

**UNIVERZITA KOMENSKÉHO V BRATISLAVE  
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**

**VIZUALIZÁCIA TEXTOVEJ KOMUNIKÁCIE  
V REÁLNO M ČASE**

Diplomová práca

**UNIVERZITA KOMENSKÉHO V BRATISLAVE  
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**

9d742261-be11-431d-a8c2-4f1f081d5962

**VIZUALIZÁCIA TEXTOVEJ KOMUNIKÁCIE  
V REÁLNO M ČASE**

Diplomová práca

Študijný program: Aplikovaná informatika  
Študijný odbor: 2511 Aplikovaná informatika  
Školiace pracovisko: Katedra aplikovanej informatiky  
Školiteľ: Mgr. Matej Novotný, PhD.

**Bratislava, 2013**

**Bc. Kamil Maráz**



Univerzita Komenského v Bratislave  
Fakulta matematiky, fyziky a informatiky

---

## ZADANIE ZÁVEREČNEJ PRÁCE

**Meno a priezvisko študenta:** Bc. Kamil Maráz  
**Študijný program:** aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)  
**Študijný odbor:** 9.2.9. aplikovaná informatika  
**Typ záverečnej práce:** diplomová  
**Jazyk záverečnej práce:** slovenský

**Názov:** Vizualizácia textovej komunikácie v reálnom čase

**Cieľ:** Navrhnuť, implementovať a demonštrovať vizualizáciu textovej komunikácie v reálnom čase (séria emailov, chat, ...) Vizualizácia bude využívať focus + context princíp a niektorú z použiteľných fyzikálnych metafor (stúpajúce bubliny, padajúce kamene, pružinový systém, ...) pre reprezentáciu textových jednotiek a ich zoradenia v čase.

**Vedúci:** Mgr. Matej Novotný, PhD.  
**Katedra:** FMFI.KAI - Katedra aplikovanej informatiky  
**Vedúci katedry:** doc. PhDr. Ján Rybár, PhD.  
**Dátum zadania:** 13.10.2010

**Dátum schválenia:** 17.10.2010

doc. RNDr. Roman Ďurikovič, PhD.  
garant študijného programu

.....  
študent

.....  
vedúci práce

Čestne prehlasujem, že som túto diplomovú prácu  
vypracoval samostatne s použitím citovaných zdrojov.

.....

Týmto si dovoľujem poďakovať vedúcemu svojej diplomovej práce Mgr. Matejovi Novotnému, PhD. za jeho odborné vedenie, cenné rady a pripomienky, ktoré mi poskytol pri spracovaní tejto práce. Chcem sa poďakovať svojej priateľke Laile za jej veľkú trpezlivosť a lásku a pánovi Michalovi Slavkovskému za jeho rady v oblasti simulácie a filozofické debaty, ktorými sa mi navždy zapísal do hlavy. Veľká vďaka patrí aj mojej mame, otcovi, sestre a zhode okolností, ktorá vyústila v moju existenciu.

# Abstrakt

Autor:	Bc. Kamil Maráz
Názov:	Vizualizácia textovej komunikácie v reálnom čase
Univerzita:	Univerzita Komenského v Bratislave
Fakulta:	Fakulta matematiky, fyziky a informatiky
Katedra:	Katedra aplikovanej informatiky
Vedúci diplomovej práce:	Mgr. Matej Novotný, PhD.
Rozsah práce:	74 strán
Rok:	2013

Cieľom tejto práce bolo navrhnúť, implementovať a demonštrovať vizualizáciu textovej komunikácie v reálnom čase. Na sémantickú analýzu prúdu textu sme aplikovali metódu Latentnej Dirichletovej alokácie a navrhli proces pre jeho analýzu v reálnom čase. Na vizualizáciu spracovaných dát z prúdu textu sme použili metaforu vesmírnych telies a na ne sme namapovali jednotlivé dáta. Nami navrhnutá vizualizácia využíva focus+context princíp a naimplementovali sme ju pomocou silovo-riadeného systému, ktorý je simuláciou gravitácie medzi telesami a je plne prispôsobený pre požiadavky vizualizácie. V tejto práci sme popísali celý proces, ktorý nás viedol k zvolenej metodike, proces implementácie vizualizácie a jej následné testovanie a vyhodnotenie.

Kľúčové slová: vizualizácia informácií, prúd textu, gravitácia, sémantika.

# Abstract

Author:	Bc. Kamil Maráz
Thesis title:	Real-time visualization of text communication
University:	Comenius University, Bratislava
Faculty:	Faculty of Mathematics, Physics and Informatics
Department:	Department of Applied Informatics
Advisor:	Mgr. Matej Novotný, PhD.
Thesis length:	74 pages
Year:	2013

The aim of this work was to design, implement and demonstrate the visualization of text communication in real time. For the semantic analysis of the text stream we applied the method called Latent Dirichlet allocation and we proposed a process for its analysis in real time. To visualize the processed data from the text stream, we used the metaphor of celestial bodies and then we mapped the individual data to it. Our suggested visualization uses focus+context principle and we have successfully implemented it using force-directed system, which is a simulation of gravity between bodies and is fully adapted to the requirements of our visualization. In this paper we describe the process that led us to the chosen methodology, implementation process of visualization and its subsequent testing and evaluation.

Keywords: information visualization, text stream, gravity, semantics.

# Predhovor

Vizualizácia prúdu textu v reálnom čase je mladým odvetvím vizualizácie a v súčasnosti táto oblasť zaznamenáva dynamický rozvoj. Táto práca vznikla ako reakcia na potrebu vizuálnej analýzy neustále pribúdajúceho množstva dát v reálnom čase či už výskumnými pracovníkmi v danej oblasti alebo bežnými používateľmi. Je potenciálne nekonečno spôsobov ako vizualizovať prúd textu a preto je obzvlášť dôležité vybrať správnu metaforu vizualizácie tak, aby bola intuitívna a vizuálne atraktívna. Cieľom tejto práce bolo teda navrhnúť a implementovať novú metaforu pre vizualizáciu prúdu textu v reálnom čase pomocou silovo-riadenej simulácie berúc do úvahy jej použiteľnosť v praxi a jej dostatočnú výpovednú hodnotu. Táto práca je určená pre každého, kto sa cíti byť zahltený textovými dátami alebo len potrebuje efektívny a hravý nástroj na ich analýzu.



# Obsah

<b>Úvod .....</b>	<b>1</b>
<b>1. Vizualizácia textu v reálnom čase .....</b>	<b>2</b>
1.1 Textová komunikácia .....	2
1.2 Vizualizácia.....	3
1.2.1 História vizualizácie .....	4
1.2.2 Vedecká vizualizácia .....	5
1.2.3 Vizualizácia objemových dát.....	6
1.2.4 Vizualizácia informácií.....	6
1.3 Vizualizácia informácií a hĺbková analýza dát .....	6
1.4 Klasifikácia techník vizuálnej hĺbkovej analýzy.....	7
1.4.1 Rozdelenie podľa typu vizualizovaných dát.....	8
1.4.2 Vizualizačné techniky.....	9
1.4.3 Techniky interakcie a skreslenia.....	9
1.5 Techniky vizuálnej analýzy textu.....	10
1.6 Súčasné prístupy k vizualizácii prúdu textu.....	11
1.6.1 STREAMIT .....	11
1.6.2 TwitterScope.....	12
1.6.3 StreamSqueeze .....	13
1.6.4 TweetInfo .....	14
1.6.5 TextPool .....	16
1.7 Ostatné riešenia a zhrnutie .....	17
<b>2. Sémantická analýza textu .....</b>	<b>19</b>
2.1 Kľúčové komponenty pri analýze textu .....	19
2.2 Metódy sémantickej analýzy textu.....	22
2.2.1 Model vektorového priestoru.....	22
2.2.2 Term frequency – inverse document frequency (tf-idf).....	23
2.2.3 Latentná sémantická analýza (LSA) .....	26
2.2.4 Pravdepodobnostná latentná sémantická analýza (pLSA).....	30
2.2.5 Latentná Dirichletova alokácia (LDA) .....	31

<b>3. Silovo-riadená vizualizácia prúdu textu .....</b>	<b>35</b>
3.1 Algoritmy silovo-riadeného kreslenia grafov a ich využitie vo vizualizácii .....	35
3.1.1 Pružinový systém .....	37
3.1.2 Prístup pomocou teoretických vzdialeností .....	37
3.1.3 Techniky kreslenia dynamických grafov .....	38
3.2 Simulácia gravitácie .....	39
3.2.1 Problém $n$ telies .....	40
3.2.2 Simulácia $n$ telies .....	41
3.2.3 Simulácia Barnes-Hut .....	41
3.3 Zhrnutie .....	43
<b>4. Implementácia a vyhodnotenie.....</b>	<b>45</b>
4.1 Implementácia aplikácie ChatVis .....	46
4.1.1 Použité technológie .....	46
4.1.2 Chat .....	46
4.1.3 Proces LDA.....	49
4.1.4 Vizualizácia .....	52
4.2 Pohľad na beh aplikácie .....	60
4.2.1 Vizualizácia prúdu textu .....	61
4.2.2 Interakcia .....	62
4.2.3 Nastavenia.....	62
4.3 Testovanie a vyhodnotenie testov .....	63
4.3.1 Vizualizácia webového portálu DSL.sk.....	63
4.3.2 Vyhodnotenie testu .....	64
<b>Záver .....</b>	<b>68</b>
<b>Literatúra .....</b>	<b>70</b>
<b>Zoznam elektronických príloh.....</b>	<b>74</b>

# Zoznam obrázkov

Obrázok 1, Zobrazenie pohybu planét, Zdroj: [7] .....	4
Obrázok 2, Ťaženie a ústup Napoleona, Zdroj: [8].....	5
Obrázok 3, Klasifikácia vizualizačných techník, Zdroj: [10] .....	7
Obrázok 4: STREAMIT, Zdroj: [11] .....	12
Obrázok 5: TwitterScope, Zdroj: [12] .....	13
Obrázok 6: StreamSqueeze, Zdroj: [13] .....	14
Obrázok 7: TweetInfo, Zdroj: [14] .....	15
Obrázok 8: TextPool, Zdroj: [15] .....	16
Obrázok 9, Získavanie znalostí o texte.....	20
Obrázok 10, Aplikácia metódy SVD na kolekciu dokumentov, Zdroj: [25].....	28
Obrázok 11, Model pLSA, Zdroj: [26] .....	30
Obrázok 12, Grafický model reprezentujúci Latentnú Dirichletovu alokáciu, Zdroj: [29]..	32
Obrázok 13, Najlepšie výsledky dostávame pre hodnotu $T = 300$ , Zdroj: [18] .....	32
Obrázok 14, Simplex troch tém vnorený do simplexu slov pre tri slová, Zdroj: [28].....	33
Obrázok 15, Silovo-riadená vizualizácia grafu, Zdroj: [31] .....	36
Obrázok 16, Gravitačná sila pôsobiaca medzi dvoma telesami rôznych veľkostí.....	39
Obrázok 17, Simulácia 100 prvkov pomocou Barnes-Hut v 2D, Zdroj: [33] .....	42
Obrázok 18, Proces komunikácie s databázou.....	50
Obrázok 19, Návrh rozloženia grafického používateľského rozhrania pre vizualizáciu. ....	53
Obrázok 20, Beh aplikácie ChatVis z pohľadu používateľa.....	60
Obrázok 21, Inferencia pomocou LDA .....	64
Obrázok 22, Inicializácia vizualizácie DSL.sk .....	65
Obrázok 23, Snímka vizualizácie DSL.sk .....	65
Obrázok 24, Snímka vizualizácie DSL.sk .....	66

# Zoznam tabuliek

Tabuľka 1, Priradenie farieb skupinám tém vo vizualizácii .....	64
---	----

# Úvod

Keďže oblasť výskumu vizualizácie prúdu textu je ešte veľmi mladá a rozvíjajúca sa, existuje v súčasnosti len málo plnohodnotných vizualizačných nástrojov použiteľných v profesionálnej analýze textu a oveľa menej pre bežnú používateľskú sféru. Počas prečítania prvej vety úvodu tejto práce používateľa Twitteru odoslali približne 25000 správ, bolo vytvorených okolo 142 nových webstránok, používatelia Wordpressu publikovali zhruba 86 nových blogových príspevkov a bolo odoslaných približne 51 miliónov e-mailov. To je obrovské množstvo textu a potreba analyzovať ho bez nutnosti čítania rastie doslova každou sekundou.

My sme túto dieru na trhu identifikovali a venovali sme jej v tejto práci náležitú pozornosť. Cieľom našej práce je vytvoriť plnohodnotný nástroj na vizualizáciu prúdu textu v reálnom čase aplikovateľný na celú paletu rôznych scenárov použitia, najmä na vizualizáciu textovej komunikácie. A to za použitia moderných webovských technológií, čo našu prácu predurčuje pre jej použitie na webe, prípadne integrovanie do známych webových služieb, čo je oproti doterajšiemu výskumu významný krok vpred. Na výsledkoch tejto práce môže stavať každý výskumník alebo výskumný tím z oblasti vizualizácie textu a naša aplikácia môže byť použitá ako v súkromnom sektore, tak pre komerčné využitie všade, kde je potreba analýzy množstva textu v reálnom čase.

Myšlienka tejto práce môže byť rozvíjaná mnohými smermi a preto je pre autorov výzvou chopiť sa jej a ukázať čitateľovi nový rozmer vo vizualizácii textu.

Práca je rozdelená do štyroch kapitol. V prvej kapitole oboznámime čitateľa so základnými pojmami z prostredia vizualizácie, uvedieme ho do problematiky tejto práce a ponúkneme mu pohľad na doterajší výskum v tejto oblasti. V nasledujúcej kapitole sa presunieme do oblasti sémantickej analýzy textu, zistíme aké sú jej úskalia a navrhujeme použitie konkrétnej metódy použiteľnej v našej práci. V tretej kapitole popíšeme metódy silovo-riadenej vizualizácie, oboznámime sa s používanými algoritmi a simuláciami a ich možnosťami použitia vo vizualizácii prúdu textu v reálnom čase. Posledná kapitola je venovaná implementácii a testovaniu nami zvolenej metódy vizualizácie. Vytvorenú aplikáciu otestujeme a vyhodnotíme výsledky testov. Zhrnieme si výhody a možné nedostatky našej implementácie a následne vyvodíme vlastný prínos v tejto oblasti.

# 1. Vizualizácia textu v reálnom čase

V prvej kapitole tejto práce čo najdetailnejšie opíšeme súvisiacu problematiku a uvedieme do nej čitateľa. Následne uvedieme známe prístupy k riešeniu tejto problematiky. V súčasnosti nie je na trhu mnoho riešení, ktoré by umožnili vizualizáciu textovej komunikácie v reálnom čase. Vo všeobecnosti sa však potýkame s problémom vizualizácie textu v reálnom čase. Predstavíme si teda ciele práce a načrtujeme ich riešenia. V tejto súvislosti sa môžu vyskytnúť rôzne výzvy a problémy, ktorým budeme čeliť. Preto sa pokúsime vhodne popísať očakávané vstupy a výstupy nášho riešenia.

## 1.1 Textová komunikácia

Pojmu *textová komunikácia* intuitívne rozumieme a stretávame sa s ním v praxi bežne. Ide o elektronickú formu textových správ odosielaných po sieti. V tejto práci máme namysli sieť – internet, ale môžu to byť aj iné formy siete, napríklad mobilná sieť.

**Textová komunikácia** môže byť „neštruktúrovaný prúd textových dát, ako sú krátke články, správy na Twittri, alebo používateľské príspevky, ktoré sú generované v obrovských množstvách každým dňom“ [1]. Ďalej ako príklad môžeme uviesť príspevky v službách disponujúcich komunikačnými nástrojmi, ako napríklad Facebook, u nás rozšírený Pokec a pod.

**Prúd textu** uvažujeme ako zväčša nepretržitý dátový tok. Ako je uvedené v práci [2], spĺňa tieto vlastnosti:

- a) „Dátové elementy v prúde prichádzajú online.“
- b) „Systém nemá kontrolu nad poradím v akom dáta prichádzajú., aby boli spracované ako aj vo vnútri dátového toku, tak aj naprieč rôznymi dátovými tokmi.“
- c) „Dátové toky sú potenciálne neohraničené veľkosťou.“
- d) „Akonáhle bol prvok z dátového toku spracovaný, je archivovaný a odstránený. Pokiaľ ho nemáme explicitne uložený v pamäti, tak k nemu nie je možné

prístupovať, alebo obnoviť ho. Prvok dátového toku je typicky relatívne malý oproti veľkosti celého dátového toku.“

V ďalších častiach tejto práce budeme využívať uvedené definície textovej komunikácie.

## 1.2 Vizualizácia

V počítačovej grafike pod pojmom **vizualizácia** rozumieme proces vytvárania obrazovej informácie, pomocou ktorej chceme sprostredkovať používateľovi správu, resp. informáciu o spracovávaných dátach. V praxi na to často využívame obrazy, animácie, diagramy a pod. Proces vytvárania vizualizácie často vedie k vytvoreniu interaktívneho používateľského nástroja, ktorý umožňuje pohodlnejšie spracovať zobrazovanú informáciu.

Základnou ideou je prezentovať dáta v takej forme, aby človek mohol nahliadnuť do týchto dát, vyvodit' následne nejaké závery a zároveň plnohodnotne interagovať s týmito dátami. Takáto metóda je užitočná najmä ak vieme veľmi málo o dátach, ktoré skúmame.

Z existujúcich definícií: „Vizualizácia informácií sa sústreďuje na vývoj a empirickú analýzu metód určených na prezentáciu vo vizuálnej forme. Vizuálne zobrazenie informácie umožňuje ľuďom omnoho ľahšie pochopenie esenciálnych faktov, skôr si uvedomiť podobnosti a odchýlky v dátach a tým pádom hlbšie pochopiť zobrazované dáta. Interaktívna vizualizácia ďalej čerpá zo schopnosti ľudí ľahšie identifikovať zaujímavé fakty, keď sa mení ich zobrazenie a zároveň umožňuje manipulovať so zobrazením dát [3].“ Podľa [4] je to „zobrazenie informácií pomocou priestorových alebo grafických reprezentácií na uľahčenie porovnávania, rozpoznávania vzorov, detegovania zmien a ďalších kognitívnych schopností využitím vizuálneho systému.“ Vizualizáciu informácií vynikajúco popisuje [5] ako „štúdiu ako efektívne prezentovať informáciu vizuálne.“

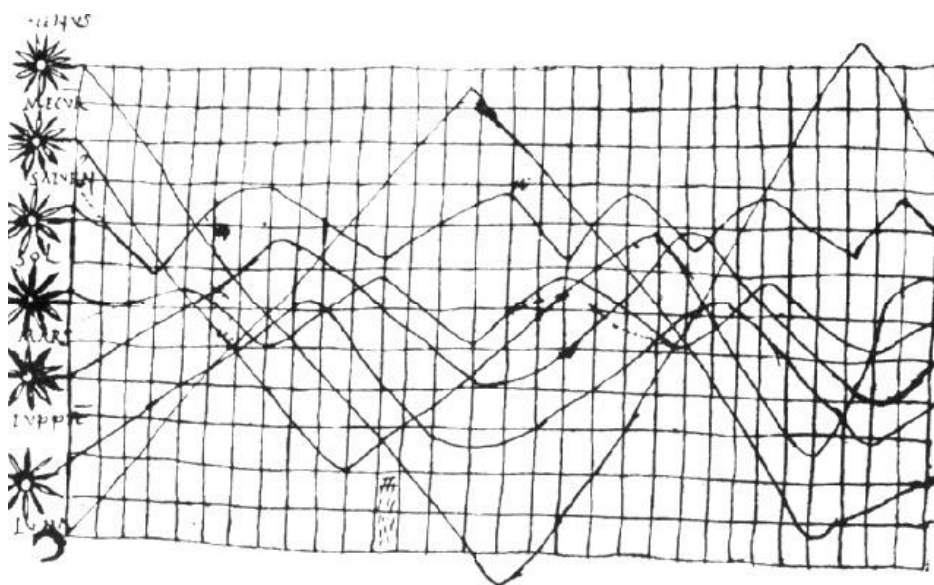
Výhodou grafickej vizualizácie je rýchlosť percepcie oproti ostatným vstupom, ako je zvuk alebo text. Nie je závislá na jazyku a je pomocou nej možné relatívne jednoducho zobraziť konkrétne, ale aj abstraktné dáta.

Krátky pohľad do histórie vizualizácie nám zdôrazní jej potrebu, najmä v dnešnej dobe, kedy ľudstvo disponuje väčším objemom informácií a znalostí ako kedykoľvek predtým. Podľa [6] „sa momentálne znalosť ľudstva zdvojnásobuje každých 5 rokov a očakáva sa,

že okolo roku 2020 sa znalosť ľudstva bude zdvojnásobovať každých 73 dní. Predpokladá sa, že množstvo informácií vo svete sa zdvojnásobuje každé štyri roky.“

### 1.2.1 História vizualizácie

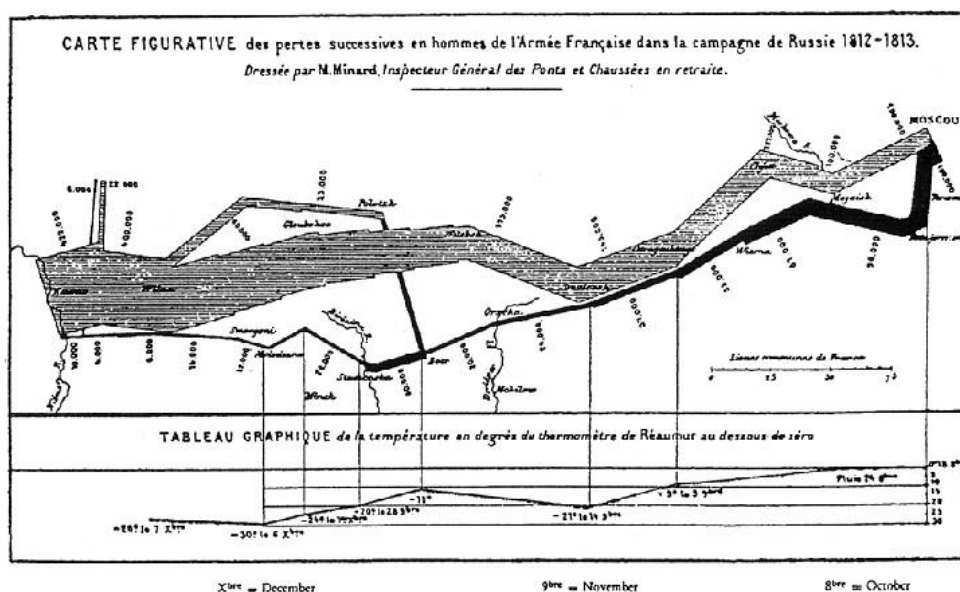
Prvé pokusy o vizualizáciu konkrétnych dát z pozorovaní siahajú až do 10. storočia kedy sa vtedajší pozorovatelia nočnej oblohy pokúšali zaznamenať pohyb planét na oblohe v čase. Ide o pomerne nepresnú vizualizáciu a jej presný pôvod nie je jasný.



Obrázok 1, Zobrazenie pohybu planét, Zdroj: [7]

Jedným z najznámejších diel v histórii vizualizácie je dielo Charlesa Josepha Minarda z roku 1869, ktoré zobrazuje Napoleonovo ťaženie do Ruska odohrávajúce sa v roku 1812. Táto vizualizácia zobrazuje postup vojska ako aj straty spôsobené v boji, zlým počasím či nedostatkom zdrojov a jeho neskorší ústup.





Obrázok 2, Ťaženie a ústup Napoleona, Zdroj: [8]

Vstupom do vizualizácie môžu byť ľubovoľné údaje prichádzajúce z rôznych zdrojov. Pre lepšiu adresáciu problémov je teda vhodné vizualizáciu kategorizovať. V praxi je zaužívaných niekoľko všeobecných kategórií vizualizácie, ale čitateľ by mal brať na vedomie, že mnoho z nich sa vo svojich atribútoch prelína a hranica medzi nimi býva často rozostrená. V neskorších deleniach sme načrtli aj iný pohľad, tam budú rozdiely zrejmé z pohľadu vstupu.

### 1.2.2 Vedecká vizualizácia

Počas vedeckého výskumu je často produkované veľké množstvo dát. Výber týchto dát, ich transformácia a nakoniec ich reprezentácia môže byť vedeckou vizualizáciou. Často ide o taký druh dát, ktoré je možno ľahko geometricky namapovať na dvojrozmerný povrch alebo do trojrozmerného objektu. Ide o veľmi dôležitý druh vizualizácie.

Odvetvia vedy, ktoré využívajú vizualizáciu najviac sú: medicína, astrofyzika, chémia a fyzika. Vo vedeckej vizualizácii využívame techniky počítačovej animácie, simulácie a celkovo samotné odvetvie vizualizácie a to vizualizáciu informácií, ktorú sme popísali v ďalšom texte.

### 1.2.3 Vizualizácia objemových dát

Vizualizácia objemových dát je zväčša podmnožinou vedeckej vizualizácie. V tomto druhu vizualizácie sa používajú špeciálne renderovacie techniky na vizualizáciu dát, a to napríklad rekonštrukcia izopovrchov, renderovanie objemov a pod.

Používame:

- a) povrchové metódy – obrysové prístupy, voxelový model, marching cubes
- b) objemové metódy – priame zobrazovanie izopovrchov, zobrazovanie objemov

Postupujeme nameraním dát (CT, MRI, USG, PET, SPECT) alebo použijeme syntetické dáta (voxelizácia). Následne dáta spracujeme v 2D, tu prevádzame konverziu formátov, redukuje veľkosť dát, filtrujeme prípadný šum a následne v poslednom kroku rekonštruujeme povrch.

### 1.2.4 Vizualizácia informácií

Vizualizácia informácií sa od vedeckej vizualizácie odlišuje najmä tým, že nevizualizujeme dáta namerané z rôznych fyzikálnych experimentov. Vizualizujeme abstraktné dáta ako napríklad siete relácií na internete, bibliografické databázy, text, resp. v prípade našej práce prúdy textu a pod.

V konečnom dôsledku nám vizualizácia umožňuje rýchlejšie skúmať dáta a jej plná sila sa ukazuje v situáciách, kde automatizované algoritmy zlyhávajú.

## 1.3 Vizualizácia informácií a hĺbková analýza dát

V súvislosti s analýzou skúmaných dát sa často stretávame s pojmom *Information Seeking Mantra* [9]. Ide o trojúrovňový proces:

- a) Najprv si sprav široký pohľad
- b) Priblíž a vyfiltruj
- c) A nakoniec detaily na požiadanie

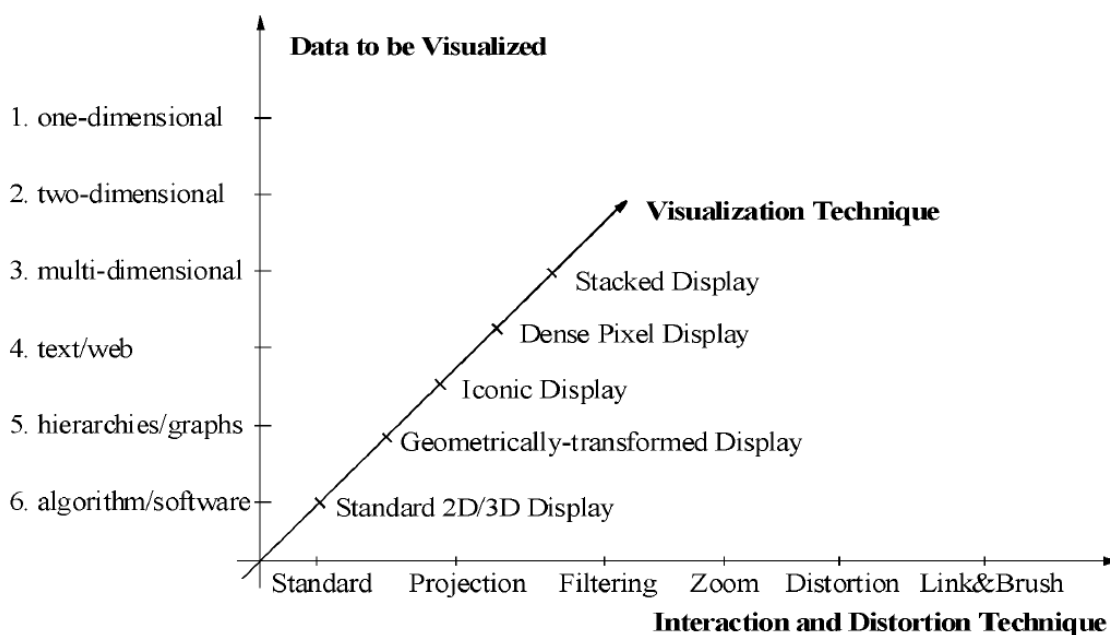
Inými slovami, najprv je potrebné, aby si používateľ spravil celkový prehľad o dátach. Používateľ identifikuje vzory, ktoré ho zaujmú a preniesie na ne svoju pozornosť. Potom používateľ odfiltruje dáta, ktoré identifikoval ako nepotrebné a nakoniec, ak je to potrebné,

tak použije interaktívne prvky vizualizácie na zobrazenie potrebných detailov. „Je potrebné uvedomiť si, že vizualizačné technológie by nemali napomáhať len v uvedených troch krokoch, ale aj vyplňať priestor medzi týmito krokmi [10].“

## 1.4 Klasifikácia techník vizuálnej hĺbkovej analýzy

Vizualizáciu informácií je vhodné použiť v prípadoch ak je potrebné namapovať zložité abstraktné dáta na obrazovku zariadenia. V posledných rokoch bolo vyvinutých mnoho techník vizualizácie. Podľa [10] ich možno klasifikovať na základe troch kritérií:

- Podľa dát, ktoré chceme vizualizovať
- Podľa použitej vizualizačnej techniky
- Podľa použitých interakcií a skresľujúcej techniky



Obrázok 3, Klasifikácia vizualizačných techník, Zdroj: [10]

V tejto podkapitole si priblížime najmä prvú skupinu, do ktorej spadá aj textová informácia.

### 1.4.1 Rozdelenie podľa typu vizualizovaných dát

Vizualizované dáta môžu byť reprezentované rôznou formou. Typicky pri uvádzaní atribútov dát hovoríme o dimenzionalite dát. Dimenzionalitu určujeme podľa počtu premenných v jednom elemente. Dimenzionalita môže byť rôzna, od 1-rozmerných dát až po mnoho-rozmerné dáta. Ďalej poznáme komplexné dátové typy ako napríklad text a hypertext.

Rozdelenie podľa [10]:

- a) **1-rozmerné:** typicky ide o body v čase, kedy na vyjadrenie stačí jedna dimenzia. Každý bod má hodnotu, ktorá je s ním asociovaná. Vyobrazovanie takýchto hodnôt dáva zmysel hlavne ak sú dáta zoradované a jednotlivé body pospájané pre lepšiu vizuálnu reprezentáciu.
- b) **2-rozmerné:** dátové elementy majú dve rozdielne dimenzie. Najčastejším príkladom použitia takýchto prvkov sú mapy a koordináty na nich, zobrazovanie bodov na sieti, alebo iné geografické dáta. Pri vykresľovaní takýchto dát je potrebné si uvedomiť, že príliš mnoho prvkov môže viesť k neprehľadnosti vizualizácie.
- c) **Mnoho-rozmerné:** častým prípadom vo vizualizácii je, keď potrebujeme zobraziť dáta o viac ako dvoch či troch dimenziách. Vtedy sa nám ponúka viacero možností. Môžeme siahnuť po niektorej z bežne používaných techník ako sú paralelné koordináty, kde prvok je vyobrazený ako lomená čiara pretínajúca horizontálne čiary v určitých bodoch zodpovedajúcich hodnote v danej dimenzii prvku. Takýmito dátami môžu byť údaje z databáz a pod. Ak nie je nevyhnutné zobrazovať každú z daných dimenzií, môžeme siahnuť po technikách redukovania dimenzionality ako napríklad PCA.
- d) **Text a hypertext:** niektoré dáta sa nedajú jednoducho reprezentovať číselne a nedá sa na ne aplikovať pojem dimenzionality. Sem patrí práve text. Ten si vyžaduje veľmi špecifický prístup k vizuálnej reprezentácii, preto používame rôzne mapovacie techniky. Jednou z možností je priradovanie atribútov ako početnosť, alebo tf-idf, prípadne pomocou špeciálnych algoritmov vyhľadať v texte latentné údaje a tie použiť v reprezentácii.
- e) **Hierarchie a grafy:** niektoré dáta majú jasne určené vzťahy medzi sebou. Často sú reprezentovateľné sieťami a pod. Hierarchické štruktúry bývajú reprezentované stromami, inými slovami špeciálnymi typmi grafov. Grafy sú množiny bodov –

vrcholov, pospájaných hranami, ak medzi jednotlivými vrcholmi existuje relácia. Hranám a vrcholom často pripisujeme rôzne atribúty ako mieru podobnosti a pod.

- f) **Algoritmy a softvér:** sú samostatnou kategóriou. Algoritmy, resp. programy môžu byť reprezentované tisíckami riadkov kódu. Vizualizácia takéhoto typu dát býva využívaná v návrhoch softvéru, prípadne pri ladení veľkého množstva kódu. Opäť je cieľom sprehľadniť situáciu a poskytnúť používateľovi nadhľad.

#### 1.4.2 Vizualizačné techniky

Poznáme obrovské množstvo vizualizačných techník. Tieto sa môžu líšiť v závislosti od použitého zariadenia, na ktorom budeme dáta vizualizovať. Medzi štandardné techniky patria histogramy, grafy, siete, 2D a 3D ploty a pod.

Medzi špeciálne typy techník patria napríklad **techniky zobrazujúce pomocou geometrických transformácií**. Tieto sa zameriavajú najmä na hľadanie zaujímavostí v multidimenzionálnych dátach. Klasické sú paralelné koordináty, scatterploty a pod.

Ďalej sú zaužívané **techniky ikonických displejov**. Tieto využívajú myšlienku namapovať multidimenzionálne dáta do atribútov ikon. Napríklad to môžu byť mnohouholníky. Ich jednotlivé atribúty ako plocha, uhly medzi hranami, farba, ohraničenie a iné.

**Technika zobrazovania tzv. hustých displejov** sa využíva pri zobrazovaní obrovského množstva dát. Každému typu dát je priradená farba a jednotlivé prvky sú zobrazované ako pixle displeja. Tieto techniky je nevhodné využívať tam, kde sa kladie dôraz na detail, prípadne ich je možné kombinovať s niektorými prvkami interakcie ako tzv. rybie oko. „Najznámejšie spomedzi týchto techník sú technika rekurzívnych vzorov a technika kruhových segmentov [10].“

#### 1.4.3 Techniky interakcie a skreslenia

Techniky interakcie a skreslenia sú vo vizualizácii veľmi dôležité, pretože napomáhajú kvalitnejšej hĺbkovej analýze dát. Používateľ si vďaka nim dokáže ľahko zmeniť kontext, v ktorom robí analýzu vizualizovaných dát.

**Interakčné techniky** umožňujú používateľovi interagovať priamo s vizualizáciou, a tak mu poskytujú možnosť meniť vizualizáciu podľa jeho vlastných potrieb. Interakcia prebieha pomocou na to určených nástrojov v rámci prostredia, v ktorom sú dáta

vizualizované. Takýmto štýlom si je možné ľahko vytvoriť viacero vizualizácií zobrazujúcich tie isté dáta, no iným spôsobom a následne ich porovnávať a tým pádom spraviť kvalitnejšiu analýzu.

**Techniky skreslenia** si kladú za úlohu zobraziť vybrané časti dát vo vyššej úrovni detailu.

Uvedené dve techniky sa veľmi často kombinujú a vytvárajú veľmi silný nástroj. Sem patrí „interaktívne približovanie, interaktívne filtrovanie, dynamické projekcie, interaktívne skresľovanie a kombinácie týchto techník [10].“

## 1.5 Techniky vizuálnej analýzy textu

Nasledovné techniky môžu byť použité k zobrazovaniu štruktúrovaných aj neštruktúrovaných atribútov vstupného textu, ktoré sme získali počas fázy spracovania textu. Najčastejšie sa snažíme získať a znázorniť sémantiku textu, keďže práve tá je jeho najdôležitejším atribútom. Z historického hľadiska k najzákladnejším technikám patrí *sémantické zoskupovanie*, alebo *sémantické mapovanie*. Tieto techniky si kladú za cieľ vytvoriť reprezentáciu veľkých kolekcii dokumentov pri zachovaní všeobecnej informácie o sémantickej povahe textu. Používajú sa v prípade, že používateľovi stačí všeobecná znalosť o vstupnej kolekcii dokumentov.

Medzi prvé takéto techniky patrí **Sammonovo mapovanie**. Je to nelineárna mapovacia technika, generujúca mapy podobnosti medzi malými kolekciami dokumentov. Tieto mapy síce neboli naozajstné sémantické mapy, ale poskytovali iný pohľad na text a jeho vnútornú charakteristiku, takže si právom vyslúžili pozornosť vtedajšej vedeckej komunity. Táto technika mapuje multidimenzionálne dáta do nižších dimenzií. Ostala dlhodobo využívaná v oblasti vizualizácie textu. Algoritmus Sammonovho mapovania sa snaží minimalizovať tzv. Sammonovu chybu

Vývoj v tejto oblasti stagnoval približne 30 rokov, kedy výskum pokročil a začali sa využívať **Kohonenove mapy príznakov**, alebo inak nazývané **Samoorganizujúce sa mapy (SOM)**. Sú jedným zo základných modelov umelých neurónových sietí. Zaradujeme ich medzi modely učiace sa bez učiteľa. Pomocou SOM je možné reprezentovať mnohorozmerné dáta tak, že zachováme ich črty a príznaky. Dáta boli vizualizované ako dvojrozmerná sieť, kde vrcholy siete predstavovali jednotlivé

dokumenty z kolekcie a regióny so spoločnými vlastnosťami boli označené slovnými popiskami.

Začiatky **latentnej sémantickej analýzy** siahajú do 90tych rokov.

## **1.6 Súčasné prístupy k vizualizácii prúdu textu**

V mnohých prípadoch ja vizualizácia textu v reálnom čase kľúčovým aspektom v analýze dát. Preto je v blízkej budúcnosti očakávateľný rozvoj v tomto odvetví výskumu. V súčasnosti je výskum zameraný najmä na vizualizáciu dokumentov resp. textov o veľkom počte znakov. Vizualizácia krátkeho textu si vyžaduje osobitnú pozornosť. Pri tomto type vizualizácie sa kladie dôraz najmä na vyjadrenie sémantiky textu.

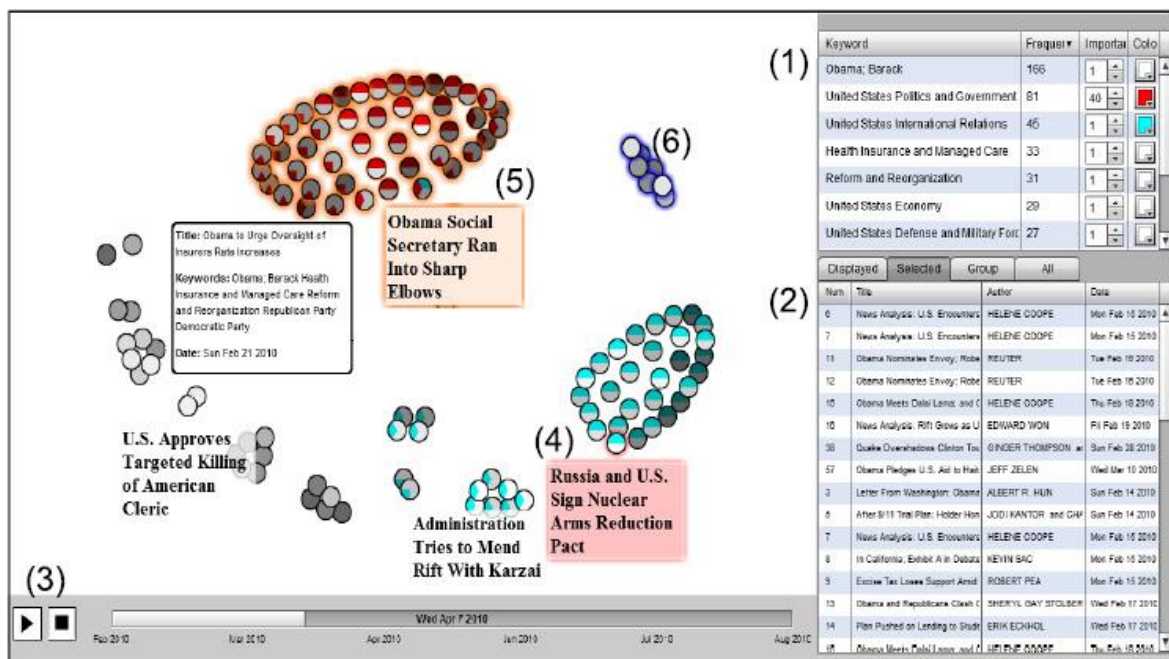
### **1.6.1 STREAMIT**

STREAMIT je interaktívny systém, ktorý umožňuje používateľovi prehliadať veľké korpusy dát v toku bez ich apriórnej znalosti. Systém priebežne načítava dáta zo zdroja a zoskupuje ich v rámci vizualizácie do zhlukov. „Zoskupovanie prebieha na základe podobnosti obsahu dokumentov prichádzajúcich zo zdroja [11].“

Používateľ má k dispozícii vizualizáciu v reálnom čase, čo mu umožňuje reálnu analýzu dát, ale aj možnosť prehliadať evolúciu dát spätne. Na toto používa interaktívne prostredie a animácie.

Na samotnú vizualizáciu si autori zvolili dynamický silovo-riadený systém. Dokumenty sú v ňom reprezentované ako častice ovplyvňované Newtonovskými zákonmi. Každá dvojica častíc má medzi sebou určitú energiu. Podobnosť medzi časticami je vypočítaná vzhľadom na energiu medzi nimi a podľa kľúčových slov, ktoré reprezentujú.

Dôležitou vlastnosťou STREAMITu, je nový dynamický prístup akým sa vypočítava dôležitosť kľúčových slov v texte nazvaný Dynamic Keyword Importance. Kľúčové slovo sa určuje na základe frekvencie jeho výskytu v čase a podobnosti s inými slovami v texte.



Obrázok 4: STREAMIT, Zdroj: [11]

STREAMIT vo vizualizácii využíva rozpoznávanie dokumentov podľa tém, ktoré reprezentujú. Na toto využíva dáta, ktoré sú tematicky príbuzné s tokom textu. Na nich je prv spustená Latentná Dirichletova alokácia, pomocou ktorej sa extrahujú z korpusu jednotlivé témy. V priebehu vizualizovania STREAMIT analyzuje, či prichádzajúci dokument obsahuje extrahované témy pomocou ich kľúčových slov. K reprezentatívnemu klastrovaniu prispieva aj fakt, že počet kľúčových slov zhruba zodpovedá počtu tém.

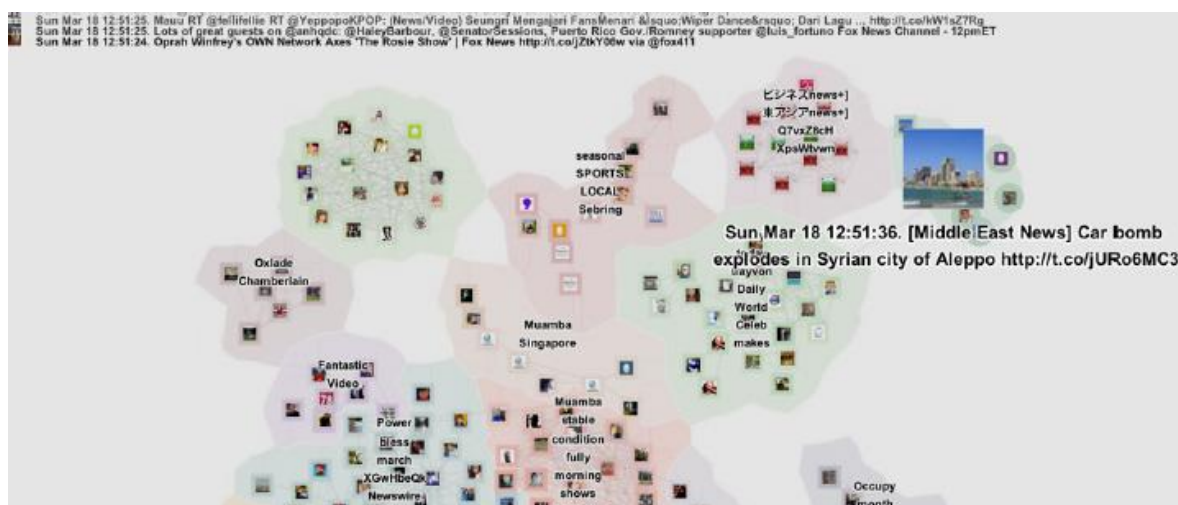
Vizualizácia obsahuje základné okno, kontrolný panel pre animácie, tabuľky kľúčových slov a tém a tabuľku dokumentov. Jednotlivé klastre je možné označovať, čo prispieva k prehľadnosti. Používateľ môže zároveň nastavovať dôležitosť kľúčových slov, označovať klastre, prehľadávať pomocou kľúčových slov, priradovať im farby, či zobrazovať časovú periódu vizualizácie. Na dôvažok je posledná verzia vizualizácie STREAMITu akcelerovaná pomocou GPU.

### 1.6.2 TwitterScope

Aplikácia TwitterScope sa špecializuje na spracovanie a vizualizáciu krátkych textov ako sú správy v službe Twitter, alebo Facebook. Oproti STREAMITu sa TwitterScope musí vysporiadať s problémom vysokého dátového toku. Správy sú síce krátke, často



okolo 160 znakov, no prichádzajú v obrovských množstvách a v krátkych časových intervaloch. Často tisíce správ za sekundu.



Obrázok 5: TwitterScope, Zdroj: [12]

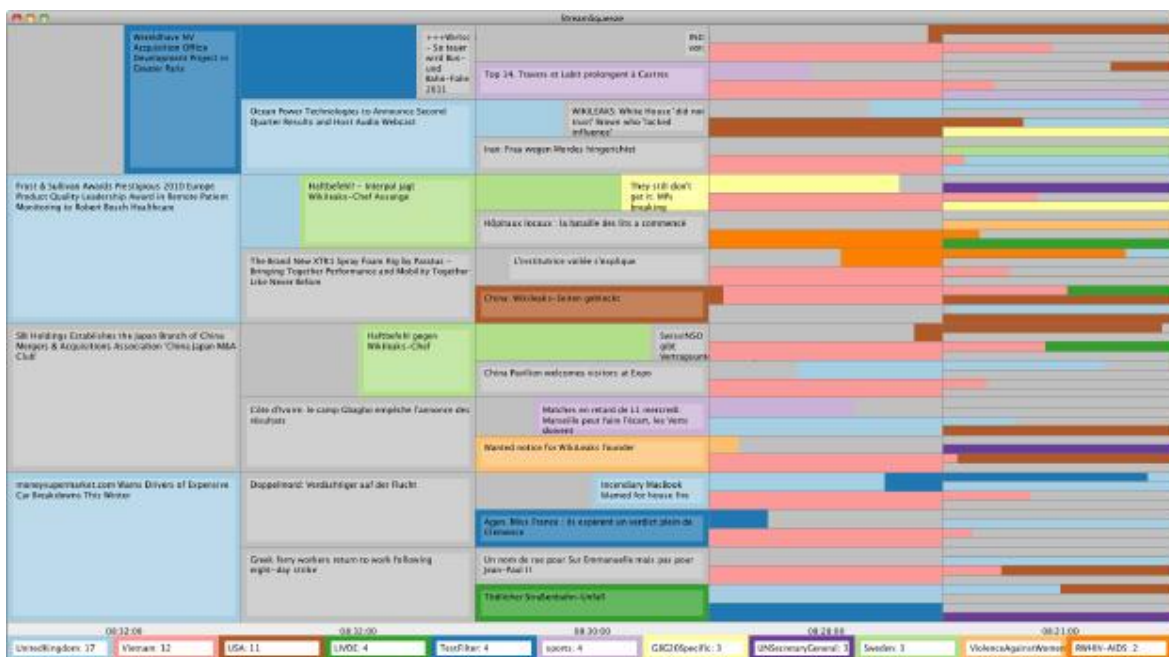
Vizualizácia TwitterScope využíva metaforu mapy. S takouto metaforou je bežne používateľ stotožnený. Jednotlivé tematické celky sú zobrazené vo forme štátov a vybrané kľúčové slová ako mestá. Pre zachovanie vizuálnej stability je použitá Prokrústova transformácia a algoritmus riešiaci problém balenia v 2D pre zachovanie stability jednotlivých komponentov, čo je podľa autorov „inovatívne v tejto oblasti [12].“

Prvým krokom v analýze je klastrovanie správ a vypočítavanie podobnosti medzi správami pomocou LDA alebo tf-idf. Ďalej sa kolekcie správ vyhodnocujú ako graf, pričom každá správa je vrchol a dĺžka hrany je podobnosť správ. Pri dynamickom vyvíjaní sa grafu v čase môže nastať veľká zmena vo vizualizácii, toto sa ošetruje ukotvovaním niektorých vrcholov, prípadne pridávaním umelých hrán medzi vrcholy. V tejto práci sa kladie dôraz na vývoj algoritmu riešaceho problém balenia, s ktorým sa vizualizácia musela vysporiadať.

### 1.6.3 StreamSqueeze

Autori aplikácie StreamSqueeze predstavujú novú obraz-vypĺňajúcu techniku vizualizácie textu. Koncept vizualizácie buduje na dvoch základných myšlienkach [13]:

- „nové udalosti v dátovom toku majú väčšiu relevanciu z ich povahy v čase“
- „hladké prechody vo vizualizácii zjednodušujú schopnosť vystopovať prvky v nej“



Obrázok 6: StreamSqueeze, Zdroj: [13]

Vizualizácia je reprezentovaná jednoduchým binárnym stromom, pričom tento strom reprezentuje aj vývoj v čase, kde koreň sú najnovšie správy a listy stromu najstaršie. Každá úroveň stromu je v rámci vizualizácie stĺpcom, takzvaným listom. Jednotlivé listy reprezentujú časové obdobia. Uzol stromu sa postupne vyplňa správou a neskôr je presunutý do ďalšieho listu. Vďaka takto navrhutej vizualizácii je pri presune správy potrebné minimálne množstvo animácií, čo sa odráža na lepšej čitateľnosti celku.

StreamSqueeze používa pre zobrazovanie tematických okruhov 12 farieb pre hlavné okruhy a šedú farbu pre ostatné. Toto rozdelenie „napomáha dobrej vizuálnej rozlíšiteľnosti [13].“

#### 1.6.4 TweetInfo

V prvom rade autori tejto aplikácie vytvorili interfejs pre jazyk TweepQL, jazyk podobný SQL jazykom. TweepQL je založený na syntaxe SQL jazykov, teda aj disponuje klasickými príkazmi ako SELECT, JOIN a podobne. Je to interfejs, ktorý je nadstavbou pre Twitter API. Umožňuje jednoduchú prácu so streamom od Tweeteru, vytváranie funkcií definovaných používateľom, či ukladanie výsledkov do tabuliek. V selectoch na Twitter stream dokonca umožňuje použitie regulárnych výrazov a iné pokročilé funkcie ako výber podľa GPS koordinátov, či klasifikáciu Twitter streamu podľa nálady na dobré správy a zlé správy.

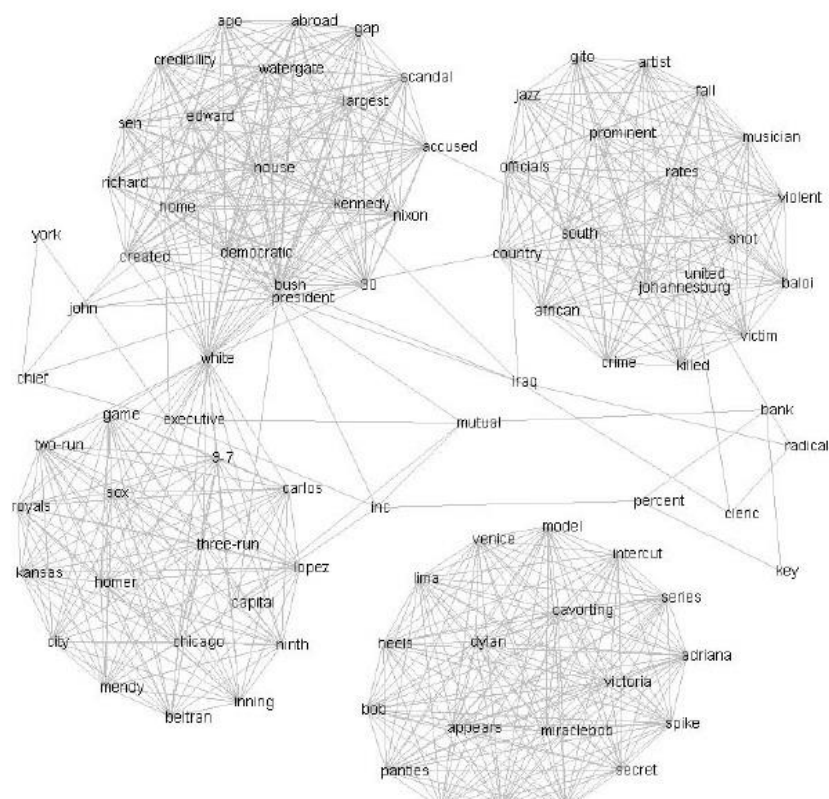
TweetInfo je aplikácia napísaná ako nadstavba pre TweeQL. Je to jednoduchý nástroj umožňujúci monitorovať tok podľa vopred napísaných požiadaviek na Twitter stream.



15

### 1.6.5 TextPool

Jednou z prvých prác v oblasti vizualizácie textu v reálnom čase je TextPool. Autori ním reagujú na stále zvyšujúce sa dostupné množstvo textovej informácie. V tejto práci je zhrnutých mnoho cenných poznatkov a pozorovaní, ktoré však neboli zahrnuté do samotnej vizualizácie.



Obrázok 8: TextPool, Zdroj: [15]

Na vizualizáciu TextPool používa metódu dočasného rozdeľovania (temporal pooling method) a jednoduché vypisovanie kľúčových slov. Na vykresľovanie príbuznosti slov používa spojenie slov hranou v rámci čiastočne spojitého grafu. Vzdialenosti uzlov vyjadrujú mieru príbuznosti. Výrazy, ktoré nemajú nič spoločné, sa vykresľujú v dvojnásobnej vzdialenosti, vzhľadom na maximálnu vzdialenosť výrazov vo vizualizácii, čím sa zdôrazňuje ich odlišnosť. „Aplikácia spracúva tok textu v reálnom čase a extrahuje z neho najčastejšie sa vyskytujúce kľúčové slová reprezentujúce tok a zobrazuje ich ako koláž [15].“

TextPool v rámci spracovania toku textu vytvára pre každý dokument z korpusu takzvaný obsahový vektor salientných výrazov. V podstate ide o asociatívne pole výrazov z dokumentu a ich početnosti v dokumente. Ďalšia sumarizácia dát sa vypočítava mierou salience pre celý tok textu.

Na zvýraznenie zmeny vo vizualizácii sa využíva tradične pohyb. TextPool na toto využíva sériu pravidiel, vďaka ktorým sa zmena javí primeraná ľudskej percepcii. Nové výrazy pribúdajú do vizualizácie a málo salientné výrazy z nej miznú, pričom sa neustále mení dĺžka hrán medzi výrazmi berúc do úvahy zmenu príbuznosti jednotlivých výrazov.

Používateľ má rámci interakcie možnosť nechať si zobrazovať dáta len pre určité výrazy, prípadne meniť časový úsek zobrazovaných výrazov. Toto mu podľa autorov „umožňuje lepšie skúmanie toku textu [15].“

## 1.7 Ostatné riešenia a zhrnutie

Väčšina prác v tejto oblasti sa venuje len vizualizácii textu [16] alebo evolúcie textu v čase [17], či vizualizácii tematických celkov v texte [18]. Len málo z nich sa venuje aj spracovaniu textu v reálnom čase [11] [12] [13] [15] [19]. Z veľkej časti je to dané tým, že „táto oblasť výskumu je ešte veľmi mladá [1] [20].“ Z menšej časti je to potrebou optimalizácie už existujúcich algoritmov pre špecifické potreby v tejto oblasti vizualizácie. Je nesporným faktom, že je ešte mnoho toho, čo sa dá v tejto oblasti vylepšovať.

Každá z prác sa „musí vysporiadať s výzvou aké algoritmy vlastne použije [1].“ Zo spomínaných prác je väčšina zameraná aj na tento aspekt a snaží sa priniesť v tejto oblasti niečo nové. Konkrétne STREAMIT sa špeciálne zaoberá optimalizáciou a odhadom zložitosti jednotlivých algoritmov. Veľkým plusom je akcelerácia pomocou GPU. V našej práci sa zameriame taktiež na optimalizáciu jednotlivých procesov. To sa môže odraziť na kvalite zobrazovaných dát. Tento nežiaduci efekt však možno minimalizovať do čo najnižšej možnej miery použitím vhodných techník popísaných v nasledujúcich kapitolách tejto práce.

Ďalším dôležitým aspektom týchto prác je vysporiadanie sa s modelovaním tém v čase. V tejto oblasti je veľkým prínosom využitie metódy LDA, ktorú používajú najmä STREAMIT a TwitterScope. Pri veľkom počte tém je však nutné sa vysporiadať s hierarchiou tém, čo je možné spraviť manuálne, no výzvou je spraviť to algoritmicky.

Tak isto použitie LDA odstraňuje problémy s inicializačnou fázou vizualizácie, keď je najskôr potrebné spracovať nejaké dáta, pretože LDA je vopred natrénované a informáciu potrebnú k inicializácii už máme.

V real-time vizualizácii toku textu je dôležité priebežne poskytovať používateľovi kontext. Tento nemá byť zmätočný, má v ňom byť ľahko identifikovateľná zmena, no bez zbytočného preťažovania kognitívneho aparátu používateľa. Toto sa dosahuje rôznymi technikami, napríklad metóda temporal pooling z TextPoolu, alebo silovo-riadené systémy zo STREAMITu. V našej práci využívame prvky silovo-riadených systémov, pričom kladieme dôraz na pohyb a animáciu prechodov. Nevýhodou tohto prístupu môže byť zmätočná počiatočná fáza vizualizácie, no vďaka interaktívnym prvkom vizualizácie je možné tento efekt minimalizovať, až úplne odstrániť. Toto vedie k nepreťažovaniu používateľovho kognitívneho aparátu a teda k pozitívnemu zážitku. Širší záber v hĺbkovej analýze prúdu textu umožňujú použité interaktívne techniky, možnosť ukladať aktuálny stav vizualizácie a pod.

Aby sme používateľovi umožnili lepšie analyzovať vizualizovaný text, je veľmi dôležité venovať pozornosť predspracovaniu dát a vhodnému výberu témy-modelujúcich algoritmov. Najmä v slovenskom jazyku to je obzvlášť výzvou, keďže je veľmi modulárny.

V konečnom dôsledku je vizualizácia textu v reálnom čase vyvíjajúca sa téma a ešte v nej bude mnoho čo skúmať.

## 2. Sémantická analýza textu

Pod pojmom *sémantická analýza textu* rozumieme spracovanie kolekcie dokumentov, textu, či prúdu textu a identifikáciu tematických okruhov, ktoré reprezentujú. Následne určenie podobnosti jednotlivých dokumentov z kolekcie, alebo častí textu.

Ako sme uviedli v prvej kapitole, dokument môže byť ľubovoľnej veľkosti a obsahu. Môžeme týmto pojmom označiť krátku správu v textovej komunikácii, alebo dlhý článok na webovom portáli. Tematický okruh resp. **téma** je atribútom popisujúcim obsah dokumentu. Pod týmto pojmom rozumieme vyjadrenie skutočností, ktoré sú z dokumentu známe. Tento atribút nám umožňuje pohodlne kategorizovať dokumenty podľa podobnosti.

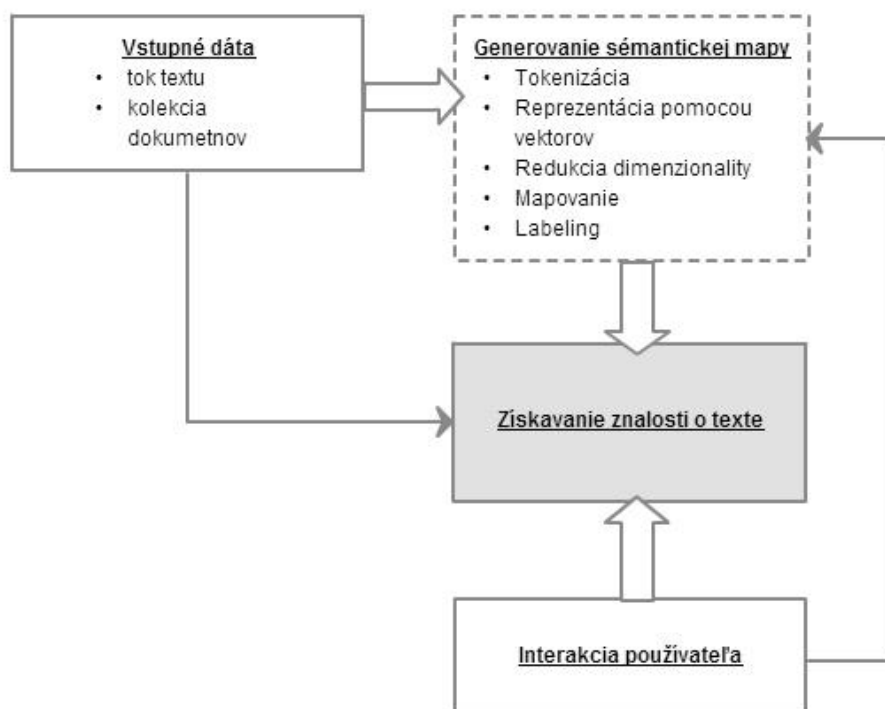
V ďalšom texte si predstavíme postupy pri sémantickej analýze textu a jej vizualizácii, ukážeme si aj jednotlivé metódy sémantickej analýzy od najjednoduchších po komplexné a vyberieme metódu, ktorú použijeme v implementačnej časti tejto práce.

### 2.1 Kľúčové komponenty pri analýze textu

Proces sémantickej analýzy kolekcie dokumentov pozostáva z niekoľkých dôležitých krokov [21]:

- 1) „Tokenizácia“
- 2) „Reprezentácia pomocou vektorov“
- 3) „Redukcia dimenzionality“
- 4) „Mapovanie výstupu z predošlých krokov do 2D alebo 3D priestoru“
- 5) „Označkovanie, tzv. labeling“
- 6) „Interakcia“

Prvých päť krokov tohto procesu označujeme ako generovanie sémantickej mapy. Sú kľúčové pre získavanie znalosti o spracovanom texte a preto si ich popíšeme bližšie.



Obrázok 9, Získavanie znalostí o texte.

Používateľ dokáže získavať znalosť o texte už zo samotného vstupu, no na to je potrebné, aby prehliadal kompletne kolekcie textu. V snahe uľahčiť tento proces sa tok textu neustále spracováva. Tento proces nazývame generovanie sémantickej mapy.

**Tokenizácia** je proces extrakcie textu na lexikálne jednotky. Tie môžu byť slová, frázy, symboly, alebo iné zmysluplné časti textu. V tejto práci pod tokenizáciou budeme rozumieť vytvorenie slovníka unikátnych slov nachádzajúcich sa v texte a ich namapovanie na dokumenty kolekcie prostredníctvom indexov. Často je ale nejednoznačné čo rozumieme pod pojmom *slovo* a preto vykonávame predspracovanie textu, o ktorom si povieme neskôr v práci. Výstupom predspracovania textu bude tzv. *bag-of-words model*, v ktorom bude slová možné ľahko identifikovať ako postupnosti znakov oddelené medzerou.

**Reprezentácia pomocou vektorov** je proces vygenerovania reprezentácie kolekcie dokumentov do modelu vektorového priestoru (popísaný v podkapitole 2.2.1 Model vektorového priestoru). Výstupom je matica čísel. Veľkosť tejto matice je  $|D| \times |W|$ , kde



$|D|$  je počet dokumentov v kolekcii a  $|W|$  je počet slov v slovníku, čo môže byť rádovo v tisíckach slov, prípadne v desať tisícoch.

**Redukcia dimenzionality** je proces, ktorý nie je vždy potrebný. V tomto kroku sa prepočítava maticová reprezentácia kolekcie dokumentov tak, aby výsledná reprezentácia zachovala sémantiku dokumentu a reflektovala ho v čo najvyššej možnej miere, no za súčasného zníženia dimenzionality. Toto má výhodu najmä ak sa snažíme znížiť výpočtový výkon potrebný na prácu s výslednou reprezentáciou. Týmto spôsobom je taktiež možné dobre eliminovať prípadný šum. Na redukciiu dimenzionality sa typicky používajú metódy ako Principal component analysis (PCA) [22] alebo Latent semantic indexing (LSI) [23] využívajúce metódu Singular value decomposition (SVD). Veľkosť matice po spracovaní býva je  $|D| \times |W_R|$ , kde  $|D|$  je počet dokumentov v kolekcii a  $|W_R|$  je veľkosť slovníka po redukcii, čo býva rádovo v desiatkach až stovkách.

**Mapovanie výstupu** je proces, pri ktorom vygenerovaný výsledný model z predošlého kroku chceme pripraviť na vizualizáciu. Kvôli tomuto je potrebné vybrať vhodnú reprezentáciu do 2D alebo 3D a namapovať doň náš vektorový model. Dimenzionalita po tomto kroku býva najviac v desiatkach, pretože takéto atribúty je už pomerne jednoduché vyjadriť veľkosťou objektov, ich farbou, tvarom, umiestnením v priestore a pod. Cieľom mapovania je vytvoriť tzv. *mapu sémantickej podobnosti*, ktorej účelom je zoskupiť sémanticky podobné dokumenty.

Pod pojmom **labeling** máme na mysli proces priradovania označení skupinám prvkov na mape sémantickej podobnosti. Tento krok sprehľadňuje používateľovi vizualizáciu a ponúka mu hlbší pohľad na kontext vizualizácie. Označenia môžeme vygenerovať automaticky, ale dobrým zvykom je nechať používateľovi možnosť tieto označenia si prispôbiť k vlastnej potrebe.

V **interakcii** „spočíva skutočná sila vizuálnej analýzy textu. Môžeme simultánne prezentovať charakteristiku textu a zároveň vizuálne sumarizovať jeho sémantický obsah [21].“ V tomto kontexte sa využíva prvok vizualizácie, ktorý nazývame *metaforou*. Asociovanie metafory pri vizualizácii je vo všeobecnosti dobrý prístup, keďže používateľovi uľahčujeme mentálne sa s ňou stotožniť a pochopiť ju. Pomáha mu udržať si presnú mentálnu reprezentáciu vizualizácie a dotvoriť celkový obraz. Do interakcie spadajú techniky spomínané v kapitole 1.4.3 Techniky interakcie a skreslenia.

## 2.2 Metódy sémantickej analýzy textu

### 2.2.1 Model vektorového priestoru

Je to algebraický model reprezentujúci dokumenty z kolekcie, resp. text. Na identifikáciu dokumentov využíva vektorový priestor, v ktorom je dokument reprezentovaný maticou. Riadky tejto matice reprezentujú jednotlivé vety dokumentu, stĺpce reprezentujú slová (termy) nachádzajúce sa v celom dokumente. Na pozícii  $i, j$  vo vektorom priestore, kde  $i$  je riadok a  $j$  je stĺpec, je booleovská hodnota určujúca, či sa  $j$ -te slovo nachádza v  $i$ -tej vete dokumentu.

Pri konštrukcii modelu jednoduchého vektorového priestoru najprv získame slovník všetkých unikátnych slov nachádzajúcich sa v kolekcii dokumentov. Následne si určíme poradie jednotlivých slov, napríklad abecedné a určíme si indexy jednotlivých slov. V ďalšom kroku zapíšeme booleovské hodnoty do priestoru.

Vyjadrenie  $k$ -teho dokumentu z kolekcie, kde  $t_i$  vyjadruje booleovskú hodnotu pre  $i$ -te slovo zo slovníka:

$$d_k = (t_1, t_2, \dots, t_n)$$

**Príklad.** Majme kolekciu troch dokumentov, kde každý z dokumentov je jednoduchá správa z prúdu textu:

- 1) *Študent píše prácu - píše diplomovú prácu.*
- 2) *Študent horlivo píše diplomovú prácu na intrárovej izbe.*
- 3) *Na susednej intrárovej izbe sa oslavuje.*

Abecedne zoradený slovník použitých slov obsahuje tieto slová:

( 'diplomovú', 'horlivo', 'intrárovej', 'izbe', 'na', 'oslavuje', 'píše', 'prácu', 'sa', 'susednej', 'študent' ).

Každému slovu priradíme index podľa poradia a vytvoríme vektorový priestor pre kolekciu nasledovne:

- 1) ( 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1 )
- 2) ( 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1 )
- 3) ( 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0 )

Vektory tohto vektorového priestoru sémanticky popisujú kolekciu dokumentov, no keďže ide o najjednoduchší spôsob reprezentácie dokumentu, tento prístup prináša aj rad nevýhod. Tento model nie je citlivý na poradie slov, ani na početnosť slov. Nezohľadňuje ohýbanie slov, homonymitu, synonymitu a pod.

Medzi výhody tejto reprezentácie okrem spomínanej jednoduchosti patrí jednoduchý výpočet podobnosti dokumentov, napríklad pomocou výpočtu uhla, ktorý zvierajú jednotlivé vektory medzi sebou.

### 2.2.2 Term frequency – inverse document frequency (tf-idf)

Predchádzajúci uvedený model môžeme vylepšiť, ak vezmeme do úvahy aj početnosť slov, tzv. *term-frequency*, ktorá priradzuje slovám zo slovníka váhu.

**Príklad.** Vyššie uvedená kolekcia dokumentov po zahrnutí početnosti slov vyzerá nasledovne:

1) ( 1, 0, 0, 0, 0, 0, 2, 2, 0, 0, 1 )

2) ( 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1 )

3) ( 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0 )

Nie je to veľká zmena, ale poskytuje nám dôležitú dodatočnú informáciu o kolekcií. Nevýhodou však je fakt, že slová, ktoré nám zo sémantického hľadiska veľa nepovedia majú väčšinou veľkú váhu v kolekcií. Sú to slová ako častice, spojky, skratky a pod. Túto nevýhodu môžeme značne ošetriť aplikovaním techník predspracovania textu, ktorým sa budeme venovať neskôr v práci.

Ďalším vylepšením je tzv. *inverse document frequency*. Je to miera toho či je slovo zriedkavé alebo bežné v rámci dokumentu. Kolekcia dokumentov bude vyjadrená vektormi váh pre jednotlivé dokumenty z kolekcie:

$$\mathbf{v}_d = [w_{1,d}, w_{2,d}, \dots, w_{N,d}]^T, \text{ kde}$$

$$w_{t,d} = tf_{t,d} \cdot \log \frac{|D|}{|\{d' \in D \mid t \in d'\}|}, \text{ pričom}$$

$$tf_{t,d} = \frac{f_{t,d}}{\max\{f_{w,d}: w \in d\}}$$

$tf_{t,d}$  je frekvencia slov v dokumente ako sme si uviedli v poslednom príklade, kde  $f_{t,d}$  je čistá frekvencia slov v dokumente a  $\max\{f_{w,d}: w \in d\}$  je maximálna frekvencia výskytu ľubovoľného slova v dokumente.

$\log \frac{|D|}{|\{d' \in D | t \in d'\}|}$  je inverzná frekvencia dokumentu počítaná ako logaritmus z podielu medzi celkovým počtom dokumentov a počtom dokumentov obsahujúcich aktuálne slovo  $t$  zo slovníka.

**Príklad.** Po vypočítaní jednotlivých hodnôt a dosadení do predošlého príkladu (kde  $|D| = 3$ ) dostávame:

1)	0,264	0	0	0	0	0	0,176	0,176	0	0	0,264
2)	0,176	0,477	0,1761	0,176	0,176	0	0,176	0,176	0	0	0,176
3)	0	0	0,176	0,176	0,176	0,477	0	0	0,477	0,477	0

Z uvedeného vidno, že často opakované slová majú nižšiu váhu. Takto sme vlastne odstránili spomínané nevýhody klasického *term-frequency* modelu. Vo všeobecnosti môžeme tvrdiť, že výhody tf-idf oproti obyčajnému modelu vektorového priestoru sú najmä ováhovanie slov, kedy nepoužívame booleovské hodnoty, ale sme precíznejší. Takto vieme presnejšie vypočítať aj podobnosť medzi jednotlivými vektormi a teda vieme už niečo málo povedať aj o sémantike kolekcie dokumentov.

Nevýhodou tejto metódy je, že stále nerieši synonymitu ani homonymitu, a že slová musia presne sedieť so slovami v slovníku, aby sme ich mohli dobre vyhodnotiť. Tento model zároveň predpokladá, že jednotlivé slová sú štatisticky nezávislé. Pri veľkých dokumentoch a veľkom slovníku je dosť pravdepodobné, že vektory budú obsahovať mnoho núl. Avšak v spolupráci s technikami ako PCA, alebo SVD je možné niektoré z týchto nevýhod prekonať.

**Kosínusová podobnosť** je miera medzi dvojmi vektormi určujúca kosínus uhla, ktorý tieto vektory zvierajú, a teda pomocou nej vieme určiť mieru ich podobnosti. Podobnosť vypočítame zo vzťahu:

$$\text{podobnosť} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$

Kde *podobnosť* môže nadobúdať hodnoty od -1 do 1, pričom hodnotu 1 nadobúda ak majú dva vektory zhodnú orientáciu, teda sú si najviac podobné. Pre kolmé vektory to je hodnota 0 a pre diametrálne odlišné vektory -1. Vtedy sú vektory smerujúce od seba. Pri počítaní podobnosti dvoch dokumentov sa uspokojujeme s rozsahom medzi 0 a 1, pričom 0 znamená úplne odlišné a 1 znamená rovnaké. V metódach sémantickej analýzy je logickým krokom určiť si mieru podobnosti medzi jednotlivými dokumentmi. V prípade, že v prúde textu máme nový dokument, tak je už relatívne jednoduché určiť jeho podobnosť s kolekciou dokumentov a kategorizovať ho.

Ako sme popísali v predošlom texte, veľkou nevýhodou týchto metód je obrovská dimenzionalita vektorového priestoru reprezentujúceho kolekciu dokumentov. Nasledujúce metódy riešia tento problém zaujímavým spôsobom a to detegovaním tzv. *tematických okruhov*. Výhodou tohto prístupu je, že si vopred stanovíme ich počet a to je potom veľkosť dimenzie, s ktorou pracujeme. Každý výraz z kolekcie bude teda reprezentovaný vektorom priradujúcim príslušnosť do jednotlivých tematických okruhov. Tematické okruhy vytvárame automaticky, nie manuálne, pretože toto by mohlo viesť ku skresleniu informácie a nepopisovalo by to skutočnosť čo najvernejšie.

**Príklad.** Majme zoznam slov z predošlého príkladu: ( ‘*diplomovú*’, ‘*horlivo*’, ‘*intrákovej*’, ‘*izbe*’, ‘*na*’, ‘*oslavuje*’, ‘*píše*’, ‘*prácu*’, ‘*sa*’, ‘*susednej*’, ‘*študent*’ ). Určíme si tri tematické okruhy pre tento zoznam, pomocou ktorých budeme reprezentovať dokumenty, napríklad: „*škola*“, „*zábava*“ a „*architektúra*“. Zaradíme jednotlivé slová do kategórií a priradíme im váhy, ktoré prispeli k zaradeniu do tematického celku.

1 kategória) slová: „*diplomovú*(1)“, „*intrákovej*(1)“, „*píše*(2)“, „*prácu*(2)“, „*študent*(1)“

2 kategória) slová: „*horlivo*(1)“, „*intrákovej*(1)“, „*oslavuje*(1)“, „*študent*(1)“

3 kategória) slová: „*intrákovej*(1)“, „*izbe*(1)“

Z uvedeného príkladu je zrejmé, že priradenie slov je v takejto reprezentácii nejednoznačné a niektoré výrazy môžu spadať do viacerých kategórií, čo je vlastne

odrazom reality. Pre presnejšiu klasifikáciu je dôležité správne vybrať počet tematických okruhov.

### 2.2.3 Latentná sémantická analýza (LSA)

V prvom rade si je potrebné uvedomiť význam pojmu „*latentná sémantika*.“ LSA sa od predošlých modelov líši tým, že matica podobností už nie je jednoduchou reprezentáciou frekvencií výskytov slov alebo fráz, ale je podporená silným matematickým aparátom, ktorý je schopný zaznamenať skrytú sémantiku v texte. Toto je samozrejme náročnejšie na výpočet. LSA, ako ostatné metódy, čerpá svoju znalosť len z kolekcie dokumentov.

Výsledky analýzy modelu LSA sú porovnateľné s tými, ktoré by vykonal človek, teda výhoda takejto automatizácie je jasná, no napriek tomu tento model má problémy v rôznych doménach analýzy, ktoré si spomenieme neskôr.

LSA je podľa [24] „plne automatická matematicko-štatistická technika extrakcie a inferencie vzťahov očakávaného použitia slov v pojednávaných pasážach textu. Nie je to tradičný procesor prirodzeného jazyka alebo program umelej inteligencie. Nepoužíva slovníky vytvorené človekom, sémantické siete, gramatiky, syntaktické parsery ani morfológie. Ako vstup používa čistý text prevedený do slov, definovaných ako reťazce znakov oddelených do zmysluplných pasáží ako sú vety alebo odstavce.“ Výstupom je zobrazenie kolekcie dokumentov do priestoru latentných sémantických dimenzií, z ktorého vieme povedať, že dokumenty nachádzajúce sa v rovnakej dimenzii sú si sémanticky podobné viac ako dokumenty nachádzajúce sa v rozdielnych dimenziách. Toto závisí od úrovne dimenzie, čím vyššia úroveň, tým väčšia miera podobnosti.

Analýza pomocou LSA pozostáva z niekoľkých hlavných krokov:

- 1) Predspracovanie kolekcie dokumentov
- 2) Vytvorenie matice početnosti
- 3) Ováhovanie matice početnosti
- 4) Aplikácia metódy dekompozície matice početnosti na singulárne hodnoty (SVD)
- 5) Výpočet matice podobnosti dokumentov

**Predspracovanie kolekcie dokumentov** je dôležitou súčasťou každej metódy analýzy textu. Je to prvý krok k príprave kolekcie dokumentov na ďalšie spracovanie, a preto jej treba venovať náležitú pozornosť. Medzi najdôležitejšie prvky predspracovania textu patrí:

- a) Konverzia znakovkej sady na kódovanie UTF-8
- b) Odstránenie diakritiky (konverzia na ASCII).
- c) Odstránenie všetkých znakov okrem znakov abecedy.
- d) Prevod všetkých písmen na malé písmená abecedy.
- e) Nahradzovanie všetkých prebytočných medzier práve jednou medzerou.
- f) Vymazávanie tzv. *stop-slov* – sem patria slová ako spojky, predložky, zámená a pod. Príklad prebratý z implementačnej časti tejto práce môže byť: 'a', 'aby', 'aj', 'ak', 'ako', 'ale', 'alebo', 'and', 'ani', 'ano', 'asi', 'az' ... Základné zoznamy stop-slov pre jazyky sú bežne dostupné. Často ich ale rozširujeme podľa aktuálnej potreby, najmä podľa toho na akú kolekciu dokumentov ideme zoznam použiť.
- g) Vymazávanie slov, ktoré majú nízku frekvenciu výskytu – toto si nemôžeme dovoliť ak potrebujeme zachovať špecifické detaily dokumentov.
- h) Vymazávanie krátkych dokumentov – tento krok robíme len v predspracovaní textu pre trénovaciu fázu algoritmu ak je potrebná. V prípade, že by sme tento krok vykonali aj v predspracovaní textu z prúdu textu, tak sa nám môže často stať, že výsledkom bude prázdny reťazec väčšinu času, najmä ak v prúde textu máme správny z textovej komunikácie, krátke správy zo služby Twitter (okolo 140 znakov) a pod.
- i) Vymazávanie krátkych slov – to sú väčšinou skratky a pod.
- j) Lematizácia – zmena slova na jeho základný tvar. Používa sa za účelom redukcie slovníka a v snahe zvýšiť kvalitu porovnávania dokumentov ak pozostávajú z rovnakých slov, no iných tvarov. V slovenskom jazyku je tento proces veľmi komplikovaný a existuje preň len málo hotových riešení. Tento proces môže zahŕňať aj také úlohy ako porozumenie textu, keďže slová môžu mať iný význam v rôznych častiach textu.
- k) Stemming – sa používa za tým istým účelom ako lematizácia no princíp je mierne odlišný. Je to zmena slova na jeho koreň. Týmto prevodom redukujeme všetky morfológické varianty slova na jedno slovo, tzv. *koreň*. V mnohých jazykoch sa stáva, že koreň slova nie je platným slovom v danom jazyku.

**Vytvorenie matice početnosti** sa robí rovnakým spôsobom ako bolo spomínané v predošlom texte v príklade pre Term frequency – inverse document frequency (tf-idf). Výstupom tohto kroku je matica o veľkosti  $n \times m$ , pričom  $n$  je počet slov v slovníku a  $m$  je počet dokumentov v kolekcii,  $i$ -tym riadkom matice je  $i$ -ty dokument z kolekcie

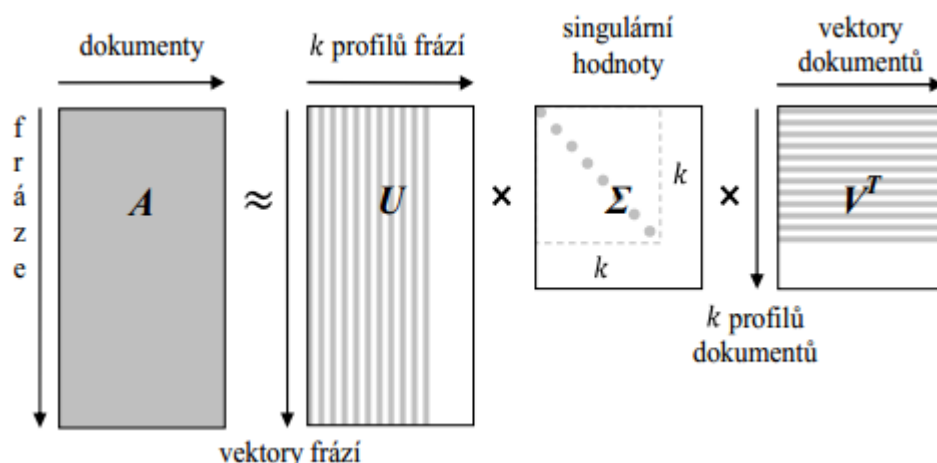
dokumentov a  $j$ -tym stĺpcom je  $j$ -te slovo zo slovníka použitých výrazov. Na pozícii  $i, j$  sa nachádza údaj o početnosti výrazu v dokumente.

**Ovážovanie matice početnosti** sa podľa [25] robí použitím vzorca:

$$a_{i,j} = \begin{cases} \frac{1}{2} + \frac{PF_{i,j} \cdot \log\left(\frac{|n|}{DF_j}\right)}{2 \cdot \max(PF_{i,j}) \cdot \log(|n|)} & \text{ak sa slovo } j \text{ nachádza v dokumente } i \\ 0 & \text{inak} \end{cases}$$

Tento vzorec vznikol modifikáciou klasického tf-idf váhovania uvedeného v kapitole 2.2.2. „Hlavný rozdiel oproti tf-idf spočíva v idf normalizácii [25].“ Ľahko vidno, že výraz  $a_{i,j}$  nadobúda hodnoty z intervalu  $\langle 1/2, 1 \rangle$  ak sa slovo nachádza v dokumente a 0 ak nie.

V kroku **aplikácie SVD** sa zisťuje miera príslušnosti slov k témam a tém ku dokumentom z kolekcie dokumentov a **vypočítava sa matica podobnosti dokumentov**. Ako pracuje metóda rozkladu matíc na singulárne hodnoty najlepšie uvidíme na obrázku a následne si metódu popíšeme.



Obrázok 10, Aplikácia metódy SVD na kolekciu dokumentov, Zdroj: [25]

SVD rozkladá maticu početnosti (na obrázku znázornená ako matica  $A$ ) na súčin troch nezávislých matíc  $U\Sigma V^T$ . Táto metóda je veľmi blízka metóde dekompozície na vlastné vektory matice. Na obrázku je  $k$  – zvolený faktor aproximácie pri prevode do latentného priestoru, ide o počet tém, ktoré chceme identifikovať v kolekcii dokumentov. Po



dekompozícii jednotlivé matice reprezentujú:  $U$  – stĺpce sú singulárnymi vektormi fráz z dokumentov;  $V$  – jej riadky predstavujú singulárne vektory dokumentov;  $\Sigma$  – je diagonálna matica singulárnych hodnôt.

Po tomto kroku je potrebné podotknúť, že metóda LSA na rozdiel od predošlých metód pracuje s priestorom tzv. latentných dimenzií a nepracuje len so slovami ako takými, ale berie do úvahy aj ich výskyt. Inými slovami – berie do úvahy či slová v dokumente tvoria nejakú frázu, a teda sa v dokumente nachádzajú blízko seba, resp. v rovnakom dokumente a pod. Podľa toho rozlišujeme aj úrovně spoločného výskytu slov. Vyššia úroveň znamená vyššiu váhu.

Výstupom tohto kroku je kolekcia dokumentov namapovaná do redukovaného priestoru latentných dimenzií resp. latentného sémantického priestoru. Hodnoty tohto výstupu môžu byť skreslené kvôli redukcii, a preto sa zvyčajne robí dodatočný výpočet konečnej matice podobnosti.

**Príklad.** Výstupom pre ováhovanie jednotlivých dokumentov (fiktívnych) môže byť:

- 1)  $1.2 * slovo1, 0.5 * slovo2, 0.1 * slovo10 \dots$
- 2)  $1 * slovo2, 0.7 * slovo3, 0.01 * slovo4 \dots$
- 3)  $0.2 * slovo5, 0.2 * slovo6, 0.2 * \dots$
- 4)  $\dots$

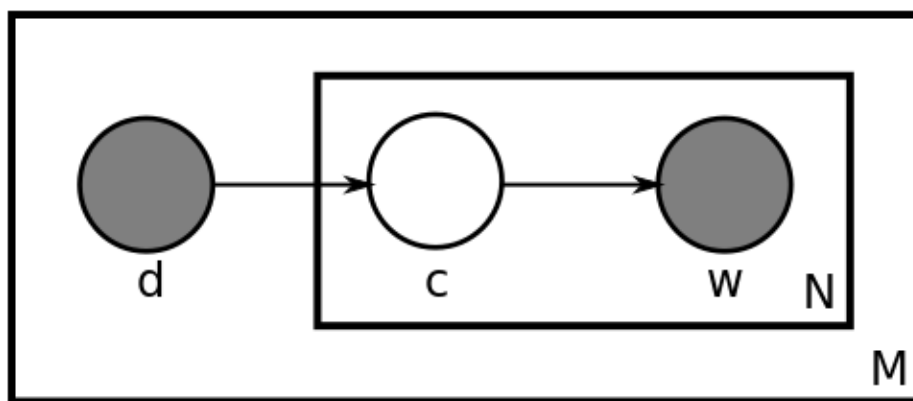
Pre jednoduchosť zápisu sme slová s nulovým zastúpením vynechali. Po aplikovaní kroku dekompozície a vyjadrení jednotlivých tém môže výstup vyzeráť nasledovne (počet tém sme si určili na 3):

- 1) téma:  $1.2 * slovo1 + 0.2 * slovo3 + \dots$
- 2) téma:  $0.2 * slovo2 + 0.2 * slovo3 + \dots$
- 3) téma:  $0.5 * slovo4 + 0.5 * slovo5 + \dots$

Limity LSA spočívajú v možnej nejednoznačnosti interpretácie vygenerovaných tematických okruhov. Niekedy ale tento krok spraviť nepotrebujeme a uspokojujeme sa so základným výstupom. Ďalej nedokáže zachytiť viacznačnosť slov.

### 2.2.4 Pravdepodobnostná latentná sémantická analýza (pLSA)

pLSA je štatistická metóda používaná na analýzu podobnosti v dátach pomocou príbuznosti k rôznym latentným prvkom. pLSA vznikla z LSA pridaním prvku pravdepodobnosti. Na rozdiel od LSA používa namiesto dekompozície na singulárne vektory, tzv. model latentných tried, resp. je založená na dekompozícii odvodennej z tohto modelu. Zaznamenávajú sa doň dáta spoluvýskytu výrazov a teda asociuje latentné premenné s pozorovaniami.



Obrázok 11, Model pLSA, Zdroj: [26]

Platňová reprezentácia na obrázku popisuje model pLSA, kde  $M$  je veľkosť kolekcie dokumentov,  $N$  je veľkosť slovníka dokumentu,  $d$  je index dokumentu z kolekcie dokumentov,  $c$  je téma slova získaná z distribúcie tém pre dokument  $P(c/d)$ ,  $w$  je slovom získaným z distribúcie slov z témy tohto slova  $P(w/c)$ . Premenné  $d$  a  $w$  sú pozorovateľné premenné a téma  $c$  je latentná premenná.

Pravdepodobnosť výskytu slova  $w$  v dokumente  $d$  je daná vzťahom:

$$P(w, d) = \sum_c P(c)P(d|c)P(w|c) = P(d) \sum_c P(c|d)P(w|c), \text{ kde}$$

v strednej časti vzťahu sa  $d$  aj  $w$  vypočítavajú z latentnej témy  $c$  podobným spôsobom a to použitím podmienených pravdepodobností  $P(d|c)$  a  $P(w|c)$ , ktoré samozrejme dopredu nepoznáme, a preto si ich potrebujeme vypočítať, napríklad pomocou algoritmu *Expectation-maximalization*. V poslednej časti vzťahu sa téma vypočítava podmiennečne z  $P(c|d)$  a slovo je potom vypočítavané následne z triedy ako  $P(w|c)$ . Stredná časť výrazu

sa nazýva symetrická parametrizácia, pravá časť výrazu sa nazýva asymetrická parametrizácia.

Výstupom tejto metódy je vektor tém podobný ako pri metóde LSA, kde jednotlivé zložky vektoru už nie sú jednoduché váhy, ale ide o mieru pravdepodobnosti, s ktorou dané slovo spadá do témy určenej vektorom. Podobnosť tém, resp. vektorov, ktoré tieto témy reprezentujú môžeme merať ako v predošlých metódach kosínusovou mierou podobnosti, ale môžeme použiť aj špecializované miery ako je tzv. *Hellingerova vzdialenosť* určená na porovnanie podobnosti pravdepodobnostných distribúcií.

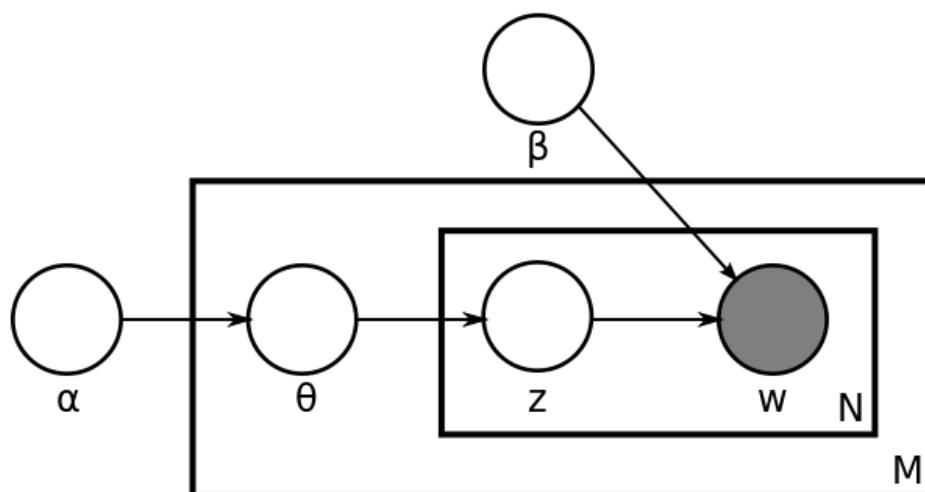
Okrem iného, výhodou oproti modelu LSA je, že rieši problém mnohoznačnosti slov ako ukázali napríklad v [27].

Medzi nedostatky tejto metódy patrí tendencia k preučovaniu. Je to spôsobené tým, že ak pridáme k natrénovanej množine ďalší dokument, tak nám stúpne aj počet parametrov, ktoré potrebujeme znova odhadnúť, keďže počet parametrov vstupujúcich do pLSA je daný aj počtom dokumentov z kolekcie. Najväčším nedostatkom tejto metódy je ale nemožnosť určiť témy pre nové dokumenty, čo je vlastne kľúčová vlastnosť potrebná pri sémantickej analýze toku textu. V porovnaní s LSA ide o výpočtovo náročnejšiu metódu no vo všeobecnosti je rozdiel zanedbateľný.

### **2.2.5 Latentná Dirichletova alokácia (LDA)**

Všetky doteraz uvedené metódy predpokladajú tzv. *bag-of-words model*. Teda predpokladajú, že slová v modeli môžu byť ľubovoľne pomiešané. Toto nazývame predpokladom zameniteľnosti, čo znamená, že „slová (alebo dokumenty) sú podmienne nezávislé a identicky distribuované, pričom podmienenosť je daná latentným parametrom pravdepodobnostnej distribúcie [28].“ Inými slovami, sémantika v reprezentácii dokumentu sa nestratí ani pri rôznej podmienenej distribúcii slov. Metóda LDA je založená na tomto predpoklade a jej autori vytvorili model, ktorý zachycuje zameniteľnosti medzi slovami, ale aj medzi dokumentmi. Toto vyúsťuje do riešenia problému ako mal napríklad model pLSA, kde nebolo možné určiť témy pre nové dokumenty.

Latentná Dirichletova alokácia je „generatívny pravdepodobnostný model kolekcie dokumentov. Základnou myšlienkou je, že dokument je reprezentovaný ako náhodné zmesi nad latentnými témami, pričom každá téma je charakterizovaná distribúciou cez slová [28].“

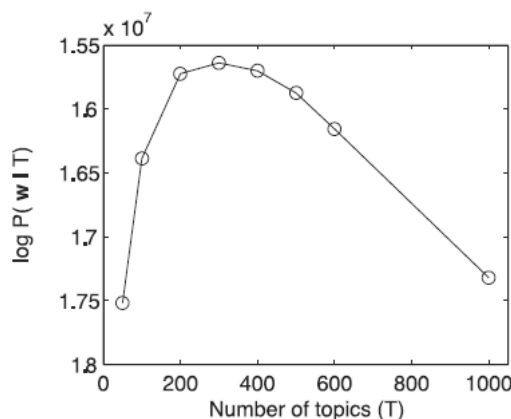


Obrázok 12, Grafický model reprezentujúci Latentnú Dirichletovu alokáciu, Zdroj: [29]

$M$  reprezentuje počet dokumentov v kolekcii,  $N$  reprezentuje počet slov v dokumente.  $\alpha$  je parametrom Dirichletovho radu distribúcie medzi dokumentmi z kolekcie a témami,  $\beta$  je parametrom Dirichletovho radu distribúcie medzi témami a slovami zo slovníka.  $\theta$  vyjadruje distribúciu tém pre aktuálny dokument,  $z$  vyjadruje tému pre aktuálne slovo  $z$  aktuálneho dokumentu a  $w$  je aktuálne slovo. Jedinou pozorovateľnou premennou je aktuálne slovo (neberúc do úvahy počet slov, počet dokumentov a počet tém, ktoré chceme určiť) a všetky ostatné premenné z modelu nazývame latentnými.

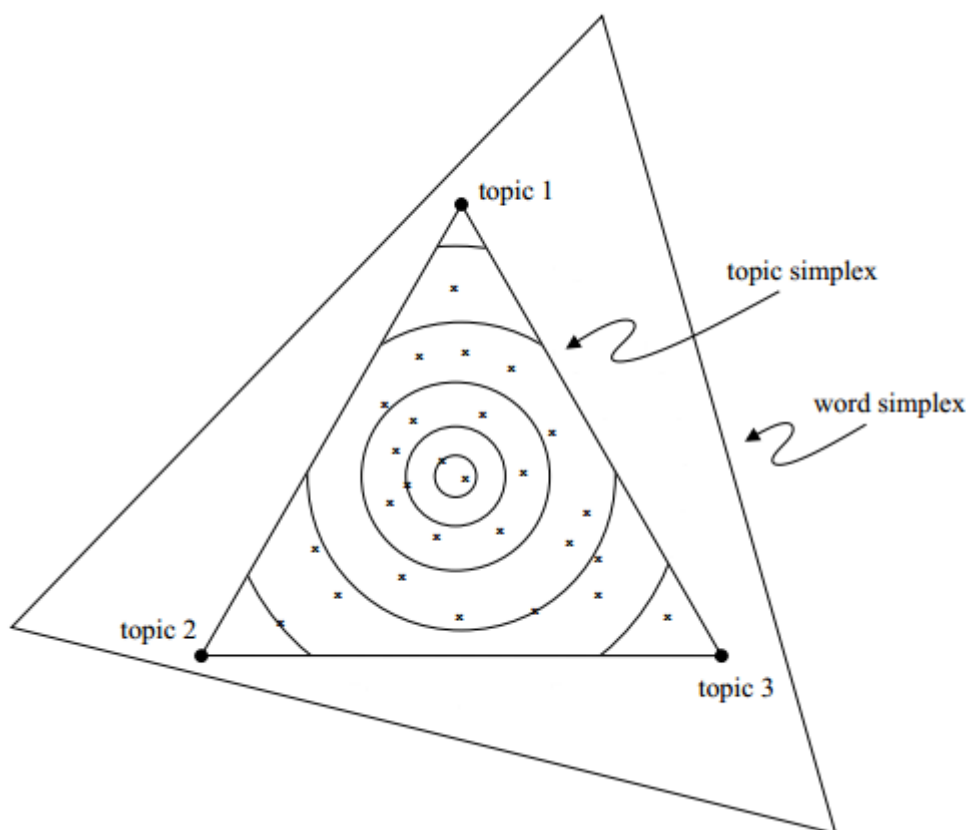
Z grafickej reprezentácie LDA vidíme, že tento model pozostáva z troch základných úrovní:

- 1) Určenie parametrov  $\alpha$  a  $\beta$ , nazývame ich hyperparametre Dirichletovho radu. Tento krok robíme len jeden raz počas inicializačnej fázy modelu. Podľa [18] tieto parametre určíme pre  $\alpha = 50 / T$ , kde  $T$  je počet tém, ktoré chceme určiť a  $\beta = 0,1$ .



Obrázok 13, Najlepšie výsledky dostávame pre hodnotu  $T = 300$ , Zdroj: [18]

- 2) Premenná  $\theta$  sa určí raz pre každý dokument.
- 3) Premenné  $z$  a  $w$  sa určujú raz pre každý dokument a pre každé slovo v ňom. Teda téma sa určuje pre každé slovo z každého dokumentu vytiahnutím z distribúcie s náhodne zvoleným parametrom. Tento parameter je zvolený z hladkej distribúcie nad simplexom tém.



Obrázok 14, Simplex troch tém vnorený do simplexu slov pre tri slová, Zdroj: [28]

Na geometrickej interpretácii latentného priestoru LDA vidíme simplex slov zo slovníka. Každá distribúcia môže byť reprezentovaná ako bod tohto simplexu. Rohy simplexu predstavujú tri rôzne distribúcie, kde v každej z nich má slovo pravdepodobnosť 1. Modely latentných premenných ako LDA alebo pLSA vyberú  $k$  bodov zo simplexu (v našom prípade  $k = 3$ ) a vytvoria podpriestor nazývaný simplex tém. Rohy simplexu pre témy predstavujú tri rôzne distribúcie nad slovami zo slovníka. Z interpretácie vidno, že v LDA v porovnaní s pLSA určuje pre témy hladšiu distribúciu, na obrázku zobrazené ako kružnice. Distribúcia pLSA je zobrazená krížikmi.

Podobne ako pri pLSA sa snažíme vyjadriť pravdepodobnosť vygenerovania našej kolekcie dokumentov a následne sa snažíme túto pravdepodobnosť maximalizovať nájdením takých parametrov, ktoré to umožňujú. Toto sa deje v nasledovných krokoch:

- 1)  $\forall d_i$  z kolekcie dokumentov vyber parametre multinomického rozdelenia  $\theta_i$  z Dirichletovho rozdelenia  $Dir(\alpha)$ .
- 2)  $\forall pozicia(j)$  v dokumente  $d_i$  vyber tému  $z_{i,j}$  s parametrami  $\theta_i$ .
- 3)  $\forall pozicia(j)$  v dokumente  $d_i$  vyber slovo  $w_{i,j}$  z pravdepodobnosti  $p(w_{i,j}|z_{i,j}, \beta)$ , ktorá je vyjadrená multinomickým rozdelením slov pre témy z Dirichletovho rozdelenia  $Dir(\beta)$ .

Na inferenciu v Latentnej Dirichletovej alokácii sa využívajú metódy ako Gibbsovo vzorkovanie alebo rôzne Bayesovské prístupy.

## **3. Silovo-riadená vizualizácia prúdu textu**

V tejto kapitole sme popísali metódy vizualizácie silovo-riadených systémov a ich využitie špeciálne vo vizualizácii prúdu textu. Vysvetlili sme ich výhody a nevýhody a ich možné použitie v implementačnej časti tejto práce. Keďže silovo-riadené systémy sa bežne používajú v kreslení grafov a teda aj ich využitie vo vizualizácii je hojné, pokúsili sme sa navrhnúť využitie iných typov silovo riadených systémov, ktoré môžeme neskôr využiť. Jedným z takýchto systémov je simulácia gravitácie. Keďže ide o súčasť našej každodennej skúsenosti môžeme predpokladať, že používateľ bude s týmto fenoménom stotožnený. Dôležitým aspektom vizualizácie je jej čitateľnosť pre používateľa a tú môžeme skúsiť maximalizovať správne zvolenou technikou.

Výstupom zo sémantickej analýzy prúdu textu je tematické zaradenie prichádzajúcich správ v prúde. Takého dáta sa dajú ľahko namapovať ako vrcholy grafu, prípadne ako prvky silovo-riadenej simulácie.

Podkapitola popisujúca algoritmy silovo-riadeného kreslenia grafov vychádza z vynikajúcej publikácie na túto tému pod názvom Handbook of Graph Drawing and Visualization [30] a jej podkapitoly pod názvom Force-directed drawing algorithms.

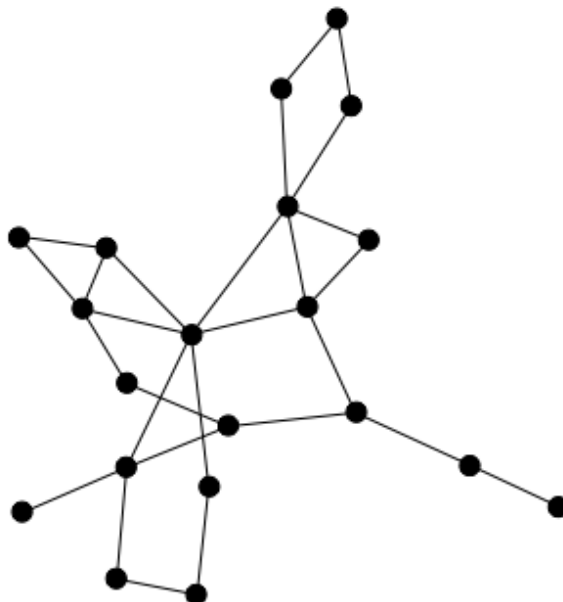
### **3.1 Algoritmy silovo-riadeného kreslenia grafov a ich využitie vo vizualizácii**

Ak sa vstupné dáta do vizualizácie dajú pohodlne interpretovať ako sieť, resp. graf, teda vieme vyjadriť súvislosti medzi atribútmi vizualizovaných prvkov ako hrany grafu a jednotlivé prvky ako jeho vrcholy, tak je často vhodné takto dáta interpretovať a následne použiť na ich vizualizáciu metódy silovo-riadeného vykresľovania grafov. Toto má výhodu v silnej základni v teórii grafov ako aj stovky až tisíce plnohodnotných systémov už využívajúcich poznatky z tejto oblasti. Ďalšími výhodami využitia týchto metód, keďže dáta sú vyjadrené vo forme siete, je ich intuitívne chápanie používateľom. Pre človeka je

intuitívnym aj použitie stromov vo vizualizácii. Stromy sú špeciálnym prípadom grafov a ich použitie je vhodné najmä na vizualizáciu hierarchie v dátach.

Nevýhodou môže byť výpočtová náročnosť, ktorú v súčasnosti vieme prekonať paralelizovaním výpočtov, resp. prevádzaním výpočtov na GPU. Nevhodným využitím silovo-riadených systémov je ak sa snažíme vizualizovať príliš mnoho prvkov, čo môže mať za následok zhoršenú čitateľnosť vizualizácie. Takáto technika je obzvlášť nevhodná pri použití mobilných zariadení, ktoré často nedisponujú takým výkonom ako bežné stolové počítače. Brať do úvahy musíme aj veľkosť displeja, ktorá je s možnosťami stolových počítačov neporovnateľná. Tomuto môžeme čiastočne predísť použitím multiplatformových technológií a odbremením používateľského systému od náročných výpočtov.

Silovo-riadené algoritmy sa dajú interpretovať ako virtuálne fyzikálne systémy, v ktorých telesá systému sú vrcholmi grafu. Tieto telesá medzi sebou interagujú fyzikálnymi silami ako je gravitácia, magnetická príťažlivosť alebo odpudivosť a pod. Príklad vizualizácie takéhoto systému vidíme na obrázku nižšie:



Obrázok 15, Silovo-riadená vizualizácia grafu, Zdroj: [31]

Jednotlivé vrcholy grafu na obrázku predstavujú dáta, ktoré chceme vizualizovať. Podobnosť medzi dátami je simulovaná vzdialenosťou medzi vrcholmi. Systém sa ustáli vďaka použitiu tzv. pružín, ktoré simulujú magnetickú odpudivosť medzi vrcholmi grafu.



„Vo všeobecnosti môžeme povedať, že silovo-riadené metódy definujú funkciu, ktorá mapuje každému rozloženiu grafu hodnotu z  $R^+$  reprezentujúcu energiu rozloženia grafu [32].“ Táto funkcia je definovaná takým spôsobom, aby nízke energie korešpondovali s takým rozložením, v ktorom susedné vrcholy reprezentovali dáta, ktoré majú spoločné vlastnosti a dáta, ktoré majú rozdielne vlastnosti, aby boli distribuované vizuálne od seba ďalej. Následne sa snažíme minimalizovať hodnotu tejto funkcie. Problémom však býva, že fyzikálne modely majú často viac lokálnych miním, hlavne ak sa snažíme zobrazit' mnoho uzlov, rádovo stovky. Tomuto sa snažíme predchádzať špecializovanými na to určenými algoritmami.

V nasledujúcom texte práce sme prebrali často používané algoritmy v oblasti silovo-riadeného vykresľovania grafov.

### **3.1.1 Pružinový systém**

Je to fyzikálny model simulujúci vrcholy grafu ako atomické častice, ktoré sa priťahujú alebo odpudzujú jedna od druhej a hrany grafu ako struny simulujúce tieto sily, splňujú niekoľko základných kritérií: „vrcholy musia byť rozmiestnené rovnomerne, všetky dĺžky hrán by mali byť rovnaké a rozloženie by malo byť tak symetrické ako sa len dá [32].“

Algoritmus strunového systému je pomerne jednoduchý a pracuje v nasledovných krokoch:

- 1) Náhodne rozmiestni vrcholy grafu, resp. vizualizované dáta
- 2) Následne opakuj vo vopred určený počet iterácií:
  - a. Vypočítaj silu pre každý vrchol zo strún, ktoré naň pôsobia
  - b. Posuň vrchol vzhľadom na vypočítanú silu
- 3) Vykresli graf

Ak do algoritmu zahrnieme aj pojem teploty a sily medzi vrcholmi grafu budeme počítat' vzhľadom na ňu, tak využijeme tzv. *simulované žihanie*.

### **3.1.2 Prístup pomocou teoretických vzdialeností**

Je to metóda, ktorej autori sa zamerali na vykreslenie esteticky prívetivého grafu ako na požiadavku, pri ktorej vzdialenosť medzi dvojicou vykreslených vrcholov v grafe je totožná s teoretickou vzdialenosťou medzi touto dvojicou vrcholov vypočítanou

algoritmom v korešpondujúcom grafe. Rozdiel oproti klasickému pružinovému systému popísanému v predošlom texte je taký, že v tomto modeli nepočítame sily medzi dvojicami vrcholov zakaždým, ale pracujeme s dvoma prípadmi:

- 1) Ak je dvojica vrcholov geometricky bližšie ako ich vzdialenosť v korešpondujúcom grafe, tak sa vrcholy budú odpudzovať.
- 2) Ak je dvojica vrcholov geometricky ďalej ako ich vzdialenosť v korešpondujúcom grafe, tak sa vrcholy budú priťahovať.

Teda princíp spočíva v minimalizovaní rozdielu medzi geometrickou vzdialenosťou a vzdialenosťou v korešpondujúcom grafe, čo dosahujeme minimalizovaním energie pružinového systému. Tento algoritmus je výpočtovo pomerne náročný, no vedie k esteticky prívetivým výsledkom.

### **3.1.3 Techniky kreslenia dynamických grafov**

Využívajú sa vo vizualizáciách, kde neustále pribúdajú dáta, ktoré chceme vizualizovať. Tak je to napríklad vo vizualizácii prúdu textu. Dynamické grafy riešia problém ako efektívne vykresliť vzťahy medzi týmito dátami. Máme na výber viacero spôsobov ako dosiahnuť požadovaného efektu, no skoro všetky prístupy používajú rôzne modifikácie silovo-riadených systémov. Podľa [32] vykresľovanie pomocou dynamických grafov musí spĺňať dve nasledovné kritériá:

- 1) „Čitateľnosť individuálnych rozložení grafu, ktoré závisí od estetických kritérií ako zobrazenie symetrií, uniformná dĺžka hrán a minimálny počet prekrížení hrán.“
- 2) „Zachovanie mentálnej mapy v sériách rozložení grafu, čo môžeme dosiahnuť tým, že sa ubezpečíme, vrcholy a hrany grafu, ktoré zobrazujeme v po sebe vykreslených grafoch ostávajú na rovnakých miestach.“

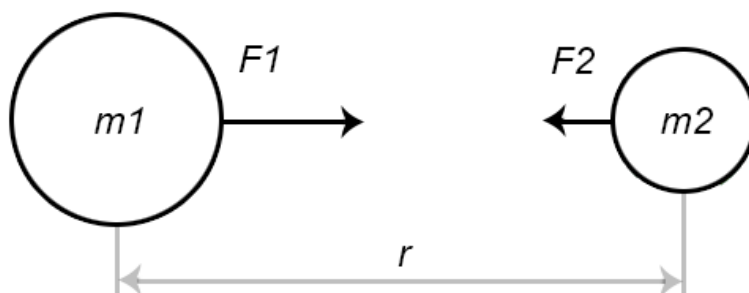
Tieto dve kritériá býva často ťažké zachovať súčasne. Techniky vykresľovania takýchto grafov spočívajú vo vykreslení grafov vedľa seba, nad sebou, naraz s prechodom alebo pomocou animácií v čase. Pri animáciách sa používajú techniky spočívajúce v postupnom miznutí vrcholov, alebo opaku.

Algoritmy silovo-riadeného vykresľovania grafov majú silný teoretický základ a ich časté využívanie vo vizualizácii vedie k neustálemu pokroku v tejto oblasti. Medzi

mechanizmy, ktoré sa využívajú v tejto oblasti patria aj genetické algoritmy a už spomínané simulované žíhanie. Medzi silné stránky týchto techník patrí flexibilita, intuitívnosť a kvalitné výsledky vo vizualizácii. V nasledujúcej podkapitole sme sa sústredili na techniky, ktoré však vo vizualizácii nie sú bežné a popísali sme si bližšie možnosti ich využitia v našej práci. To, že sme sa vybrali netradičnou cestou vo vizualizácii možno podporí neskorší výskum v tejto oblasti a to následne otvorí novú cestu k ešte lepšej vizualizácii informácií.

### 3.2 Simulácia gravitácie

Je pravdou, že vo vizualizácii dát, ktoré vieme pohodlne namapovať na graf sa často stretávame s použitím silovo-riadených systémov, ktoré sme popísali v predošlej podkapitole. Hlavným dôvodom je silný teoretický základ v oblasti teórie grafov, ďalším dôvodom je ich reálne využitie v mnohých aplikáciách vizualizácie. No taktiež je pravdou, že simulácia dynamických grafov je výpočtovo veľmi náročná a ak sa snažíme o spôsob vizualizácie, ktorý nie je tak náročný na výpočet a zároveň je zobraziteľný na väčšine súčasných zariadení, dá sa ľahko škálovať a modifikovať i samotným používateľom, je rozumné siahnuť po alternatívnych riešeniach. Takýmto riešením môže byť napríklad využitie simulácie gravitácie, ktoré je intuitívne pre každého používateľa a zároveň jednoduché na simuláciu. O výhodách a úskaliach tohto prístupu si povieme v nasledujúcom texte.



Obrázok 16, Gravitačná sila pôsobiaca medzi dvoma telesami rôznych veľkostí.

**Gravitácia** resp. gravitačná sila je jednou zo štyroch základných síl interakcie v prírode a jej efekty sú našou každodennou skúsenosťou. Je to sila, ktorá pôsobí medzi

ľubovoľnými dvoma hmotnými telesami, ktorá je priamo úmerná k súčinu ich hmotností a inverzne úmerná k štvorcu ich vzájomnej vzdialenosti.

$$F = \kappa \frac{m_1 m_2}{r^2}$$

$m_1$  a  $m_2$  sú hmotnosťami telies,  $r$  je vzdialenosť medzi nimi a  $\kappa$  je tzv. gravitačná konštanta, ktorej hodnota je  $\kappa = 6.672 \cdot 10^{-11} kg^{-1}m^3s^{-2}$ . Tento vzťah platí pre sférické telesá s rovnomerne rozloženou hmotnosťou, čo je pre podmienky použitia simulácie gravitácie vo vizualizácii plne dostačujúce. Tu si je potrebné uvedomiť, že v procese vizualizácie chceme využiť naše intuitívne chápanie sveta a nie vizualizovať pôsobenie gravitácie samotnej. Práve z tohto dôvodu si vystačíme s takouto definíciou gravitácie. Simulácia gravitácie sa najčastejšie používa vo vedeckej simulácii, konkrétne vo fyzike a astrofyzike. Najčastejšie ide o simuláciu vzájomného pôsobenia vesmírnych telies, formovanie klastrov, hmlovín, galaxií a pod., teda v simulácii nebeskej mechaniky. V našej práci vychádzame zo základov tohto prístupu, no pre používateľsky prívetivejšiu vizualizáciu zahrnieme do vizualizácie aj detekciu kolízií, ktorá sa v astrofyzikálnych simuláciách často neberie do úvahy, keďže zvyčajne stačí takáto aproximácia výpočtov pre kvalitné výsledky.

### 3.2.1 Problém $n$ telies

Pri vypočítavaní pohybu viacerých telies, ktoré na seba vzájomne pôsobia gravitačnou silou sa dostávame k tzv. problému  $n$  telies. Ide o problém predpovedania vzájomného pohybu týchto  $n$  telies, pričom pod  $n$  rozumieme zvyčajne  $n > 3$ . Inými slovami, snažíme sa nájsť počiatočné hodnoty diferenciálnych rovníc popisujúcich tento problém. Riešenie tohto problému je považované za dôležité k celkovému pochopeniu nebeskej mechaniky. Problém  $n$  telies je možné vyjadriť vzorcom, kde hľadáme riešenie pre systém druhého rádu:

$$m_i \ddot{\mathbf{r}}_i = \kappa \sum_{j=1; j \neq i}^N \left( \frac{m_i m_j (\mathbf{r}_i - \mathbf{r}_j)}{|\mathbf{r}_i - \mathbf{r}_j|^3} \right)$$

$\kappa$  je gravitačná konštanta,  $m_i$  a  $m_j$  sú váhy telies,  $\mathbf{r}$  a  $\mathbf{r}_j$  sú vektory pozícií telies a  $N$  je počet prvkov v systéme. Keďže opačné sily na seba pôsobiace sú si rovné, vieme znížiť počet operácií na:

$$\sum_{i=1}^N (N-i) = \frac{1}{2}(N-1)N \approx O(N^2)$$

Problém dvoch telies je vyriešený úplne, pre problém troch telies existujú nejaké čiastočné riešenia, no pre problém viacerých telies sa musíme zaoberať s aproximačnými metódami a využitím tzv. Taylorovho radu, keďže presné numerické riešenie nepoznáme. Obmedzením pre numerické riešenie je potreba zaokrúhľovania a samozrejme presnosť vstupných dát. Analytickým spôsobom je tento problém neriešiteľný, pretože nedokážeme analyticky popísať pohyb týchto telies v čase. Toto nás vedie k simuláciám problému  $n$  telies.

### 3.2.2 Simulácia $n$ telies

Keďže vizualizácia prvkov využitím gravitačnej sily si vyžaduje riešenie problému  $n$  telies, tak sa nevyhneme aproximačným metódam riešiacim tento problém resp. modelujúcim riešenie tohto problému. Existuje mnoho spôsobov ako ho riešiť a precíznosť riešenia sa odzrkadľuje v nárokoch na výpočtovú silu. Základný prístup je priamo počítať diferenciálne rovnice pohybu telies, no ide o výpočtovo náročný proces. Tuto sa používajú najčastejšie metódy ako Eulerova metóda, či metóda Runge-Kutta. Ďalším prístupom, ktorý sme si popísali v nasledujúcej podkapitole je simulácia metódou Barnes-Hut, pri ktorej zložitosť výpočtov klesá na  $O(n \log n)$ .

Najjednoduchší prístup je však v každej iterácii počítať gravitačnú silu pôsobiacu medzi telesami každého telesa s každým. Inými slovami, ide o výpočet hrubou silou. Tento prístup má však časovú zložitosť  $O(n^2)$ . Optimalizáciami môžeme toto riešenie zredukovať na polovicu operácií, čo pri malom počte vizualizovaných prvkov býva plne dostačujúce.

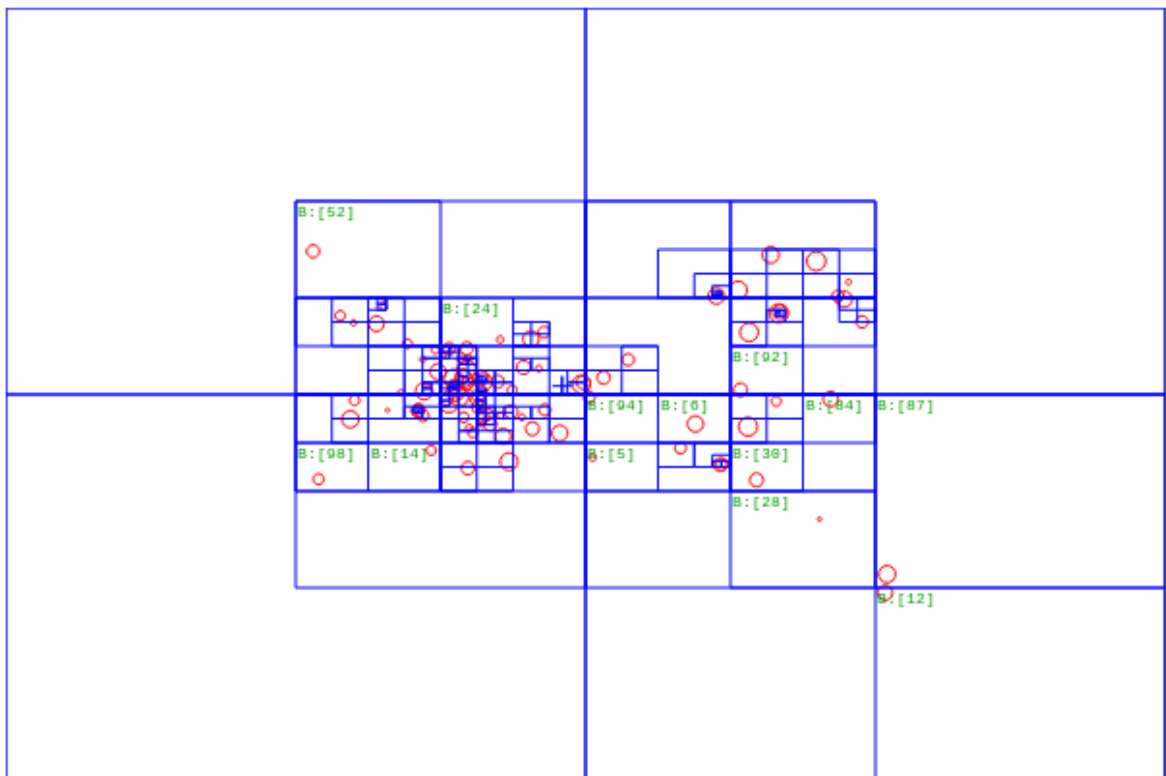
### 3.2.3 Simulácia Barnes-Hut

Simulácia Barnes-Hut vychádza z predpokladu, že klastre prvkov simulácie, ktoré sú dostatočne ďaleko môžeme zoskupovať. Tento predpoklad nazývame Multipole

Acceptance Criterion a vyjadrujeme ho pomerom medzi vzdialenosťou skupiny  $d$  a jej polomerom  $r$ :

$$\theta = \frac{d}{r}$$

Na základe tohto predpokladu môžeme priestor rekurzívne rozdeliť v 2D do kvadrantov pomocou tzv. quad-tree (v 3D priestore pomocou oct-tree). V takto rozdelenom priestore robíme individuálne výpočty len medzi prvkami, ktoré sa nachádzajú blízko seba a vzdialenejšie prvky interpretujeme ako jedno veľké teleso, ktorého ťažisko sa nachádza v strede kvadrantu.



Obrázok 17, Simulácia 100 prvkov pomocou Barnes-Hut v 2D, Zdroj: [33]

Algoritmus Barnes-Hut pracuje v nasledovných krokoch:

- Rekurzívne rozdeľ prvky do skupín v quad-tree tak, aby kvadranty obsahovali najviac po jednom prvku
- Pre výpočet síl pôsobiacich medzi prvkami traverzuj strom od jeho koreňa reprezentujúceho celú oblasť až po listy stromu:

- a. Ak je počet prvkov v uzle 1, ťažisko určí ako stred prvku a hmotnosť ako hmotnosť prvku
- b. Inak, rekurzívne vypočítaj hmotnosť kvadrantu na základe hmotností všetkých potomkov kvadrantu a ťažisko kvadrantu obdobným spôsobom, ako ováňovaný priemer ťažísk jeho potomkov ováňovaných ich hmotnosťami.
- c) Využijeme predpoklad Multipole Acceptance Criterion a ešte raz traverzujeme celý strom a vypočítame pre každý uzol  $\theta$ .
  - a. Ak  $\theta$  leží pod nami určeným prahom, gravitačnú silu môžeme aproximovať a prvky potomkov v strome nemusia mať gravitáciu počítanú zvlášť.
  - b. Inak, gravitačnú silu aproximovať nemôžeme a vnárame sa do potomkov v strome a počítame gravitáciu pre každého potomka zvlášť.

Ak pre každý vnútorný uzol vyjde hodnota  $\theta = 0$ , tak skonvergujeme ku klasickému riešeniu kedy počítame gravitáciu medzi všetkými prvkami. Tomuto môžeme predísť vhodným zvolením prahu pre  $\theta$ .

### 3.3 Zhrnutie

V tejto kapitole sme si ukázali silovo-riadené metódy, ktoré môžeme využiť v našej vizualizácii prúdu textu. Spomenuli sme si ich výhody a nevýhody a vzhľadom na naše kritériá sme sa rozhodli pre využitie simulácie pseudo-gravitácie v implementačnej časti tejto práce. Dôvody pre výber tohto spôsobu môžu byť pozornému čitateľovi jasné, no radšej ich zhrňme do niekoľkých bodov:

- 1) Použitie metaforý vesmírneho prostredia – používateľovi je potrebné zobrazit' vizualizáciu čo najintuitívnejším spôsobom. Môžeme predpokladať, že takmer každý potenciálny používateľ sa už stretol s týmto fenoménom.
- 2) Jednoduchosť – keďže sme si zvolili počet vizualizovaných tém  $T = 300$ , aj počet telies v simulácii bude nanajvýš 300. Tým pádom vieme povedať, že ak budeme výpočty v simulácii počítať hrubou silou, tak to bude najviac  $n^2 / 2$  výpočtov, čo je zvláduteľný výpočet aj na mobilných zariadeniach.
- 3) Animácia – môžeme predpokladať, že vizualizácia bude v sebe zahŕňať dynamické prvky ako príchod novej témy do vizualizácie, starnutie témy, či jej zánik. Tu sa

nám naskytujú možnosti využiť pohyb prvkov v simulácii a vytážiť zo zvolenej metafory vizualizácie čo najviac.

4) Hravý prístup a iné.



## 4. Implementácia a vyhodnotenie

V nasledovnom texte sa oboznámime s procesom implementácie aplikácie ChatVis, do ktorej sme implementovali našu metódu vizualizácie textovej komunikácie. Ako metaforu pre vizualizáciu sme si zvolili *vesmírne prostredie* a teda pri nej využijeme simuláciu gravitácie. Na extrakciu sémantiky textu sme si zvolili algoritmus LDA, pretože najviac vyhovoval našim podmienkam. Algoritmus je potrebné natrénovať len raz a ako prichádzajú nové dokumenty do kolekcie, tak sa nepretrénováva. Aplikácia je vytvorená použitím viacerých programovacích jazykov a moderných technológií a o nich si povieme v prvej podkapitole tohto textu. Aplikácia pozostáva z troch hlavných funkčných celkov:

- 1) Extrakcia sémantiky textu
- 2) Serverová časť fungujúca ako medzičlánok
- 3) Vizualizácia textu

**Extrakcia sémantiky textu** sa deje na strane servera a je dosiahnutá vynikajúcou implementáciou algoritmu LDA za použitia Gibbsovho vzorkovača [34] a jej modifikáciou pre potreby našej implementácie.

Na **serverovej časti** je spustená aplikácia chatu a funguje ako prostredný medzičlánok medzi spracovaním textu a samotnou vizualizáciou, čiže sprostredkuje vstupy a výstupy aplikácie.

**Vizualizácia textu** je výsledným produktom aplikácie a výpočty použité v nej sa dejú na strane klienta – konkrétne vo webovom prehliadači.. Vstupné dáta do nej sú odosielané serverom a tie už ďalej nie je potrebné spracovávať.

Každej časti aplikácie budeme venovať náležitú pozornosť. Počas procesu implementácie sme sa priebežne stretávali s mnohými problémami, ktorým bolo potrebné navrhnúť riešenie a implementovať ho. V závere tejto kapitoly aplikáciu otestujeme a vyhodnotíme výsledky testov. Navrhujeme aj jej ďalšie možnosti rozšírenia a použitia.

## **4.1 Implementácia aplikácie ChatVis**

### **4.1.1 Použité technológie**

Ako server je použitý voľne dostupný Apache 2.2.22 skompilovaný pre platformu Windows v 32bitovej verzii. Na ňom je spojazdnené rozšírenie pre jazyk PHP 5.4.3 a databáza MySQL 5.5.20. Pre platformu Windows je tento balíček dostupný v podobe aplikácie WampServer. Aplikácia bola programovaná v rôznych programovacích jazykoch ako C++, PHP a Javascript. Dáta aplikácie sú uchovávané databázou MySQL a sprostredkované pomocou dátového formátu JSON. Vizualizácia je vykresľovaná pomocou HTML5 do tzv. Canvasu.

Na vývoj aplikácie boli použité vývojové prostredia Netbeans IDE 7.3, Visual Studio 2010 a Matlab R2012a.

Počas implementácie vizualizácie sme použili vybranú funkcionality Javascriptovej knižnice JQuery 1.9.1. a úžasnej vizualizačnej knižnice D3 v3 od Michaela Bostocka.

### **4.1.2 Chat**

Na serveri je spustená aplikácia chatu. Je naprogramovaná v jazyku PHP. Na uľahčenie práce pri implementácii bol použitý objektovo navrhnutý framework Nette 2.0, ktorý disponuje mnohými vstavanými funkciami ako napríklad ladiacimi nástrojmi, prepracovanou podporou moderných technológií ako AJAX, DRY, KISS, Web 2.0. Používa návrhový vzor MVP resp. MVC a Dependency Injection. Podľa nezávislého testu patrí medzi najrýchlejšie vo svojej kategórii. Používa šablónovací systém Latte, ktorý prevádza zobraziteľnú časť aplikácie do HTML5 a CSS3.0. Na komunikáciu s databázou používame v jazyku PHP rozšírenie PDO.

Chat disponuje základnou funkcionality ako je:

- a) možnosť registrácie nového používateľa
- b) možnosť trvalého alebo dočasného prihlásenia
- c) možnosť komunikácie s ostatnými používateľmi verejne alebo súkromne
- d) možnosť úpravy svojho profilu
- e) možnosť odhlásenia

Z rozšírenej funkcionality je v chate naimplementovaná:

- a) galéria uložených vizualizácií

- b) možnosť vyvolať vizualizáciu na aktuálnych dátach
- c) k dispozícii je sada nástrojov na predspracovanie kolekcie dokumentov ako trénovacie dáta pre algoritmus LDA
- d) sada nástrojov na testovanie

**Sada nástrojov na spracovanie textu** obsahuje niekoľko jednoduchých, no účinných nástrojov na vygenerovanie korpusu potrebného na natrénovanie algoritmu LDA. Tieto nástroje sú v pozmenenej a automatizovanej forme neskôr používané na spracovanie textu z textovej komunikácie. Ako korpus na natrénovanie algoritmu LDA je použitá internetová encyklopédia Wikipedia v slovenskom jazyku dostupná v jednosúborovej komprimovanej podobe. Tento súbor je rozdelený do samostatných súborov podľa jednotlivých článkov encyklopédie a následne sú zo všetkých článkov odstránené tzv. wiki tagy, čo je obdoba XHTML značiek.

Nástroj na **vygenerovanie jednosúborového korpusu** prechádza všetky jednotlivé články a pomocou jednoduchého regulárneho výrazu ich extrahuje.

```
preg_match_all("/<doc[^>]*>(.*?)</doc>/is", $wikifile, $matches);
```

Zdrojový kód 1, Extrakcia článku zo súboru pomocou regulárneho výrazu

Po extrakcii článku sa text článku predspracuje. Odstránia sa z neho všetky zvyšné XHTML značky, odstráni sa z textu diakritika, všetky znaky abecedy sa prevedú na písmená malej abecedy. Vymažú sa všetky znaky, ktoré nie sú znakmi abecedy a nie sú medzerou, vymažú sa z dokumentu všetky tzv. stop-slová. Vymažú sa všetky nadbytočné medzery. Takto dostávame jednosúborový korpus, ktorý obsahuje v riadkoch jednotlivé dokumenty. Ide o bag-of-words model popísaný v predošlých kapitolách. Väčšina práce je vykonaná opäť pomocou regulárnych výrazov, ktoré sa ukázali na túto prácu vhodné.

```
public function processText($input)
{
    // Remove tags
    $input = strip_tags($input);
    // Converts accented to non-accented
    $input = iconv('UTF-8', 'ASCII//TRANSLIT//IGNORE', $input);
```

```

// String to lower
$input = strtolower($input);
// Remove all non-word and non-space chars
$input = preg_replace('/^[^\sa-z]+/S', '', $input);
// Remove all stopwords and single letters
$input = preg_replace('/\b(' . implode('|', $this->stopwords) .
'|[a-z])\b/', '', $input);
// Replace all (consecutive) whitespace characters with a single
space:
$input = preg_replace('/\s+/S', ' ', $input);
// Trim it
$input = trim($input);
return $input;
}

```

Zdrojový kód 2, Funkcia processText() predspracúvajúca text korpusu. Funkcia je používaná aj na predspracovanie príchodzieho textu z textovej komunikácie.

Nástroj na **generovanie zoznamu slov, ktoré majú nízku frekvenciu výskytu** prechádza celý korpus a postupne generuje zoznamy slov pre slová, ktoré sa nachádzajú v korpuse len jeden až päť krát, pre každú frekvenciu samostatne. Tieto slová sú potom z korpusu vymazané pomocou ďalšieho nástroja. Ide o významnú redukciu slovníka slov z korpusu a ide aj o výrazné zníženie celkovej veľkosti korpusu. Tento proces je však veľmi zdĺhavý.

Následne sa pri predspracovaní textu použije nástroj na **vymazanie krátkych článkov**. Pozorovaním sme zistili, že korpus obsahuje desiatky článkov bez pridanej informácie. Väčšinou išlo o články popisujúce počty obyvateľov alebo články, ktoré menovali rôzne oblasti v štátoch a pod. Takéto články sú z nášho pohľadu nepotrebné, pretože by došlo k menej kvalitnému natrénovaniu algoritmu LDA. Prah na určenie krátkeho článku je aspoň 20 medzier.

Ďalším krokom je použitie nástroja na **vymazanie krátkych slov** a na **vymazanie tzv. stop-slov**. Krátke slová sú vymazávané za účelom redukcie slovníka. Väčšinou sa jedná o skratky, spojky a iné neplnovýznamové slová jazyka. Prah na vymazanie krátkeho slova je daný počtom znakov 1 až 3 vrátane. Stop-slová sú určené dopredu slovníkom takýchto slov.

Po tomto procese máme vygenerovaný korpus, ktorý použijeme na natrénovanie algoritmu v ďalšej podkapitole. Z bežne dostupných zdrojov sme vybrali Wikipediú, pretože ako vstup na natrénovanie LDA je dôležité, aby každý dokument korpusu bol čo najucelenejší a teda sa týkal čo najmenej tém, čomu väčšina článkov na Wikipedii zodpovedá. V procese prípravy sme mali možnosť natrénovať algoritmus LDA aj na vstupných dátach zo skutočnej textovej komunikácie trvajúcej dva roky medzi zhruba dvoma stovkami používateľov, no výsledky dosahované na takomto korpuse boli neporovnateľné s výsledkami dosahovanými na Wikipedii. Výsledky natréovania boli posudzované empiricky.

#### 4.1.3 Proces LDA

Na extrakciu sémantiky textu z prúdu textu je použitá implementácia algoritmu LDA pomocou Gibbsovo vzorkovača GibbsLDA++ v aktuálnej verzii 0.2 [34]. Táto implementácia bola modifikovaná pre potreby tejto práce.

Ako vstup do algoritmu pre potreby tréovania algoritmu sme pripravili korpus jeho predspracovaním. Vstup je textovým súborom, ktorého časť môže vyzeráť nasledovne, pričom číslo v prvom riadku korpusu určuje počet dokumentov obsiahnutých v korpuse a za ním nasledujú v riadkoch jednotlivé dokumenty korpusu:

3

*text dokumentu jeden text dokumentu urceny trenovanie algoritmu*

*text dokumentu dva text dokumentu urceny trenovanie algoritmu*

*text dokumentu tri text dokumentu urceny trenovanie algoritmu*

Pre potreby správneho **natréovania algoritmu** bolo potrebné určiť počet tém, ktoré chceme aby nám algoritmus určil. Tento počet sme určili ako  $T = 300$  odvolávajúc sa na výskum v [35] popísaný v predošlej kapitole. Vstupné hyperparametre modelu LDA sme určili ako  $\alpha = 50/T$ , teda  $\alpha = 0.166667$  a  $\beta = 0.1$  [35]. Počet iterácií Gibbsovo vzorkovača sme empiricky odhadli na 2000 a následne sme vykonali výpočet modelu na dodatočných 1000 iteráciách. Pre potreby pomenovania tém sme si nechali z konečného modelu vypísať 50 reprezentatívnych slov pre každú z 300 určených tém. Výstupom algoritmu LDA je niekoľko súborov:

- 1) model-final.others – obsahuje vstupné parametre modelu pri tréovaní
- 2) model-final.phi – obsahuje pravdepodobnostné distribúcie medzi slovami a témami. Riadky sú témami a stĺpce slovami zo slovníka, teda na jednotlivých pozíciách obsahuje  $p(slovo_w|téma_t)$ .
- 3) model-final.theta – obsahuje pravdepodobnostné distribúcie medzi témami a dokumentmi. Riadky sú dokumentmi a stĺpce témami, teda na jednotlivých pozíciách obsahuje  $p(téma_t|dokument_d)$ .
- 4) model-final.tassign – obsahuje priradenia medzi témami a slovami z korpusu. Každý riadok súboru je dokument z korpusu a namiesto slov obsahuje index na pozíciu slova v slovníku a index témy v tvare  $index\_slova_{i,j}: téma_t$ .
- 5) model-final.twords – obsahuje vopred určený počet reprezentatívnych slov pre jednotlivé témy spolu z ich pravdepodobnosťami príslušnosti téme.
- 6) wordmap.txt – obsahuje abecedne zoradené slovník slov z korpusu s ich indexmi.

Po natrénovaní modelu LDA sme podľa reprezentatívnych slov pre každú z tém určili jej názov. Názvy sme určovali empiricky a následne sme roztriedili témy do ôsmich skupín reprezentujúcich hierarchicky vyššie tematické celky. Tieto sme následne tiež pomenovali.

**Úpravy aplikácie pre naše potreby** zahŕňajú prepísanie aplikácie tak, aby sme ju mohli použiť v procese inferencie, teda v procese určovaní tém na nových dátach prichádzajúcich z prúdu textu v reálnom čase. Zároveň obsahuje modifikácie na prácu s databázou MySQL.



Obrázok 18, Proces komunikácie s databázou.

Komunikácia s databázou prebieha v nasledovných krokoch (šípky označujú tok dát):

- 1) Proces LDA spraví požiadavku na databázu, či existuje v zásobníku správa, ktorá ešte nie je spracovaná. Ak taká správa neexistuje, tak sa pýta znova, ak taká správa existuje, algoritmus spraví inferenciu na novej správe z toku textu
- 2) Ak existujú spracované dáta, tak sa zapíšu na server a správa v zásobníku sa označí ako spracovaná.

- 3) Webová aplikácia chatu čaká na vstup od používateľa. Ak nejaký dostane, tak ho spracuje a zapisuje do databázy.
- 4) Aplikácia načítava texty z toku od ostatných používateľov a zobrazuje ich.

V procese komunikácie medzi chatom a LDA bol implementovaný zásobník FIFO (First-in first-out). Toto nám zabezpečuje, že žiadna správa nebude vynechaná v procese spracovania ani počas zahŕtenia servera správami počas špičky. Výstupom druhého kroku, teda procesu spracovania správy z toku textu je 300-rozmerný vektor pre  $i$ -tu správu v toku:

$$tdistribution_i = (a_1, a_2, \dots, a_{300}); a \in R; a_1 + a_2 + \dots + a_{300} = 1$$

Vektor sa používa vo vizualizácii samotnej sémantiky textu. Atribúty sú namapované na prvky vizualizácie, tento prístup si popíšeme v ďalšom texte. Výstup je uložený do databázy vo formáte JSON, čo zaručuje kompatibilitu s vizualizačnou časťou implementácie v Javascripte. Ďalej sa ukladá v tomto kroku do databázy údaj o téme, ktorá mala vo vektore  $tdistribution_i$  najväčšie zastúpenie, teda index pre  $\max(a_j); a_j \in tdistribution_i$ . Ako posledná časť výstupu sa do databázy ukladá matica pozostávajúca z dvojíc *slovo:pravdepodobnosť*, kde riadky matice reprezentujú témy a stĺpce matice reprezentujú spomínané dvojice, kde *slovo* je reprezentatívne slovo pre tému zo spracovanej správy a *pravdepodobnosť* je pravdepodobnosť s akou prislúcha slovo danej téme. Dvojice sú zoradené podľa pravdepodobnosti zostupne.

$$\begin{pmatrix} (w_{1,1}, p_{1,1}) & \dots & (w_{1,20}, p_{1,20}) \\ (w_{2,1}, p_{2,1}) & \dots & (w_{2,20}, p_{2,20}) \\ \vdots & \ddots & \vdots \\ (w_{300,1}, p_{300,1}) & \dots & (w_{300,20}, p_{300,20}) \end{pmatrix}$$

Maticu opäť ukladáme do databázy ako objekt formátu JSON. Používa sa vo vizualizácii ako dodatočná informácia o zmene zastúpenia jednotlivých tém v čase.

V procese implementácie procesu LDA sme sa rozhodli skúsiť viacero prístupov, ako rozšírenie pre jazyk PHP do serveru Apache [36], alebo využitie tzv. socketov. Sockety si nenašli opodstatnenie pretože proces je spustený na rovnakom serveri a pracuje s rovnakou

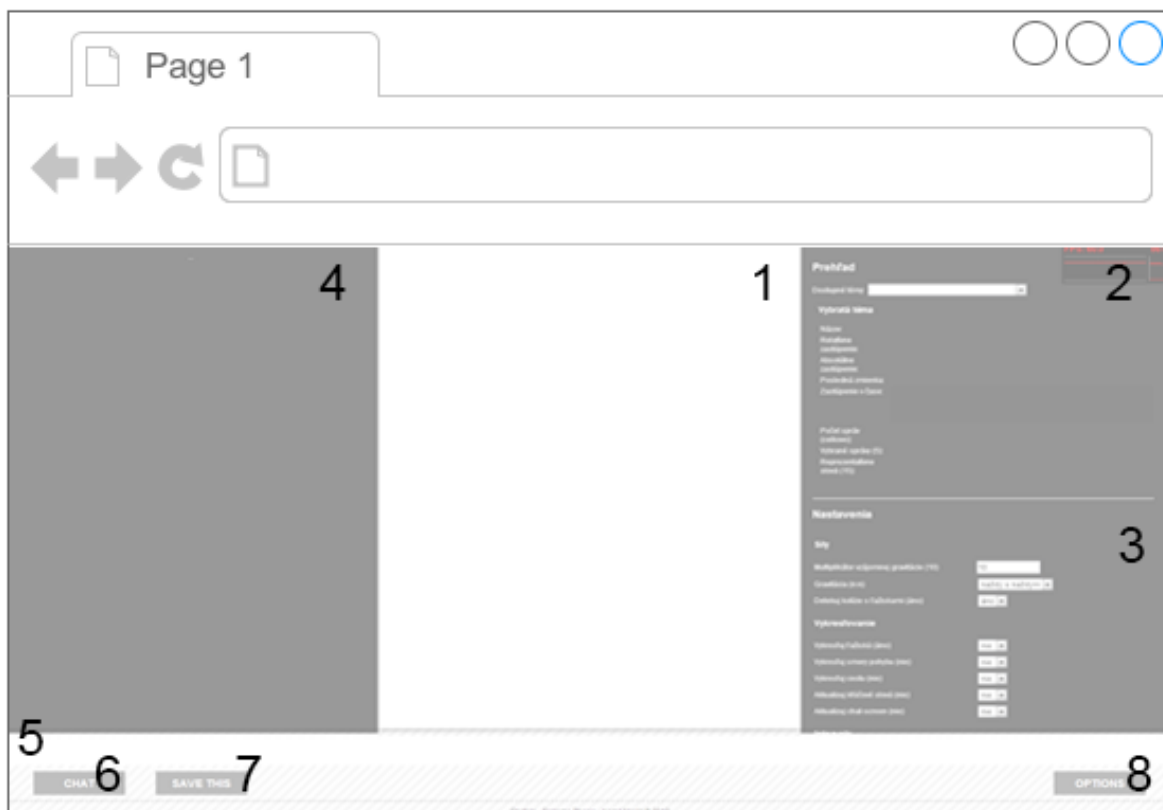
databázou, takže je zbytočné pri každej správe nadväzovať spojenie, keďže predpokladáme najhorší prípad a to neustále prichádzajúce správy v reálnom čase. Proces pracuje s tou istou databázou a pripojenie do databázy je možné spraviť aj dosadením internetovej adresy servera, nie len na lokálnom serveri. Použitie rozšírenia pre jazyk PHP by nám uľahčilo spoluprácu medzi serverom a procesom LDA, pretože by sa proces stal súčasťou jazyka a pristupovali by sme k nemu ako ku ľubovoľnej knižnici jazyka, no nevýhoda by spočívala v použití jedného procesorového vlákna, čo by spôsobilo značné spomalenie a záťaž na server.

#### **4.1.4 Vizualizácia**

Prúd textu ako výstup chatu, resp. textová komunikácia, je spracovaná procesom LDA a následne sú dáta opäť zapísané do databázy. V tomto momente sú všetky dáta potrebné na vizualizáciu dostupné v databáze vo formáte JSON. Tieto dáta potrebujeme namapovať do vizualizácie a následne zobrazit' pre používateľa.

Vizualizácia používa metaforu vesmírneho prostredia, konkrétne myšlienku telies obiehajúcich okolo seba priťahovaných gravitačnou silou. Táto metafora bola zvolená najmä pre svoju intuitívnosť. Vizualizácia bola naimplementovaná pomocou HTML5 technológie Canvas [37]. Ide o značku značkovacieho jazyka HTML5, ktorá umožňuje dynamické renderovanie 2D objektov a bitmapových obrázkov. Nemá zabudovaný graf scény. Do Canvasu vykresľujeme pomocou sady inštrukcií programovacieho jazyka Javascript. Jeho funkcionality je implementovaná vo väčšine moderných webových prehliadačov.





Obrázok 19, Návrh rozloženia grafického používateľského rozhrania pre vizualizáciu.

Na implementáciu grafického rozhrania bol použitý značkovací jazyk HTML5 a kaskádové štýly CSS. Na obrázku vidíme rozloženie jednotlivých prvkov grafického používateľského rozhrania:

- 1) Hlavná časť, kde prebieha samotná vizualizácia. Prvky 2, 3 a 4 majú nastavenú priehľadnosť a v základnom nastavení sa nezobrazujú, teda vizualizácia prebieha neustále na čo najväčšej ploche obrazovky.
- 2) Je panel zobrazujúci prehľad pre jednotlivé témy obsiahnuté vo vizualizácii. Pre každú tému je tu zobrazený jej:
  - a. Názov
  - b. Relatívne zastúpenie – teda zastúpenie v pomere k počtu správ, v ktorých bola téma spomenutá
  - c. Absolútne zastúpenie – zastúpenie v pomere ku počtu všetkých správ
  - d. Posledná zmienka – čas kedy bola téma spomenutá naposledy
  - e. Zastúpenie v čase – graf zastúpenia témy v čase v pomere k počtu všetkých správ
  - f. Počet správ – správy, v ktorých bola téma spomenutá

- g. Vybrané správy – päť posledných správ v textovej komunikácii, ktoré prispeli k zobrazeniu témy vo vizualizácii
  - h. Reprezentatívne slová – prvých 15 slov reprezentujúcich tému a ich pravdepodobnosti príslušnosti k danej téme a to priamo z korpusu a tak isto aj z prúdu textu.
- 3) Nastavenia vizualizácie, ktorými dostáva používateľ plnú kontrolu nad vizualizáciou. Disponuje možnosťami nastavovania síl pôsobiacich medzi prvkami vizualizácie, možnosťami nastavenia toho čo sa má vykresľovať, nastavenia intervalov spúšťania jednotlivých funkcií a pod.
  - 4) Panel zobrazujúci aktuálny prúd textu obsahujúci správy, meno používateľa, ktorý prispel správou a čas príspevku.
  - 5) Panel zobrazujúci histogram, ktorý reprezentuje farbou najspomínanejšiu tému v určenom časovom období a výškou jej príspevok v pomere ku všetkým správam v danom čase. Na jednotlivé zložky histogramu je možné kliknúť. Po kliknutí sa otvorí nové okno prehliadača, kde je zobrazená vizualizácia uložená v momente vytvorenia zložky histogramu. Tento prvok bližšie popíšeme v ďalšej podkapitole tejto práce.
  - 6) Tlačidlo CHAT zapínajúce a vypínajúce zobrazenie panelu 4.
  - 7) Tlačidlo SAVE THIS ukladajúce vizualizáciu v dátovej podobe do databázy pre prihláseného používateľa.
  - 8) Tlačidlo OPTIONS zapínajúce a vypínajúce panel 2 a 3.

Vizualizácia môže byť spustená používateľom z prostredia chatu. Po jej spustení prebieha inicializácia, kde sa načítajú **základné vstupné dáta** do vizualizácie. Tieto dáta sú uložené ako objekty formátu JSON:

- 1) gravity.json – pre každú dvojicu tém obsahuje kosínusovú mieru podobnosti, ktorá je použitá ako multiplikátor pri vypočítavaní gravitačnej sily pôsobiacej medzi prvkami vizualizácie.
- 2) groups.json – obsahuje názvy skupín tém a ich priradenie k farbám.
- 3) themes.json – obsahuje názvy jednotlivých tém, ich priradenie ku skupine a dvojice *slovo:pravdepodobnosť*, teda slová reprezentujúce danú tému a ich prislúchajúce pravdepodobnosti priradenia do témy.

Dáta, ktoré sú použité ako multiplikátory gravitácie počas vizualizácie sú predvypočítané pomocou kosínusovej miery v prostredí Matlab. Predvypočítali sme ich pre každú dvojicu tém a každá hodnota je ako reálne číslo v rozmedzí intervalu  $<0, 1>$ .

**Jednotlivé témy**, ktoré sme získali z prúdu textu vizualizujeme ako *planéty*. Teda naďalej používame metaforu vesmírneho prostredia. Každá téma je teda reprezentovaná vo vizualizácii ako kruh, na ktorý mapujeme jednotlivé atribúty prislúchajúce vizualizovanej téme. Jednotlivým kruhom prislúcha objekt *Ball*, ktorý je definovaný nasledovne:

```
function Ball(posX, posY, dirX, dirY, radius, group, theme, timestamp)
```

Zdrojový kód 3, Hlavička objektu Ball.

Atribúty *posX* a *posY* zodpovedajú aktuálnej pozícii kruhu v 2D priestore, *dirX* a *dirY* zodpovedajú aktuálnemu smeru pohybu kruhu, *radius* vyjadruje polomer kruhu a je naň namapovaná relatívna spomínanosť témy v prúde textu. Spomínanosť témy sa na kruh mapuje pomocou vzťahu:

$$r = p^2 \cdot (r_{max} - r_{min}) + r_{min}$$

Teda nový polomer  $r$  je daný štvorcom pravdepodobnosti  $p$ , ktorú chceme namapovať. Následne polomer preškálujeme podľa minimálneho a maximálneho možného polomeru vo vizualizácii. Použitím štvorca pravdepodobnosti, ktorú mapujeme na kruh sme zaručili správnu vizuálnu interpretáciu kruhu používateľom. Ľudský vizuálny systém interpretuje kruh ako plochu a teda jednoduché mapovanie pravdepodobnosti na polomer by nebolo dostačujúce. Atribút *group* vyjadruje príslušnosť k skupine tém a reprezentuje ho objekt *Group*, *theme* vyjadruje príslušnosť k téme a *timestamp* je čas v milisekundách kedy bola téma naposledy spomínaná v prúde textu. Objekt *Group* obsahuje identifikátor skupiny tém a farebnú reprezentáciu skupiny tém.

**Simulácia gravitácie** je implementáciou Gravitačného zákona spomenutého v druhej kapitole tejto práce. Gravitácia sa vypočítava medzi všetkými vizualizovanými témami a taktiež medzi témami a tzv. ťažiskami, ktoré reprezentujú gravitačné centrum jednotlivých skupín tém. Používateľ má možnosť meniť sekundárny multiplikátor

gravitácie a taktiež má možnosť prepnúť vypočítavanie gravitácie *každý s každým* a v *rámci skupín*. Je potrebné spomenúť, že v skutočnosti ide len o pseudo-gravitáciu a problém simulácie  $n$  telies riešime prístupom hrubou silou.

```
var dist = dx * dx + dy * dy;
var radius = balls[i].radius + balls[j].radius;
dist = dist * 2000 * (1 / (radius * radius)) * state_gravity[i][j] *
setting_gravity_multiplier;
```

Zdrojový kód 4, Časť funkcie gravity(i, j) vypočítavajúca gravitáciu pôsobiacu medzi dvoma témami vo vizualizácii.

Premenná *dist* je použitá na vypočítanie smeru kruhu vo vizualizácii. *state\_gravity* obsahuje predvypočítanú kosínusovú mieru podobnosti tém použitú ako multiplikátor gravitácie používajúc predpoklad, že podobnejšie témy sa k sebe budú gravitačne viac priťahovať ako témy menej podobné. *setting\_gravity\_multiplier* je možné použiť na dodatočné korekcie gravitačnej sily.

V procese návrhu a implementácie tejto aplikácie sme použili aj algoritmus Barnes-Hut spomínaný v predošlej kapitole. Simulácia takýmto spôsobom bola ale náročná na výpočet. Napriek tomu, že zložitosť algoritmu  $O(n \log n)$  je nižšia ako v našom prípade  $O(n^2)$ , reálny výpočet v prostredí prehliadača bol náročnejší, čo spôsobovalo nežiaduci vizuálny efekt tzv. trhania obrazovky a pod.

**Detekcia kolízií** sa vypočítava jednoduchým určením vzdialenosti medzi stredmi kruhov pomocou Pytagorovej vety a porovnaním tejto vzdialenosti so súčtom polomerov týchto kruhov. Kolízie detegujeme medzi všetkými dvojicami kruhov a zároveň aj s ťažiskami reprezentujúcimi skupiny tém. Avšak túto detekciu je možné vypnúť no vzhľadom na nasledovnú nestabilitu systému to používateľovi neodporúčame, resp. odporúčame používať berúc do úvahy túto skutočnosť. Ak nastane kolízia kruhov, tak nasleduje simulácia ich odrazu. Odraz dvoch kruhov simulujeme nasledovným vzťahom:

$$k = \frac{(l - d)}{l} * 0.5$$

Tento vzťah reprezentuje tzv. hyperbolický paraboloid, resp. sedlovú plochu.  $k$  určuje mieru zmeny smeru pohybu kruhov po kolízii na simuláciu ich odrazu.  $l$  určuje vzdialenosť kruhov a  $d$  určuje súčet polomerov kruhov. Simuláciu odrazu pomocou sedlovej plochy sme určili empiricky. Pomocou takejto simulácie však dosahujeme verné výsledky neodporujúce intuícii používateľa.

Tzv. **gravitačné ťažiská** plnia funkciu stabilizátorov vizualizácie a teda napomáhajú vizuálnej reprezentácii skupín tém. Ich hlavnou úlohou je vizuálne oddeliť jednotlivé skupiny tém pokiaľ sa gravitačne témy neovplyvnia iným spôsobom. Aby sa skupiny tém dobre opticky rozdelili je potrebné vhodne vypočítať polohu jednotlivých ťažísk. Témy sú rozdelené do ôsmich skupín, teda ani počet ťažísk nepresiahne túto hodnotu. Ťažiská do vizualizácie pribúdajú podľa toho koľko skupín je vo vizualizácii reprezentovaných. Vypočítať polohy pre triviálne prípady pre počet ťažísk 1 a 2 je jednoduché, no pre počet ťažísk rovný trom a viac je výpočtovo náročnejšie. Počas implementácie tohto kroku sa môžeme uspokojiť s rozložením ťažísk napríklad použitím Haltonovej sekvencie často používanej pre generovanie bodov v priestore pre simulácie Monte Carlo a pod.

```
function haltonSequence(index, base)
{
    var result = 0;
    var f = 1 / base;
    var i = index;
    while(i > 0)
    {
        result = result + f * (i % base);
        i = Math.floor(i / base);
        f = f / base;
    }
    return result;
}
```

Zdrojový kód 5, Výpočet Haltonovej sekvencie.

Použitie tejto metódy, ale nedávalo vizuálne dobré výsledky a ťažiská neboli rozložené rovnomerne po viditeľnej ploche simulácie. Použitie tejto metódy sme teda zavrhl

a namiesto nej sme sa rozhodli použiť implementáciu Lloydovho algoritmu pre výpočet Centroidálnej voronoiovej teselácie (CVT) pomocou vizualizačného frameworku D3. CVT je taký typ Voronoiovej teselácie, v ktorej bod každej bunky tejto teselácie je zároveň jej ťažiskom.

```
function computeCvd(v)
{
    var env = [[0, 0], [0, yres], [xres, yres], [xres, 0]];
    var envelope = d3.geom.polygon(env);
    var vertices = [];
    var i, j;
    var voronoi, polygon, center_of_mass;
    vertices = d3.range(v).map(function(d, i)
    {
        return [Math.random() * xres, Math.random() * yres];
    });
    for(j = 0; j < 100; j++)
    {
        voronoi = d3.geom.voronoi(vertices);
        for(i = 0; i < voronoi.length; i++)
        {
            polygon = voronoi[i];
            center_of_mass=d3.geom.polygon(envelope.clip(polygon))
                                .centroid();
            if(!isNaN(center_of_mass[0]))
            {
                vertices[i]=[center_of_mass[0],center_of_mass[1]];
            }
        }
    }
    return vertices;
}
```

Zdrojový kód 6, Výpočet Centroidálnej voronoiovej teselácie pomocou Lloydovho algoritmu.

Funkcia dostáva na vstup počet ťažísk, ktorých pozície chceme vygenerovať a pracuje nasledovným spôsobom:

- 1) Vygeneruj náhodne body v intervaloch  $(0, \text{šírka\_obrazovky})$  a  $(0, \text{výška\_obrazovky})$ .
- 2) Následne pre  $\text{počet\_iterácií} = 100$  sprav:
  - a. Z daných bodov vygeneruj Voronoiovu teseláciu
  - b. Pre každú bunku teselácie zisti koordináty pre jej ťažiská a pôvodné body nahraď ťažiskami
- 3) Vráť vygenerované koordináty ťažísk

Vizualizácia **komunikuje so serverom** pomocou volaní technológie AJAX. Vykonáva niekoľko druhov volaní:

- 1) Volané len raz, počas inicializačnej fázy vizualizácie: *inicializácia skupín, inicializácia tém, inicializácia gravitačných multiplikátorov.*
- 2) Volané v pravidelných intervaloch: *získanie novej správy zo serveru.*
- 3) Vyvolané interakciou používateľa: *získanie najnovších reprezentatívnych správ pre vybranú tému, uloženie vizualizácie, uloženie nastavení, vyvolanie uložených nastavení.*

V tejto súvislosti je dôležité uvedomiť si, že pre použitie týchto volaní je nutné, aby aplikácia bola spustená na webovom serveri.

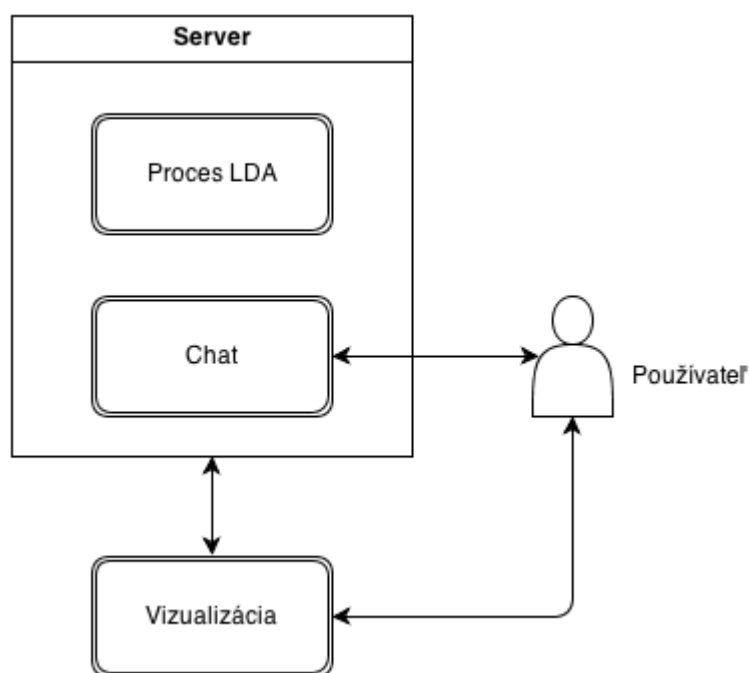
Do procesu vizualizácie sme taktiež naimplementovali tzv. **constrainty**, čo je vlastne sada pravidiel resp. obmedzení aplikovaných v pravidelných intervaloch na vizualizáciu. Týmto pravidlami sme zabezpečili plnohodnotnejší proces vizualizácie. Aplikujeme ich na vizualizované témy a na ťažiská tém. Do týchto pravidiel spadá:

- a) Vymazanie témy, ktorá je vzdialená od zobraziteľnej časti vizualizácie za určeným prahom.
- b) Vymazanie starých tém. Ide o také témy, ktoré neboli spomínané v prúde textu nejaký čas určený prahom. Starnutie tém je vizualizované postupnou zmenou priehľadnosti témy.
- c) Vymazanie malých tém, teda tém, ktoré majú vo vizualizácii percentuálny príspevok pod určeným prahom.
- d) Vymazanie ťažísk, ktoré v skupine neobsahujú zobraziteľné témy.

Vizualizačnú časť tejto práce sme naimplementovali v programovacom jazyku Javascript a optimalizovali najmä pre beh aplikácie na javascriptovom engine V8, ktorý je súčasťou webového prehliadača Chrome od spoločnosti Google. Optimalizácie zahŕňajú monomorfizmus, inicializáciu premenných, využívanie tzv. skrytých tried (hidden classes) alebo písanie krátkych funkcií a pod.

## 4.2 Pohľad na beh aplikácie

V nasledovnej podkapitole sa budeme venovať samotnému behu aplikácie a tomu ako pracuje z pohľadu používateľa. Beh aplikácie sa skladá z niekoľkých hlavných krokov, ktoré sú znázornené na obrázku nižšie.



Obrázok 20, Beh aplikácie ChatVis z pohľadu používateľa

Používateľ aplikácie interaguje s časťou aplikácie *Chat*, no neinteraguje priamo so serverovou časťou. Tak isto dokáže priamo interagovať s vizualizáciou. Tieto dva druhy interakcie (*používateľ-chat*, *používateľ-vizualizácia*) sú navzájom oddelené a od používateľa sa nevyžaduje, aby bol zapojený do oboch. V ďalšom texte sa sústredíme najmä na interakciu *používateľ-vizualizácia*.



#### 4.2.1 Vizualizácia prúdu textu

Počiatočná fáza vizualizačného procesu sa nazýva inicializačná fáza. Obrazovka vizualizácie je vo svojom východiskovom stave a žiadne dáta ešte nie sú vizualizované. Dáta sa začnú vizualizovať v stanovenom intervale, ktorý je možné meniť. Počiatočné nastavenie je jedna sekunda. Táto fáza môže javiť ako mierne chaotická, pretože témy ešte len pribúdajú do vizualizácie a tá sa preto ešte nemala čas ustáliť. Ustálenie prichádza obyčajne po niekoľkých sekundách až minútach. Urýchliť tento proces je možné skrátením intervalu medzi prichádzajúcimi správami do vizualizácie, alebo dočasným zvýšením gravitačného multiplikátora. Obyčajne nie je nutné použiť ani jednu z uvedených techník. Výhodou použitia metódy LDA v procese spracovania prúdu textu je, že výstup môžeme ihneď vizualizovať a teda inicializačná fáza nie je závislá na akumulácii dát. Z tohto dôvodu vizualizácia pre používateľa dáva zmysel už od inicializačnej fázy.

Dáta do vizualizácie pribúdajú v reálnom čase. Použitie metódy LDA rieši aj problém zahltenia vizualizácie, ktorý vyplýva z neustáleho nárastu množstva dát. Inferenciou vieme usúdiť o novej správe z toku akú tému reprezentuje. Vopred sme si určili maximálny počet tém  $T = 300$  a teda vieme, že maximálny počet vizualizovaných prvkov bude rovnaký. Týmto a tiež určením maximálneho a minimálneho polomeru kruhu reprezentujúceho tému vieme zaručiť, že vizualizácia nebude zahltená.

Nové dáta sa vždy vizualizujú v kontexte starých dát (ak také existujú). Na kruhy reprezentujúce témy je namapovaný aj atribút starnutia. Ide o postupné zvyšovanie priehľadnosti kruhu. Prah kedy nazývame tému starou je možné interaktívne meniť. Používateľovi prispieva k lepšiemu vnímaniu kontextu aj animované zväčšovanie alebo zmenšovanie kruhu ako zobrazenie zvýšenia alebo zníženia príspevku témy do vizualizácie. K zobrazeniu podobnosti tém napomáha gravitácia medzi kruhmi. Témy začínajú postupne tvoriť spolusúvisiace skupiny, najmä okolo ťažísk, ale aj medzi nimi. Takýto prístup spĺňa predpoklad, že „pre čo najpresnejšie identifikovanie a inteligentné popísanie zmeny v priestore informácií je potrebný kontext, v ktorom sú namapované nové informácie so starými [1].“

Používateľ má k dispozícii dva pohľady na vizualizáciu. Hlavný, kde sú dáta vizualizované tak ako prichádzajú, teda v nepretržitom toku alebo vedľajší, kde je vizualizácia v stanovenom časovom intervale uložená v bitmapovej podobe a je dostupná k uloženiu do galérie používateľa. Tieto dva pohľady umožňujú simultánne porovnávanie vizualizovaných dát v reálnom čase aj s odstupom času v rôznych intervaloch.

Keď sa po inicializačnej fáze vizualizácia stabilizuje, obvyčajne počas svojho behu stabilizovaná aj zostáva. Extrémnym prípadom je stav, kedy dlhú dobu neprichádzajú z prúdu textu žiadne nové dáta. Ak dáta začnú znova prichádzať, tak sa vlastne opakuje inicializačná fáza vizualizácie.

#### 4.2.2 Interakcia

Aplikácia ChatVis dáva používateľovi možnosť interakcie s vizualizáciou. Umožňuje mu to plnohodnotne analyzovať dáta z vizualizácie a manipulovať s ňou. Používateľ má k dispozícii niekoľko prvkov interakcie. Medzi ne patrí:

- a) Kliknutie do vizualizácie a držanie stlačeného ľavého tlačidla myši – funguje ako dočasné zastavenie simulácie. Nemá vplyv na vyžadovanie si nových dát vizualizáciou.
- b) Kliknutie pravým tlačidlom myši na kruh reprezentujúci tému – načíta dáta o téme do panelu s prehľadom.
- c) Kliknutie ľavým tlačidlom myši na gravitačné ťažisko a držanie tlačidla stlačeného – používa sa na presúvanie ťažiska. Pomáha pri doladovaní lepšej distribúcie tém v okne vizualizácie. Taktiež je túto techniku vhodné použiť pre zosilnenie alebo oslabenie pôsobenia gravitácie medzi témami a teda na lepšiu analýzu podobnosti tém.

#### 4.2.3 Nastavenia

Nastavenia vizualizácie by sme mohli zaradiť do kategórie interakcie s vizualizáciou. Ide o nástroj dávajúcich používateľovi takmer úplnú kontrolu nad vizualizáciou. Možností v nastaveniach je niekoľko a všetky je možné nájsť v paneli nastavení:

- a) Multiplikátor vzájomnej gravitácie – používa sa pri výpočte gravitácie pôsobiacej medzi telesami v simulácii. Nižšie číslo znamená silnejšiu gravitáciu.
- b) Gravitácia – možnosť nastaviť pôsobenie gravitácie *každý s každým*, alebo *v rámci skupín*.
- c) Detekcia kolízií s ťažiskami – toto nastavenie je odporúčané používať len ak používateľ potrebuje výrazne narušiť stabilitu vizualizácie.
- d) Vykresľovanie ťažísk
- e) Vykresľovanie smerov pohybu – užitočné pri odhadovaní smeru pohybu a rýchlosti kruhov v simulácii.

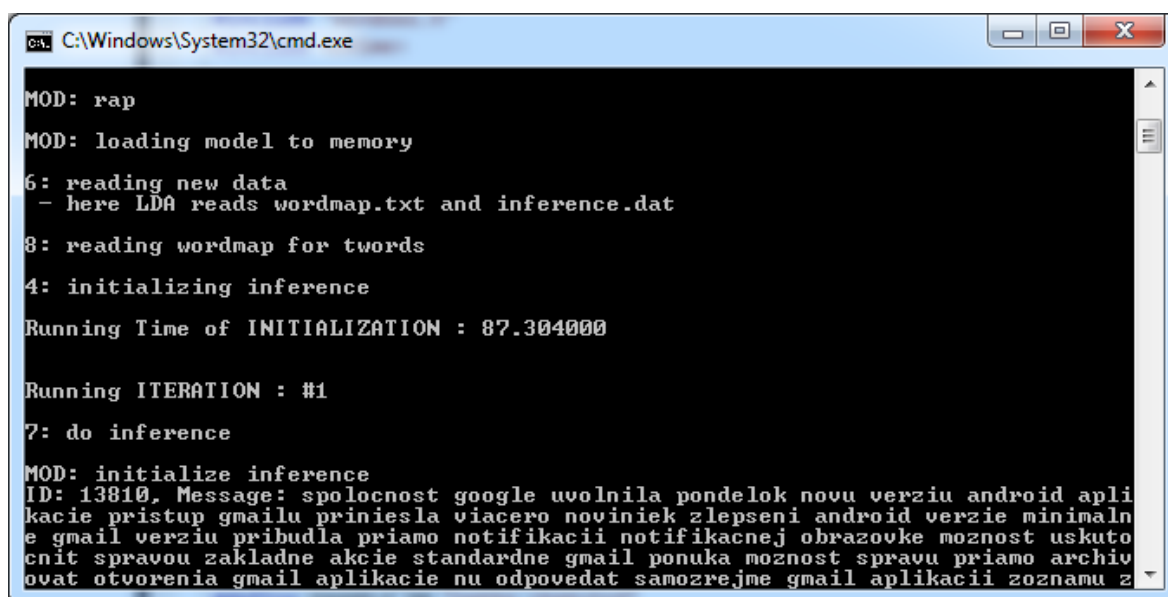
- f) Aktualizácia kľúčových slov – priebežne načítava a aktualizuje kľúčové slová reprezentujúce témy aktuálne. Používa sa na porovnanie zmien v reprezentácii danej témy.
- g) Aktualizácia chat obrazovky – aktualizuje panel so správami z prúdu textu.
- h) Tick – interval pre aktualizácie kroku v simulácii.
- i) Nová správa – interval pre vyžiadanie novej správy z prúdu.
- j) Aplikovanie constraintov
- k) Aplikovanie constraintov na ťažiská
- l) Obnova info o téme v options – interval, v ktorom sa bude obnovovať prehľad pre aktuálne zvolenú tému.
- m) Automatické uloženie canvasu – interval pre ukladanie okna vizualizácie a zobrazovanie údajov o tom v histograme.
- n) Limitná pravdepodobnosť – prah, pod ktorým sa téme považuje za nedôležitú.
- o) Limitný čas – prah, nad ktorým sa téma považuje za starú.

### **4.3 Testovanie a vyhodnotenie testov**

Aplikácia ChatVis je primárne určená pre vizualizovanie textovej komunikácie, no môže byť využitá pre vizualizáciu akéhokoľvek prúdu textu. Rozhodli sme sa, že v tejto práci otestujeme ChatVis na predpripravenej databáze, pretože testovanie v reálnom čase by bolo náročné na čas a najmä na prostriedky, keďže v čase testovania sme žiaľ nedisponovali s dostatočnou používateľskou základňou.

#### **4.3.1 Vizualizácia webového portálu DSL.sk**

Databáza pozostáva z 13881 predspracovaných článkov webového portálu DSL.sk, ktoré sme predspracovali pomocou nástrojov dostupných v administrátorskej časti aplikácie Chatu v sekcii pre generovanie korpusu. Takto pripravené články sme vložili do databázy a pripravili tak na spracovanie procesom LDA. Po spustení sa proces LDA musí inicializovať, čo pozostáva z načítania modelu do pamäte a následne začne vykonávať inferenciu na dosiaľ nespracovaných dátach z databázy.



```

C:\Windows\System32\cmd.exe

MOD: rap
MOD: loading model to memory
6: reading new data
  - here LDA reads wordmap.txt and inference.dat
8: reading wordmap for twords
4: initializing inference
Running Time of INITIALIZATION : 87.304000
Running ITERATION : #1
7: do inference
MOD: initialize inference
ID: 13810. Message: spolocnost google uvolnila pondelok novu verziu android aplikacie pristup gmailu priniesla viacero noviniek zlepzeni android verzie minimalne gmail verziu pribudla priamo notifikacii notifikacnej obrazovke moznost uskutočnit spravou zakladne akcie standardne gmail ponuka moznost spravu priamo archivovat otvorenia gmail aplikacie nu odpovedat samozrejme gmail aplikacii zoznamu z...

```

Obrázok 21, Inferencia pomocou LDA

Po spracovaní celej databázy sme sa prihlásili do aplikácie Chatu a následne spustili vizualizáciu. Vizualizovanie všetkých článkov trvalo približne päť hodín. Bolo spustené až po automatické aplikovanie posledného constraintu a teda vymazanie posledných najmenej spomínaných tém. Nastavenia vizualizácie boli ponechané na základných hodnotách.

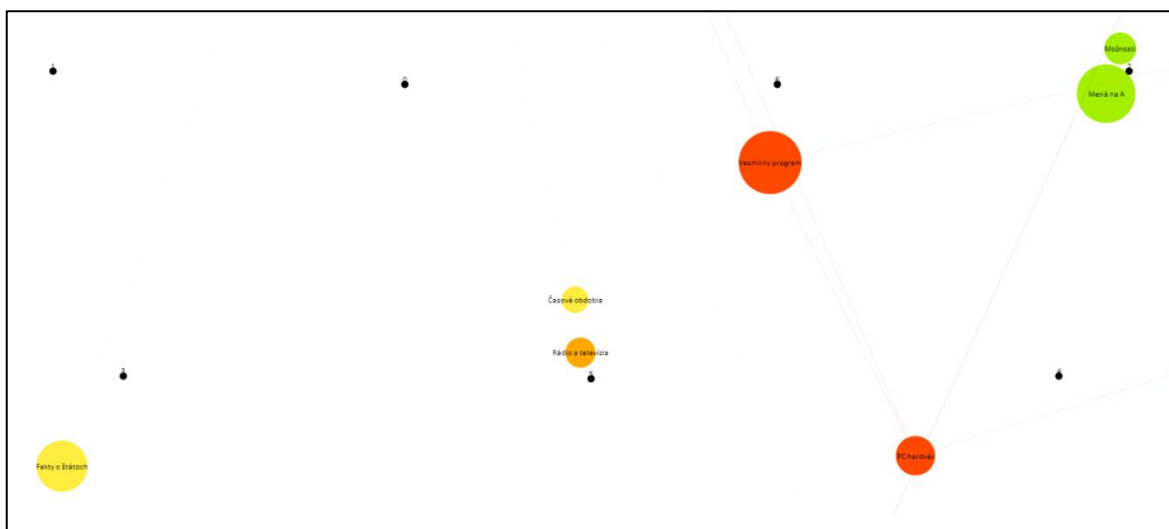
Názov skupiny	Hodnota farby (hex.)	Farba
Politika a ekonomika	#ffed40	
Jazyk a písmo	#a5ef00	
Krajiny	#c9007a	
Človek a príroda	#510fad	
Veda a technika	#ff4900	
Kultúra a umenie	#ffa900	
Šport	#4479d4	
Nezaradené témy	#00af64	

Tabuľka 1, Priradenie farieb skupinám tém vo vizualizácii

#### 4.3.2 Vyhodnotenie testu

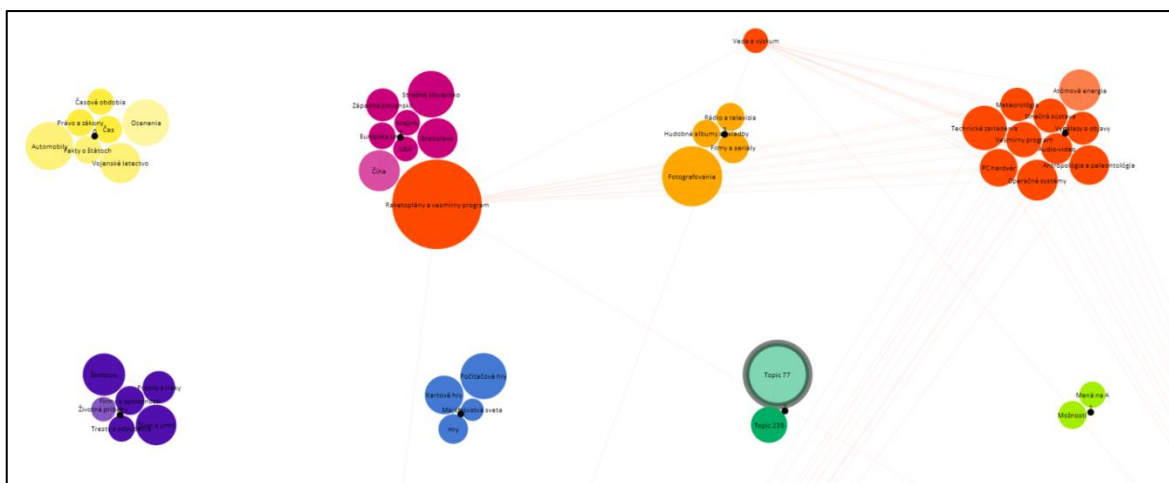
Analýza vizualizácie webového portálu DSL.sk prebiehala v dvoch fázach. Počas jej behu sme analyzovali vizualizáciu v hlavnom okne v reálnom čase a zároveň sme ukladali vizualizáciu v náhodných intervaloch pre neskoršiu analýzu. Webový portál DSL.sk má

zameranie na informačné technológie, takže sme očakávali vizualizáciu korešpondujúcu s týmto faktom. Tento predpoklad sa nám aj potvrdil. Inicializačná fáza prebiehala mierne chaosne, čo bolo očakávané chovanie a bez potreby zmeny akýchkoľvek nastavení sa vizualizácia po krátkej dobe ustálila.



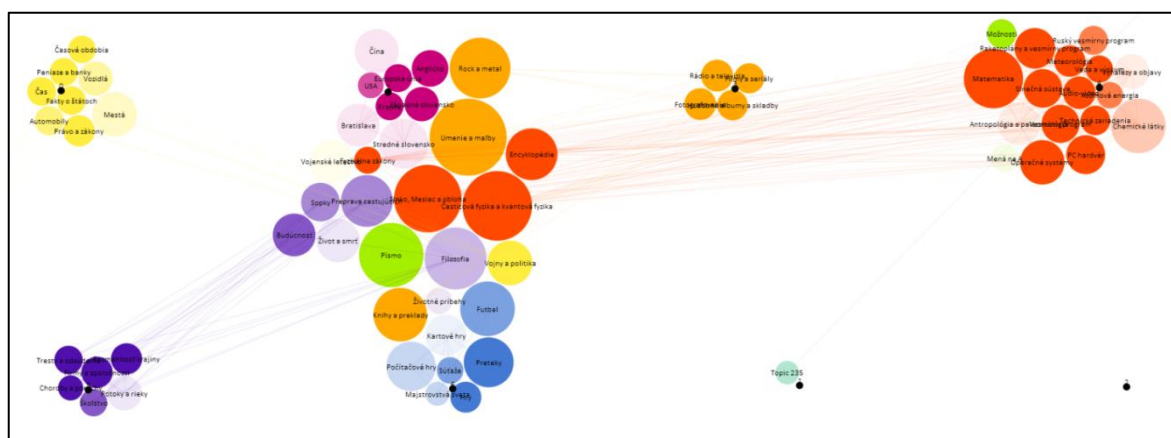
Obrázok 22, Inicializácia vizualizácie DSL.sk

Dominantná skupina zobrazená červenou farbou má názov *Veda a technika*, čo zodpovedá predpokladanému zastúpeniu na portáli DSL.sk. Do tejto skupiny spadajú témy pomenované *Vesmírny program* a *PC hardvér*, ktoré majú stúpajúcu tendenciu. Na obrázku vidno vygenerovaných sedem gravitačných ťažísk pre skupiny, no len štyri vyobrazené skupiny. Dôvodom je, že snímka bola spravená v čase, keď sa dané témy nachádzali ešte mimo viditeľného okna vizualizácie.



Obrázok 23, Snímka vizualizácie DSL.sk

Na ďalšej snímke vidíme opäť dominantné zastúpenie skupiny *Veda a technika*. Je reprezentovaná témami ako *Operačné systémy*, *Technické zariadenia*, *Atómová energia*, ktorú už vidíme ako starnúcu a pod. Zaujímavosťou je téma *Raketoplány a vesmírny program*, ktorá má najväčšie zastúpenie spomedzi všetkých tém a napriek tomu, že je zo skupiny *Veda a technika* je priťahovaná najviac témami ako *Čína*, *USA*, *Európska únia* a pod., ktoré sú zo skupiny *Krajiny*. Časť správy reprezentovanej touto tému získaná z jej aktuálneho prehľadu: „americka vesmirna agentura nasa utorok poobede nasho casu upgrade pocitacov vesmirnej stanici iss stratila telekomunikacny kontakt stanicou takmer tri hodiny na ...“ z dátumu 2013-01-31 12:00:00. Takéto náhľady na správy prispievajúce témam nám pomohli k lepšej analýze vizualizovaných dát.



Obrázok 24, Snímka vizualizácie DSL.sk

Zhruba v dvoch tretinách behu vizualizácie začali témy formovať nové zaujímavé skupiny. Téma *Raketoplány a vesmírny program* sa zmenšila na polovicu a pripojila k skupine *Veda a technika*, ktoré má stále dominantné postavenie. Novými lídrami sú témy ako *Časticová fyzika a kvantová fyzika*, *Slnko*, *Mesiak a obloha*, *Matematika* a *Encyklopédie*. Dominantnou témou je momentálne téma *Umenie a maľby* zo skupiny *Kultúra a umenie*. Najväčší zhluk vznikol úplným spojením skupín *Krajiny* a *Šport*, do ktorého sa pripojili niektoré súvisiace témy z iných skupín ako napríklad *Preprava cestujúcich*, *Filozofia*, či *Umenie a maľby*. Tento zhluk sa neskôr v behu vizualizácie zmenšil a presunul a nakoniec zanikol úplne.

Z tém, ktoré boli počas testovania zobrazené ľahko vieme vyčítať tematický obsah prúdu vizualizovaného textu. Pre špecifickejšie údaje o zobrazovanom texte si môžeme

nechať zobrazit' prehľad pre jednotlivé témy a zobrazit' konkrétne správy prispievajúce jednotlivým témam, alebo si nechať zobrazit' prúd textu samotný. Z neustále vznikajúcich štruktúr môžeme usudzovať o príbuznosti jednotlivých tém a skupín tém, čo prispieva k lepšiemu pochopeniu vizualizácie a jej následnej analýze. Vizualizované témy zodpovedajú našim predpokladom. Animácia vo vizualizácii nám uľahčuje stotožnenie sa s atribútom času, ktorý síce nie je presne namapovaný na pohyb, ale prispieva k lepšej skúsenosti používateľa. Čas teda možno vyčítať z postupného spriedhľadňovania tém, ktoré nám umožňuje vidieť reálny časový údaj v kontexte ostatných tém. Z histogramu, do ktorého sa ukladajú postupne časové úseky vizualizácie a môžeme porovnávať vizualizáciu v čase. ale aj konkrétne z panelu prehľadu, kde máme dostupné presné údaje.

Všetky dáta z testu sú dostupné na priloženom DVD nosiči.

# Záver

Úvodom tejto práce sa nám podarilo čitateľovi poskytnúť stručný náhľad do problematiky vizualizácie textovej komunikácie v reálnom čase, čím sme zároveň identifikovali diery na trhu v tejto oblasti vizualizácie. Oboznámili sme čitateľa so zámerom našej práce, teda snahou o navrhnutie a implementáciu metódy spracovania, spracovania a vizualizácie prúdu textu v reálnom čase. Keďže vieme, že výskum v tejto oblasti zďaleka nedosiahol svoje maximum, existuje priestor na neustále zdokonaľovanie toho, čo už na trhu máme, no zároveň existuje priestor na nové, kreatívne nápady a tejto myšlienky sme sa uchopili.

Rozhodli sme sa teda navrhnuť vlastný netradičný prístup vizualizácie a následne sme ho úspešne implementovali. V prvom rade sme museli navrhnuť metódu spracovania textu, ktorá by bola použiteľná v tomto kontexte. Mnohé existujúce metódy sú rýchle a spoľahlivé, no často trpia nedostatkami, ktoré by mohli vyústiť v nepoužiteľnosť týchto metód v reálnom čase. Preto sme sa rozhodli o použitie metódy Latentnej Dirichletovej alokácie. Je to robustná metóda, ktorá dáva pre potreby našej vizualizácie optimálny výstup. Inšpiráciou nám bolo taktiež jej využitie vo fulltextových vyhľadávачoch ako Google alebo Seznam.cz, čo síce nie je argumentom pre vhodnosť tejto metódy v našej práci, no bolo to silnou motiváciou pre jej preskúmanie, keďže oba prípady sa zaoberajú sémantickou analýzou textu. Táto metóda má veľký potenciál pri spracovaní textu v reálnom čase a výzvou pre nás do budúcnosti bude jej paralelizácia na GPU. Následne sme sa potýkali s prispôbením tejto metódy pre naše podmienky, teda jej prispôbenie pre inferenciu neustále prúdiaceho toku dát v reálnom čase. Implementovali sme niekoľko prístupov, z ktorého sme vybrali ten najvhodnejší pre nás a to klasický prístup do databázy pomocou procesu spusteného v nekonečnej slučke. Požiadavky na databázu sú kladené v predpísaných intervaloch a načítavanie z databázy a zápis do nej nevyúsťuje do konfliktov medzi klientmi. Pri veľkej záťaži na server, s ktorou sme sa v našej práci nepotýkali je možné databázový server distribuovať na viaceré servery podľa rôznych kritérií ako je geografická poloha a pod., čo prispieva k ďalšej optimalizácii. Toto bude potrebné najmä pri nasadzovaní výsledkov našej práce v komerčnej sfére pre bežných používateľov.



Následne sme popísali rôzne možné prístupy k vizualizácii prúdu textu pomocou silovo-riadených techník. Vybrali sme sa netradičnou cestou a namiesto zaužívaných riešení, o ktorých sme predpokladali, že budú fungovať, sme sa rozhodli použiť spôsob vizualizácie, ktorý sa bežne nepoužíva. Rozhodli sme sa použiť metaforu vesmírneho prostredia a teda sme namapovali výstupy zo sémantickej analýzy prúdu textu na simuláciu pseudo-gravitácie pôsobiacej na kruhové objekty v našej vizualizácii. Takýto spôsob vizualizácie je pre používateľa intuitívny a je v súlade s každodennou skúsenosťou používateľa, čo má za následok rýchle stotožnenie sa s nami naimplementovanou metaforou. Ide o hravý a nenásilný prístup vo vizualizácii, ktorý môže byť použiteľný ako v odbornej sfére, tak aj pre bežného používateľa.

V súvislosti s nami navrhnutým riešením sme sa však potýkali s problémom simulácie gravitácie medzi  $n$  telesami, ktoré sme pre nízky počet telies vyriešili výpočtom hrubou silou. Tento prístup sa nám z rôznych navrhnutých javil ako najlepší, čo sa na prvý pohľad podľa časových zložitostí algoritmov nezdá, no reálna implementácia je vzhľadom na nami vykonané optimalizácie pre javascriptový engine V8 rozhodne plynulá a nenáročná.

Vizualizáciu sme naimplementovali v moderných webových technológiách, ktoré nám umožňujú jej neskoršie rozšírenie na mnohé zariadenia od stolných počítačov po tablety a smartfóny. Možným rozšírením tejto časti práce je použitie Javascriptového servera Node.js a frameworku AngularJS, ktoré by priniesli ďalšie možnosti pre optimalizácie vizualizačného procesu pre prostredie webu.

Výsledkom tejto práce je teda plnohodnotná aplikácia vizualizujúca prúd textu v reálnom čase, použiteľná taktiež z predpripravenej databázy a následné simulovanie prúdu textu, čo sme využili pri testovaní a vyhodnocovaní našej implementácie. Aplikácia ponúka používateľovi mnohé nastavenia a možnosti interakcie, ktoré mu majú dopomôcť k lepšej analýze prúdu textu. Výsledky testov vizualizácie dopadli podľa našich očakávaní a vzhľadom na relatívnu úspešnosť niektoré metódy použité v tejto práci budú neskôr použité v implementácii vizualizácie prúdu textu do komerčného sektoru.

# Literatúra

- [1] C. Rohrdantz, D. Oelke, M. Krstajic a F. Fischer, „Real-Time Visualization of Streaming Text Data: Tasks and Challenges (Best Paper Award),“ rev. *Workshop on Interactive Visual Text Analytics for Decision-Making at the IEEE VisWeek 2011*, 2011.
- [2] B. Babcock, S. Babu, M. Datar, R. Motwani a J. Widom, „Models and issues in data stream systems,“ rev. *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, Madison, Wisconsin, ACM, 2002, pp. 1-16.
- [3] P. Dourish, A. Kobsa, G. Mark a D. Redmiles, „Information Visualization Research,“ University of California, [Online]. Dostupné na internete: <<http://www.isr.uci.edu/research-visualization.html>>. [Cit. 08 04 2013].
- [4] M. Hearst, „Information visualization: Principles, promise, and pragmatics.,“ rev. *Conference on Human Factors in Computing Systems*, 2003.
- [5] Usability first, „Usability first,“ [Online]. Dostupné na internete: <<http://www.usabilityfirst.com/glossary/information-visualization/>>. [Cit. 08 04 2013].
- [6] C. Tünnermann Bernheim a M. d. S. Chaui, „Challenges of the university in the knowledge society, five years after the world conference on higher education,“ rev. *UNESCO Forum Occasional Paper Series*, São Paulo, Brazil, 2003.
- [7] H. G. Funkhouser, „A Note on a Tenth Century Graph,“ *Osiris*, zv. 1, pp. 260-262, 1936.
- [8] The University of Georgia, „Minard's 1861 Map of Napoleon's Army's Russian Campaign,“ [Online]. Dostupné na internete: <<http://studio.coe.uga.edu/seminars/visualization/minard.html>>. [Cit. 15 04 2013].
- [9] B. Shneiderman, „The eyes have it: a task by data type taxonomy for information visualizations,“ rev. *Visual Languages, 1996. Proceedings., IEEE Symposium on*, 1996.
- [10] D. A. Keim, „Information Visualization and Visual Data Mining,“ *IEEE Transactions on Visualization and Computer Graphics*, zv. 8, %1. vyd.1077-2626, pp. 1-8, 2002.
- [11] J. Alsakran, Y. Chen, D. Luo, Y. Zhao, J. Yang, W. Dou a S. Liu, „Real-Time Visualization of Streaming Text with a Force-Based Dynamic System,“ *IEEE*

- Comput. Graph. Appl.*, zv. XXXII, %1. vyd.0272-1716, pp. 34-45, 2012.
- [12] E. R. Gansner, Y. Hu a S. C. North, „Visualizing Streaming Text Data with Dynamic Maps,“ *CoRR*, zv. 1206.3980, 2012.
  - [13] F. Mansmann, M. Krstajic, F. Fischer a E. Bertini, „StreamSqueeze: A Dynamic Stream Visualization for Monitoring of Event Data,“ rev. *Proceedings of Conference on Visualization and Data Analysis (VDA '12)*, 2012.
  - [14] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden a R. C. Miller, „Twitinfo: aggregating and visualizing microblogs for event exploration,“ rev. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Vancouver, BC, Canada, 2011.
  - [15] C. Albrecht-Buehler, B. Watson a D. A. Shamma, „Visualizing live text streams using motion and temporal pooling,“ *Computer Graphics and Applications, IEEE*, zv. 25, %1. vyd.3, pp. 52-59, 2005.
  - [16] A. Endert, P. Fiaux a C. North, „Semantic interaction for visual text analytics,“ rev. *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, New York, NY, USA, 2012.
  - [17] W. Cui, S. Liu, L. Tan, C. Shi, Y. Song, Z. Gao, H. Qu a X. Tong, „TextFlow: Towards Better Understanding of Evolving Topics in Text,“ *Visualization and Computer Graphics, IEEE Transactions on*, zv. 17, %1. vyd.12, pp. 2412-2421, 2011.
  - [18] T. L. Griffiths a M. Steyvers, „Finding scientific topics,“ *Proceedings of the National Academy of Sciences of the United States of America*, zv. 101, pp. 5228-5235, 2004.
  - [19] A. Hubmann-Haidvogel, A. M. P. Brasoveanu, A. Scharl, M. Sabou a S. Gindl, „Visualizing Contextual and Dynamic Features of Micropost Streams,“ rev. *Making Sense of Microposts*, 2012.
  - [20] A. Šilić a B. Bašić, „Visualization of Text Streams: A Survey,“ rev. *Knowledge-Based and Intelligent Information and Engineering Systems*, Zagreb, Springer Berlin / Heidelberg, 2010, pp. 31-43.
  - [21] J. Risch, A. Kao, S. Poteet a Y. Wu, „Text Visualization for Visual Text Analytics,“ rev. *Visual Data Mining*, Heidelberg, Springer Berlin / Heidelberg, 2008, pp. 154-171.
  - [22] L. I. Smith, *A tutorial on principal components analysis*, USA: Cornell University, 2002.
  - [23] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer a R. Harshman, „Indexing by latent semantic analysis,“ *Journal of the American Society for Information Science*,

- zv. 41, % 1. vyd.6, pp. 391-407, 1990.
- [24] T. K. Landauer, P. W. Foltz a D. Laham, „An introduction to latent semantic analysis,“ *Discourse processes*, zv. 25, % 1. vyd.2-3, pp. 259-284, 1998.
- [25] Z. Češka, *Využití moderních přístupů pro detekci plagiátů*, Plzeň: Západočeská univerzita v Plzni, 2008.
- [26] Bkkbrad, „Wikipedia,“ 24 02 2009. [Online]. Dostupné na internete: <<http://en.wikipedia.org/wiki/File:Plsi.svg>>. [Cit. 15 04 2013].
- [27] T. Hofmann, „Probabilistic Latent Semantic Analysis,“ rev. *In Proc. of Uncertainty in Artificial Intelligence, UAI'99*, 1999.
- [28] D. M. Blei, A. Y. Ng a M. I. Jordan, „Latent dirichlet allocation,“ *J. Mach. Learn. Res.*, zv. III, % 1. vyd.1532-4435, pp. 993-1022, 2003.
- [29] Bkkbrad, „Wikipedia,“ 24 02 2008. [Online]. Dostupné na internete: <[http://en.wikipedia.org/wiki/File:Latent\\_Dirichlet\\_allocation.svg](http://en.wikipedia.org/wiki/File:Latent_Dirichlet_allocation.svg)>. [Cit. 15 04 2013].
- [30] Tamassia, Roberto et al., *Handbook of Graph Drawing and Visualization*, CRC Press, 2013.
- [31] D. Hotson, „Antipodean,“ [Online]. Dostupné na internete: <<http://dhotson.tumblr.com/post/520720950/force-directed-graph-layout-in-javascript>>. [Cit. 15 04 2013].
- [32] S. G. Kobourov, „Force-Directed Drawing Algorithms,“ rev. *Handbook of Graph Drawing and Visualization*, Arizona, CRC Press, 2013, pp. 383-408.
- [33] Lucidation, „Wikipedia,“ 03 01 2012. [Online]. Dostupné na internete: <[http://en.wikipedia.org/wiki/File:2D\\_Quad-Tree\\_partitioning\\_of\\_100\\_bodies.png](http://en.wikipedia.org/wiki/File:2D_Quad-Tree_partitioning_of_100_bodies.png)>. [Cit. 17 04 2013].
- [34] X.-H. Phan a C.-T. Nguyen, *GibbsLDA++: A C/C++ implementation of latent Dirichlet allocation (LDA)*, 2007.
- [35] G. Heinrich, „Parameter estimation for text analysis,“ 2004.
- [36] S. Golemon, „Extension Writing Part I: Introduction to PHP and Zend,“ *Zend Developer Zone*, 01 05 2005. [Online]. Dostupné na internete: <<http://devzone.zend.com/303/extension-writing-part-i-introduction-to-php-and-zend/>>. [Cit. 20 04 2013].

- [37] WHATWG, „The canvas element,“ 15 04 2013. [Online]. Dostupné na internete: <<http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html>>. [Cit. 20 04 2013].
- [38] M. Grčar, V. Podpečan, M. Juršič a N. Lavrač, „Efficient visualization of document streams,“ rev. *Proceedings of the 13th international conference on Discovery science*, Canberra, Australia, 2010.
- [39] M. Stone, „In Color Perception, Size Matters,“ *IEEE Computer Graphics and Applications*, zv. 32, pp. 8-13, 2012.
- [40] G. Gorrell, *Generalized Hebbian Algorithm for Dimensionality Reduction in Natural Language Processing*, Linköping: Institutionen för datavetenskap, 2006.
- [41] J. Materna, „Blog fulltextového týmu,“ Seznam.cz, 22 12 2011. [Online]. Dostupné na internete: <<http://fulltext.sblog.cz/2011/12/22/semanticka-analyza-textu-6/>>. [Cit. 14 04 2013].
- [42] J. McIntire, O. I. Osesina a M. Craft, „Development of visualizations for social network analysis of chatroom text,“ rev. *Collaboration Technologies and Systems (CTS), 2011 International Conference on*, 2011.

# Zoznam elektronických príloh

Na priloženom DVD nosiči sa nachádzajú:

- Práca v elektronickej podobe
- Zdrojové kódy aplikácie
- Dokumentácia zdrojových kódov
- Návod na spoznanie aplikácie
- Testovacia databáza
- Súbory nastavenia pre server
- Ukážkové videá behu aplikácie