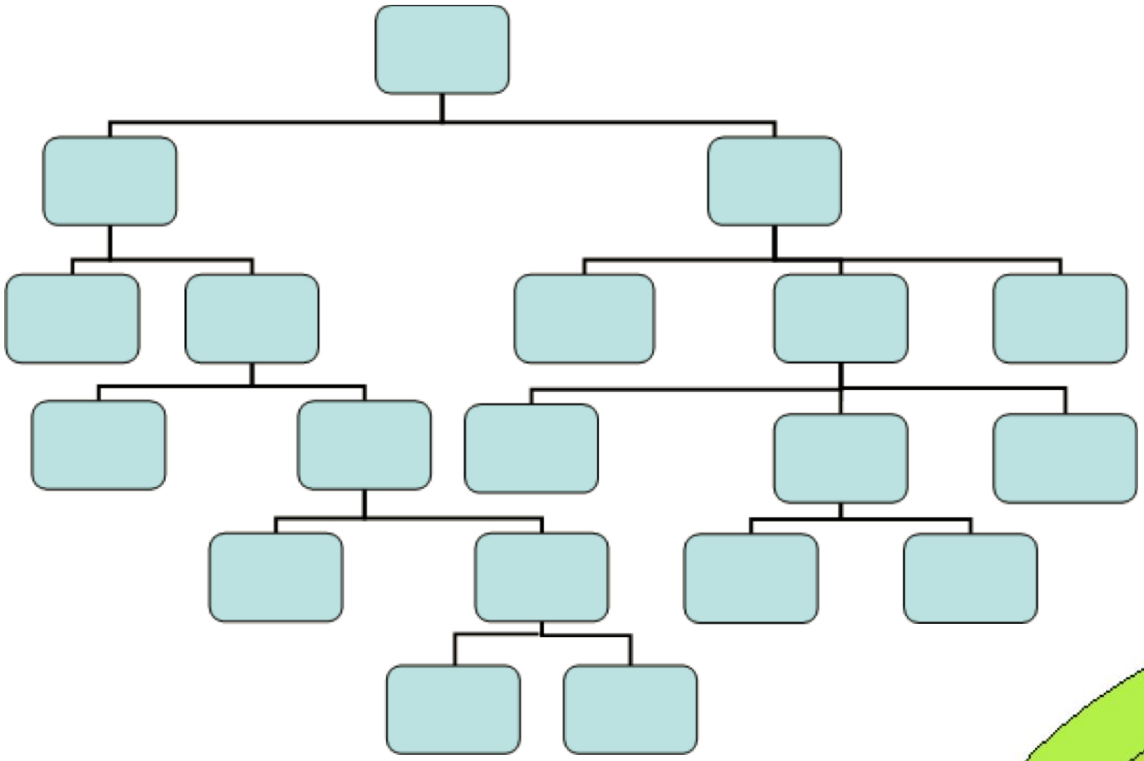
Sunburst - Pseudocode

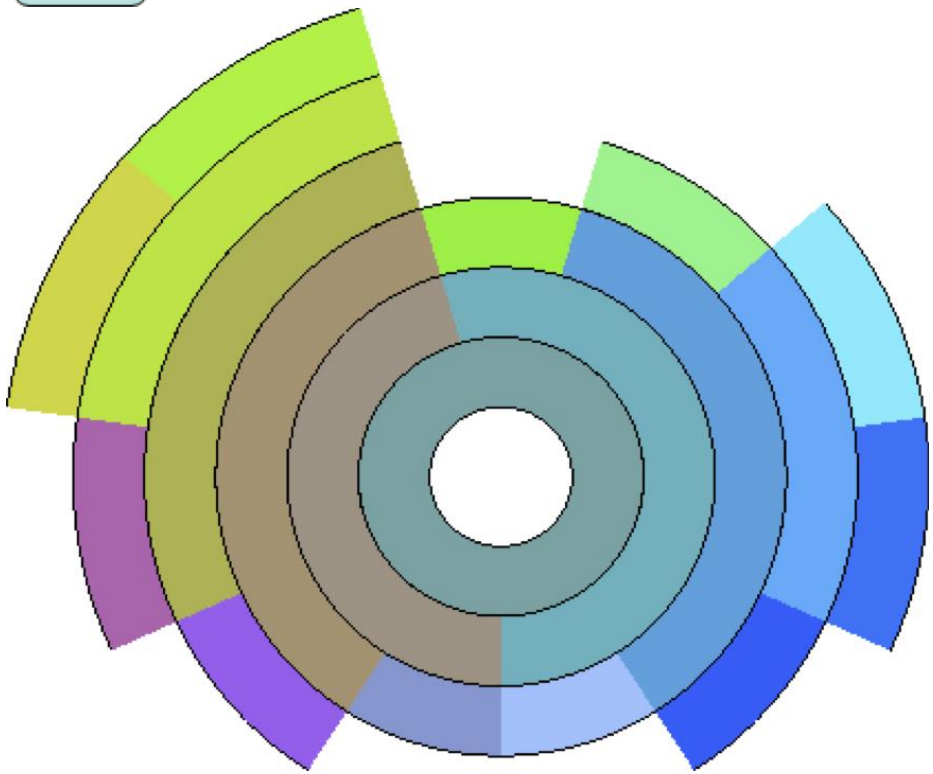
Naming:

```
Start = start angle of the node (initially 0)
End = end angle of the node (initially 360)
Origin = position of the center of the radial view
         (e.g., 0, 0])
Level = current level of hierarchy (initially 0)
Width = width of each ring – based on maximal depth
        of the hierarchy and the size of display
```

sunburst(Node n, Start st, End en, Level l)

```
sunburst(Node n, Start st, End en, Level l)
   if n is the leaf node (has no child nodes)
      draw_radial_arc(Origin, st, en, l*Width,
      (l+1)*Width);
      return;
   for each child node of the n (child_i)
       get number of leaf nodes in the subtree;
   sum up the total number of the leaf nodes;
   compute the percentage of leaf nodes in each
   subtree (percent_i);
   for each subtree
      compute the initial/end angle based on the size
      of subtrees, their order and the range of
      angles;
      sunburst(child_i, st_i, en_i, l+1);
```





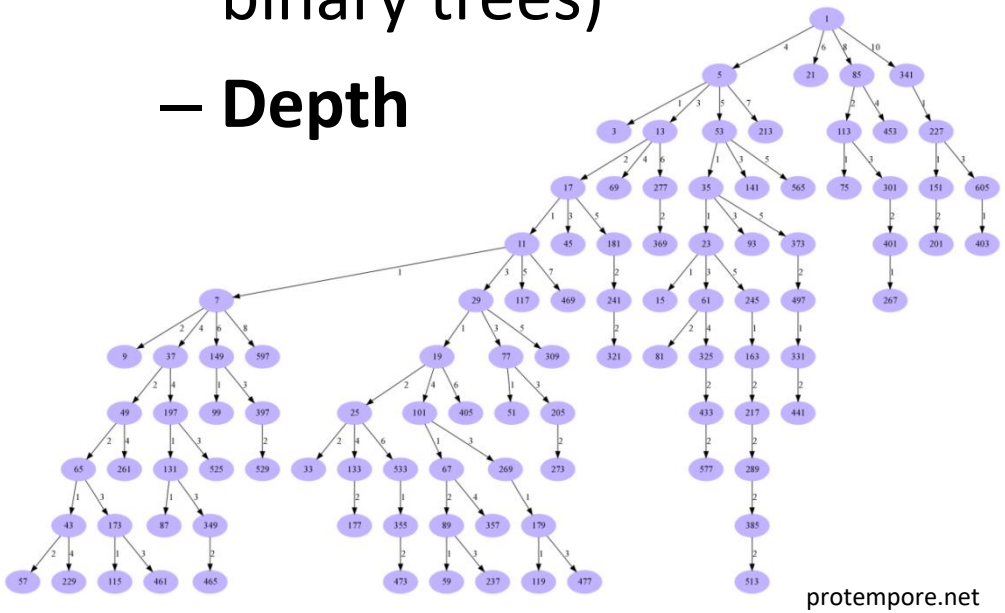
Color in Hierarchical Techniques

- Highlighting many attributes, e.g.:
  - Values in nodes
  - Enhancement of hierarchical relationships (similar color for parent and child nodes)

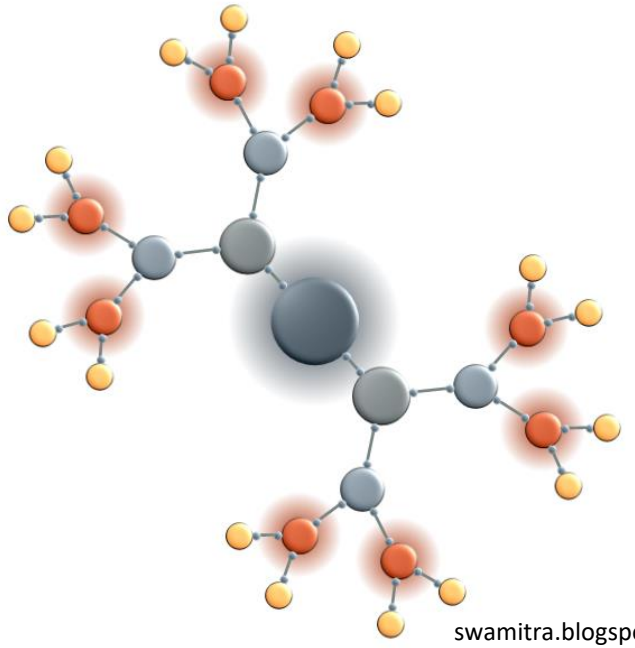
- Additional properties displayed by symbols, marks and labels placed into rectangular or radial segments

Non-Space-Filling Methods

- Node-link diagrams are the most common
- Diagrams dependent mostly on two factors:
  - **Degree of the nodes** (number of branches, e.g., binary trees)
  - **Depth**



protempore.net



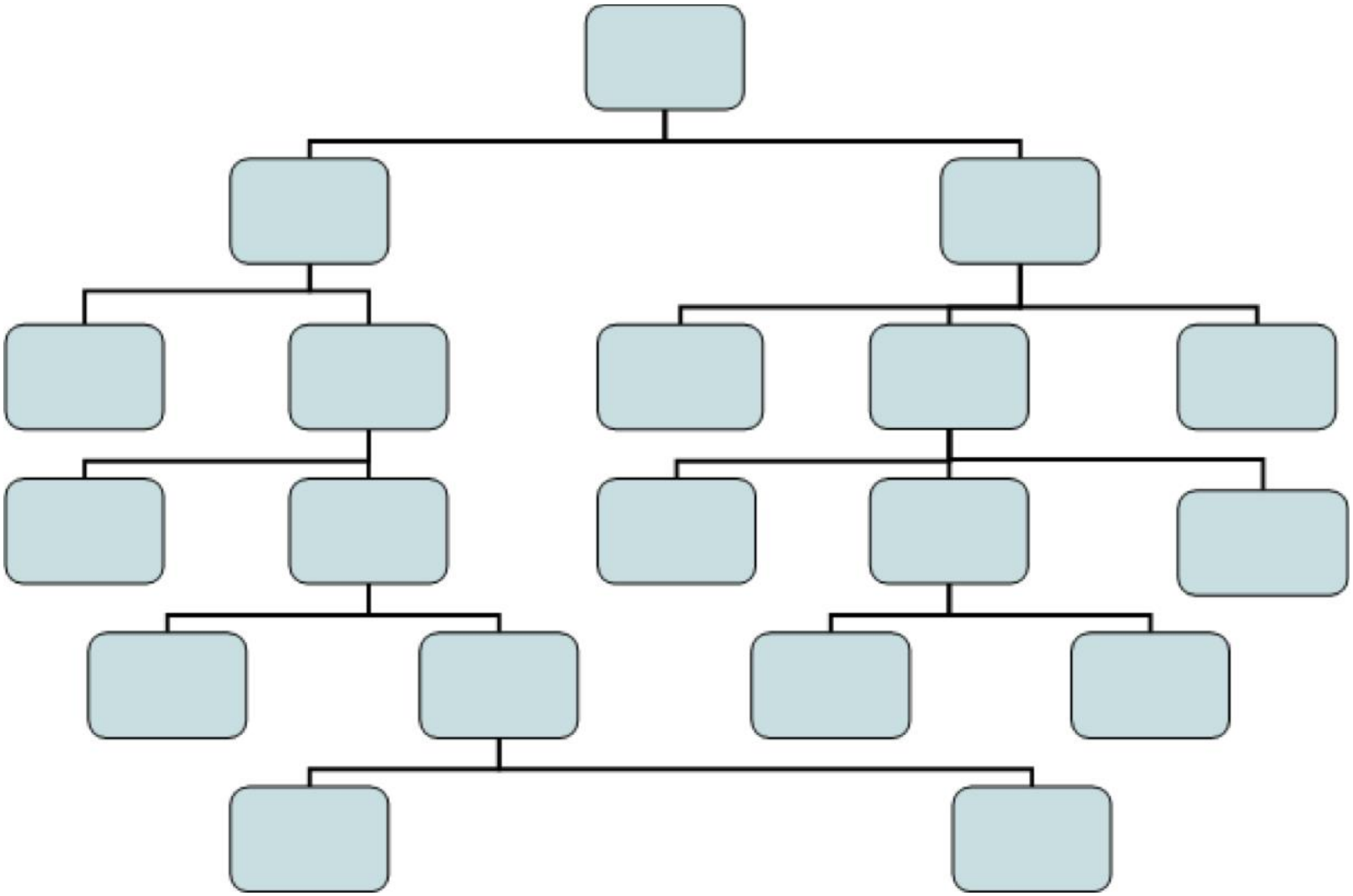
swamitra.blogspot.com

Design of the Algorithms for Rendering Node-Link Graphs

- Three categories of rules:
  - **Rendering conventions** – shape and curvature of edges, placement of the nodes into fixed grid, neighbors at the same vertical position, …
  - **Restrictions** – placement of the nodes at given position, displaying nodes in close proximity, orientation of edges, …
  - **Aesthetics** – minimization of edge intersections, keeping the width/height ratio of the graph, minimization of the overall graph area, minimization of the edge length, minimization of the edge bending, minimization of the number of different angles and curves, effort to keep the symmetry

Design of the Algorithms for Rendering Trees

1. Divide the rendering area into slices of the same height – the number of slices is derived from the depth of the tree
2. For each level of the tree, determine how many nodes need to be drawn
3. Divide each slice into rectangles of same size – derived from the number of nodes at the given level
4. Draw each node into the corresponding rectangle
5. Draw connecting line leading from the center of the bottom edge of each node to the center of the top edge of the its child nodes

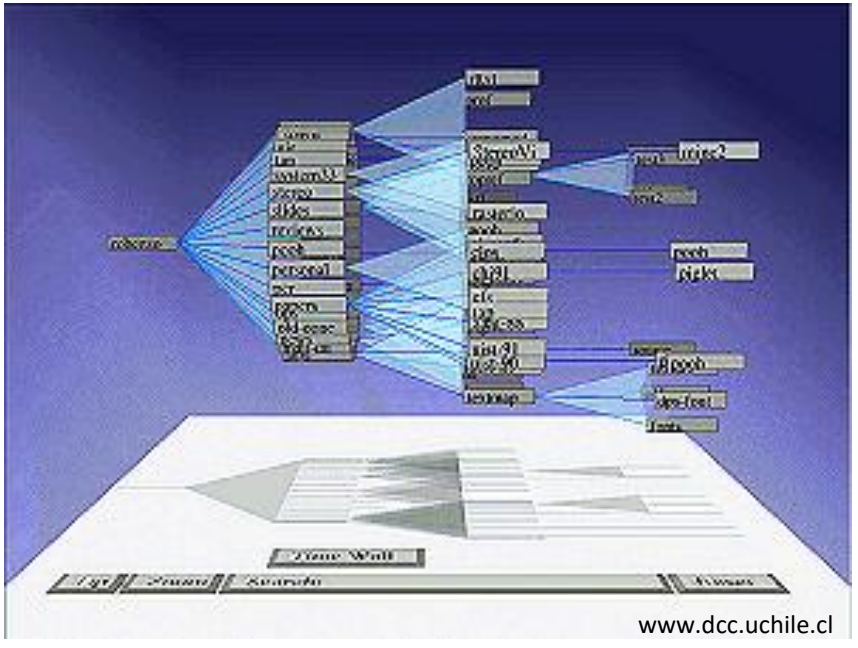


Optimalisation

- Improve the usage of the screen space
- Examples:
  - Each level divided by the number of leaf nodes in the corresponding subtree
  - Uniform layout of the leaf nodes + centering of their parent nodes
  - Adding additional gaps between neighboring nodes that do not have common parent (i.e., are not siblings)
  - Reorganization of the tree in order to improve the symmetry and balance
  - Root node at the center of the screen, child nodes arranged radially around it

Application of Third Dimension

- For large trees, complemented by rotation, translation and zooming
- E.g., **cone trees** – child nodes are uniformly radially placed around parent node
  - Primary parameters are radius and translation distance



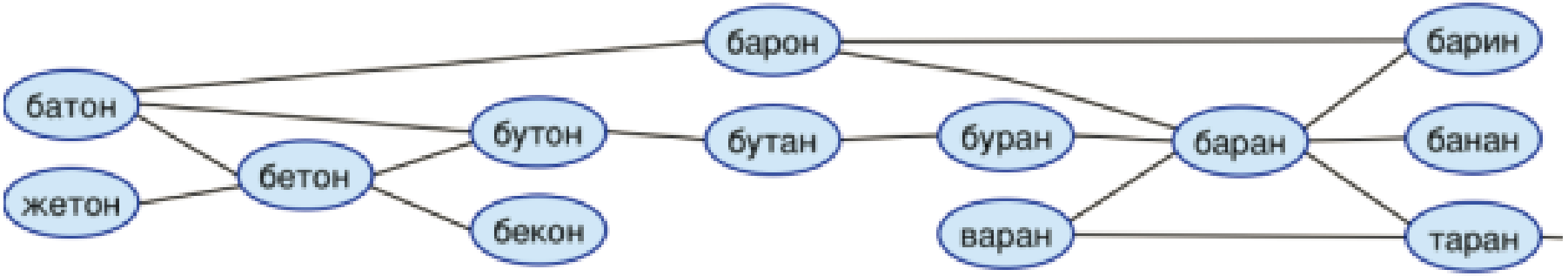
www.dcc.uchile.cl

Arbitrary Graphs/Networks

- Trees are just one category of **graphs** – connected unweighted acyclic graphs
- Other graph representations exist, e.g., graphs with weighted edges, undirected graphs, cyclic graphs, disjoint graphs, …
- We will focus on **arbitrary graphs** – their structure is not known. Two different approaches:
  - **Node-link diagrams**
  - **Matrix displays**

Node-Link Diagrams

- Force-directed graphs – strings between nodes
- Links between nodes are iteratively adjusted, until the stress values is minimalized
- Example:https://vega.github.io/vega/examples/force-directed-layout/



http://www.absint.com/aisee/gallery.htm

Planar Graphs

- Edges are not intersecting
- Very popular
  - Long history, many studies
  - Edge intersections complicate the interpretation of the graph, it is better to avoid them
  - Sparse – according to Euler's formula for *n* nodes there are at most *3n – 6* edges
- Graphs with intersections can be transformed to planar graphs by introduction of „dummy" nodes at intersection points – after transforming the graph to planar layout, the dummy nodes are removed

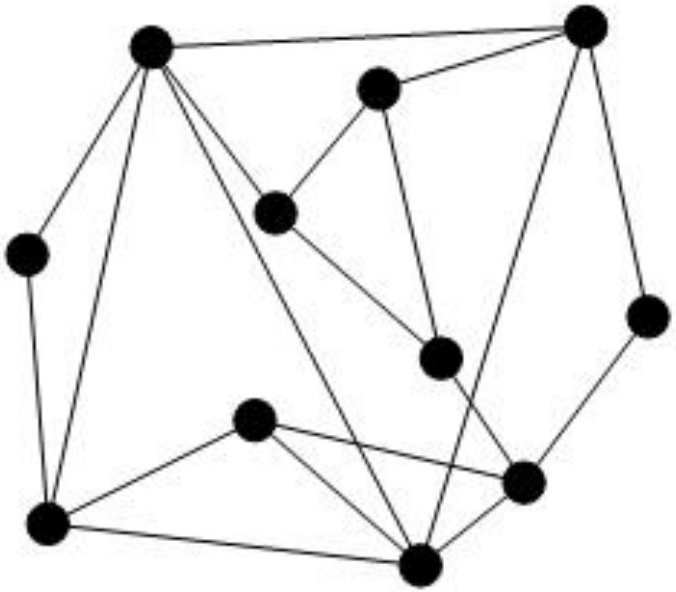
Planar Graphs

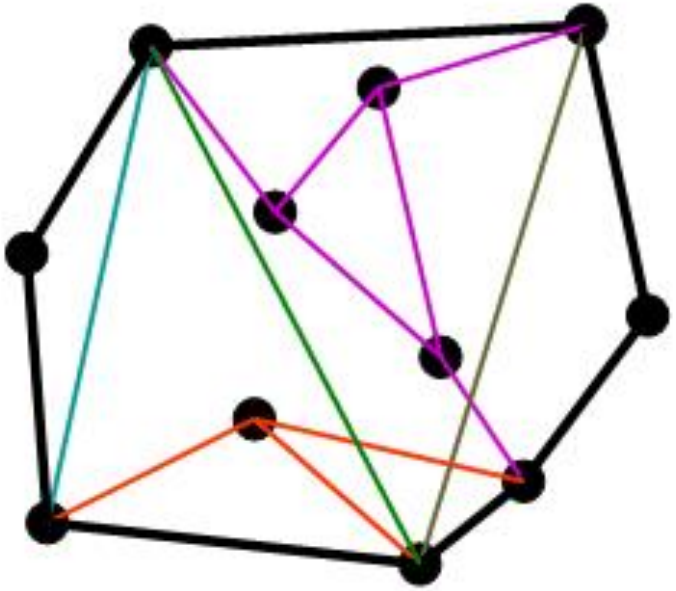
- We have to set a strategy, by which to decide if the graph is planar – algorithms are either complex or computationally expensive
- Simplification:
  - Graph is planar if all its connected components are planar
  - Connected graph is planar only if all its biconnected components are planar
- We therefore only need algorithm for detection of planarity of biconnected graph

Algorithm for Detection of Planarity of Biconnected Graph

- Divide and conquer approach
- If the graph contains cycle and does not contain any other cycle sharing an edge of the original cycle, then after removal of this cycle only paths (“pieces”) starting and ending in one of the cycle vertices remain
- If two pieces intersect, one must be drawn inside of the cycle and one outside
- If we create a graph of paths, where the intersecting paths are divided by an edge and the graph is bipartite (separable into two sets of nodes, such that within the nodes of one set no edge exists), then the original graph is planar

Algorithm for Detection of Planarity of Biconnected Graph

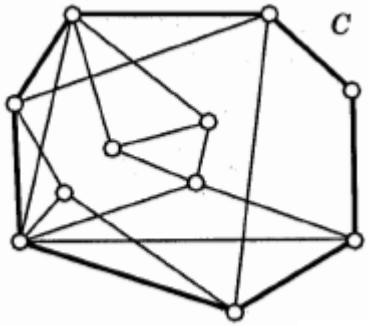


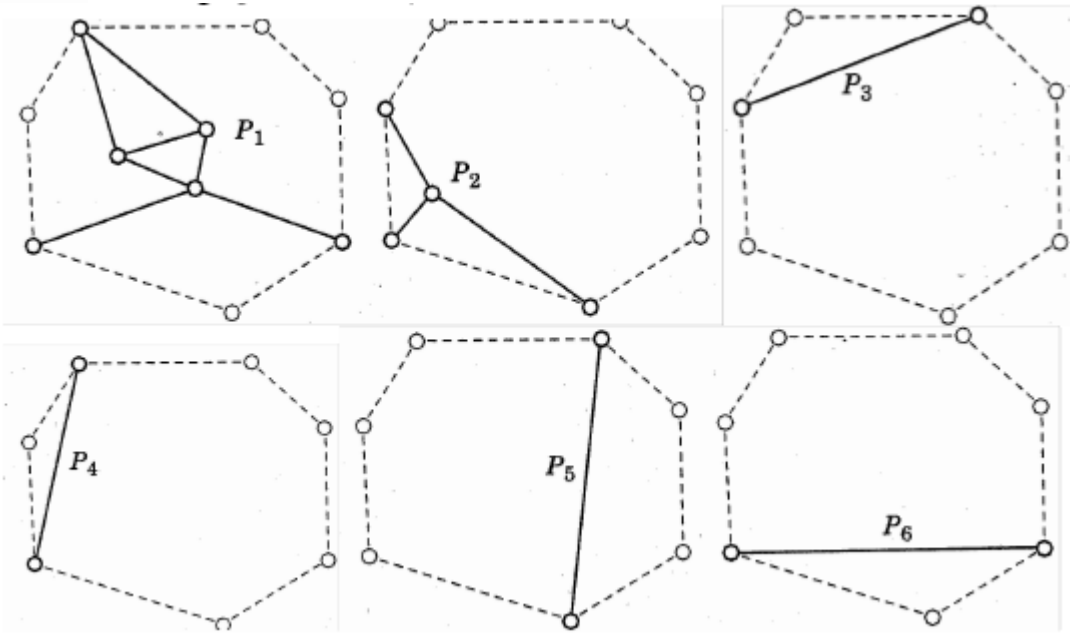


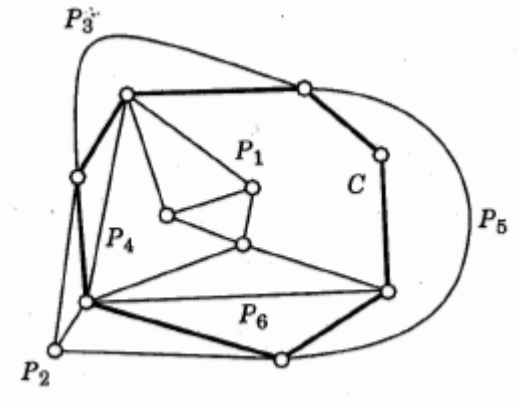
Algorithm for Detection of Planarity of Biconnected Graph

- If the graph still contains cycles after the removal of the edges of the original cycle, it means that one or more pieces contain cycle
- Create subgraph containing this piece and the parts of original cycle containing the endpoints of the piece
- Recursively call the algorithm for detection of planarity

Example



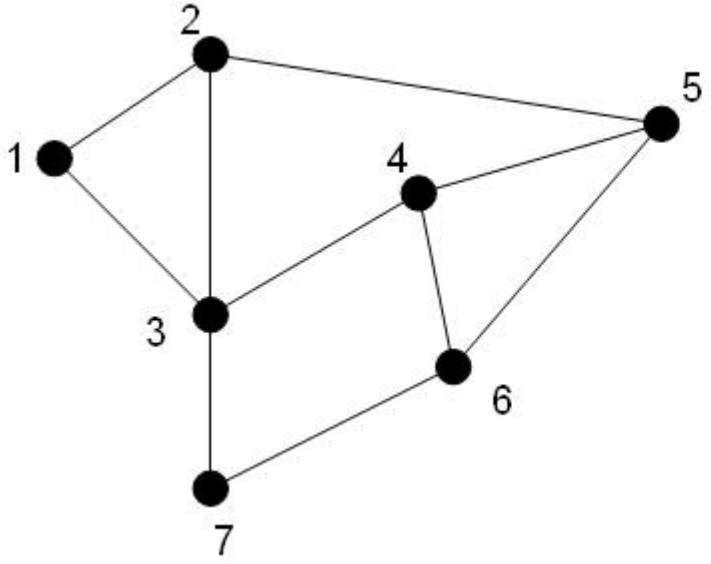


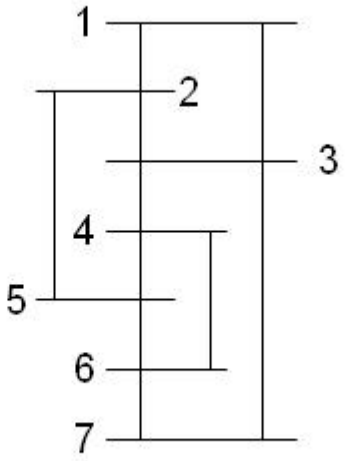


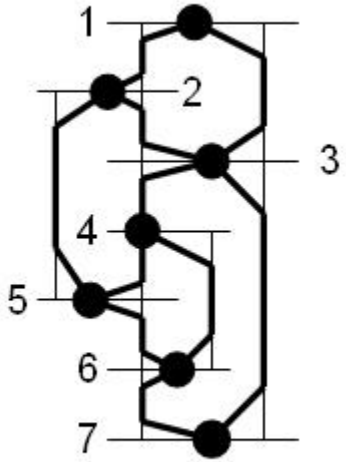
Rendering Planar Graphs

- Visibility approach – 2 steps:
  - **Visibility step** – create a representation, where each node is rendered as horizontal segment and each edge as vertical line connecting corresponding segments of the nodes
  - **Replacement step** – each segment corresponding to a node „shrinks“ into a point and each vertical connector is replaced by polyline

Rendering Planar Graphs







Matrix Representation of Graphs

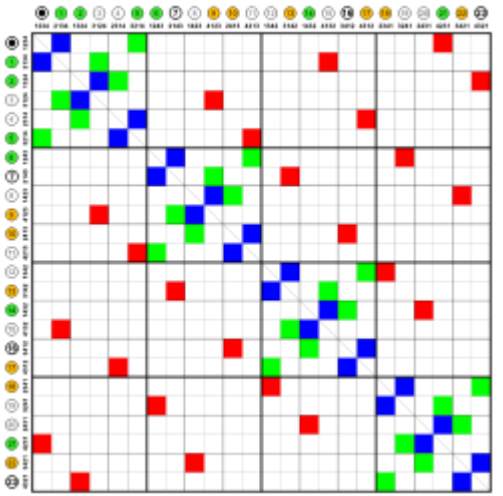
- **Adjacency matrix** = grid of NxN dimensions (N is the number of nodes)
  - Binary or containing values of forces or weights of edges
  - Overcomes the problem of edge intersections
  - The strategy for organisation of rows and columns is crucial for revealing interesting patters and structures of the graph

	a	b	c	d	e	f	g	h
a		●	●			●		
b	●			●		●		
c	●				●		●	●
d	●	●				●		
e			●				●	●
f	●	●		●				
g			●		●			●
h			●		●		●	

	p	q	r	s	t	u	v	w
p		●	●	●				
q	●		●	●				
r	●	●		●				
s	●	●	●		●			
t				●		●	●	●
u					●		●	●
v					●	●		●
w					●	●	●	

Matrix Representation of Graphs

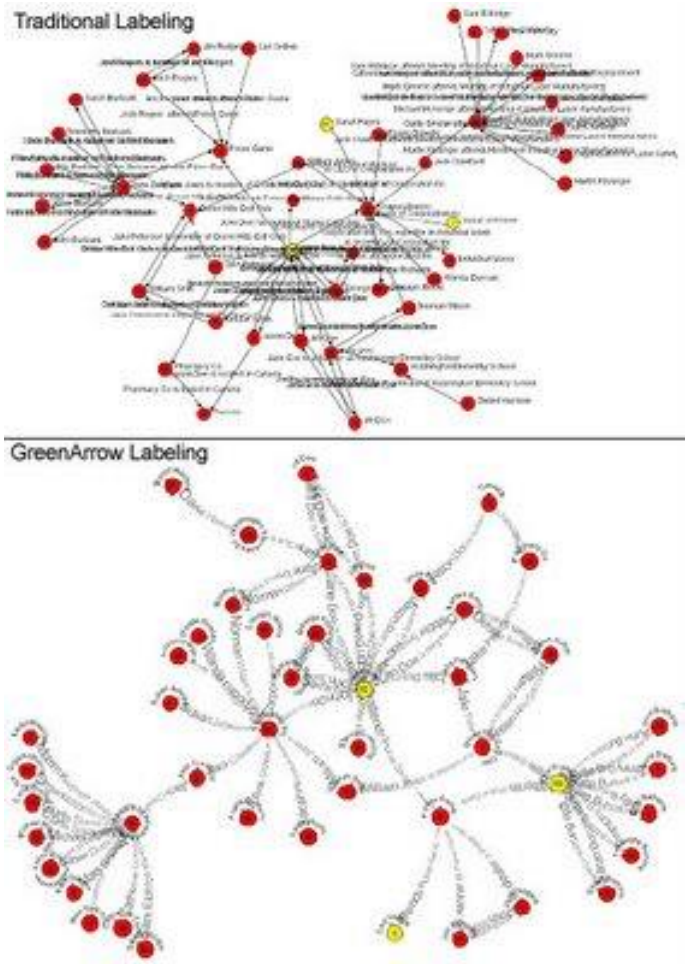
- Variety of algorithms for row and column reorganisation
  - User driven / automatic
  - Finding the optimal solution is NP-complete problem, therefore the application various heuristics is necessary



en.wikipedia.org

Labeling

- Necessary for understanding what the graph represents
- When rendering trees and graphs the labelling is difficult, because the graphs may contain large amount of nodes and labelling of edges may also be required



labelwallpapers.blogspot.com

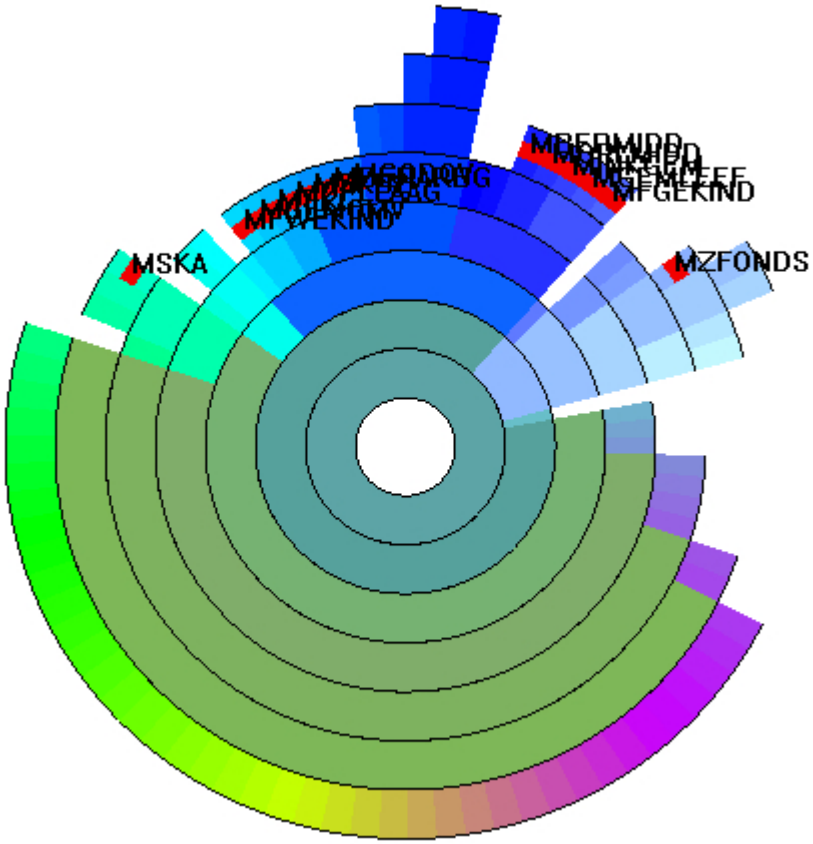
Labeling

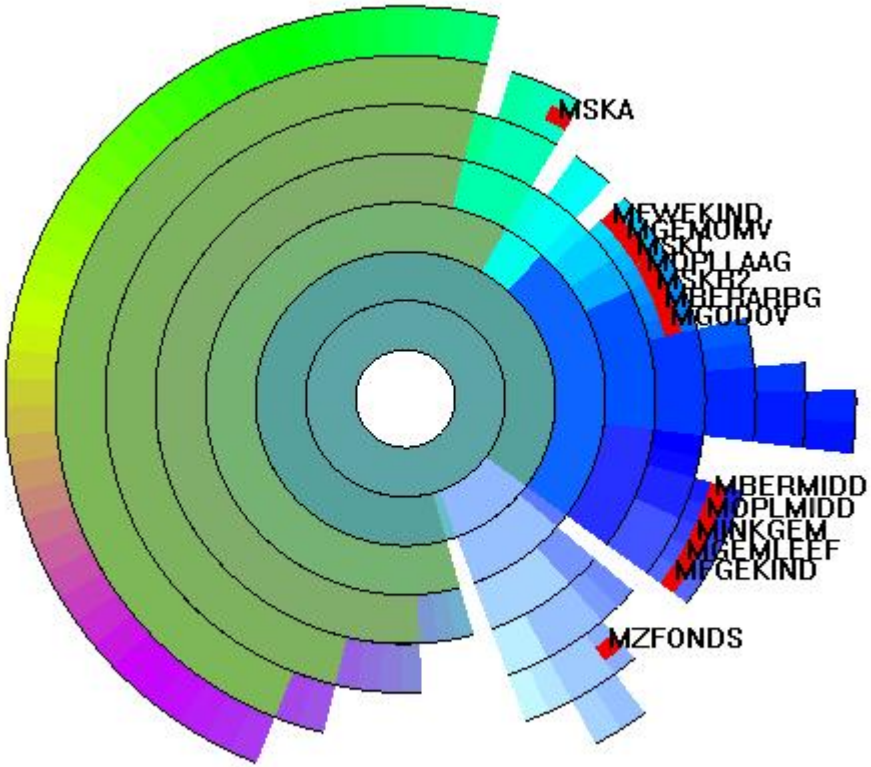
- For small amount of different labels it is better to use color, size, or shape of the node or color, thickness and the style of the edge line
- If the number of different labels is over 5 or 6 – use textual labels
- Small graphs – inserted directly into the node (rectangular or oval shape of node) – size of the nodes based on the longest text
- Unified placement of labels for edges:
  - Vertical – everything left or everything right to the edge
  - Horizontal – everything above or everything below the edge

Labeling

- If the number of different labels is too large, showing them all at once is not effective. We can use different strategies:
  - Showing labels only in the neighbourhood of current cursor position
  - Various deformations of the visualization, so we can dedicate the given part of the graph larger amount of screen space
  - Rotation of graph in order to reduce overlaps and occlusions of the labels

Labelling


MSKA
MZFONDS
MFGEKIND

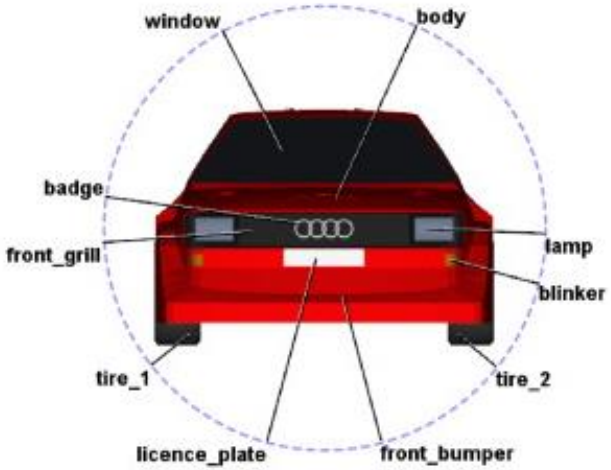

MSKA
MEWEKIND
MGEMOMV
MOPLLAAG
MBERARBG
MGODOV
MBERMIDD
MOPLMIDD
MINKGEM
MGEMLEEF
MFGEKIND
MZFONDS

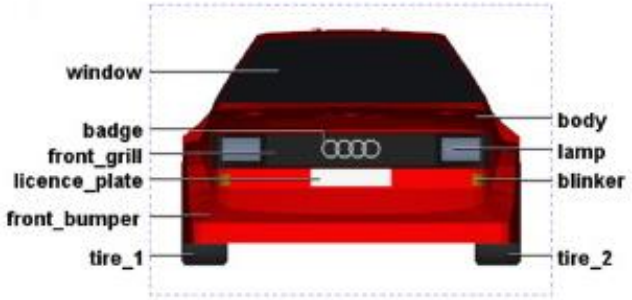
Labeling

- Showing random subset of labels for short time interval and then showing different random subset etc.
- Short-term memory of the user enables remembering of larger number of labels than when using static view

Labelling

- Ladislav Čmolík, Jiří Bittner, Layout-aware optimization for interactive labelling of 3D models, Computers & Graphics, Volume 34, Issue 4, August 2010, pages 378-387, ISSN 0097-8493





Trees, Graphs, and Interaction

- Generic types of interaction
  - E.g., trailing camera, zooming
  - Common for all types of visualization
- Specialized interactions
  - E.g., focus + context
  - Applicable for a large variety of visualizations, but primarily developed for visualization of trees and graphs

Interactions with Virtual Camera

- Common interactions (trailing camera, zoom, rotation) that are considered to be simple changes applied to the virtual camera observing a certain part of the scene
- Operations are directed manually or automatically (e.g., flights over the scene, automatic rotation of 3D objects)

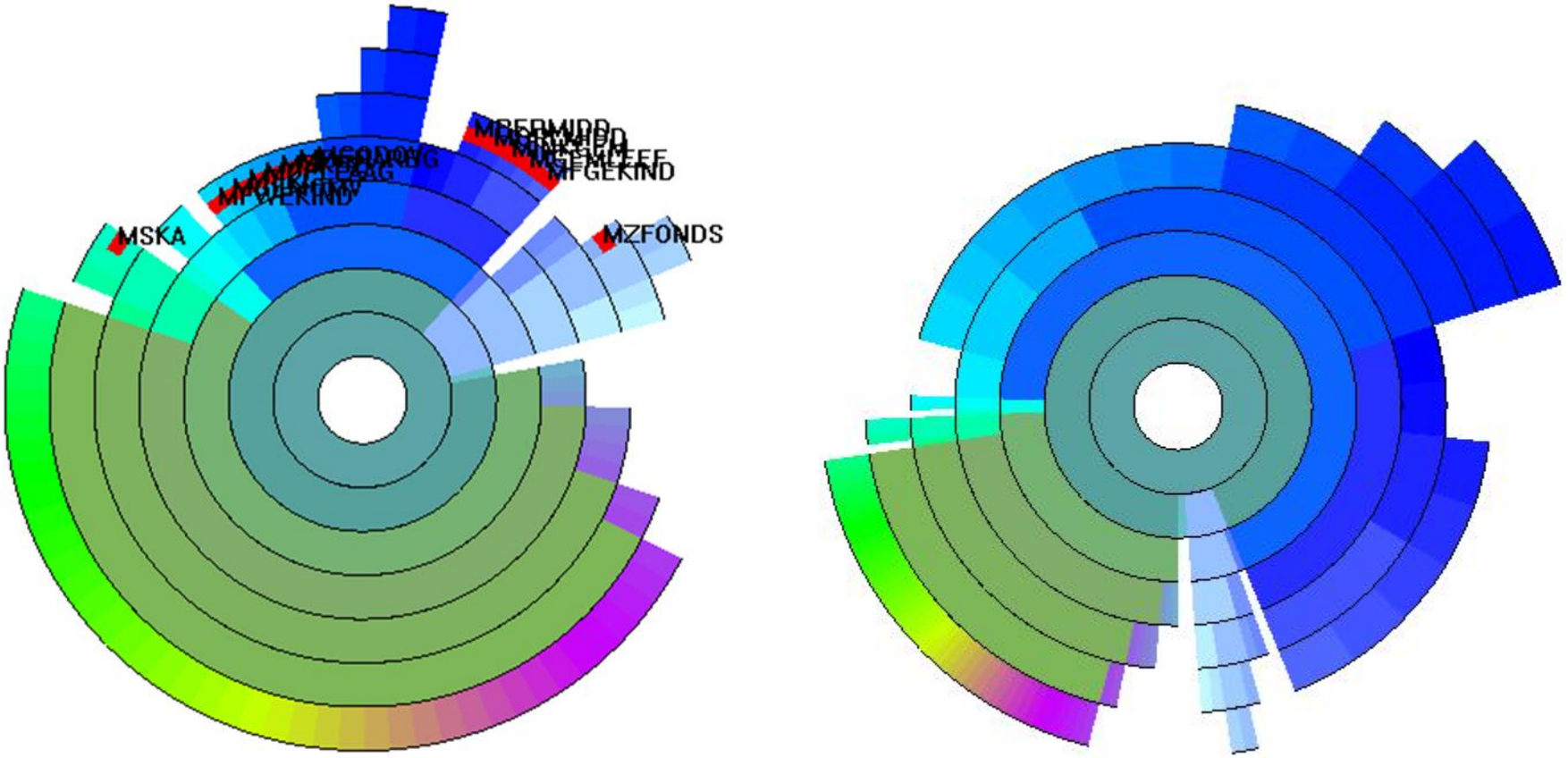
Interactions with Graph Elements

- Starts with selection
  - We isolate a single or more graph components
- Graphs with disarranged complicated clusters can be adjusted:
  - Select a set of nodes and drag it to the less cluttered part of the screen
  - Select, translate, or change the shape of edges in order to eliminate their intersections or increase the aesthetical value of the graph
- Problems – in dense cluttered regions it is almost impossible to select object unambiguously

Interactions with Graph Structure

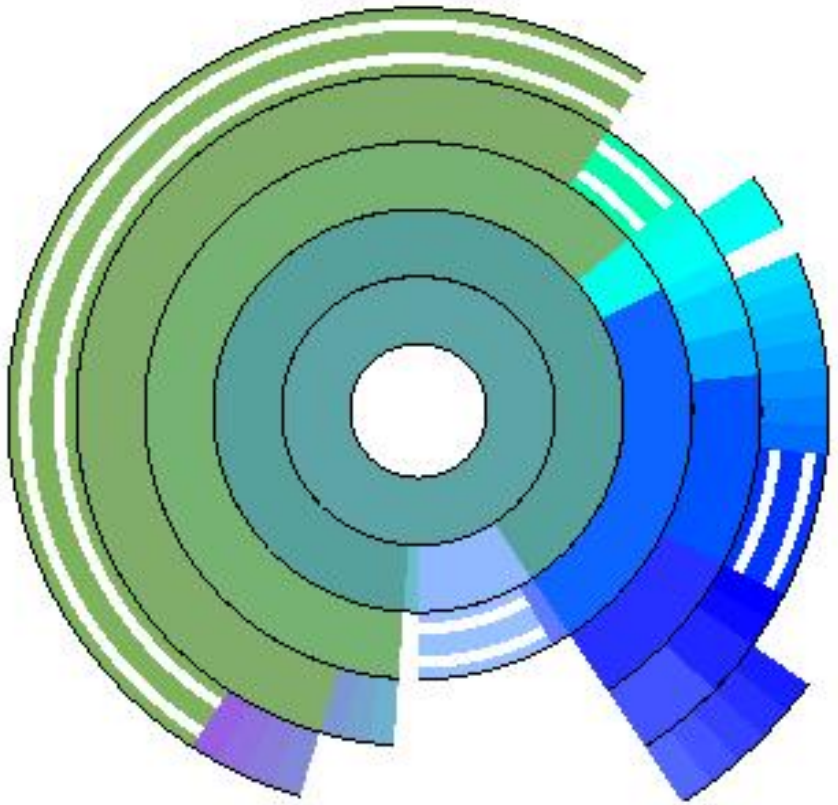
- Two basic classes of interaction:
  1. Changing the structure of the graph (e.g., reordering the branches of the tree)
  2. Focus+context techniques, where a subset of the structure is represented in detail and the rest of the structure just in outline (e.g., fish eye)
     - Performed in screen space
     - Performed in structure space – suitable for graphs (enlargement of one branch (see image on the next slide), highlighting of edges adjacent to a given node, …)

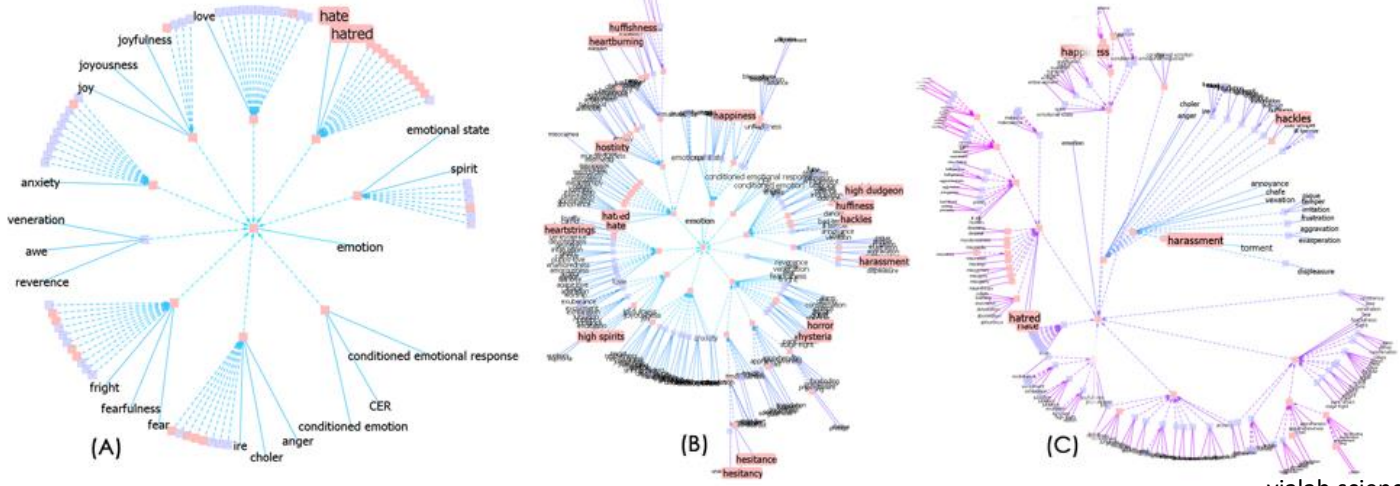
Interactions with Graph Structure



Interactions with Graph Structure

- Selective hiding or removal of the selected parts of the graph





vialab.science.uoit.ca

Text and Document Visualization

giladlotan.coms

joeganley.com

Definitions

- **Corpus** = set of documents
- We work with objects inside of corpus – words, sentences, paragraphs, whole documents, other corpora as well as images and videos
- Texts and documents – structured, contain attributes and metadata
- Systems for data mining from documents – querying

Levels of Text Representation

- Three levels:
  - Lexical
  - Syntactic
  - Semantic
- Each level requires certain conversion of unstructured text to some form of structured data

Lexical Level

- Transformation of string of characters into sequence of atomic entities = **tokens**
- Lexical analysers process sequence of characters with a given set of rules into new sequence of tokens
- Tokens contain characters , n-grams, words, lexemes, phrases, …
- Finite state automata

Syntactic Level

- Identification and annotation of function of each token
- Assignment of marks – position of sentence, lexical category, singular or plural, affinity of tokens, …
- Process of annotation extraction = **NER (named entity recognition)**

Semantic Level

- Extraction of the meaning and the relationship of the findings derived from the structures identified in syntactic level
- Goal is to define analytic interpretation of the whole text in the given context or independent of the context

Vector Space Model (VSM)

- Algebraic model for representation of text documents
- „Term vector“ = vector, in which each dimension represent a weight of a given word in a document
- Removal of „stop words“ (the, a, …) for reduction of noise
- Aggregation of words with the same root
- Example of term vector is a hash table which maps unique terms to the number of their occurrences in the document

Vector Space Model

```
Count-Terms(tokenStream)
1. terms := ∅; /* initialize terms
   to empty hash table*/
2. for each token t in tokenStream
3.   do if t is not stop word
4.      do increment (or initialize
           to 1) terms[t];
5. return terms;
```

Example

- **Text example:**

There is a great deal of controversy about the safety of genetically engineered foods. Advocates of biotechnology often say that the risks are overblown. ‘‘There have been 25,000 trials of genetically modified crops in the world, now, and not a single incident, or anything dangerous in these releases,’’ said a spokesman for Adventa Holdings, a UK biotech firm. During the 2000 presidential campaign, then-candidate George W. Bush said that ‘‘study after study has shown no evidence of danger.’’ And Clinton Administration Agriculture Secretary Dan Glickman said that ‘‘test after rigorous scientific test’’ had proven the safety of genetically engineered products.

Example

- The aforementioned paragraph contains 98 string tokens, 74 terms and 48 terms after removal of stop words
- Example of term vector generated by the pseudocode:

genetically	said	safety	engineered	study	test	great	deal	controversy	foods
3	3	2	2	2	2	1	1	1	1

VSM – Computation of Weights

- Various methods of weight assignment, most well known is **term frequency inverse document frequency (TfIdf)**
- If **Tf(w)** is term frequency = number of occurences of a word w in a document, **Df(w)** is document frequency = number of documents, which contain word w, **N** is number of documents, then TfIdf(w) is:

$$TfIdf(w) = Tf(w) * \log\left(\frac{N}{Df(w)}\right)$$

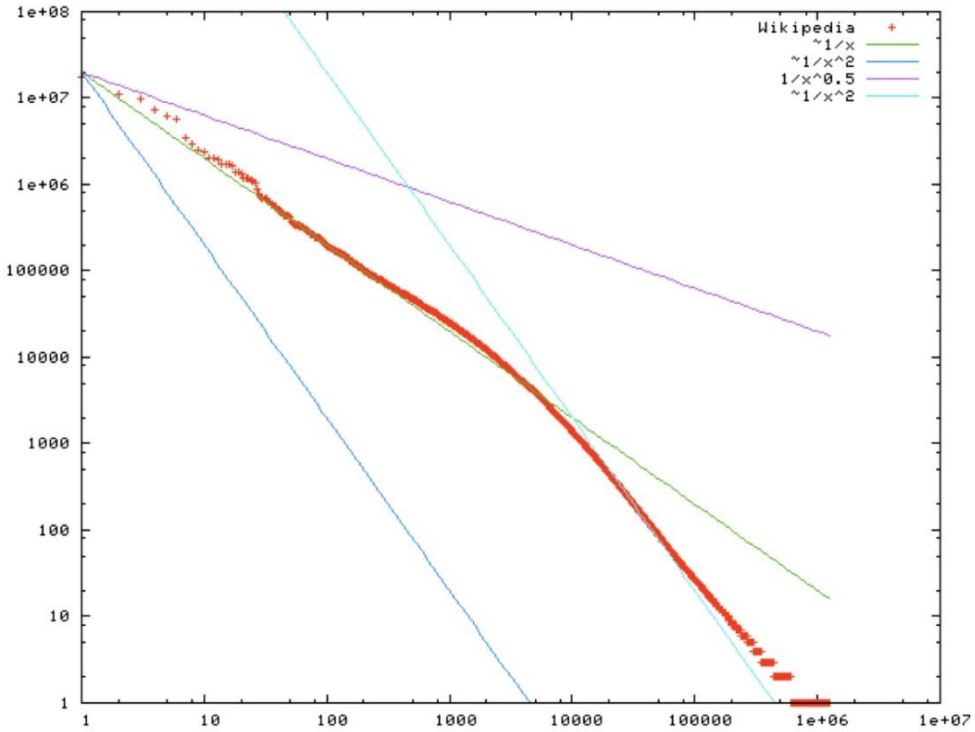
VSM – Computation of Weights

- We are interested in words that are frequently occurring in one document but are not so common in other documents of the corpus

id	men	entered	bank	charlotte	missiles	masks	aryan	guns	witnesses	reported	silver	suv	august
seg1.txt	0.239441	0	0.153457	0.195243	0	0.237029	0	0.195243	0.237029	0.140004	0.195243	0.237029	0
seg13.txt	0	0	0	0	0	0	0	0	0	0	0	0	0
seg14.txt	0	0.192197	0	0	0	0	0	0	0	0	0	0	0.172681
seg15.txt	0	0	0	0	0	0	0	0	0	0	0	0	0.149652
seg16.txt	0	0	0	0	0	0	0	0	0	0	0	0	0
seg17.txt	0	0	0	0	0	0	0	0	0	0	0	0	0
seg18.txt	0	0.158432	0	0	0	0	0	0	0	0	0	0	0
seg19.txt	0	0	0	0.197255	0	0	0	0	0	0.141447	0	0	0.155038
seg2.txt	0	0	0	0	0	0	0	0	0	0	0	0	0
seg20.txt	0	0.234323	0	0	0	0	0	0	0	0	0	0	0
seg21.txt	0	0	0	0	0	0	0	0	0	0	0	0	0
seg22.txt	0	0	0	0	0.139629	0	0.127389	0	0	0	0	0	0
seg23.txt	0	0	0	0	0	0	0	0	0	0.180656	0	0	0
seg24.txt	0	0	0	0	0	0	0.117966	0	0	0.117966	0	0	0
seg25.txt	0	0	0	0	0	0	0	0	0	0	0	0	0
seg26.txt	0	0	0	0	0	0	0	0	0	0	0	0	0
seg27.txt	0	0	0.235418	0	0	0	0.214781	0	0	0	0	0	0
seg28.txt	0	0	0	0	0.151753	0	0	0	0	0	0	0	0
seg29.txt	0	0	0	0	0	0	0.129852	0	0	0	0	0	0.142329
seg3.txt	0	0	0	0	0.18432	0	0	0	0	0	0	0	0
seg30.txt	0.078262	0	0	0	0	0	0	0	0	0	0	0	0
seg31.txt	0	0	0.213409	0	0	0	0.194701	0	0	0	0	0	0
seg32.txt	0	0	0	0	0	0	0	0	0	0	0	0	0

Zipf's Law

- In a typical document in a natural language the frequency of any word is inversely proportional to its rank in the frequency table.

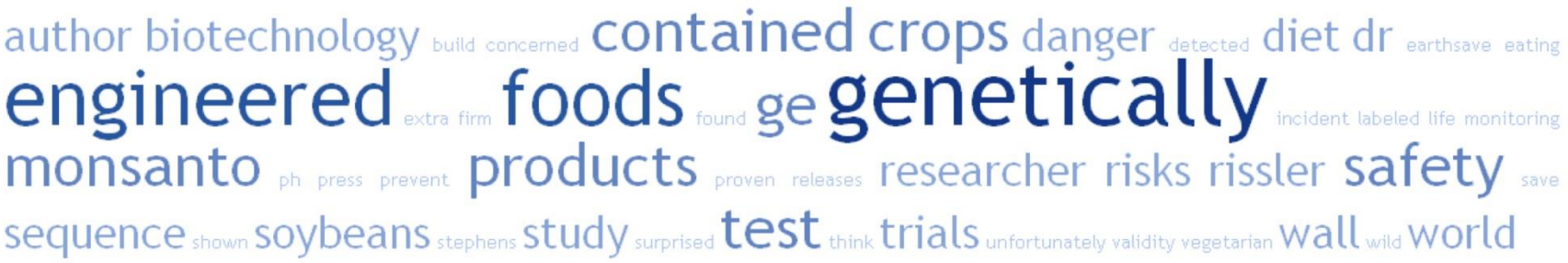


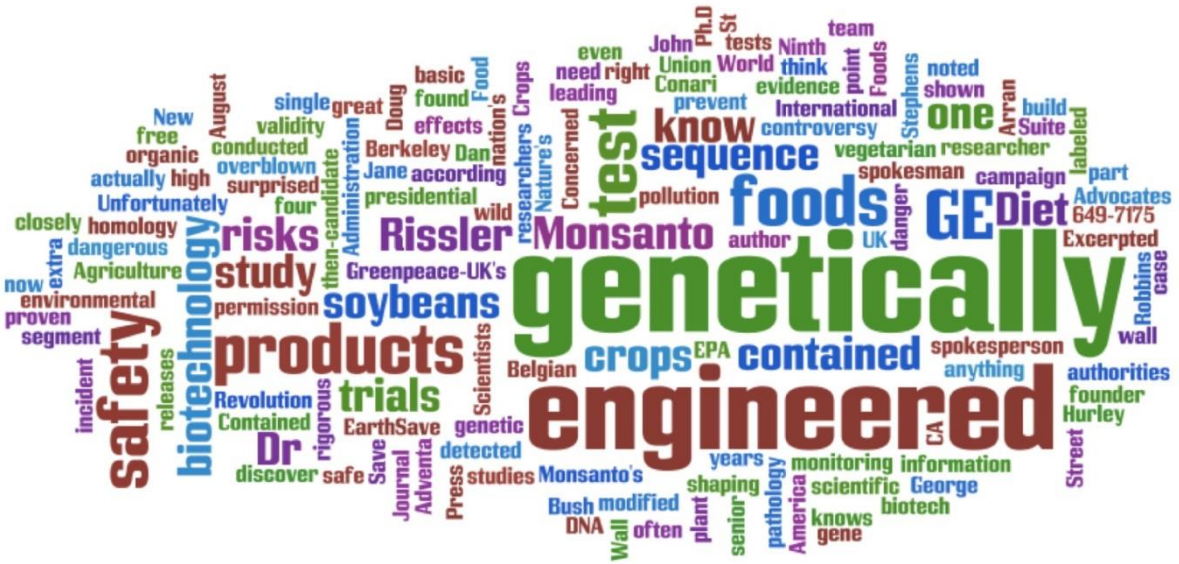
Tasks Using Vector Space Model

- Vector Space Model complemented by distance metric allows for whole set of various tasks:
  - TfIdf in combination with VSM – identification of interesting documents, querying
  - Providing information about the meaning of the whole corpus – searching for patterns, clusters, …
- Visualization pipeline can be well mapped onto visualization of the documents:
  - Data acquisition (corpus), transformation to vectors, processing by algorithms based on the tasks, visualization

Visualization of Individual Documents

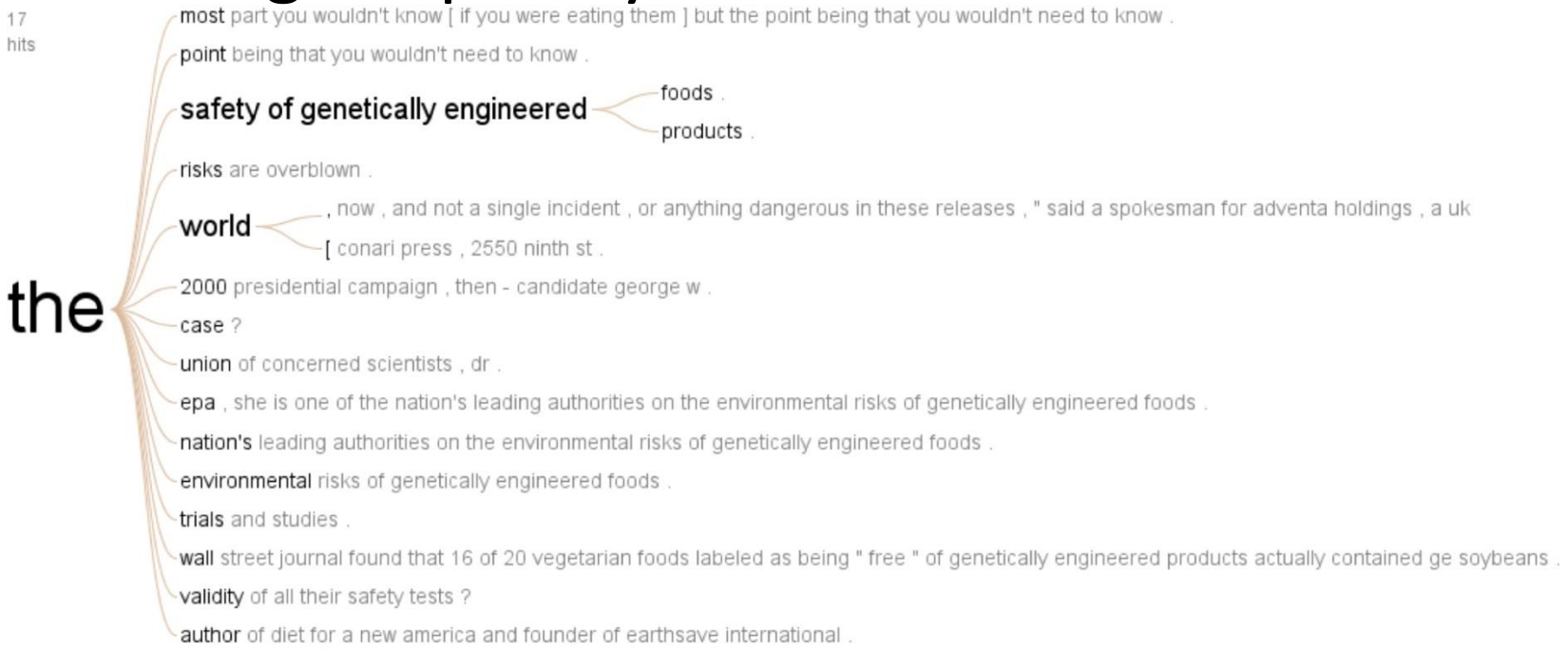
- Tag clouds





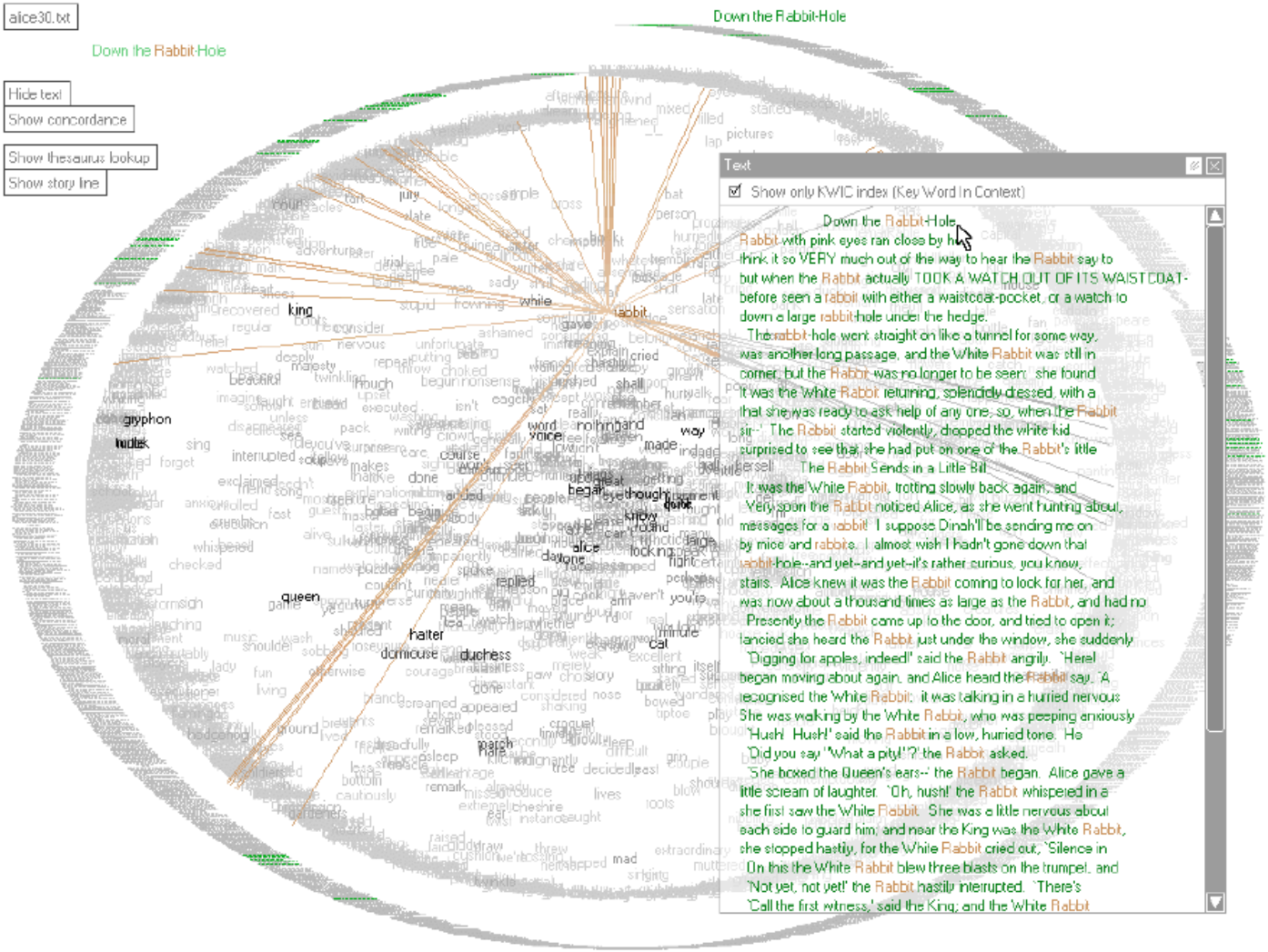
Visualization of Individual Documents

- WordTree – tree branches represent various contexts in which the word from the root of the tree occurs in the document – also showing frequency of terms

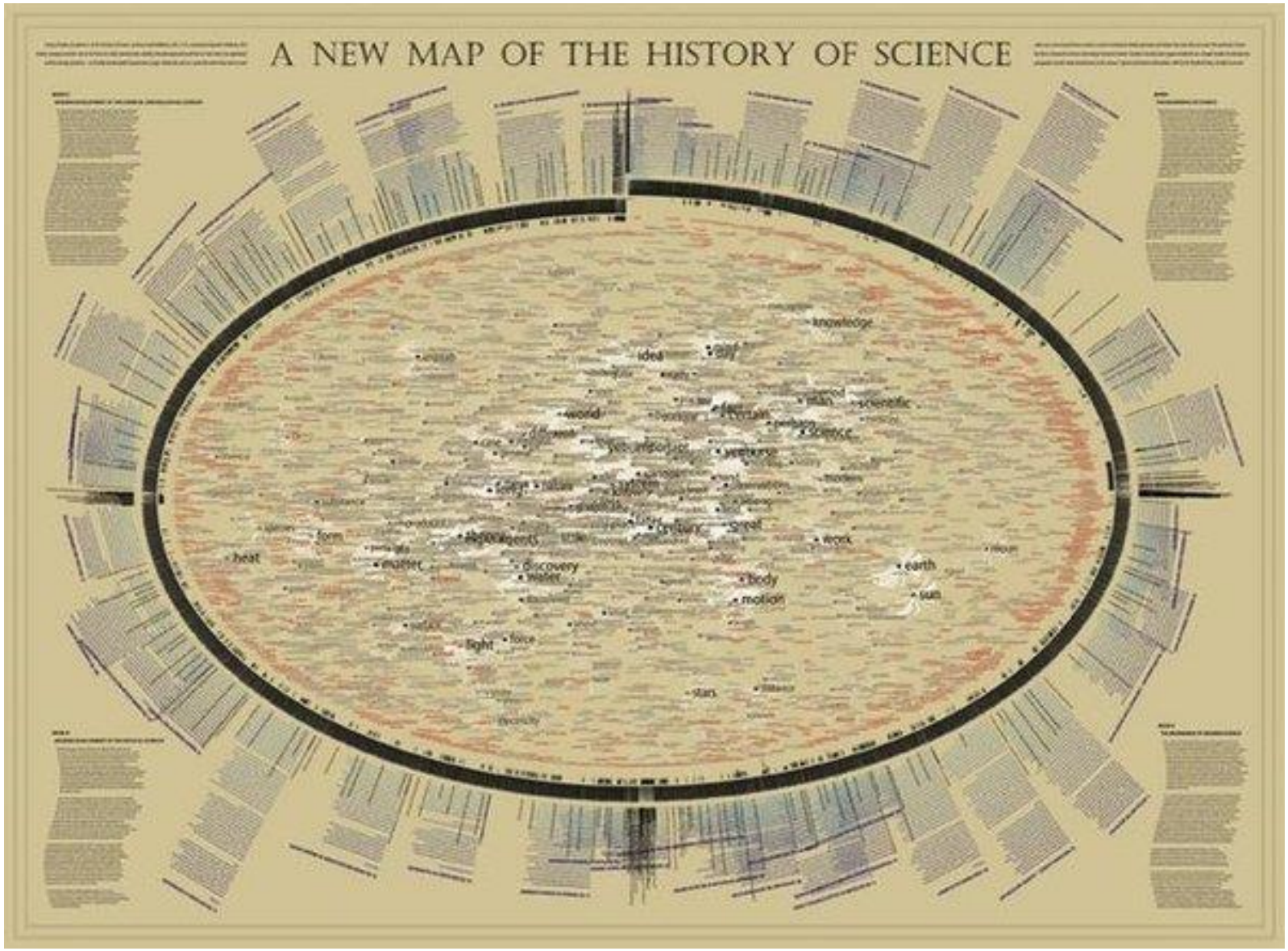


Visualization of Individual Documents

- TextArc

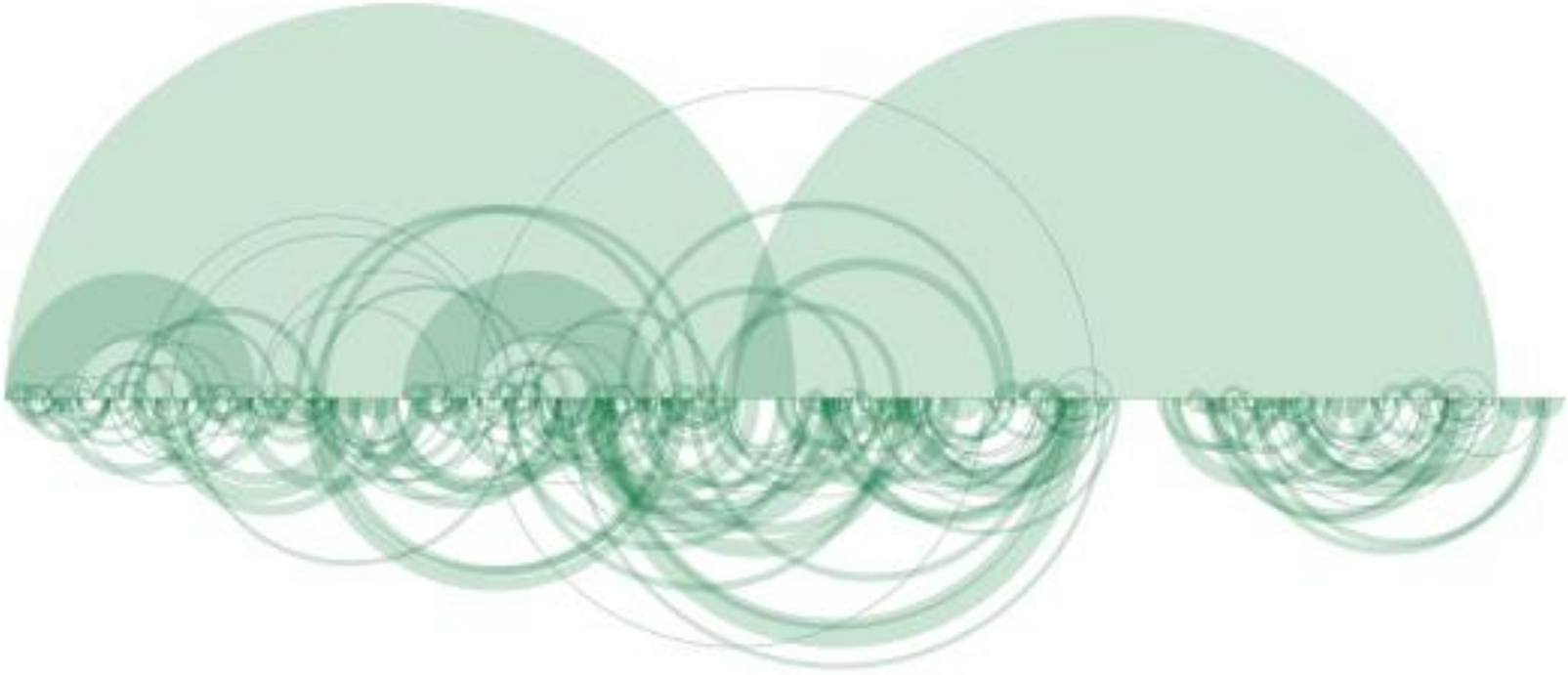


Smith Williams – A History of Science



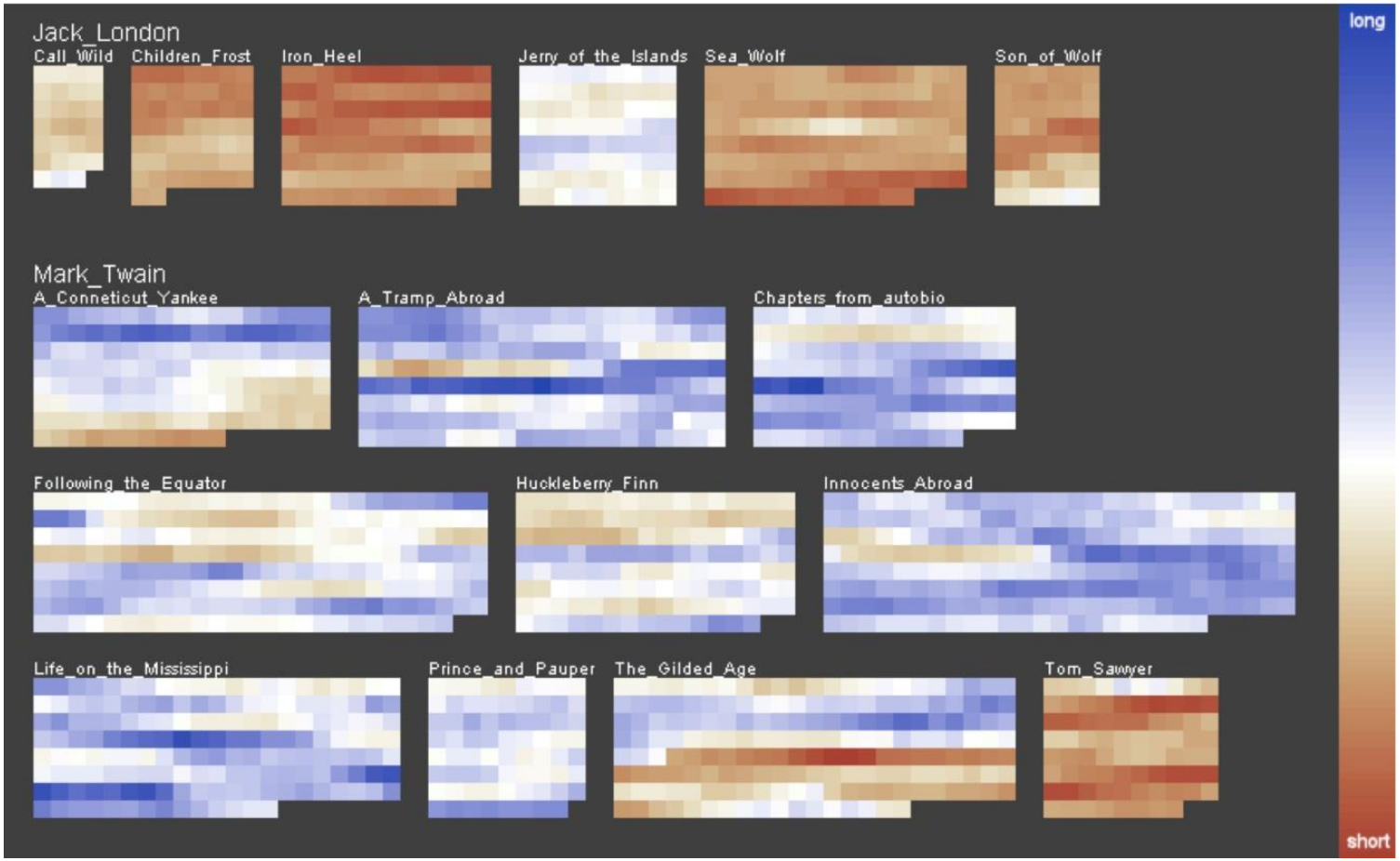
Visualization of Individual Documents

- Arc Diagrams
  - visualizing repeated occurrences in text
- Bach's minuet in G dur:



Visualization of Individual Documents

- Literature Fingerprints



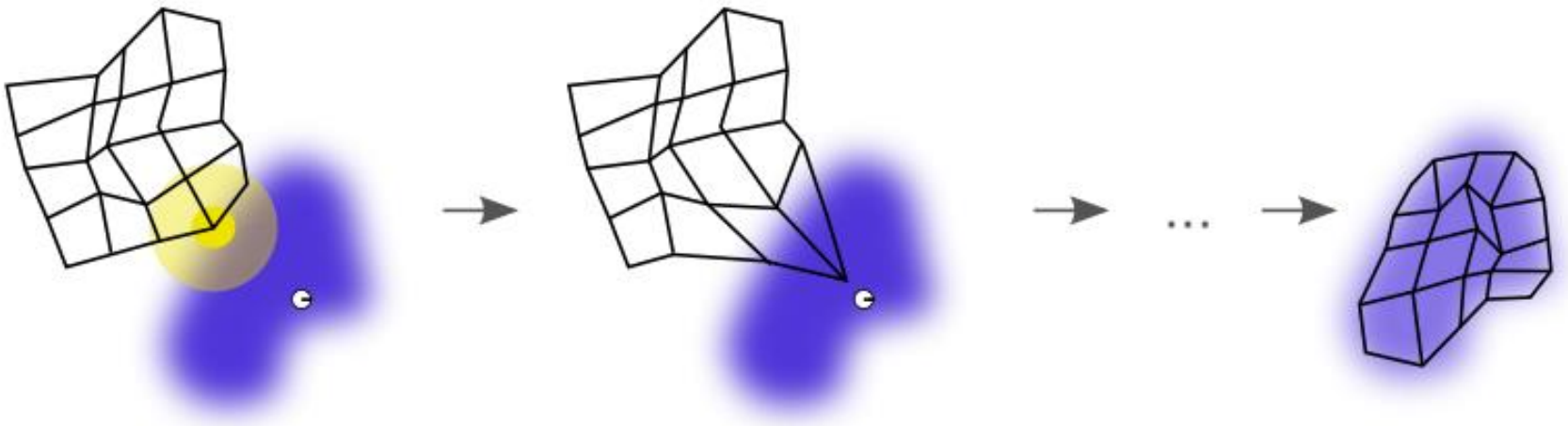
Visualization of Set of Documents

- The goal is to place similar documents close to each other and different documents far apart
- Algorithm computes similarity between all pairs of documents and this then drives their placement – $O(n^2)$ complexity

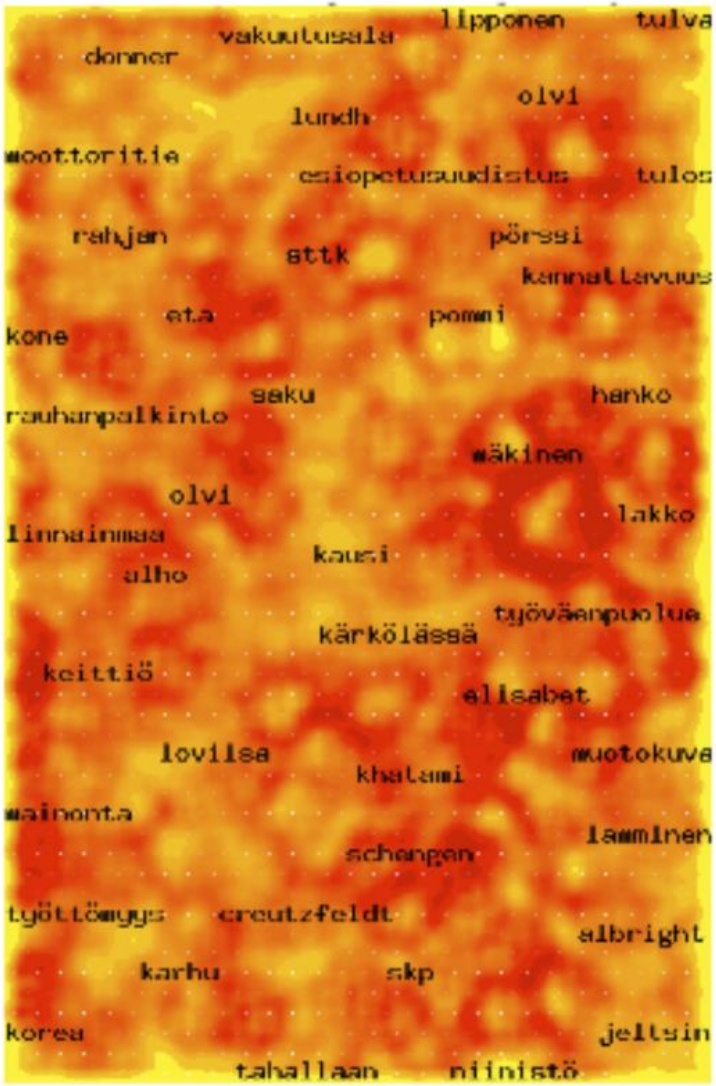
Visualization of Set of Documents

- Self-Organizing Maps – machine learning algorithm. Collection of 2D nodes, into which we place the documents. Each node contains vector of the same dimensionality as the input vectors of the training documents
- At the beginning initialize the SOM nodes – typically randomly. Select random vector, compute its distance from the other nodes. Assign weights to the closest nodes (within given radius) – the closest node has the largest weight. Iterate over input vectors while decreasing the radius.

Self-Organizing Maps

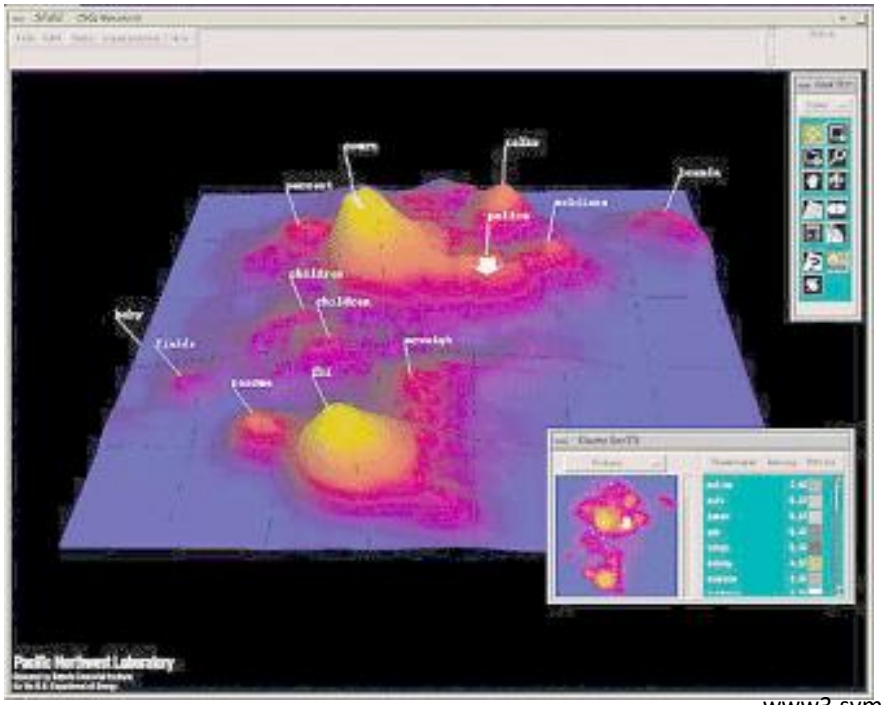


Self-Organizing Maps - Example



Visualization of Set of Documents

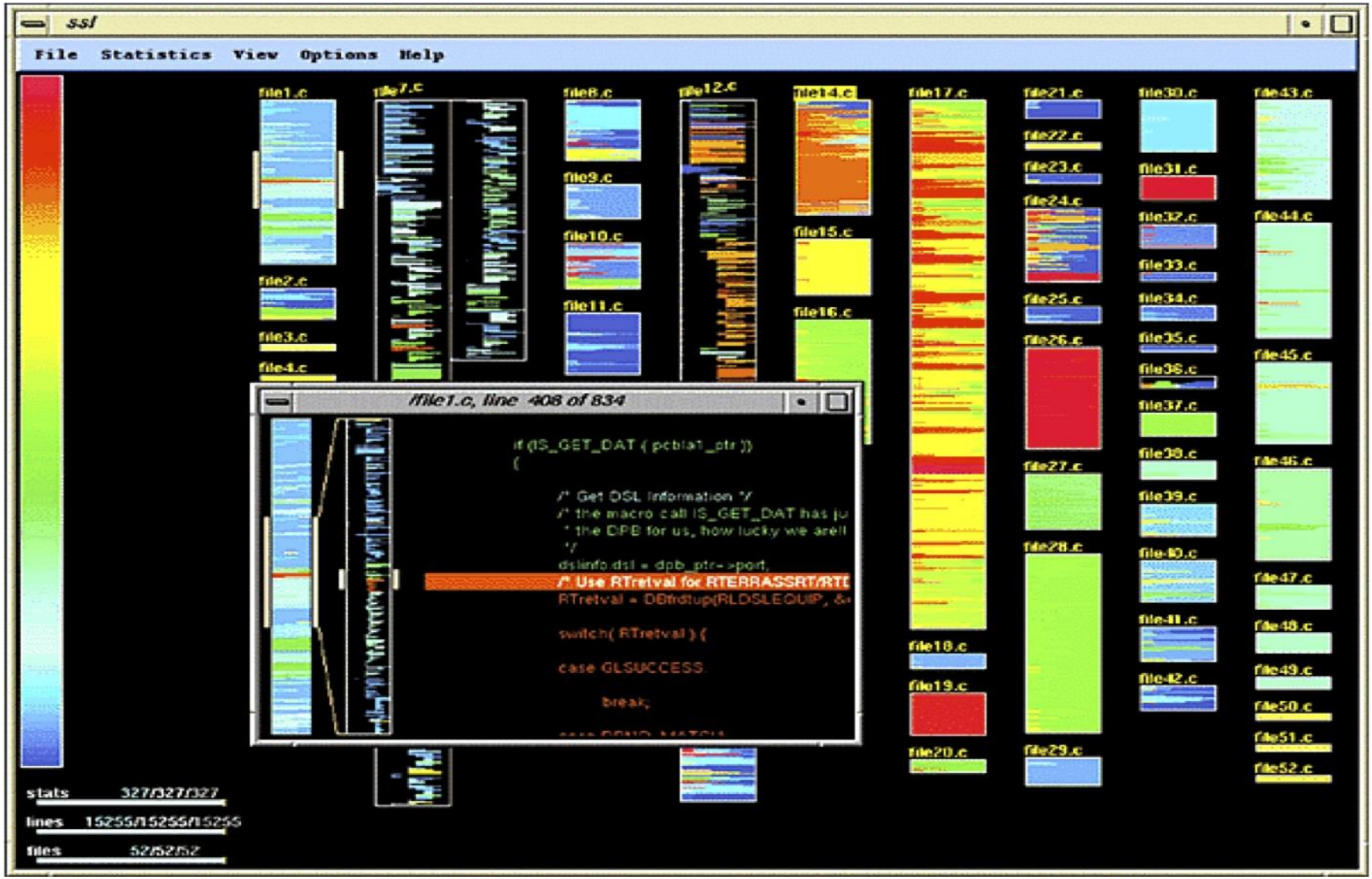
- Themescapes – summary information about corpora in the form of 3D terrains
- Height and color used for representation of density of similar documents



Extended Methods for Text Visualization

- Include also metadata:
  - Software Visualization
  - Search Result Visualization
  - Temporal Document Collection Visualizations
  - Representing Relationships

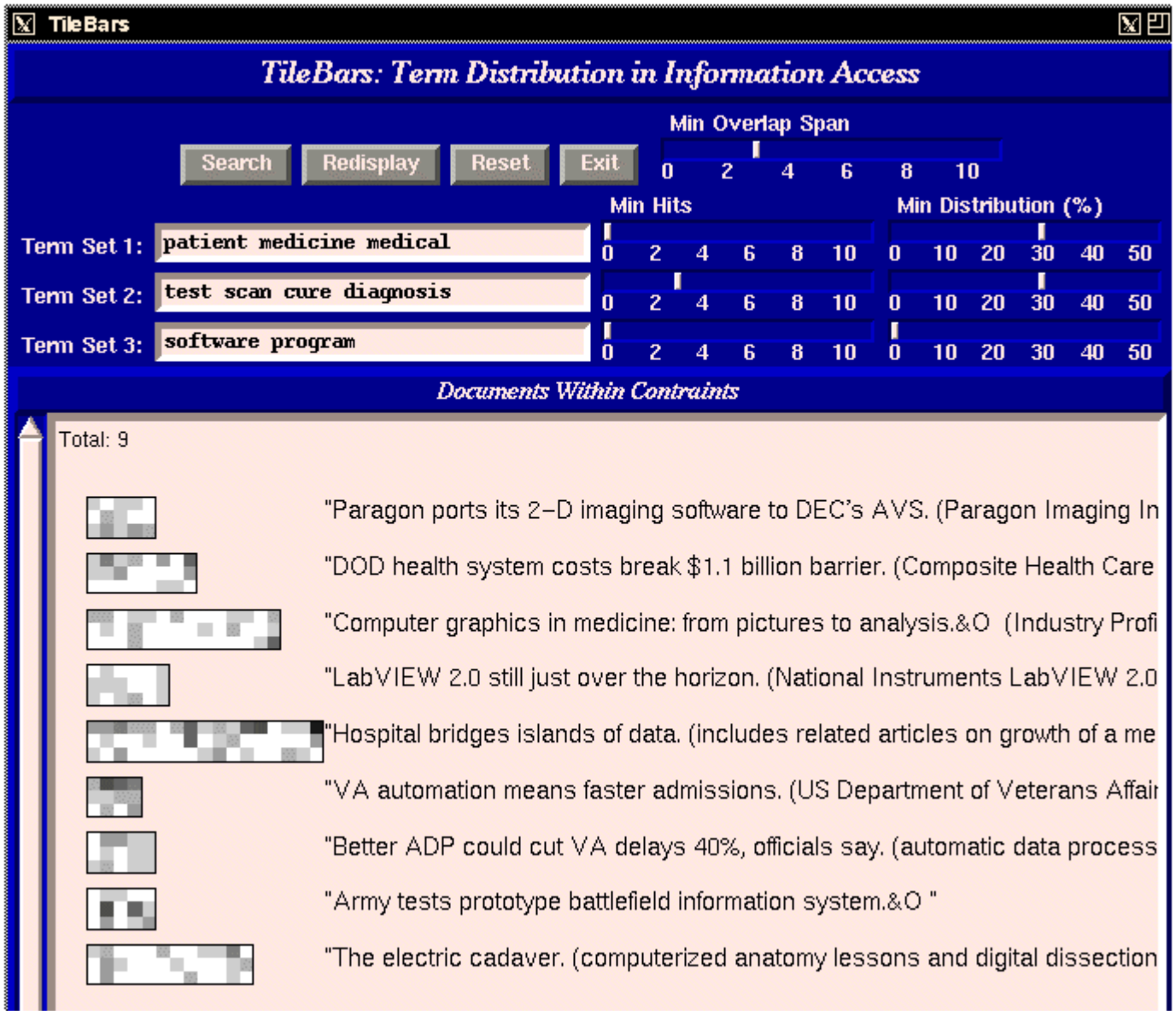
Software Visualization



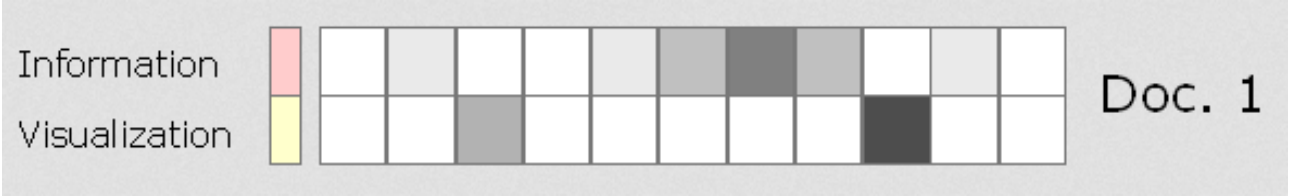
Search Result Visualization

- TileBars – each document of the result set is represented by rectangle where width indicates relative length of the document and layered squares inside correspond to text segments. The darker the square the higher the frequency of the queried set of terms
- Compact representation and information about the structure of the document reflecting relative length of the document, frequency of the queried terms and their distribution

Search Result Visualization



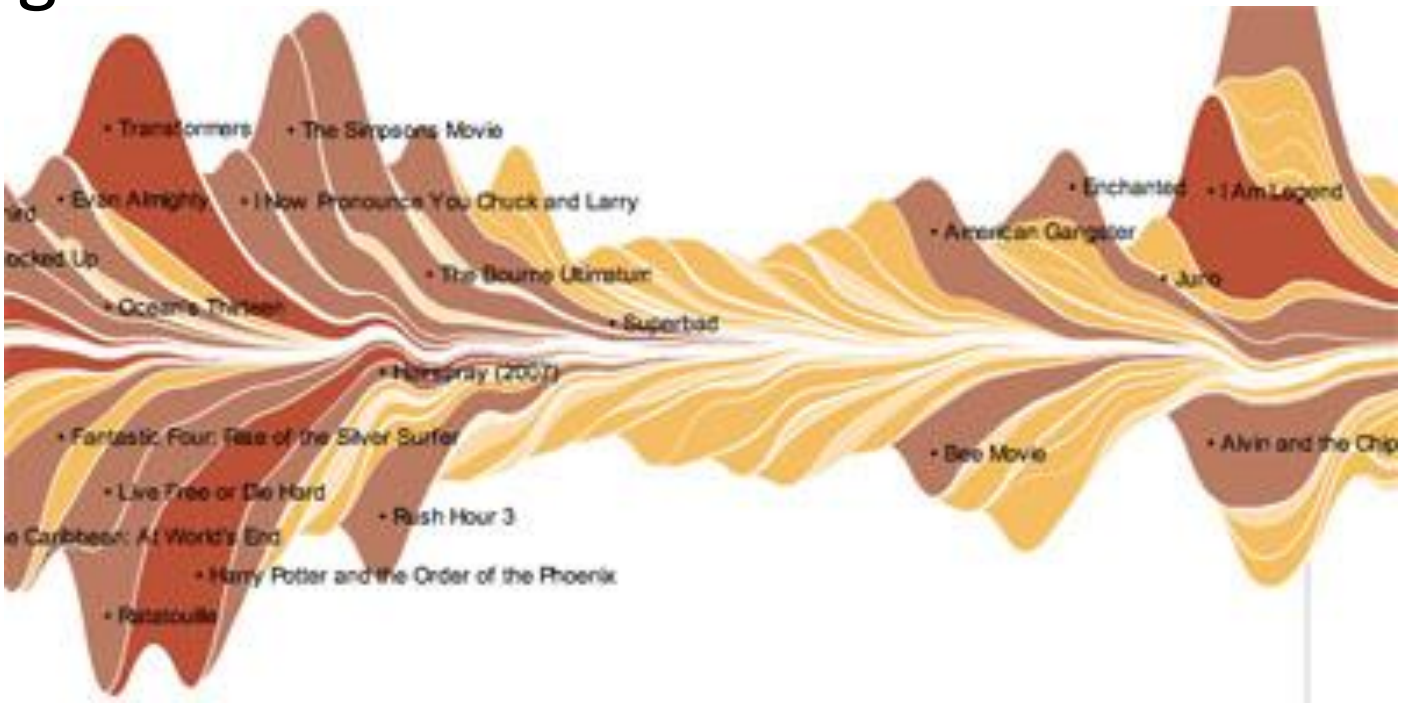
Example


Information
Visualization
Doc. 1


Information
Visualization
Doc. 2

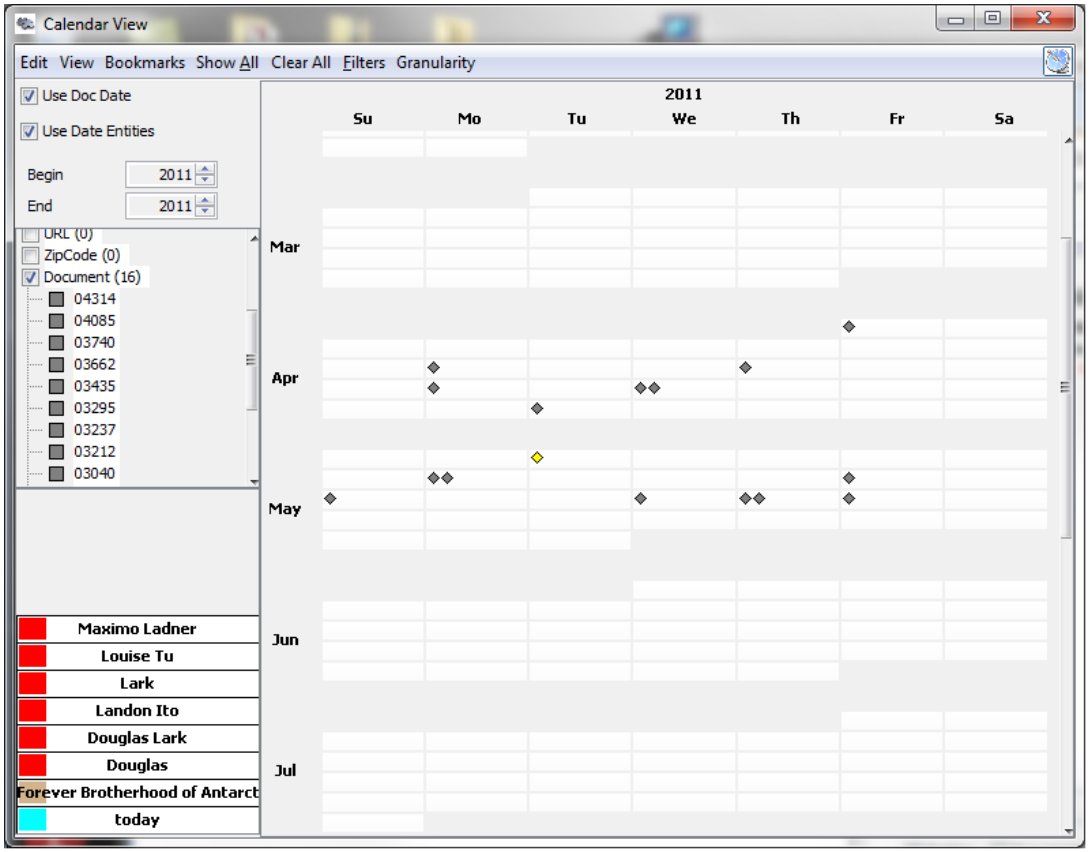
Temporal Document Collection Visualizations

- ThemeRiver – visualization of thematic changes in time for a set of documents, vertical thickness corresponds to frequency in the given time



Temporal Document Collection Visualizations

- Jigsaw calendar view – visualization and investigation of text corpora using calendar layout
- Newspaper articles



Representing Relationships

- Jigsaw List view