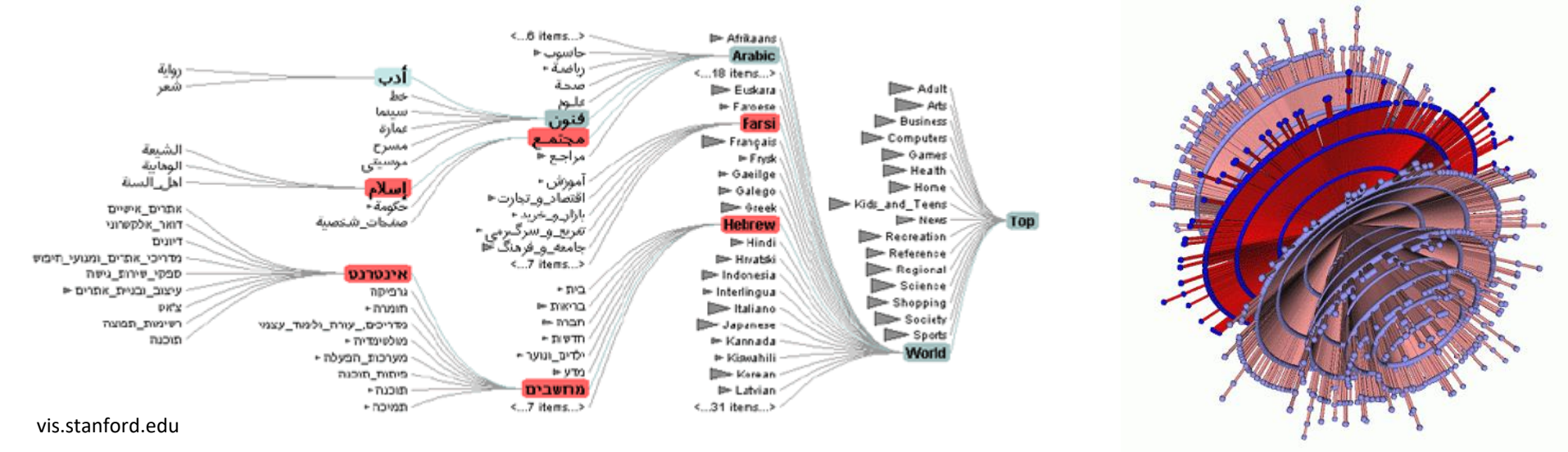
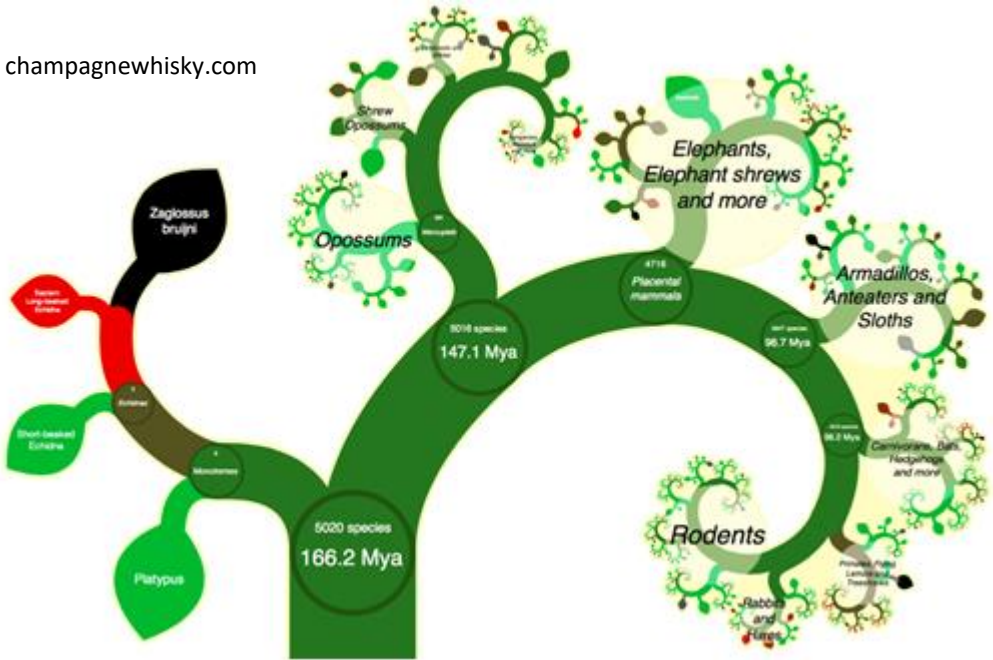


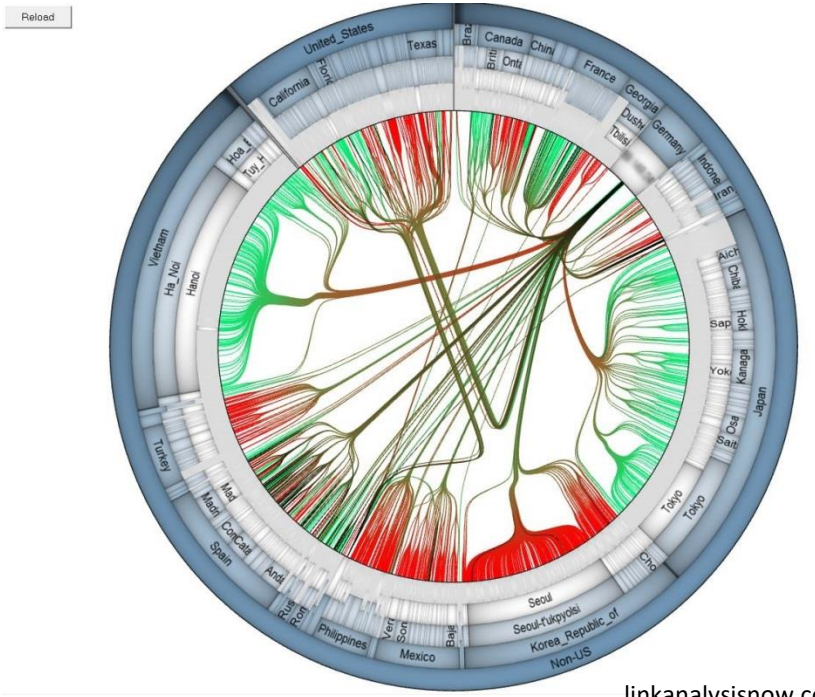
vis.stanford.edu

www.ibiblio.org

7. Stromy, grafy, sítě

champagnewhisky.com

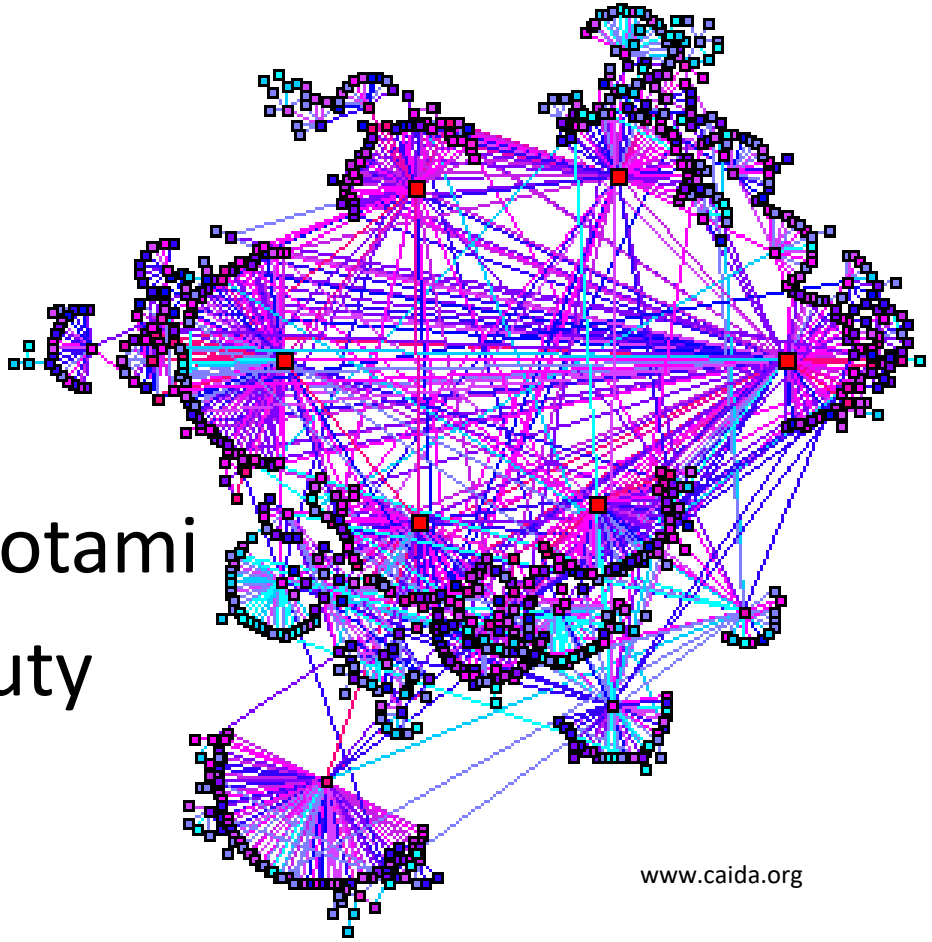




linkanalysisnow.com

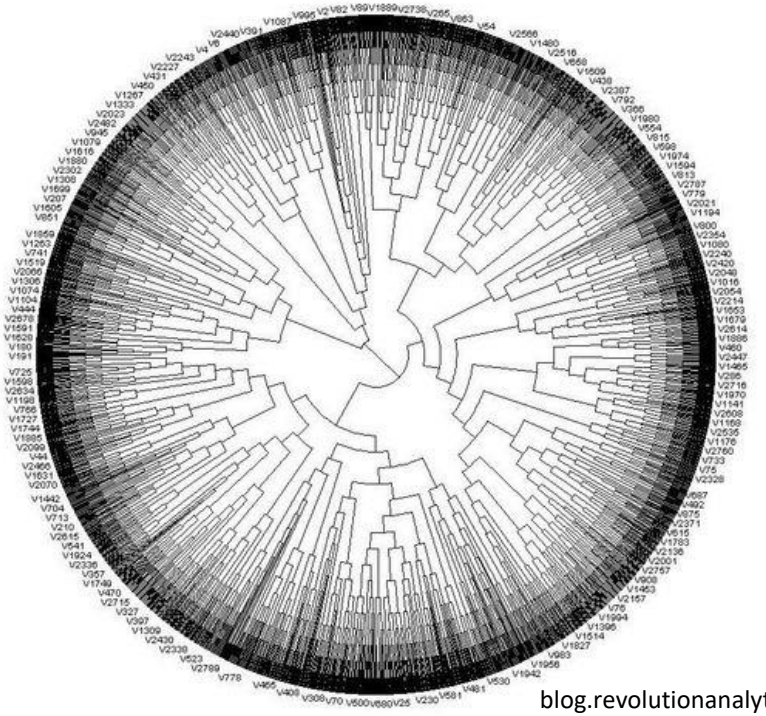
Vztahy mezi daty

- Hierarchické vztahy
- Propojenost, spojitost
- Odvození (sekvence)
- Sdílená klasifikace
- Podobnost mezi hodnotami
- Podobnost mezi atributy



Zobrazení hierarchických struktur

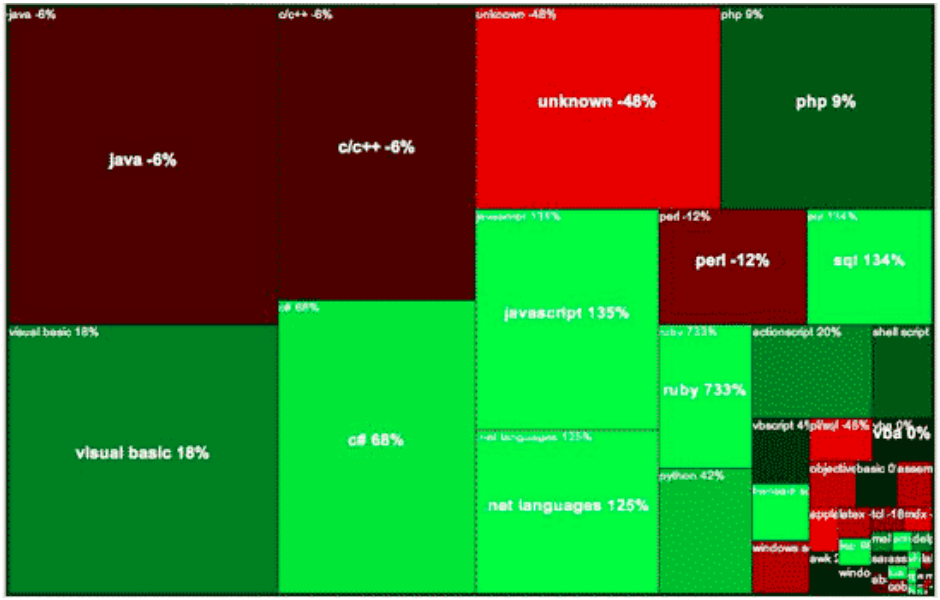
- Hierarchické struktury = stromy
- Algoritmy pro vizualizaci:
  - Space-filling – maximální využití prostoru obrazovky
  - Non-space-filling

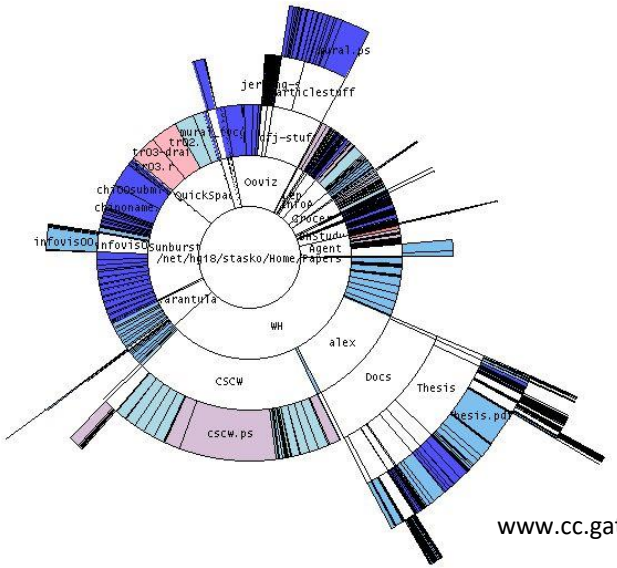


blog.revolutionanalytics.com

Space-filling metody

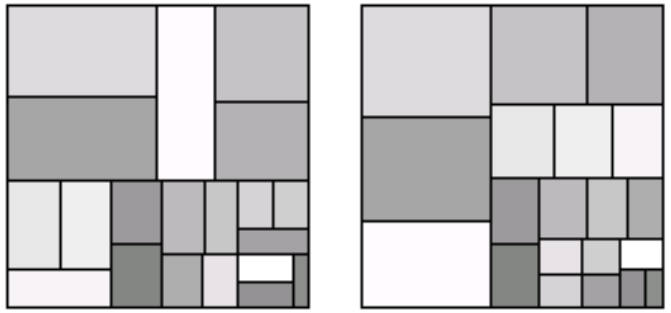
- Dva nejběžnější typy:
  - Obdélníkové rozložení
    - Treemaps
  - Radiální rozložení
    - Sunburst zobrazení



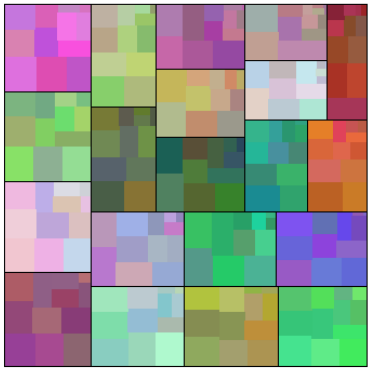


Obdélníkové rozložení

- Treemaps
  - V základní variantě jde o rekurzivní dělení obdélníka střídavě pomocí horizontálních a vertikálních čar
  - Různé varianty, např.
    - Squarified treemaps
    - Vnořené (nested) treemaps



hcil2.cs.umd.edu



www.mathworks.com

Treemap - pseudokód

Označení:

Width = šířka obdélníka

Height = výška obdélníka

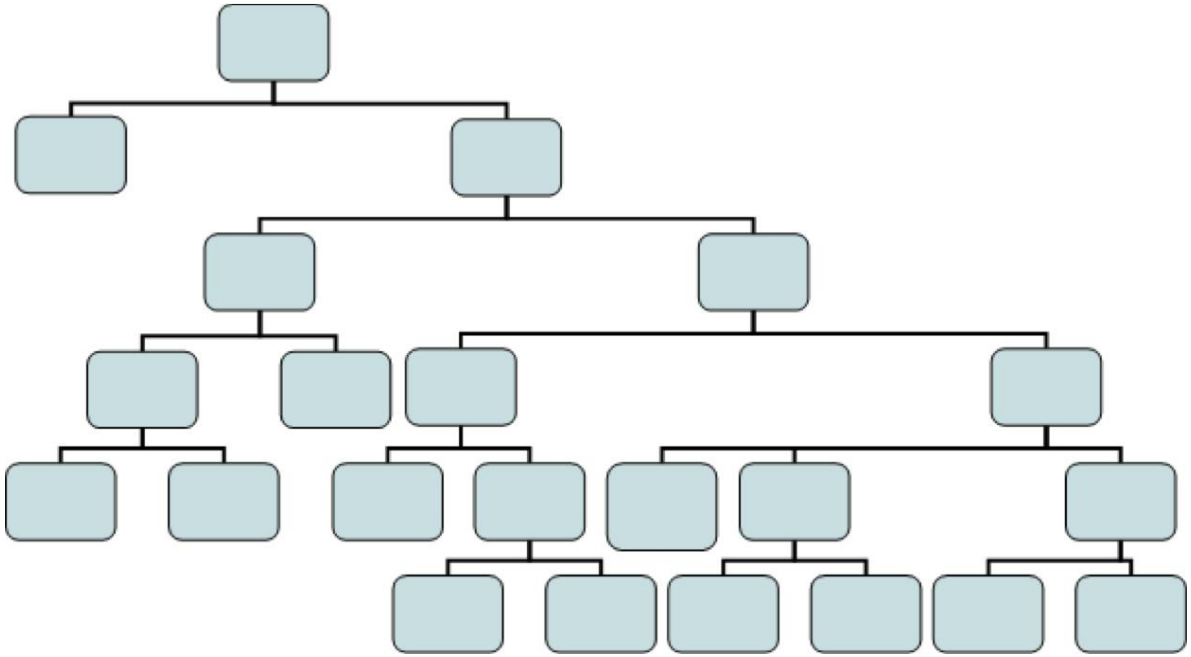
Node = kořenový uzel stromu

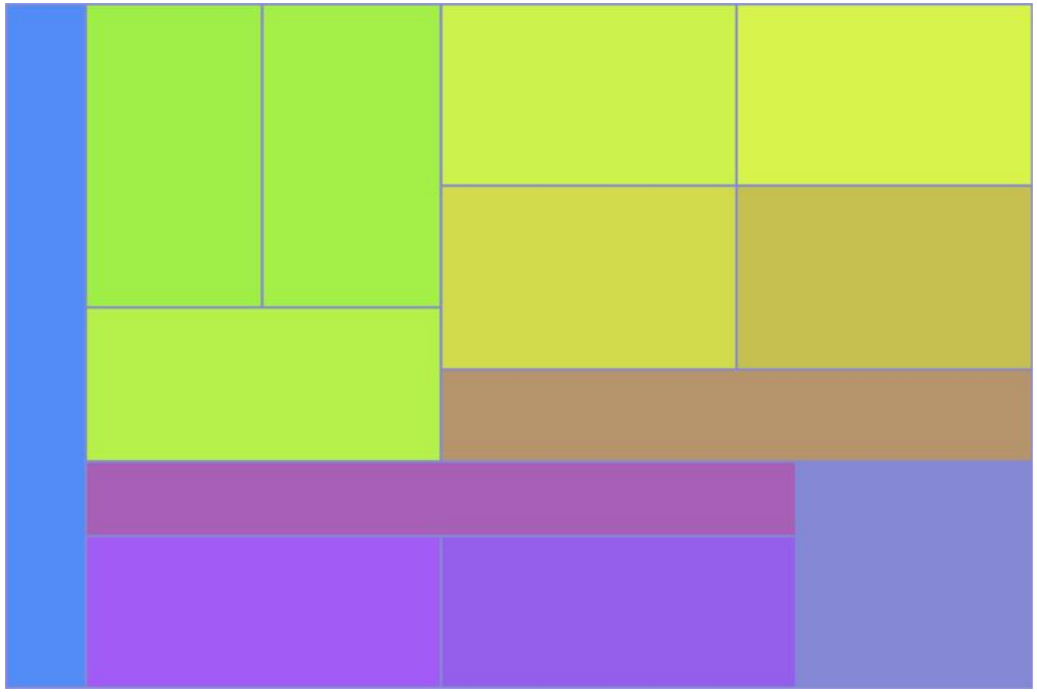
Position = pozice obdélníka (např. [0, 0])

Orientation = směr řezů – střídání horizontálního a vertikálního řezu

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

```
treemap(Node n, Orientation o, Position orig, Width w, Height
h)
   if n je koncový uzel (nemá potomky)
       vykresli_obdélník(orig, w, h);
       return;
   for each potomek uzlu n(child_i) získej počet koncových
       uzlů v podstromu;
   sečti počet koncových uzlů;
   spočti procentuální poměr koncových uzlů v každém
       podstromu(percent_i);
   if orientace je horizontální
       for each podstrom
          spočti offset počátku na základě počátku a
              šířky (offset_i);
          treemap(child_i, vertical, orig + offset_i,
              w * percent_i, h);
   else
       for each podstrom
          spočti offset počátku na základě počátku a výšky
              (offset_i);
          treemap(child_i, horizontal, orig + offset_i,
              w, h * percent_i);
```





Radiální rozložení

- Sunburst displays
- Kořen hierarchie umístěn ve středu radiálního zobrazení, jednotlivé vrstvy reprezentovány soustřednými prstenci
- Prstence rozděleny na základě počtu uzlů v dané úrovni
- Radiální techniky zobrazují rovněž vnitřní uzly

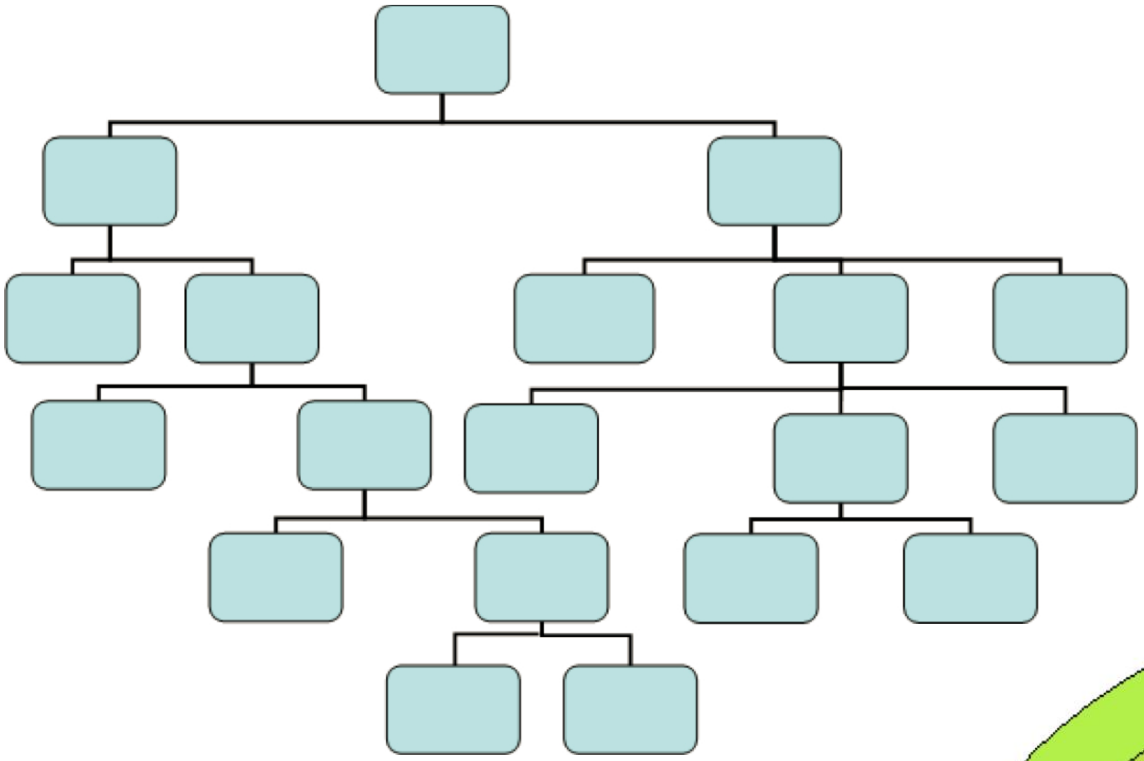
Sunburst - pseudokód

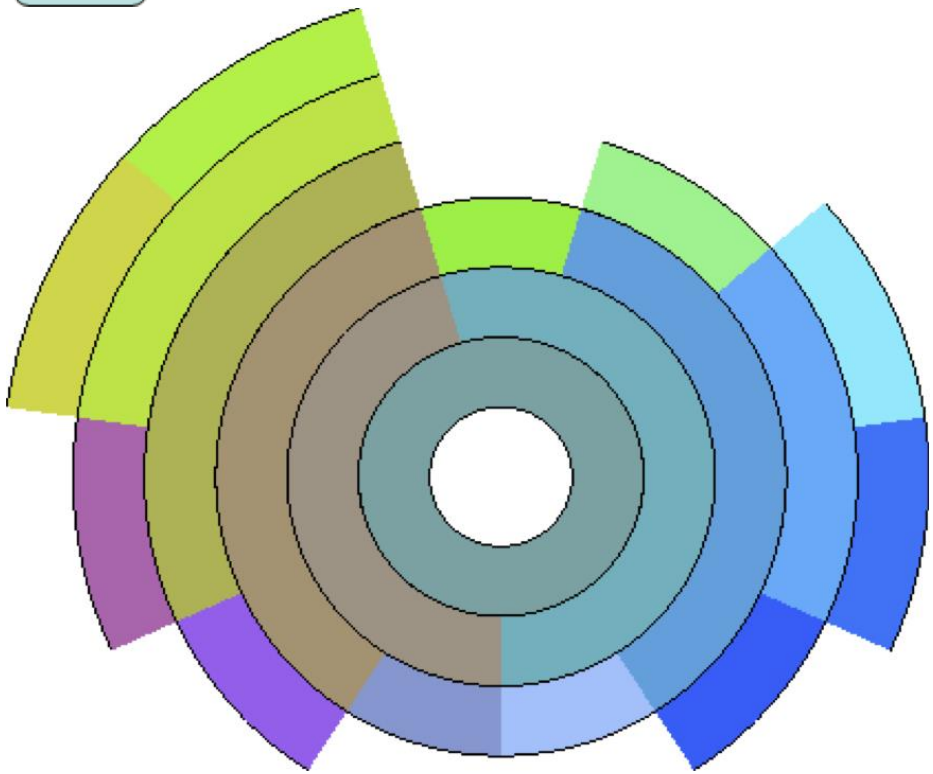
Označení:

```
Start = počáteční úhel uzlu (iniciálně 0)
End = koncový úhel uzlu (iniciálně 360)
Origin = pozice středu radiálního zobrazení (např.
      [0, 0])
Level = aktuální stupeň hierarchie (iniciálně 0)
Width = tloušťka každého radiálního pásu – založena
na maximální hloubce a velikosti displeje
```

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

```
sunburst(Node n, Start st, End en, Level l)
   if n je koncový uzel (nemá  potomky)
       vykresli_radiální_sekci(Origin, st, en, l * Width,
          (l+1) * Width);
       return;
   for each potomek n(child_i) získej počet koncových uzlů
       v podstromu;
   sečti počet koncových uzlů;
   spočti procentuální poměr koncových uzlů v každém
       podstromu(percent_i);
   for each podstrom
       spočti počáteční/koncový úhel na základě velikosti
          podstromů, jejich uspořádání a rozsahu úhlů;
       sunburst(child_i, st_i, en_i, l+1);
```





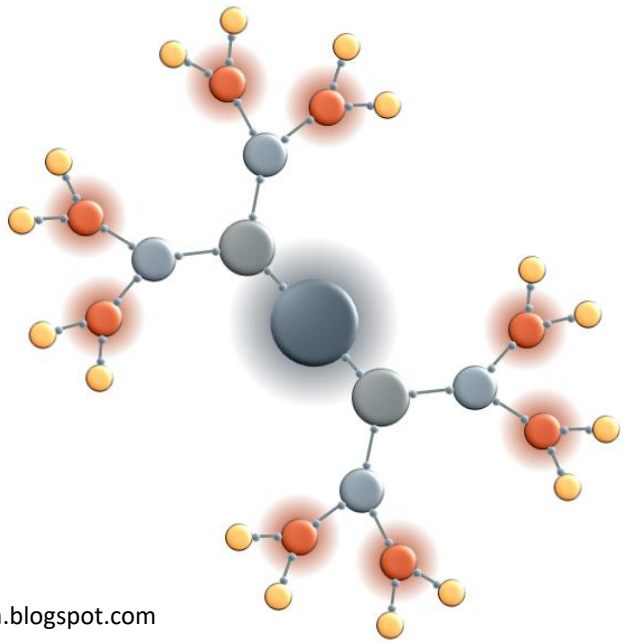
Využití barvy u hierarchických technik

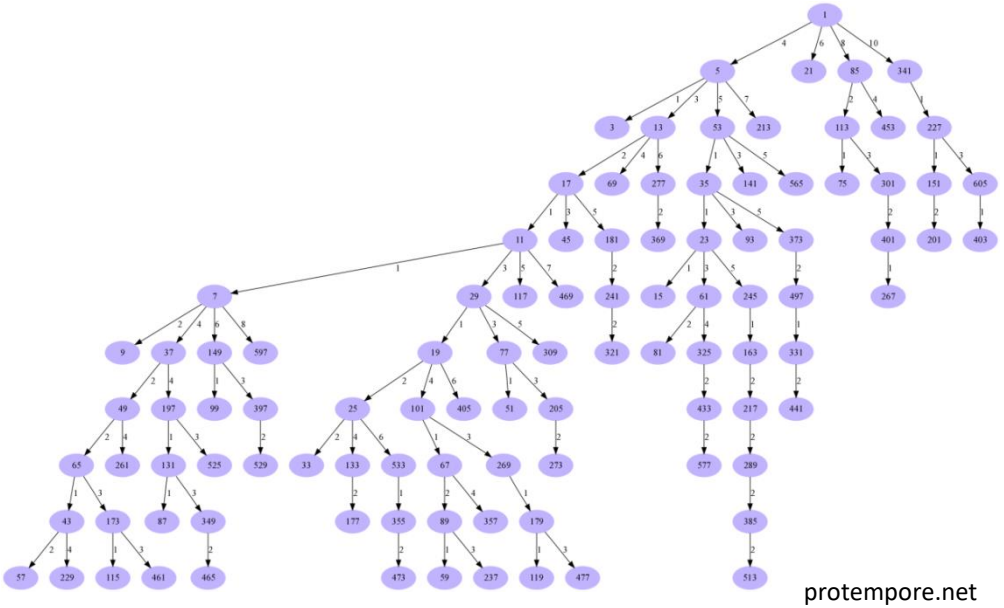
- Zvýraznění mnoha atributů, např.
  - Hodnoty v uzlech
  - Zesílení hierarchických vztahů (podobná barva rodičů a potomků)

- Další vlastnosti zobrazeny pomocí symbolů a označení umístěných do obdélníkových či kruhových segmentů

Non-space-filling metody

- Nejběžnější jsou spojové (node-link) diagramy
- Zobrazení je nejvíce ovlivněno dvěma faktory:
  - **Stupněm větvení** (např. binární stromy)
  - **Hloubkou**





swamitra.blogspot.com

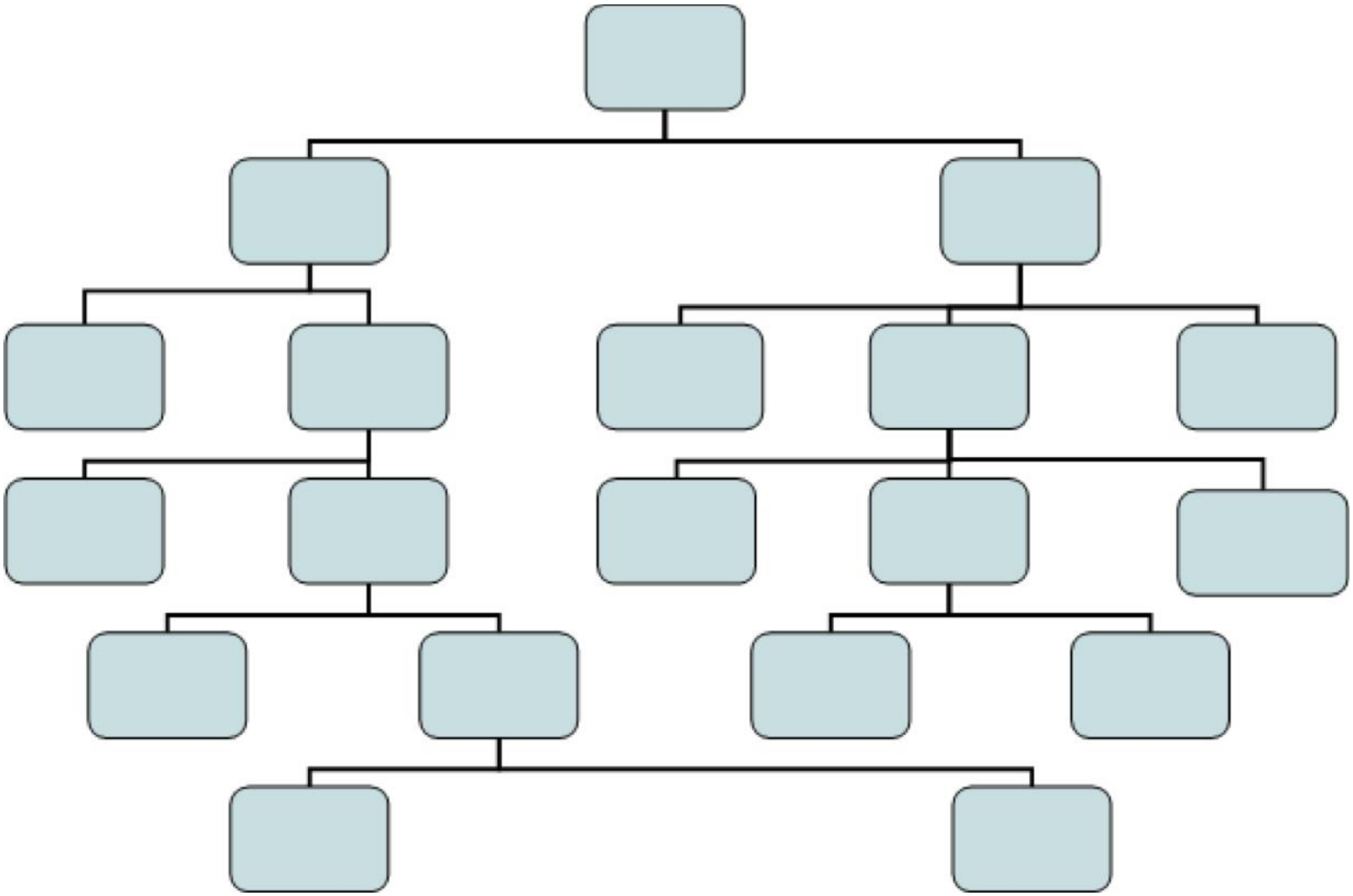
protempore.net

Návrh algoritmů pro vykreslení spojového diagramu

- Tři kategorie pravidel:
  - **Konvence vykreslování** – tvar a zakřivení hran, umístění uzlů do fixní mřížky, sourozenci na stejné vertikální pozici, …
  - **Omezení** – umístění uzlů na danou pozici, zobrazení uzlů v těsné blízkosti, orientace čar, …
  - **Estetika** – minimalizace křížení čar, udržení poměru šířky a výšky obrazu, minimalizace celkové plochy obrazu, minimalizace délky hran, minimalizace počtu ohybů hran, minimalizace počtu různých úhlů a zakřivení, snaha o udržení symetrie

Návrh algoritmů pro vykreslení spojového diagramu

1. Rozdělit plochu pro vykreslení do plátů o stejné výšce – toto rozdělení je odvozeno od hloubky stromu
2. Pro každou úroveň stromu určit, kolik uzlů musí být vykresleno
3. Rozdělit každý plát na obdélníky stejné velikosti – odvozeno od počtu uzlů v dané úrovni
4. Vykreslit každý uzel do středu odpovídajícího obdélníka
5. Vykreslit spojovací čáru vedoucí ze středu spodní hrany každého uzlu do středu horní hrany jeho potomka(ů)

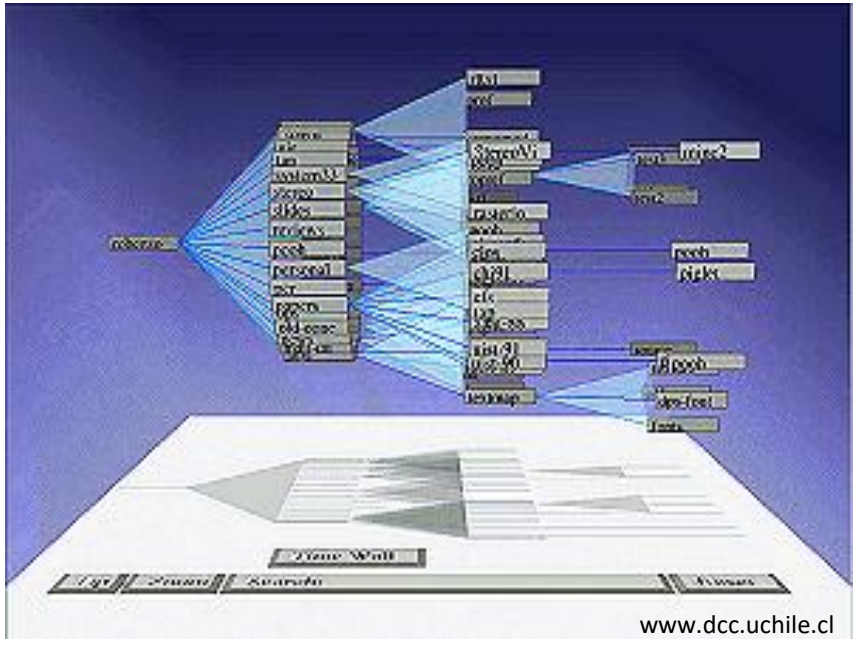


Optimalizace

- Zlepšují využití prostoru obrazovky
- Příklady:
  - Každá úroveň rozdělena podle počtu koncových uzlů v odpovídajícím podstromu
  - Rovnoměrné rozložení koncových uzlů + vystředění jejich rodičovských uzlů
  - Přidání dodatečných mezer mezi sousední uzly, které nejsou sourozenci
  - Reorganizace stromu za účelem zlepšení symetrie a vyvážení
  - Kořenový uzel ve středu obrazovky, potomci radiálně kolem

Využití třetí dimenze

- Pro velké stromy, doplněno o rotaci, translaci a zoomování
- Příkladem jsou **cone trees** – potomci jsou rovnoměrně radiálně rozloženi kolem uzlu
  - Zásadní parametry jsou poloměr a vzdálenost posunutí



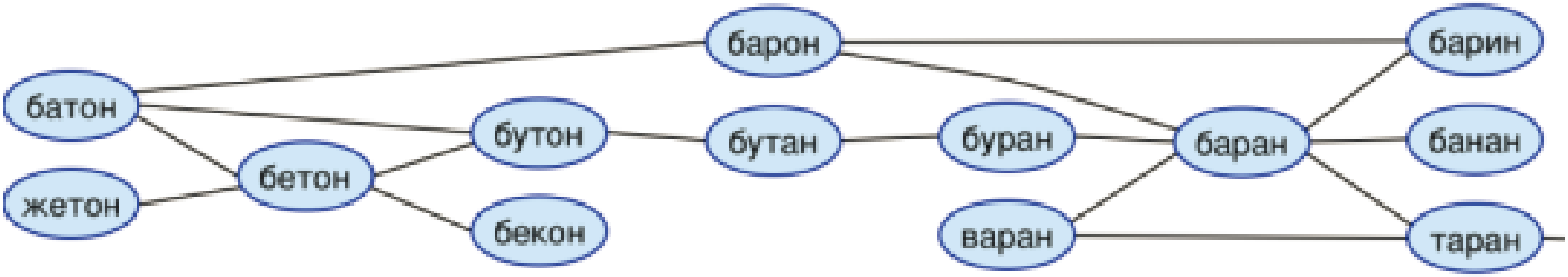
www.dcc.uchile.cl

Zobrazování libovolných grafů/sítí

- Stromy jsou pouze jedním typem **grafů** – spojové nevážené acyklické grafy
- Existuje řada různých reprezentací, např. grafy s váženými hranami, neorientované grafy, grafy s cykly, nesouvislé grafy, …
- Zaměříme se na **arbitrary graphs** – není známa jejich struktura. Dva odlišné přístupy:
  - **Node-link diagrams** (spojové diagramy)
  - **Matrix displays** (maticová zobrazení)

Node-link graphs

- Force-directed grafy – struny mezi uzly
- Spoje mezi uzly jsou iterativně upravovány, dokud není minimalizována hodnota stress



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

Node-link graphs

- Pro každou dvojici spojených uzlů jsou spočteny dvě síly:
- f_{ij} = síla odvozena od struny mezi nimi (Hookův zákon)

$$f_{ij}(x) = k_{ij} * (d(i,j) - s_{ij}) * (x_i - x_j) / d(i,j)$$

kde d(i,j) je Euklidovská vzdálenost uzlů i, j; s_{ij} je délka struny; k_{ij} je napnutí struny. Vzorec počítá x-ovou komponentu síly f_{ij}.

- g_{ij} = elektrický odpor zabraňující uzlům dostat se příliš blízko sebe (zákon převrácených čtverců)

Node-link graphs

- r_{ij} = velikost odporu mezi uzly *i* a *j*, pak x-ová komponenta síly g_{ij} je spočtena jako

$$g_{ij}(x) = (r_{ij} / d(i, j)^2) * (x_i - x_j) / d(i, j)$$

- V každém kroku je spočten součet sil pro každý uzel a úměrně součtu je uzel posunut
- Cílem je dosáhnout globálního energetického minima

Planární (rovinné) grafy

- Hrany se neprotínají
- Velmi populární
  - Dlouhá historie, řada studií
  - Křížení hran komplikuje interpretaci grafu, lepší se jim vyhnout
  - Jsou řídké – podle Eulerovy věty pro *n* vrcholů existuje nejvýše *3n – 6* hran
- Grafy s křížením mohou být převedeny na planární graf pomocí „falešných“ uzlů v průsečících a po provedení algoritmu pro planární rozvržení pozic uzlů jsou falešné uzly odstraněny

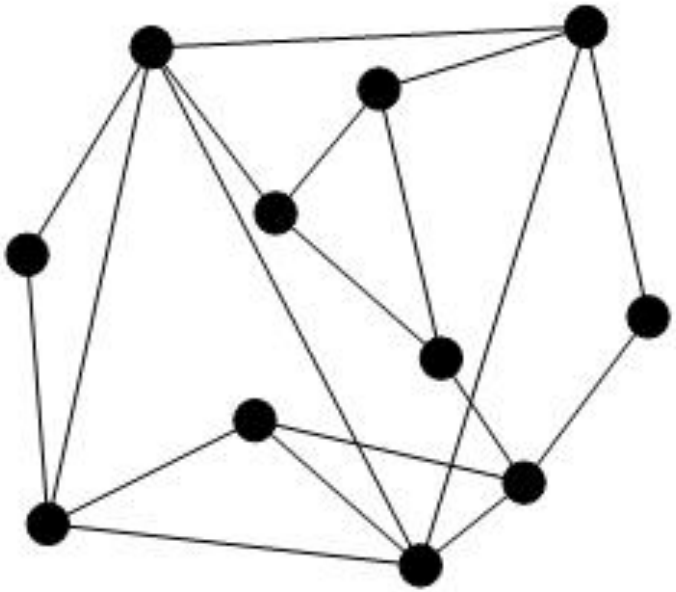
Planární (rovinné) grafy

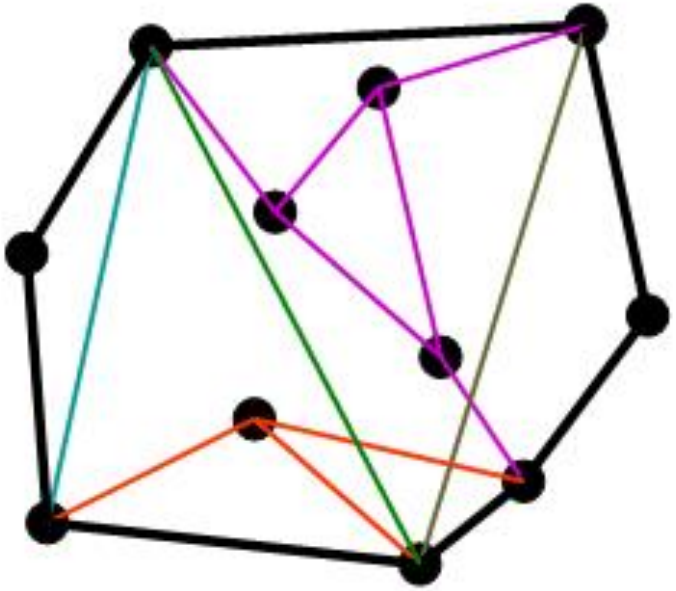
- Musíme určit strategii, podle které rozhodneme, zda je graf planární – algoritmy jsou buď složité nebo výpočetně náročné
- Zjednodušení:
  - Graf je planární, pokud všechny jeho spojité komponenty jsou rovněž planární
  - Spojitý graf je planární, pouze pokud všechny biconnected komponenty jsou planární
- Stačí tedy algoritmus pro určení, zda biconnected graf je planární

Algoritmus pro detekci planarity biconnected grafu

- Přístup rozděl a panuj
- Obsahuje-li graf cyklus a nemá v sobě žádný jiný cyklus neobsahující hranu původního cyklu, pak po odstranění tohoto cyklu zůstanou cesty (kusy) začínající a končící v jednom z vrcholů cyklu
- Pokud se cesty protínají, musí být jedna vykreslena uvnitř cyklu a jedna vně
- Vytvoříme-li graf cest, kdy protínající se cesty jsou odděleny hranou a takovýto graf je bipartitní (rozdělitelný do dvou sad vrcholů tak, že mezi příslušníky dané sady neexistuje žádná hrana), pak původní graf je planární

Algoritmus pro detekci planarity biconnected grafu

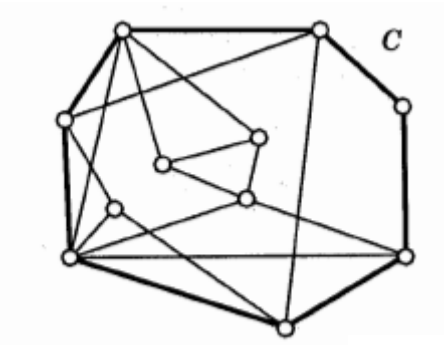


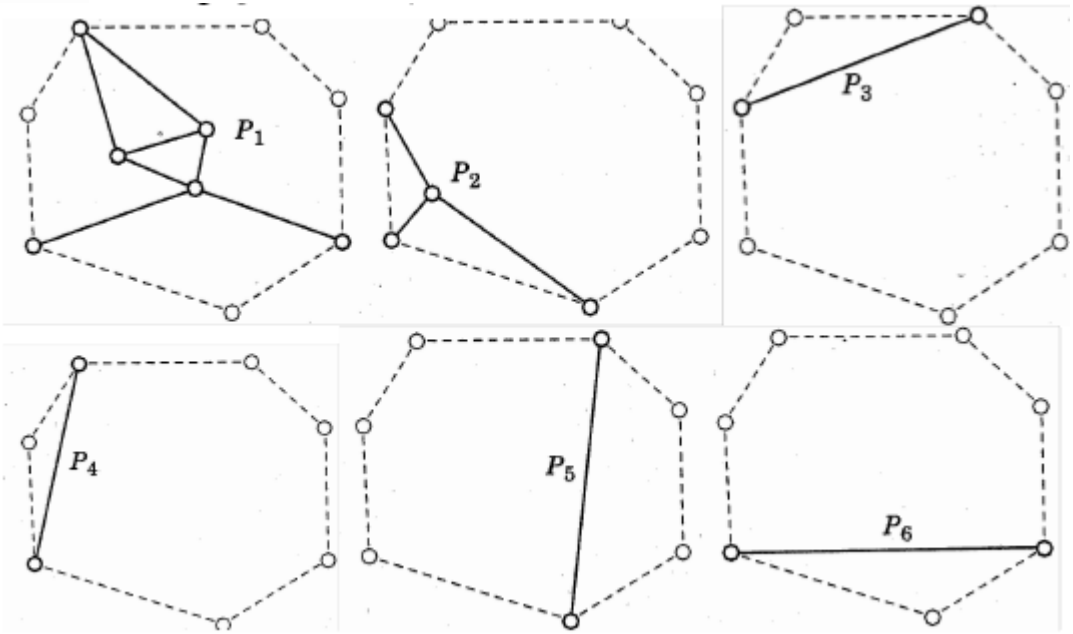


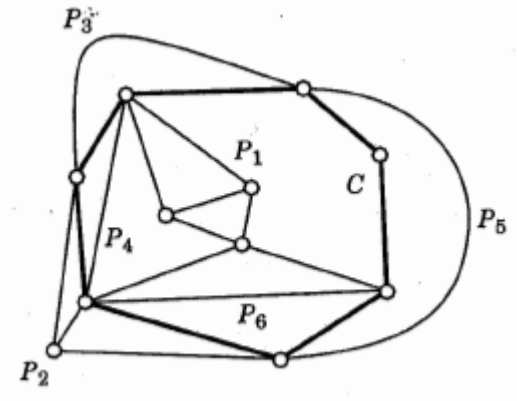
Algoritmus pro detekci planarity biconnected grafu

- Obsahuje-li graf po odstranění hran původního cyklu stále cykly, znamená to, že jeden nebo více kusů obsahuje cyklus
- Vytvoříme podgraf obsahující tento kus a část původního cyklu spojující koncové body
- Rekurzivně voláme algoritmus testování planarity grafu

Příklad







Algoritmus pro detekci planarity biconnected grafu

- Oddělující cyklus je takový cyklus, který generuje alespoň dva kusy

Mějme biconnected graf G a oddělující cyklus C.

1. Spočteme všechny kusy G vzhledem k C.
2. Pro každý kus P, který není jednoduchou cestou (obsahuje cyklus)
   a. Vytvoříme graf G′ skládající se z P a C.
   b. Vytvoříme cyklus C′ skládající se z cesty skrz P a z části C spojující konce.
   c. Aplikujeme tento algoritmus na (G′, C′). Pokud je výsledek nerovinný, pak je i G nerovinný.
3. Spočítáme proložení grafu I kusů G.
4. Jestliže I není bipartitní, G je nerovinný; jinak je G rovinný.

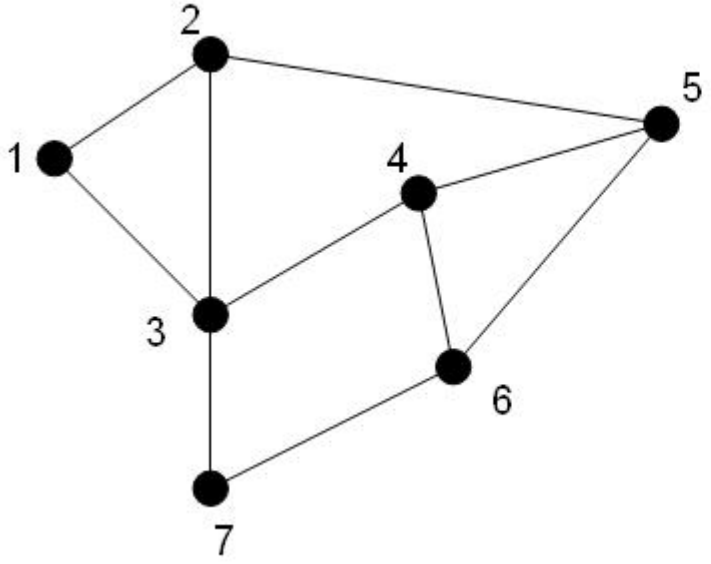
Algoritmus pro detekci planarity biconnected grafu

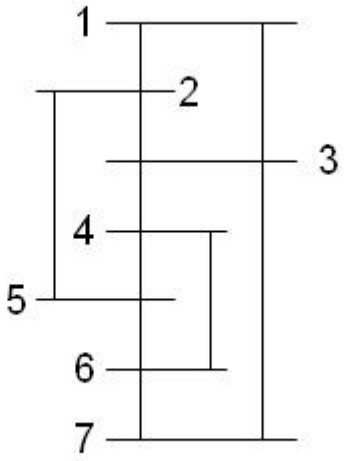
- Pokud je graf nerovinný, můžeme z něj udělat rovinný:
  1. Určíme největší rovinný podgraf původního grafu
  2. Zbývající vrcholy umístíme do plošek tak, že minimalizujeme křížení hran
  3. Pro každý průsečík hran rozdělíme odpovídající hrany na dvě části a v průsečíku vytvoříme nový „falešný“ (dummy) vrchol

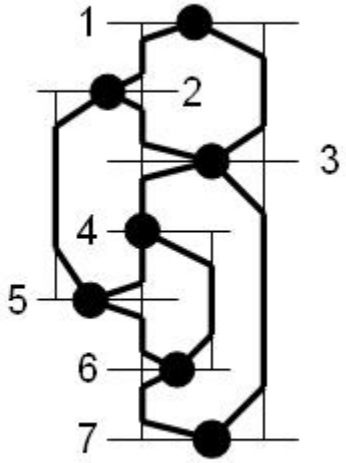
Vykreslení planárních grafů

- Visibility approach – 2 kroky:
  - **Visibility step** – vytvoříme reprezentaci, kdy je každý vrchol vykreslen jako horizontální úsečka a každá hrana je vykreslena jako vertikální čára spojující odpovídající segmenty vrcholů
  - **Replacement step** – každý segment odpovídající vrcholu se „smrskne“ na bod a vertikální spojnice jsou nahrazeny polyčárou

Vykreslení planárních grafů







Maticová reprezentace grafů

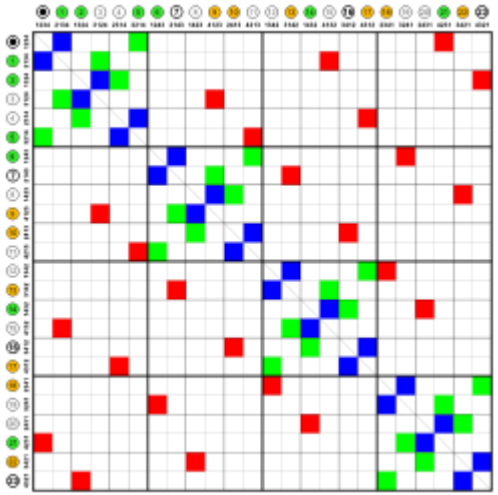
- **Matice sousednosti** = mřížka o velikosti NxN (N je počet uzlů)
  - Binární nebo obsahující hodnoty síly či váhy hrany
  - Překonává problém křížení hran
  - Důležitá je zvolená strategie pro organizace řádek a sloupců – odhalení zajímavých struktur v grafu

	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					●	●	●	

Maticová reprezentace grafů

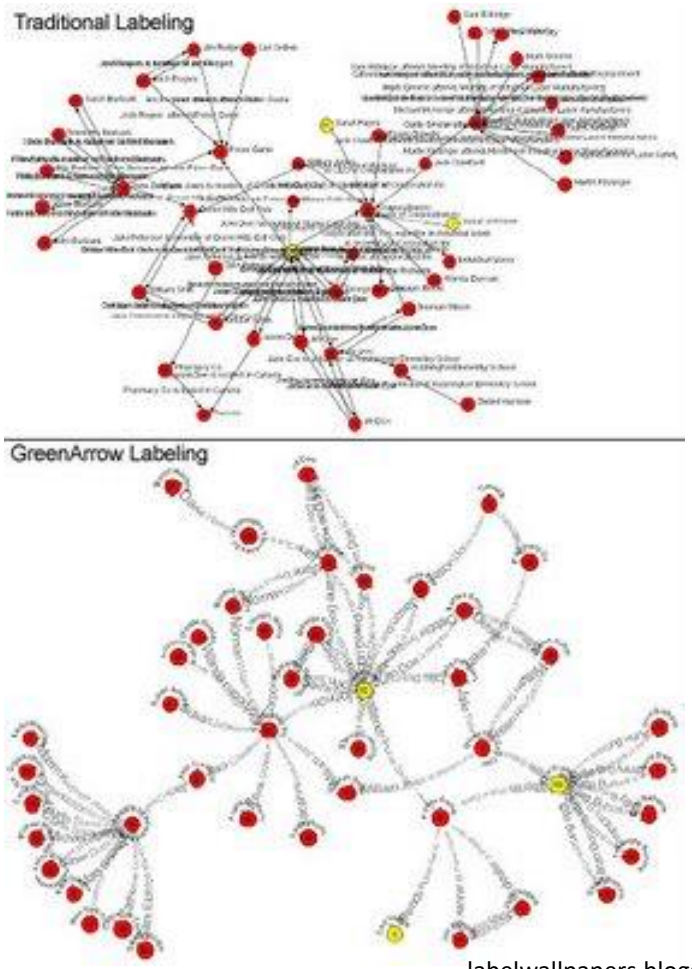
- Řada různých algoritmů pro reorganizaci řádek a sloupců
  - Řízené uživatelem x automatické
  - Nalezení optimálního řešení je NP-úplný problém, proto bylo nutné zavést řadu heuristik



en.wikipedia.org

Labeling

- Nezbytný pro pochopení, co vlastně zobrazujeme
- Při vykreslování stromů a grafů není labeling jednoduchý, protože mohou obsahovat velké množství uzlů a navíc můžeme požadovat označení hran



labelwallpapers.blogspot.com

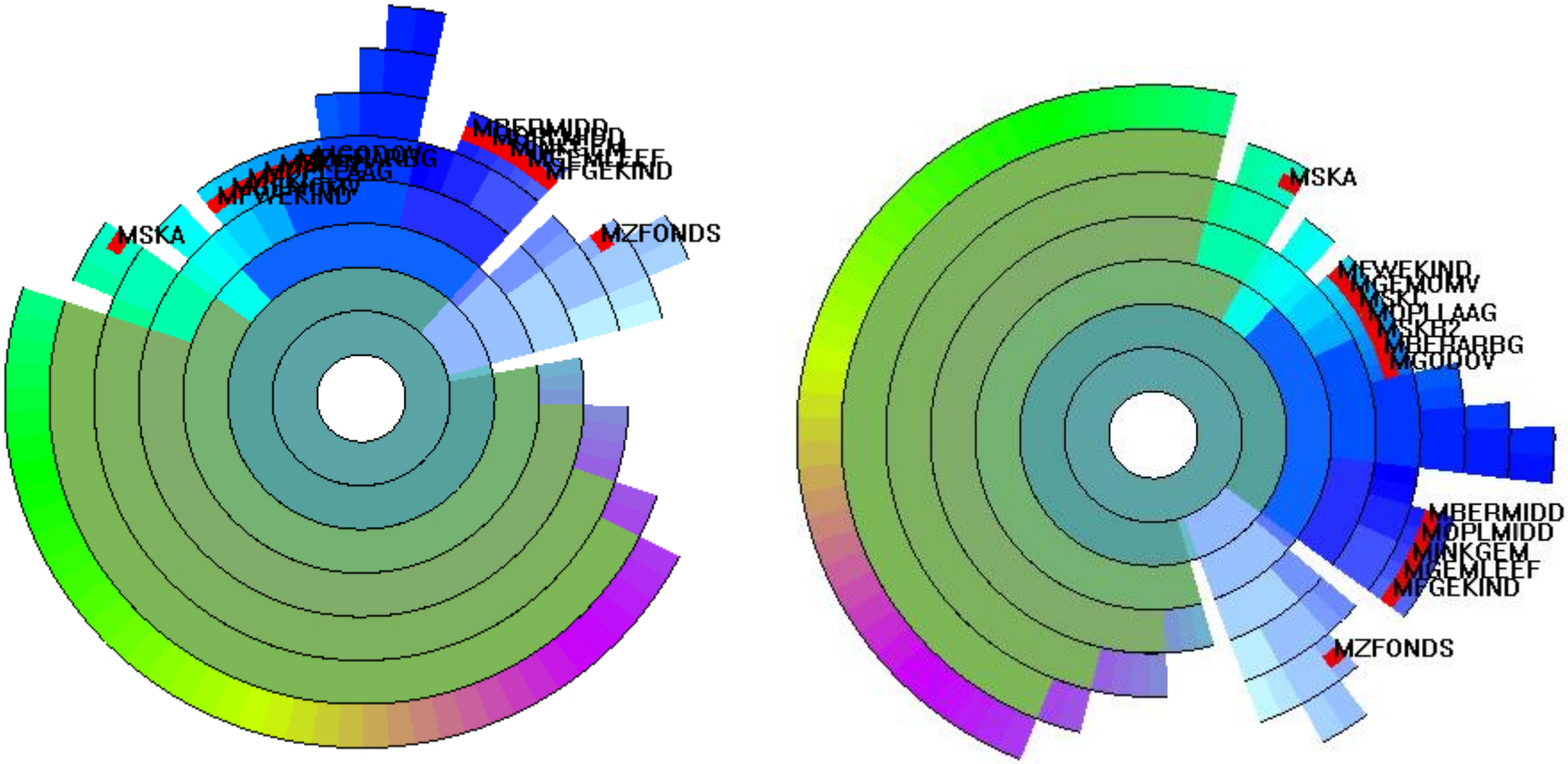
Labeling

- Při malém množství značek je lepší využít barvu, velikost, tvar uzlu nebo barvu, tloušťku a styl čáry hrany
- Počet různých značek překračuje 5 nebo 6 – použití textových označení
- Malé grafy – vložení přímo do uzlů (obdélníkové či oválné tvary uzlů). Velikost uzlu podle nejdelšího labelu.
- Sjednocené umístění labelů pro hrany:
  - Vertikální – všechny vlevo nebo všechny vpravo
  - Horizontální – všechny nahoře nebo všechny dole

Labeling

- Při velkém množství různých označení není jejich souběžné zobrazení efektivní. Můžeme využít různé strategie:
  - Zobrazení labels pouze v okolí aktuální pozice kurzoru
  - Různé deformace vizualizace, abychom mohli dané sekci grafu věnovat větší oblast na obrazovce
  - Rotování grafů za účelem redukce překryvů označení

Labeling

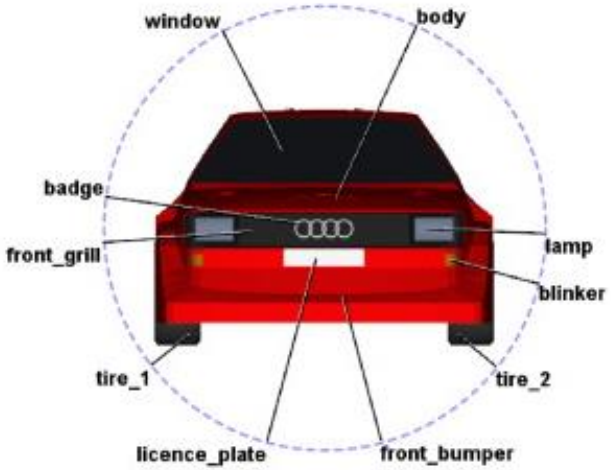

MSKA
MZFONDS
MFGEKIND
MSKA
MZFONDS
MBERMIDD
MOPLMIDD
MINKGEM
MFGEKIND

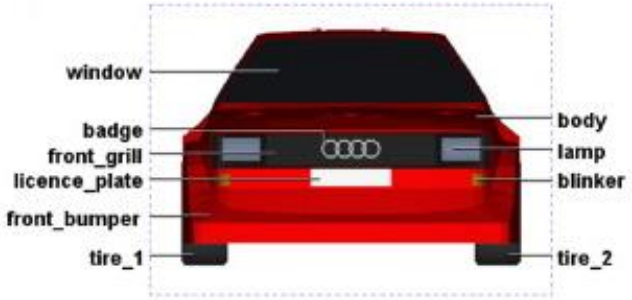
Labeling

- Zobrazení náhodné podmnožiny označení po krátký časový okamžik, poté zobrazení jiné náhodné podmnožiny atd.
- Krátkodobá paměť pozorovatele umožňuje zapamatovat si větší množství označení než při použití statického zobrazení

Labeling

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





Stromy, grafy a interakce

- Obecné typy interakce
  - např. sledování kamerou, zoomování
  - Běžné pro všechny typy vizualizace
- Specializované interakce
  - např. focus + context
  - Aplikovatelné na celou škálu vizualizací, ale prvotně vyvinuty pro vizualizaci stromů a grafů

Interakce s virtuální kamerou

- Běžné interakce (sledování kamerou, zoomování, rotace) jsou považovány za jednoduché změny aplikované na virtuální kameru zabírající určitý segment scény
- Operace jsou řízeny manuálně nebo automaticky (např. průlety nad scénou, automatická rotace 3D objektů)

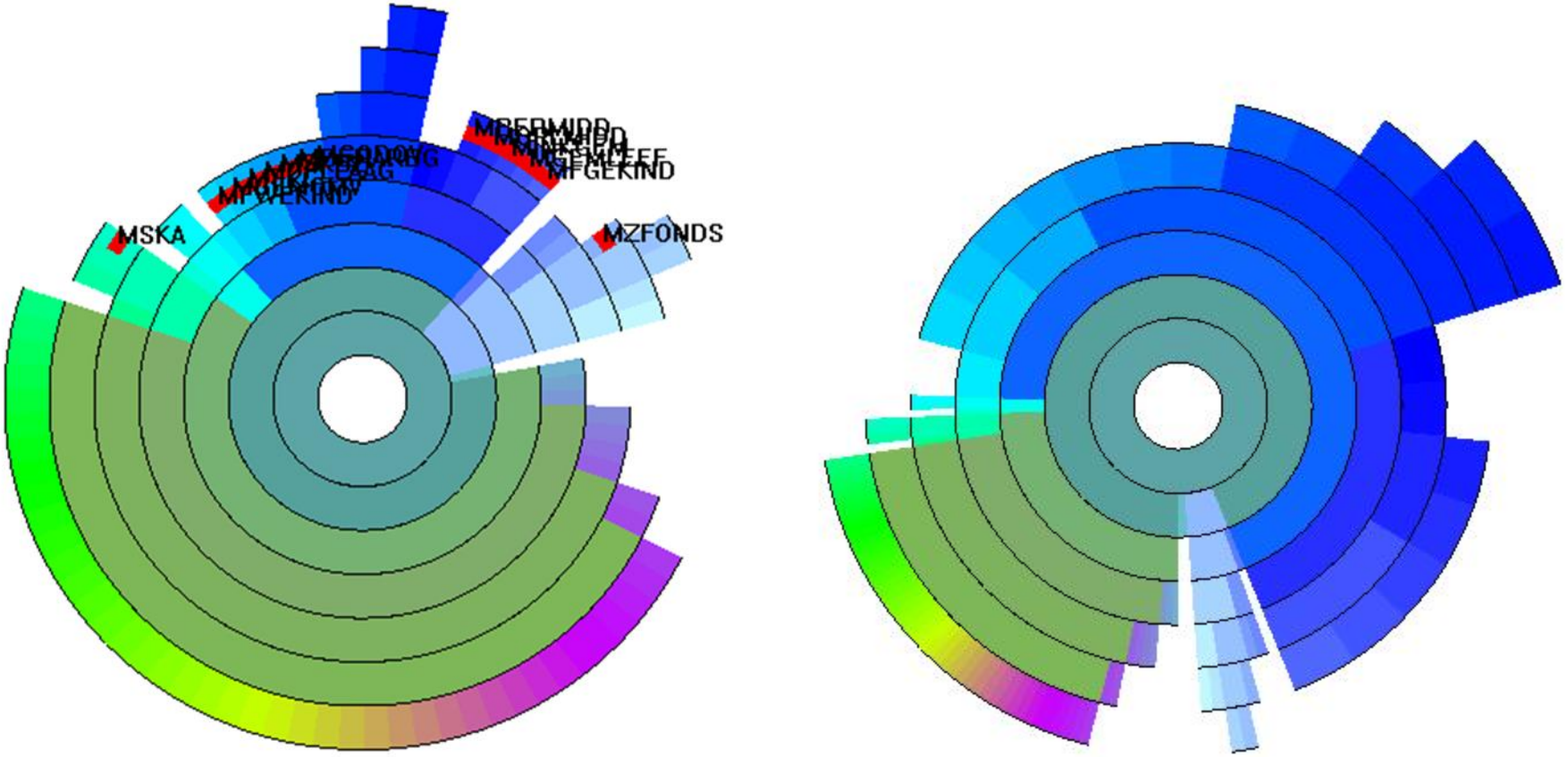
Interakce s prvky grafu

- Začíná výběrem (selekcí)
  - Izolujeme jednu nebo více komponent grafu
- Grafy s nepřehlednými shluky můžeme upravit:
  - Vybereme sadu uzlů a přetáhneme ji do méně obsazené oblasti obrazovky
  - Vybereme, posuneme či změníme tvar hran za účelem eliminace jejich křížení nebo zvýšení estetické hodnoty grafu
- Problém – v hustých regionech je téměř nemožné vybrat objekty jednoznačně

Interakce se strukturou grafu

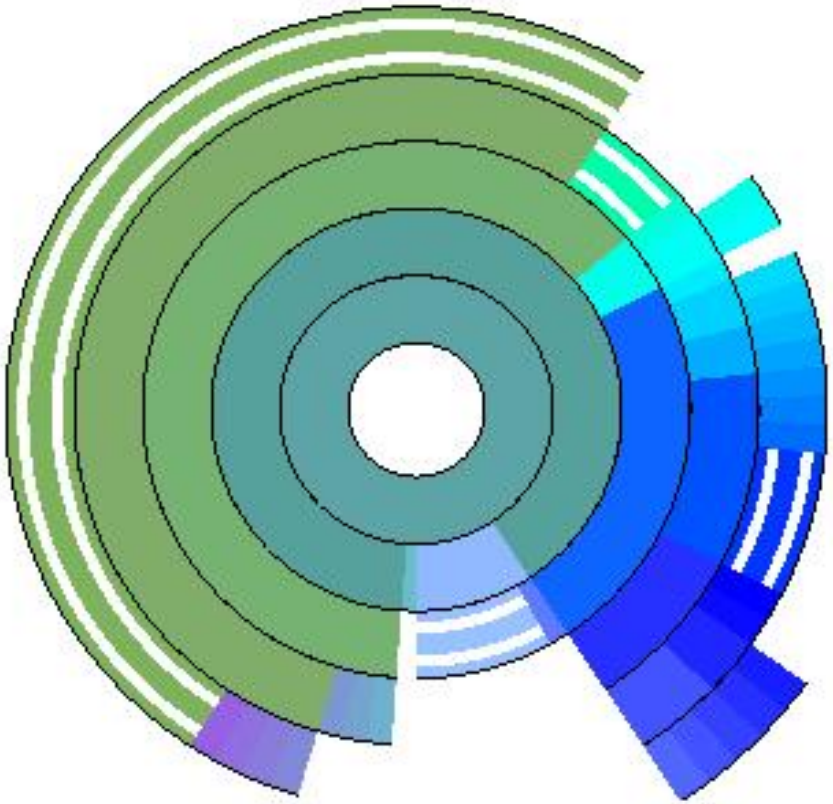
- Dvě základní třídy interakce:
  1. Mění samotnou strukturu grafu (např. přeskládání pořadí větví stromu)
  2. Focus+context techniky, kdy je podmnožina struktury prezentována detailně, zbytek pouze v obrysech (např. rybí oko)
     - Provedeno v prostoru obrazovky
     - Provedeno v prostoru struktury – vhodné pro grafy (zvětšení jedné větve (viz obrázek na dalším slajdu), zvýraznění hran vycházejících z daného uzlu, …)

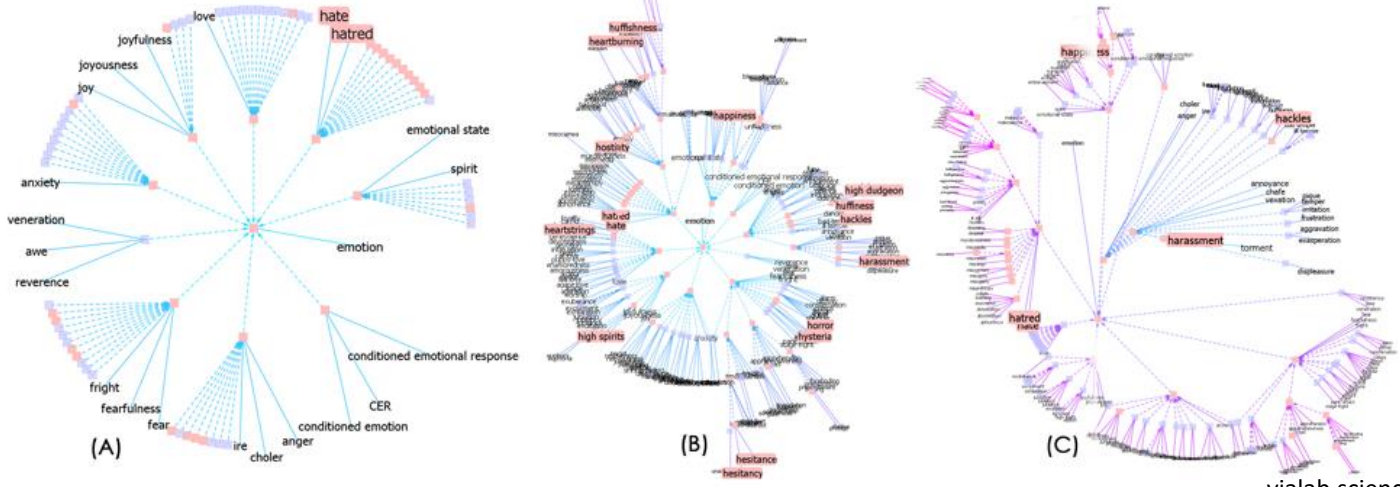
Interakce se strukturou grafu



Interakce se strukturou grafu

- Selektivní schovávání či odstraňování sekcí grafu





vialab.science.uoit.ca

Vizualizace textu a dokumentů

giladlotan.coms

joeganley.com

Definice

- **Korpus** = sada dokumentů
- Pracujeme s objekty uvnitř korpusu – slova, věty, odstavce, celé dokumenty, další korpusy. I obrázky a videa.
- Texty a dokumenty – strukturovány, obsahují atributy a metadata
- Systémy pro dolování informací z dokumentů – dotazování

Stupně reprezentace textu

- Tři stupně:
  - Lexikální
  - Syntaktický
  - Sémantický
- Každý stupeň požaduje jistou konverzi nestrukturovaného textu do nějaké formy strukturovaných dat

Lexikální stupeň

- Transformace řetězce znaků do sekvence atomických entit = **tokens**
- Lexikální analyzéry zpracovávají sekvenci znaků s danou sadou pravidel do nové sekvence tokenů
- Tokeny obsahují znaky (characters), n-gramy, slova, lexémy, fráze, …
- Konečné stavové automaty

Syntaktický stupeň

- Identifikace a označování (anotace) funkcí každého tokenu
- Přiřazení značek – pozice věty, slovní druh, jednotné či množné číslo, příbuznost k ostatním tokenům, …
- Proces extrahování anotací = **NER (named entity recognition)**

Sémantický stupeň

- Extrakce významu a vztahů mezi poznatky odvozenými ze struktur identifikovaných v syntaktickém stupni
- Cílem je definovat analytickou interpretaci celého textu v daném kontextu nebo i nezávisle na kontextu

Vector Space Model (VSM)

- algebraický model pro reprezentaci textových dokumentů
- „term vektor“ = vektor, ve kterém každá dimenze reprezentuje váhu daného slova v dokumentu
- Odstranění „stop words“ (the, a, …) pro redukci šumu
- Agregace slov o stejném základu (kořeni)
- Příkladem term vektoru je hašovací tabulka, která mapuje unikátní termy na počet jejich výskytů v dokumentu

Vector Space Model

```
Count-Terms(tokenStream)
1. terms := Ø; /* inicializace
   terms na prázdnou hašovací
   tabulku */
2. for each token t in tokenStream
3.   do if t není stop word
4.       do inkrementuj (nebo
          inicializuj na 1) terms[t];
5. return terms;
```

Příklad

- **Vzorový text:**

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.

Příklad

- Předchozí odstavec obsahuje 98 řetězcových tokenů, 74 termů a 48 termů po odstranění stop words
- Příklad term vektoru generovaného pseudokódem:

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

VSM – počítání vah

- Různé metody přiřazování vah, nejznámější je **term frequency inverse document frequency (TfIdf)**
- Nechť **Tf(w)** je term frequency = počet výskytů slova w v dokumentu, **Df(w)** je document frequency = počet dokumentů, které obsahují slovo w, **N** je počet dokumentů. Pak TfIdf(w) je

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

VSM – počítání vah

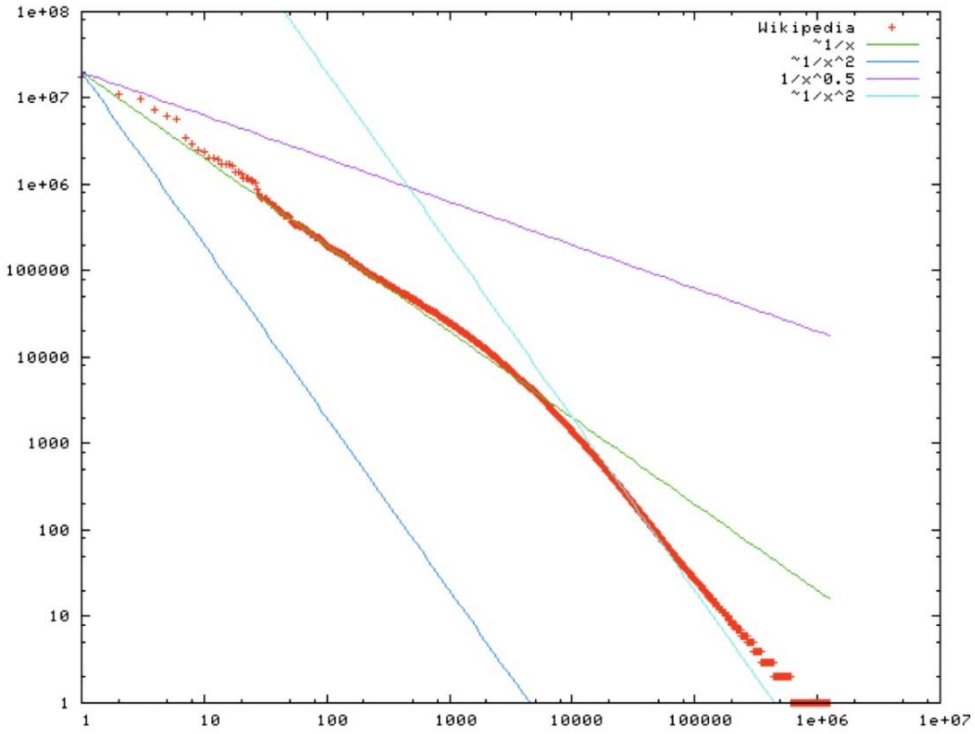
- Zajímají nás slova vyskytující se často v jednom dokumentu, ale nejsou tak časté v ostatních dokumentech korpusu

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

```
Compute-TfIdf(documents)
1  termFrequencies ← ∅ // Looks up term count tables for document names.
2  documentFrequencies← ∅ // Counts the documents in which a term occurs.
3  uniqueTerms← ∅ // The list of all unique terms.
4  for each document d in documents
5        do docName ← Name(d) // Extract the name of the document.
6           tokenStream ← Tokenize(d) // Generate document token stream.
7           terms ← Count-Terms(tokenStream) // Count the term frequencies.
8           termFrequencies[docName] ← terms // Store the term frequencies.
9           for each term t in Keys(terms)
10                do increment (or initialize to 1) documentFrequencies[t]
11                      uniqueTerms← uniqueTerms ∪ t
12
13 tfIdfVectorTable ← ∅ // Looks up tf-idf vectors for document names.
14 n ← Length(documents)
15 for each document name docName in Keys(termFrequencies)
16       do tfIdfVector ← create zeroed array of length Length(uniqueTerms)
17          terms ← termFrequencies[docName]
18          for each term t in keys(terms)
19               do tf ← terms[t]
20                     df ← documentFrequencies[t]
21                     tfIdf ← tf ∗ log(n/df)
22                     tfIdfVector[index of t in uniqueTerms]← tfIdf
23                     tfIdfVectorTable[docName] ← tfIdfVector
24 return tfIdfVectorTable
```

Zipfův zákon

- V typickém dokumentu v přirozeném jazyce je frekvence slov jakéhokoliv slova inverzně úměrná jeho ranku ve frekvenční tabulce

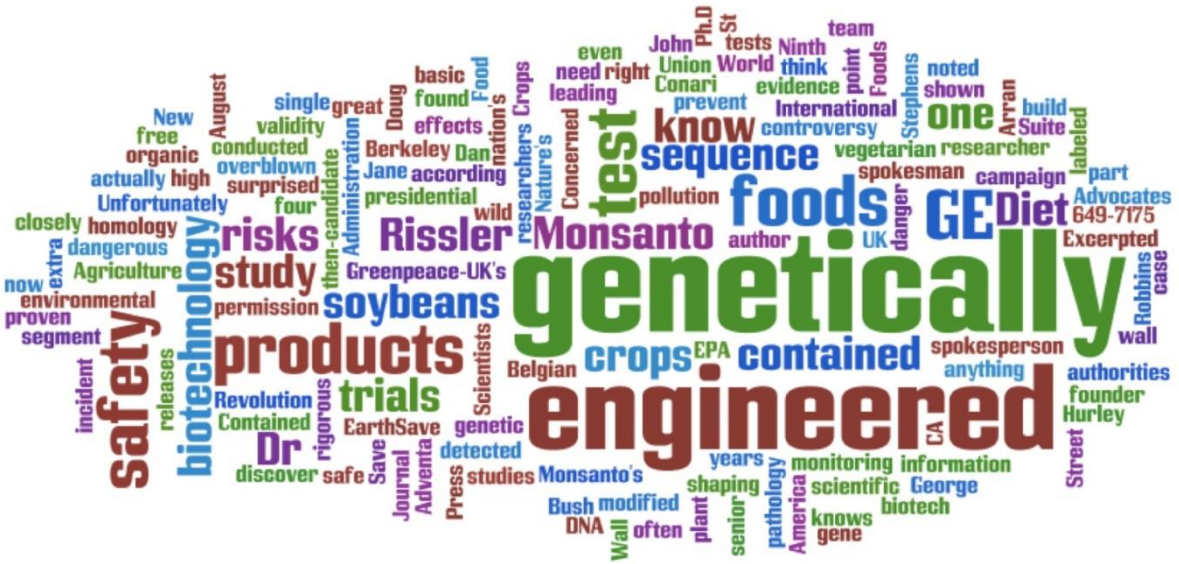


Úkoly využívající Vector Space Model

- Vector Space Model doplněný o vzdálenostní metriku umožňuje provádět celou řadu úkolů:
  - TfIdf v kombinaci s VSM – identifikace zajímavých dokumentů, querying
  - Poskytnutí informace o významu celého korpusu – hledání vzorů, klastrů, …
- Vizualizační pipeline se dobře mapuje na vizualizaci dokumentů:
  - Získáme data (korpus), převedeme na vektory, spustíme algoritmy podle typu úkolu, vizualizujeme

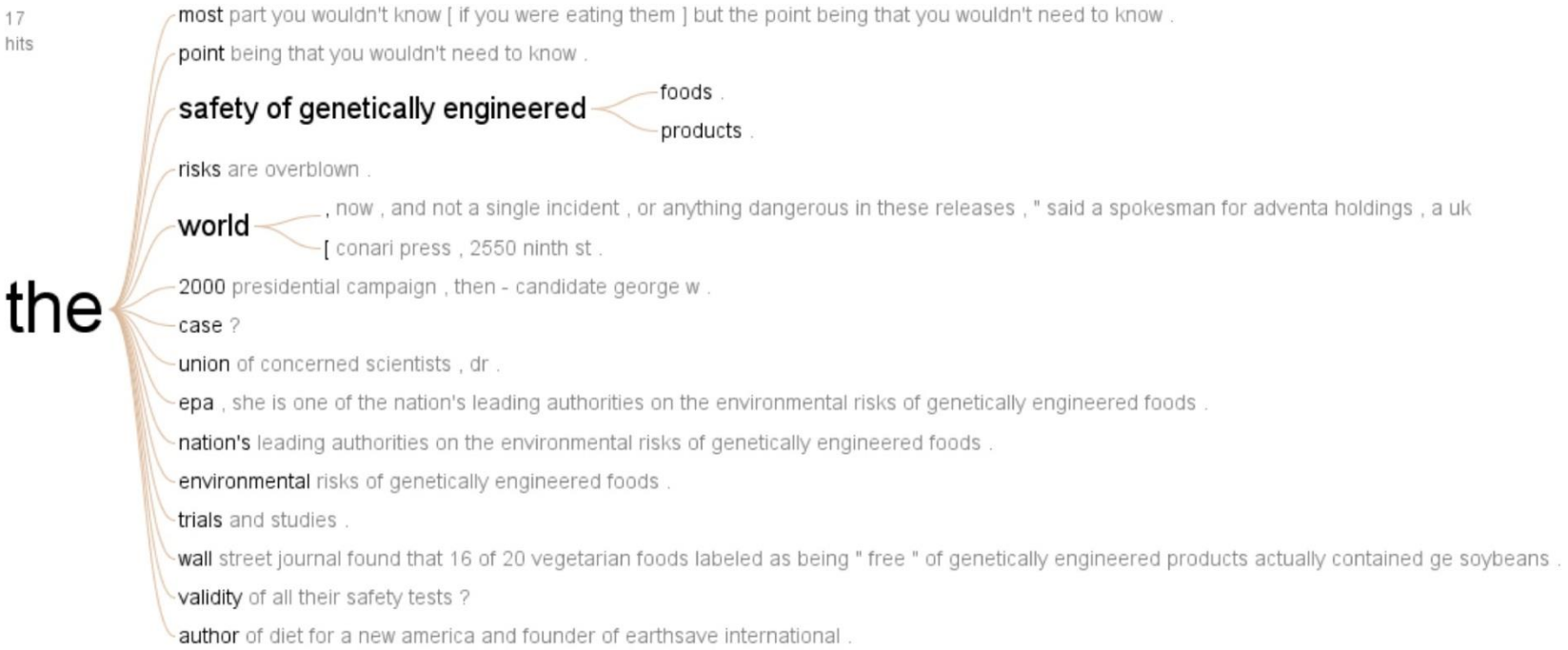
Vizualizace jednotlivých dokumentů

- Tag clouds



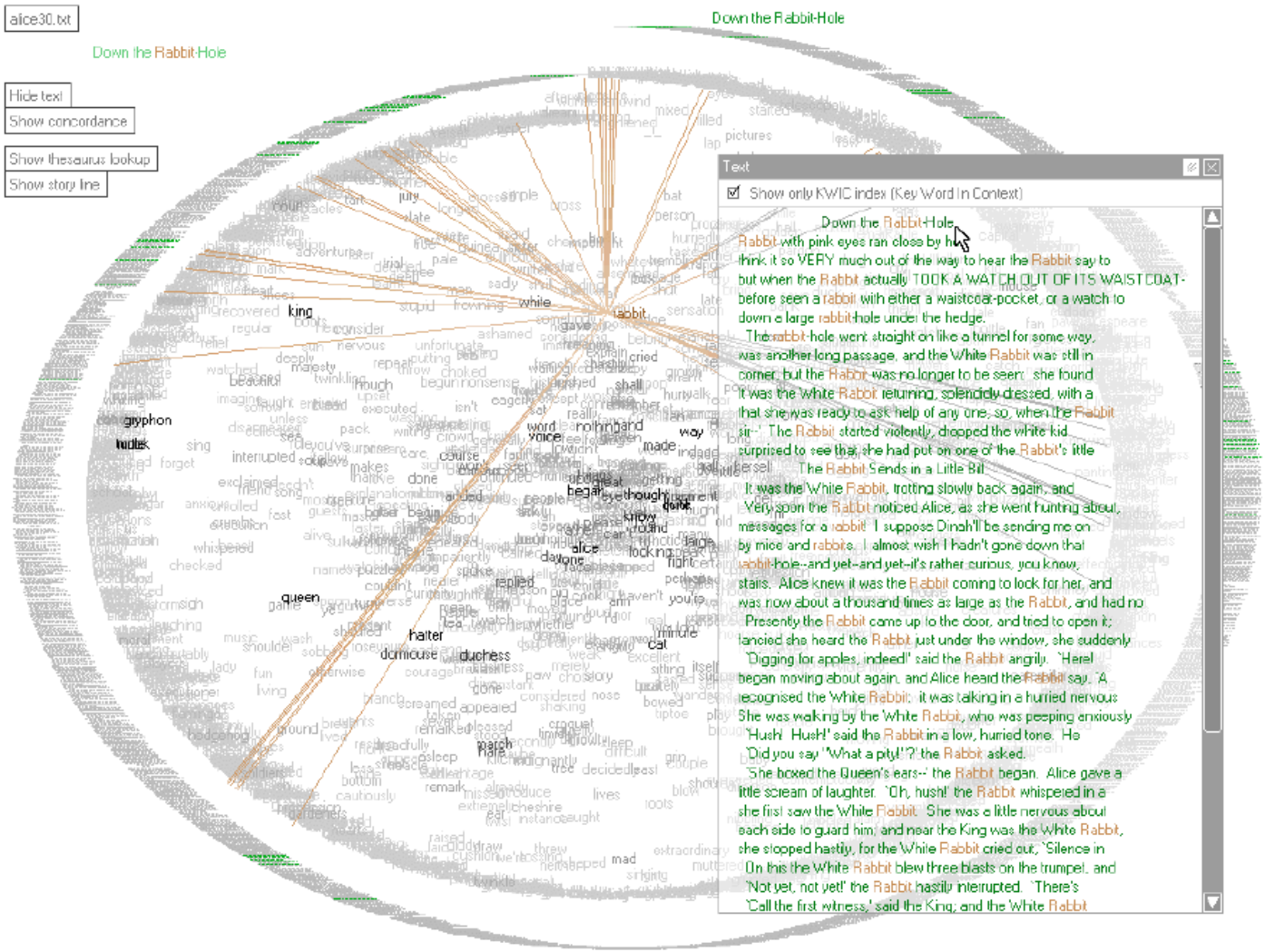
Vizualizace jednotlivých dokumentů

- WordTree – větve stromu reprezentují různé kontexty, ve kterých se slovo v kořeni v dokumentu nachází, zobrazena i frekvence termů

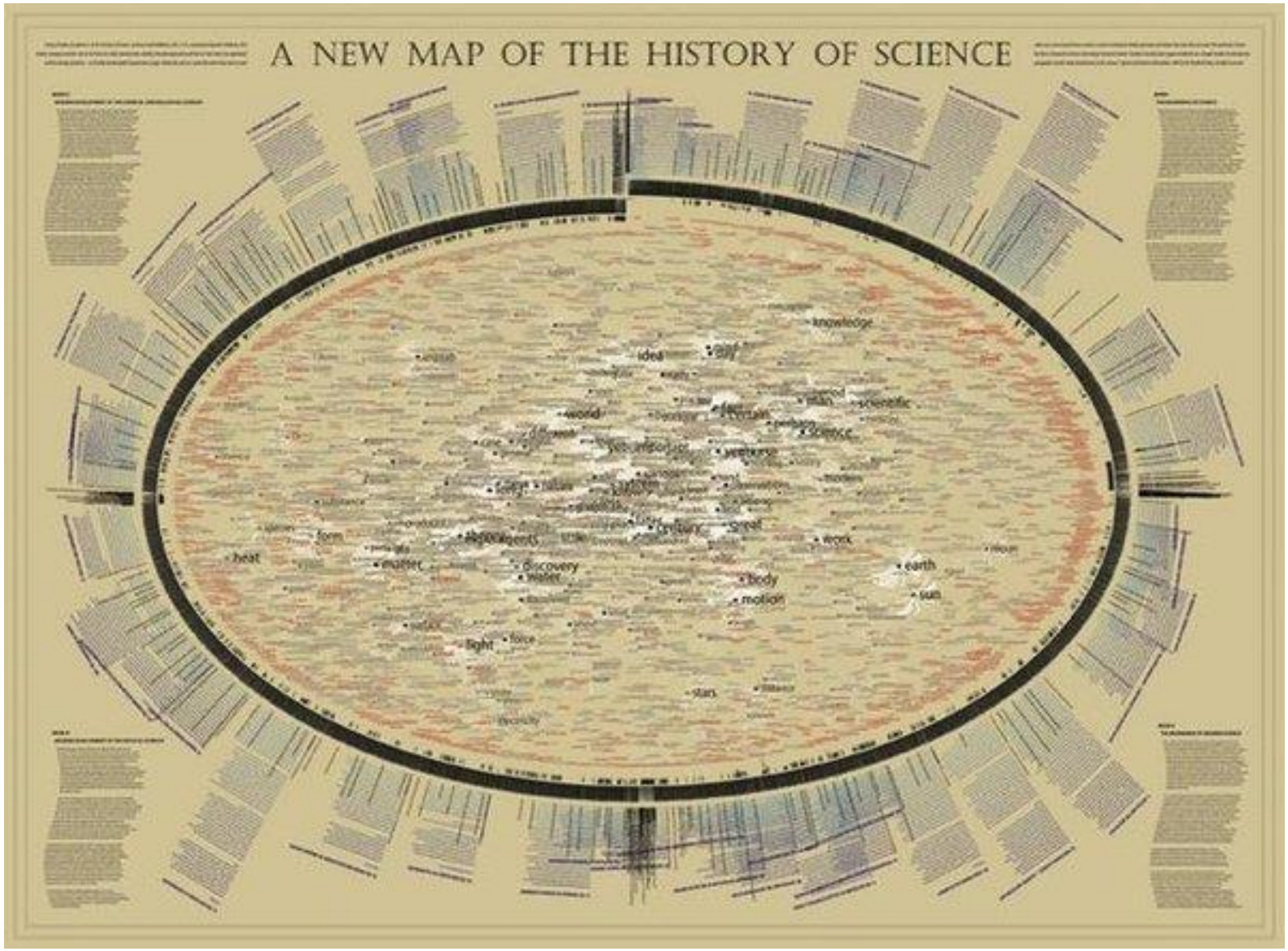


Vizualizace jednotlivých dokumentů

- TextArc

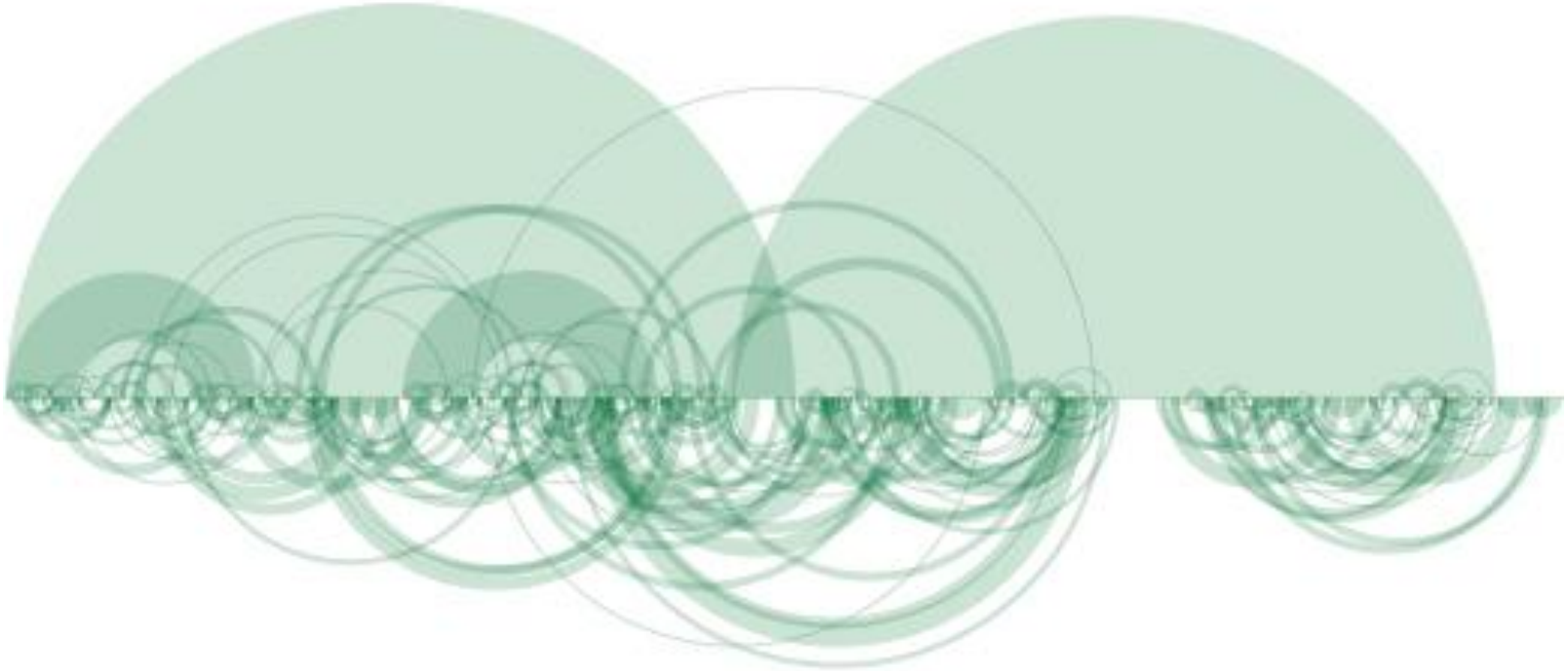


Smith Williams – A History of Science



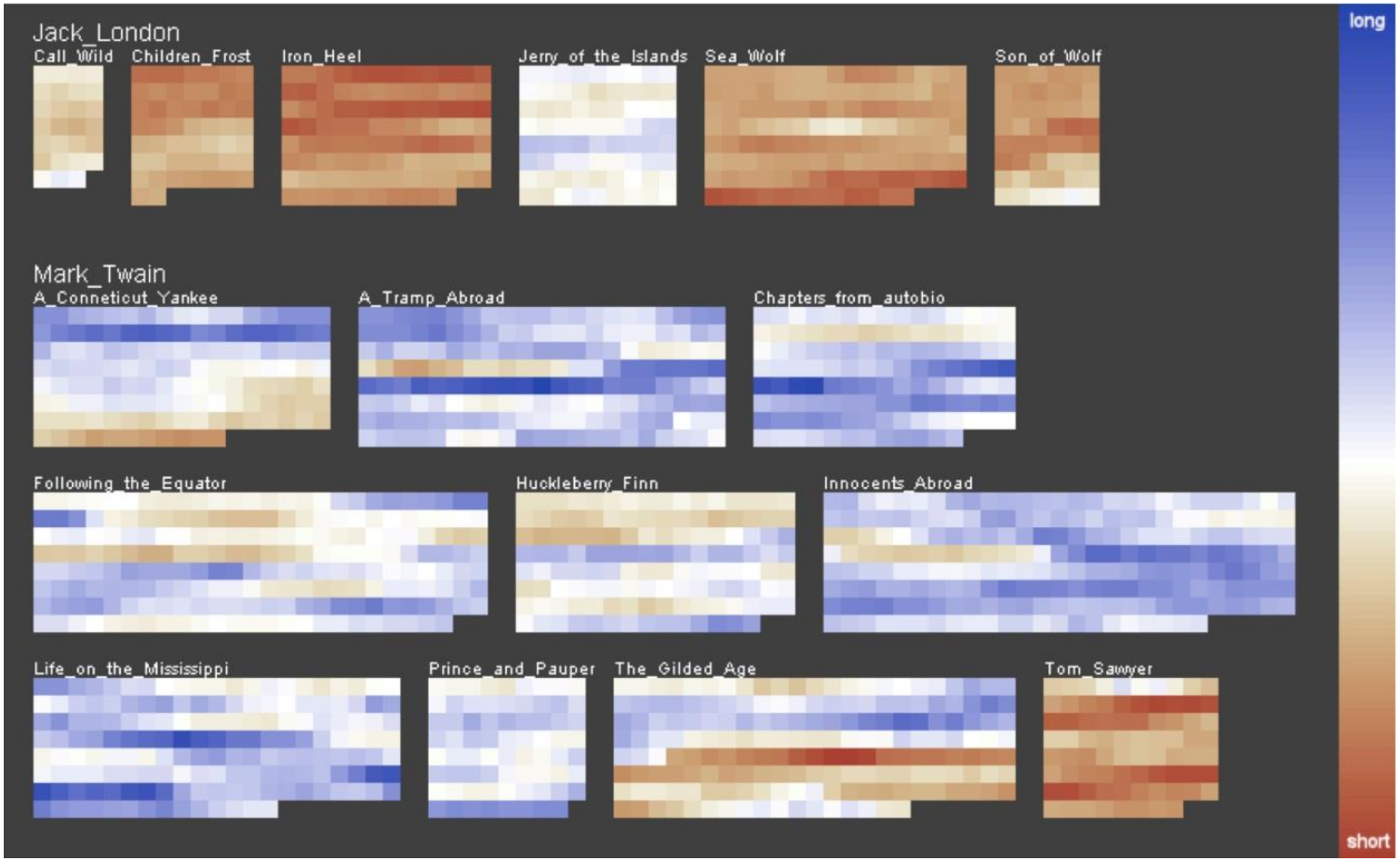
Vizualizace jednotlivých dokumentů

- Arc Diagrams – zobrazení opakování v textu
- Bachův menuet v G dur:



Vizualizace jednotlivých dokumentů

- Literature Fingerprints



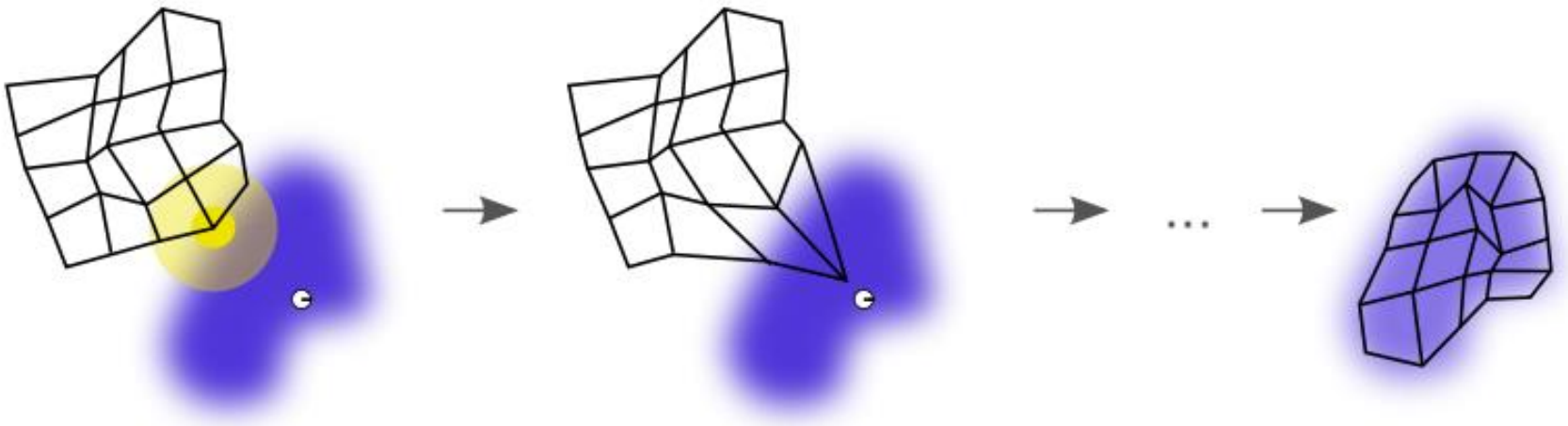
Vizualizace sady dokumentů

- Cílem je umístit podobné dokumenty co nejblíže sebe a naopak rozdílné dokumenty co nejdále od sebe
- Algoritmus spočte podobnost mezi všemi páry dokumentů a tím je určeno jejich rozložení – složitost $O(n^2)$

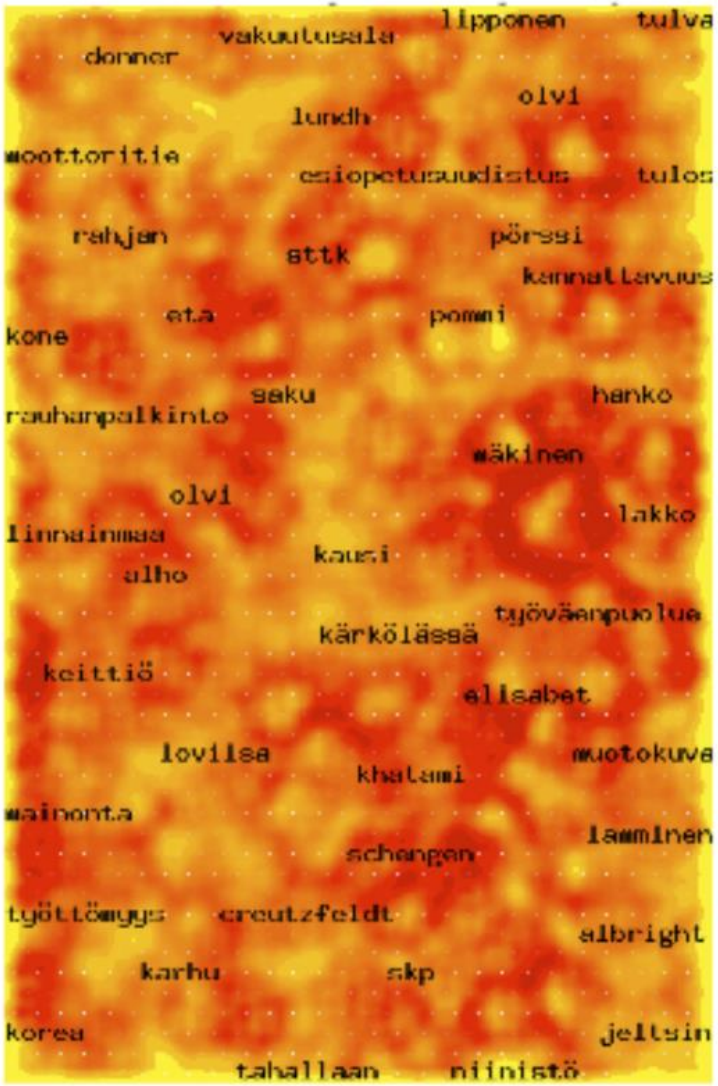
Vizualizace sady dokumentů

- Self-Organizing Maps – algoritmus strojového učení, kolekce 2D uzlů, do kterých umístíme dokumenty. Každý uzel obsahuje vektor stejné dimenzionality, jako mají vstupní vektory dokumentů
- Na počátku inicializujeme SOM uzly – typicky náhodně. Vybereme náhodný vektor, spočteme jeho vzdálenost od ostatních uzlů. Přiřadíme váhy nejbližším uzlům (v daném poloměru) – nejbližší uzel má největší váhu. Iterace přes vstupní vektory, postupně zmenšujeme poloměr.

Self-Organizing Maps

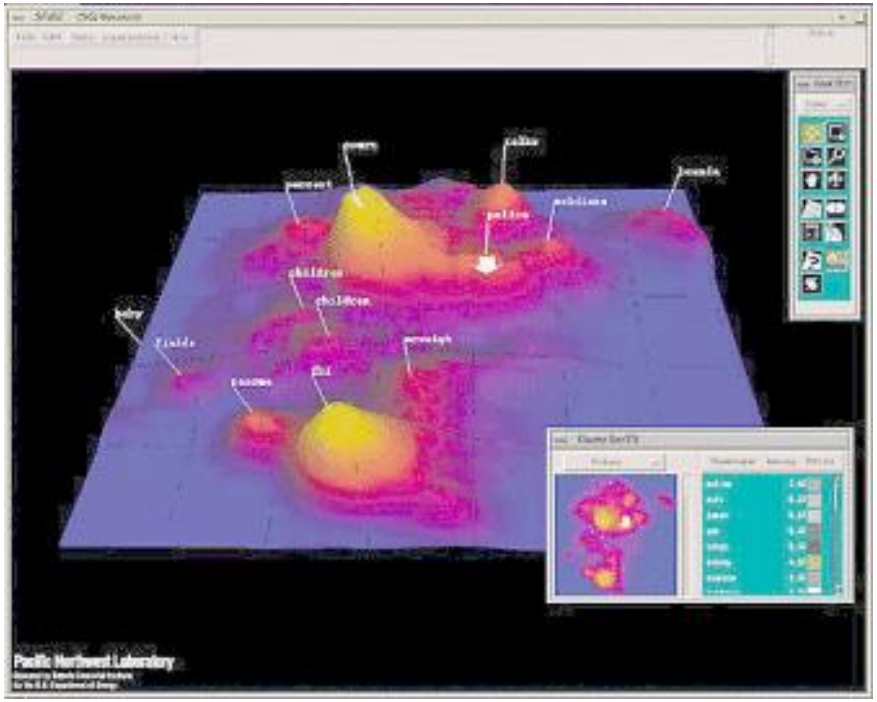


Self-Organizing Maps - příklad



Vizualizace sady dokumentů

- Themescapes – souhrnné informace o korpusech ve formě 3D terénů
- Výška a barva využity pro reprezentaci hustoty podobných dokumentů

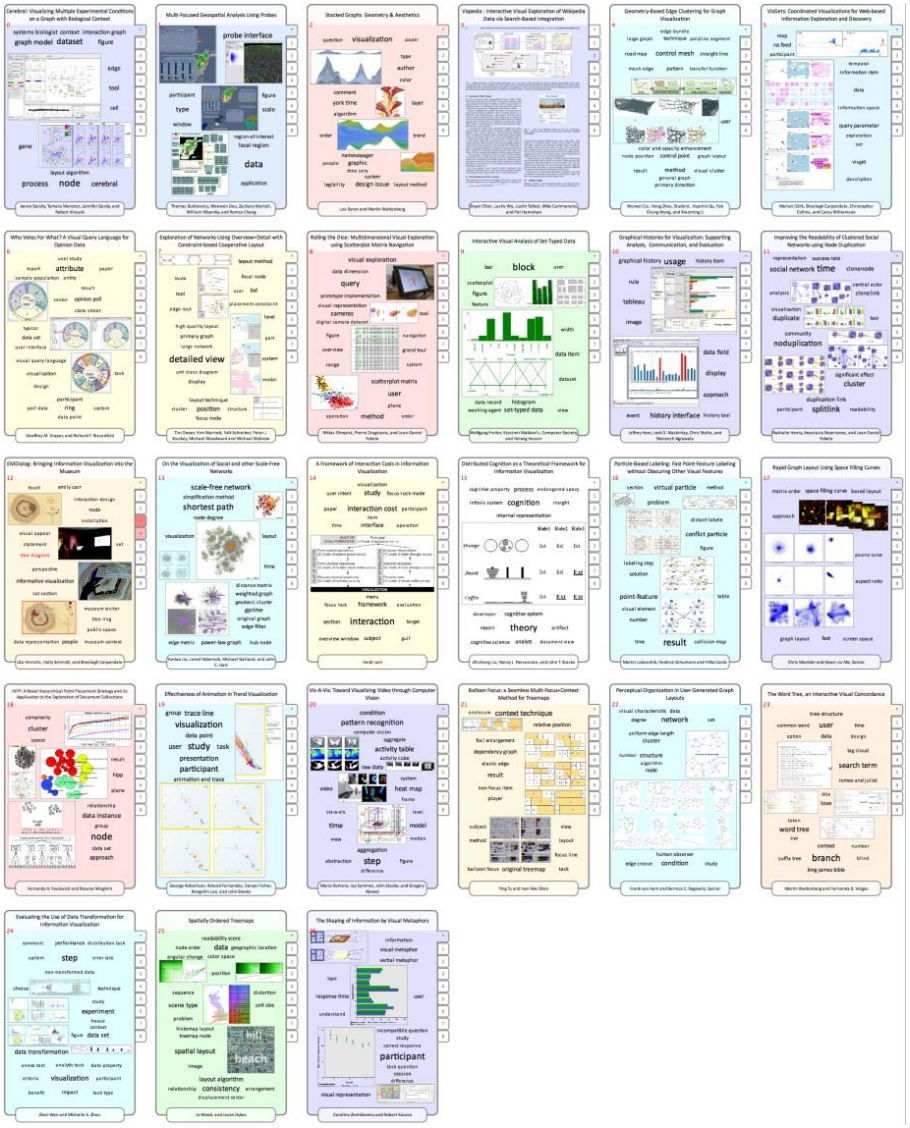


www3.sympatico.ca

Vizualizace sady dokumentů

- Document cards – reprezentují klíčovou sémantiku dokumentů ve formě směsi obrázků a klíčových termů
- Termy získáme pomocí pokročilých algoritmů pro dolování textu
- Obrázky získáme za použití grafických heuristik
- Využívá barevné histogramy obrázků, které je klasifikují a řadí do tříd:
  - 1. třída: fotografie/renderované obrázky, třída 2: diagramy/náčrty/grafy, třída 3: tabulky
  - Zobrazíme alespoň jednoho reprezentanta každé neprázdné třídy

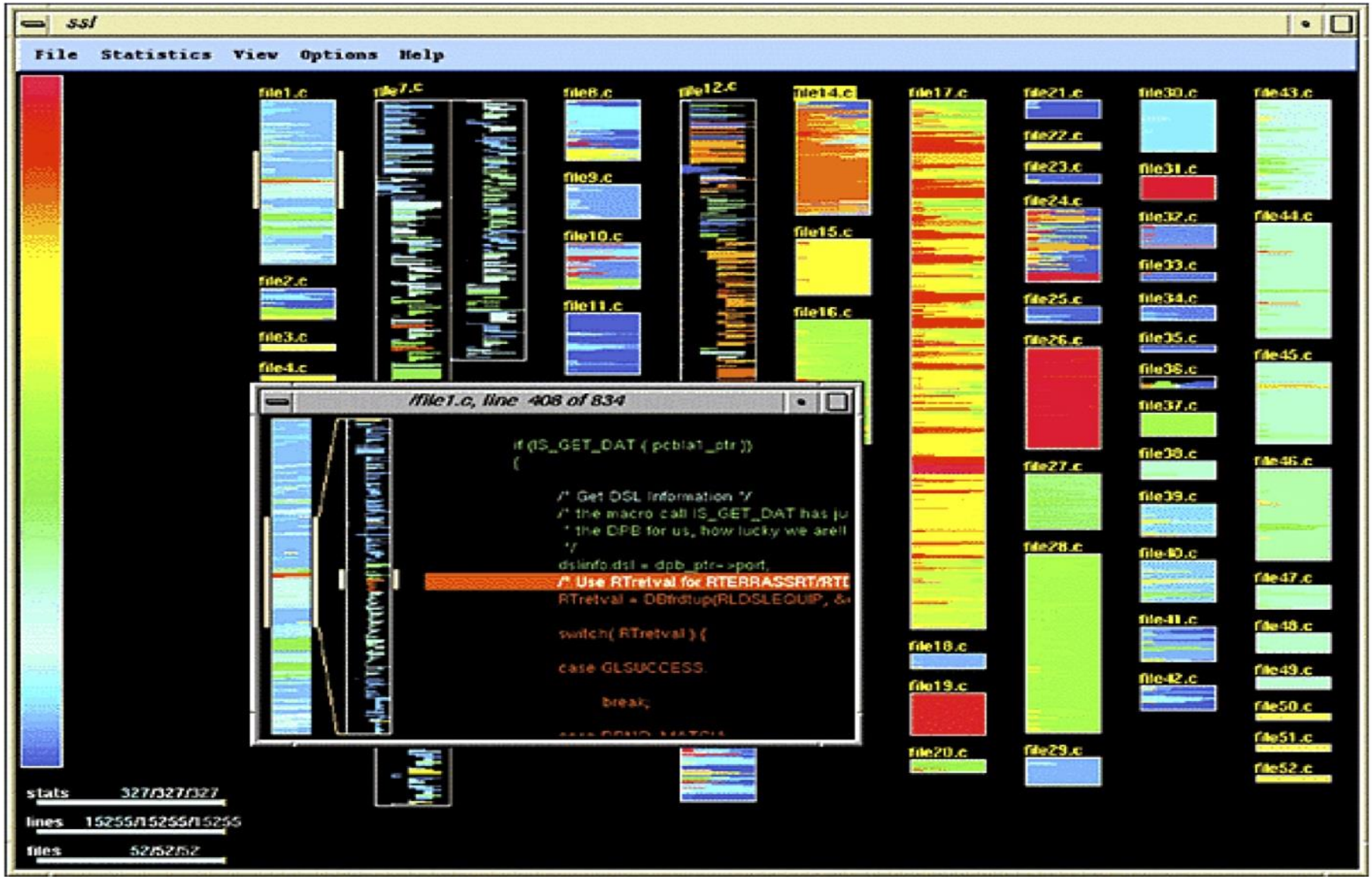
Document cards



Rozšířené metody vizualizace textu

- Zahrnují i metadata:
  - Software Visualization
  - Search Result Visualization
  - Temporal Document Collection Visualizations
  - Representing Relationships

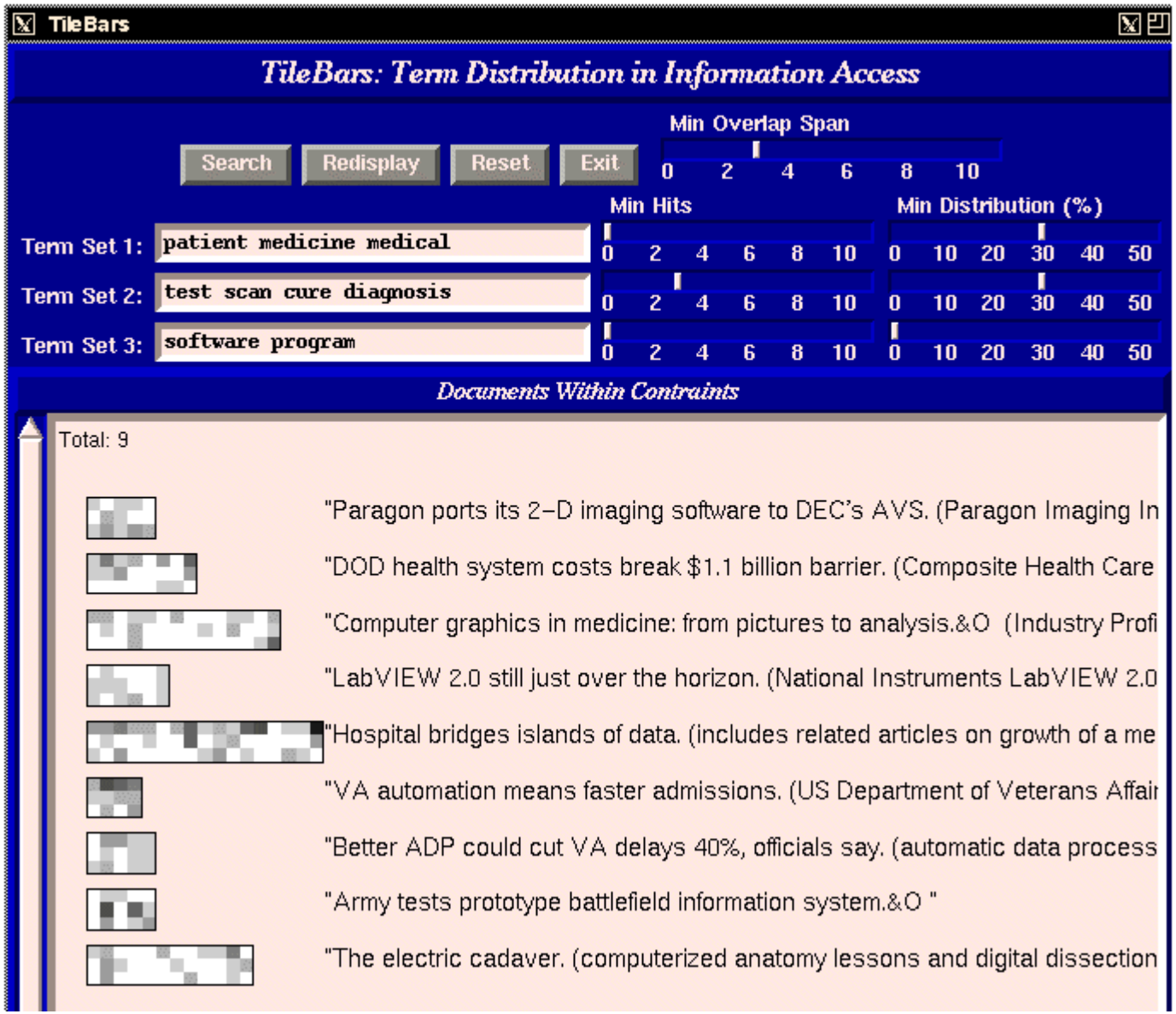
Software Visualization



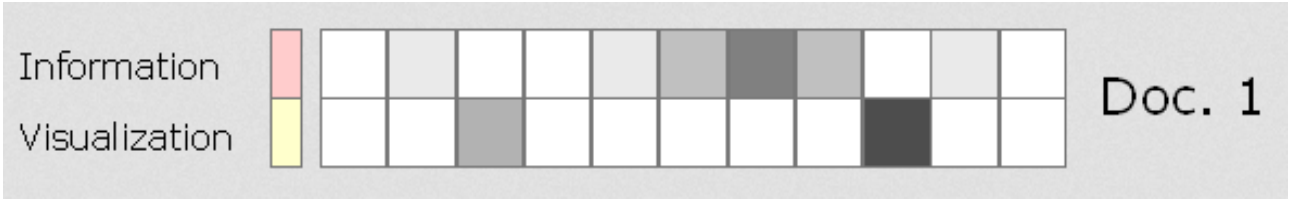
Search Result Visualization

- TileBars - každý dokument výsledné množiny je reprezentován obdélníkem, kde šířka naznačuje relativní délku dokumentu a navrstvené čtverce uvnitř odpovídají segmentům textu. Čím tmavší je čtverec, tím je větší frekvence sady dotazovacích termů.
- Kompaktní reprezentace a informace o struktuře dokumentu odrážející relativní délku dokumentu, frekvenci dotazovacích termů a jejich rozložení

Search Result Visualization



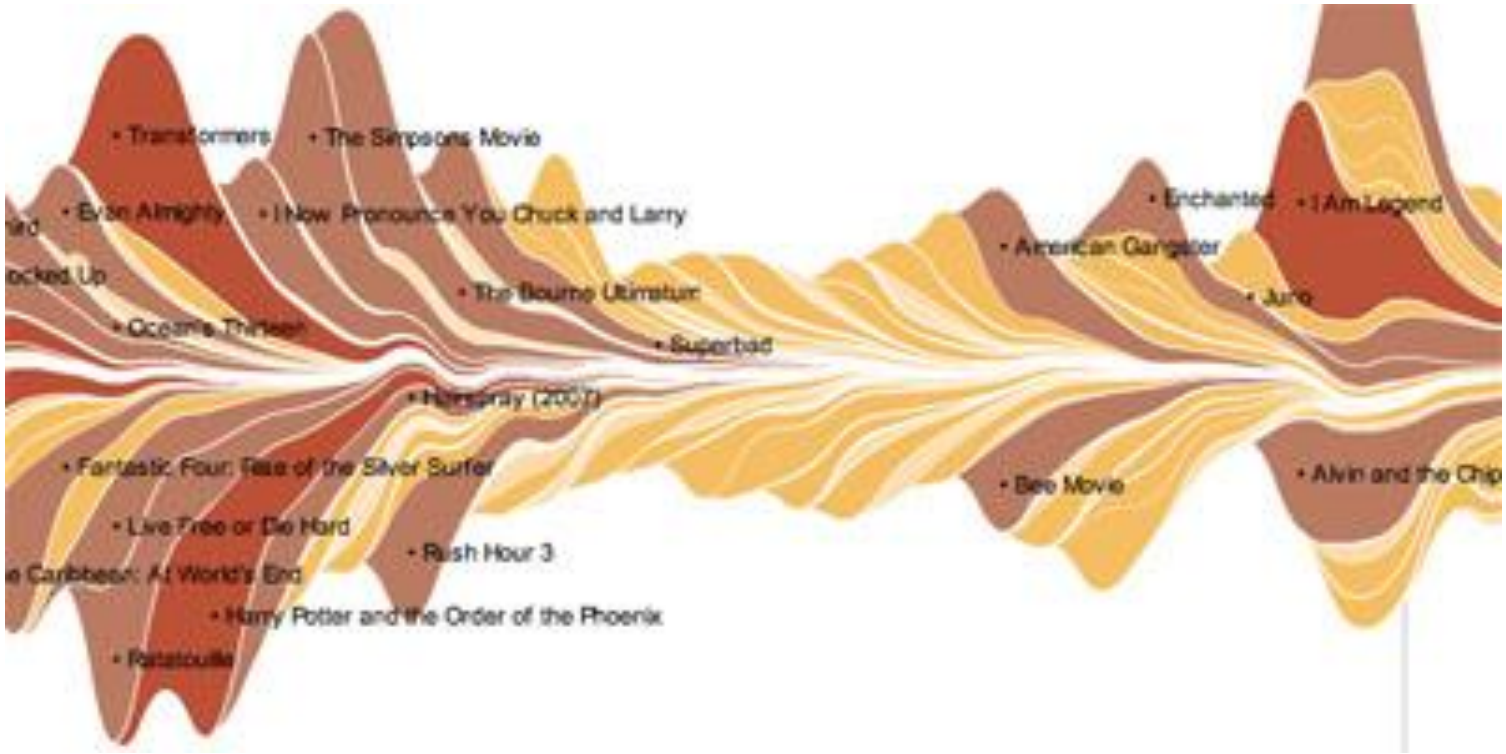
Příklad


Information
Visualization
Doc. 1


Information
Visualization
Doc. 2

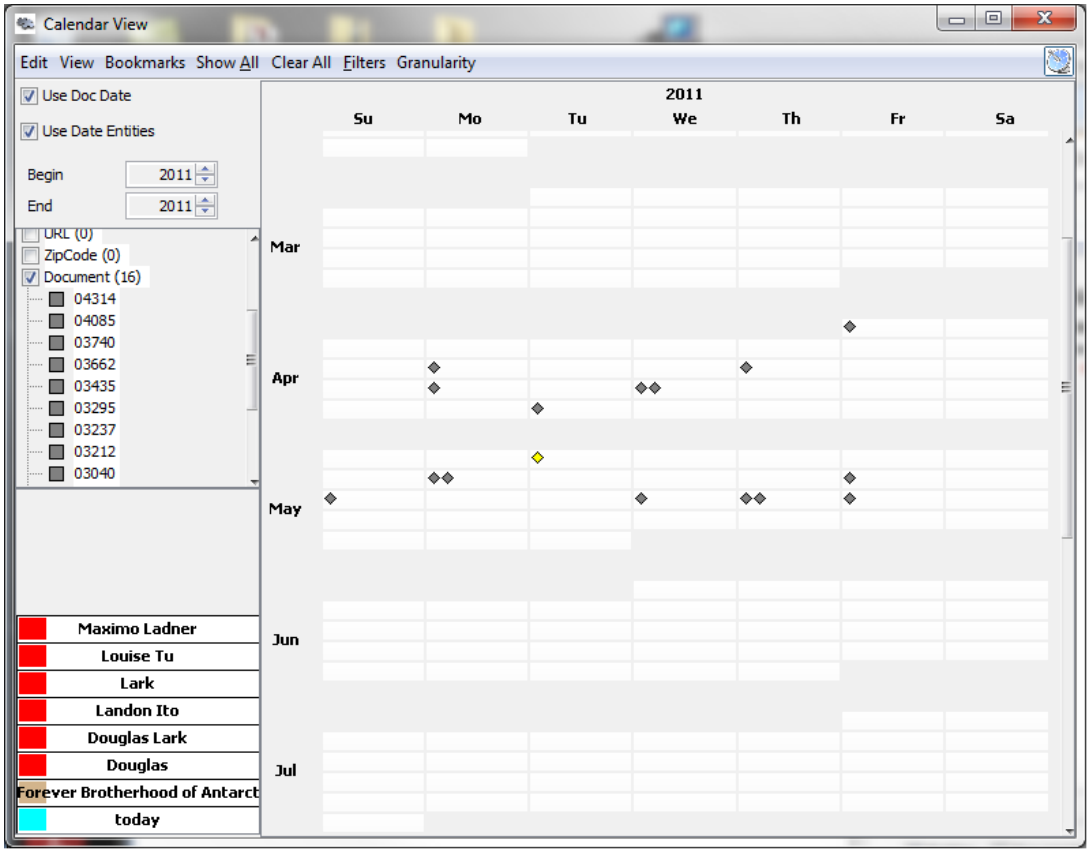
Temporal Document Collection Visualizations

- ThemeRiver – vizualizace tematických změn v čase pro sadu dokumentů, vertikální tloušťka odpovídá frekvenci v daném čase



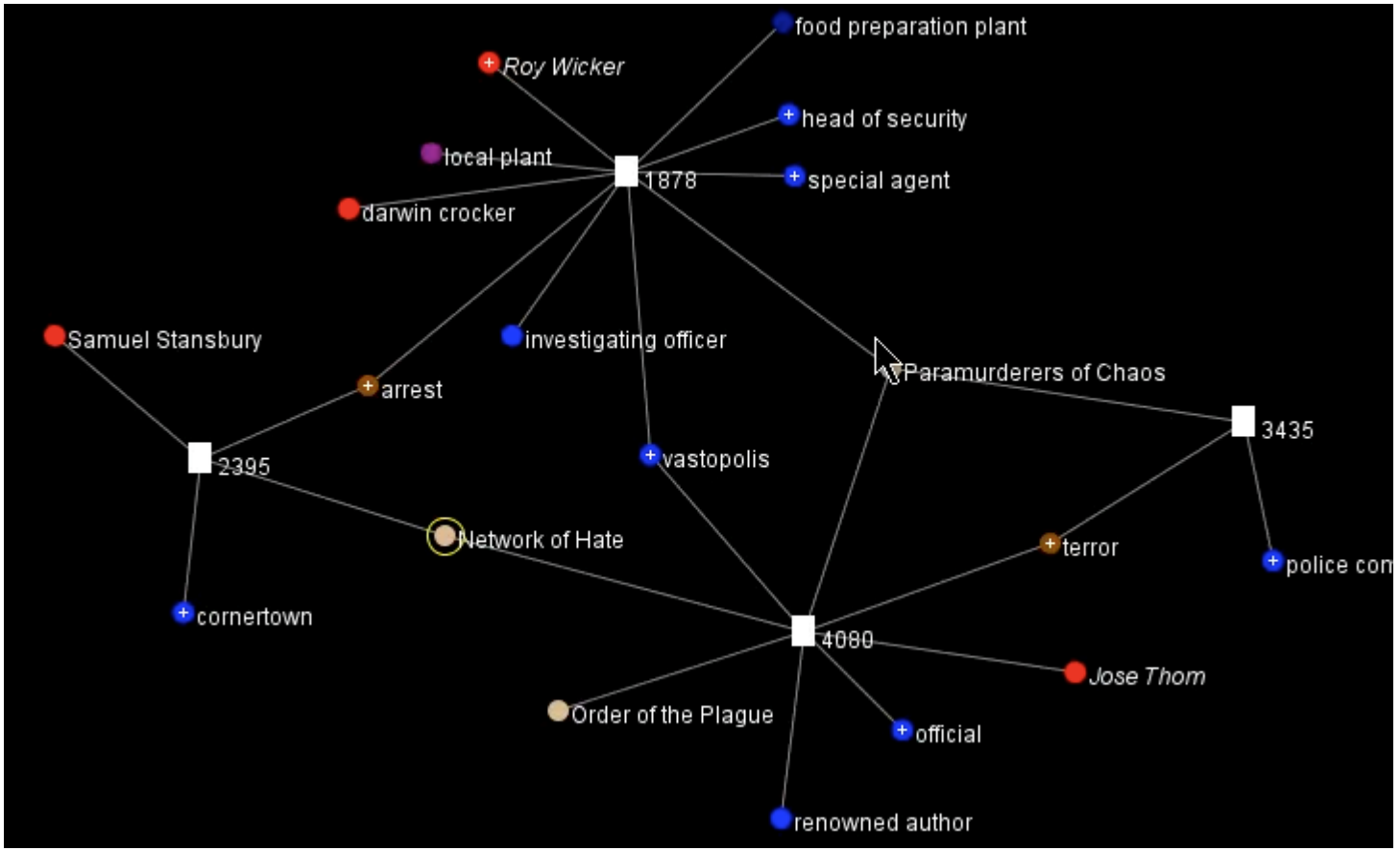
Temporal Document Collection Visualizations

- Jigsaw calendar view – vizualizace a prozkoumávání textových korpusů využívající vzhled kalendáře
- Novinové články



Representing Relationships

- Jigsaw Graph entity view



Representing Relationships

- Jigsaw List view