VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

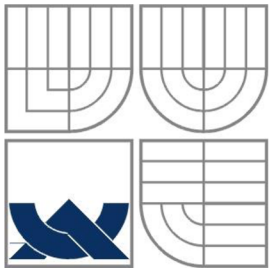
SÉMANTICKÁ PODOBNOST TEXTŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

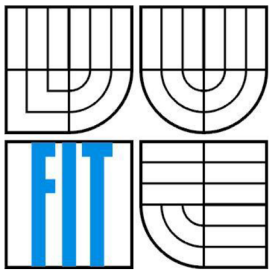
AUTOR PRÁCE
AUTHOR

VÁCLAV BRADÁČ

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SÉMANTICKÁ PODOBNOST TEXTŮ

SEMANTIC SIMILARITY OF TEXTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VÁCLAV BRADÁČ

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. RNDr. Pavel Smrž, Ph.D.

BRNO 2015

Abstrakt

Tato práce se zabývá problematikou určování sémantické podobnosti textů se zaměřením na škálovatelnost. Součástí zpracování je teoretický přehled nástrojů pro implementaci systému na testovaných datech. Testovaný korpus obsahuje odborné články v anglickém jazyce. Cílem práce je tyto články analyzovat, modifikovat pro snadnější analýzu jejich sémantické obdoby. Jedním z nejdůležitějších využitých nástrojů je reprezentace dat ve vektorovém prostoru.

Abstract

This paper deals with the determination of semantic similarity texts, focusing on scalability. Part of treatment is a theoretical overview of the tools to implement the system on test data. Tested corpus contains expert articles in the English language. The aim is to analyze these articles, modified to facilitate the analysis of their semantic analogues. One of the most utilized tools is a representation of data in a vector space model.

Klíčová slova

Sémantická podobnost, TF-IDF, Latentní sémantická analýza, Latentní sémantická indexace, Singulární rozklad, Latentní Direchletova alokace, Python, Gensim, PHP, Elasticsearch, MoreLikeThis

Keywords

Semantic similarity, TF-IDF, Latent semantic analysis, Latent semantic indexing, Singular value decomposition, Latent Direchletova allocation, Python, Gensim, PHP, Elasticsearch, MoreLikeThis

Citace

Bradáč Václav: Sémantická podobnost textů, bakalářská práce, Brno, FIT VUT v Brně, 2015

Sémantická podobnost textů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Doc. RNDr. Pavel Smrž, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Václav Bradáč
20. května 2015

Poděkování

Rád bych poděkoval vedoucímu práce panu Doc. RNDr. Pavlu Smržovi, Ph.D. za jeho odborné vedení a trpělivou pomoc.

Obsah

1 Úvod

Pod sémantickou analýzou textů si můžeme představit rozčlenění textových dokumentů do tzv. témat, které budou sloužit k určení sémantické podobnosti dokumentů. Pod těmito tématy se rozumí například pro dokumenty obsahující výuku na základní škole označení matematika, český jazyk a chemie. Zatím nám tyto analýzy vždy nezaručují, aby označení témat bylo vždy výstižné, co do obsahu jejich lidského vyjádření. Hlavním cílem této analýzy je pochopení spojitostí mezi jednotlivými dokumenty.

V bakalářské práci jsou charakterizovány tři různé metody pro určování sémantické podobnosti textů, které byly využity při návrhu a implementaci výsledného systému. Data pro zkoumání analogie textů obsahují články v anglickém jazyce z různých konferencí a workshopů. Vytvoření podobnostního modelu se řadí mezi nejvíce časově náročný proces systému.

Z dalších technologií, které byly využity při implementaci výsledného systému, stojí za zmínku fulltextový vyhledávač. Ten zpracovává řetězcové dílčí úlohy vedoucí k vyhledávání na základě dotazu ve vytvořené databázi.

Vyhodnocení systému bylo uskutečněno pomocí menšího testovacího korpusu, v kterém se vyskytovaly články ze stejného workshopu, a pomocí nich byla určena referenční data pro testování.

Nedílnou součástí bakalářské práce bylo vytvoření dotazníku. Dotazník byl poskytnut sto osmdesáti čtyřem respondentům. Zjištěné odpovědi dotazovaných jsou chápany jako lidský úsudek pro porovnání zkoumaných metod.

2 Teoretická část

2.1 Určování sémantické podobnosti

Existuje mnoho modelů pro usnadnění zjišťování sémantické podobnosti textových dokumentů. V následujících kapitolách si uvedeme ty, které byly použity pro tvorbu finální verze aplikace. Každý z nich zhodnotíme a uvedeme si jejich klady a zápory.

2.1.1 Booleovský

Booleovskou koncepcí zjišťující podobnost dokumentů je ohodnocení nalezených termů v textu logickou *1* pokud byl nalezen, a logickou *0* pokud nebyl. Negativem použité koncepce v praxi je, že výskyt termu v textu určuje pouze relativní podobnost. To znamená, že se v dokumentu hledaný text nalézá, ale nemusí souviset v kontextu vyhledávání. Booleovský model se využívá v dotazovacím jazyku SQL.

2.1.2 Vektorový

Další ze zkoumaných modelů je vektorový. Ten ohodnotí jejich podobnost ve vymezeném intervalu od 0 do 1. O tomto prototypu se dozvíme více v kapitole 2.2.

2.1.3 Strukturální

Strukturální předloha využívá pro vyhodnocení podobnosti vektorový model nebo současně i strukturu dokumentu. Strukturu dokumentu využívá tak, že pokud jsou dokumenty stejně formátované, můžeme si při zpracování dokumentu určit jaká část dokumentu je nejdůležitější nebo která je bezvýznamná a podle toho určíme vektorům jejich váhu.

2.2 Vektorový model (VSM)

Nejpoužívanějším modelem pro reprezentaci textových dokumentů v oblasti získávání informací je vektorový model. Dokumenty jsou zastoupeny jako body v n-rozměrném prostoru. Dimenze v tomto prostoru představují jedinečná slova korpusu, se kterým pracujeme. Nyní pomocí slov v dokumentu lze určit jeho vektor a to tak, že složky vektoru, který dokument obsahuje, bude mít nenulovou hodnotu. Zjištěný fakt může být chápán jako jeho nedostatek, jelikož většina hodnot vektoru bude vždy nulová. Výhodou vektorového modelu je bezesporu jeho jednoduchost. [1]

2.2.1 Motivace

Nejzákladnějším požadavkem na vektorový prostor je to, že bez ohledu na metrickou vzdálenost je vybaven modelem lidského úsudku pro zjištění sémantické podobnosti. Sémantická podobnost je chápána tak, že sekvence slov, které jsou stejného významu, jsou v prostoru blízko u sebe, a naprosto nesouvisející by měli být vzdálenější. [1]

2.2.2 Ukázka vektorového prostoru

Jak už jsme si dříve popsali, každý dokument je v prostoru rozčleněn na jednotlivá slova. Nyní si na následujících dokumentech vysvětlíme, jak lze zjistit jejich vektory v prostoru. Pro jednoduchost budeme předpokládat dokumenty o velikosti jedné věty.

#	Dokument
1.	The girl have the dog and cat
2.	The girl have the phone
3.	The cat is missing
4.	The dog eats sausage

Tabulka 1 - Ukázkové dokumenty

Nejprve si vytvoříme seznam všech jedinečných slov obsažených v dokumentech.

1. „and“
2. „cat“
3. „dog“
4. „eats“
5. „girl“
6. „have“
7. „is“
8. „missing“
9. „phone“
10. „sausage“
11. „the“

Nyní si podle slov v prostoru ohodnotíme jednotlivé dokumenty podle četnosti výskytu slov v textu. Takové ohodnocení se nazývá TF (*term frequency*).

#	Vektory dokumentů – TF bez normalizace										
	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
	and	cat	dog	eats	girl	have	is	missing	phone	sausage	the
1.	1	1	1	0	1	1	0	0	0	0	2
2.	0	0	0	0	1	1	0	0	1	0	2
3.	0	1	0	0	0	0	1	1	0	0	1
4.	0	0	1	1	0	0	0	0	0	1	1

Tabulka 2 - Vektory ukázkových dokumentů

Většinou se výpočet TF normalizuje vydělením počtem nejčastějšího výskytu slova v dokumentu, aby se nenadhodnocovali dlouhé dokumenty na úkor krátkých. Pro výpočet se tedy použije vztah:

Rovnice:

$$TF_{(s,d)} = \frac{počet_{(s,d)}}{počet_{(max,d)}; max \in d} \quad (2.1.)$$

kde značí $počet_{(s,d)}$ množství výskytu slova s v dokumentu d a max je slovo s největším počet výskytu v dokumentu d.

#	Vektory dokumentů – TF s normalizací										
	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
	and	cat	dog	eats	girl	have	is	missing	phone	sausage	the
1.	0.5	0.5	0.5	0	0.5	0.5	0	0	0	0	1
2.	0	0	0	0	0.5	0.5	0	0	0.5	0	1
3.	0	1	0	0	0	0	1	1	0	0	1
4.	0	0	1	1	0	0	0	0	0	1	1

Tabulka 3 - Vektory dokumentů s využitím vzorce pro TF

2.2.3 TF-IDF

Nevýhodu klasifikace pouze pomocí počtu výskytu slova v textu je chápán tak, že nastavuje velkou váhu i slovům, které nejsou pro podobnost dokumentu důležitá. Zvýhodnit budeme chtít pouze slova, která mají v textu velký výskyt a nejsou často obsaženy v ostatních dokumentech. Pro tuto problematiku bylo vytvořeno matematické řešení IDF (*Inverse Dokument Frequency*), které se vypočítá pomocí tohoto vztahu:

Rovnice:
$$IDF_{(i)} = \log(\frac{N}{DF_{(i)}}) \quad (2.2.)$$

Výsledek IDF se rovná logaritmu počtu dokumentů N děleno počtem dokumentů s výskytem slova DF.

Rovnice:
$$TF - IDF_{(s,d)} = TF_{(s,d)} * IDF_{(s)} \quad (2.3.)$$

#	**Vektory dokumentů – TF-IDF**										
	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
	and	cat	dog	eats	girl	have	is	missing	phone	sausage	the
1.	0.301	0.151	0.151	0	0.151	0.5	0	0	0	0	0
2.	0	0	0	0	0.151	0.5	0	0	0.301	0	0
3.	0	0.301	0	0	0	0	0.602	0.602	0	0	0
4.	0	0	0.301	0.602	0	0	0	0	0	0.602	0

Tabulka 4 - Vektory pro dokumenty TF-IDF

2.2.4 Latentní sémantická analýza (LSA)

Jedním z možných nástrojů usnadňující vyhledávání podobných souborů ve velkém množství dimenzí se nazývá Latentní sémantická analýza, dále jen LSA. Někdy je také nazývána LSI (Latentní sémantická indexace). LSA je založena na statistice a lineární algebře. Tato analýza funguje na principu automatického shlukování slov do témat. Pod těmito tématy si můžeme představit rozdělení článků dle kategorií, kterými se článek zabývá. Například jsou zařazené do skupin v odvětví sportu, literatury nebo informačních technologií. Jedno konkrétní slovo se může objevit i ve více tématech, ale s odlišným významem. Jako příklad použijeme slovo *list*, které by se mohlo zařadit jak do shluku o zahradnictví, tak i o kancelářských potřebách. Význam slov se určí pomocí předpokladu, že slova podobného významu se objevují v podobných textech. [2]

U výše uvedené metody je třeba určit vhodný počet těchto skupin, aby nebyly obsáhlejší, než je nutné nebo naopak moc konkrétní. Vhodný počet skupin se dá určit podle toho, kolik různých námětů lze v korpusu najít. Například jedná-li se výhradně o témata z odvětví informačních technologiích, tak bude stačit korpus rozčlenit na počet témat v řádu stovek. Pokud se bude jednat o témata z více odvětví, jako jsou sport, lékařství a strojírenství, bude zapotřebí až ztrojnásobit jejich množství pro jejich lepší zařazení do skupin. LSA pracuje na principu, že slova obsažená ve více dokumentech společně jsou si sémanticky podobné. Tato sémantická podoba je ve více úrovních. Pro zjištění takto blízkých slov se využívá matematická metoda nazývaná singulární rozklad.

2.2.4.1 Singulární rozklad (SVD - Singular Value Decomposition)

SVD je jedna z matematických metod aproximace matic. Za použití redukce jsou nezměněny shluky obdobných dokumentů. Touto metodou lze určit podobnost skupin i ve více úrovních podobnosti jak přes přímou podobnost, tak i nepřímou, a to pomocí výskytu slov v kontextu v jiných dokumentech [3].

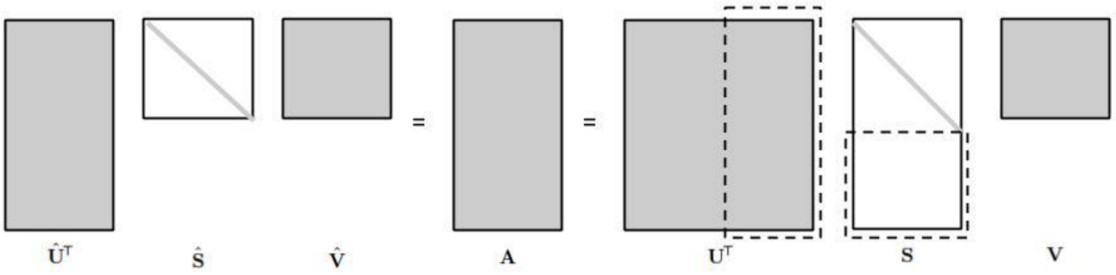
Nechť A je libovolná matice reálných čísel v rozsahu m x n, pak existuje ortogonální matice U n x n, která reprezentuje matici termů, V m x m, která zastupuje matici dokumentů a diagonální matice S. Na jejichž diagonále jsou vlastní čísla matice $\sqrt{A^T A}$ (pro $m \geq n$) nebo $\sqrt{AA^T}$ (pro $m \leq n$) tak, že vztah u rovnice 2.4 se nazývá singulární rozklad a vlastní čísla matice $\sqrt{A^T A}$ jsou označeny jako singulární čísla matice A.

Rovnice: $$A = U^T S V \tag{2.4.}$$

Hodnost redukované matice je dána množstvím nenulových diagonálních prvků matice A.

Je matematický dokázané, že při vlastnosti m > n a násobení $U^T S$ je zbytečné znát posledních m - n sloupců matice U^T [4]. Jak by taková redukce vypadala, je zobrazena v rovnici 2.5 a na Obrázek 1.

Rovnice: $$A = \hat{U}^T \hat{S} \hat{V} \quad (2.5.)$$



Obrázek 1 - Redukce SVD při rozměrech matice m>n

2.2.5 Pravděpodobnostní latentní sémantická analýza (pLSA)

Pravděpodobnostní latentní sémantická analýza řeší stejný problém jako LSA, ale na rozdíl od ní je založena na pravděpodobnosti a statistice.

Nyní si teoreticky popíšeme princip pLSA [5] [6]. Předpokládejme, že chceme náš korpus roztřídit do *k* témat a tématem *i* je spojeno rozdělení pravděpodobnosti přes všechny slova *w*, která lze nalézt v množině dokumentů. Jedná se tedy o funkci, která pro téma *i* přiřazuje jeho pravděpodobnost a dále bude značena $P_{(w|i)}$. Dále se domnívejme, že funkci $P_{(i|D)}$, která udává pravděpodobnost tématu *i* v dokumentu D. Pomocí zmíněných funkcí by se dala pravděpodobnost výskytu slova vyjádřit takto:

Rovnice:

$$P(w|D) = \sum_{i=i}^{k} P(w|i)P(i|D) \tag{2.6.}$$

Pravděpodobnost dokumentu za předpokladu, že každé slovo dokumentu je generováno z jednoho daného tématu a slova v něm jsou navzájem nezávislá, lze vyjádřit vztahem:

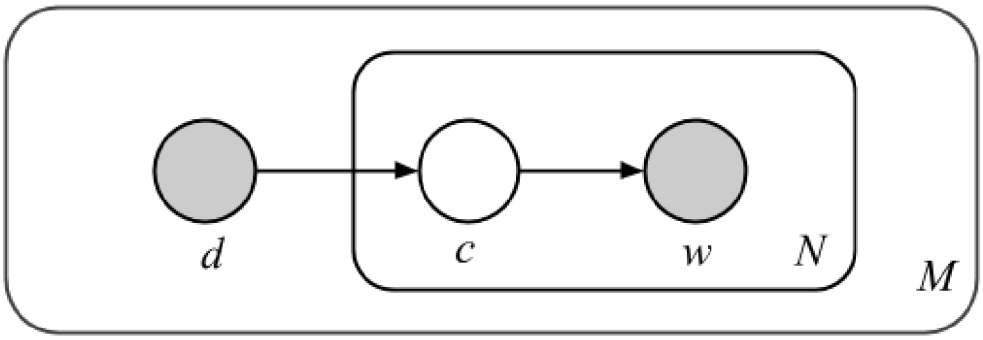
Rovnice:

$$P_{(D)} = \prod_{w} P(w|D)^{c(w,D)} \tag{2.7.}$$

kde, *c(w, D)* značí počet výskytu slova *w* v dokumentu *D*. Díky tomu, že je logaritmus monotónní funkce, lze celý výraz zjednodušit do tvaru:

Rovnice:

$$\log(P(D)) = \sum_{w} c(w, D) \log(P(w|D)) \tag{2.8.}$$

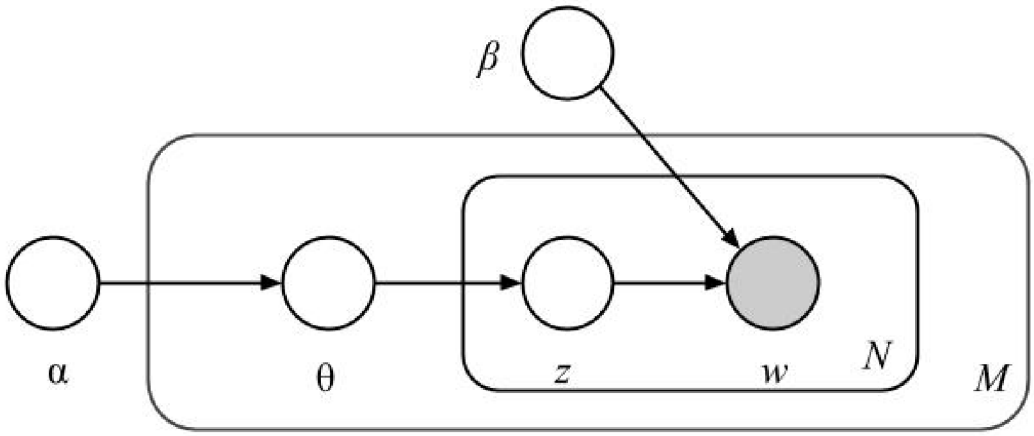


Obrázek 2 - plate notace pLSA, kde N je pozice slova a M je dokument, zbytek dle označení v popisu

Tato analýza není obsažena ve výsledném systému, slouží pouze pro lepší vysvětlení LDA.

2.2.6 Latentní Dirichletova alokace (LDA)

Latentní Dirichletova alokace je schopná vytvářet pravděpodobnostní model z množiny dokumentů. Základní myšlenkou je, že dokumenty jsou reprezentovány jako náhodné kombinace nad latentními tématy, kde každé téma je charakterizováno jako distribuce slov. Oproti pLSA, ze kterého LDA vychází, už nemá sklony k přeučování. Tento problém je zde vyřešen využitím předpokladu, že témata mají Dirichletovo pravděpodobnostní rozdělení. [7]



Obrázek 3 - plate notace LDA kde N je pozice slova a M je dokument, zbytek dle označení v popisu

Předpokládejme, že chceme korpus rozdělit do *k* témat. Pak by generativní proces vypadal takto:

1. Pro každý dokument *d* z množiny všech dokumentů se určí distribuce pravděpodobnosti nad tématy na základě multinomického rozdělení s θ_j z Dirichletova rozdělení s parametrem α (vektor o *k* reálných číslech menší než 1), který je společný pro celý korpus.
2. Pro každé slovo *t* dokumentu *d*
    a. Zvolme téma $z_{p,j}$, které náleží do {1, …, k} na základě hodnot témat θ_j – multinomického rozdělení s parametrem θ_j
    b. Zvolme slovo $w_{p,j}$ z $p(w_{p,j}|z_{p,j}, \beta)$ – toto rozdělení má také parametry generované z Direchetova rozložení, tentokrát je ale využit jako parametr hyperparametr β.

Pro výpočet úplné podobnosti tohoto modelu lze využít vztah:

Rovnice:

$$p(w, z, \theta, \beta|\alpha, \eta) = \prod_{i=1}^{k} p(\beta_i|\eta) \prod_{j=1}^{n} p(\theta_j|\alpha) \prod_{t=1}^{|d_j|} p(z_{p,j}|\theta_j) p(w_{p,j}|\beta_{z_{p,j}}) \qquad (2.9.)$$

kde $|d_j|$ značí délku dokumentu j ve slovech a α (resp. η) jsou prioritní funkce na množinu témat pro dokument. Pro odhad skrytých proměnných LDA se využívá variační algoritmus EM (Expectation Maximization) nebo Gibbsovo vzorkování [8] [9].

2.2.7 Nástroje pro redukci slov v prostoru

Jak už jsme si sdělili v předešlé části práce, každé unikátní slovo v prostoru má svoji dimenzi. Proto před uložením termů do prostoru se budeme snažit, aby jeho rozměry byli co nejmenší a k tomu lze využít tyto nástroje:

- ***case-folding*** – slouží k převodu všech velkých písmen v textu na malá. Například pokud máme text „POST post“, tak by VSM obsahoval ve slovníku, jak slovo „POST“, tak i „post“. S využitím nástroje by obsahoval jen „post“ s počtem výskytu v textu 2.
- ***stopwords*** – pod tímto názvem si můžeme představit slova, která pro určení podobnosti nebudeme potřebovat a to v angličtině budou např. *a, and, of, the, for, in, to* a další. Tento prostředek by nám v ukázkovém prostoru (Tabulka 1) snížil počet dimenzí z 11 na 9.
- ***lemmatizace*** – nahrazuje slova tak, že k původním vyhledá jejich základní tvar. Jako příklad si můžeme uvést slovo *počítačových*, pro které se vyhledá základní tvar *počítač*. Určitým úskalím při jeho využití mohou být slova s více významy, kterých v českém jazyce nalezneme mnoho, jako například výraz *tancích*. Nalezený slovní základ může být jak tank, tak i tanec.
- ***stemmatizace*** – zjišťuje kmen slova pomocí toho, že ze slov odstraní předpony a přípony. Na ukázku si uvedeme anglické slovo *development*, kde by jeho kmen byl *develop*.

2.3 Podobnost vektorů

V této kapitole je shrnuto jak zjistit míru podobnosti článků. Pokud jsme již zjistili hodnoty vektorů v prostoru, tak dokážeme určit i podobnost jakéhokoliv vyhledávacího termu. To zjistíme tak, že jej zobrazíme ve vytvořeném prostoru. Nejpodobnějším textem se rozumí takový, který se nachází nejblíže k dotazovanému vektoru. Na tuto problematiku lze použít standardní matematické postupy. Nejznámější jsou shrnuty v podkapitolách 2.3.1 a 2.3.2.

2.3.1 Euklidovská vzdálenost

Jako nejjednodušší matematický postup pro výpočet vzdáleností mezi vektory se nazývá Euklidovská vzdálenost. Výsledná blízkost se vypočítá jako vzdálenost koncových bodů vektorů. [1]

Rovnice:

$$S_{(D1,D2)} = \sqrt{\sum_{i=1}^{N}(D1_i - D2_i)} \tag{2.10.}$$

kde N je počet hodnot vektorů a D1, D2 označují dokumenty.

Nyní si uvedeme příklad výpočtu na dokumentech D1 a D2 z ukázkového prostoru uvedeného v tabulce 4.

Příklad:

$$\begin{aligned} S_{(D1,D2)} &= \sqrt{(D1_0 - D2_0) + \cdots + (D1_{11} - D2_{11})} \\ &= \sqrt{(0.301 - 0) + \cdots + (0 - 0)} = 0.302 \end{aligned} \tag{1.1.}$$

2.3.2 Kosinova podobnost

Výsledkem Kosinovy podobnosti je vzdálenost, která se získá výpočtem svírajícího úhlu mezi dvěma vektory. [1]

Rovnice:

$$podobnost = \cos(\Theta) = \frac{D1 * D2}{||D1|| * ||D2||} = \frac{\sum_{i=1}^{N} D1_{i*}D2_i}{\sqrt{\sum_{i=1}^{N}(D1_i)^2 * \sum_{i=1}^{N}(D2_i)^2}} \tag{2.11.}$$

, kde N je velikost vektoru

Pomocí rovnice (2.11.) zjistíme podobnost dokumentů D1 a D2 z Tabulka 4, vyobrazené na straně 7.

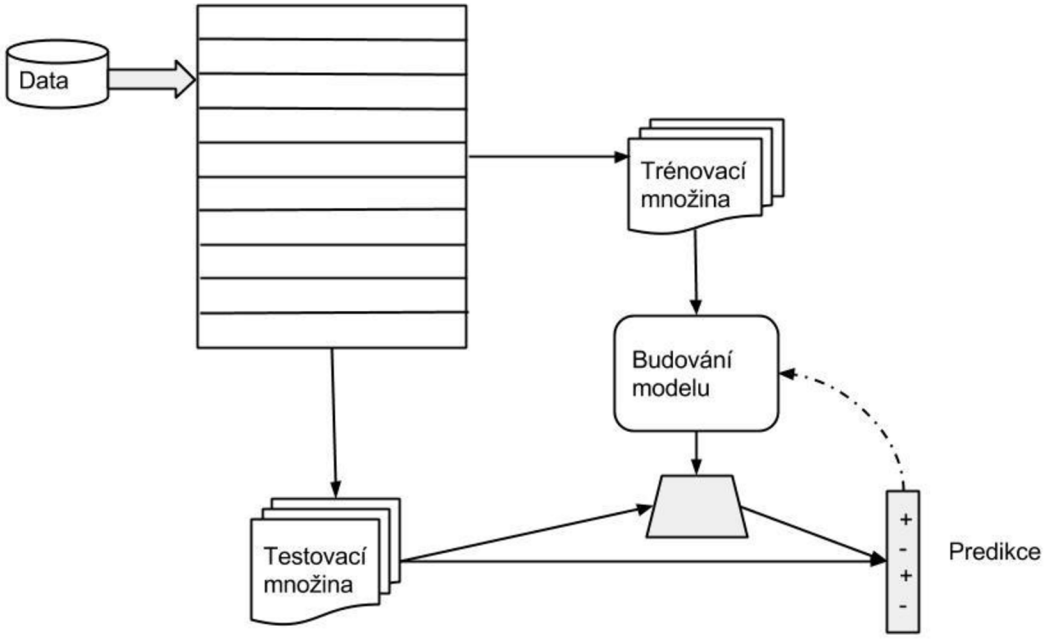
Příklad:

$$\begin{aligned} podobnost_{(D1,D2)} &= \frac{(D1_1 * D2_1) + \cdots + (D1_{11} * D2_{11})}{\sqrt{\left({D1_1}^2 + \cdots + {D1_{11}}^2\right) * \left({D2_1}^2 + \cdots + {D2_{11}}^2\right)}} \\ &= \frac{(0.301 * 0) + \cdots + (0 * 0)}{\sqrt{(0.301^2 + \cdots + 0^2) * (0^2 + \cdots + 0^2)}} = 0.708 \end{aligned} \tag{1.2.}$$

Kosinová podobnost má definovaný výsledek z intervalu od <-1,1>. Vektory představují počet opakovaných slov v textu tak nikdy nebude záporná. Proto, aby byl výsledek podobnosti z intervalu <0,1>, se výsledek odečte od 1. Výsledná podobnost se bude rovnat 1-0.708=0.292.

2.4 Křížová validace

Tato metoda zjišťuje, do jaké míry bude vytvořený model ovlivňovat nové nezávislé vzorky dat. Principem křížové validace je, že vstupní data jsou rozdělena do několika stejně velkých množin. Zde vždy jedna množina bude sloužit jako testovací data a ostatní jako data určená pro trénování. Tento úkon se opakuje pokaždé s jinou podmnožinou tvořící trénovací a testovací množinu. [10].



Obrázek 4 - znázornění křížové validace

3 Praktická část

3.1 Použité nástroje

V této části bakalářské práce si popíšeme použité nástroje pro implementaci systému.

3.1.1 Elasticsearch

Elasticsearch (nyní už jen Elastic[1]) je fulltextový vyhledávač, který je založen na Apache Lucene. Je vyvíjen v jazyce Java. Lze se na něj dotazovat pomocí webového rozhraní. Hlavní výhodou je jeho rychlost a využití REST API, které slouží k jednoduchému vytváření, čtení, editovaní nebo mazaní informací ze serveru pomocí jednoduchých HTTP volání. Elasticsearch ukládá indexy do bezešvé databáze. Tím je chápáno to, že se nemusí dopředu definovat, jak mají být indexy uloženy.

3.1.2 Python

Python[2] je dynamicky interpretovaný objektově orientovaný jazyk. Lze jej zařadit mezi více paradigmatické jazyky. Tím se rozumí, že se nemusí nutně používat jen objektově orientovaný vzor programování, ale je možné použít i funkcionální a procedurální prototyp. Ve standardu obsahuje velké množství knihoven, které je možné průběžně doplňovat novými. Pro rychlejší vyhodnocování příkazů, jsou některé výkonnostně kritické knihovny implementovány v jazyce C.

3.1.2.1 Gensim

Mezi nejvíce využívaný nástroj pro práci s informacemi uloženými ve vektorovém prostoru v jazyce Python se nazývá Gensim[3], který byl vytvořen jako open-source. Je zaměřen na automatickou analýzu rozsáhlých textových dokumentů v přirozeném jazyce. Mezi hlavní výhody patří jeho rychlost, kterou získává při analýze výpočtem matematických operací s využitím knihoven napsaných v jazyce C. V jeho útrobách lze analyzovat text pomocí TF-IDF, LSA a LDA.

3.1.2.2 Regulární výrazy

Pod regulárním výrazem si představíme řetězec v regulárním jazyce popisující vzor, jak má vyhledávaná nebo editovatelná množina znaků vypadat. Pro využívání celé síly regulárních výrazů je třeba v jazyce Python naimportovat knihovnu *re*.

[1] https://www.elastic.co/
[2] https://www.python.org/
[3] https://radimrehurek.com/gensim/

3.1.2.3 Elasticsearch

V jazyce Python existuje oficiální Elasticsearch klient pro správu indexů uložených na definovaném serveru. Jedná se o velmi tenký obal okolo Elasticsearch REST API.

3.1.3 PHP

PHP[4] je skriptovací programovací jazyk. Mezi jeho hlavní využití patří dynamické vytváření webových stránek a webových aplikací například ve formátu HTML. Dá se užít i pro tvorbu desktopových aplikací, při využití jeho kompilované verze. Při použití na webové aplikace se vyhodnocení skriptu vykonává na straně serveru.

3.1.3.1 Nette

Mezi nejznámější framework u nás pro PHP bezpochyby patří Nette[5]. Jedná se o český open-sourcový projekt. Jeho hlavní devízou je odstranění bezpečnostních rizik, podpora AJAX, MVC a znovupoužitelný kód. Disponuje také ladícím nástrojem TRACY, který usnadňuje určování chyb v kódu. Využitá verze 2.3.2.

3.1.3.2 Elastica

Pro PHP, stejně jako pro Python, existuje Elasticsearch klient. V bakalářské práci je použit nástroj Elastica[6] řadící se rovněž mezi open-source.

[4] http://php.net/
[5] http://nette.org/
[6] https://github.com/ruflin/Elastica

3.2 Implementace

3.2.1 Analýza a čtení dat ze souboru

Jedním z prvních úkonů při tvorbě systému je zjistit nad jakými daty bude aplikace pracovat. Následuje krátký popis, jak a kde budeme textová data extrahovat.

3.2.1.1 Analýza

Zvolená data pro zjišťování podobnosti byla získána od skupiny ACL Anthology Network interface z Michiganské univerzity Clair[7], který byl autory zveřejněn na svých webových stránkách pro možné další testování sémantické podobnosti textů. Data obsahují adresář „*papers_text*“ a „release“. V první zmíněné složce nalezneme přes dvacet tisíc textových souboru, ve kterých se nachází výhradně písemná část odborných článků v anglickém jazyce. Druhý adresář obsahuje jednotlivé vydání dle roku a zjištěných informací z článků. Pro maximální využití systému byla zvolena verze z roku 2013, která patří ve složce k poslednímu vydání. V již zmíněném závěrečném vydání použijeme dva soubory *acl.txt*, který obsahuje informace o tom, kam vybraný článek cituje, a *acl_metadata.txt* ve kterém se objevují informace k jednotlivým článkům jako unikátní označení, název, jméno autorů, místo konání konference a rok vydání.

3.2.1.2 Načtení dat

V následující části si popíšeme jakým způsobem zpracovat data z testovaných souborů uvedených v analýze. Tuto problematiku má ve své kompetenci třída *classParseFile*, která je obsažena v souboru *ParseData.py*. Nejprve extrahujeme data ze souboru *acl-metadata.txt*. Náhled na formátování souboru je vyobrazen na Obrázek 5.

```
id = {D10-1014}
author = {Huang, Zhongqiang; Cmejrek, Martin; Zhou, Bowen}
title = {Soft Syntactic Constraints for Hierarchical Phrase-Based
Translation Using Latent Syntactic Distributions}
venue = {EMNLP}
year = {2010}

id = {D10-1015}
author = {Luong, Minh-Thang; Nakov, Preslav; Kan, Min-Yen}
title = {A Hybrid Morpheme-Word Representation for Machine Translation of
Morphologically Rich Languages}
venue = {EMNLP}
year = {2010}

id = {D10-1016}
author = {Genzel, Dmitriy; Uszkoreit, Jakob; Och, Franz Josef}
title = {"Poetic" Statistical Machine Translation: Rhyme and Meter}
venue = {EMNLP}
year = {2010}
```

Obrázek 5 - Náhled formátu souboru "acl-metadata.txt"

[7] http://clair.eecs.umich.edu/aan/

Existuje několik možných přístupů k rozčlenění formátovaných informací z textu použitím konečného automatu nebo regulárních výrazů. Nejvhodnějším u zvoleného dokumentu bude použití regulárních výrazů, kvůli zobrazení se stále opakujících vzorů textu. Jak by takový zápis v regulárním jazyce mohl vypadat lze spatřit na Obrázek 6.

```
id = \{([^\}]+)\}\s+author = \{([^\}]+)\}\s+title = \{([^\}]+)\}\s+venue = \{([^\}]+)\}\s+year = \{([^\}]+)\}
```

- \{ - znak(„{“)
- \} – znak(„}“)
- ( ) - značí tzv. group při vyhledávání
- [^\}]+ - 1 až více jakýchkoliv znaků vše kromě („{“)
- \s+ - 1 až více bílých znaků

Obrázek 6 - Ukázkový regulární výraz

Jakmile už známe meta-data o článcích, tak bude nejvhodnější také zjistit, na které práce z celkové množiny dat autoři článků citují. K tomuto účelu využijeme soubor *acl.txt*, který zahrnuje výpis unikátních identifikačních termů k této problematice, kde ACL id na levé straně operátoru "==>" označuje, ze kterého článku se cituje, a na pravé straně citovaný článek. Náhled k souboru je vyobrazen na Obrázek 7.

```
D09-1141 ==> A00-1002
D12-1027 ==> A00-1002
E06-1047 ==> A00-1002
H05-1110 ==> A00-1002
N01-1020 ==> A00-1002
N13-1036 ==> A00-1002
P13-2001 ==> A00-1002
P13-2073 ==> A00-1002
W11-2602 ==> A00-1002
W13-2702 ==> A00-1002
C10-1054 ==> A00-1004
W06-1008 ==> A00-1004
W03-0704 ==> A00-1005
```

Obrázek 7 - Náhled na formát souboru "acl.txt"

Pro extrahování informací, použijeme jednoduchý regulární výraz.

K nejdůležitějším informacím potřebným určení sémantické podobnosti je samotný text článků, který se nachází v adresáři *papers_text*. V této složce nalezneme textové soubory s názvy podle jejich unikátního označení. Pro vyhodnocení sémantické podobnosti pomocí anotace článku si z textu vyextrahujeme celou textovou část s abstraktem.

3.2.2 Vektorový prostor

Zpracování a uložení textových dat do vektorového prostoru vykonává třída *classLoadData* obsažená v souboru *LoadData.py*. Tato třída nejprve převede celý text na malá písmena a poté rozdělí text na jednotlivá slova. Následně tato slova převede do základního tvaru pomocí stemmeru obsaženého

v knihovně pro jazyk Python *nltk*. Ve zmíněné knihovně nalezneme seznam *stopwords* slov, které je nutné z výsledného pole odstranit.

3.2.2.1 Vyhodnocení počtu výskytu slov v textech

Pro vyhodnocení četnosti slov obsažených v textech byl vytvořen slovník, kde jako jméno klíče je samotné slovo a hodnota obsahuje četnost jeho výskytu. Dalším krokem bude odstranění slov, které se v celém korpusu dokumentů vyskytují pouze jednou. Takové slovo nám není nápomocno pro určení podobnosti vůči jiným dokumentům, pokud slovo není obsaženo v jiných dokumentech. Tato získaná data pro vektorový prostor si uložíme do slovníku, pomocí kterého si vytvoříme TF-IDF model korpusu.

3.2.2.2 Vyhodnocení dat pomocí LSA a LDA

Pomocí funkcí v knihovně gensim si vytvoříme modely pro vyhodnocení LSA a LDA. Tyto modely slouží k rozčlenění souborů do tematických skupin. U modelů lze zvolit počet výsledných témat.

3.2.3 Vytváření indexu

Na vytváření indexů je nejdříve třeba zadat parametry pro Elasticsearch server. Tyto informace lze nastavit v souboru *config.xml*. Ve zmíněném souboru je nutné vyplnit adresu, port serveru, název indexu a přihlašovací údaje. Správa indexu se provádí v souboru *CreateIndex.py*. Tento soubor obsahuje třídu *classCreateIndex,* která vlastní funkci *functionCreateIndex()* pro vytvoření indexu.

3.2.3.1 Mapování indexu

Pod mapováním indexu si můžeme představit definování, výběr typu dané části pole indexu, filtry a analyzátory k využití. V indexu každého článku definovaného typu *doc* nalezneme pole:

- *„id_“* - jedinečnou identifikaci článku
- *„title“* - název článku
- *„author“* - jména autorů
- *„venue“* - název místa konání konference nebo workshopů kde byl článek zveřejněn
- *„year“* - rok konání
- *„abstract“* - textová část článku obsahující abstrakt
- *„references“* - id článků na které je citováno a jsou obsaženy v testované množině dokumentů
- *„text“* - textová část článku bez abstraktu a referencí
- *„topics_lda“, “topics_lsi“*- označení tří témat, kterým článek vyhovuje dle modelu

Příklad mapování polí *„abstract“* a *„text“, lze spatřit v* Obrázek 8.

```
mapping_doc = {
        "doc" : {
          "properties" : {
                "text" : {
                  "type" : "string",
                  "analyzer" : "snowball",
                  "language" : "English" ,
                  'tokenizer' : 'standard',
                  'filter' : 'lowercase'
                },
                "abstract" :{
                  "type" : "string",
                  "analyzer" : "snowball",
                  "language" : "English",
                  'tokenizer' : 'standard',
                  'filter' : 'lowercase'
                }
          }
        }
    }
```

Obrázek 8 - Ukázka mapování polí indexu

3.2.3.2 Nahrávání dat

Nahrávání dat o dokumentech je uskutečněno pomocí *bulk API*, která umožňuje provádět více operací nad indexy pomocí jednoho volání, což nám značně může zvýšit rychlost.

3.2.4 Ovládání terminálové aplikace pro předzpracování

Aplikace umožňuje ovládání pomocí spustitelného souboru *Makefile*, který lze jednoduše spustit pomocí vyvolání příkazu *make* v terminálu. Za předpokladu, že terminál je spuštěn ze stejného adresáře, kde se *Makefile* nachází. Tento příkaz vyvolá sekvenci úkonů od načtení dat ze souboru až po nahrání celé studie na server s *Elasticsearch*. Při zadání parametru *help* u příkazu *make* se vypíše nápověda, kde lze vyhledat popis dalších parametrů, které spustitelný soubor nabízí.

3.2.4.1 Popis aplikace

Terminálová aplikace pro předzpracování je rozčleněna do pěti částí. V první části se pomocí dat ze souborů vytvoří slovník *doc.dict*, který bude dále k dispozici v pracovním adresáři. Ve druhé úseku se vytvoří *tf-idf* model dle vytvořeného korpusu, který se také uloží do pracovního adresáře pro další využití. V dalších dvou pasážích se vytvoří modely pro LSA a LDA. Ty se opět zaznamenají pro využití v budoucnu. V poslední sekci aplikace se umístí zjištěná data na server, ze kterého pak čerpá informace webové grafické rozhraní.

#	Jméno souboru	Funkce
1.	Config.xml	konfigurační soubor
2.	CreateIndex.py	veškerá komunikace s Elasticsearch
3.	LoadConfig.py	načtení konfigurace klienta do třídy
4.	LoadData.py	vytvoření slovníku
5.	Main.py	řídící skript, vytváření korpusu, LSA a LDA modelu
6.	Makefile	spouštěcí soubor
7.	ParseData.py	načtení dat ze souboru

Tabulka 5 - seznam vlastních zdrojových souborů s popisem funkce u konzolové aplikace

3.2.5 Grafické rozhraní

Grafické uživatelské rozhraní slouží, aby se lidé co nejrychleji naučili s aplikací pracovat a nemuseli u toho umět všechny příkazy, které jsou nahrazeny grafickými prvky. Pro náš výzkum bylo zvoleno webové grafické rozhraní, které nepotřebuje žádnou instalaci u uživatelů. Naše webové rozhraní je vytvořeno s pomocí skriptovacího jazyku Php s využitím *frameworku Nette*. Konfigurační informace pro server s Elasticsearch se nastavuji v souborech *HomepagePresenter.php* a *SearchPresenter.php* na okomentovaném místě.

#	Jméno souboru	Adresářová cesta	Funkce
1.	HomepagePresenter.php	/bp/app/presenters	spravuje latte šablony default a venue
2.	SearchPresenter.php	/bp/app/presenters	spravuje latte šablony show a help
3.	default.latte	/bp/app/templatesHomepage	šablona na úvodní stránky
4.	venue.latte	/bp/app/templates/Homepage	šablona pro zobrazení míst konání konferencí, které jsou obsaženy na webu
5.	show.latte	/bp/app/templates /Search	šablona pro zobrazení konkrétního článku a jemu podobných článků dle volby v liště
6.	help.latte	/bp/app/templates/Search	šablona pro možné úpravy nepřesných informací.
7.	@layout.latte	/bp/app/templates	šablona pro záhlaví a zápatí stránky

Tabulka 6 - - seznam vlastních nebo upravených zdrojových souborů s popisem funkce u webové aplikace

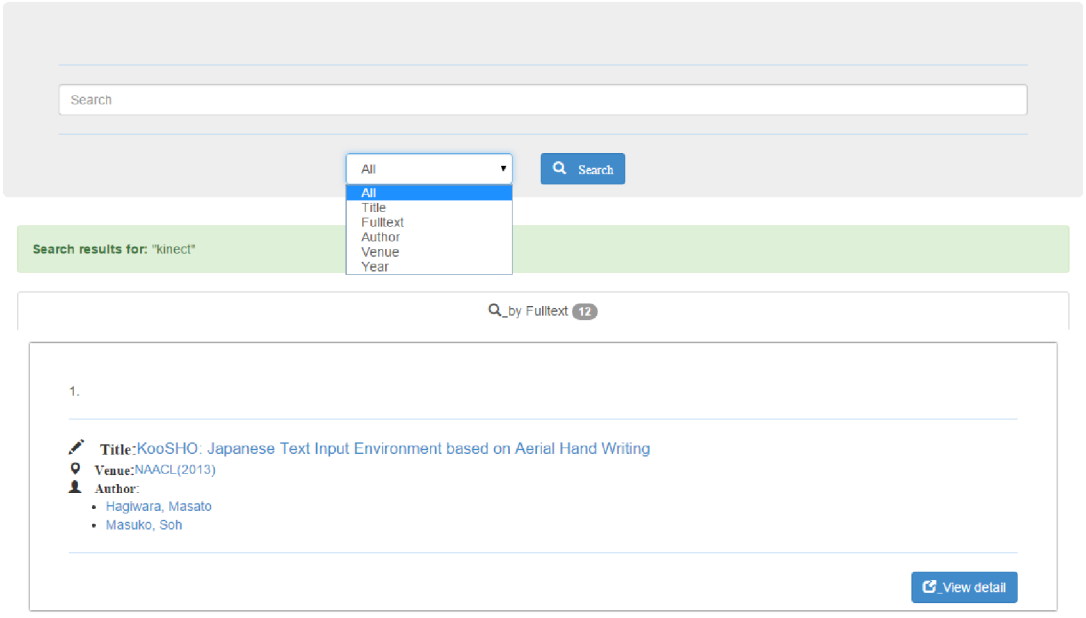
3.2.5.1 Práce a dotazování s Elasticsearch

Veškerou práci s bezešvou databázi Elasticsearch v grafickém rozhraní nalezneme v souborech HomepagePresenter.php a SearchPresenter.php, které využívají nástroj Elastica. V celé práci jsou využity tyto typy dotazu:

- Search – dotaz sloužící pro porovnání kratších textů, jako jsou jméno autora, nadpis, místo konání konference a rok.
- Morelikethis – se využívá pro porovnání podobnosti delších textů, jako v bakalářské práci abstrakt nebo celý text článku. Je zde možno zadat vlastnosti typu:
    - Minimum Term Frequeny – frekvence pod kterou budou termy ignorovány ve zdrojovém souboru.
    - Minimum Document Frequency – frekvence při které budou ignorována taková slova, které se nevyskytují v nejméně tolika dokumentech.
- Bool – slouží ke spojení více různých dotazů pod jeden výsledek pomocí vlastností should a must.

3.2.5.2 Vzhled a členění webu

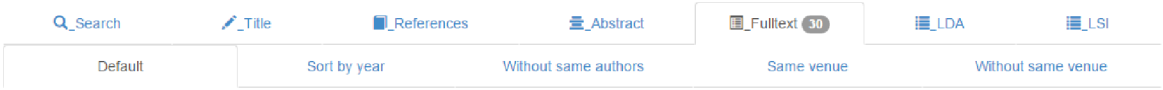
Webové rozhraní obsahuje dvě důležité části. První z nich je domovská, kde nalezneme lištu sloužící k vyhledávání dle zadaného termu a výběrem pole ve, kterém se má vyhledávat. Druhou částí lze určit zobrazení samostatného článku.



Obrázek 9 - Náhled vyhledávání v domovské části

Pro vyhledání podobných článků pro právě zobrazený lze využít lišta v dolní části webu, která zobrazuje možnosti vyhledávání:

1. Search – tato možnost je defaultní a lze zde nalézt formulář pro vyhledávání v celé množině dat.
2. Title – pod touto záložkou se vyhledají obdobné články dle jejich názvu k zobrazenému.
3. References – zde naleznete články, na které zobrazený článek cituje a existuje v množině testovaných dat.
4. Abstract – při zvolení této záložky se zobrazí obdobné články dle textu zvoleného článku a jeho abstraktu, který je ve zkoumání obdobnosti zvýhodněn.
5. Fulltext – zde se zobrazí obdobné články dle vyhodnocení fulltextového vyhledavače a jeho API moreLikeThis.
6. LDA – po zvolení této záložky, se zobrazí ve výsledku obdobné články, které byli vyhodnoceny zvolenou metodou
7. LSI – stejně jako LDA, jen byla využita metoda LSA

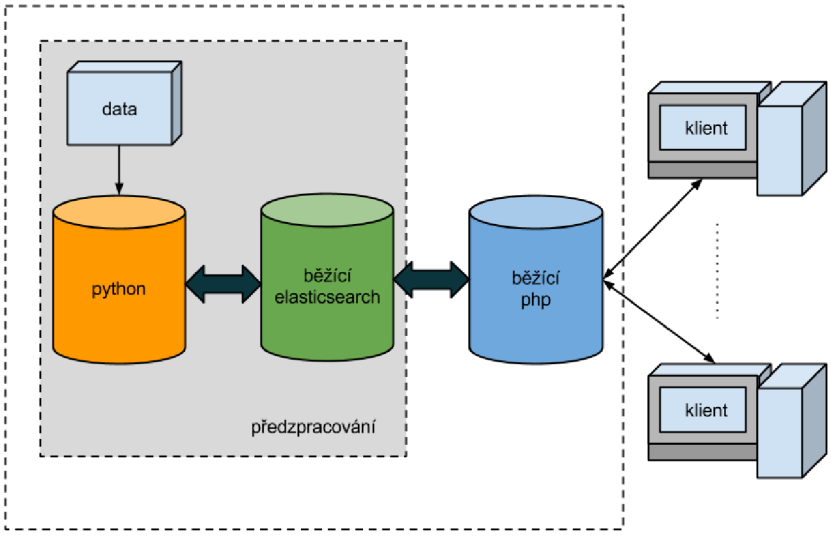


Obrázek 10 - Ukázka lišty pod zobrazeným článkem

3.2.6 Obecný popis celého systému

Celý systém lze popsat, obecně v několika bodech a to těchto:

1. Server, který obsahuje skriptovací jazyk Python, vykoná předzpracování a výsledek uloží do bezešvé databáze Elasticsearch.
2. Běžící server obsahující skriptovací jazyk Php zobrazuje klientům ve webovém prohlížeči grafické rozhraní, a tím nechává klienta dotazovat se na databázi se zjištěnými daty.

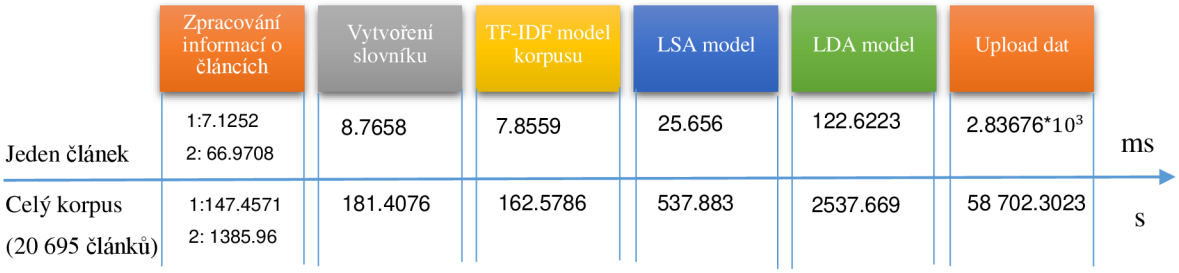


Obrázek 11 - Model systému

4 Testování a vyhodnocení

4.1.1 Vyhodnocení časové náročnosti předzpracování

Operace, která je pro systém časově nejnáročnější, nalezneme v sekci pro předzpracování. Na Obrázek 12, jsou zobrazeny jeho jednotlivé části, časové náročnosti pro celou množinu testovaných článků a průměrný čas pro jeden článek. V sekci pojmenované „Zpracování informací o článcích“ první hodnota (1:) značí časovou informaci o načtení textových dat ze souborů a jejich rozčlenění. Druhá (2:) znázorňuje rychlost úpravy textu pomocí lowercase, tokenizace a stemmeru, které jsou popsány v odstavci 2.2.7.



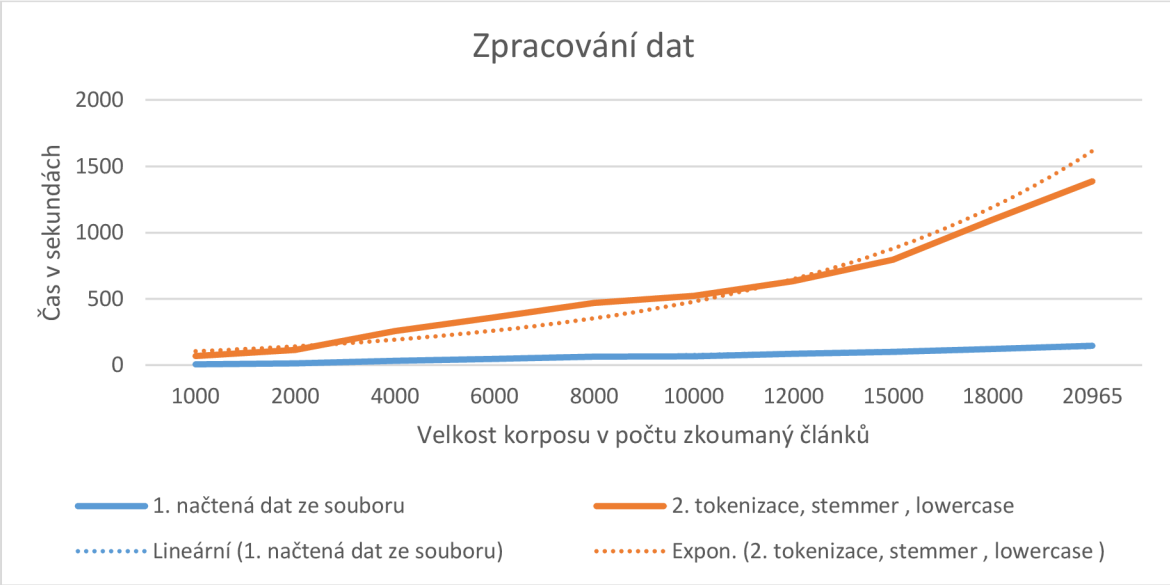
Obrázek 12 - Časové zhodnocení předzpracování

4.1.2 Časová náročnost předzpracování dle různých kritérií

Zde popíšeme časovou náročnost jednotlivých částí předzpracování například dle velikosti korpusu a počtu zvolených výsledných témat.

4.1.2.1 Zpracování dat (extrakce)

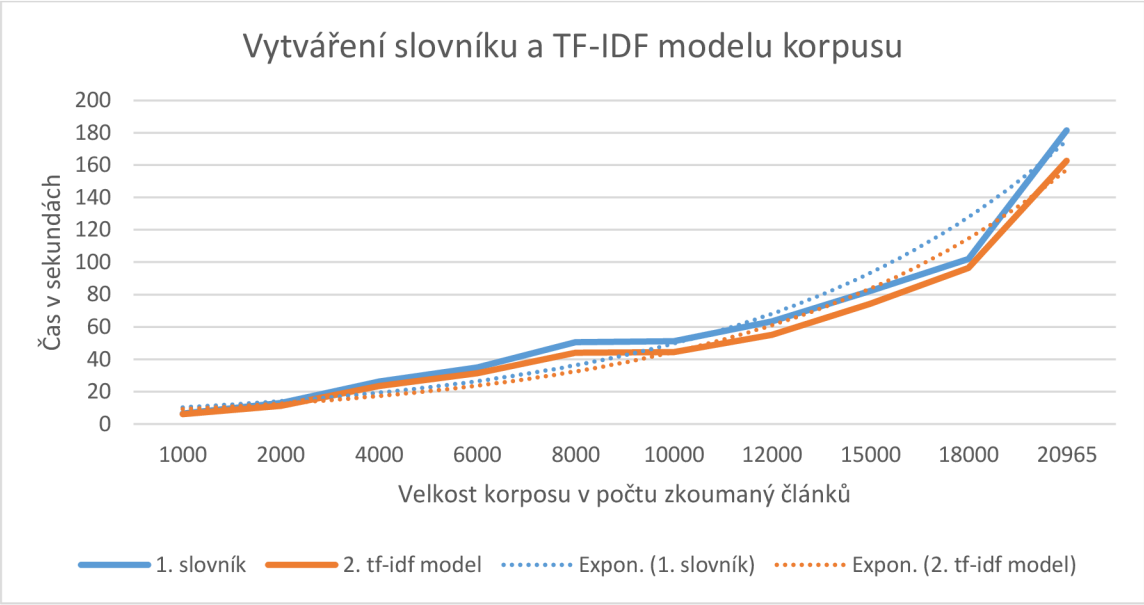
Graf 1 znázorňuje závislost časové náročnosti zpracování dat na velikosti korpusu. Z grafu je zřejmé, že u načtených dat ze souboru se průběh projevuje lineárně. Vyhodnocení tokenizace a dalších nástrojů nabývá exponenciální průběh.



Graf 1 - Znázorňuje čas zpracování dat / velikosti korpusu

4.1.2.2 Vytváření slovníku a TF-IDF modelu

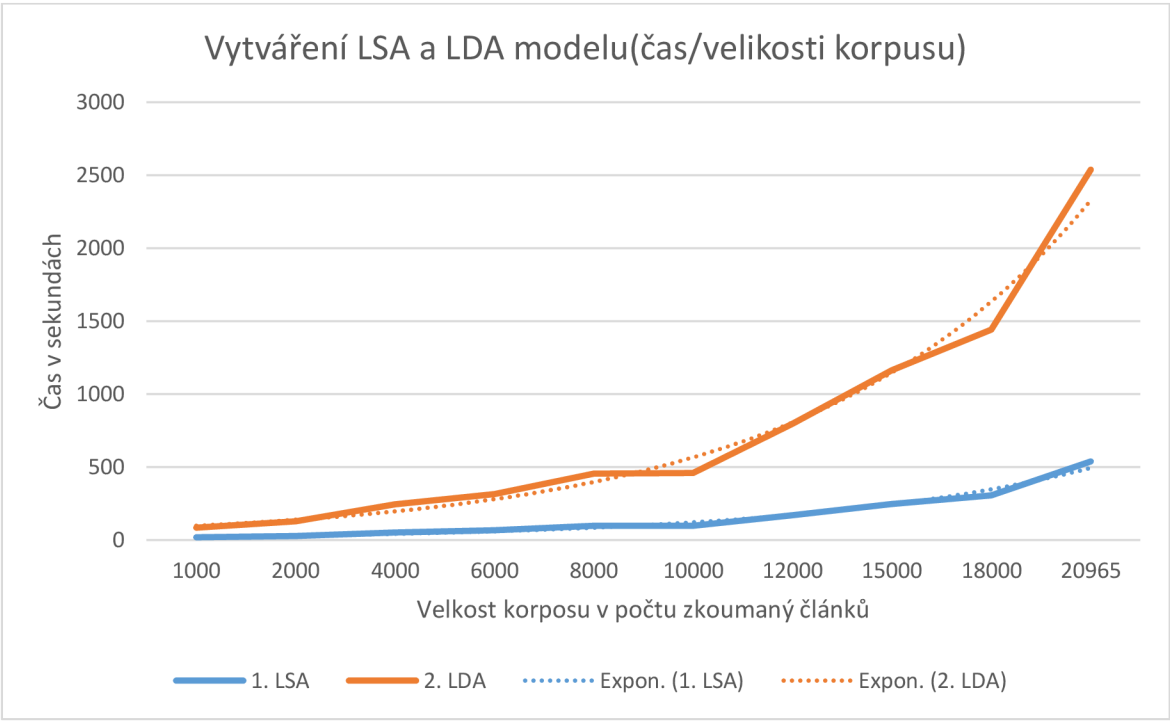
Další dvě části předzpracování, u kterých vyhodnotíme jejich průběh v závislosti času na velikosti korpusu, jsou vytváření slovníku a vytváření TF-IDF modelu. Jak lze vyčíst z Graf 2, mají obě vytváření exponenciální průběh.



Graf 2 - znázorňuje průběh vytváření slovníku a tf-idf modelu

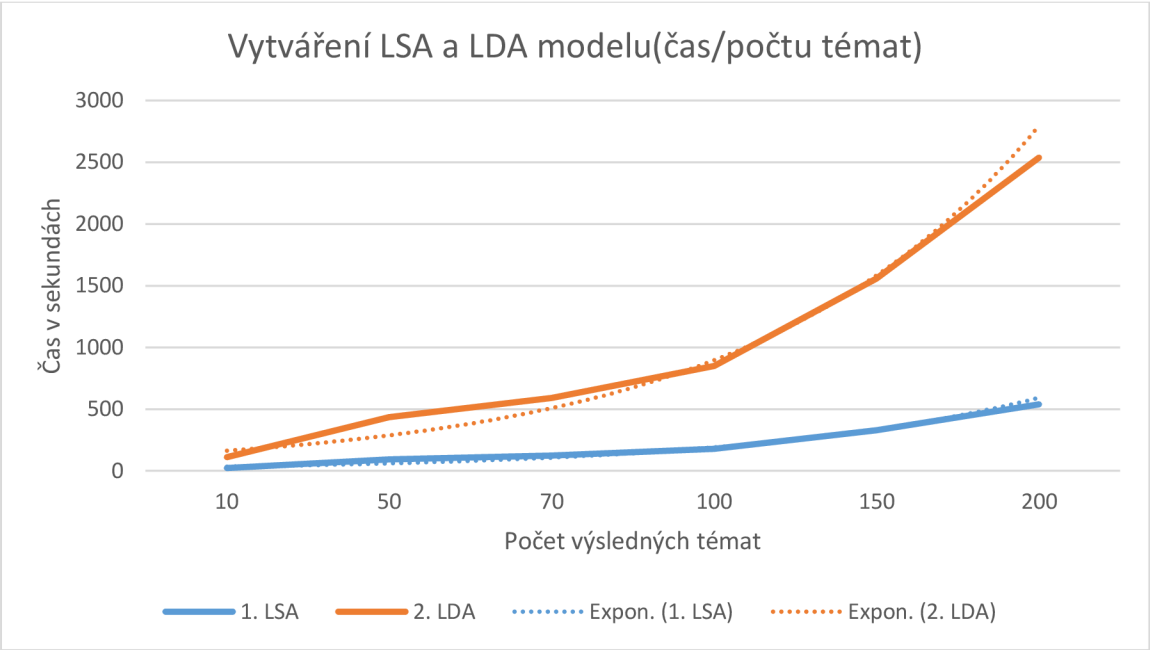
4.1.2.3 Vytváření LSA a LDA modelu

U Graf 3 je znázorněn průběh závislosti času na velikosti korpusu, u vytváření modelu LSA a LDA. U obou modelů se jedná o exponenciální průběh, i když jak lze z grafu vyčíst vytváření modelu LSA je o mnoho rychlejší než LDA, který zkoumá pravděpodobnost.



Graf 3 - znázorňuje průběh vytváření LSA a LDA modelu

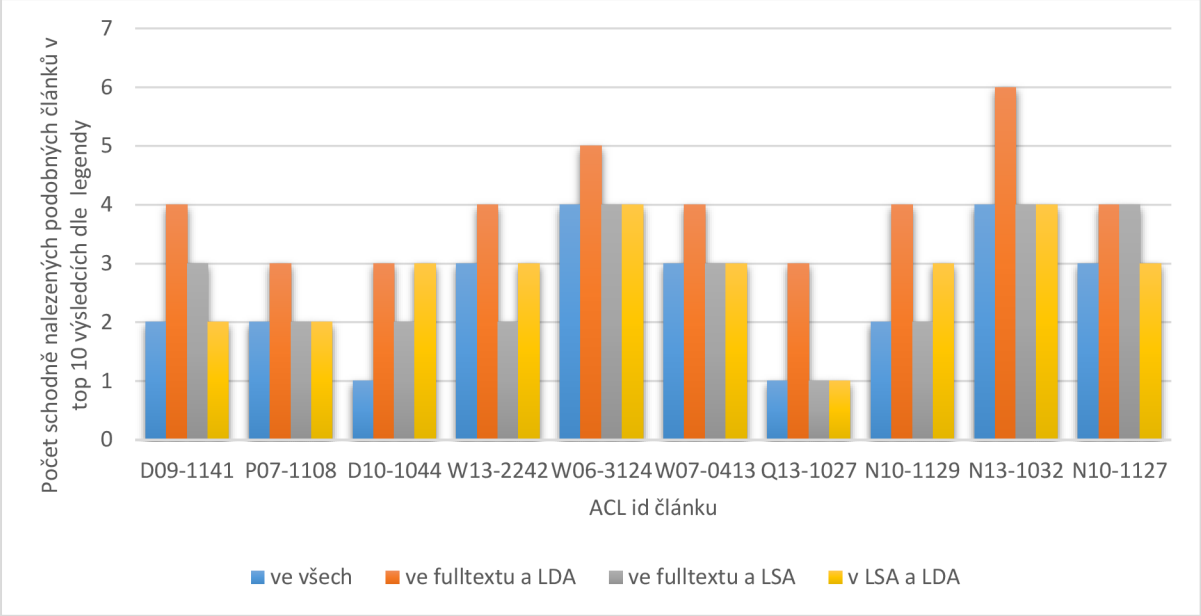
Zatímco u Graf 4 lze zjistit časová závislost vytváření modelu LDA a LSA na výsledném počtu vyhledávaných témat, kde se opět jedná o exponenciální průběh.



Graf 4 - znázorňuje průběh závislosti vytváření modelu LDA a LSA s různým počtem výsledných témat

4.1.3 Vyhodnocení výsledků vyhledáváni podle využitých metod

Pro vyhodnocení rozdílů mezi metodami bylo využito posouzení jejich deseti nejlepších výsledků podobnosti pro zvolený článek. Dalším kritériem bylo zjištění počtu shodně vyhledaných článků mezi nimi. Takto testováno bylo deset náhodných článků. Z Graf 5 lze vyčíst, že nejvíce stejných článků měly mezi sebou metody fulltext a LDA. Nejméně totožných článků měly metody fulltext a LSA. Ve všech porovnávaných metodách bylo k nalezení průměrně kolem dvou shodných článků. U vyhodnocování bylo zanedbáno pořadí výsledných článků.



Graf 5 - znázorňuje shodnosti výsledků jednotlivých metod v top 10 výsledcích

	ACL id	ve všech	ve fulltextu a LDA	ve fulltextu a LSA	v LSA a LDA
1	D09-1141	2	4	3	2
2	P07-1108	2	3	2	2
3	D10-1044	1	3	2	3
4	W13-2242	3	4	2	3
5	W06-3124	4	5	4	4
6	W07-0413	3	4	3	3
7	Q13-1027	1	3	1	1
8	N10-1129	2	4	2	3
9	N13-1032	4	6	4	4
10	N10-1127	3	4	4	3

Tabulka 7 - porovnání mezi metodami celého korpusu

4.2 Testování a vyhodnocení pomocí zpětné vazby z dotazníku

Pro vyhodnocení podobnosti článků byl využit dotazník, který sloužil jako zpětná vazba od respondentů. Vyhodnocení dat od dotazovaných z dotazníku dostaneme lidský úsudek na pohled podobnosti mezi články u zkoumaných metod.

4.2.1 Dotazník

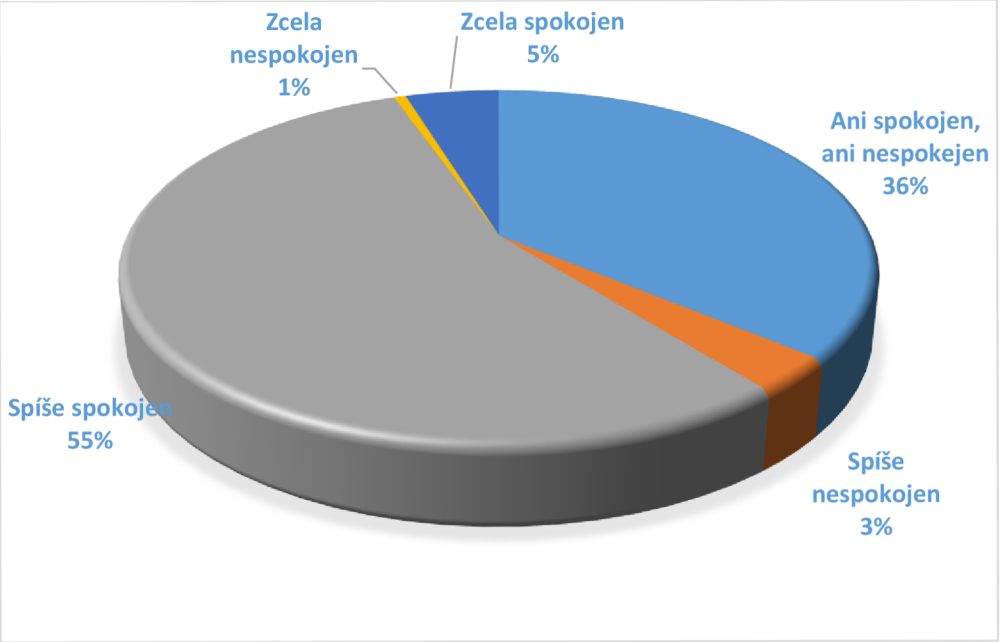
Dotazník byl vytvořen jako *google* formulář. Ve formuláři lze nalézt dvacet jedna otázek. První otázka je povinná. Pod ní se zkoumá kompetentnost respondentů, dále zda rozumí anglicky psaným textům a považují se za zdatné v oboru informačních technologii. Dalších dvacet otázek se zobrazí jen za předpokladu, že dotazovaný odpoví u první otázky kladně. Tyto otázky už jsou v podstatě stejné až na rozdílný odkaz na odkazovaný článek, kde je hodnocena spokojenost s výsledkem podobnosti dle zvolené metody.

Vyhodnocení výzkumu pomocí dotazníku se zúčastnilo celkem sto osmdesát čtyři respondentů. Na první otázku ze všech dotazovaných odpovědělo kladně sto čtyřicet osm. Dále v grafech je zohledňováno sto čtyřicet osm respondentů.

4.2.2 Vyhodnocení zpětné vazby

4.2.2.1 Metoda Fulltext

Vyhodnoceni metody Fulltext je zobrazeno na Graf 6. Hhodnoty naleznete v Tabulka 8, kde dle nich lze určit, že respondenti jsou zcela spokojeni z výsledkem z 5% a spíše spokojeni z 55%. Z tohoto důvodu lze usoudit, že respondenti jsou celkem spokojení s výsledkem metody, ale předpokládali lepší výsledky.



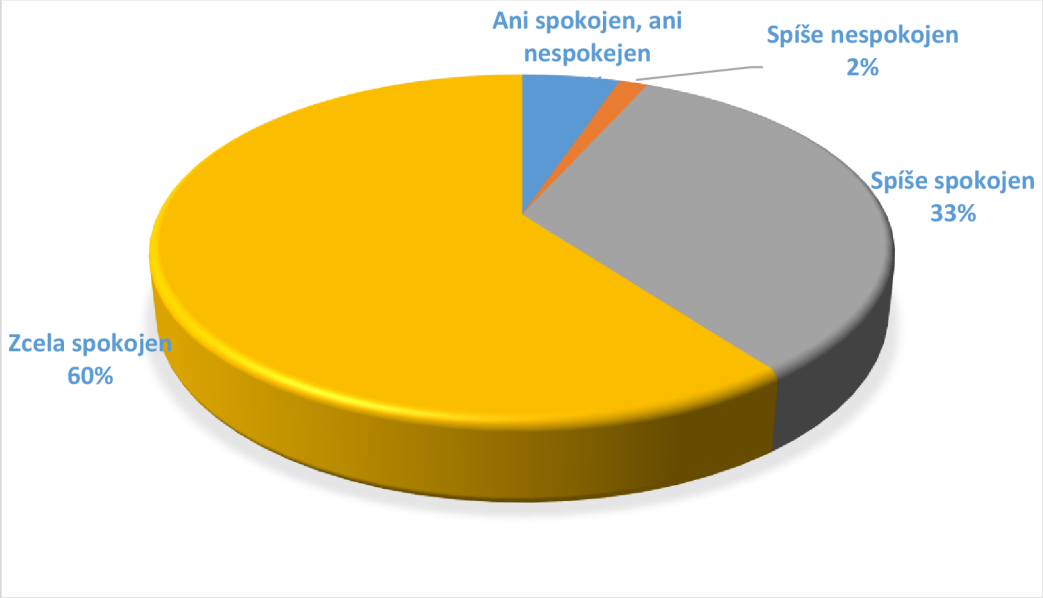
Graf 6 - vyhodnocení metody Fulltext respondenty

Popis odpovědí	Počet odpovědí
Zcela spokojen	140
Spíše spokojen	1630
Ani spokojen, ani nespokojen	1064
Spíše nespokojen	94
Zcela nespokojen	18
Celkový součet	**2946**

Tabulka 8 - hodnoty vyhodnocení dotazníku metody Fuultext

4.2.2.2 Metoda LDA

Další metodu, kterou vyhodnotíme pomocí odpovědí dotazovaných, je metoda LDA. Pro vyhodnocení využijeme informace z Tabulka 9, které jsou vyobrazeny na Graf 7. Oproti předešlé metodě, kde byli respondenti zcela spokojeni z 5%, jsou u této metody zcela spokojeni z 60%. Podle čehož můžeme usoudit, že tato technika má o dost lepší lidský úsudek podobnosti článků.



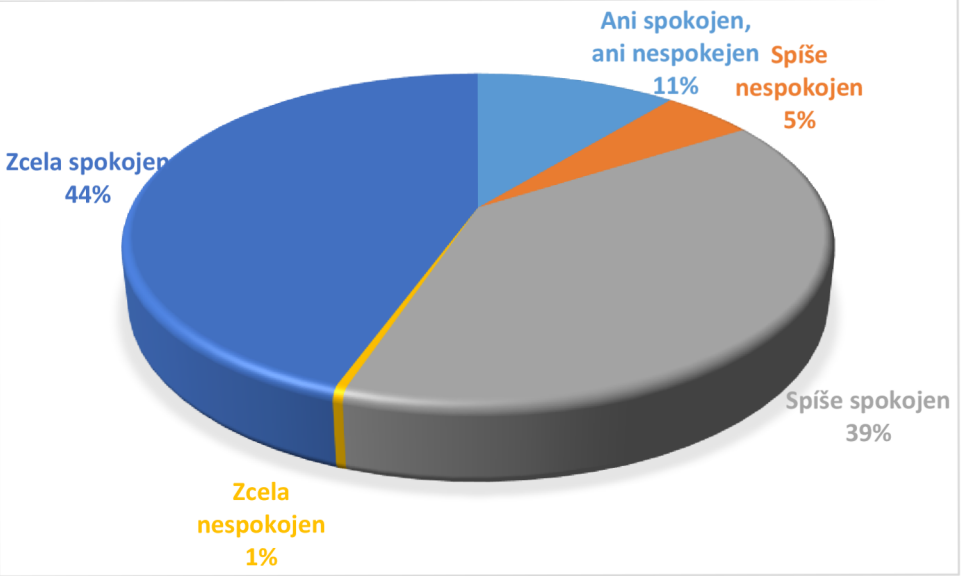
Graf 7 - vyhodnocení metody LDA respondenty

Popis odpovědí	Počet odpovědí
Zcela spokojen	1772
Spíše spokojen	978
Ani spokojen, ani nespokojen	152
Spíše nespokojen	44
Celkový součet	**2946**

Tabulka 9 – hodnoty vyhodnocení metody LDA respondenty

4.2.2.3 Metoda LSA

Poslední metoda, která byla obsažena v dotazníku, je označována LSA. Hodnoty k ní jsou v Tabulka 10 a jsou znázorněny na Graf 8. S výsledkem je zcela spokojeno 44% dotázaných. V porovnání spokojenosti respondentů se tato metoda umístila na druhém místě hned za LDA.



Graf 8 - vyhodnocení metody LSA respondenty

Popis odpovědí	Počet odpovědí
Zcela spokojen	1308
Spíše spokojen	1154
Ani spokojen, ani nespokojen	324
Spíše nespokojen	148
Zcela nespokojen	12
Celkový součet	**2946**

Tabulka 10 - hodnoty vyhodnocení metody LSA respondenty

4.3 Vyhodnocení automatického systému

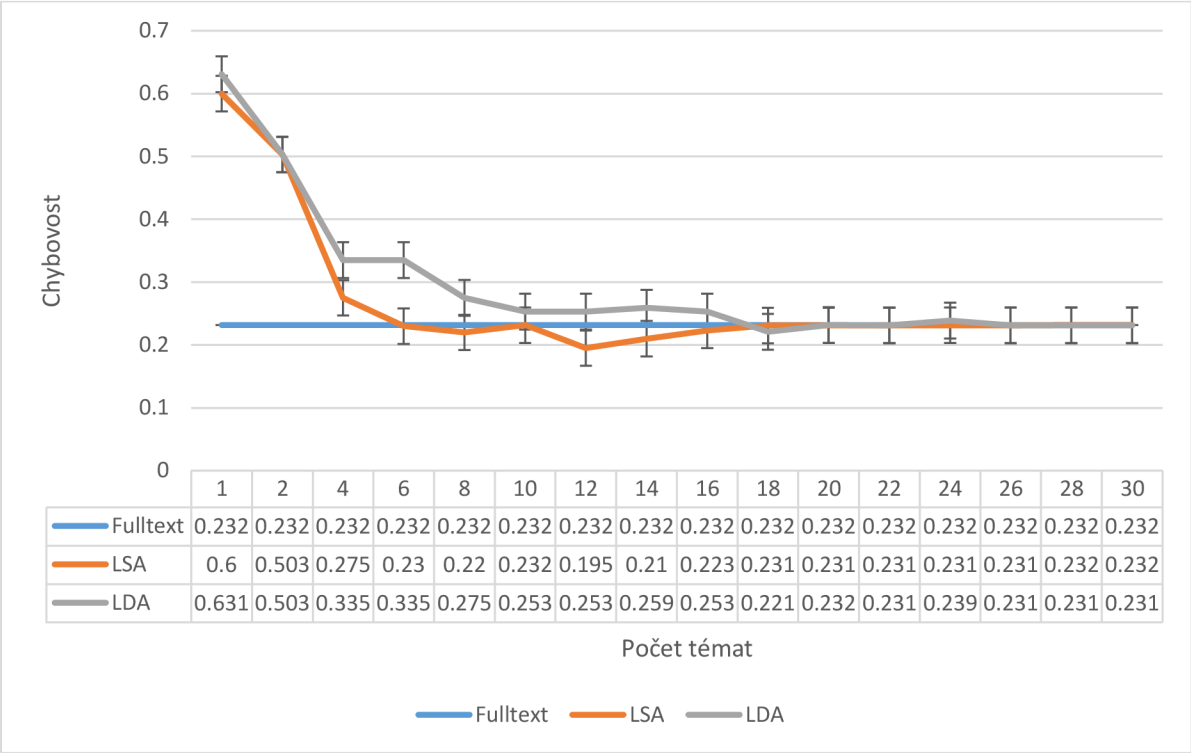
Pro vyhodnocení výsledného systému neexistují referenční data, proto si vytvoříme korpus o velikosti 50 článků, do kterých zahrneme podobné články ze stejného workshopu, kde jeden bude určen na dotazování.

Jako referenční data podobnosti v testovací korpusu určíme články z workshopu *Workshop On Multilingual And Mixed-Language Named Entity Recognition (2003).* K dotazování určíme článek s ACL označením *W03-1501*. Celá kontrolní množina pro testovací korpus se nachází v Tabulka 11. Zbývající obsah testovacího korpusu jsou náhodné články z různých konferencí.

Acl označení	**Název**
W03-1508	*Transliteration Of Proper Names In Cross-Lingual Information Retrieval*
W03-1506	*Multi-Language Named-Entity Recognition System Based On HMM*
W03-1502	*Automatic Extraction Of Named Entity Translingual Equivalence Based On Multi-Feature Cost Minimization*
W03-1505	*NE Recognition Without Training Data On A Language You Don't Speak*
W03-1504	*Low-Cost Named Entity Classification For Catalan: Exploiting Multilingual Resources And Unlabeled Data*
W03-1503	*Construction And Analysis Of Japanese-English Broadcast News Corpus With Named Entity Tags*
W03-1507	*Multilingual Resources For Entity Extraction*
W03-1509	*Chinese Named Entity Recognition Combining Statistical Model Wih Human Knowledge*

Tabulka 11 - hodnoty referenční množiny

U metod LSA a LDA určíme vhodný počet témat pro testování pomocí chybovosti. Na vyhodnocení použijeme desetinásobnou křížovou validaci z odstavce 2.4. Průběh této problematiky je znázorněn na Graf 9.



	1	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
Fulltext	0.232	0.232	0.232	0.232	0.232	0.232	0.232	0.232	0.232	0.232	0.232	0.232	0.232	0.232	0.232	0.232
LSA	0.6	0.503	0.275	0.23	0.22	0.232	0.195	0.21	0.223	0.231	0.231	0.231	0.231	0.231	0.232	0.232
LDA	0.631	0.503	0.335	0.335	0.275	0.253	0.253	0.259	0.253	0.221	0.232	0.231	0.239	0.231	0.231	0.231

Graf 9 - Chybovost systému dle počtu témat

Z grafu lze vyčíst, že vhodný počet témat u testovacího korpusu, je v rozmezí od osmnácti po třicet. Již bylo zjištno podle Graf 4 na stránce 27, že čím je větší počet témat, tím delší dobu systém stráví vyhodnocováním. Jako vhodný počet témat určíme nejnižší hodnotu - osmnáct.

4.3.1 Vyhodnocení (přesnost, úplnost, spolehlivost)

Nyní si vyhodnotíme systém pomocí tetovací korpusu pro počet témat rovný osmnácti. Výsledky jsou znázorněny v Tabulka 12.

Metoda	ACL označení shodných článků v top 10 výsledcích s ref.	Počet shodných článků
Fulltext	W03-1508, W03-1506	2
LDA	W03-1502, W03-1506, W03-1508, W03-1504	4
LSA	W03-1508, W03-1502, W03-1506	3

Tabulka 12 - porovnání metod s referenčními hodnotami

Rovnice:

$$Přesnost = \frac{|M \cap R|}{|M|} \quad (4.12.)$$

Přesnost, úplnost a F-míru systému lze vypočítat pomocí vztahu zaznamenaném v rovnici 4.12., 4.13. a 4.15., kde M značí množinu výsledných hodnot a R označuje množinu referenčního hodnot. Výsledná přesnost metody je vypočítána v Tabulka 13.

Rovnice:

$$Úplnost = \frac{|M \cap R|}{|R|} \quad (4.13.)$$

Rovnice:

$$F\text{-}míra = 2 * \frac{Přesnost * Úplnost}{Přesnost + Úplnost} \quad (4.14.)$$

Metoda	Přesnost	Úplnost	F-míra(spolehlivost)
Fulltext	0.2	0.25	0.222
LDA	0.4	0.5	0.444
LSA	0.3	0.375	0.333

Tabulka 13 - vyhodnoceni přesnosti, úplnosti a spolehlivosti systému

4.4 Vyhodnocení výsledků

Při vyhodnocení všech výsledků, jak zhodnocení zpětné vazby pomocí dotazníku, tak vyhodnocením systému pomocí testovacího korpusu s nadefinovanou referenční množinou, lze pronést, že nejvhodnější metoda pro určování sémantické podobnosti testovaných dat je model latentní Direchletovy alokace. Na druhém místě se umístila technika latentní sémantické analýzy a na posledním místě využití pouze *moreLikeThis* API u fulltextového vyhledávače Elasticsearch.

5 Závěr

Cílem bakalářské práce bylo prozkoumat modely pro určování sémantické podobnosti odborných článků psaných přirozeným jazykem. Pro tuto problematiku byl vytvořen systém, který u testovaných dat určil vzájemnou podobnost.

V teoretické části jsou definované základní pojmy zpracování přirozeného jazyka, principy použitých metod pro určení sémantické obdoby tak, aby čtenář získal potřebný přehled o zkoumané problematice.

Základním úkonem systému bylo vytvoření vektorového modelu pro celou množinu testovaných dat. Pro určení nejlepší použité metody byla využita zpětná vazba od respondentů dotazníku a ohodnocení automatického systému pomocí vytvořeného zmenšené korpusu, u kterého byla určena referenční podobnost článků.

U určení podobnosti pomocí metod pro shlukovou analýzu výrazně ovlivňují výsledky nastavení parametrů jako počet témat, iterací a jiné.

6 Bibliografie

[1]. Grossman, David A. a Ophir, F. *Information Retrieval Algorithms and Heuristics.* Second edition. Dordrecht : Springer Netherlands, 2004. ISBN 978-1-4020-3004-8.

[2]. Latent semantic indexing. *Wikipedia, the free encyclopedia.* [Online] 31. 3 2015. [Citace: 5. 5 2015.] http://en.wikipedia.org/wiki/Latent_semantic_indexing.

[3]. BERRY, Michael W a Murray BROWNE. *Understanding search engines: mathematical modeling and text retrieval.* 2nd ed. Philadelphia, PA : SIAM, Society for Industrial and Applied Mathematics, 2005. ISBN 08-987-1581-4.

[4]. Dostál, Zdeněk. *Lineární algebra.* Ostrava : Vysoká škola báňská, 2011.

[5]. Jiří. *Sémantická analýza textů.* [Online] 4. 12 2011. [Citace: 5. 5 2015.] URL http://fulltext.sblog.cz/2011/12/04/semanticka-analyza-textu-5/.

[6]. Probabilistic latent semantic analysis. *Wikipedia, the free encyclopedia.* [Online] 2015. [Citace: 1. 5 2015.] Dostupné z: http://en.wikipedia.org/wiki/Probabilistic_latent_semantic_analysis.

[7]. BLEI, David M., NG, Andrew Y. a JORDAN, Michael I. Latent Dirichlet Allocation. *Journal of Machine Learning Research 3.* [Online] 2003. [Citace: 1. 5 2015.] Dostupné z: https://www.cs.princeton.edu/~blei/papers/BleiNgJordan2003.pdf.

[8]. Veselovský, Martin. *Sémantická podobnost článků, bakalářská práce.* Brno : FIT VUT v Brně, 2014.

[9]. Řehůřek, Radim. *Scalability of Semantic Analysis in Natural Language Processing.* Dizertační práce : Masaryk University, Faculty of Informatics, 2011.

[10]. Falkner, Benjamin a Schröder, Gunnar F. *Cross-validation in cryo-EM–based structural modeling.* - : National Academy of Sciences, 2013. Sv. Proceedings of the National Academy of Sciences of the United States of America. ISSN: 0027-8424.

[11]. GOLUB, Gene H. *Matrix computations.* 3rd ed. Baltimore : Johns Hopkins University Press, 1996. ISBN 08-018-5414-8.

7 Příloha 1 – Ovládaní a požadavky na systém

- ## Část systému předzpracování

Nejprve je potřeba nastavit připojení na server Elasticsearch v souboru config.xml, kde lze nastavit cesty k meta-datům o článcích a k adresáři s texty článků.

Tato část se spouští příkazem *make* v terminálu v rozbaleném adresáři *Preprocess*. Příkaz *make help* slouží k získání konkrétnějších informací o spuštění systému pomocí různých parametrů.

Požadavky: Python verze 2.7

- Potřebné knihovny Python:
  - Elasticsearch
  - os, sys, re, string
  - xml.etree.ElementTree
  - pickle, time, nltk
  - gensim, logging, numpy

- ## Webová část systému

Nejprve je třeba nastavit připojení na server Elasticsearch v souborech *HomepagaPresenter.php* a *SearchPresenter.php* na místa k tomu určených dle komentářů. Zadané parametry musí být identické jako u předzpracování.

Tato část systému se zpřístupní nakopírováním obsahu složky WWW na budoucí server, nastavení práv pro čtení, zápis adresáře *www* a nastavení správného routování adresy v souboru „*.htaccess*“.

Požadavky: PHP verze 5.3.29

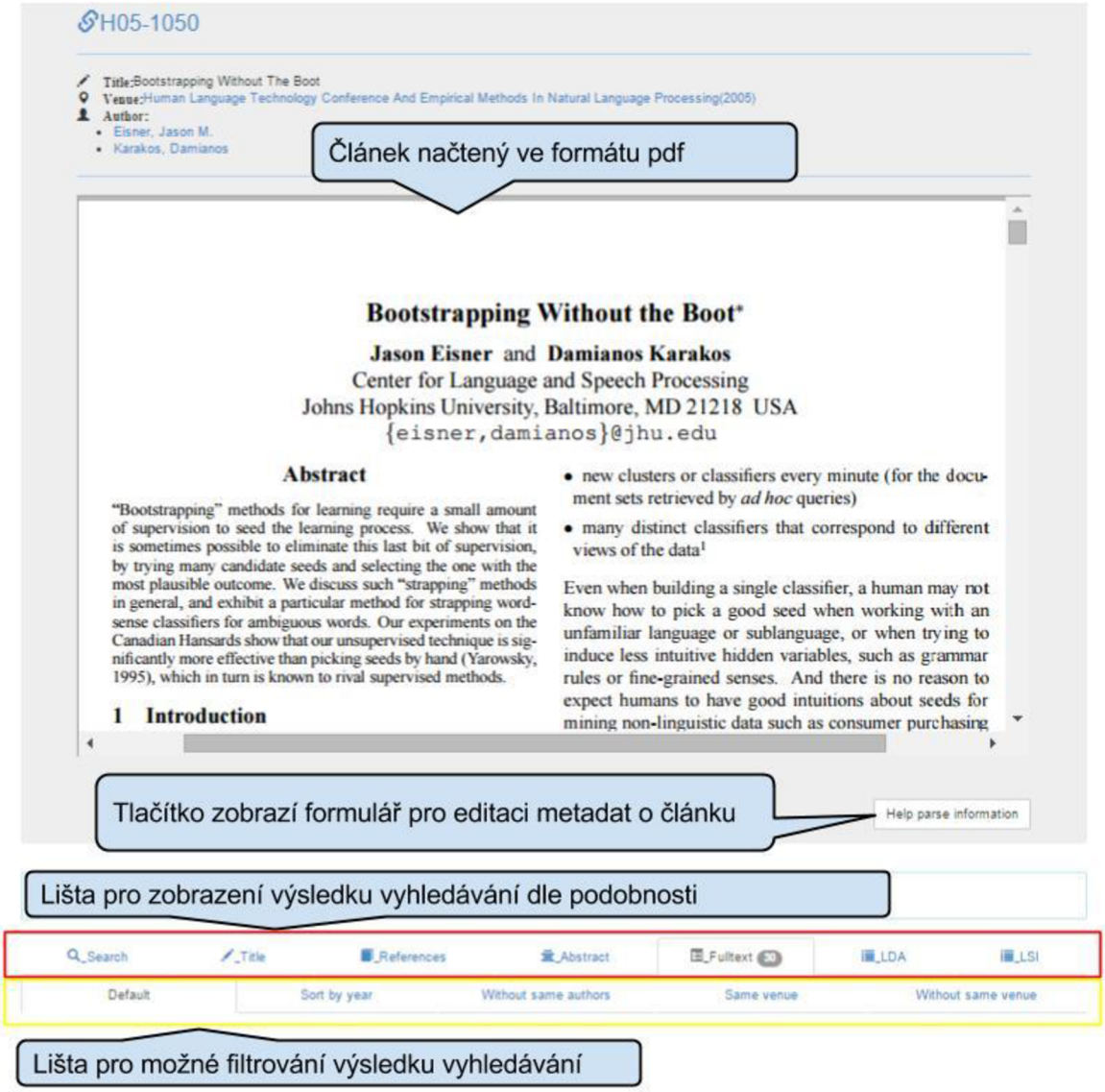
- Php framework Nette 2.3.2
  - potřebné nástroje pro Nette:
    - Elasticsearch verze 1.0 Elasticsearch Php
    - elasctica verze 0.16.0.0 Elasticsearch Php client
    - je třeba nakopírovat do složky vendor v Nette projektu Sandbox dále je nutné tyto klienty nastavit v souboru composer.json

```
"require": {
        "elasticsearch/elasticsearch": "~1.0",
        "ruflin/elastica": "dev-master"
}
```

- složka *www* v Nette projektu Sandbox
  - ve složce js nahrát Jquery verzi 1.11.1
  - Ve složkách js a css nahrát Boostrap verzi 3.3.4

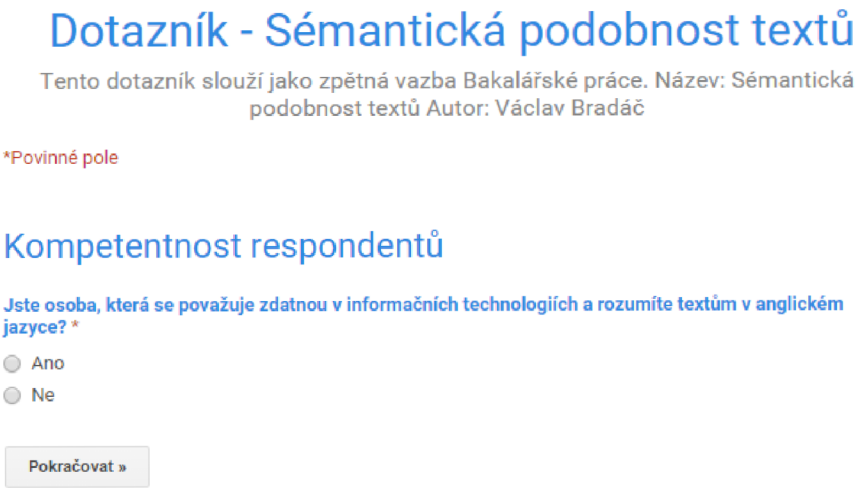
Ovládání domovské stránky:

Ovládání stránky se zobrazeným článkem:

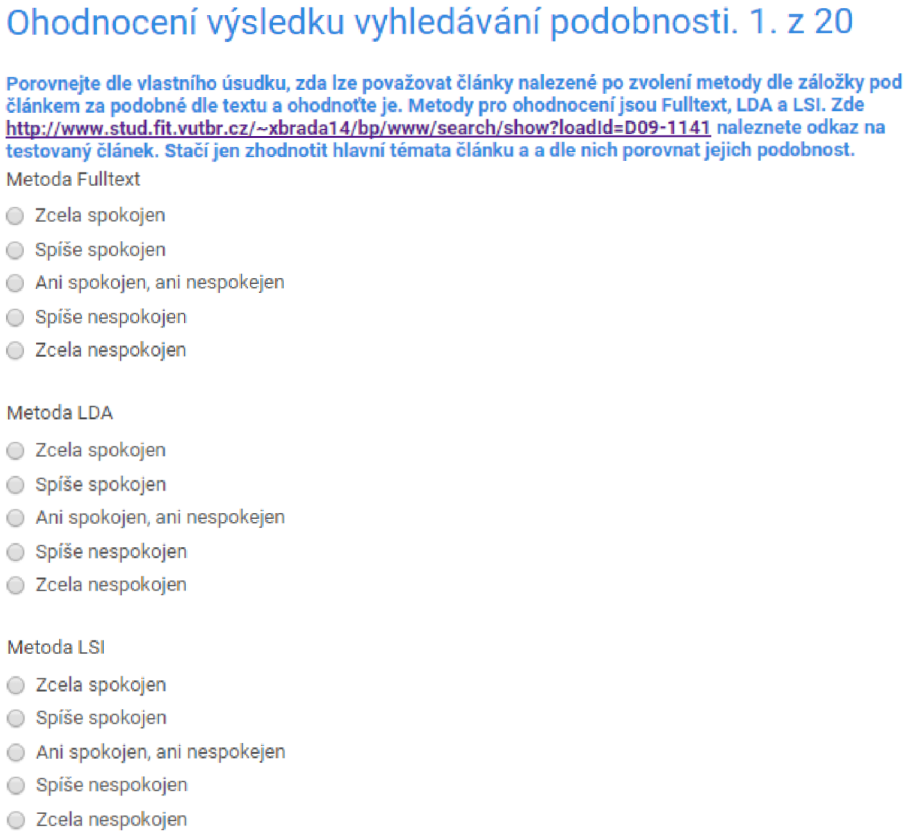


8 Příloha 2 - Dotazník

První otázka:



Dalších 20 otázek pokud bude první zodpovězena Ano:¨



Obrázek 13 - ukázka formátu dvaceti otázek

Rozdíl mezi nimi je jen v odkazu na zkoumaný článek.

Výpis ACL označení zkoumaných článků
1.D09-1141, 2.P07-1108, 3.D10-1044, 4.W13-2242, 5.W06-3124, 6.W07-0413, 7.Q13-1027, 8.N10-1129, 9.N13-1032, 10.N10-1127, 11.P06-1067, 12.N04-1022, 13.W10-3804, 14.E09-1044, 15.W07-0414, 16.P06-1077, 17.D07-1029, 18.D08-1010, 19.W08-0316, 20.W11-1209

Metoda Fulltext					
Otázka	Zcela spokojen	Spíše spokojen	Ani spokojen, ani nespokojen	Spíše nespokojen	Zcela nespokojen
1	0	106	42	0	0
2	20	108	20	0	0
3	0	82	48	12	6
4	0	114	28	0	6
5	6	82	46	14	0
6	20	88	34	6	0
7	6	94	48	0	0
8	6	94	48	0	0
9	0	58	90	0	0
10	0	108	40	0	0
11	6	74	68	0	0
12	6	46	90	6	0
13	20	54	74	0	0
14	6	110	26	6	0
15	0	52	90	6	0
16	14	54	62	12	6
17	0	102	40	6	0
18	12	70	60	6	0
19	12	80	42	14	0
20	6	54	68	6	0
Celkem:	**140**	**1630**	**1064**	**94**	**18**

Metoda LDA					
Otázka	Zcela spokojen	Spíše spokojen	Ani spokojen, ani nespokojen	Spíše nespokojen	Zcela nespokojen
1	84	50	0	14	0
2	116	32	0	0	0
3	96	20	26	6	0
4	68	74	6	0	0
5	54	88	0	6	0
6	68	60	20	0	0
7	96	38	14	0	0
8	122	20	6	0	0
9	48	100	0	0	0
10	82	60	6	0	0
11	54	88	6	0	0
12	68	74	6	0	0
13	88	60	0	0	0
14	90	52	6	0	0
15	37	29	2	0	0
16	110	6	20	12	0
17	110	32	6	0	0
18	124	18	0	6	0
19	82	54	12	0	0
20	68	60	6	0	0
Celkem:	**1772**	**978**	**152**	**44**	**0**

Metoda LSI					
Otázka	Zcela spokojen	Spíše spokojen	Ani spokojen, ani nespokojen	Spíše nespokojen	Zcela nespokojen
1	70	42	24	12	0
2	110	32	0	6	0
3	54	48	34	12	0
4	68	48	20	12	0
5	42	68	26	12	0
6	42	88	12	6	0
7	56	74	12	6	0
8	76	54	6	12	0
9	28	100	20	0	0
10	62	62	18	6	0
11	28	88	18	14	0
12	48	60	20	20	0
13	84	52	12	0	0
14	62	74	6	6	0
15	28	29	9	2	0
16	98	6	32	6	6
17	76	66	0	6	0
18	104	32	0	6	6
19	82	54	12	0	0
20	62	40	32	0	0
Celkem:	**1308**	**1154**	**324**	**148**	**12**

9 Příloha 3 – Plakát systému