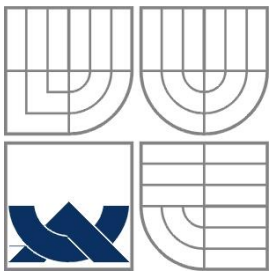
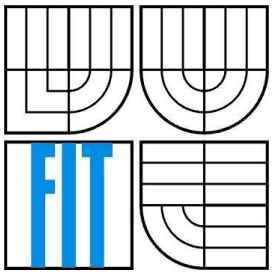


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SÉMANTICKÁ PODOBNOST TEXTŮ

SEMANTIC SIMILARITY OF TEXTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Martin Hajdin

VEDOUCÍ PRÁCE
SUPERVISOR
Doc. RNDr. Pavel Smrž, Ph.D.

BRNO 2016

Zadání bakalářské práce/18690/2015/xhajdi00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Hajdin Martin**
Obor: Informační technologie
Téma: **Sémantická podobnost textů**
Semantic Similarity of Texts
Kategorie: Umělá inteligence

Pokyny:

1. Prostudujte metody měření sémantické podobnosti slov a delších textových úseků.
2. Seznamte se s existujícími nástroji, které mohou být použity ke shlukování velkého počtu textových dokumentů na základě podobnosti.
3. Shromážděte data potřebná pro průběžné testování jednotlivých fází řešení problému.
4. Na základě získaných poznatků navrhněte a realizujte systém, který dokáže utřídit zadané texty do skupin a navrhnout pojmenování identifikovaných skupin na základě společného obsahu.
5. Vyhodnoťte realizované řešení a porovnejte zvolený přístup s jinými metodami.

Literatura:

- dle doporučení vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- funkční prototyp řešení

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Smrž Pavel, doc. RNDr., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Táto práca sa zaoberá problematikou určovania sémantickej podobnosti textov so zameraním na kategorizáciu webových dokumentov, v tomto prípade záložiek. Súčasťou spracovania je teoretický prehľad metód, pre implementáciu systému. Popisuje sa aj návrh a implementácia jednotlivých metód použitých v systéme. Práca sa taktiež zaoberá vyhodnotením jednotlivých metód, kde sú vybrané metódy otestované podľa určitých kritérií.

Abstract

This paper deals with the determination of the semantic similarity of texts focusing on categorization of web documents in this case bookmarks. The part of the process is a theoretical overview of methods for system implementation. It describes the design and implementation of the various methods used in the system, too. This paper also deals with the evaluation of various methods where the chosen method are tested according to specified criteria.

Kľúčové slová

sémantická podobnosť, vektorový model, spracovanie prirodzeného jazyka, Python, Gensim, Scikit-learn, TFIDF, LDA, NMF, SVD

Keywords

semantic simlarity, vector space model, natural language processing, Python, Gensim, Scikit-learn, TFIDF, LDA, NMF, SVD

Citácia

HAJDIN, Martin. *Sémantická podobnost textů*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Smrž Pavel.

Sémantická podobnosť textov

Prehlásenie

Prehlasujem, že som túto prácu vypracoval samostatne pod vedením Doc. RNDr. Pavel Smrž, Ph.D. Uviedol som všetky literárne pramene z ktorých som čerpal.

……
Martin Hajdin
11. 5. 2016

Poďakovanie

V tejto sekcii by som rád poďakoval vedúcemu Doc. RNDr. Pavlu Smržovi, Ph.D. práce za jeho pripomienky, čas a rady.

Obsah

1 Úvod

V súčasnej dobe sa často stretávame s veľkým množstvom neštruktúrovaných dát, ktoré by bolo vhodné spracovať s cieľom získania nejakého konkrétneho výstupu. Nie vždy je možné tieto dáta prechádzať a spracovávať, pretože by to bolo časovo a finančne náročné. Preto by bolo výhodnejšie spracovať systém, ktorý by dokázal roztriediť rôzne texty na základe sémantickej podobnosti. Pod sémantickou podobnosťou si môžeme predstaviť určovanie tém, alebo podobnosť dvoch a viacerých textových dokumentov. Pod témou môžeme rozumieť napr. počasie, šport, informatika a pod. Nie vždy je možné priradiť tému, ktorá by presne určovala obsah dokumentu. To znamená, že určovanie sémantickej podobnosti textov má veľké využitie v úlohách spracovania prirodzeného jazyka, napr. pri zisťovaní významu slov alebo získavaní informácii.

Cieľom tejto práce bolo zoznámiť sa s metódami určovania sémantickej podobnosti textov a navrhnúť a implementovať systém, ktorý bude schopný usporiadať webové dokumenty, v tomto prípade záložky podľa sémantickej podobnosti do kategórií a navrhovať pomenovania pre dané kategórie. Táto aplikácia uľahčí a urýchli triedenie veľkého počtu záložiek čo ušetrí čas každému z nás.

V kapitole 2 je zhrnutie metód, ktoré sa dnes používajú pri určovaní sémantickej podobnosti textov. Ide o časť obsahujúcu teoretický rozbor jednotlivých algoritmov. Kapitola 3 sa venuje návrhu a implementácii samotného systému. V tejto časti si priblížime dáta, s ktorými pracujeme a jednotlivé nástroje, ktoré boli použité pri implementácii. V predposlednej časti sa sústredíme na testovanie a vyhodnotenie samotného systému.

2 Teoretická časť

2.1 Určovanie sémantickej podobnosti textov

Existuje veľa modelov pre určovanie sémantickej podobnosti textov. Každý z nich má svoje klady a zápory, ktoré v nasledujúcich kapitolách zhodnotíme a priblížime tie, ktoré sa dnes používajú a ktoré boli použité vo finálnej aplikácii.

2.1.1 Booleovský model

Booleovské vyhľadávanie je najjednoduchšie z týchto metód, pretože sa spolieha na použitie logických operátorov. K zisťovaniu podobnosti dokumentov slúži ohodnocovanie slov v texte logickou jednotkou alebo nulou. Nevýhodou je, že podľa výskytu slov v texte môžeme určiť len približnú pravdepodobnosť. To znamená, že sa hľadaný výraz sa môže vyskytovať v texte, ale nemusí súvisieť s kontextom vyhľadávania. Využíva sa pri dopytovaní v jazyku SQL, kde môžeme hľadané výrazy spojovať s operátormi AND, OR a NOT .

2.1.2 Vektorový model

Jedná sa o jeden z najpoužívanejších modelov pre reprezentáciu dokumentov. Dokumenty, alebo dotazy sú reprezentované ako body v n-rozmernom priestore, kde každá dimenzia odpovedá určitému slovu z kolekcie dokumentov. Prítomnosť slova v konkrétnom dokumente môžeme reprezentovať napr. jednotkou alebo nulou. Tento model si viac priblížime v kapitole 2.2.

2.1.3 Pravdepodobnostný model

Tento model predpokladá, že pravdepodobnosť významu závisí na dotaze a reprezentácii dokumentov. Podľa pravdepodobnosti určí či je dokument relevantný k dotazu. Nevýhodou je, že slová nemajú váhu a sú považované za navzájom nezávislé. Využíva sa to pri zoraďovaní dokumentov podľa ich významu.

2.2 Vektorový model

Najpoužívanejší model pre reprezentáciu dokumentov. Každý z dokumentov je reprezentovaný ako bod v n-rozmernom priestore. Dimenzie v tomto priestore predstavujú jednoznačné slová z kolekcie dokumentov. Ak sa vyskytne nejaké jednoznačné slovo v niektorom z dokumentov, tak je hodnota vo vektore nenulová. Existujú rôzne výpočty týchto hodnôt, ktoré môžeme nazvať váhami jednoznačných slov. V nasledujúcich podkapitolách si priblížime najznámejšie metódy a vysvetlíme ich na príklade.

2.2.1 Ukážka priestoru

Táto podkapitola čerpá zo zdrojov [1] a [2].
Na nasledujúcej ukážke si vysvetlíme, ako určiť vektory nasledujúcich dokumentov. Máme kolekciu troch dokumentov(pre jednoduchosť je v každom dokumente práve jedna veta):

#	Dokument
1	today is sunny day on the beach
2	the sun is on the sky
3	the day was sunny

Tabuľka 1 Obsah ukážkových dokumentov

Ak vyberieme všetky použité slová a abecedne ich zoradíme, dostaneme nasledovný vektor slov [today, is, sunny, day, the, sun, on, sky, beach, was]. Pre dané zoradenie slov budú vektory reprezentujúce dokumenty vyzerať takto:

Vektor slov	today	is	sunny	day	the	sun	on	sky	beach	was
Dokument1	1	1	1	1	1	0	1	0	1	0
Dokument2	0	1	0	0	1	1	1	0	0	0
Dokument3	0	0	1	1	1	0	0	1	0	1

Tabuľka 2 Vektory dokumentov

Binárna reprezentácia je jednoduchá a v mnohých prípadoch postačujúca. Nevýhodou je, že nedokážeme určiť počet výskytov slov v dokumente. K tomu slúži metóda nazvaná TF[1].

Vektor slov	today	is	sunny	day	the	sun	on	sky	beach	was
Dokument1	1	1	1	1	1	0	1	0	1	0
Dokument2	0	1	0	0	2	1	1	0	0	0
Dokument3	0	0	1	1	1	0	0	1	0	1

Tabuľka 3 Vektory dokumentov s použitím term frequency

Nevýhodou tejto metódy je, že nastavuje vyššiu váhu obecne častým slovám, ktorými sú napr. spojky, predložky apod. Väčšinou sa používa normalizovaná forma TF, ktorá nenadhodnocuje slová. Pre výpočet sa používa vzťah:

$$tf_{(t,d)} = \frac{f_{(t,d)}}{max_{\{t',d\}} f_{(t',d)}} \tag{2.1}$$

Kde $f_{(t,d)}$ je počet výskytov slova v dokumente d a max je slovo s najväčším počtom výskytov v dokumente d.

[1] term frequency

2.2.2 TF-IDF

Táto podkapitola čerpá zo zdroja [3].
TF-IDF [2]je metóda, ktorá sa používa pre získanie unikátnych slov v dokumente, t. j. slová, ktoré sa v dokumente vyskytujú často a zároveň sa nevyskytujú v iných dokumentoch. Používa k tomu metódu IDF(Inverse Document Frequency), ktorá je definovaná ako logaritmus podielu celkového počtu dokumentov a počtu dokumentov v ktorých sa vyskytuje dané slovo:

$$idf_t = log \frac{N}{df_t} \tag{2.2}$$

Pre výpočet tf-idf sa použije nasledovná rovnica:

$$\mathrm{tfidf}_{\mathrm{t,d}} = \mathrm{tf}_{\mathrm{t,d}} \times \mathrm{idf}_t \tag{2.3}$$

Kde $\mathrm{tf}_{\mathrm{t,d}}$ je počet výskytu slova t v dokumente d.

Vektor slov	today	is	sunny	day	the	sun	on	sky	beach	was
Dokument1	0,48	0,18	0,18	0,18	0	0	0,18	0	0,48	0
Dokument2	0	0,18	0	0	0	0,48	0,18	0	0	0
Dokument3	0	0	0,18	0,18	0	0	0	0,48	0	0,48

Tabuľka 4 Vektory dokumentov s použitím TF-IDF

2.2.3 Skip-thoughts Vectors

Táto podkapitola čerpá informácie zo zdrojov [4] a [17].
Jedná sa o algoritmus strojového učenia bez učiteľa distribuovaného vetným kóderom. Použitím textu z kníh, je možné trénovať kódovací a dekódovací model, ktorý sa snaží rekonštruovať okolité vety z kódovacieho priechodu. Vety, ktoré zdieľajú sémantické a syntaktické vlastnosti sú mapované na podobné vektorové reprezentácie. To môže byť použité k určovaniu sémantickej podobnosti textov.

Skip-thoughts Vectors používa kódovacie a dekódovacie modely. To znamená, že kóder mapuje slová na vektory a dekóder slúži na generovanie okolitej vety. Kóder využíva slová vo vete *W* a ich počet *N* k vytvoreniu skrytého stavu *h*. Každým ďalším krokom kóder vytvára skrytý stav h^t , ktorý je interpretovaný ako reprezentácia vety, takže skrytý stav h^N predstavuje úplnú vetu. Zakódovať vetu môžeme nasledujúcou postupnosťou rovníc:

$$r^t = \sigma\,(W_r x^t + U_r h^{t-1}) \tag{2.4}$$

$$z^t = \sigma\,(W_z x^t + U_z h^{t-1}) \tag{2.5}$$

$$\bar{h}^t = \tanh(W x^t + U(r^t \odot \mathrm{h}^{t-1})) \tag{2.6}$$

$$h^t = (1 - z^t) \odot h^{t-1} + z^t \odot \bar{h}^t \tag{2.7}$$

kde $\bar{h}^t$ je navrhovaná aktualizácia stavu v čase *t*, z^t je aktualizácia vstupu, r^t je opätovné načítanie vstupu (Θ) označujúce komponentu. Obe aktualizácie nadobúdajú hodnoty medzi nulou a jednotkou.

[2] Term Frequency – Inverse Document Frequency

Dekodér je neurálny jazykový model, ktorý je podmienkou na výstupe kódera. Výpočet je podobný ako u kódera s výnimkou matríc C_r,C_z a C, ktoré sa používajú k vyhnutiu sa aktualizácii na vstupe, opätovnému načítaniu vstupu a skrytému stavu vypočítaného z vetného vektoru. Dekódovanie zahŕňa iterácie v nasledujúcom poradí:

$$r^t = \sigma(W_r^d x^{t-1} + U_r^d h^{t-1} + C_r h_i) \tag{2.8}$$

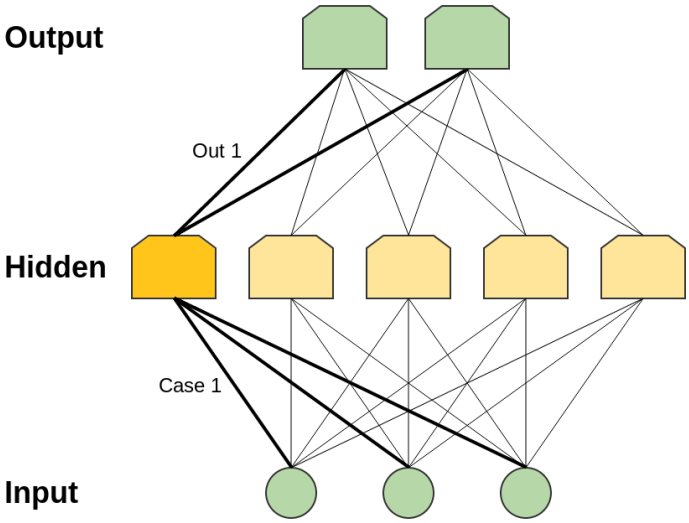
$$z^t = \sigma(W_z^d x^{t-1} + U_z^d h^{t-1} + C_z h_i) \tag{2.9}$$

$$\bar{h}^t = \tanh(W^d x^{t-1} + U^d(r^t \odot \mathrm{h}^{t-1}) + C h_i) \tag{2.10}$$

$$h_{i+1}^t = (1 - z^t) \odot h^{t-1} + z^t \odot \bar{h}^t \tag{2.11}$$

Tento model pracuje s tromi vrstvami :

- Vstupná vrstva
- Skrytá vrstva
- Výstupná vrstva



Obrázok 1 Graficky znázornené RNN[3].Prevzatý z [17]

Vstupná vrstva je v tomto prípade kóder a výstupná vrstva je dekóder. Na vstupe sa nachádzajú jednotlivé tokeny mapované na vysoko-rozmerný vektor. To nám umožňuje zistiť ich podobnosť medzi sebou, pretože vektory s podobnými hodnotami sú si podobné. Skrytý stav sa snaží predpovedať presné hodnoty na výstupe. Napríklad veta „The book is blue“ bude podobná s vetou „The book is brown“ alebo „The dictionary is blue“

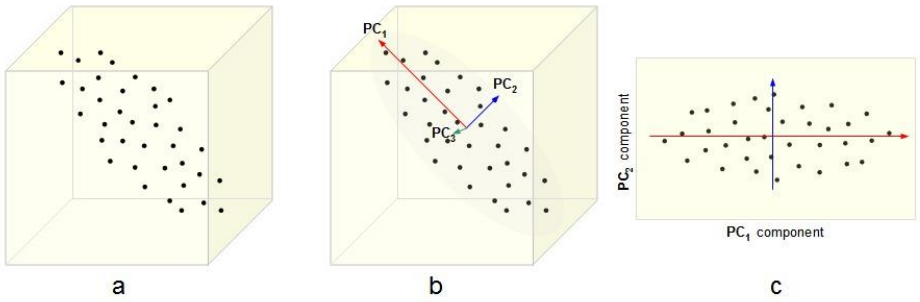
[3] Recurrent Neural Network

2.3 Dimenzionálna redukcia

Po určení sémantickej podobnosti textov a získaní vektorov nastáva v mnohých prípadoch problém, že vypočítané dáta sú vysoko-rozmerné. Práca s takýmito vektormi je veľmi náročná, preto je lepšie zredukovať rozmer týchto vektorov. V tejto kapitole priblížime metódy použité pri dimenzionálnej redukcii.

2.3.1 PCA

Táto podkapitola čerpá informácie zo zdrojov [12], [13] a [14].
PCA [4]je veľmi populárna technika pre dimenzionálnu redukciu. PCA si kladie za úlohu nájsť lineárny podpriestor o rozmere *d* v súbore dát menšieho ako ich počiatočný rozmer. Takto znížený podpriestor sa pokúša udržať variabilitu dát. Lineárny podpriestor môže byť špecifikovaný ortogonálnymi vektormi, ktoré tvoria nový systém súradníc nazývanými hlavné komponenty. Hlavnými komponentami rozumieme ortogonálne lineárne transformácie pôvodných dátových bodov, takže nemôže obsahovať viac rozmerov ako majú originálne komponenty.



Obrázok 1 Ukážka zredukovania 3 rozmerného priestoru na 2 rozmery. Obrázok je prevzatý z [14]

Základnou charakteristikou každého hlavného komponentu je jeho rozptyl. Hlavné komponenty sú zoradené podľa dôležitosti, t. j. podľa klesajúceho rozptylu. Prvý hlavný komponent obsahuje najviac informácii o pôvodných dátach, druhý hlavný komponent zase o rozptyle pôvodných dát, neobsiahnutých v prvom komponente. Najmenej informácii je obsiahnutých v poslednom komponente.

K výpočtu hlavných komponent potrebujeme mať maticu X typu $n\ x\ M$, ktorá je vstupom a M obsahuje n-rozmerné dáta vo svojich stĺpcoch. Okrem toho predpokladá, že najnižšia hodnota pre všetky rozmery je nula. To znamená, že dáta sú sústredené na stred. Cieľom je nájsť ortonormálnu transformačnú maticu $M\ x\ M$ obsahujúcu hlavné komponenty tak, že :

$$Y = P^T X \tag{2.12}$$

Kde stĺpce matice Y sú zobrazené ako hlavné komponenty.

$$PP^T = I \tag{2.13}$$

P je ortonormálna matica.

$$YY^T = D \tag{2.14}$$

[4] Principal components analysis

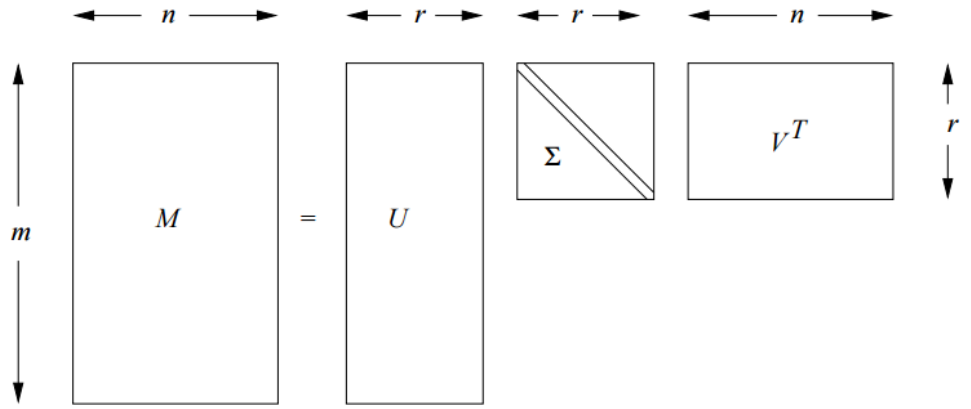
Kovariancia matice, ktorá je zobrazená v bodoch Y, je diagonálna matica, takže výsledné zobrazenie je nekorelované.

2.3.2 SVD

Táto podkapitola čerpá informácie zo zdroja [5].
SVD[5] slúži k redukcii vysoko-rozmerných matíc. Tento prístup umožňuje presnú reprezentáciu akejkoľvek matice a tiež uľahčuje odstránenie menej dôležitých častí tohto zobrazenia k produkovaniu približnej matice s požadovaným počtom rozmerov.

Nech M je $m \times n$ matica, a hodnosť matice M bude r. Hodnoť matice je maximálny počet riadkov (alebo stĺpcov) pre ktorú nie je lineárna kombinácia riadkov nulový vektor. Potom môžeme nájsť matice U, Σ, a V, ako je znázornené na obrázku 2 s nasledujúcimi vlastnosťami:

1. U je m x r stĺpec ortonormálnej matice; to znamená, že každý z jej stĺpcov je jednotkový vektor a skalárny súčin akýchkoľvek dvoch stĺpcov je nula.
2. V je n x r stĺpec ortonormálnu matice. Vždy používame transponovanú formu V, takže riadky V^T sú ortonormálne.
3. $\sum$ je diagonálna matica to znamená, že všetky prvky ktoré nie sú na diagonále sú nulové. Prvky $\sum$ sa nazývajú singulárne hodnoty M.



Obrázok 2 forma singulárneho rozkladu. Prevzatý z [5]

2.4 Zhlukovanie dokumentov

Zhlukovanie dokumentov predstavuje vytváranie jednotlivých kategórie. Existujú rôzne metódy vytvárania zhlukov, napr. hierarchické zhlukovanie na báze ťažiska, zhlukovanie na základe hustoty a ďalšie. V nasledujúcich častiach si priblížime metódy na báze ťažiska.

2.4.1 K-means

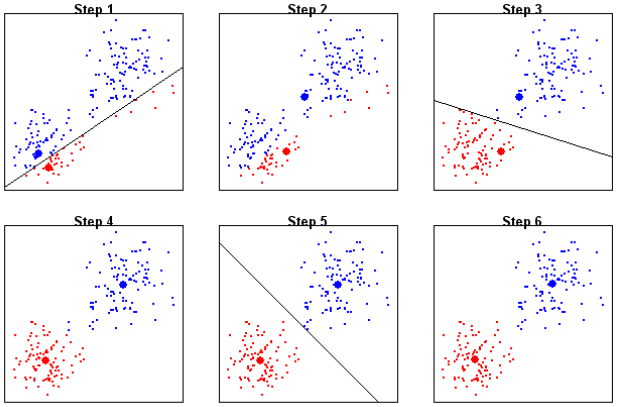
Táto podkapitola čerpá informácie zo zdrojov [3], [6] a [7]. K-means je metóda, ktorá umožňuje triedenie do zhlukov. Táto metóda sa zaraďuje do nehierarchických zhlukovacích metód.

Základnou charakteristikou tejto metódy je, že podobnosť jednotlivých dátových bodov a zhlukov sa meria ako ich vzdialenosť na priemernú hodnotu zhluku. Cieľom metódy je minimalizácia priemernej vzdialenosti medzi dátovými bodmi v rovnakom zhluku. Na počiatku je nutné určiť počet

[5] Singular-Value Decomposition

zhlukov a náhodne sa vyberú stredy zhlukov. Nasledovne sa prevedie daný algoritmus. Kroky algoritmu:

1. Náhodne je vybraných k dátových bodov, ktoré sú považovane za stredy zhlukov C = $\{c_1,c_2,c_3,...,c_k\}$
2. Každý dátový bod je potom priradený k najbližšiemu stredu zhluku C.
3. Prepočítajú sa nové stredy zhluku.
4. Ak sú dátové body v rovnakom zhluku ako to bolo v predošlej iterácii, tak sa algoritmus ukončí. Pokiaľ tomu tak nie je algoritmus opakuje kroky 2. a 3. dokiaľ nie je splnená podmienka. Ukončovacia podmienka môže byť rôzna, napr. dosiahnutie maximálneho počtu iterácii.



Obrázok 3 Algoritmus popísaný v krokoch. Prevzatý z http://sherrytowers.com/2013/10/24/k-means-clustering/

Algoritmus K-means neponúka žiadne záruky presnosti, ale jeho jednoduchosť a rýchlosť sú veľmi atraktívne pre to, aby sa používali v praxi.

K-means neposkytuje záruku nájdenia aproximácie. K-means++ rieši tento problém pri inicializácii stredov zhluku. S touto inicializáciou je zaručené, že nájdeme riešenie, ktoré je $O(\log k)$ konkurencieschopný. Kroky algoritmu:

1. Vyberieme jeden stred prvého zhluku náhodne zvolený z dátových bodov
2. Pre každý dátový bod x vypočítame $D(x)$, čo je vzdialenosť od dátového bodu k najbližšiemu stredu ktoré bolo určené
3. Vyberiem náhodne nový dátový bod ako nové váhové centrum , kde je bod x patriaci do dátových bodov zvolený s váhou pravdepodobnosti $D(x)^2$
4. Opakujem body 2. a 3. dokiaľ nie sú vybrané všetky K stredy
5. Pokračuje s použitím klasickej metódy K-means.

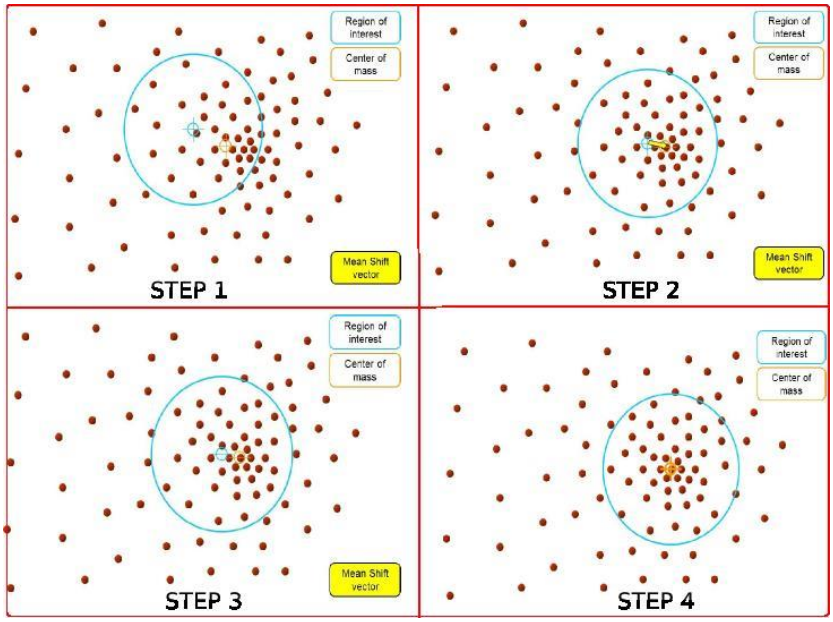
2.4.2 Mean shift

Táto podkapitola čerpá informácie zo zdrojov [10] a [11].
Hlavnou myšlienkou algoritmu mean shift je spracovať vlastnosti bodov v n rozmernom priestore ako empirickú funkciu hustoty pravdepodobnosti, kde oblasti v priestore zodpovedajú lokálnym maximám alebo základným distribúciám. Pre každý bod v priestore vypočíta jednu procedúru gaussova stúpania na odhadovanej hustote až konvergencie. Stacionárne body tohto prístupu predstavujú spôsoby distribúcie. Navyše body, ktoré sú spojené s rovnakým stacionárnym bodom sú považované za členov rovnakého klastra.

Mean shift vektor m ukazuje na smer maximálneho zvýšenia hustoty a je priamo úmerný na odhade gradientu hustoty v bode x.

Majme dáta $X = \{x_0, x_1, \ldots, x_i\}$, tak pre každé *x* z *X* vypočítame mean shift nasledovne:

1. Vypočítame mean shift vektor $m(x_i^t)$
2. Posunieme x podľa odhadu hustoty $m(x_i^t)$
3. Opakujeme body 1 a 2 až do konvergencie



Obrázok 4 Mean shift v krokoch. Prevzatý z http://shukra.cedt.iisc.ernet.in/edwiki/MTech_Projects-2012--Multiple_Camera_Surveillance_System

Mean shift je postup pre lokalizáciu maxima funkcie hustoty z diskrétnych dát. Ide o iteračnú metódu, pri ktorej sa začína odhadnutím počiatočného *x*. Funkcia $K(x_i - x)$ musí byť daná. Táto funkcia určuje váhy okolitých bodov k odhadu strednej hodnoty. Obvykle sa používa Gaussova funkcia na vzdialenosti od aktuálneho odhadu.

$$K(x_i - x) = e^{-c||x_i - x||^2} \tag{2.14}$$

Vážená stredná hodnota hustoty sa potom vypočíta v priestore určeným *K*.

$$m(x) = \frac{\sum x_i \in N(x)\, K(x_i - x) x_i}{\sum x_i \in N(x)\, K(x_i - x)} \tag{2.15}$$

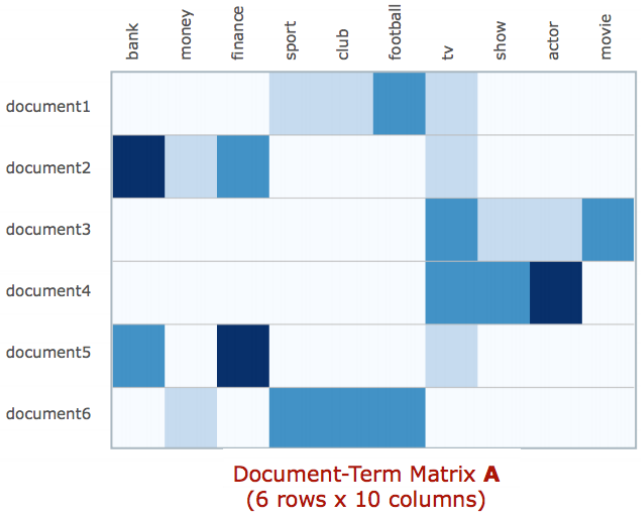
N(x) je okolie bodu *x*, množina bodov pre ktoré platí $K(x) \neq 0$

2.5 Získavanie názvu kategórii

Získavanie názvu pre kategórie je dôležitou súčasťou pri vytváraní zhlukov. Aby bolo možné užívateľa informovať o tom aké dokumenty sa nachádzajú v zhluku, je nutné vytvoriť témy pre dokumenty podľa ktorých môžeme identifikovať obsah dokumentov. V nasledujúcich podkapitolách si priblížime metódy, ktoré sa používajú pri získavaní tém.

2.5.1 NMF

Táto podkapitola čerpá informácie zo zdrojov [8] a [15].
NMF[6] je alternatívny prístup k rozkladu a predpokladá, že dátové komponenty sú nezáporné. Vstupom je nezáporná matica, ktorú rozdelí do dvoch menších matíc *W* a *H*, z ktorých každá má rozmer *k*. Po vynásobení týchto matíc sa priblížime pôvodnej matici *A*. Riadky matice nám poskytujú vstupné dokumenty, stĺpce poskytujú podmienky vzťahu k témam. Zistením hodnôt v danom stĺpci a zoradením top pojmom môžeme vybrať vhodný popis k danej téme.



Obrázok 5 Ukážka vytvárania tém. Prevzatý z [8]

Po vytvorení matice A môžeme nasledovne použiť NMF pre získanie tém :

- Vyberieme dvojicu faktorov s rozmermi k. NMF algoritmy sú často inicializované s náhodnými faktormi. To však môže viesť k nevyhovujúcim výsledkom v závislosti na výbere náhodného faktoru. Pre zabezpečenie jednotného výstupu sa vygenerujú počiatočné faktory pomocou metódy Non-negative Double Singular Value Decomposition.
- Použitím štandardnej euclideanskej formy NMF a s rozložením faktorov pre určitý počet iterácii vytvoríme výsledné matice *W* a *H*
- Výsledné témy sú potom definované ako:
    a) Popis témy je daný z najunikátnejších slov, ktoré sa nachádzajú v stĺpcoch faktora *W*
    b) Téma určená pre dokument a jej váha je daná hodnotami v riadkoch *H*

2.5.2 LDA

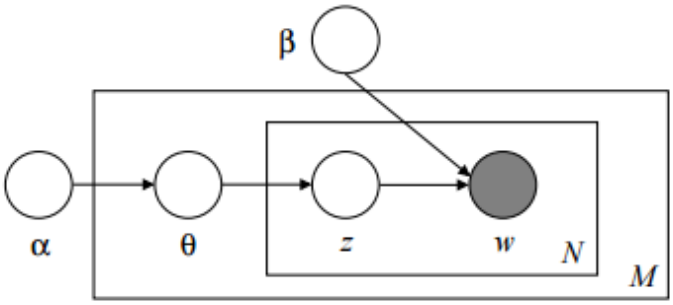
Táto podkapitola čerpá informácie zo zdrojov [9] a [16]. LDA[7] je pravdepodobnostný model generovaný na korpuse. Základnou myšlienkou je, že dokumenty sú reprezentované ako náhodné kombinácie nad latentnými témami, kde je každá téma charakterizovaná distribúciou slov.

LDA predpokladá nasledujúci generatívny proces pre každý dokument *w* v korpuse *D*:

1. Vyber $N \sim Poisson(\xi)$
2. Vyber $\theta \sim DIR(\alpha)$
3. Pre každé slová w_n z N:
    a) Vyber tému $z_n \sim Multinomial(\theta)$

[6] Non-negative Matrix Factorization
[7] Latentná Dirichletova alokácia

b) Vyber slovo w_n z $p(w_n|z_n,\beta)$, kde je multinomická pravdepodobnosť podmienená na tému z_n.



Obrázok 6 Grafický model reprezentácie LDA, prevzatý z [9]

Na obrázku 6 je LDA reprezentovaný ako pravdepodobnostný grafický model. Môžeme si všimnúť, že vyobrazenie LDA je rozdelené do troch úrovní. Parametre α a β sú parametre na úrovni korpusu. Predpokladá sa, že výber týchto vzoriek prebieha v procese generovania korpusu. Premenné θ_d sú na úrovni dokumentov. Premenné z_{dn} a w_{dn} sú na úrovni slov a každé slovo je vzorkované raz pre každý dokument.

LDA pracuje s určitým modelom. Predpokladá sa, že dokument obsahuje rôzne témy a slová v dokumente sú generované z týchto tém. Všetky dokumenty obsahujú špecifický súbor tém, ale podiel tém v dokumente je rozdielny. Generovanie modelu možno opísať takto:

- Majme dirichletovu distribúciu parametrov α a β o dĺžke *K* a *V*
- Pre tému číslo jedna až po tému číslo *K* vytvor distribúciu slov s multinomálnym vektorom $\emptyset_k$ podľa β
- Pre dokument číslo jedna až téma číslo *M* vytvor distribúciu tém s multinomálnym vektorom θ podľa α. Potom pre každé slovo v dokumente vytvor:
    - Tému z podľa θ, $kde\ z \sim Multinominal(\theta)$
    - Slovo w podľa $\emptyset_z$, $kde\ w \sim Multinominal(\emptyset_z)$

Dĺžka *V* je mohutnosť slovnej zásoby. Slová v dokumentoch sú pozorované, ale vytváranie tém je latentné.

2.6 Metódy predspracovania textu

Spracovanie problémov v oblasti prirodzeného jazyka pre určenie sémantickej podobnosti textov potrebujú určité predspracovanie textov. V tejto kapitole si stručne vysvetlíme základné metódy predspracovania textu.

2.6.1 Tokenizácia

Tokenizácia rozdeľuje slová na informačné celky. Podľa potreby sa môže jednať o vety, slová alebo entity (jednoslovné alebo viacslovné). Za token sa považuje akýkoľvek reťazec znakov medzi dvoma medzerami, aj jednotlivé znaky interpunkcie, ktoré nemusia byť oddelené medzerou od predchádzajúceho alebo nasledujúceho tokenu.

2.6.2 Stop-Words

Odstránenie nadbytočných slov, ktoré v prípade sémantickej podobnosti nenesú žiadnu sémantickú informáciu (napr. predložky a spojky).

2.6.3 Stemming

Určenie takzvaného „koreňa“ slova. To znamená, že stemmer pracuje so slovami a odstraňuje z nich predpony, prípony a koncovky. Výsledkom je koreň slova, ktorý nemusí byť zmysluplný.

2.6.4 Lematizácia

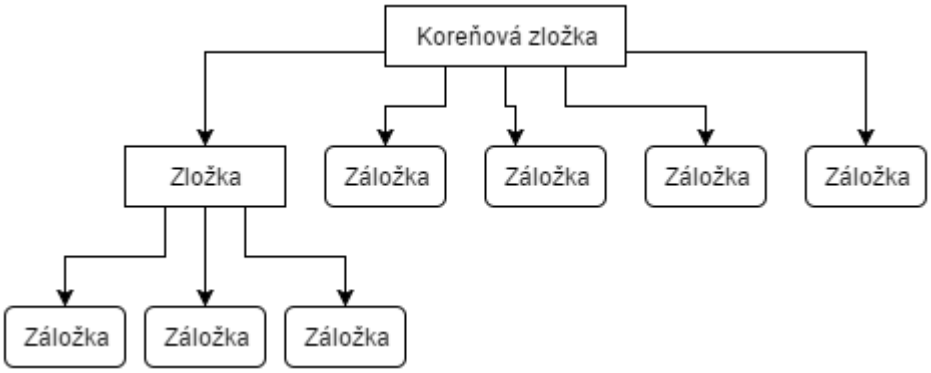
Určenie základného tvaru slova. Často používa morfologickú analýzu, takže výsledok záleží na kontexte slova. Ak lematizátor nemôže posúdiť kontext, tak nie je schopný zvoliť správny význam slova.

3 Návrh a implementácia

3.1 Analýza dát

3.1.1 Záložky

V oblasti WWW[8] sú záložky takzvané jednotné identifikátory zdroja (URI). Všetky moderné prehliadače obsahujú funkcie pre prácu zo záložkami. Záložky sú organizované v stromovej štruktúre, kde každý uzol je buď záložkou alebo zložkou.



Obrázok 7 Ukážka stromovej štruktúry záložiek

Jednotlivé záložky sú reprezentované ako objekty, kde je každý objekt zložený z atribútov :

1. ID - jedinečný identifikátor pre uzol. ID sú jedinečné v rámci aktuálneho profilu a zostávajú v platnosti aj po tom čo je prehliadač reštartovaný.
2. ParentId - ID nadradeného priečinka. Neexistuje u koreňového uzla.
3. Index- Poloha daného uzla v rámci nadradenej zložky.
4. URL[9] - Adresa jednotného vyhľadávača zdroja na ktorý sa bude navigovať v prípade kliknutia na záložku. Neobsahujú zložky.
5. Title - Názov alebo titulok daného uzlu.
6. DateAdded - Dátum kedy bol vytvorený uzol, v milisekundách od epochy.
7. DateGroupModified - Posledná zmena obsahu danej zložky, v milisekundách od epochy.

3.1.2 Získavanie obsahu zo záložiek

Pre získanie obsahu je potrebné vedieť, ako pristúpiť k dokumentu a kde hľadať obsah. Prístup k dokumentu je zabezpečený pomocou atribútu URL v záložkách. Dokumentom sa v tomto prípade rozumie HTML[10] súbor stiahnutý z URL. HTML je značkový jazyk určený na vytváranie webových

[8] World Wide Web
[9] Uniform Resource Locator – Jednotný vyhľadávač zdrojov
[10] Hypertextový značkový jazyk

stránok a iných informácii zobraziteľných vo webovom prehliadači. Základná štruktúra HTML je zložená z elementu ohraničujúceho stránku, hlavičky a tela.

Obrázok 8 Ukážka základnej štruktúry HTML stránky. Prevzaté z http://owebu.org/cze/html/html_struktura.php

Zo štruktúry stránky je možné vidieť, že obsah samotnej stránky je obsiahnutý v elemente <body>. Keď vieme kde sa nachádza obsah stránky, tak môžeme prejsť k extrakcii dôležitých časti.

Existuje veľa rôznych prístupov k extrakcii textu, napr. použitím regulárnych výrazov. V tomto prípade je použitá knižnica Boilerpipe, ktorá zabezpečuje sťahovanie a extrakciu stránok. Stiahnutie a extrakciu obsahu zabezpečuje funkcia _Donwload, ktorá sa nachádza v súbore Downloader.py. Pre urýchlenie sťahovania sú použité vlákna, takže sťahovanie viacerých stránok prebieha paralelne. Po stiahnutí obsahu rozdelí obsah stránky na tokeny, ktoré následne prevedie na malé písmená a odstráni z obsahu stopwords. V prípade, že stránka neexistuje alebo nie je možné stiahnuť obsah, zaradí takúto stránku do kategórie „Unsorted“. Chyby nastávajú aj v prípade, keď je obsah stránky tvorený z jazyka Javascript, alebo aplikácie flash. V takomto prípade získame prázdny obsah a stránka je tiež zaradená do kategórie „Unsorted“.

3.2 Vektorový priestor

Vytvorenie vektorového priestoru zabezpečuje funkcia skip_thoughts, ktorá s nachádza v súbore Algo.py. Funkcia vytvára vektory jednotlivých dokumentov podľa ich sémantickej podobnosti. Keďže skip-thoughts vectors pracuje s celým článkom, tak je nutné spojiť tokeny jednotlivých článkov, pretože pri sťahovaní bol text jednotlivých dokumentov rozdelený na tokeny, aby bolo možne odstrániť stopwords. Použitý model je dostupný priamo od autorov a jeho stiahnutie je zabezpečené použitím skriptu GetModels.sh. Samotný model bol vytvorený na 11038 knihách rôznych žánrov. Tento model bol použitý, pretože trénovanie modelu je časovo náročné zvlášť pri vyšších rozmeroch a je nutné mať veľký korpus rôznych textov.

3.2.1 Redukcia dimenzií

Model poskytnutý od autorov vytvára vektory o rozmere 4800 to znamená, že dimenzia vytvorených vektorov je príliš vysoká pre ďalšie spracovanie a tak je nutné ju zredukovať. To je zabezpečené metódou SVD, ktorá zredukuje rozmer vektorov na dve.

3.2.2 Vytváranie zhlukov

Pre vytvorenie zhlukov je použitý algoritmus k-means++ vysvetlený v kapitole 2.4.1. Použitím funkcie kmeans, ktorá sa nachádza v súbore Algo.py. Samotná funkcia vytvára určitý počet zhlukov, ktoré sú predom známe na dvoj rozmernom vektorovom priestore.

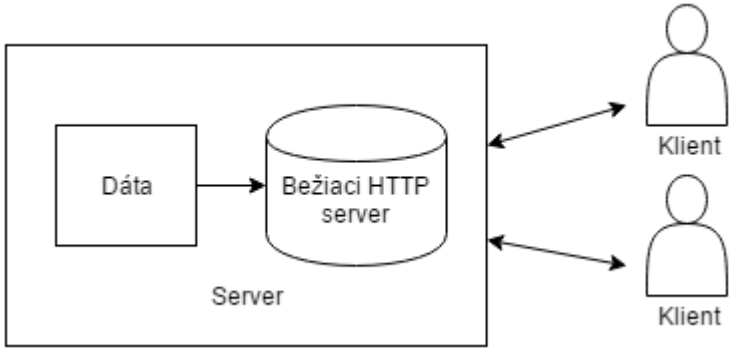
3.2.3 Získavanie názvu kategórie

Získanie názvu kategórie zabezpečuje funkcia GetCategories, ktorá sa nachádza v súbore Algo.py. V tejto funkcii sú spracované dve metódy NMF a LDA. Klient rozhoduje o tom, ktorá metóda sa použije. Získavanie kategórii pracuje s dokumentami, ktorých text je predspracovaný. To znamená, že je rozložený na tokeny a sú odstránené slová, ktoré sa v texte objavujú len raz a taktiež sú slová prevedené na koreňové slová.

3.3 Popis systému

Systém je vytvorený na komunikácii medzi serverom a klientom. Celý systém môžeme teda rozdeliť do dvoch častí:

1. Serverová časť- Používa jednotlivé metódy a algoritmy pre určovane sémantickej podobnosti textov a následné vytvorenie kategórii a ich pomenovania.
2. Klientská časť- Sprostredkúva nástroje pre prácu so záložkami a komunikáciu so serverom.



Obrázok 9 Štruktúra systému

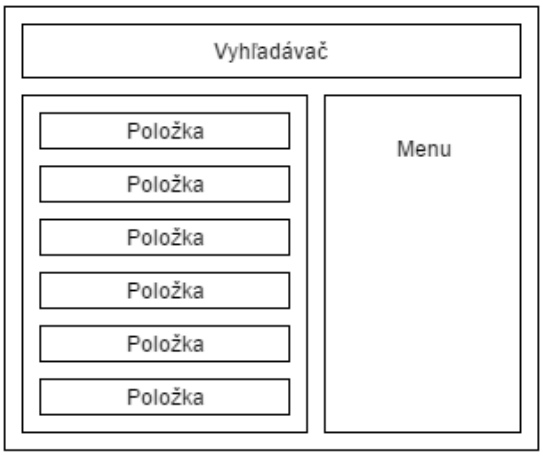
3.3.1 Klient

Klient je vytvorený ako doplnok v prehliadači. K práci so záložkami používa Google Chrome API, ktoré umožňujú rôznu manipuláciu s prehliadačom. Užívateľské rozhranie doplnku je pomerne jednoduché, aby sa uľahčila a urýchlila interakcia s užívateľom.

Možnosti ktoré bude umožňovať doplnok sú:

- Vyhľadávanie v záložkách
- Prehľadávanie záložiek, ktoré sa nachádzajú v prehliadači
- Základné možnosti pre prácu zo záložkami:
    - Otvorenie záložky v novom okne
    - Editovanie záložky alebo priečinku
    - Odstránenie záložky

- Vytváranie nových záložiek a priečinkov
- Usporadúvanie záložiek

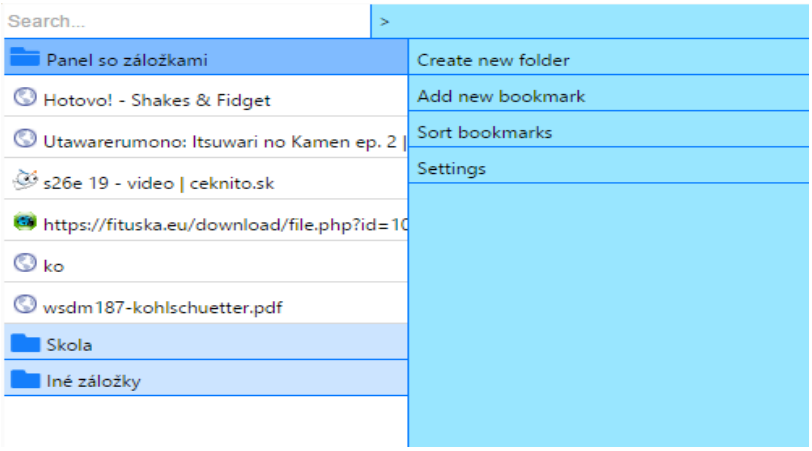


Obrázok 10 Návrh užívateľského rozhrania

Užívateľské rozhranie je rozdelené na tri časti a to z vyhľadávača, menu a časti v ktorej sa zobrazujú záložky. Vyhľadávač je spracovaný v skripte CreateTree.js, ktorý po zadaní textu do poľa pre vyhľadávanie, vyhľadá len tie záložky v ktorých sa zadaný text nachádza.

Zobrazenie záložiek zabezpečuje funkcia CreateTree, ktorá je spracovaná v skripte CreateTree.js. K vytvoreniu stromovej štruktúry záložiek je použitá funkcia z Chrome Extension API nazvaná bookmarks. Táto funkcia vracia pole záložiek, ktoré sa nachádzajú v prehliadači. Pre vygenerovanie záložiek sú vytvorené dve šablóny, ktoré sa nachádzajú vo funkcii Children. Prvá šablóna slúži pre vytvorenie priečinku. Záložky ktoré sa nachádzajú v priečinku sa zobrazia až po otvorení, to urýchľuje vytváranie položiek. Druhá šablóna vytvára samotné záložky. Jednotlivé položky sú následne vložené do pripraveného bloku v súbore popup.html.

Menu je spracované v skripte SideBar.js. V tomto skripte sú spracované jednotlivé funkcie položiek menu. Samotné položky sú vytvorené v súbore popup.html. Položky pre vytvorenie novej záložky alebo zložky je použitá funkcia chrome.bookmarks.create, ktorá pri zadaní URL vytvorí záložku, inak vytvorí zložku. V položke nastavenia môžeme zmeniť jazyk a nastaviť IP adresu a port serveru. Pre načítanie iného jazyka je použitá funkcia chrome.i18n.getMessage, ktorá zoberie text zo súboru _locales a prepíše text v elementov. Prepísanie textov zabezpečuje skript locales.js. Taktiež je použité cookies pre uloženie dát, ktoré ostanú uložené aj po vypnutí prehliadača.



Obrázok 11 Užívateľské rozhranie doplnku

O komunikáciu so serverom zabezpečuje skript background.js, ktorý beží na pozadí prehliadača. To znamená, že skript pracuje aj po zatvorení užívateľského rozhrania. Vďaka tomu môže užívateľ ďalej pracovať a nemusí čakať, napr. na odpoveď zo serveru. Doplnok komunikuje so skriptom na pozadí pomocou zasielania správ, ktorých nástroje sú obsiahnuté v Chrome Extension API. Tento skript taktiež odosiela a prijíma dáta zo serveru, ktoré po prijatí spracuje.

Názov	Funkcia
_locales	Priečinok obsahujúci jazyky aplikácie
css	Priečinok obsahujúci štýl stránky
Background.js	Skript, ktorý spracúva požiadavky servera
ContextMenu.js	Skript na vytvorenie kontext menu
CreateTree.js	Skript na vytvorenie stromu záložiek
locales.js	Skript, ktorý rieši jazyk aplikácie
SideBar.js	Skript na vytvorenie bočného panela
manifest.json	Súbor obsahujúci potrebne informácie pre spustenie doplnku
popup.html	HTML súbor do ktorého sa generuje vzhľad

Tabuľka 5 Popisujúca zdrojové súbory klienta a ich funkcie

Aplikácia obsahuje základnú funkčnosť, avšak niektoré prvky nie sú dopracované a nebola odtestovaná na rôznych užívateľoch. To znamená, že aplikácia môže obsahovať chyby. Doplnok funguje na prehliadači Opera a Google chrome.

3.3.2 Server

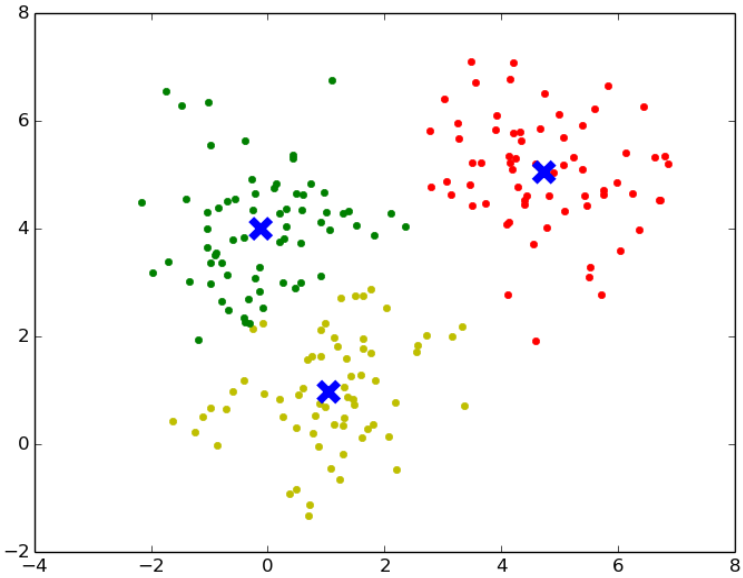
Server spracováva požiadavky užívateľa. Server pracuje ako konkurentný čo znamená, že dokáže spracovať požiadavky od viacerých užívateľov súčasne. Funkciu serveru môžeme rozdeliť na dve časti:

1. Trénovanie modelov
2. Spracovanie požiadavkou

Trénovanie modelu je spustené prepínačom –t. Po spustení serveru v tomto režime, server spracuje záložky, ktoré boli zadané ako parameter za prepínačom –t. Stiahnutie zabezpečuje funkcia _Download, ktorá sa nachádza v súbore downloader.py. Po stiahnutí prebehne predspracovanie textu dokumentov. To znamená, že bude text rozdelený na tokeny a sú odstránené slová, ktoré sa nachádzajú v dokumentoch len raz. Následne sa vytvorí slovník pomocou gensim.corpora.Dictionary, slovník sa uloží do priečinku dic pre ďalšiu prácu. Keď je vytvorený slovník, tak sa uloží aj samotný korpus do priečinku Corpus. Následne sa prejde na vytváranie LDA modelu. Pre toto vytvorenie modelu je nutné vedieť počet tém, tie sú zadané prepínačom –n. Jednotlivé dokumenty prevedieme na TF pomocou funkcie doc2bow ktorá sa nachádza v knižnici gensim. Následne sa potom dokumenty prevedú na TF-IDF. Keď máme vytvorené TF-IDF dokumenty prejdeme k trénovaniu LDA modelu. Po natrénovaní modelu je nutné model uložiť pre ďalšie použitie.

Spracovanie požiadavkou je spustené prepínačom –s. Po spustení serveru v tomto režime, sa načíta slovník a potrebné modely pre vytvorenie názvu kategórií a vypočítanie sémantickej podobnosti. Server pracuje na protokole HTTP, ktorý pracuje s rôznymi dotazovacími metódami, napr. POST a GET. Tento server používa metódu POST, ktorá odosiela dáta na server a čaká na odpoveď. Server príjme dáta vo formáte JSON. Prijaté dáta obsahujú list objektov (záložiek), kde jednotlivé objekty obsahujú potrebné atribúty pre vytvorenie záložiek. Prvý objekt obsahuje informácie od klienta o počte kategórii a zvolenej metóde.

Po spracovaní dát začne server sťahovať dáta podobne ako bolo popísané pri spúšťaní s prepínačom -t. Keď sú dáta predspracované, server začne počítať sémantickú podobnosť jednotlivých záložiek. Následne zredukuje rozmer a prejde na vytváranie kategórií.



Obrázok 22 Ukážka priestoru po vytvorení kategórií. Modrou farbou sú označené stredy klastrov.

Po vytvorení kategórií priradí do objektov nový atribút „Category“ podľa ktorého sú potom vytvorené priečinky a popresúvané záložky do správnych kategórii. Pri pokuse o zaradenie jedného dokumentu do niektorej kategórie je použitá len metóda LDA. Metóda NMF nedokáže určiť tému jedného dokumentu, vždy musia byť aspoň dva.

Názov	**Funkcia**
Algo.py	Skript obsahujúci algoritmy použité pri vytvorení sémantickej podobnosti a získania kategórii
Downloader.py	Skript ktorý sťahuje a predspracuje obsah
main.py	Spúšťači skript
GetModels.sh	Skript ktorý sťahuje skip-thoughts model

Tabuľka 6 Popisujúca zdrojové súbory serveru a ich funkcie

3.4 Použité nástroje

V tejto časti popíšeme nástroje, ktoré boli použité pri implementácii systému pre kategorizovanie záložiek.

3.4.1 Python

Python[11] je interpretovaný, objektovo orientovaný, vysoko-úrovňový, dynamický programovací jazyk. Je kompatibilný so všetkými najznámejšími a najpoužívanejšími operačnými systémami, konkrétne so systémom Windows, Linux Mac OS. Python má jednoduchú a ľahko zvládnuteľnú syntax, čo prispieva k lepšej čitateľnosti kódu. Interpret jazyka Python a rozsiahla knižnica štandardných funkcií sú dostupné zadarmo a môžu byť voľne distribuované.

3.4.1.1 Gensim

Gensim[12] je voľne šíriteľná knižnica v Python licencovaná pod GNU LGPL. Gensim je rýchly a robustný nástroj, ktorý umožňuje intuitívne konštruovať rôzne sémantické modely nad typickými textovými dokumentami reprezentovanými prostredníctvom vektorového priestoru. Gensim umožňuje pracovať s dátami, ktoré presahujú množstvo dostupnej pamäte, preto je vhodný pre prácu s veľkými dátovými súbormi.

3.4.1.2 Scikit-learn

Scikit-learn[13] je knižnica v Python licencovaná pod BSD. Knižnica vznikla v roku 2007 v rámci Google Summer of Code. Knižnica obsahuje veľa implementovaných algoritmov strojového učenia a súvisiacich nástrojov. Využíva bežne dostupné knižnice pre numerické výpočty numpy a scipy, takže je pomerne nenáročný na inštaláciu.

3.4.1.3 Natural Language Toolkit

Natural Language Toolkit[14] (NLTK) je knižnica pre Python, ktorá súvisí so spracovaním prirodzeného jazyka. Obsahuje rôzne nástroje pre prácu s textom ako tokenizátory, stemmery, stop-words a parsery. Taktiež má vstavanú podporu rôznych voľne dostupných korpusov a taktiež manager pre ich sťahovanie.

3.4.1.4 BoilerPipe

BoilerPipe[15] je založený na rozhodovacích stromoch. Bol vyvinutý primárne pre extrahovanie textu článkov zo spravodajských portálov, ale je použiteľný aj na extrahovanie textu z bežných HTML stránok. HTML stránku rozdeľuje na časti vo všetkých elementoch. Časti sú následne rozdelené do tried použitím rozhodovacieho stromu. BoilerPipe je pod licenciou Apache 2.0.

[11] https://www.python.org/
[12] https://radimrehurek.com/gensim/
[13] http://scikit-learn.org/stable/
[14] http://www.nltk.org/
[15] https://boilerpipe-web.appspot.com/

3.4.2 Javascript

Javascript[16] je objektovo orientovaný skriptovací jazyk používaný najmä pri tvorbe HTML stránok. Zapisuje sa priamo do HTML kódu čo je veľká výhoda, pretože je to jednoduché. Jedná sa o klientsky skript. To znamená, že sa program odosiela so stránkou na klienta a potom sa vykonáva. Často je porovnávaný s jazykom Java. Java je samostatný jazyk a s jazykom Javascript má len podobnú syntax.

3.4.2.1 JQuery

JQuery[17] je Javascript knižnica, ktorá kladie dôraz na interakciu medzi jazykom Javascript a HTML. Knižnica obsahuje veľa rôznych funkcii ako manipuláciu s HTML elementami, spracovanie udalostí, animácie a Ajax, ktoré fungujú naprieč rôznymi prehliadačmi. JQuery je slobodný a otvorený software pod MIT licenciou.

3.4.2.2 Chrome Extension API

Chrome Extension API[18] je Javascript balík pre jednoduchšie programovanie, rozšírený pre webový prehliadač. Obsahuje rôzne funkcie pre prácu s prehliadačmi ako BrowserActions, Contextmenu, Bookmarks, Networking a ďalšie. Rozšírenia sú tvorené pomocou webových technológii ako HTML, CSS a Javascript.

[16] https://www.javascript.com/
[17] https://jquery.com/
[18] https://developer.chrome.com/extensions

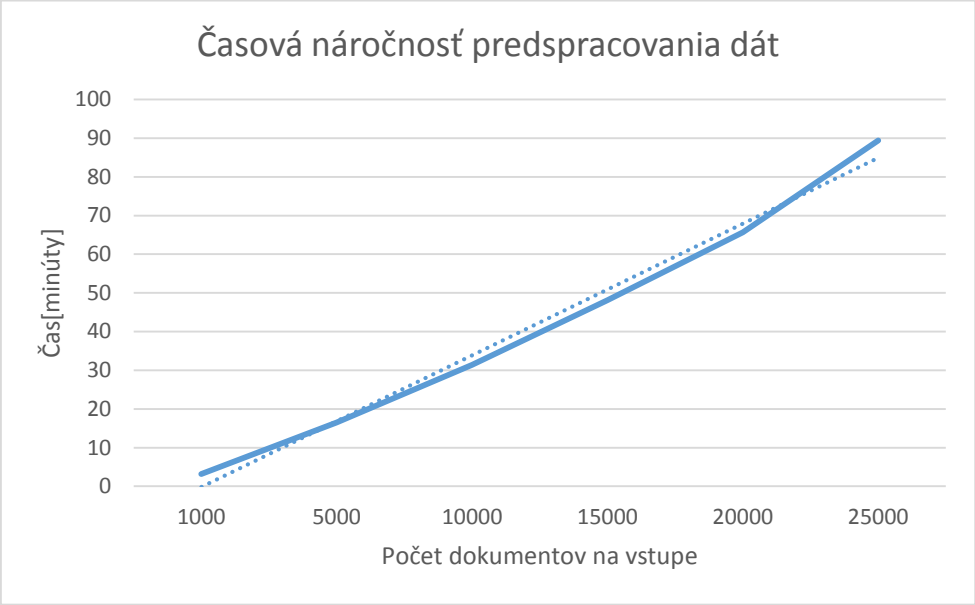
4 Testovanie a vyhodnotenie systému

4.1 Časová náročnosť jednotlivých častí systému

V nasledujúcej časti popíšeme časovú náročnosť jednotlivých časti systému podľa rôznych kritérií napr. podľa veľkosti korpusu, počtu tém alebo podľa počtu kategórii.

4.1.1 Predspracovanie dát

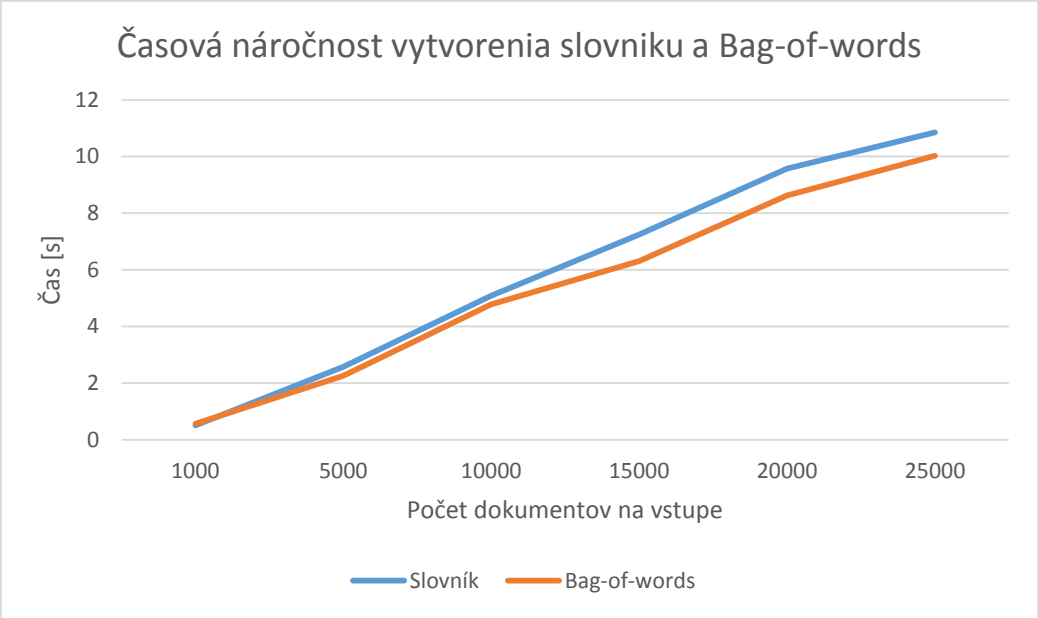
Predspracovanie dát je jedna z časovo najnáročnejších častí systému, ktorú môže ovplyvniť veľa faktorov, napr. rýchlosť sťahovania, veľkosť jednotlivých webstránok, veľkosť obsahu a ďalšie. Graf 1 znázorňuje predspracovanie dát na rôznych veľkostiach vstupného korpusu. Z grafu môžeme vidieť, že predspracovanie dát prebieha lineárne.



Graf 1 Časová náročnosť predspracovania dát

4.1.2 Vytváranie slovníku a bag-of-words

Ďalšou časťou predspracovania je vytvorenie slovníku a bag-of-words. Testovanie prebehlo na viacerých veľkostiach vstupného korpusu. Z grafu 2 môžeme vidieť, že vytvorenie slovníku a aj bag of words je časovo rovnako náročné a priebeh je lineárny.



Graf 2 Časová náročnosť vytvorenia slovníku a bag-of-words

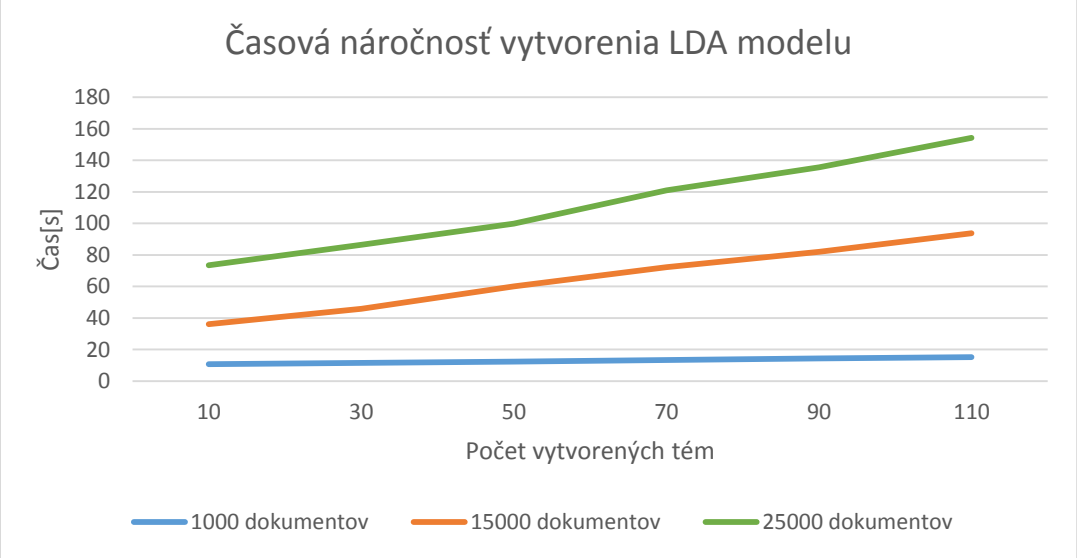
4.1.3 Vytváranie TF-IDF a LDA modelu

Časová náročnosť pri vytváraní modelov záleží na zvolených počiatočných podmienkach. U TF-IDF nie sú zadané žiadne podmienky takže sa čas môže ovplyvniť len veľkosťou korpusu. U LDA modelu môžeme zadávať rôzne podmienky, napr. počet tém. Z tabuľky 7 môžeme vidieť, že čas potrebný pre spracovanie modelu stúpa so zvyšovaním korpusu.

Počet dokumentov	1000	5000	10000	15000	20000	25000
TFIDF	1,0626	5,147	10,3739	14,7971	18,651	22,2655

Tabuľka 7 Časová náročnosť vytvorenia TF-IDF modelu

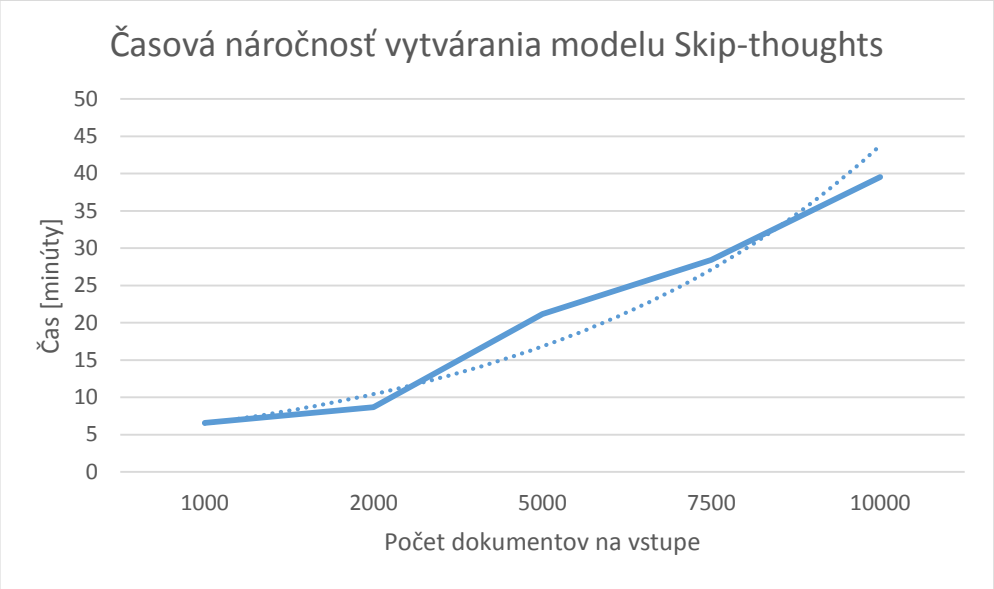
Z grafu 3 môžeme vidieť, že čas potrebný pre vytvorenie LDA modelu závisí od počtu zvolených tém a aj na počte dokumentov. Pri zvyšovaní počtu tém a počtu vstupných dokumentov časová náročnosť rýchlo stúpa.



Graf 3 Časová náročnosť vytvorenia LDA modelu

4.1.4 Vytváranie modelu Skip-Thoughts vector

Vytváranie modelu pre Skip-Thoughts vectors je časovo náročné. Testovanie teda prebiehalo len na malom korpuse a s prispôsobenými podmienkami, napr. dimenzia slov, dimenzia skrytého stavu, počet epoch atď. Zníženie dimenzii má za následok zníženie presnosti pri určovaní sémantickej podobnosti. Pre sa vo výslednom systéme pracuje s modelom od autorov. V tomto prípade bol počet dimenzií slov 50 a maximálny počet epoch 5. Z grafu 4 môžeme vidieť že časová závislosť nadobúda exponenciálny priebeh.



Graf 4Vytváranie Skip-thoughts modelu

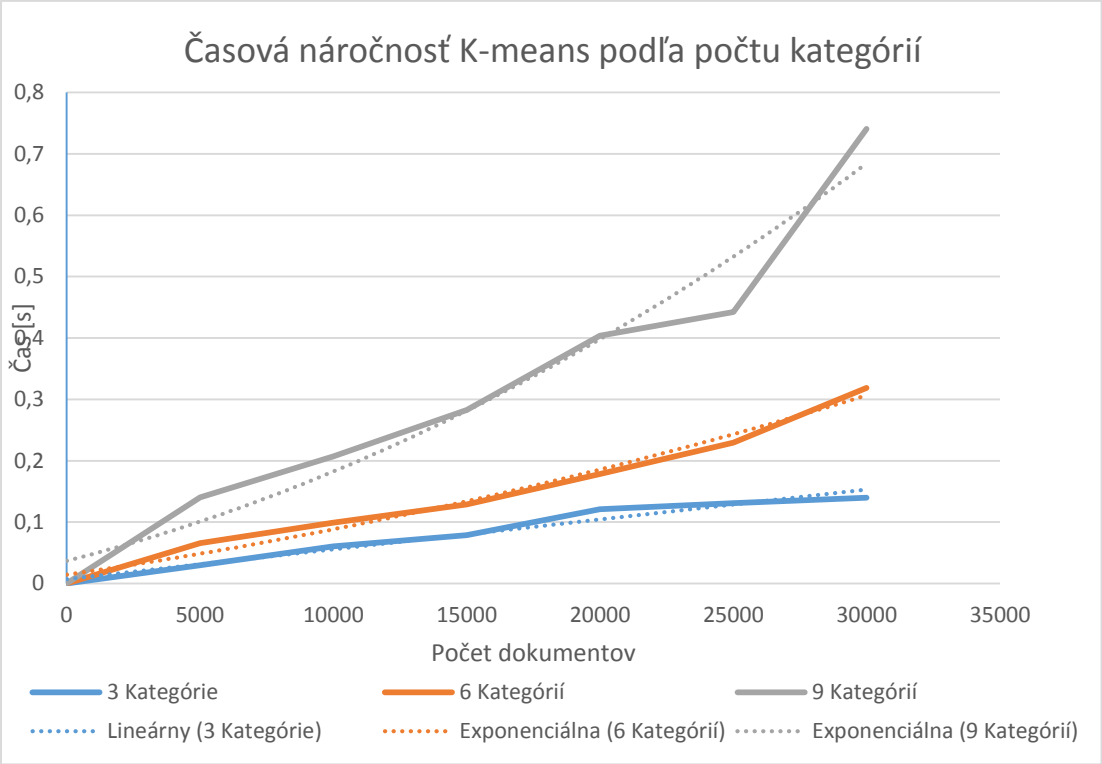
4.1.5 Vytváranie kategórií

Časová náročnosť pri vytváraní kategórií závisí od veľkosti dát a počtu kategórií. V tomto prípade boli testované dve metódy popísané v kapitole 2.4. V tabuľke môžeme vidieť čas potrebný pre nájdenie kategórií na určitej veľkosti vstupných dát. Z hodnôt môžeme vidieť, že algoritmus mean shift je oveľa náročnejší na čas a to aj pri pomerene malom vstupnom korpuse. Preto je vo výslednej aplikácii použitý algoritmus kmeans++, ktorého priebeh je v podstate lineárny.

Počet dokumentov na vstupe	500	1000	1500	2000	2500	3000
K-means [s]	0,01120	0,01413	0,02201	0,01866	0,01746	0,02390
Mean Shift [s]	0,64375	1,43201	2,54164	4,52996	5,34200	7,35981

Tabuľka 8 Porovnanie časov potrebný pre vytvorenie 3 kategórií

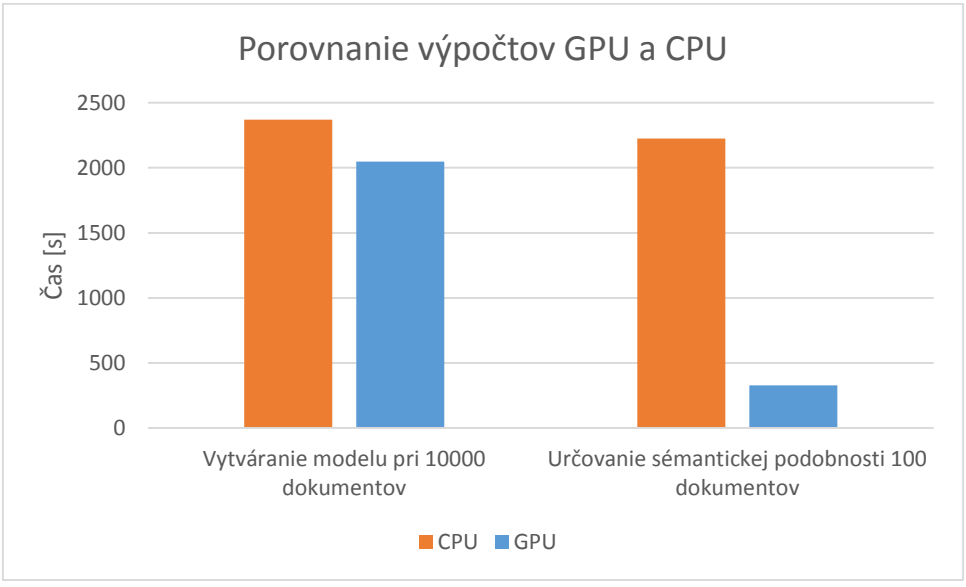
O rýchlosti kmeans rozhoduje hlavne počet zvolených klastrov. Z grafu 5 môžeme vidieť, že zvyšovanie počtu kategórií zvyšuje aj čas potrebný pre nájdenie stredových bodov. Pri nízkom počte kategórií vytvára časová závislosť lineárny priebeh, ale pri zvýšení kategórií nadobúda exponenciálny priebeh.



Graf 5 Časová náročnosť vytvárania kategórií

4.1.6 Rozdiel pri použití CPU a GPU

Vytvorenie alebo výpočet sémantickej podobnosti pomocou skip-thoughts vectors je časovo náročné. To sme si ukázali na grafe 4, kde bolo pre vytvorenie modelu použité CPU. V tomto teste poukážeme na rozdiel pri používaní CPU a GPU na výpočty. Graf 6 ukazuje, že pri vytváraní modelu je rozdiel medzi výpočtom na GPU a CPU zanedbateľný. Pri vytváraní vektorov dosiahneme použitím GPU výrazné zrýchlenie výpočtov.



Graf 6 Porovnanie výpočtov na GPU a CPU

4.2 Vyhodnotenie systému

Vyhodnotenie predspracovania dát prebiehalo na korpuse poskytnutom od vedúceho práce a pre určenie správnosti sémantickej podobnosti bol použitý mnou vytvorený korpus. Všetky testy prebiehali na počítači, ktorého technická špecifikácia je nasledovná:

- Procesor AMD FX-8320 8-jadro 3,5GHz
- Operačná pamäť DDR3 8GB 1600MHz
- Grafická karta NVIDIA GeForce GTX660M

4.2.1 Predspracovanie dát

Predspracovanie dát je jednou z najdôležitejších častí, pretože rozhoduje o tom koľko dokumentov bude usporiadaných a koľko nie. Pri tomto teste bol použitý korpus rozdelený do viacerých častí s rôznymi veľkosťami. Každý korpus je tvorený inou množinou dokumentov aby sa neopakovali dokumenty, ktoré nie je možné stiahnuť.

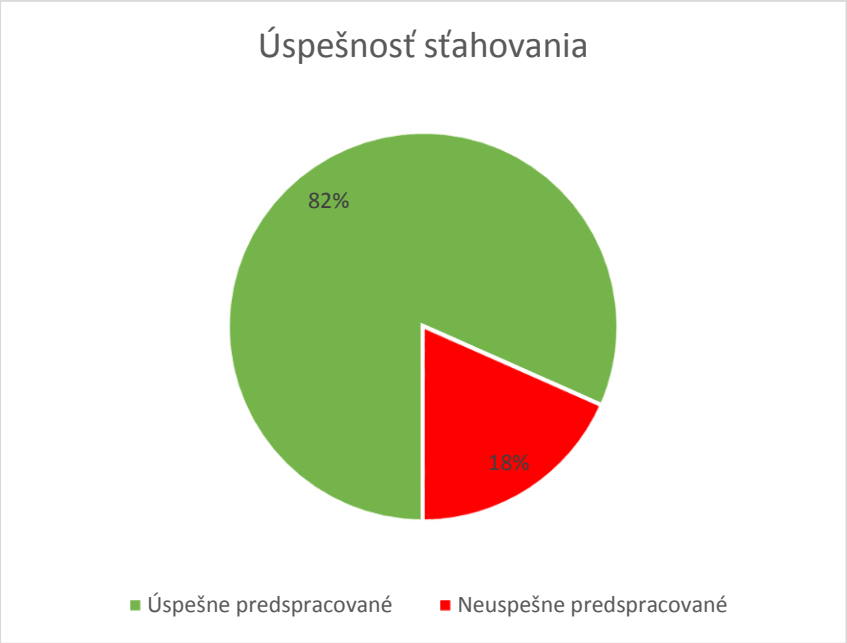
Z tabuľky 9 môžeme vidieť, že úspešnosť sťahovania stránok je značne vysoká oproti neúspešne stiahnutým. Neúspešnosť stiahnutia môže byť zapríčinená týmito dôvodmi:

- Stránka už neexistuje alebo sa vyskytla chyba pri jej otváraní.
- Obsah stránky je generovaný pomocou skriptov. To zapríčiňuje, že po predspracovaní obsahu dostaneme prázdny dokument.
- Server nemá oprávnenie na zobrazenie stránky alebo k prístupu na doménu.

Test	Korpus	Úspešné	Neúspešné
1	1000	859	141
2	856	719	137
3	1256	1074	182
4	2756	2359	397
5	756	616	140
6	1025	831	194
7	1425	1028	397
8	738	544	194
9	3538	2992	546
10	1538	1135	403
Priemer	1488,8	1215,7	273,1

Tabuľka 9 Ukážka úspešnosti predspracovania na rôznych korpusoch

Graf 6 ukazuje percentuálnu úspešnosť sťahovania. V grafe sú použité hodnoty z tabuľky 9, ktorej hodnoty sú spriemerované.



Graf 7 Úspešnosť sťahovania v percentách

4.2.2 Sémantická podobnosť a vytváranie kategórií

Pri vyhodnotení sémantickej podobnosti a vytváraní kategórií bolo použitých viacero metód:

- Homogenita
- Úplnosť
- V-measure
- Adjusted rand index

Pre výpočet týchto metód musia byť použité dva korpusy. Prvý korpus je takzvaný „tréningoví“ a ukazuje ako by mal vyzerať výsledok. Druhý je vytvorený pomocou metód pre určovanie sémantickej podobnosti a zhlukovanie dokumentov. Testovanie prebiehalo na mnou vytvorených korpusoch obsahujúce rôzne kategórie, napr. cestovanie, technológie, hudbu atď.

Homogenita

Homogenita nadobúda hodnôt 0 a 1. Výsledok zhlukovania je rovnorodý, keď všetky prvky jednej triedy ležia v jednom zhluku. Nezáleží na poradí prvkov vo vektore výsledku zhlukovania.

Úplnosť

Úplnosť nadobúda hodnoty od 0 do 1. Výsledok je úplný, pokiaľ sú zhluknuté všetky prvky a zároveň sú zhluky totožné s triedami.

V-measure

V-measure je harmonický priemer medzi homogenitou a úplnosťou, ktorý vypočítame nasledujúcou rovnicou:

$$v = 2 * \frac{(homogenita * úplnosť)}{(homogenita + úplnosť)} \tag{2.16}$$

Nadobúda hodnoty od 0 po 1, kde 1 predstavuje zhodné kategórie.

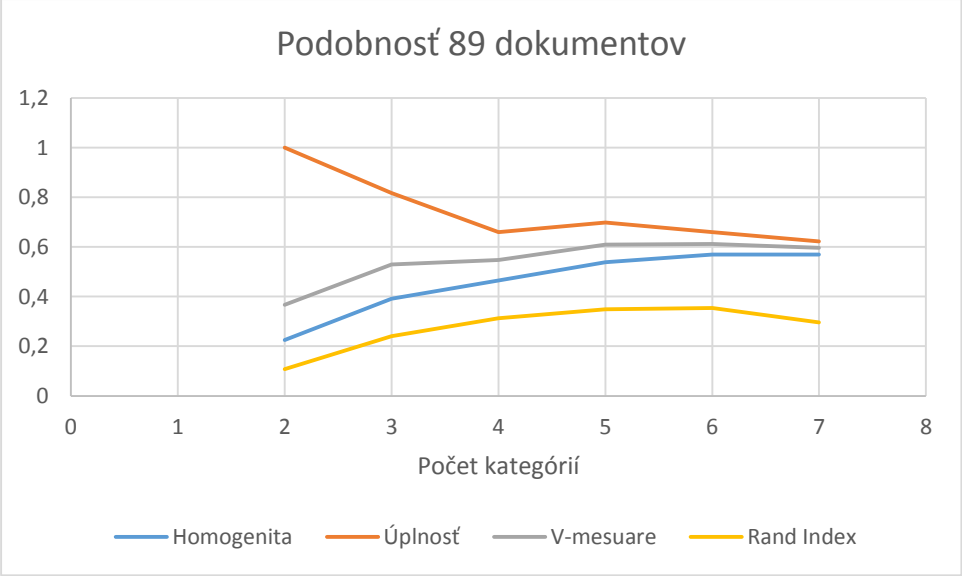
Adjusted rand index

Vypočítava mieru podobnosti medzi dvomi kategóriami tak, že zvažuje všetky dvojice vzoriek a počíta páry, ktoré sú v rovnakých alebo v rôznych skupinách vo vytvorenej a skutočnej kategórii.

$$ARI = \frac{RI - Predpokladaný\_RI}{(\max(RI) - Predpokladaný\_RI)} \tag{2.16}$$

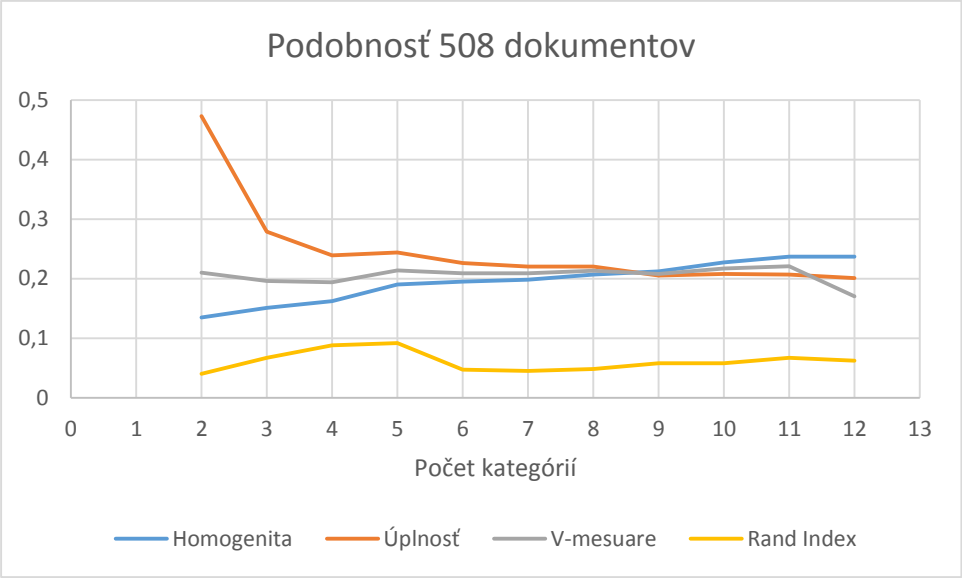
Kde *RI* je miera podobnosti medzi dvomi kategóriami. Nadobúda dvoch hodnôt -1 a 1, kde 1 určuje, že kategórie sú identické a -1, že kategórie sú rozdielne. Hodnoty okolo 0 naznačujú, že sa kategórie prekrývajú.

Nasledovný experiment bol vytvorený na korpuse s počtom dokumentov 89, ktoré obsahujú rôzne témy. Korpus obsahuje 7 tried s ktorými porovnám výsledné roztriedenie na rôznom počte kategórií. Na grafe 8 môžeme vidieť, že pri menšom počte kategórií bola úplnosť vytvorených kategórií 1. To znamená, že kategórie boli úplné, čiže sú vytvorené kategórie totožné s triedou. Homogenita vyšla pomerne nízko pri dvoch kategóriách. Keďže sa porovnáva zo 7 triedami nikdy nebudú prvky jednej triedy ležať v jednom zhluku. Najlepšie výsledky dostávame až od 5 kategórie, kde môžeme vidieť podobné hodnoty homogenity a úplnosti.



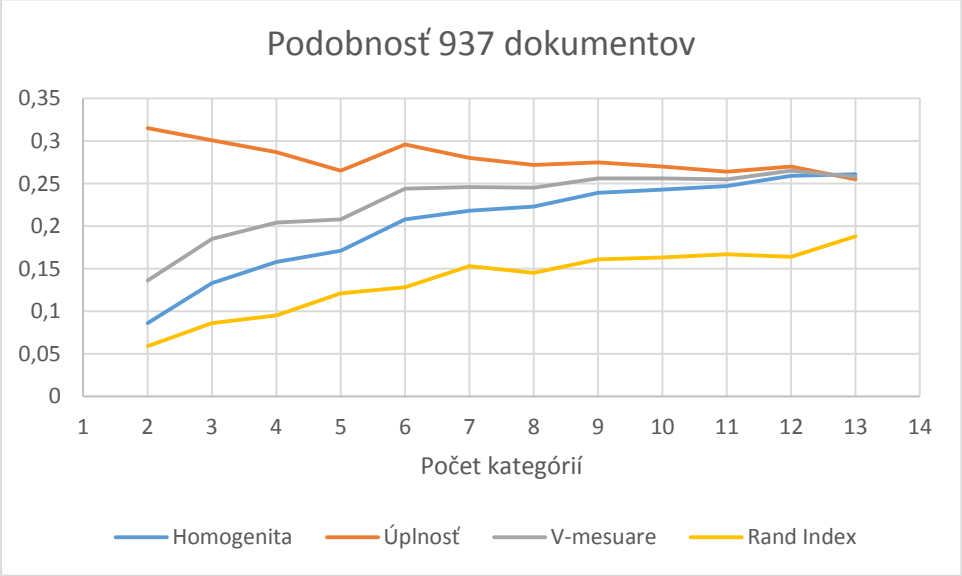
Graf 8 Experiment na 89 dokumentoch

Ďalší experiment bol vytvorený na korpuse s 508 dokumentami. Korpus obsahuje 12 tried z ktorými porovnávame výsledné rozdelenie. Z grafu 9 môžeme vidieť, že zvýšením počtu dokumentov znížime presnosť určenia kategórií. Je to dané tým, že zvýšenie počtu podobných dokumentov môže spôsobovať prekrývanie zhlukov. Najlepší výsledky kategórií dostávame od 6 kategórií. Hodnoty sú však veľmi nízke, takže vo výsledných kategóriách budú dokumenty z rôznych tried.



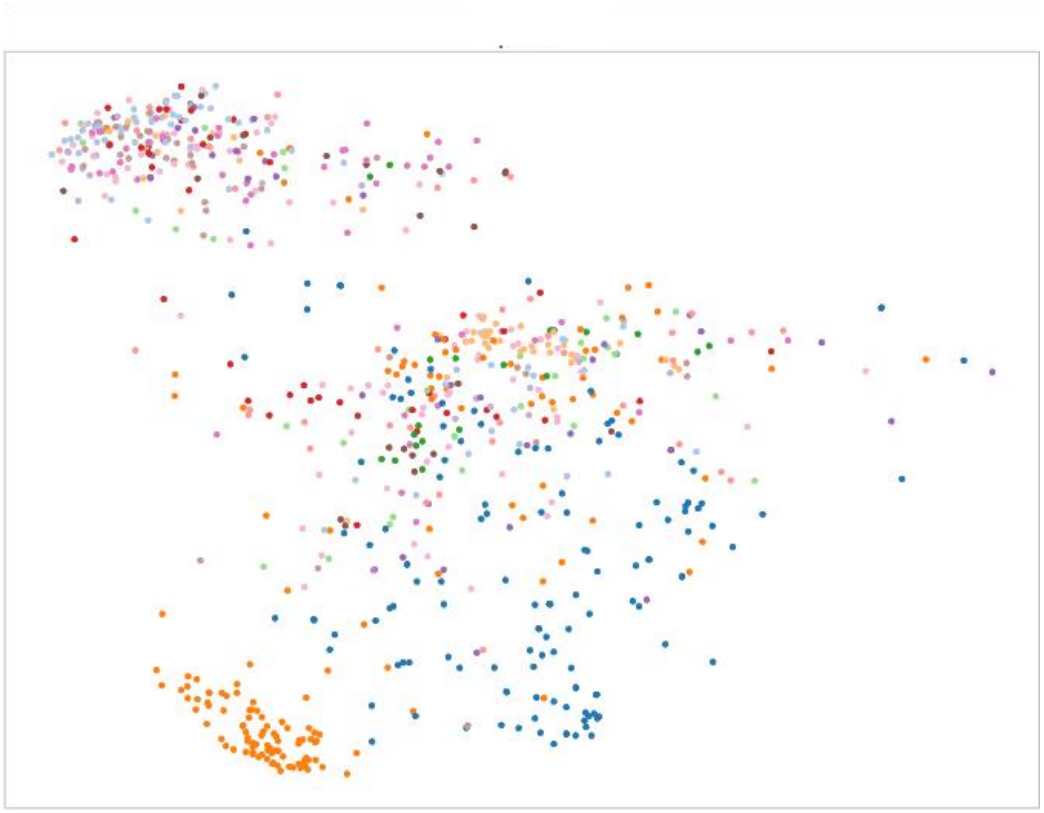
Graf 9 Experiment na 508 dokumentoch

Nasledujúci experiment bol vytvorený na korpuse o 937 dokumentoch. Korpus je rozdelený do 13 tried s ktorými budeme porovnávať výsledné rozdelenie. Na grafe 10 môžeme vidieť, že pri použití väčšieho počtu kategórií dosahujeme lepšej zhody kategórií. V tomto prípade nastáva chyba vo vytváraní zhlukov. Niektoré triedy napr. šport sú rozdelené do menších tried hokej, tennis a futbal. Tieto triedy sú veľmi podobné, čiže zvýšením počtu kategórií dostávame prekrývajúce sa zhluky.



Graf 10 Experiment na 937 dokumentoch

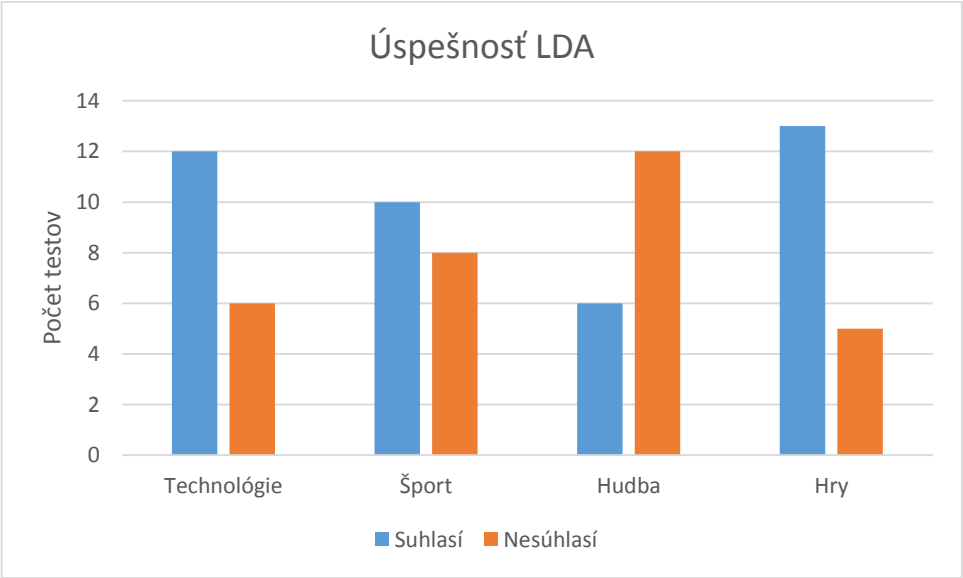
Na obrázku 13 môžeme vidieť rozloženie 937 dokumentov podľa ich sémantickej podobnosti. Farby dokumentov označujú ich triedu, ktorých je v tomto prípade 13 a jedná sa o triedy použité pri experimentoch. Všimnime si, že jednotlivé triedy sa prekrývajú. To ale nemusí znamenať, že je zle určená sémantická podobnosť. Niektoré triedy sú z rovnakých oblastí. To nám ale znemožňuje určenie presného zhluku dokumentov z rovnakých tried.



Obrázok 13 Ukážka rozloženia 937 dokumentov v 13 triedach

4.2.3 Názvy kategórií

Pre vyhodnotenie názvu kategórií boli vytvorené 2 experimenty na danom korpuse. V prvom experimente pracujeme s určovaním názvu na jednom dokumente. V druhom experimente vyhodnotíme použitie metód na získanie názvu viacerých dokumentov.



Graf 11Ukážka určovania tém LDA na roznych dokumentoch

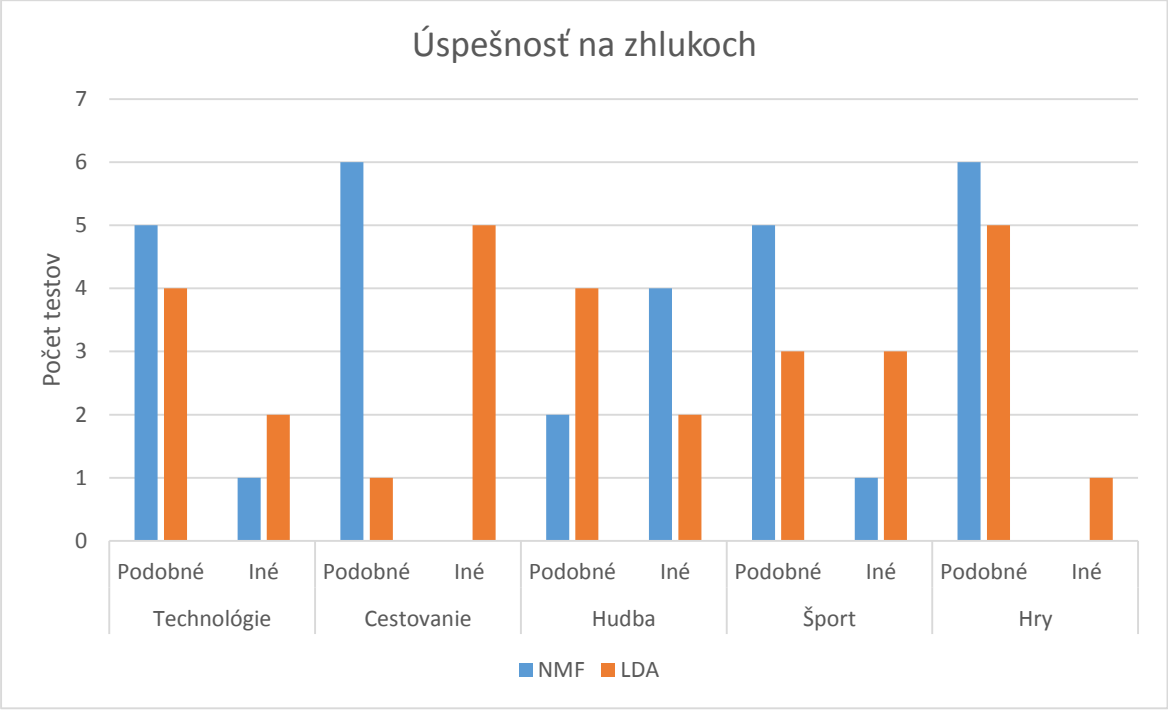
Prvý experiment prebiehal na menšom počte dokumentov z rôznych kategórií. Testovalo sa hlavne vyhľadávanie správnej témy jedného dokumentu. Z Grafu 11 môžeme vidieť, že určovanie tém v niektorých kategóriách je pomerne nízky. To môže byť zapríčinené tým, že model, ktorý bol použitý pre vytvorenie tém obsahuje viacej informácií z oblasti technológií. Najhorší výsledok bol získaný v kategórii hudba, kde počet správne nájdených kategórií bol 6. To môže byť spôsobené tým, že model pracuje len s anglickými výrazmi a väčšina dokumentov bola písaná po slovensky alebo česky.

Druhý experiment bol vytvorený na väčšom počte dokumentov z rovnakej kategórie. Celkovo bolo urobených 6 testov pre každú kategóriu. Zo získaných hodnôt, ktoré môžeme vidieť v tabuľke 10 môžeme povedať, že metóda NMF je vhodnejšia pre určovanie tém. Avšak zlé výsledky LDA môžu byť spôsobené korpusom na ktorom bolo LDA trénované.

	Technológie		Cestovanie		Hudba		Šport		Hry	
Metóda	Podobné	Iné	Podobné	Iné	Podobné	Iné	Podobné	Iné	Podobné	Iné
NMF	5	1	6	0	2	4	5	1	6	0
LDA	4	2	1	5	4	2	3	3	5	1

Tabuľka 10 Experiment na viacerých dokumentoch

Na grafe 12 môžeme vidieť, že LDA malo najlepšie výsledky v kategórii technológie a hry. Zlepšenie môžeme pozorovať na hudbe, kde pri viacerých dokumentoch s rôznymi jazykmi dosahujeme vyššej úspešnosti. U metódy NMF máme v niektorých kategóriách malú mieru neúspešnosti. Najvyššiu neúspešnosť má u hudby, kde je to spôsobené rôznymi jazykmi stránok.



Graf 12 Úspešnosť vytvárania tém zhlukov

5 Záver

Cieľom tejto bakalárskej práce bolo zoznámiť sa s metódami sémantickej podobnosti textov a vytvoriť systém, ktorý by dokázal zoradiť a navrhnúť pomenovania tém pre webové dokumenty v tomto prípade záložky. Používatelia s veľkým množstvom neštrukturalizovaných dokumentov by tak mali nástroj, pre jednoduché roztriedenie a pomenovanie kategórií.

Tento systém mal byť navrhnutý ako klient-server aplikácia, kde by bola aplikácia pre klienta vytvorená ako doplnok, optimalizovaný pre viaceré prehliadače. Úlohou bolo zoznámiť sa s problematikou pri určovaní sémantickej podobnosti textov a určovaním pomenovania pre webové dokumenty. Na základe spracovaných informácií vytvoriť a implementovať tento systém.

Samotný systém bol následne vyhodnotení pomocou experimentov na menších korpusoch, na ktorých som poukázal na výhody a nevýhody použitých metód. V týchto experimentoch sa ukázalo, že je lepšie používať metódu NMF pri určovaní názvov kategórií, pokiaľ nemáme k dispozícii kvalitný LDA model.

Výsledkom je systém na báze klient-server aplikácie, ktorá dokáže usporiadať veľké množstvo webových dokumentov v tomto prípade záložiek obsiahnutých v prehliadači. Klient taktiež umožňuje editáciu a vytváranie nových záložiek, taktiež umožňuje vyhľadávanie medzi záložkami.

Vďaka tejto práci som zistil princíp fungovania a implementácie metód používaných pri analýze a spracovávaní textov. Zároveň som si prehĺbil programovacie skúsenosti v skriptovacích jazykoch Python a JavaScript.

V tejto bakalárskej práci je možne naďalej pokračovať a tým ju vylepšiť. Vypracovať presnejší systém určovania názvov kategórií, alebo vytvoriť databázový systém pre synchronizáciu záložiek na viacerých počítačoch.

Literatúra

[1] GORRELL, Genevieve. *Generalized Hebbian Algorithm for Dimensionality Reduction in Natural Language Processing*. Linkoping University, 2006. Dizertačná práca. Dostupné z: http://liu.diva-portal.org/smash/get/diva2:22830/FULLTEXT01

[2] MATERNA, Jiří. *Sémantická analýza textů* In:Seznam.cz [online]. 2011-08-30 [cit. 2016-05-02]. Dostupné z: http://vyhledavani.sblog.cz/2011/08/30/semanticka-analyza-textu-1/

[3] MANNING, Christopher D., *Introduction to information retrieval*. Cambridge: Cambridge university press, 2008. ISBN: 978-0-521-86571-5.

[4] KIROS R.,ZHU Y.,SALAKHUTDINOV R.,ZEMEL R. S.,TORRALBA A.,URTASUN R., and FIDLER S.. *Skipthought vectors* [online]. NIPS, 2015. [cit. 2016-05-02]. Dostupné z: http://arxiv.org/pdf/1506.06726v1.pdf

[5] LESKOVEC, Jure. RAJARAMAN, Anand. ULLMAN, Jeffrey David. *Mining of massive datasets*. Cambridge University Press, 2014. ISBN: 978-1-107-01535-7

[6] ARTHUR, David. VASSILVITSKII, Sergei. K-*means++: The Advantages of Careful Seeding* [online]. Stanford, 2006-06-06 [cit. 2016-05-03]. Dostupné z: http://ilpubs.stanford.edu:8090/778/

[7] *K-means Clustering* In:VLFeat.org [online]. [cit. 2016-05-05]. Dostupné z: http://www.vlfeat.org/overview/kmeans.html

[8] GREENE, Derek. *NMF for topic modeling*[online]. 2015-03-27 [cit. 2016-05-06]. Dostupné z: http://derekgreene.com/nmf-topic

[9] BLEI, David M., NG, Andrew Y. a JORDAN, Michael I. *Latent Dirichlet Allocation. Journal of Machine Learning Research 3*[Online]. 2003 [cit. 2016-05-06] Dostupné z: https://www.cs.princeton.edu/~blei/papers/BleiNgJordan2003.pdf.

[10] DERPANIS, Konstantinos G. *Mean shift clustering*. 2005 [cit. 2016-05-06]. Dostupné z: http://www.cse.yorku.ca/~kosta/CompVis_Notes/mean_shift.pdf.

[11] COMANICIU, D., MEER, P. (2002). *Mean shift: A robust approach toward feature space analysis*. IEEE Transactions on Pattern Analysis and Machine Intelligence. 24(5). 603–619

[12] GHODSI, Ali. *Dimensionality Reduction A Short Tutorial*[online]. University of Waterloo. 2006 [cit. 2016-05-13]. Dostupné z: http://www.stat.washington.edu/courses/stat539/spring14/Resources/tutorial_nonlin-dim-red.pdf

[13] RICHARDSON, Mark. *Principal Component Analysis* [online]. 2009 [cit. 2016-05-13]. Dostupné z: http://www.sdss.jhu.edu/~szalay/class/2015/SignalProcPCA.pdf

[14] KAVRAKI, Lydia E. *Dimensionality Reduction Methods for Molecular Motion* [online]. [cit. 2016-05-13]. Dostupné z: https://cnx.org/contents/9cMfjngH@6.3:Av9d0v4w@10/Dimensionality-Reduction-Metho

[15] KUANG, Da. CHOO, Jaegul. PARK, Haesun. *Nonnegative matrix factorization for interactive topic modeling and document clustering*. In: *Partitional Clustering Algorithms*. Springer International Publishing, 2015. p. 215-243.

[16] CHEN, *Kuan-Yu Menphis. Latent Dirichlet Allocation* [online]. 2007 [cit. 2016-05-13]. Dostupné z: http://acsweb.ucsd.edu/~yuw176/report/lda.pdf

[17] OLAH, Christopher. *Deep Learning, NLP, and Representations* [online]. 2014-07-07 [cit. 2016-05-17]. Dostupné z: http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/#fn2

Prílohy

Príloha 1. Obsah priloženého CD:

- /Addon – Obsahuje inštalačné a zdrojové súbory doplnku
- /Server – Obsahuje zdrojové súbory serverovej aplikácie
- /návod – Obsahuje manuál a dokumentáciu aplikácie