Bakalářská práce

Kategorizace uživatelů na základě historie stahovaných webových dokumentů

Dušan Jenčík

Vedoucí práce: Ing. Jan Šedivý, CSc.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Praha 2015

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Dušan J e n č í k

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Kategorizace uživatelů na základě historie stahovaných webových dokumentů

Pokyny pro vypracování:

Cílem této práce je kategorizovat uživatele internetu na základě znalosti historie stahování webových dokumentů. Úkolem kategorizace je zařadit uživatele do různých kategorií (např. ženy, děti, podle věku apod.). Pro práci bude poskytnuta databáze posloupností stahovaných URI skutečných uživatelů internetu. Postupujte podle následujících kroků:

- Prostudujte metody pro klástrování do kategorií.
- Prostudujte generativní statistické modely pro uspořádání URI do neznámých kategorií.
- Proveďte základní statistickou analýzu poskytnuté databáze.
- Vyberte vhodné algoritmy na základě předchozí analýzy a aplikujte je na databázi
- Nalezněte vhodná kritéria pro posouzení kvality kategorizace vybranými algoritmy a posuďte jejich přesnost a vhodnost.

Seznam odborné literatury:

[1] Kanungo, Tapas, et al. "An efficient k-means clustering algorithm: Analysis and implementation." Pattern Analysis and Machine Intelligence, IEEE Transactions on 24.7 (2002): 881-892.

[2] Ahmed, Amr, and Alexander Smola. "Www 2011 invited tutorial overview: latent variable models on the internet." Proceedings of the 20th international conference companion on World wide web. ACM, 2011.

[3] Attardi, Giuseppe, Antonio Gulli, and Fabrizio Sebastiani. "Automatic Web page categorization by link and context analysis." Proceedings of THAI. Vol. 99. No. 99. 1999.

[4] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12 (2011): 2825-2830.

Vedoucí bakalářské práce: Ing. Jan Šedivý, CSc.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 5. 2. 2015

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne ....

Podpis autora práce

Poděkování

Rád bych poděkoval především mému vedoucímu Ing. Janovi Šedivému, CSc., který mi nabídl toto téma bakalářské práce a po celou dobu jejího vypracování byl perfektním vedoucím.

Dále bych své poděkování věnoval i Ing. Tomášovi Gogárovi, Ing. Tomášovi Tunysovi a Ing. Tomášovi Bařinovi za obětavou pomoc při vypracovávání této práce.

Výpočetní prostředky byly poskytnuty MetaCentrem v rámci programu LM2010005 a skupinou CERIT-SC v rámci programu Center CERIT Scientific Cloud, která je součástí Operational Program Research and Development for Innovations, reg. č. CD.1.05 / 3.2.00 / 08.0144.

Mé rodině.

Abstrakt

Cílem této práce je nalezení metod a postupů vedoucích ke kategorizaci uživatelů dle historie jejich záznamů z procházení internetu. Práce využívá analytické a statistické metody, kterými se snaží nalézt kategorie webových stránek charakteristických pro určitou skupinu uživatelů. Bylo zjištěno, že shlukovací algoritmy nejsou dostatečně popisné pro nalezení požadovaných kategorií, a tak bylo využito topic-model algoritmu pLSA. Díky tomuto algoritmu byla nalezena témata tvořená distribucemi webových stránek a zároveň každý uživatel byl popsán distribucí nalezených témat. Popis témat byl doplněn o kategorie z DMOZ databáze a následně o nejvýznamnější slova, která se vyskytují na stránkách charakterizujících dané téma. Pro tuto práci byla poskytnuta zanonymizovaná data nejmenovanou antivirovou společností.

Klíčová slova kategorizace, shluková analýza, topic-model, pLSA

Abstract

The aim of this thesis is to find methods and procedures which are leading to categorization of users with respect to history of their records from internet browsing. The work uses analytical and statistical methods, by which it tries to find some categories of websites, which are characteristic for a specific group of users. It has been found that clustering algorithms are not sufficiently descriptive for finding required categories, and thus it has been used topic-model algorithm named pLSA. The topics have been found thanks to this algorithm. The topics are formed by distribution of websites and every user is described by distribution of the found topics. The description of topics has been supplemented with categories from DMOZ database and later with the most important words, which are appeared on web pages describing the topic. Anonymized data was provided by unnamed antivirus company.

Keywords: categorization, clustering analysis, topic-model, pLSA

Obsah

Seznam tabulek

Seznam obrázků

Seznam zdrojových kódů

Kapitola 1

Úvod

Na internetu je v dnešní době téměř každý. Lidé využívají internetových služeb a tyto služby využívají dat svých uživatelů. Internetové společnosti jsou dnes nuceni analyzovat své uživatele, aby byly schopné držet krok s konkurencí. Tato práce ukazuje postupy při analýze nasbíraných dat o uživatelích.

1.1 Motivace

Některé softwarové společnosti sbírají data od svých uživatelů. V případě webových produktů se nejčastěji používá clickstream[1]. Názorným příkladem může být antivirová společnost, která pro zefektivnění svých antivirových produktů sbírá záznamy z procházení internetu od některých svých uživatelů. Jedna taková společnost[2] poskytla data pro tuto práci. Sběr dat je prováděn při zadání URL[3] adresy do prohlížeče uživatelem. Předtím než uživatel dostane obsah stránky, tak antivirus danou webovou stránku prověří na výskyt škodlivého kódu a v negativním případě uživateli stránku povolí načíst. Pokud na chtěné stránce bude objeven virus či jiný nežádoucí kód, tak je uživatel varován a přístup na stránku mu je rozmlouván, popřípadě zamítnut. Tato a jí podobné společnosti takto mohou sbírat velké množství dat, která z povahy clickstreamu mohou narůstat do enormních velikostí (stovek GB či dokonce až jednotek TB za jediný den). Takto velké množství dat není triviální zpracovat a je nutné použít sofistikované metody k získání hlubší informace. Těmito metodami se zabývá tato práce.

[1] Záznam navštívených adres [14].
[2] Partner ČVUT.
[3] Uniform Resource Locator. Unikátní webová adresa.

1.2 Definice problému

1.2.1 Data

Vzhledem k tomu, že clickstream spadá do kategorie velmi citlivých dat, tak je nutné data anonymizovat. Jedná se o proces, kde se určitým způsobem skryje či odstraní ta část dat, která je citlivá.

Struktura clickstreamu bývá přibližně následující[4]:

Tabulka 1.1: Struktura clickstreamu

ID počítače	čas UTC	HTTP referrer	URL	IP adresa

Nejdůležitějšími parametry clickstreamu jsou ID počítače (prozatím bráno jako ID uživatele[5]) a URL adresa na kterou směřoval.

V této práci je popisováno zpracování již zanonymizovaných dat, která jsou popsatelná vektorem webových adres[6] a maticí četností[7] obsahující na svých sloupcích 586 624 URL a na řádcích jsou náhodně seřazení uživatelé (resp. ID počítače), kterých je 224 679. Každý uživatel (resp. ID počítače) je zastoupen pouze jednou. Jedná se přitom pouze o populaci žijící v USA, kde data byla nasbírána za období jednoho měsíce.

1.2.2 Problém

Hlavní ideou této práce je kategorizace uživatelů na základě jejich procházení internetu. Cílem je popsat metody vedoucí k nalezení skupin webových stránek, které jsou pro určitou skupinu populace charakteristické. Tedy se jedná o nalezení takových skupin stránek, na které chodí „podobní“ uživatelé. Tato informace ale není z kontextu dat přímo jasná, a proto je potřebné data analyzovat.

[4]Konkrétní clickstream může obsahovat několik dalších údajů jako jsou stát, město atp. Tyto a další parametry nejsou získávány přímo od uživatele, ale na základě heuristik postavených na čistém clickstreamu.

[5]Na jednom počítači, který je monitorován, může pracovat více členů domácnosti. Z důvodu zjednodušení problému je uvažováno o záznamech z jednoho počítače tak, že jsou generovány pouze jedním uživatelem.

[6]Byly zvoleny pouze domény 2. řádu. Příkladem *facebook.com*, *google.com*...

[7]V buňkách matice jsou počty návštěv konkrétního uživatele na konkrétní URL adrese.

1.3 Struktura práce

První kapitola Popis problému a struktury dat.

Druhá kapitola Práce pojednává o řešení podobného problému jinými.

Třetí kapitola Zde je popsána povaha dat a různé metody (algoritmy) vhodné pro řešení. Ke konci kapitoly jsou popsány postupy, které nevedly k uspokojivým výsledkům.

Čtvrtá kapitola Pojednání o finálním zpracování dat. Popsané konkrétní postupy, výpočty a získané výsledky.

Páta kapitola Shrnut závěr výzkumu, dosažené výsledky a popsána cesta pro zdokonalení výsledků.

Kapitola 2

Související práce

Analýzou clickstreamu se v dnešní době zabývá mnoho společností. Některé pro vylepšení svého marketingu, jiné proto, aby byly lépe konkurenceschopné, ale téměř všechny to dělají za primárním cílem: zjistit o zákazníkovi vše důležité a následně tyto znalosti zmonetizovat. V této kapitole jsou zmíněny různé přístupy vycházející z analýzy clickstreamu, které více či méně byly přínosné pro analýzu v této práci.

2.1 Analýza clickstreamu

Analýza clickstreamu je v dnešní době poměrně častým jevem, ale primárně je tvořena společnostmi, které analyzují své uživatele (např. e-shop) a dle jejich chování usměrňují své další kroky v marketingu. Takové analýzy vznikají díky zaznamenané cestě (sled webových adres - clickstream) uživatelem. Tyto cesty jsou tvořeny pouze na stránkách vlastněných nějakou společností (např. e-shop). Díky analýze sledů svých uživatelů lze určit například pohlaví svých zákazníků jak je tomu psáno v [15]. Každý uživatel má jiné chování, ale lze najít jisté charakteristické rysy v procházení zmíněného e-shopu typické pro muže a pro ženy. *Path analysis*, neboli analýza sledů, dávají významnou informaci například o tom, jestli zákazník provede nákup či nikoli. Díky statistickým metodám lze vypozorovat, jaké sledy událostí (navštívení částí webu) vedou k úspěšnému nákupu např.: {*Domovská stránka* → *Kategorie produktů* → *Kategorie produktu* → *Nákupní košík*}. Naopak pokud sled vypadá: {*Domovská stránka* → *Informace o webu* → *Domovská stránka* → *Informace o webu* → *Kategorie produktu* ...}, tak je velká pravděpodobnost, že zákazník s takovým clickstreamem nákup produktu neuskuteční, nebo alespoň ne ihned. Zmíněná práce se zabývá tím, jak predikovat nákup a kde jsou ta důležitá místa v clickstreamu, která rozhodují o tom, jestli zákazník nakoupí či nikoli. Detailnější popsání clickstreamu je v práci [23].

Bohužel naše situace je poněkud odlišná, a to tím, že data z clickstreamu nejsou pro jeden e-shop, ale pro větší část celého internetu. Další negativum je v tom, že nám nebyl poskytnut „surový“ clickstream, ale již data transformována do matice četností. Na tomto základě je nutné uvažovat nad problémem z jiných úhlů.

2.2 Analýza matice četností

Vzhledem k tomu, že máme k dispozici matici četností, jak již bylo naznačeno v části 1.2.1, tak je nutné hledat zpracování právě této matice a nikoli čistého clickstreamu, přestože matice četností vychází právě z clickstreamu. Práce [24] se zabývá analýzou matice četností z pohledu sémantických vektorů. Vektorový prostor je zde obhajován pro jeho snazší a rychlejší zpracování. Načež práce [11] navazuje vysvětlením efektivního shlukovacího algoritmu K-means. Shlukovací algoritmy jsou pro kategorizaci velmi vhodné. V práci [12] je zmíněno zpracování TF–IDF algoritmu, vysvětleny důležitosti normalizace vektorů a benefity kosinové podobnosti. Z výše zmíněných prací a mnohých dalších je v tomto výzkumu vycházeno. Vzhledem k tomu, že úplně stejným tématem se nezabývá žádná práce, tak je nutné s pomocí dílčích znalostí zkonstruovat postup, který je v této práci popsán.

Kapitola 3

Analýza

V této kapitole je popsána povaha dat, která jsou následně pomocí několika algoritmů zanalyzována. Data bylo nutné předzpracovat, aby vynikly diskriminující informace. Jsou zde popsány některé z nejvýznamnějších algoritmů, které jsou vhodné pro zpracování dat podobného charakteru.

3.1 Povaha dat

Jak již bylo zmíněno v úvodu 1.2.1, tak výzkum probíhá nad maticí četností s rozměry 586 624 URL × 224 679 uživatelů. Matice je uložena v CSR[1] formátu a vypadá následovně:

$$\begin{array}{c} \\ u_1 \\ u_2 \\ \vdots \\ u_{224\,679} \end{array} \begin{array}{c} \begin{array}{cccc} w_1 & w_2 & \dots & w_{586\,624} \end{array} \\ \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,586\,624} \\ a_{2,1} & a_{2,2} & \dots & a_{2,586\,624} \\ \vdots & \vdots & \ddots & \vdots \\ a_{224\,679,1} & a_{224\,679,2} & \dots & a_{224\,679,586\,624} \end{pmatrix} \end{array}$$

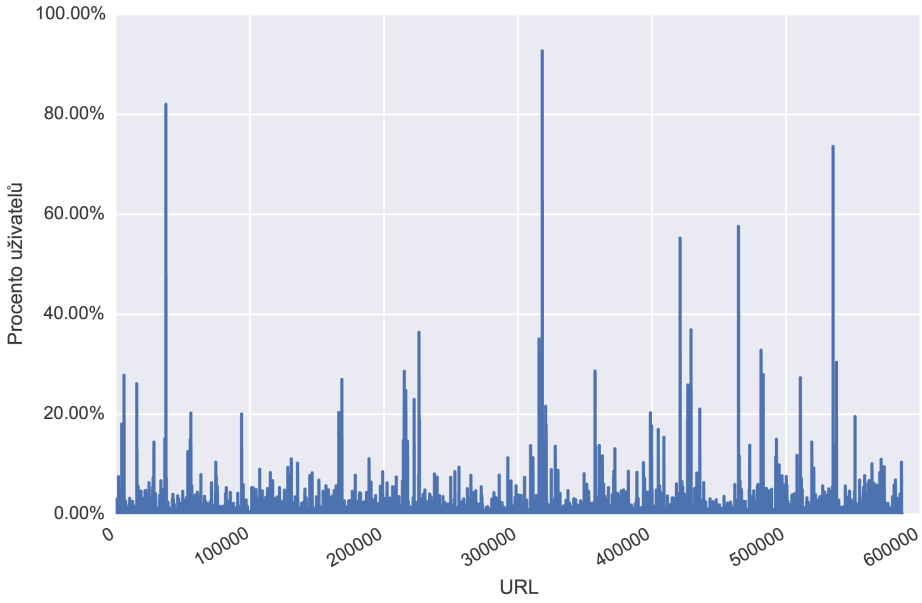
kde w jsou webové stránky (konkrétně pouze domény 2. řádu), u jsou uživatelé a a značí počet navštívení konkrétní domény uživatelem.

Aby se zjistilo, jakou mají data povahu, myšleno jejich rozdělení, rozložení a další charakteristiky, je vhodné provést několik popisných náhledů na data. Pro lepší orientaci ve výsledcích je využito grafického znázornění.

[1] Compressed Storrage Row. Standardní formát pro ukládání řídkých matic. Jsou uloženy pouze nenulové hodnoty.

3.1.1 Rozložení návštěvnosti podle URL stránek

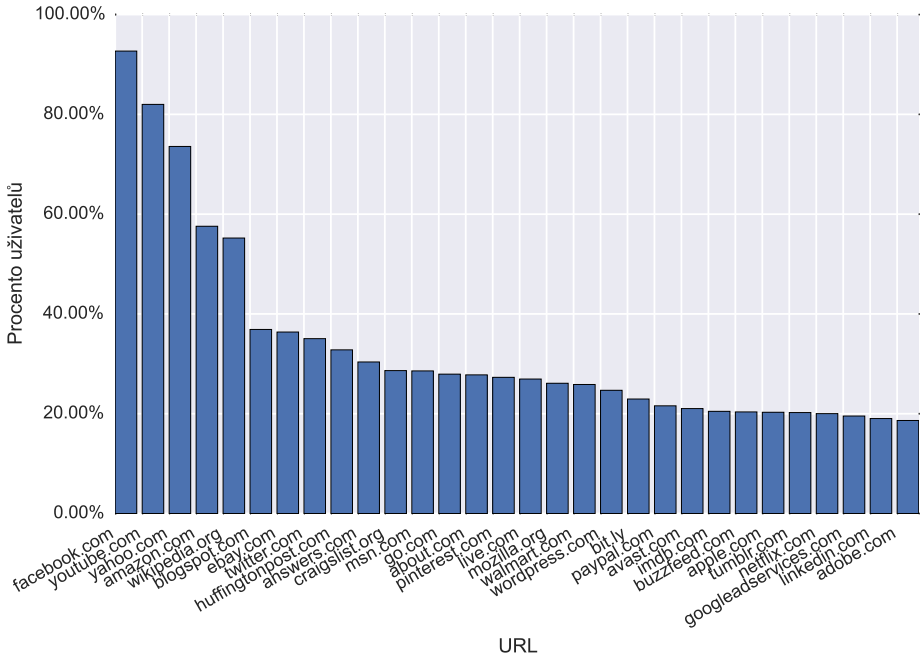
V prvním kroku je na data nahlédnuto z pohledu návštěvnosti URL stránek. Tedy kolik jedinečných uživatelů chodí na konkrétní URL adresu. U každého uživatele je započítána nenulová návštěva jen jednou. Hodnoty na ose Y jsou vytvořeny tak, že se sečetly sloupečky matice[2], které byly poté znormalizovány vůči celkovému počtu uživatelů.



Obrázek 3.1: Počet uživatelů na URL adrese

Hodnoty v grafu výše (viz obrázek 3.1) ukazují, že existuje velmi málo adres, které mají významnější návštěvnost. Proto v následujícím grafu (viz obrázek 3.2) je ukázka prvních 30 největších adres. Jednotlivé procentuální návštěvnosti jsou seřazeny sestupně.

[2]Každá nenulová hodnota matice byla nahrazena číslem 1.



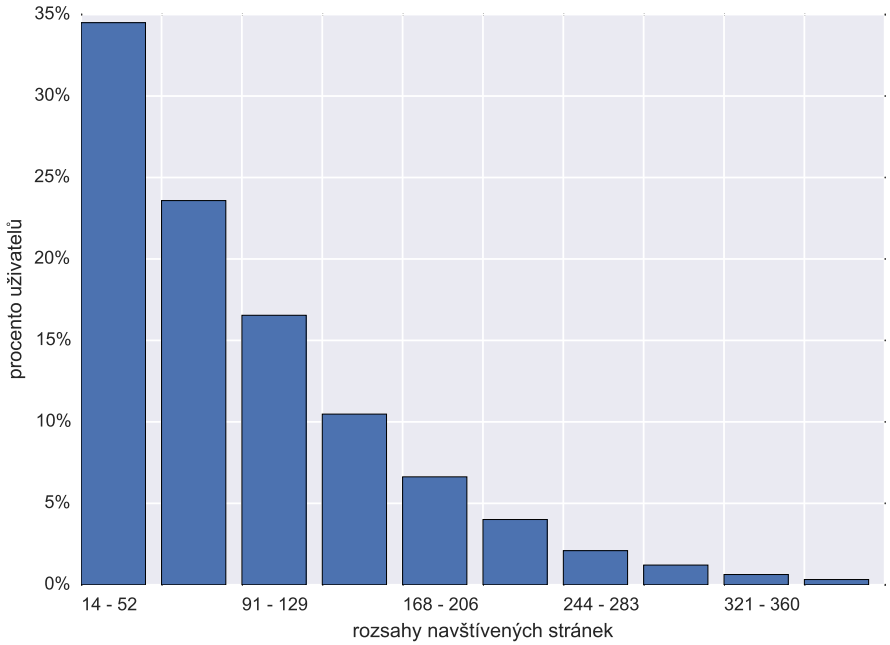
Obrázek 3.2: Počet uživatelů na 30 největších URL aresách

Dominující adresou je *facebook.com*. Z letmého pohledu lze rozpoznat další často navštěvované weby, které jsou všeobecně známé. Bohužel tyto „velké“ webové stránky, na které chodí téměř všichni uživatelé jsou pro diskriminaci zcela nepoužitelné. Na tyto stránky chodí významná většina uživatelů, a tedy významnost těchto domén je tímto razantně ponížena. Podobně je to i s těmi webovými stránkami, na které chodí velmi málo uživatelů.

3.1.2 Rozložení návštěvnosti podle navštívených stránek

V druhém kroku je vhodné diskutovat náhled na data z pohledu uživatele. Nejvhodnější metrikou se zde jeví měření rozsahu navštívených stránek pro jednotlivé skupiny uživatelů. Matice je nyní sečtena po řádcích[3], čímž vznikl vektor znázorňující počet různých navštívených domén pro každého uživatele. Protože je ale uživatelů hodně a žádný z nich nenese významnější popisnou informaci[4], je zde vhodné tyto hodnoty z vektoru shromáždit pomocí histogramu, kde lépe vyniknou jednotlivé skupiny obyvatel.

[3]Každá nenulová hodnota byla nahrazena číslem 1.
[4]Uživatelé byli anonymizováni.



Obrázek 3.3: Počet navštívených stránek uživateli

V grafu výše (viz obrázek 3.3) jsou jednotlivé počty navštívených stránek uživateli sloučeny do 10 tříd. První a nejpočetnější třída reprezentuje tu skupinu uživatelů, ve které každý uživatel chodí na 14 až 52 unikátních webových stránek (domén). Těchto uživatelů je přibližně 34,5 %. V poslední třídě je 0,32 % uživatelů, kteří chodí na 360 až 398 stránek[5]. 10 tříd bylo vybráno pouze jako názorná ukázka, protože při větším počtu tříd není dostatečně čitelná informace o počtu navštívených stránek. Tendence (trend) grafu tímto není pozměněna.

[5]Číslo 398 není v grafu z důvodu přehlednosti ostatních čísel zobrazeno. Jedná se o maximální počet různých webových domén, které byly navštíveny alespoň jedním uživatelem.

3.2 Předzpracování dat

Vzhledem k tomu, že povaha dat není ideální, je nutné data jistým způsobem upravit, aby další výsledky měly relevantnější charakter.

3.2.1 Redukce počtu URL

Redukcí počtu, neboli ořezáním URL je myšleno to, že z celého seznamu domén budou odstraněny ty adresy, které nevyhoví následujícím požadavkům. Ve své podstatě budou vymazány konkrétní domény i příslušné slupce v matici četností. Úkol ořezání domén je z důvodu přehlednosti rozdělen na dva po sobě jdoucí kroky.

Prvotní redukce

Při pohledu na obrázek 3.1 a seřazení hodnot sestupně zjistíme, že počet menších hodnot je výrazně větší než počet větších hodnot. Tato skutečnost nás musí vést k zamyšlení, jaké že webové stránky jsou dostatečně popisné pro další analýzu.

Již bylo zmíněno v odstavci 3.1.1, že dominantní adresy[6] a málo významné adresy[7] jsou pro další zpracování spíše nevyhovující. Na tomto základě je na místě tato nežádoucí data společně s adresami odstranit. Motivace k tomuto by měla být taková, že zanedbáním nediskriminujících adres získáme odfiltrované více diskriminující adresy.

Po prozkoumání výpisu domén (seřazených sestupně dle návštěvnosti) jsme dospěli k závěru, že u hranice 10 % návštěvnosti (odpovídá přibližně 22 000 uživatelů navštěvujících konkrétní doménu) může být pomyslná oddělující linie pro obecné a zájmově užší webové stránky. Mezi domény obecnějšího charakteru můžeme zařadit *facebook.com*, *youtube.com* či *amazon.com*. Zájmově užší doménou můžeme nazvat weby podobné *indeed.com*[8] či *foodnetwork.com*[9]. Tedy webové stránky, kde návštěva takového webu o uživateli už něco vypovídá.

Na druhé straně lze mluvit o těch webových stránkách, na které chodí naopak velmi malý počet uživatelů. Tyto webové stránky jsou převážně soukromého charakteru (př. lokální firma). V daném měřítku přibližně půl milionu různých domén jsou tyto malé weby víceméně nevýznamné. Rozhodli jsme se, že webová stránka

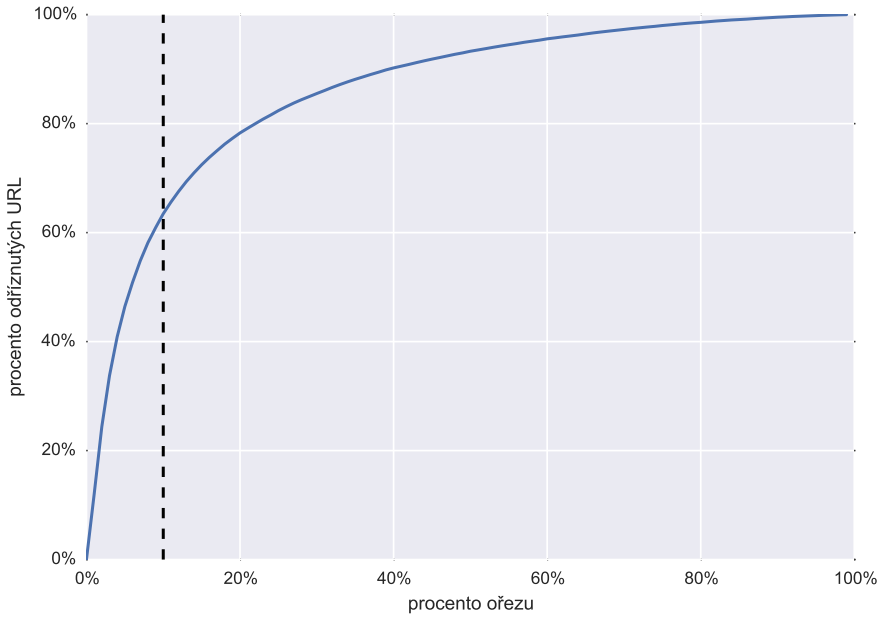
[6] Adresy na které chodí větší počet uživatelů.
[7] Významné z hlediska počtu návštěvníků.
[8] Portál s nabídkami práce.
[9] Portál s recepty na vaření.

významnějšího charakteru (lze se z ní dozvědět nějaká podstatná informace) může začínat na 10 různých návštěvních.

Na základě těchto úvah jsme odstranili (neboli nepoužili v dalších výpočtech) ty webové stránky, které jsou navštěvovány větším než 10% množstvím populace a také ty domény, na které chodí méně než 10 lidí. Z celkového počtu 586 624 webových adres zůstalo po této redukci pouze 102 453.

Redukce pro získání větší vypovídající hodnoty

V předchozím kroku jsme poměrně hrubším sítem odstranili několik zjevně nediskriminujících domén. V tomto kroku se ale zaměříme na ponechání pouze těch domén, které z hlediska pohledu na celkovou populaci mohou mít významnější charakter. Tento charakter lze určit například podle toho, jak moc je daná doména významná v rámci počtu návštěv pro jednoho uživatele. Existují domény webových stránek, na které uživatel chodí opakovaně a tyto webové stránky mohou tvořit jistou významnost v rámci oblíbenosti webu. Hranici mezi významnými a méně významnými doménami pro jednoho uživatele jsme určili na 10 %. Tedy ty adresy, které tvoří alespoň 10 % ze všech navštívených webů daným uživatelem jsou pro něj pravděpodobně významné. Ostatní adresy jsme v dalších výpočtech zanedbali. Zůstaly tedy pouze ty domény, které tvoří alespoň 10 % ze všech návštěv pro alespoň jednoho uživatele.



Obrázek 3.4: Procento odstraněných stránek v závislosti na procentu ořezu

V grafu na předchozí stránce (viz obrázek 3.4) je ukázáno, kolik procent webových stránek bude odstraněno (osa Y) v závislosti na procentu ořezu významných či nevýznamných adres pro alespoň jednoho uživatele (osa X). V grafu je též ukázána i přerušovaná linie znázorňující výše zmíněný ořez na 10 %. Je patrné, že tímto ořezem bude odstraněno veliké množství domén. Na druhou stranu zůstanou pouze ty domény, které jsou výrazně významnější pro měřenou populaci. Po odstranění těchto méně významných adres zůstane 37 441 z původních 102 453 adres.

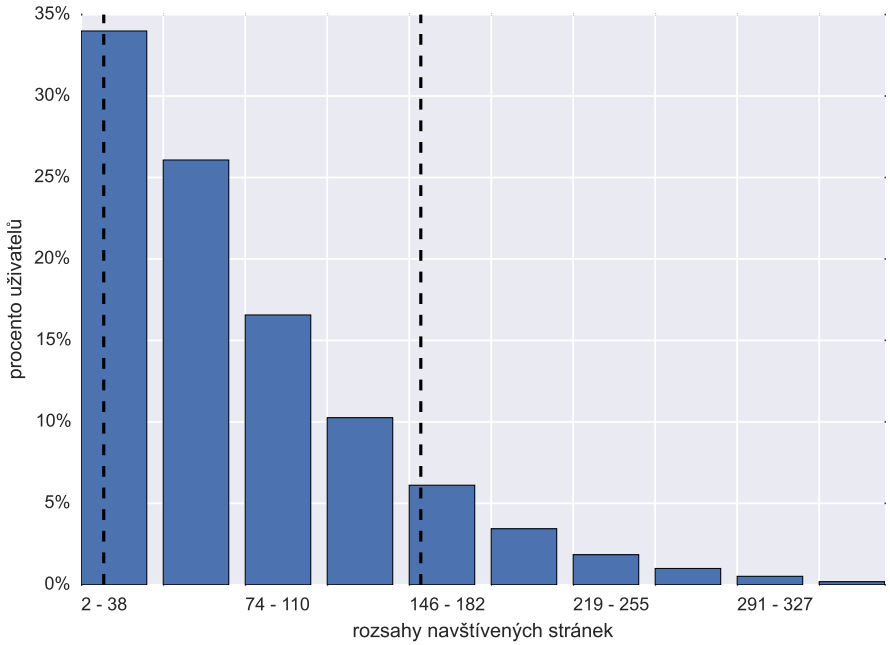
3.2.2 Redukce počtu uživatelů

Podobně jako jsme redukovali URL, nyní přistoupíme k redukci uživatelů. Motivací je to, že v dané populaci existují uživatelé, kteří chodí na velmi velké množství různých domén. Různorodost zájmů těchto uživatelů je poměrně veliká. Také existují uživatelé, kteří naopak chodí na velmi malé množství domén. Obě tyto skupiny by mohly negativně ovlivnit další zpracování.

V rámci zachování zájmu o nalezení obecných kategorií jsme nuceni přistoupit i k redukci uživatelů. Málo různorodí uživatelé nemají takovou vypovídající hodnotu, neboť jejich zájmy na internetu jsou tvořeny pouze malými shluky. Naopak uživatelé s příliš širokými zájmy by bylo později obtížné zakategorizovat. Z těchto důvodů budou odstraněny tyto dvě skupiny uživatelů.

Vycházíme z první části předešlé sekce (viz 3.2.1). Jednotlivé domény byly redukovány dle počtu návštěvníků na hranici 10 % ze strany velkých domén a na hranici 10 uživatelů ze strany domén s nižšími počty návštěvníků. Je vhodné zakomponovat redukci uživatelů ihned po tomto odstranění domén. Důvod je následující. Při redukci domén se změní počty unikátních navštívených domén uživateli (viz obrázek 3.3, který zobrazuje rozložení navštívených stránek bez jakéhokoliv odstranění). Po aplikování ořezu počtu domén se počty navštívených stránek změní následovně (viz obrázek 3.5 na další stránce). V grafu se změnily hlavně rozsahy navštívených stránek. Důležité rozsahy jsou u první a čtvrté třídy.

Po prozkoumání tohoto grafu jsme dospěli k závěru, že relevantnější informace dostaneme tehdy, když odstraníme ty uživatele, kteří chodí na méně než 10 a více než 150 různých domén. Tedy z původního počtu 224 679 uživatelů jsme dostali 191 676 uživatelů, kteří nemají tolik extrémní chování. Spodní (levý) a horní (pravý) práh jsou na obrázku vykresleny přerušovanou čárou. Uživatelé spadající do těchto mezí byli zachováni.



Obrázek 3.5: Počet navštívených stránek uživateli po odstranění nežádoucích domén

Po redukci uživatelů následovala další redukce domén na 10 % (viz druhá část předešlé sekce 3.2.1).

Z původního rozměru matice (224 679 uživatelů × 586 624 URL) jsme získali po celkovém ořezání matici velikosti 191 676 uživatelů × 37 441 URL, což je úspora přibližně 94,6 % buněk při zachování (ba zlepšení) charakteru dat. Pro upřesnění je vhodné dodat, že jsme tímto ušetřili přibližně 57,7 % dat na disku. Vzhledem k tomu, že je matice stále velmi řídká (přibližně 99,875 % nulových buněk), je pro další výpočty zachován formát CSR.

3.3 Metodika

V této části jsou popsány nejběžnější algoritmy a postupy pro zpracování dat povahy matice četností. Všechny zde zmíněné algoritmy jsme na datech vyzkoušeli, ale některé nevedly k požadovaným výsledkům.

Vzhledem k nejasné cestě k cíli jsme se rozhodli, že nejvhodnějším nástrojem pro zpracování bude programovací jazyk Python doplněný o statistické a výpočetní knihovny v čele s *scikit-learn* [17] a *SciPy* [10]. Tyto knihovny jsou velmi jednoduché k použití, obsahují optimalizované algoritmy a mají velmi podrobnou dokumentaci. Spolu s velmi jednoduchou syntaxí Pythonu je zde vytvořeno prostředí prospívající relativně rychlému vývoji a testování většího počtu experimentů. Mezi nevýhody tohoto řešení jistě patří rychlost Pythonu. Jedná se o interpretovaný jazyk, který je vhodný hlavně k zaznamenání myšlenek a algoritmů.

Většina výpočtů byla prováděna v MetaCentru[10] hlavně vzhledem k většímu objemu dat.

3.3.1 Algoritmus TF-IDF

TF-IDF[11] (zkratka pro anglická slova mající význam četnosti slova v dokumentu a převrácené četnosti slova ve všech dokumentech) je numerická statistika zohledňující důležitost slov v daném korpusu[12] [16]. V našem případě je korpus tvořen maticí četností. Dokumentem je zde nazván uživatel (řádek matice) a slovem je zde zastoupena konkrétní URL doména (sloupec matice).

Vysvětlení pojmů:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{3.1}$$

kde $n_{i,j}$ je počet výskytů slova t_i (webu) v dokumentu d_j (pro uživatele). Každý počet výskytů je vydělen součtem všech výskytů slov v celém dokumentu d_j.

$$idf_i = \log \frac{|D|}{|\{j : t_i \in d_j\}|} \tag{3.2}$$

[10] Jedná se o sdružení superpočítačů rozmístěných po celé České rebublice. Lze si zde na omezenou dobu zarezervovat veliké množství výpočetních prostředků.

[11] Term Frequency - Inverse Document Frequency.

[12] Rozsáhlý soubor textů (dokumentů).

kde $|D|$ je počet dokumentů (uživatelů) a $|\{j : t_i \in d_j\}|$ je počet dokumentů obsahujících slovo t_i [16]. Poté se finální hodnota spočítá následovně:

$$tfidf_{i,j} = tf_{i,j} \cdot idf_i \tag{3.3}$$

Výsledkem je matice stejné velikosti jako původní matice četností. Tato matice reprezentuje významnost každého slova v dokumentu napříč všemi dokumenty. Tedy pro každé slovo v každém dokumentu existuje číselná hodnota, která pro větší čísla značí větší důležitost daného slova v dokumentu a pro menší čísla důležitost menší. Po seřazení tohoto vektoru (řádek $tfidf$ matice) získáme mapováním slova, která jsou pro daný dokument významná, neboli diskriminující.

3.3.2 Algoritmus K-means

K-means[13] je clusterovací (shlukovací) algoritmus vytvářející k disjunktních clusterů (shluků). Iterativně minimalizuje odchylky centroidů (středů) od bodů v daném shluku. Algoritmus náhodně[14] umístí předem zvolené k bodů (centroidů) do daného prostoru a přiřadí nejbližší body z okolí k danému centroidu v závislosti na zvolené metrice. V dalších iteracích umístí každý centroid do středu (průměru) daného clusteru a tento postup opakuje do té doby, dokud se mění rozložení bodů napříč clustery. Cílem je dosáhnout co nejmenších rozdílů centroidu od bodů uvnitř clusterů [28].

Předpoklady

Seznam bodů $x = \{x_1, \ldots, x_n\} \in \mathbb{R}^m$,
číslo $k \in \mathbb{N}; k \leq n$
a $S = \{S_1, \ldots, S_k\}$ jakožto seznam clusterů.

Hledá se lokální optimum následovně:

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 \tag{3.4}$$

kde místo $\|\cdot\|^2$ může být libovolná metrika.

[13] Shlukovací algoritmus.
[14] Existují efektivnější metody pro počáteční inicializaci, například algoritmus K-means++ [1].

Místo metriky lze i použít tzv. kosinovou podobnost určující svírající úhel mezi dvěma vektory (body od počátku).

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{m} A_i \times B_i}{\sqrt{\sum_{i=1}^{m}(A_i)^2} \times \sqrt{\sum_{i=1}^{m}(B_i)^2}} \tag{3.5}$$

kde $A, B \in \mathbb{R}^m$ jsou dva body v prostoru [27].

Po nalezení lokálního optima algoritmus končí s tím, že je původní prostor bodů rozdělen do k clusterů. Vzdálenost centroidu ke všem bodům v daném clusteru je nejmenší možná pro daný počet k. Vzhledem k tomu, že je nalezeno pouze lokální optimum určeno počátečním rozdělením, je vhodné algoritmus spustit vícekrát s jiným počátečním rozdělením a brát v úvahu pouze ten výsledek, který měl nejmenší součet všech kvadrátů vzdáleností. Tímto opakováním lze najít takové lokální optimum, které je nejlepší možné pro daný počet opakování (vhodné jsou desítky až stovky).

3.3.3 Algoritmus PCA

PCA[15], neboli analýza hlavních komponent, je statistická procedura používající ortogonální transformaci k dekorelaci dat. Používá se ke snížení dimenze dat s co nejmenší ztrátou informace [31].

PCA je matematicky definováno jako ortogonální lineární transformace, která transformuje data do nového souřadnicového systému [31]. Hlavní komponenty (principal components) dávají nekorelované faktory, které jsou uspořádány sestupně dle rozptylu (variance) [32]. Největší komponenta leží na první ose nového souřadnicového systému, druhá komponenta na druhé atd. [31]. K získání redukovaných dat se používá SVD[16] rozklad:

$$A = USV^T \tag{3.6}$$

kde $A \in \mathbb{R}^{m,n}$ je původní matice četností (vhodnější je analyzovat matici transformovanou pomocí TF-IDF), $S \in \mathbb{R}^{m,n}$ je diagonální matice obsahující singulární čísla seřazená sestupně, $U \in \mathbb{R}^{m,m}$ a $V \in \mathbb{R}^{n,n}$ jsou ortogonální matice [26, kapitola 7]. Pro redukci dimenze jsou vhodné ta singulární čísla, která jsou větší než 1 [21]. Požadovaná dimenze je získána tak, že se v matici S nechá pouze tolik singulárních čísel, kolik je požadovaná dimenze (ostatní se vynulují), a zpětně se vynásobí matice následovně:

[15]Principal Component Analysis.
[16]Singular Value Decomposition.

$$A_R = US_RV^T \tag{3.7}$$

kde $S_R \subseteq S$ a $A_R \in \mathbb{R}^{m,r}$, kde r je požadovaná dimenze, protože vzniknou nulové sloupečky, které je vhodné odstranit.

Tímto lze získat matici A_R redukované (požadované) dimenze, která by měla mít co nejmenší ztrátu informace oproti původní matici A.

3.3.4 Algoritmus LSA

S využitím znalostí o SVD (viz sekce 3.3.3) na řadu přichází LSA[17]. Jedná se o techniku zpracování přirozeného jazyka [30], která dokáže analyzovat vztahy mezi dokumenty (uživateli) a slovy (URL) [29]. Tato metoda má pomoci potlačit nežádoucí důsledky synonymie. Je založena na algebraickém SVD rozkladu, kde můžeme drasticky snížit dimenzi, a tím získat efektivnější výpočty a nalezení významných podobností mezi dokumenty a slovy. SVD je puštěno nad maticí četností (viz část 3.3.1), která může být transformována pomocí TF-IDF.

$$U \cdot S \cdot V^T = svd(A) \simeq A \tag{3.8}$$

neboli

$$\begin{pmatrix} \begin{pmatrix} | \\ u_1 \\ | \end{pmatrix} & \dots & \begin{pmatrix} | \\ u_r \\ | \end{pmatrix} \end{pmatrix} \cdot \begin{pmatrix} s_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & s_r \end{pmatrix} \cdot \begin{pmatrix} \begin{pmatrix} - & v_1 & - \end{pmatrix} \\ \vdots \\ \begin{pmatrix} - & v_r & - \end{pmatrix} \end{pmatrix} \simeq \begin{pmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \dots & a_{m,n} \end{pmatrix} \tag{3.9}$$

kde $A \in \mathbb{R}^{m,n}$ je matice četností a $S \in \mathbb{R}^{r,r}$ je diagonální matice obsahující singulární čísla, kde r je označení pro redukovanou dimenzi. Pokud budeme uvažovat matici četností ve tvaru takovém, že na sloupečcích jsou jednotlivá slova (URL) a řádky jsou jednotlivé dokumenty (uživatelé), tak matice $U \in \mathbb{R}^{m,r}$ reprezentuje vztah mezi dokumenty a „nějakými“ kategoriemi, které jsou vytvořeny podobnostmi mezi slovy a matice $V \in \mathbb{R}^{r,n}$ reprezentuje vztah opačný, tedy vztah mezi jednotlivými doménami vůči „nějakým“ kategoriím vytvořených z podobností uživatelů[18]. Slovem „nějaký“

[17] Latent Semantic Analysis.

[18] Vzhledem k požadované nižší dimenzi dat $r \leq min\{m, n\}$ lze velikosti matic U, S, V omezit pomocí r, přestože SVD rozklad to ve své původní formě neumožňuje. Je nutné použít tzv. redukovaný SVD rozklad, díky kterému není nutné počítat plné SVD, které bychom následně ořezali na příslušnou dimenzi.

je myšleno to, že výsledné kategorie nelze předem odhadnout. Jsou vytvořeny shlukováním podobností mezi jednotlivými dokumenty či slovy.

Matice U, V (singulární vektory) promítají původní data do nových prostorů, ve kterých vynikne povaha dat.

3.3.5 Algoritmus pLSA

pLSA[19] je statistická technika ideově vycházející z LSA, avšak stojící na jiných podkladech. Zatímco LSA (potažmo SVD) využívá L2 nebo Frobeniovu normu [26, kapitola 7.3], tak pLSA maximalizuje věrohodnost (likelihood) [6]. Jedná se o pravděpodobnostní topic-model systém, snažící se nalézt skryté (latentní) „topics“ (témata) shlukující dokumenty (uživatele) nebo slova (URL) [7, 22]. Startovním bodem pLSA je statistický model zvaný *aspect model* [9]. Tento *aspect model* zohledňuje společné výskyty dat nad asociovanými třídami proměnné $z = \{z_1, \ldots, z_k\}$, neboli tématy [7]. Algoritmus se učí pomocí EM[20] algoritmu [3]. První fáze algoritmu (E - expectation) počítá posteriorní pravděpodobnosti pro latentní proměnné a druhá fáze (M - maximization) aktualizuje parametry.

E-krok:

$$P(z_k|d_i, w_j) = \frac{P(w_j|z_k)P(z_k|d_i)}{\sum_{k'=1}^{K} P(w_j|z_{k'})P(z_{k'}|d_i)} \tag{3.10}$$

M-krok:

$$P(w_j|z_k) = \frac{\sum_{i=1}^{M} n(d_i, w_j)P(z_k|d_i, w_j)}{\sum_{i=1}^{M}\sum_{j'=1}^{N} n(d_i, w_{j'})P(z_k|d_i, w_{j'})} \tag{3.11a}$$

$$P(z_k|d_i) = \frac{\sum_{j=1}^{N} n(d_i, w_j)P(z_k|d_i, w_j)}{\sum_{j=1}^{N} n(d_i, w_j)} \tag{3.11b}$$

kde $z = \{z_1, \ldots, z_k\}$ jsou latentní témata, $d = \{d_1, \ldots, d_m\}$ jsou dokumenty (uživatelé), $w = \{w_1, \ldots, w_n\}$ jsou slova (URL) a $n(d, w)$ je hodnota z matice četností [2]. $P(w|z)$ znázorňuje distribuci slov přes jednotlivá témata a $P(z|d)$ znázorňuje distribuci témat přes jednotlivé uživatele.

Každou iterací pLSA algoritmus maximalizuje věrohodnost (loglikelihood) následovně:

$$\log L = \sum_{i=1}^{M}\sum_{j=1}^{N} n(d_i, w_j) \log\left(\sum_{k=1}^{K} P(z_j|d_i)P(w_j|z_k)\right) \tag{3.12}$$

[19] Probabilistic Latent Semantic Analysis.
[20] Expectation–Maximization.

Důležitými omezeními jsou následující omezující podmínky:

$$\sum_{j=1}^{N} P(w_j|z_k) = 1 \tag{3.13a}$$

$$\sum_{k=1}^{K} P(z_k|d_i) = 1 \tag{3.13b}$$

které zajistí správnou normalizaci.

Výsledkem tohoto postupu je získání distribucí slov (URL) nad jednotlivými tématy. Tyto distribuce (pravděpodobnostní rozdělení) významně definují jednotlivá témata, jež není lehké jednotnou formou popsat, protože se jedná o modely směsí (mixture model), které ve své podstatě vůbec nemusejí reflektovat jednotný popisný systém. V sekci 4.3 je tato problematika rozebrána podrobněji. Dalším výsledkem pLSA je distribuce témat přes jednotlivé uživatele. Tedy pro každého uživatele (z trénovací sady) je známo jeho pravděpodobnostní rozdělení do nalezených témat. Tato informace je vhodná k tomu, abychom zjistili více informací o stávajících uživatelích. Pro případ, že budeme potřebovat zařadit nového uživatele d do natrénovaného modelu, je možné využít proceduru *folding-in* [8], která je inkrementovanou variantou EM algoritmu, přičemž v M-kroku se nemění již natrénovaná $P(w|z)$. Tímto dostaneme distribuci témat pro nové uživatele, neboli jejich umístění do prostoru témat.

3.4 Nevydařené experimenty

V této části práce jsou shrnuty některé postupy, které nakonec nebyly použity k finálnímu zpracování, protože jejich výsledky nebyly dostatečně uspokojivé. Přesto je vhodné tyto experimenty zmínit, protože jejich výsledky byly velmi nápomocné v hledání vhodnější cesty při analýze dat.

3.4.1 Shlukovací algoritmy, PCA, LSA

Prvním nápadem při hledání kategorií jsou metody shlukování (clusterování). Jedná se o statistické metody sloužící ke klasifikaci objektů. Shlukovací metody rozdělují vzorky do skupin, ve kterých jsou si tyto vzorky nejpodobnější z hlediska zvolené metriky. Jedním z významných shlukovacích algoritmů je K-means popsaný v části 3.3.2.

Aby algoritmus K-means dával rozumné výsledky bylo nutné nejprve data transformovat pomocí TF-IDF algoritmu. Tím data měla zároveň znormalizované řádky. Cílem bylo nalezení shluků webových stránek. Vzhledem k velikosti matice nebylo možné spuštění klasické K-means implementace [18] na plné velikosti matice. Následovala částečná redukce této matice a využití rychlejší implementace *MiniBatchKmeans* [19, 20]. Nyní již bylo možné puštění K-means nad daty, výsledky nebyly nijak přesvědčivé. Hlavním problémem bylo určení k (počtu shluků) a to, že některé vytvořené shluky byly obrovské v porovnání s jinými. Tyto problémy umí řešit algoritmy jako jsou *Afinní propagace* [25] či *DBSCAN* [5], které dokáží odhadnout k a zároveň řešit hustotu nalezených shluků, ale jsou výpočetně mnohem náročnější.

Při prozkoumání výsledků získaných pomocí výše zmíněných shlukovacích algoritmů jsme naráželi stále na ten samý problém, a tedy velmi husté umístění bodů (domén) okolo počátku. Proto vhodnějším řešením bylo promítnutí daného prostoru do jiné dimenze, kde souřadné osy budou korelovat s hlavními komponentami prostoru dat. Problém promítnutí dat do nového prostoru reflektujícího hlavní komponenty systému řeší například algoritmus PCA blíže popsaný v sekci 3.3.3. Pomocí tohoto algoritmu bylo možné snížit dimenzi prostoru se zachováním maximální informační hodnoty. Zpětně rekonstruovaná matice nižší dimenze byla opět pomocí shlukovacích algoritmů (primárně K-means) zpracována, ale mnohem lepší výsledky jsme dostali při použití LSA, odkud jsme zpracovali pouze matici V z SVD rozkladu. Při porovnání výstupů vycházejících z TF-IDF matice a binární matice[21] byly

[21] Binární matice je taková, která má místo nenulových hodnot číslo 1 a jinde 0.

zaznamenány výrazně lepší výsledky právě při použití binární matice, a tedy tato matice byla použita v následujících výpočtech. Shlukovací algoritmy vrátily mnoho shluků (clusterů), které byly rozprostřené po celém prostoru, ale našly také velmi úzce zaměřené clustery. Při odfiltrování clusterů s širším záběrem[22] zůstaly pouze ty clustery, které splňovali následující podmínky v konjuknci:

- Nejnavštěvovanější stránka z clusteru byla navštěvována alespoň 10 % uživateli z daného tématu.
- Tuto nejnavštěvovanější stránku navštívilo alespoň 10 uživatelů.
- Celkový počet uživatelů v clusteru je alespoň 15.

Výstupem byly pouze takové shluky webových stránek, které svým obsahem byly velmi úzce specializovány. Níže v tabulce je ukázka dvou náhodně vybraných clusterů, které mají velmi úzké zaměření:

Tabulka 3.1: Velmi úzce specializované clustery

Cluster 1	Cluster 2
getdogsex.com	cooks.com
3animalsextube.com	allrecipes.com
pornsocket.com	yummly.com
zootube365.com	bettycrocker.com
animalsexfun.com	myrecipes.com
petsex.com	epicurious.com
homeanimaltube.com	kraftrecipes.com

Cluster 2 zobrazuje stránky zabývající se pouze vařením. Tedy bychom mohli jásat, že přesně takové specializace hledáme. Bohužel zbylých cca 70 % clusterů se zabývají sexuálním obsahem a kvůli velmi úzké specializaci byly nalezeny i tak nechutné clustery podobné Clusteru 1, které se zabývají sexuálními stránkami zaměřenými na zoofilii. Nutno podotknout, že takto úzce specializované clustery obsahovaly pouze přibližně 2 % ze všech webových stránek v dostupném korpusu, a právě tyto stránky byly tak moc úzce specializovány, že neměly vypovídající hodnotu pro celý dataset. Z těchto důvodů byla pozornost dalšího výzkumu upřena na postupy s jinými přístupy, mezi které patří například algoritmus pLSA. Dalším významným důvodem je fakt, že clusterovací algoritmy uspořádávají data do disjunktních skupin, které ale nedostatečně popisují různorodého uživatele[23]. Uživatel může mít zájmy zakategorizované do více skupin, ale tyto skupiny ve svých disjunktních podobách popsat uživatele (například pomocí URL) nedokáží, a je tedy nutné využít jiného přístupu.

[22]Širším záběrem je zde myšleno to, že v daném clusteru bylo velmi mnoho adres, které byly velmi málo navštěvovány populací daného clusteru.

[23]Myšleno z hlediska počtu různých zájmů.

3.4.2 Sériové pLSA

Bohužel v použitých knihovnách (*scikit-learn* a *ScpiPy*) není algoritmus pLSA implementován, a tedy bylo nutné použít jinou implementaci. Jako vhodné řešení byla zvolena implementace kódu [13], který je v následující části popsán.

Zdrojový kód 3.1: Implementace serializovaného pLSA

```
import numpy as np
...
def plsa(term_doc_matrix):
    p_z_d = np.zeros([number_of_documents,
                      number_of_topics])
    p_w_z = np.zeros([number_of_topics,
                      number_of_words)])
    p_z_d_w = np.zeros([number_of_documents,
                        number_of_words,
                        number_of_topics])
    for i in range(max_iter):
        # E-step
        for d_index in range(number_of_documents):
            for w_index in range(number_of_words):
                prob = p_z_d[d_index, :] * p_w_z[:, w_index]
                normalize(prob)
                p_z_d_w[d_index][w_index] = prob
        # M-step
        for z in range(number_of_topics):
            for w_index in range(number_of_words):
                s = 0
                for d_index in range(number_of_documents):
                    count = term_doc_matrix[d_index][w_index]
                    s = s + count * p_z_d_w[d_index,
                                            w_index,
                                            z]
                p_w_z[z][w_index] = s
            normalize(p_w_z[z])
        for d_index in range(number_of_documents):
            for z in range(number_of_topics):
                s = 0
                for w_index in range(number_of_words):
                    count = term_doc_matrix[d_index][w_index]
                    s = s + count * p_z_d_w[d_index,
                                            w_index,
                                            z]
                p_z_d[d_index][z] = s
            normalize(p_z_d[d_index])
```

Kód na předchozí stránce (viz algoritmus 3.1) je jednoduchou implementací EM a pLSA algoritmu. Vstupem je matice četností `term_doc_matrix` a výstupem jsou matice `p_z_d` (značící podmíněné pravděpodobnosti tématu a dokumentu $P(z|d)$) a matice `p_w_z` (značící podmíněné pravděpodobnosti slova a tématu $P(w|z)$). Níže jsou ukázány složitosti[24]:

Tabulka 3.2: Složitosti serializovaného algoritmu pLSA

časová složitost	paměťová náročnost
$\Omega(I \cdot D \cdot W \cdot (1 + 2Z))$	$\Omega(Z \cdot (D + W + D \cdot W) + D \cdot W)$

kde I je počet iterací, D je počet dokumentů (uživatelů), W je počet slov (URL) a Z je požadovaný počet témat. S těmito parametry tato implementace není vhodná pro větší matice. V případě použití této implementace narazíme jak na časový, tak i na paměťový problém. Již při menším datasetu výpočetní čas neúměrně rychle roste a algoritmus si zbytečně udržuje informace v matici `p_z_d_w` (značící posteriorní pravděpodobnost $P(z|d, w)$), jejíž rozměry náleží $\mathbb{R}^{D,W,Z}$.

Abychom zrychlili výpočet, je vhodné uvažovat o jeho paralelizaci. V originálním kódu [13] je i paralelní implementace téhož algoritmu, ovšem kvůli několika „drobnostem“ nesmírně paměťově náročná. Data (všechny matice se kterými se počítá) jsou kopírována ke každému procesu. Tedy paměťová náročnost tímto vzroste na $\Omega(P \cdot Z \cdot (D + W + D \cdot W) + P \cdot D \cdot W)$, kde P značí počet procesů a pro doplnění poslední $D \cdot W$ je matice četností, ale v dense[25] formátu, protože původní algoritmus nikterak nepracuje s CSR formátem matice četností.

[24] V tabulce 3.2 je sice znak Ω pro spodní odhad asymptotické složitosti, ale multiplikativní konstanty jsou zde poměrně důležité, a tedy nebyly odstraněny. Slouží pro pozdější detailnější porovnání s paralelní optimalizovanou verzí algoritmu pLSA.

[25] Formát matice, v němž jsou uloženy všechny buňky dané matice. Oproti CSR formátu jsou zde uloženy i nulové prvky.

Kapitola 4

Finální zpracování

Tato kapitola práce pojednává o postupech, které vedly k nejlepším výsledkům, a tedy kroky popsané v této části jsou nazvány finálními v zájmu cílů této práce. Kroky jsou provedeny postupně ve stejném pořadí, jako jsou číslovány sekce v této kapitole.

4.1 Redukce dat

Redukcí dat se poměrně obsáhle zabývá část 3.2. Přesto jsou v této sekci zopakovány postupy vedoucí ke sníženi dimenze dat jak u URL, tak u uživatelů.

Prvně je provedena redukce počtu URL stránek, tedy odstranění příliš velkých a příliš malých domén. Velikostí domény je zde myšlena její popularita, tedy počet unikátních návštěvníků. Po ořezu zůstanou pouze ty domény, které jsou navštěvovány maximálně 10 % populace a minimálně 10 unikátními návštěvníky. Vyjádřeno pomocí matematiky:

$$W' = \{w \in W; \quad 10 \leq |w| \leq 10\ \%\} \tag{4.1}$$

kde W jsou všechny domény, W' jsou domény, které zůstaly po ořezu a $|w|$ značí počet unikátních uživatelů pro konkrétní w jakožto URL. Pro připomínku 10 % populace je přibližně 22 000 uživatelů.

Další odstraňování je provedeno na uživatelích. Jak již bylo zmíněno v části 3.2.2, tak je vhodné ponechat pouze ty uživatele, kteří chodí na 10 až 150 různých domén.

$$U' = \{u \in U; \quad 10 \leq |u| \leq 150\} \tag{4.2}$$

kde U jsou všichni uživatelé, U' jsou jen ti uživatelé, kteří zůstali po odstranění a $|u|$ je počet navštívených různých domén pro konkrétního uživatele u.

Posledním krokem je ponechání pouze významných domén, a tedy odstranění těch „méně“ významných. Hranice mezi nimi byla určena na 10 % (viz druhá část sekce 3.2.1). Tedy významnou doménou je taková stránka, která tvoří alespoň 10 % návštěvnosti pro alespoň jednoho uživatele.

$$W'' = \{w \in W'; \quad \exists u \in U' : V(u, w) \geq 10\ \%\} \tag{4.3}$$

kde U' jsou uživatelé z předešlého odstavce, W' jsou domény z předešlého odstavce a W'' je konečný počet významných domén. Ve výrazu 4.3 se říká, že množinu konečných webových stránek tvoří takové domény u nichž existuje alespoň jeden uživatel u, pro kterého je významnost (funkce $V(u, w)$ vrací procentuální významnost dané domény w pro konkrétního uživatele u) větší než 10 %.

Po dokončení všech operací redukce dat je výsledná velikost matice
191 676 uživatelů × 37 441 URL z původní velikosti 224 679 uživatelů × 586 624 URL.

4.2 Paralelní pLSA

Vzhledem k tomu, že se topic-model algoritmus pLSA osvědčil jako velmi relevantní pro danou problematiku, tak jeho využití bylo víceméně jisté. Funkčnost algoritmu je popsána v části 3.3.5.

4.2.1 Popis paralelního algoritmu pLSA

Již bylo naznačeno v části 3.4.2, že je vhodnější použít paralelní verzi algoritmu. Bohužel ale paralelní implementace ze zmíněné části měla jisté neduhy, a proto bylo nutné paralelní algoritmus optimalizovat a zcela jej přepsat. Analýzou jednotlivých kroků bylo možné detekovat jistou redundanci vznikající při výpočtech. Paměťový problém (kopírování dat s každým procesem) je způsoben tím, že systémové volání `fork()` (které je implicitně voláno při paralelizaci) vytvoří nový nezávislý proces, který má své vlastní paměťové úložiště. Aby tento proces mohl pracovat s daty, je

nutné jejich zkopírování k procesu. Proto bylo nutné přijít s jiným řešením. Tímto řešením je ukládání dat do sdílené paměti. Vzhledem k tomu, že v MetaCentru puštěné úlohy pracují na několika různých procesorech, je využívání sdílené paměti poněkud obtížnější. Je nutné sdílet paměť přes diskové úložiště, ke kterému mají přístup všechny procesy na všech procesorech. Bylo tedy nutné využít `memmap`[1]. Díky tomu bylo možné pracovat s mnohem větší pamětí, než byla velikost operační paměti (RAM) při zachování relativně rychlého přístupu při sekvenčních operacích. Níže následuje ukázka zdrojového kódu ukazující inicializaci sdílené paměti:

Zdrojový kód 4.1: Inicializace sdílené paměti v paralelním pLSA

```
1  import numpy as np
2  global p_z_d, p_w_z, p_w_z_temp_array, p_z_w_di_array
3  p_z_d = np.memmap("p_z_d.mm", shape=(number_of_documents,
4                                       number_of_topic))
5  p_w_z = np.memmap("p_w_z.mm", shape=(number_of_topic,
6                                       number_of_words))
7  p_w_z_temp_array = np.memmap("p_w_z_temp_array.mm",
8                               shape=(number_of_chunks,
9                                      number_of_topic,
10                                     number_of_words))
11 p_z_w_di_array = np.memmap("p_z_w_di_array.mm",
12                            shape=(number_of_chunks,
13                                   number_of_topic,
14                                   number_of_words))
```

kde matice `p_z_d` $\in \mathbb{R}^{D,Z}$ je $P(z|d)$ a `p_w_z` $\in \mathbb{R}^{Z,W}$ je $P(w|z)$, kde W je počet slov (URL), D je počet dokumentů (uživatelů) a Z je počet témat. Matice `p_w_z_temp_array` $\in \mathbb{R}^{C,Z,W}$ slouží pro dočasné ukládání výsledků matice $P(w|z)$ z EM algoritmu a `p_z_w_di_array` $\in \mathbb{R}^{C,Z,W}$ obstarává pozici $P(z|w,d)$. Proměnná C značí počet částí (chunků), které jsou paralelně rozděleny mezi jednotlivé procesory. Tímto bychom měli mít vyřešený problém se sdílenou pamětí.

Dalším úkolem bylo rozdělení výpočetních částí v kódu tak, aby jednotlivé části (chunky) byly na sobě nezávislé, a tedy paralelizace byla vůbec možná.

Hlavní idea paralelizace pLSA Každý procesor dostane na starost část dokumentů (př. rozsah 1. až 30. dokument), které zpracuje a vrátí výslednou matici $P(w|z)$ pro dané dokumenty (1. až 30.). Po dokončení výpočtu všech částí se tyto matice $P(w|z)$ uložené v `p_w_z_temp_array` sečtou a výsledkem je nová matice $P(w|z)$ pro další iteraci algoritmu.

[1]Mapování diskové paměti jakožto operační paměti RAM.

Pro lepší pochopení souvislostí je níže bližší popis postupu:

1. Alokace sdílených matic. Vytvoření matic mapovaných na disk.
2. Výpočet intervalů mezi dokumenty (chunků C), které se později předají jednotlivým procesům.
3. Pro každou iteraci:
   (a) Vytvoření C procesů.
   (b) Každému z procesů se předá jedna část (chunk C).
   (c) Pro každou část (chunk), kde i náleží indexu dokumentu z intervalu přiřazeného dané části:
      i. **E-krok:**

$$\begin{aligned} P(z|w,d_i) &= P(z|d_i) \cdot P(w|z) + 10^{-8} \\ \begin{pmatrix} x_{1,1} & \dots & x_{1,W} \\ \vdots & \ddots & \vdots \\ x_{Z,1} & \dots & x_{Z,W} \end{pmatrix} &= \begin{pmatrix} a_1 \\ \vdots \\ a_Z \end{pmatrix} \cdot \begin{pmatrix} b_{1,1} & \dots & b_{1,W} \\ \vdots & \ddots & \vdots \\ b_{Z,1} & \dots & b_{Z,W} \end{pmatrix} + 10^{-8} \end{aligned} \tag{4.4}$$

      kde konstanta 10^{-8} je „renormalizační“ faktor (matice plná těchto čísel), který odstraní nulu vzniklou při součinu $P(z|d_i) \cdot P(w|z)$. Vzniklá nula by se totiž propagovala do dalších kroků a při normalizaci by se dělilo nulou.
      ii. Normalizace sloupečků $P(z|w,d_i)$ dle L1 normy.
      iii. **M-krok, část první:**

$$\begin{aligned} Y^{P(z|w,d_i)}_{n(w,d_i)} &= n(w|d_i) \cdot P(z|w,d_i) \\ \begin{pmatrix} y_{1,1} & \dots & y_{1,W} \\ \vdots & \ddots & \vdots \\ y_{Z,1} & \dots & y_{Z,W} \end{pmatrix} &= \begin{pmatrix} n_{D_i,1} & \dots & n_{D_i,W} \end{pmatrix} \cdot \begin{pmatrix} x_{1,1} & \dots & x_{1,W} \\ \vdots & \ddots & \vdots \\ x_{Z,1} & \dots & x_{Z,W} \end{pmatrix} \end{aligned} \tag{4.5}$$

      kde $Y^{P(z|w,d_i)}_{n(w,d_i)} \in \mathbb{R}^{Z,W}$ je dočasná proměnná a kde $n(w|d_i)$ je i-tý řádek (dokument, uživatel) z matice četností.

iv. **M-krok, část druhá:**

$$
\begin{aligned}
P(z|d_i) &= \sum_{j=1}^{W} Y_{n(w,d_i)}^{P(z|w,d_i)} \\
\begin{pmatrix} a_1 \\ \vdots \\ a_Z \end{pmatrix} &= \sum_{j=1}^{W} \begin{pmatrix} y_{1,1} & \dots & y_{1,W} \\ \vdots & \ddots & \vdots \\ y_{Z,1} & \dots & y_{Z,W} \end{pmatrix}
\end{aligned}
\tag{4.6}
$$

kde matice $Y_{n(w,d_i)}^{P(z|w,d_i)}$ je sečtena po sloupečcích.

v. Normalizace vektoru $P(z|d_i)$ dle L1 normy.

vi. **M-krok, část třetí:**

$$
\begin{aligned}
P(w|z)_{temp_c} &= P(w|z)_{temp_c} + Y_{n(w,d_i)}^{P(z|w,d_i)} \\
\begin{pmatrix} b_{1,1} & \dots & b_{1,W} \\ \vdots & \ddots & \vdots \\ b_{Z,1} & \dots & b_{Z,W} \end{pmatrix} &= \begin{pmatrix} b_{1,1} & \dots & b_{1,W} \\ \vdots & \ddots & \vdots \\ b_{Z,1} & \dots & b_{Z,W} \end{pmatrix} + \begin{pmatrix} y_{1,1} & \dots & y_{1,W} \\ \vdots & \ddots & \vdots \\ y_{Z,1} & \dots & y_{Z,W} \end{pmatrix}
\end{aligned}
\tag{4.7}
$$

kde v matici $P(w|z)_{temp_c}$ jsou akumulovány jednotlivé dílčí výpočty matice $P(w|z)$ a c je index části výpočtu (chunku).

(d) Všechny dílčí výsledky z jednotlivých částí (chunků) uložené v matici $P(w|z)_{temp_c}$, kde c odděluje výpočty z těchto částí (chunků). Tyto části jsou po dokončení všech procesů sečteny, a tím vznikne nová matice $P(w|z)$:

$$
\begin{aligned}
P(w|z) &= \sum_{c=1}^{C} P(w|z)_{temp_c} \\
\begin{pmatrix} b_{1,1} & \dots & b_{1,W} \\ \vdots & \ddots & \vdots \\ b_{Z,1} & \dots & b_{Z,W} \end{pmatrix} &= \sum_{c=1}^{C} \begin{pmatrix} b_{1,1} & \dots & b_{1,W} \\ \vdots & \ddots & \vdots \\ b_{Z,1} & \dots & b_{Z,W} \end{pmatrix}_c
\end{aligned}
\tag{4.8}
$$

kde C je počet částí (chunků, procesů).

4. Navrácení matic `p_z_d` a `p_w_z`.

Pro doplnění je níže konkrétní implementace paralelního EM algoritmu pLSA:

Zdrojový kód 4.2: Implementace paralelního EM algoritmu pLSA

```
1  import numpy as np
2  from sklearn.preprocessing import normalize
3  def do_EM(p):
4      start, stop = start_stop_intervals[p]
5      p_w_z_temp_array[p][:] = np.zeros([num_z, n_w_d.shape[1]])
6      p_z_w_di_array[p][:] = np.zeros([num_z, n_w_d.shape[1]])
7      for d_index in range(stop - start):
8          # E-step:
9          np.multiply(p_z_d[start + d_index][np.newaxis].T,
10                     p_w_z,
11                     out=p_z_w_di_array[p])
12         np.add(p_z_w_di_array[p], 1e-8,
13                out=p_z_w_di_array[p])
14         normalize(p_z_w_di_array[p], norm="l1", axis=0,
15                   copy=False)
16         # M-step 1:
17         data = n_w_d.data[n_w_d.indptr[start + d_index]:
18                           n_w_d.indptr[start + d_index + 1]]
19         y = 0. * p_z_w_di_array[p]
20         for j, idx in enumerate(n_w_d.indices[
21                                 n_w_d.indptr[
22                                  start + d_index]:
23                                 n_w_d.indptr[
24                                  start + d_index + 1]]):
25             y[:, idx] = data[j] * p_z_w_di_array[p][:, idx]
26         # M-step 2:
27         np.sum(y, axis=1,
28                out=p_z_d[start + d_index])
29         np.divide(p_z_d[start + d_index],
30                   p_z_d[start + d_index].sum(),
31                   out=p_z_d[start + d_index])
32         # M-step 3:
33         np.add(p_w_z_temp_array[p], y,
34                out=p_w_z_temp_array[p])
```

Na řádcích 17 až 25 zdrojového kódu 4.2 je implementováno násobení CSR matice (vektoru) a matice v dense formátu. Bohužel žádná z použitých knihoven tuto operaci nemá implementovanou, a tak bylo nutné tuto operaci implementovat ručně. S využitím znalosti o CSR maticích, které se skládají ze tří polí (data, indices, indptr), lze přistupovat pouze k nenulovým prvkům, a tím značně zrychlit výpočet. Též jsou zde hojně využívány knihovní matematické funkce s výstupním parametrem `out`, které jsou optimalizovány a výpočet je tímto též značně urychlen. Jak již bylo

zmíněno v popisu algoritmu u E-kroku (viz 3(c)i), tak „renormalizační“ faktor 10^{-8} je do výpočtu zahrnut kvůli tomu, aby nemohla vzniknout nula při násobení. Tento faktor ovšem nijak nezkresluje výsledek, protože konstanta 10^{-8} je přičtena v každé iteraci.

Abychom mohli měřit rychlost konvergence, je vhodné počítat na konci každé iterace hodnotu loglikelihoodu (logaritmu věrohodnosti). Tu lze počítat opět paralelně s podobnou ideou jako u EM algoritmu výše zmíněného.

Zdrojový kód 4.3: Implementace výpočtu loglikelihoodu

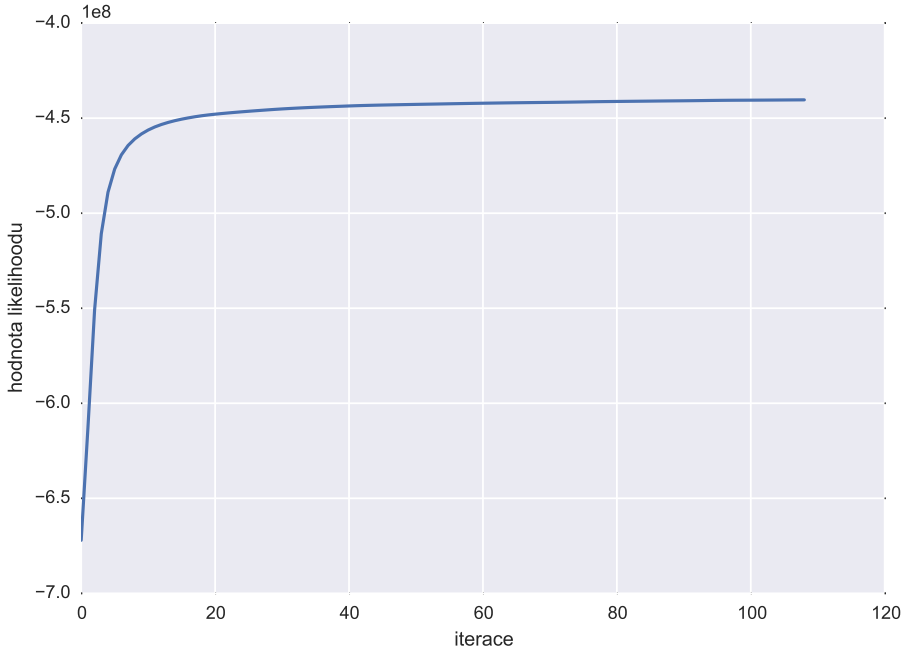
```
import numpy as np
def get_loglikehood(p):
    start, stop = start_stop_intervals[p]
    L = 0.
    for d_index in range(stop - start):
        log_p_wd = np.log10((p_z_d[start + d_index]
                                  [np.newaxis].T * p_w_z)
                            .sum(axis=0))
        data = n_w_d.data[n_w_d.indptr[start + d_index]:
                          n_w_d.indptr[start + d_index + 1]]
        for j, idx in enumerate(n_w_d.indices[
                                  n_w_d.indptr[
                                   start + d_index]:
                                  n_w_d.indptr[
                                   start + d_index + 1]]):
            L += data[j] * log_p_wd[idx]
    return L
```

kde funkce `get_loglikelihood` je volána pro každou část (chunk) a následně jsou výsledky z těchto částí sečteny do jedné hodnoty představující loglikelihood v dané iteraci.

4.2.2 Porovnání rychlosti

Popsaný algoritmus 4.2.1 díky své paralelizaci a minimalizaci redundantního kódu je mnohem rychlejší, než jeho sériová verze popsaná v části 3.4.2.

Rychlost konvergence popsatelná grafem loglikelihoodu je oproti sériové verzi stejná. Tento fakt je způsoben tím, že oba algoritmy (sériový i paralelní) provádí tytéž akce při výpočtu, ale v paralelní verzi jsou tyto akce provedeny rychleji.



Obrázek 4.1: Rychlost konvergence pLSA

Na obrázku výše (viz obrázek 4.1) je ukázán graf loglikelihoodu pro 108 iterací. Budeme-li hledět na relativní diferenci (změnu) loglikelihoodu mezi iteracemi (lze tuto diferenci připodobnit k procentuální změně), která se asymptoticky blíží nule, počítanou následovně:

$$\frac{L_i - L_{i-1}}{L_{i-1}} \tag{4.9}$$

kde L_i představuje aktuální hodnotu loglikelihoodu a L_{i-1} je hodnota loglikelihoodu z předchozí iterace, tak zjistíme, že hodnota relativní diference se přibližně od 60. iterace pohybuje na menších hodnotách než 10^{-5}. Lze tedy konstatovat, že rychlost konvergence je už natolik pomalá (menší než $0,00001$ % za iteraci), že výpočet dalších iterací už nemá razantní význam v přesnosti.

Kvalitu výpočtu a počet iterací lze také odhadnout pomocí výpočtu perplexity [7] nebo pomocí zpětného dosazení natrénovaných dokumentů (uživatelů) do procedury `folding-in`. Detailněji se jedná o postup takový, že se nejprve natrénuje $P(z|d)$ a $P(w|z)$ pomocí pLSA algoritmu. Poté se vyberou některé dokumenty (řádky matice četností) nad kterými byly trénovány zmíněné pravděpodobnosti a tyto „staré“ dokumenty se vloží na vstup metody `folding-in`, která vrátí $P'(z|d)$, což jsou nové distribuce témat pro uživatele. Porovnáním distribucí $P(z|d)$ z pLSA s nově zobrazenými distribucemi $P'(z|d)$ pomocí `folding-in` pro stejné dokumenty lze určit „přesnost“ trénování. Jedná se o rozdíl v distribucích témat pro konkrétní dokument. Opět je zde použit iterativní postup trénování s měřením loglikelihoodu. Křivka loglikelihoodu má podobnou tendenci jako při trénování, ale s tím rozdílem, že počáteční hodnotou loglikelihoodu je poslední hodnota z trénování pLSA. Po přibližně 120 iteracích změna loglikelihoodu je menší než 10^{-10} a rozdíl $P(z|d)$ a $P'(z|d)$ je v řádu 10^{-5}. Při větším počtu iterací při trénování lze tento rozdíl ještě snížit, ale jak je patrné z obrázku 4.1, tendence růstu není rychlá, a tedy přesnějšího výsledku se bude dosahovat jen velmi pomalu.

Pro exaktní porovnání rychlosti je vhodné uvažovat nad porovnáním složitostí[2] s verzí popsanou v části 3.4.2. V tabulce na další stránce (viz tabulka 4.1), kde poslední řádek (paralelní*) je algoritmus popsaný v této sekci 4.2.1, I je počet iterací, D je počet dokumentů (uživatelů), W je počet slov (domén), Z je počet témat, P je počet procesů a ε je aditivní konstanta zahrnující režii vzniklou se zpracováním paralelizmu. V tabulce 4.1 je u paralelní* implementace zobrazena paměťová náročnost pouze pro operační paměť RAM, kde $sparse(D \cdot W)$ značí matici četností v CSR formátu. Tato matice je sice kopírována ke každému procesu, ale vzhledem k její zanedbatelnější velikosti (s porovnáním s dense formátem) její přesunutí na disk nebylo nezbytné. Protože algoritmus využívá diskového úložiště pro uložení velkých matic, je vhodné dodat, že zabraná paměť[3] na tomto diskovém úložišti je $\Theta(Z \cdot (D + W + 2 \cdot C \cdot W))$, kde C značí počet částí (chunků) předaných k procesům[4].

[2] V tabulce 4.1 je sice znak Ω pro spodní odhad asymptotické složitosti, ale multiplikativní konstanty jsou zde poměrně důležité, a tedy nebyly odstraněny. Slouží pro detailnější porovnání „staré“ a optimalizované verze algoritmu pLSA.

[3] Znak Θ značí přesnou asymptotickou náročnost a opět zde z důvodu korektního porovnání nejsou odstraněny multiplikativní konstanty.

[4] Vzhledem k tomu, že C se v paralelní* implementaci rovná P, tak je paralelizmus využit na maximum, a je tedy dosaženo maximálního výkonu. Zato v paralelní implementaci [13] bylo vytvořeno D procesů, které byly postupně zpracovávány P procesory, a tedy vzniklá režie byla mnohonásobně vyšší než u paralelní* verze.

Tabulka 4.1: Porovnání složitostí implementací algoritmu pLSA

algoritmus	časová složitost	paměťová náročnost
sériový [13]	$\Omega(I \cdot D \cdot W \cdot (1 + 2Z))$	$\Omega(Z \cdot (D + W + D \cdot W) + D \cdot W)$
paralelní [13]	$\Omega\Big(\frac{I \cdot D \cdot W \cdot (1+2Z)}{P} + 3\varepsilon\Big)$	$\Omega(P \cdot (Z \cdot (D + W + D \cdot W) + D \cdot W))$
paralelní*	$\Omega\Big(\frac{I \cdot D}{P} + \varepsilon\Big)$	$\Omega(P \cdot (Z \cdot W + sparse(D \cdot W)))$

Níže v tabulce 4.2 je ukázka konkrétních časových a paměťových náročností v závislosti na jednotlivých konkrétních parametrech, kde A_1 je paralelní implementace [13] a A_2^* je implementace z části 4.2.1.

Tabulka 4.2: Porovnání rychlostí a složitostí implementací algoritmu pLSA

D	W	Z	P	I	algoritmus	čas [h]	RAM [GB]	HDD [GB]
1 146	255	30	1	1	A_1	47,919	0,230	0
1 146	255	30	1	1	A_2^*	0,109	0,001	0,001
191 676	37 441	200	14	36	A_1	5 525,716	510 368,854	0
191 676	37 441	200	14	36	A_2^*	46,999	12,193	2,2
191 676	37 441	200	14	108	A_1	16 577,149	510 368,854	0
191 676	37 441	200	14	108	A_2^*	140,997	12,193	2,2

V posledním řádku můžeme vidět 14 procesů, 108 iterací za čas cca 6 dnů u optimalizované paralelní implementace. Lze jasně vidět, že optimalizace byla velmi potřeba. Těchto 6 dní výpočtu při 14 výpočetních jádrech procesoru Intel Xeon E5-2650v2 2,6 GHz a na velmi rychlých SSD discích je přijatelná časová zátěž pro získání výsledků. Jak již bylo zmíněno výše v této části práce, tak 108 iterací je dostatečných pro vynikající přesnost[5]. Vyjádřeno v procentech, A_2^* je cca o 99,15 % rychlejší, než algoritmus A_1 a paměťově je A_2^* o 99,99 % méně náročný.

Algoritmus byl vyzkoušen pro 100, 150 a 200 témat. Nejlepších výsledků bylo dosaženo pro 200 témat, kde výsledné popisy témat (viz sekce 4.3) byly nejvíce popisné a také zůstalo nejvíce témat vyhovujících omezujícím podmínkám popsaných v části 3a sekce 4.3.1, které slouží pro odstranění málo definujících témat. U 100 témat těmto podmínkám vyhovělo pouze 14 z nich, zato u 200 témat to bylo již přes 45. Větší počet témat je samozřejmě úměrný většímu počtu vyhovujících témat, tedy pro detailnější přesnost je možné počítat s větším počtem témat, ale bylo rozhodnuto, že pro získání vypovídajících výsledků je tento počet (200 témat) dostatečný.

[5]Algoritmus byl puštěn 3 krát na 47 hodin kvůli prioritě dané fronty v MetaCentru. Po 3 těchto spuštěních byl výsledek natolik dostatečný, že dalšího puštění nebylo třeba. Omezení na 14 jader bylo z důvodu takového, že v MetaCentru stroje s SSD disky mají méně procesorových jader. Rychlost při použití HDD byla přibližně o 73 % pomalejší, než u použití SSD disků.

4.3 Popis nalezených témat

Výsledkem pLSA algoritmu jsou pravděpodobnosti $P(z|d)$ a $P(w|z)$. Nalezené distribuce $P(w|z)$ (pravděpodobnosti témat pro jednotlivé domény) by bylo dobré nějakým způsobem popsat či dokonce pojmenovat. Bohužel zde ale vycházíme z faktu, že algoritmus nijak nereflektuje jednotný systém, který by bylo snadné popsat. Algoritmus vychází ryze z matice četností a hledá takové skupiny domén, které jsou pro nějakou skupinu uživatelů charakteristické.

Zde je vhodné poukázat na fakt, že data, nad kterými je výzkum prováděn, nemají dokonalý charakter, ale zato jsou vytvořena z reálných dat. Konkrétní uživatelé opravdu navštívili dané domény. Ovšem pojem „uživatel“ je v tomto pohledu poněkud zavádějící. Jak již bylo naznačeno v sekci 1.2.1 o popisu dat, tak pojem „uživatel“, který je zde hojně skloňován, nemusí reprezentovat konkrétní osobu v reálném světě, ale pouze identifikuje počítač, který sbírá data. Na tomto počítači může pracovat více členů domácnosti, ba dokonce tento počítač může být absolutně veřejný (například v internetové kavárně). To, jak je daný počítač využíván, z dat, která jsme dostali, nelze zjistit. Proto pro zjednodušení problému je jeden řádek matice četností uvažován pro jednoho uživatele. Tím, že může být počítač sdílen mezi více osobami dochází k jisté desinformaci, neboli zkreslení dat. Každá osoba používající počítač sbírající data může a velmi pravděpodobně má jiné zájmy, než osoba jiná využívající stejný počítač. V matici četností jsou ale zaznamenány zájmy všech těchto jedinců dohromady, a tedy výsledný „zájem“ (okruh webových domén) může být mnohem širší, než kdyby počítač používala pouze jedna osoba. Na tomto základě je poměrně složité určit kategorie zájmů jednotlivých uživatelů, protože právě záznamy z jednotlivých počítačů mohou pocházet od vícero jedinců, a tedy profil člověka sestaven nad těmito záznamy může být do jisté míry nepřesný.

4.3.1 Open Directory Project - DMOZ

Vhodným řešením k popisu stránek může být ODP; *Open Directory Project neboli ODP neboli DMOZ*[6] *je největší lidmy budovaný katalog internetových stránek, který vytvářejí dobrovolní editoři z celého světa* [4]. Tento katalog obsahuje více než čtyři miliony stránek, které jsou rozděleny do více než jednoho milionu kategorií a podkategorií. Na tomto portále existuje 137 582 domén v průniku se zpracovávaným plným datasetem, ale pouze 17 073 domén v průniku s ořezaným datasetem dle sekce 4.1. Tento fakt, že existuje popis kategorie pouze pro 45,6 % URL může být značně

[6] The Open Directory Project (ODP), největší lidmy budovaný katalog internetových stránek.

omezující pro úplnost výsledku. Lepší portál s větším počtem výsledků nebyl bohužel nalezen.

Kategorie v DMOZ jsou řazeny hierarchicky. Do prvního stupně spadají kategorie: *Arts*, *Business*, *Computers*, *Games*, *Health*, *Home*, *Kids and Teens*, *News*, *Recreation*, *Reference*, *Regional*, *Science*, *Shopping*, *Society*, *Sports* a *World*. Každá z těchto kategorií je poté na druhém stupni opět rozřazena do podkategorií atd. Hloubka některých podkategorií může být i 8.

Abychom docílili pojmenování tématu, je nutné zanalyzovat kategorie pro jednotlivé domény nacházející se v daném tématu. Sjednocením těchto kategorií, které vznikly vyhledáním domény na DMOZ portálu, lze určit rozsah zájmů daného tématu. Pro rychlejší zpracování byly staženy do offline použití všechny „kategorické stromy“ pro každou doménu. Jak již bylo zmíněno, těchto shod bylo nalezeno pouze 17 073.

Protože z DMOZ databáze lze získat jen kategorické umístění v ručně sestaveném stromě kategorií, nelze tímto postupem získat konkrétní pojmenování témat, ale lze zjistit nejvýznamnější kategorie tvořící dané téma. Princip získání kategorií pro dané téma je následující:

1. Výpočet pLSA, nebo načtení matice $P(w|z)$ z předchozího výpočtu pLSA.
2. Načtení kategorií z DMOZ pouze pro domény odpovídající danému datasetu.
3. Sestavení bloků domén z tématu. Pro každé téma, tedy $P(w|z_i)$, kde i je index tématu:
   (a) Ponechání bloku i, pokud zastoupení první domény (seřazeno sestupně) je větší než 20 %[7].
4. Pro každý blok domén:
   (a) Výpis prvních 30 domén (seřazeno sestupně dle pravděpodobnosti $P(w|z_i)$, jejich pravděpodobností v tématu a výpis kategorií z DMOZ pro každou doménu.
   (b) Výpis sestupně seřazeného počtu výskytů kategorií z prvních 30 domén spolu s názvem kategorie.

Analýzou těchto výpisů bylo zjištěno, že při použití 100 tematických bloků jsou výsledná témata poněkud obecnějšího charakteru. Bylo tedy rozhodnuto o použití algoritmu pro 200 témat, odkud jsme dostali přesvědčivější výsledky. Část výsledků je zobrazena v následující tabulce 4.3. V tabulce jsou zobrazena dvě náhodně vybraná témata. Lze rozeznat, že například v prvním tématu jsou nejvýznamnější webové

[7]Hranice 20 % byla nastavena takto z důvodu, že z 200 vygenerovaných témat ne všechna měla dostatečně bohaté zastoupení domén. Některé kategorie měly více rovnoměrnější distribuce, a tedy výrazně širší zájmovou oblast.

stránky zaměřeny primárně na vzdělávání a sekundárně na novinky ve vojenství. V druhém tématu jsou webové stránky zaměřeny na logistiku či motorismus. Nalezení přesného a hlavně jednotného popisu těchto témat v zásadě nelze, nebo alespoň nelze dokonale. Témata byla nalezena pomocí algoritmu pLSA, který nijak nereflektuje závislosti slov (domén) mezi sebou. Vychází se pouze z matice četností.

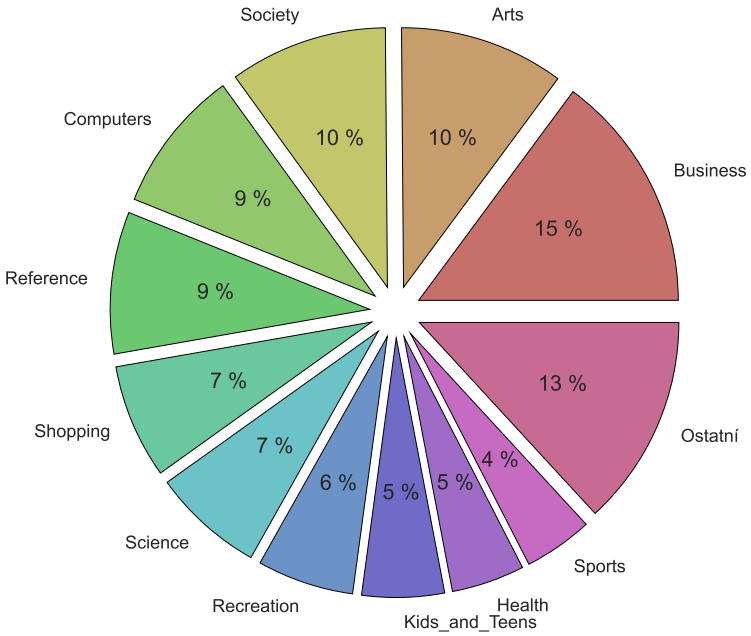
Tabulka 4.3: Kategorie z DMOZ

Téma 1			Téma 2		
Doména	$P(w\|z_i)$	Kategorie	Doména	$P(w\|z_i)$	Kategorie
northwood.edu	57,7 %	Education	paccar.com	37,4 %	Trucking
militarytimes.com	6,7 %	News	state.tn.us	8,6 %	Logistics
fhu.edu	5,9 %	Education	wyoming.com	3,7 %	Regional
army.mil	5,7 %	News	calstate.edu	2,3 %	Education

V tabulce lze vidět kategorie, které jsou relativně obecného charakteru, ale jsou poměrně specifické. Bylo zde použito zobrazení druhé až třetí podkategorie, protože první stupeň kategorií byl v daném kontextu až příliš obecný. První téma je zájmově poměrně úzké. Jedná se pravděpodobně o studenty Northwoodské univerzity z Michiganu nebo Floridy. Lidé, kteří navštěvují stránky této vysoké školy též navštěvují ty webové stránky, které se zabývají novinkami ve vojenství. Toto téma bylo tedy nalezeno poměrně dobře. Ovšem druhé již tak přesvědčivé není. Jsou zde na prvních příčkách stránky o motorismu, ale s nižšími pravděpodobnostmi a další domény mají různorodější charakter. Tedy toto téma nemůže mít jednotný popis, ale je nutné využít k popisu všechny kategorie, z nichž je téma charakterizováno. Bohužel toto je případ poměrně častý, protože algoritmus pLSA vychází pouze z konkrétních návštěvností, aniž by nějakým způsobem dával váhu kontextovým informacím.

V obrázku 4.2 na další stránce lze vidět významnosti všech kategorií[8] v daném korpusu pro 200 nalezených témat. Nejpočetnější kategorie *Regional* či velmi početná kategorie *World* (které nejsou v grafu ukázány) určují geografické umístění a jsou často jako doplňující kategorií u většiny popisů domén. Z cca 46 % byla zastoupena severní Amerika, tedy se jedná o očekávatelný výsledek, protože jsou data sbírána na území USA. V grafu je poslední kategorie nazvaná *Ostatní*, která shlukuje ty kategorie, které měly méně než 3% významnost.

[8]Jsou zobrazeny kategorie pouze z první vrstvy kategorií, tedy ty nejobecnější. To proto, že v druhé vrstvě celkový počet kategorií byl 258 a ve třetí vrstvě dokonce 1548 různých podkategorií. Z těchto důvodů pro přehlednost zobrazení je ukázáno pouze rozdělení pro první vrstvu kategorií.



Obrázek 4.2: Zastoupení kategorií z DMOZ

DMOZ dává významný náhled na kategorizaci domén, bohužel ne všechna zakategorizování jsou úplná a správná. Některé kategorie, které přísluší dané doméně neodpovídají zcela pravdě. Z tohoto důvodu je vhodné doplnit tuto informaci o kategoriích ještě o nějakou další informaci.

4.3.2 Nejvýznamnější slova dle webového obsahu

Další vhodnou cestou k pojmenování témat může být výpis nejrelevantnějších slov vyskytujících se na doménách tvořící dané téma. Protože v daném korpusu jsou pouze domény druhého řádu, tak objektivnost celého tohoto „pokusu“ není všeříkající. Kdybychom mohli analyzovat slova ze všech konkrétních URL adres, které byly navštíveny uživateli, tak by tato technika byla mnohem významnější.

K tomu, abychom mohli rychle přistupovat k jednotlivým obsahům je nutné stáhnout si HTML[9] dokument dané URL adresy (domény). Zpravidla se jedná pouze o domovskou stránku, která v některých případech může, ale také nemusí obsahovat slova, která jsou pro danou doménu typická. I přes toto riziko je ale tento experiment vhodný, protože přidává další informaci sloužící k popisu daného tématu. Každá

[9]HyperText Markup Language.

URL stránka byla stažena do interního úložiště jakožto HTML soubor, kde počet všech stažených domén byl 549 769 a zabrané místo 33,3 GB. Důvodem, proč nebylo staženo všech 586 624 domén je takový, že doména ve skutečnosti na internetu nebyla dosažitelná. V takovém případě uživatel mohl navštívit podstránku hostovanou na dané doméně, ale samotná domovská stránka neexistovala. Dalšími důvody mohlo být to, že daná stránka byla přesměrována či zrušena. Po stažení bylo zjištěno, že několik stránek bylo prázdných či obsahovaly pouze informaci o přesměrování. Tyto stránky nijak nepřispívají k nalezení slov charakteristických pro dané téma, protože obsah z nich získaný nereflektoval kontext dané stránky. Z celkového počtu 6 000 domén (200 témat × 30 nejvýznamnějších stránek pro téma) bylo dosažitelných pouze 5 568. V případě omezení na 20 % u nejvýznamnější domény z tématu (viz 3a z části 4.3.1) z 2 880 zbylo 2 688 domén, které na disku zabírají 168 MB.

Významná slova lze získat pomocí TF-IDF algoritmu. Jako dokumenty jsou zde brány jednotlivé obsahy z domén a slova jsou pomocí TF-IDF vypočítána. Vznikne slovník všech slov[10] o délce S, která se vyskytují v daném tématu z_i na obsazích webových stránek a také matice reprezentující TF-IDF hodnoty pro každé slovo z každého dokumentu. Protože každá webová stránka má jinou významnost v daném tématu, je vhodné převážit řádky TF-IDF matice právě vektorem významnosti konkrétní domény:

$$\begin{array}{ccccc} X_i & = & P(w|z_i) & \times & T_i \\ \begin{pmatrix} x_1 & \dots & x_S \end{pmatrix} & = & \begin{pmatrix} b_1 & \dots & b_{30} \end{pmatrix} & \times & \begin{pmatrix} t_{1,1} & \dots & t_{1,S} \\ \vdots & \ddots & \vdots \\ t_{30,1} & \dots & t_{30,S} \end{pmatrix} \end{array} \tag{4.10}$$

kde $P(w|z_i)$ je řádek z matice $P(w|z)$ značící pravděpodobnosti domén w v daném tématu z_i, T_i je TF-IDF matice příslušející tématu i a X_i je výsledný vektor délky S. Tímto maticovým převážením vznikne vektor obsahující všechna významná slova vyskytující se v daném tématu, která jsou vážena pravděpodobností domény. Tedy příkladem: pokud na první doméně bylo slovo s_1 se stejnou hodnotou TF-IDF jako slovo s_2 nacházející se na druhém dokumentu (tedy $t_{1,1} = t_{2,2}$, kde ale $s_1 \neq s_2$), tak po převážení $x_1 = b_1 \times t_{1,1}$ a $x_2 = b_2 \times t_{2,2}$ bude $x_1 > x_2$, protože $b_1 > b_2$. Tedy po převážení a seřazení vektoru X_i získáme významná slova pro dané téma.

[10]Je vhodné odstranit „stop-words“ (slova, která se vyskytují často, ale nenesou významnou informaci).

Pro doplnění je vhodné poznamenat, že TF-IDF transformace byla provedena na obsazích domén jednoho tématu, ale také na obsazích všech domén. Z obsahů všech domén vznikl velký slovník slov[11], ze kterého byly vypůjčeny hodnoty IDF (inverzní hodnota výskytu slova napříč všemi dokumenty), které byly nahrazeny místo IDF vypočítaných napříč tématem. Tímto jsme dosáhli lepší váženosti TF-IDF hodnoty slova. Použitím „globálních“[12] hodnot IDF byla významnost jednotlivých slov vyhodnocena v rámci významnosti slova přes celý dataset a nikoli jen přes dané téma. Ve výsledku tyto dvě metody výpočtu TF-IDF matice podávaly víceméně stejně závěry, protože nejvýznamnější roli v převážení slov hrají jednotlivé pravděpodobnosti domén v tématu.

Níže je ukázka nalezených významných slov pro témata z tabulky 4.3:

Tabulka 4.4: Nejvýznamnější slova dle obsahů stránek

Téma 1		Téma 2	
Významné slovo	X_1	Významné slovo	X_2
university	0,357	news	0,179
business	0,197	global	0,104
school	0,143	careers	0,083
programs	0,132	services	0,064
college	0,110	transportation	0,024

V prvním sloupečku (Téma 1) jsou významná slova jasně ukazující školní prostředí. To proto, že největší pravděpodobnost v tomto tématu měla Northwoodská univerzita, a tedy nejvýznamnější slova pocházejí právě ze stránek této univerzity. V seznamu dále poté byla slova jako *military*, která ukazovala na zájem tématu i v této oblasti. V druhém tématu (Téma 2) jsou převážně slova, která odpovídají slovům na stránce *paccar.com*, která se zabývá prodejem a dalšími službami ohledně logistických kamionů. Ze seznamu nejvýznamnějších slov je poměrně složité jasně odhadnout zaměření tématu, ale při posouzení většího počtu významných slov lze usuzovat, že se jedná o téma nabízející služby ohledně kamionové dopravy. Za povšimnutí též stojí i výrazně menší hodnoty X_2 u druhého tématu oproti hodnotám X_1 z tématu prvního. To je pravděpodobně způsobeno širším zájmovým okruhem daného tématu, neboť Northwoodská univerzita tvořila z 57,7 % Téma 1, ale stránka *paccar.com* tvořila Téma 2 jen z 37,4 %. Z této dedukce lze uvažovat nad jednotným měřítkem šíře zájmu daného tématu a to například pomocí entropie, která určuje

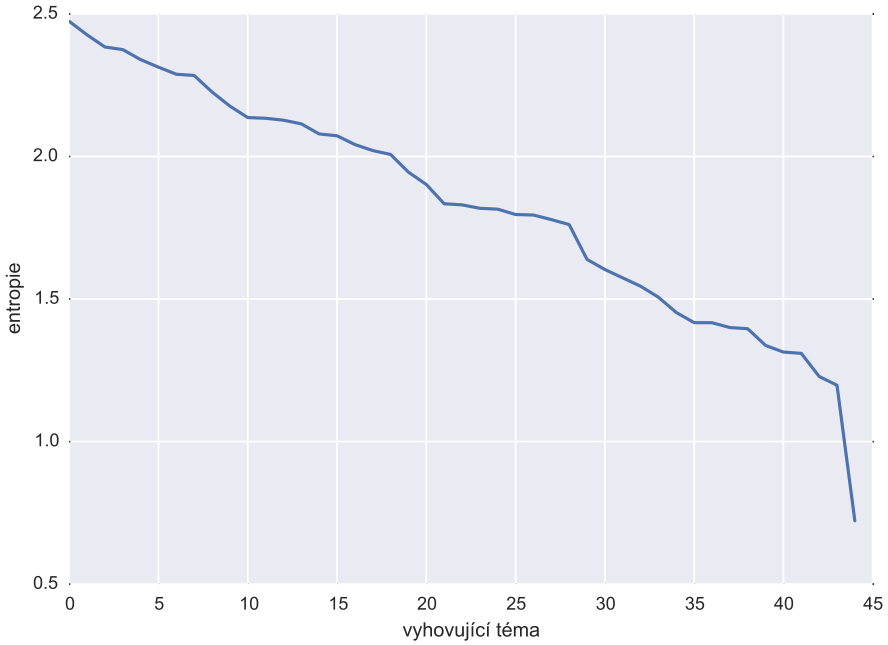
[11] V nijak neomezeném počtu bylo přes 16 milionů slov, které ale měly dost „šumu“, který bylo nutné odstranit.

[12] Myšleno globální v rámci datasetu domén.

informační hodnotu. Čím menší je entropie tématu, tím je užší zájem daného tématu, neboť zde existují takové domény, které mají vysokou pravděpodobnost a jsou tedy pro dané téma více charakteristické.

$$H(P(w|z_i)) = -\sum_{j=1}^{30} P(w_j|z_i) \cdot log_2(P(w_j|z_i)) \tag{4.11}$$

kde $H(P(w|z_i))$ je entropie daného tématu i a kde $P(w_j|z_i)$ je pravděpodobnost domény v tématu. Entropie Tématu 1 je 1,4172 a Tématu 2 je 2,2841, což souhlasí s tím, že Téma 2 má širší zájmovou skupinu, než Téma 1, protože má nižší pravděpodobnosti domén, které dané Téma 2 tvoří.



Obrázek 4.3: Hodnoty entropie napříč vyhovujícími tématy

Poměr úzce zaměřených témat definujících nejvýznamnější doménou je přibližně podobný počtu témat šířeji zaměřených. Tento fakt lze vyčíst i z obrázku 4.3, kde vidíme sestupně seřazené hodnoty entropií pro všechna témata, která vyhovují podmínkám 3a z části 4.3.1. Při zpřísnění těchto podmínek lze vyselektovat témata s mnohem nižší entropií, a tedy s užším zaměřením.

4.4 Finální výsledky

Aplikací kroků zmíněných v této kapitole 4 jsme získali seznam témat, která se skládala z distribucí domén. Tato témata jsme poté popsali pomocí kategorií z DMOZ databáze a následně pomocí jednotlivých obsahů webových stránek jsme získali významná slova charakteristická pro dané téma.

Výsledkem těchto kroků je postup, díky kterému lze získat kategorie webových stránek. Ideou práce byla původně ale kategorizace uživatelů na základě jejich procházení internetu. Tato idea bohužel není z poskytnutých dat proveditelná úplně, vzhledem k tomu, že jsme dostali data již značně omezená[13]. Z poskytnutých dat není možné dokonale analyzovat zájmy populace, lze ale využít poznatky k sestavení postupu plné analýzy. Hlavním problémem je pojmenování témat. Bez jakékoli bližší znalosti o konkrétních uživatelích neumíme za daných podmínek pojmenovat nalezená témata zájmově či dokonce demograficky. Místo pojmenování témat je v této práci zmíněn popis témat pomocí kategorií webových stránek z DMOZ databáze, které bohužel nejsou dostatečně přesné. Dále pomocí slov charakteristických pro obsahy domovských stránek přístupných na poskytnutých doménách.

Aby bylo možné pojmenování nalezených témat zájmově či demograficky, je nutné přidání další popisné složky k uživatelům a zúžit řád domén. Nyní jsou nalezená témata charakteristická pro „nějakou“ skupinu populace. Tuto skupinu umíme charakterizovat pomocí distribucí domén, a ty umíme popsat pomocí kategorie domény. Také umíme popsat téma pomocí slov specifických pro danou skupinu domén. Chybí zde ale informace o popisu uživatele.

Abychom mohli popsat nalezená témata například pomocí zájmů, potřebujeme znát zájmy alespoň několika málo uživatelů, kteří jsou charakterističtí pro dané téma.

Z algoritmu pLSA lze získat $P(z|d)$, což jsou distribuce nalezených témat pro jednotlivé uživatele. Popřípadě pomocí metody *folding-in* vysvětlené v části 3.3.5 lze umístit nové uživatele do již natrénovaného prostoru témat. Každý uživatel je popsán distribucí témat. Pokud bychom znali bližší zájmové informace o konkrétních uživatelích, mohli bychom tyto zájmy dle pravděpodobností témat promítnout do konkrétních témat, a tím zpřesnit jejich popis. Totéž platí i pro demografické

[13] Z původního clickstreamu byla vytvořena matice četností, čímž se zanonymizovala informace o průběžném průchodu uživatele internetem. Dále byly konkrétní URL adresy omezeny pouze na domény 2. řádu, čímž téměř zmizela informace o konkrétních zájmech uživatele na specifické URL adrese. Tato informace byla zredukována pouze na doménu, která ale nemusí být jako celek dostatečně subjektivní vzhledem k zájmům uživatele.

informace. Příkladem: věděli bychom, že uživatel U_1 má rozložení pravděpodobností témat *Téma 1: 35 %, Téma 2: 5 %, Téma 3: 60 %* a věděli bychom, že uživatel U_1 je 35letý svobodný muž pracující jako účetní a bydlící v Praze v Dejvicích a se zájmem hraje volejbal a chodí plavat, tak bychom všechny tyto informace mohli promítnout dle pravděpodobností do jednotlivých témat. Na základě určité popsané skupiny uživatelů bychom mohli získat dostatečný popis témat v podobě demografických či zájmových oblastí. Z těchto informací by poté bylo možné takové téma pojmenovat. Každé téma by bylo popsáno jistými pravděpodobnostmi uživatelových popisných informací a prostým převážením těchto informací by bylo možné dokonce pojmenovat dané téma. Pokud by například nějaké téma bylo charakteristické pro několik uživatelů, kteří jsou sportovci, lze o takovém tématu říci, že je zaměřeno na sport. Čím více by bylo popsaných uživatelů, tím lepší popisný model bychom mohli získat. Momentálně jsme dočista bez této informační hodnoty, a tedy řádné pojmenování témat na základě zájmů či obecně popisu uživatele není uskutečnitelné za daných podmínek.

Kapitola 5

Závěr

Tato práce se zabývá popisem metod vhodných ke kategorizaci webových stránek charakteristických pro určitou skupinu populace na základě jejich chování. Tímto chováním je myšleno navštěvování domén na internetu. Pro tuto práci byla poskytnuta zanonymizovaná data vycházející z reálných záznamů procházení internetu uživateli USA za dobu jednoho měsíce. Tato data jsou ve formátu matice četností, nad kterou je prováděn celý výzkum.

Prvně bylo nutné data zanalyzovat a poté redukovat méně důležité části dat, aby vynikly části s větší vypovídající hodnotou. Touto redukcí byla původní matice četností o rozměrech 224 679 uživatelů × 586 624 URL redukována na 191 676 uživatelů × 37 441 URL při zachování a zvýraznění diskriminujících dat. Následně byl zvolen tzv. topic-model algoritmus pLSA, který dokáže najít $P(w|z)$ popisující pravděpodobnosti domén w pro nalezená témata z a $P(z|d)$ popisující pravděpodobnosti témat charakterizujících konkrétní uživatele d. Vhodná implementace tohoto algoritmu nebyla, a tedy bylo nutné algoritmus reimplementovat, aby byl schopný v rozumném čase s rozumnými prostředky dodat výsledky pro poskytnutá data. Po nalezení témat popisujících uživatele pomocí domén bylo vhodné tato témata nějakým způsobem pojmenovat. Z důvodu nedostatečného popisu uživatelů bylo nutné přistoupit k popisu témat pomocí jiných popisných metod. Byla použita největší ručně tvořená databáze kategorií webových stránek DMOZ, u které se později zjistilo, že některé popisy domén pomocí kategorií neodpovídají skutečnosti. Popis témat byl proto doplněn o seznamy charakteristických slov získaných z obsahů webových stránek. Výsledkem je postup vedoucí k nalezení témat kategorizujících uživatele pomocí témat, která jsou kategorizována pomocí domén.

Cílem práce bylo popsání metod vedoucích k nalezení skupin webových stránek, které jsou pro určitou skupinu populace charakteristické. Tento cíl se do jisté míry

podařilo splnit pomocí témat charakterizujících jednotlivé uživatele pomocí domén, přičemž výsledné pojmenování nalezených témat nebylo možné z důvodu nedostatečného popisu uživatelů. Místo pojmenování témat byla tato témata popsána pomocí obecných kategorií z DMOZ databáze a pomocí slov charakteristických pro domény obsažené v tématech.

Při dodání dodatečných popisných informací o uživatelích je možné na základě této práce pojmenovat témata shlukující jisté části populace. Charakter popisných dat určí popisné schopnosti při pojmenování témat. Tato práce může sloužit jako výchozí bod při hledání kategorií popisujících uživatele internetu, popřípadě jako studijní materiál ukazující využití různých metod kategorizace.

Příloha A

Obsah přiloženého CD

latex-src/ Zdrojové soubory pro LaTeX k této práci.

source-code/ Implementace algoritmů v Pythonu.

Jencik-thesis-2015.pdf Tato práce v elektronické podobě.

Literatura

[1] BAHMANI, Bahman; MOSELEY, Benjamin; VATTANI, Andrea; KUMAR, Ravi a VASSILVITSKII, Sergei. Scalable K-means++. *Standford University, CA* [online]. Poslední aktualizace 2012 [cit. 2015-04-26]. Dostupné z: `http://vldb.org/pvldb/vol5/p622_bahmanbahmani_vldb2012.pdf`.

[2] CHEN, Ting. Notes on EM and pLSA. *Dataera* [online]. Poslední aktualizace 1. 4. 2014 [cit. 2015-04-26]. Dostupné z: `http://dataera.org/2014/04/notes-on-em-and-plsa/`.

[3] DEMPSTER, A. P.; LAIRD, M. a RUBIN, D. B. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Stastistical Society. Series B, volume 39. MIT* [online]. Poslední aktualizace 1977 [cit. 2015-04-26]. Dostupné z: `http://web.mit.edu/6.435/www/Dempster77.pdf`.

[4] DMOZ. Open Directory Project. *AOL Inc.* [online]. Poslední aktualizace 26. 4. 2015 [cit. 2015-05-14]. Dostupné z: `http://www.dmoz.org`.

[5] ESTER, Matrin; KRIEGEL, Hans-Peter; SANDER, Jörg a XU, Xiaowei. A Density-Based algorithm for Discovering Clusters. *Institute for Computer Science, University of Munich, Germany* [online]. Poslední aktualizace 1996 [cit. 2015-05-16]. Dostupné z: `https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf`.

[6] FARAHAT, Ayman a CHEN, Francine. Improving Probabilistic Latent Semantic Analysis with Principal Component Analysis. *A Digital Archive of Research Papers in Computational Linguistics* [online]. [cit. 2015-04-27]. Dostupné z: `http://www.aclweb.org/anthology/E06-1014`.

[7] HOFMANN, Thomas. Probabilistic Latent Semantic Analysis. *Uncertainity in Artificial Intelligence, University of California, Berkeley & International Computer Science Institute, Berkeley, CA* [online]. Poslední aktualizace 1999 [cit. 2015-04-26]. Dostupné z: `http://cs.brown.edu/~th/papers/Hofmann-UAI99.pdf`.

[8] HOFMANN, Thomas. Probabilistic Latent Semantic Indexing. *International Computer Science Institute, Berkeley, CA* [online]. Poslední aktualizace 1999 [cit. 2015-04-27]. Dostupné z: `http://cs.brown.edu/~th/papers/Hofmann-SIGIR99.pdf`.

[9] HOFMANN, Thomas; PUZICHA, Jan a JORDAN, Michael I. Unsupervised learning from Dyadic Data. *Advances in Neural Information Processing Systems, volume 11. MIT Press* [online]. Poslední aktualizace 1999 [cit. 2015-04-26]. Dostupné z: `http://www.cs.berkeley.edu/~jordan/papers/dyadic.pdf`.

[10] JONES, Eric; OLIPHANT, Travis; PETERSON, Pearu a další. SciPy: Open source scientific tools for Python. *scikit-learn* [online]. Poslední aktualizace 2015 [cit. 2015-05-16]. Dostupné z: `http://www.scipy.org`.

[11] KANUNGO, Tapas a ostatní. An Efficient k-Means Clustering Algorithm: Analysis and Implementation. *Pattern Analysis and Machine Intelligence, IEEE Transaction (2002): 881-892* [online]. Poslední aktualizace 2002 [cit. 2015-02-17]. Dostupné z: `https://www.cs.umd.edu/~mount/Projects/KMeans/pami02.pdf`.

[12] KAUCHAK, David. TF-IDF. *Standford University, CA* [online]. Poslední aktualizace 2009 [cit. 2015-02-17]. Dostupné z: `http://www.cs.pomona.edu/~dkauchak/classes/f09/cs160-f09/lectures/lecture5-tfidf.pdf`.

[13] KONG, Alex. pLSA implementace v Pythonu. *GitHub, Inc.* [online]. Poslední aktualizace 6. 10. 2012 [cit. 2015-05-02]. Dostupné z: `https://github.com/dx88968/PLSA-1`.

[14] MOE, Wendy W. a FADER, Peter S. Capturing evolving visit behavior in clickstream data. *Journal of interactive marketing, volume 18, number 1* [online]. Poslední aktualizace 2004 [cit. 2015-04-18]. Dostupné z: `https://marketing.wharton.upenn.edu/files/?whdmsaction=public:main.file&fileID=3817`.

[15] MONTGOMERY, Alan. Predicting Consumer Behavior using Clickstream Analysis. *Carnegie Mellon University* [online]. Poslední aktualizace 13. 6. 2003 [cit. 2015-02-17]. Dostupné z: `http://www.andrew.cmu.edu/user/alm3/presentations/WebShop%202003.pdf`.

[16] MORIK, Katharina a KÖPCKE, Hanna. Analysing Insurance Data or The Advantage of TF/IDF Features. *University of Dortmund, DE* [online]. [cit. 2015-04-19]. Dostupné z: `https://km.aifb.kit.edu/ws/LLWA/akkd/2.pdf`.

[17] PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M. a DUCHESNAY, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[18] scikit-learn. K-means implementace v Pythonu. *scikit-learn* [online]. Poslední aktualizace 25. 3. 2015 [cit. 2015-05-16]. Dostupné z: `http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`.

[19] scikit-learn. Mini Batch K-means implementace v Pythonu. *scikit-learn* [online]. Poslední aktualizace 25. 3. 2015 [cit. 2015-05-16]. Dostupné z: http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html.

[20] SCULLEY, D. Web-Scale K-means Clustering. *Google, Inc. Pittsburgh. PA USA* [online]. Poslední aktualizace 20. 4. 2010 [cit. 2015-05-16]. Dostupné z: http://www.eecs.tufts.edu/~dsculley/papers/fastkmeans.pdf.

[21] SEBERA, Martin. Analýza hlavních komponent a faktorová analýza. *Vícerozměrné statistické metody, Masarykova universita* [online]. Poslední aktualizace 13. 1. 2012 [cit. 2015-04-20]. Dostupné z: http://www.fsps.muni.cz/~sebera/vicerozmerna_statistika/pca.html.

[22] STEYVERS, Mark a GRIFFITHS, Tom. Probabilistic Topic Models. *University of California, Irvine; Brown University* [online]. [cit. 2015-04-26]. Dostupné z: http://psiexp.ss.uci.edu/research/papers/SteyversGriffithsLSABookFormatted.pdf.

[23] SUKHWANI, Sumit; GARLA, Satish a CHAKRABORTY, Goutam. Analysis of Clickstream Data Using SAS. *Oklahoma State University, Stillwater, UK* [online]. Poslední aktualizace 2012 [cit. 2015-02-17]. Dostupné z: http://support.sas.com/resources/papers/proceedings12/100-2012.pdf.

[24] TURNEY, Peter D. a PANTEL, Patrick. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research 37 (2010) 141–188* [online]. Poslední aktualizace 2010 [cit. 2015-02-17]. Dostupné z: https://www.jair.org/media/2934/live-2934-4846-jair.pdf.

[25] WANG, Kaijun; ZHANG, Junying; LI, Dan; ZHANG, Xinna a GUO, Tao. Adaptive Affinity Propagation Clusterint. *China Jiliang College, Xidian University, China* [online]. Poslední aktualizace 2007 [cit. 2015-05-16]. Dostupné z: http://arxiv.org/pdf/0805.1096.pdf.

[26] WERNER, Tomáš. Optimalizace. *ČVUT* [online]. Poslední aktualizace 19. 12. 2014 [cit. 2015-04-20]. Dostupné z: https://cw.fel.cvut.cz/wiki/_media/courses/a4b33opt/opt.pdf.

[27] WIKIPEDIA. Cosine similarity. *Wikipedia* [online]. Poslední aktualizace 17. 3. 2015 [cit. 2015-04-26]. Dostupné z: http://en.wikipedia.org/wiki/Cosine_similarity.

[28] WIKIPEDIA. K-means clustering. *Wikipedia* [online]. Poslední aktualizace 7. 4. 2015 [cit. 2015-04-26]. Dostupné z: http://en.wikipedia.org/wiki/K-means_clustering.

[29] WIKIPEDIA. Latent semantic analysis. *Wikipedia* [online]. Poslední aktualizace 25. 4. 2015 [cit. 2015-04-26]. Dostupné z: http://en.wikipedia.org/wiki/Latent_semantic_analysis.

[30] WIKIPEDIA. Natural language processing. *Wikipedia* [online]. Poslední aktualizace 18. 4. 2015 [cit. 2015-04-26]. Dostupné z: `http://en.wikipedia.org/wiki/Natural_language_processing`.

[31] WIKIPEDIA. Principal component analysis. *Wikipedia* [online]. Poslední aktualizace 10. 4. 2015 [cit. 2015-04-20]. Dostupné z: `http://en.wikipedia.org/wiki/Principal_component_analysis`.

[32] ŠKALOUDOVÁ, Alena. Metody extrakce faktorů. *Faktorová analýza* [online]. Poslední aktualizace 2010 [cit. 2015-04-20]. Dostupné z: `http://kps.pedf.cuni.cz/skalouda/fa/extrakce_fak.htm`.

Použité zkratky

URL Uniform Resource Locator. Unikátní webová adresa.

CSR Compressed Storrage Row. Standardní formát pro ukládání řídkých matic. Jsou uloženy pouze nenulové hodnoty.

TF-IDF Term Frequency - Inverse Document Frequency.

K-means Shlukovací algoritmus.

PCA Principal Component Analysis.

SVD Singular Value Decomposition.

LSA Latent Semantic Analysis.

pLSA Probabilistic Latent Semantic Analysis.

EM Expectation–Maximization.

DMOZ The Open Directory Project (ODP), největší lidmy budovaný katalog internetových stránek.

HTML HyperText Markup Language.