Mendelova univerzita v Brně
Provozně ekonomická fakulta

Hledání sémantické informace v textových datech s využitím latentní analýzy

Diplomová práce

Vedoucí práce:
doc. Ing. František Dařena, Ph.D.

Bc. Pavel Řezníček

Brno 2015

Děkuji vedoucímu práce doc. Ing. Františku Dařenovi, Ph.D. za jeho cenné rady, konstruktivní připomínky a čas, který mi při konzultacích věnoval. Také děkuji své rodině a nejbližším přátelům, kteří mě při psaní této práce podporovali.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Hledání sémantické informace v textových datech s využitím latentní analýzy**
vypracoval/a samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom/a, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 21. května 2015 _____

Abstract

Řezníček, P. Search of semantic information in text data using latent analysis. Diploma thesis. Brno: Mendel University, 2015.

The first part of thesis focuses on theoretical introduction to the methods of text mining – Information retrieval, classification and clustering. LSA method is presented as an advanced model for representing textual data. Furthermore, the work describes source data and methods for their preprocessing and preparation used to enhance the effectiveness of text mining methods. For each chosen text mining method there are defined evaluation metrics and used already existing, or newly implemented, programs are presented. The results of experiments comparing the effects of different preprocessing type and use of different models of the source data are then demonstrated and discussed in the conclusion.

Keywords

Text mining, LSI database, SVD, stemming, stop-words, clustering, classification, Information retrieval, Python, Cluto.

Abstrakt

Řezníček, P.. Hledání sémantické informace v textových datech s využitím latentní analýzy. Diplomová práce. Brno: Mendelova univerzita v Brně, 2015.

První část práce se věnuje teoretickému úvodu do vybraných metod text miningu – Information retrieval, klasifikace a shlukování. Je představena metoda LSA jakožto pokročilejší model pro reprezentaci textových dat. Dále jsou v práci popsána zdrojová data a metody pro jejich předzpracování a přípravu použité za účelem zvýšení efektivity text miningových metod. Pro jednotlivé oblasti text miningu jsou definovány hodnotící metriky a jsou představeny použité již existující, případně nově implementované, programy. Výsledky experimentů, srovnávajících vliv různého předzpracování a využití odlišných modelů zdrojových dat, jsou následně přehledně demonstrovány a v závěru práce diskutovány.

Klíčová slova

Text mining, LSI databáze, SVD, stemming, stop-slova, shlukování, klasifikace, vyhledávání informací, Python, Cluto.

Obsah

1 Úvod

V dnešní době se už jen velice málo typů dat ukládá v jiné formě než ve formě digitální. Výjimkou mohou být například lékařské zprávy, soudní spisy nebo účetní doklady. Současně se však už i v odvětvích, kde se člověk bez papíru a tužky dříve neobešel, stále více, zejména vlivem modernizace, data digitalizují. Staré tištěné záznamy se převádí do své digitální podoby, nové se vytvářejí na počítačích, nebo jsou z nich vytvářeny alespoň digitální kopie a ty následně ukládány. Vzniká tak obrovské množství dat, které s postupem času stále rychleji roste a je ukládáno na různých datových uložištích po celém světě. Data jsou jejich vlastníkům dostupná přes nejrůznější média a často slouží jako zálohy nebo důkazy, které je možné v případě nutnosti dohledat. Ukládáním dat bez jejich dalšího následného využití se však připravujeme o možnost získání cenných informací. Je potřeba si uvědomit, že čím větším množstvím dat vlastník disponuje, tím pravděpodobnější je fakt, že v nich mohou být nalezeny určité pravidelnosti a vzory. Tyto nově získané informace lze následně využít ať už k výzkumným, podnikatelským či dalším nejrůznějším účelům a dále s nimi pracovat. Zásadním faktem zůstává, že pokud člověk již jednou data vlastní a na nějakém místě se mu je podařilo shromáždit, většinou stojí za snahu se je pokusit určitým způsobem analyzovat a získat z nich další cenné informace.

Hledáním hodnotných vzorů a užitečných informací v datech se zabývá oblast data mining (dolování z dat). Pro možnost aplikace data miningových metod na určitá data je zpravidla vždy nutné mít tato data v určité strukturované formě. Pro predikci hodnot z historických dat se často využívají numerické ukazatele reprezentující stav reality v určitém okamžiku, na základě kterých se následně odhadují hodnoty ukazatelů v budoucnosti méně či více vzdálené. Není však pravidlem, že data, jež máme k dispozici, jsou uložena v číselné podobě jakožto hodnoty konkrétních ukazatelů. Častější formou reprezentace dat je jejich textová forma. Vyhledáváním souvislostí v textových datech se zabývá oblast text miningu. Text mining se od data miningu, jak by se mohlo zdát, příliš neliší. Oba přístupy vycházejí z dat z minulosti, jejichž podoba se sice výrazně liší, avšak využívané učící metody jsou podobné. Důvodem je, že tyto učící metody vyžadují na vstupu striktně strukturovaná data a pro zpracování textových dat je tedy vždy potřeba tyto data určitým způsobem nejdříve převést do jejich číselné reprezentace. Velice častým způsobem reprezentace textových dat jsou matice.

K nejvíce zkoumaným a studovaným metodám data miningu potažmo text miningu patří predikce a klasifikace (angl. Classification). Zjednodušeně lze predikci chápat jako zpracování dostupných vzorků dat s jejich předem známými hodnotami, s cílem odhadnout správné hodnoty pro vzorky nové. Příkladem může být například úloha kategorizace textů. Koncept klasifikace lze rozšířit pro data, kde jejich hodnoty nejsou předem jasně známy. Úkolem je následně data organizovat takovým způsobem, aby bylo možné jednotlivým skupinám dat přiřadit určité označení a používat jej pro data nová. Tento proces je označován

jako shlukování (angl. Clustering). Rozřazování dokumentů do skupin se provádí na základě měření podobnosti mezi dokumenty, což je základ pro nejrůznější analýzy dokumentů, zejména v oblasti vyhledávání informací (angl. Information retrieval, IR). Jak již bylo zmíněno, pro možnost aplikace uvedených metod na textová data, je nezbytné jejich převod do numerické podoby ve strukturované formě (Weiss, Indurkhya, Zhang, Damerau, 2010).

Jednou z nejpoužívanějších forem reprezentací textových dat umožňující snadné provádění algebraických operací s hodnotami jsou vektory (Howland, Park, 2004). S prvky textových dat, označovanými jako dokumenty, se poté pracuje v rámci vektorového prostoru o řádech až několika tisíců. Tento vektorový model (angl. Vector space model) je nejrozšířenější metodou pro modelování obsahu textových dat. Každý dokument je reprezentován jedním příznakovým vektorem, kde hodnoty odpovídají výskytům termů v daném dokumentu. Těmto hodnotám mohou být navíc přiřazeny statistické váhy určující důležitost jednotlivých slov – označovaných jako termy. (Wang, Zhang, Vassileva, 2010; Zhang, Zhu, 2005). Vektorový model má určité nedostatky, které lze eliminovat s využitím modernějších metod Latentní sémantické analýzy umožňujících odstranění přebytečného šumu v datech a navíc možného odhalení skrytých vazeb mezi prvky textových dat.

2 Cíl práce

Cílem práce je analyzovat dostupnou sadu nestrukturovaných textových dat v několika světových jazycích a s využitím vhodných metod text miningu z této sady dolovat určité cenné znalosti. Pomocí vhodných metod budou získaná data nejdříve různě předzpracována a vybranými metodami text miningu, zejména metodami klasifikace, shlukování a Information retrieval, budou z těchto dat s využitím metod latentní sémantické analýzy dolovány určité znalosti. Pomocí již existujících případně dílčích implementovaných programů bude v práci zkoumán vliv využití jedné z metod redukce dimenzí zdrojových dat na výsledky text miningových metod. Sledován bude také vliv různých metod předzpracování na získané výsledky. V závěru práce budou získané znalosti pro sady textových dat vybraných jazyků diskutovány a bude zhodnocen jejich přínos.

3 Dolování znalostí

3.1 Information retrieval

Oblast vyhledávání informací je nejvíce spojována s online dokumenty a jejich vyhledáváním v rámci Internetu. Na položený dotaz jsou nejdříve za pomoci určitých poskytnutých vodítek z dostupných kolekcí dokumentů vybrány vhodné dokumenty a ty jsou následně prezentovány jako odpověď na dotaz. Vodítka lze chápat jako slova, jež pomáhají identifikovat relevantní dokumenty. Typicky se vyhledávání provádí za pomocí menšího počtu klíčových slov, která jsou porovnávána s každým prvkem uložených kolekcí dokumentů. Dokumenty s nejlepší shodou jsou poté vráceny jako výsledek. Tento proces lze generalizovat na proces porovnávání dokumentů. Místo výčtu klíčových slov je jako dotaz použit celý dokument a jako výsledek jsou získány dokumenty s největší podobností. Měření podobnosti je základním konceptem IR. Srovnání se provádí v rámci dvou dokumentů a liší se podle využitého modelu reprezentujícího textová data (Weiss, Indurkhya, Zhang, Damerau, 2010).

3.2 Klasifikace

Kategorizace textů je široce používaný, avšak ne úplně přesný, název pro klasifikaci dokumentů. Jedná se o metodu učení s učitelem, kde součástí trénovacích dat je také „správná hodnota“. Metodou klasifikace lze na zdrojových datech převedených do numerického modelu natrénovat model umožňující třídit dokumenty do specifických témat, tedy je klasifikovat. Klasifikaci dokumentů lze využít například při detekci nevyžádaných emailů (SPAM) nebo pro úlohu automatického třídění přijatých dokumentů. Model umožňující provádět klasifikaci nad zdrojovými daty je nazýván klasifikátor (Weiss, Indurkhya, Zhang, Damerau, 2010).

Původní klasifikátory z období okolo roku 1980 využívali techniky znalostního inženýrství jako expertní systémy, které se typicky skládali z množiny manuálně definovaných logických pravidel vytvářených experty. Později byly vyvinuty mechanismy vytvářející klasifikátory na základě sledovaných charakteristik v sadě trénovacích dokumentů bez nutnosti definice pravidel člověkem. Bylo vytvořeno již mnoho typů klasifikátorů jako například naivní Bayesovské modely, K nejbližších sousedů, rozhodovací stromy, SVM (Support Vector Machines) nebo neuronové sítě (Silva, Ribeiro, 2009). Principy vybraných klasifikátorů budou blíže popsány v následující části textu.

3.2.1 Nejbližší sousedé

Metodou klasifikace založenou na faktu, že pokud spadají dva prvky do stejné třídy, musejí si být určitým způsobem podobné, je metoda K nejbližších sousedů (angl. K nearest neighbours, KNN). Klasifikace vychází ze zvolené metriky po-

dobnosti záznamů, podle které je každý záznam porovnáván a klasifikován do třídy obsahující mu nejbližší (nejpodobnější) prvky. Předpokládá se, že je k dispozici sada trénovacích dat, ke kterým jsou přiřazeny klasifikační třídy. Pro nově klasifikovaný záznam je při klasifikaci vybráno *k* trénovacích záznamů, jež jsou klasifikovanému nejblíže. Klasifikovaný záznam je zařazen do třídy, do které patří nejvíce z vybraných trénovacích záznamů. K určení podobnosti (vzdálenosti) záznamů používá KNN nejčastěji metriku Euklidovské vzdálenosti (Han, Kamber, 2000).

3.2.2 Support Vector Machines

Algoritmus podpůrných vektoru (SVM) pomáhá s řešením problému nalezení nadroviny v *n*-dimenzionálním prostoru. V oblasti text miningu jde o nalezení takové nadroviny, kde dokumenty s různými tématy leží v různých poloprostorech a vzdálenost mezi nimi je co největší (Cristianini, Shawe-Taylor, 2000).

Důležitým pojmem pro SVM je jádrová funkce (angl. kernel function). Lze očekávat, že hledání lineárního oddělovače (přímka, rovina, nadrovina) bude v originálním prostoru neúspěšné. Oddělovač však lze hledat ve vícerozměrném prostoru, do kterého jsou data převedena pomocí jádrové transformace (angl. kernel transformation). Původně lineárně neseparovatelná úloha je tak převedena na lineárně separovatelnou. Ve vícerozměrném prostoru se poté hledá oddělovač pomocí jádrové funkce, kterou je vhodné volit různě podle podoby dat (Steinwart, Christmann, 2008).

Při využití SVM klasifikátoru se v každé iteraci výpočtu provádí řada operací, jejichž výpočetní čas roste spolu s rostoucím počtem dimenzí zdrojových dat. Existují však metody optimalizace klasického SVM, které problém určitým způsobem redukují.

Shrinking

Technika shrinking pomáhá k redukci velikosti řešeného problému dočasnou eliminací proměnných, které dosáhly své horní popřípadě spodní hranice (hraniční proměnné). Během iteračního procesu jsou vektory rozdělovány do tří skupin – non support vectors, bounded support vectors a free support vectors. Non support vectors a bounded vectors jsou při každé iteraci kandidáti na odstranění a další iterace probíhají pouze na zbylých podpůrných vektorech, tím se redukuje řešený problém (Buttou, Lin, 2006).

Caching

Při vyhodnocování kernel funkce dochází v každé iteraci k výpočtu řádků zdrojové matice. Metodou jak se vyhnout opětovnému přepočítávání řádků této matice je cachování (angl. caching). Jelikož zdrojová matice je obvykle řídká matice teoreticky řádově až s desítkami tisíc dimenzí, nemusí být dostatek paměti pro její kompletní načtení a používá se speciálního uložiště pro uložení posledních používaných řádků matice. Touto metodou se snižují paměťové nároky výpočtu

a je vhodné ji používat při výpočtech na zařízeních disponujících menším paměťovým prostorem (Wu, 2007).

3.2.3 Rozhodovací stromy

Algoritmus tvorby rozhodovacího stromu (Decision tree) je založen na metodě rozděl a panuj. Trénovací data na začátku tvoří jednu množinu a postupně se rozdělují na menší a menší podmnožiny tak, aby se v jednotlivých podmnožinách nacházely prvky ze stejné třídy. Tento postup se často označuje jako „top down induction of decision trees“ (TDIDT). Algoritmus pro tvorbu rozhodovacího stromu se skládá ze tří základních kroků:

1. Zvol jeden atribut jako kořen dílčího stromu
2. Rozděl data v uzlu na podmnožiny podle zvoleného atributu a vytvoř nové uzly pro obě podmnožiny
3. Existuje-li uzel, obsahující data z více než jedné třídy, opakuj pro tento uzel kroky 1 – 3, jinak skonči

Klíčovou částí algoritmu je výběr vhodného atributu pro rozdělení prvků do různých podmnožin. Pro volbu tohoto atributu se využívají charakteristiky z teorie informace a pravděpodobnosti jako jsou entropie, informační zisk, Gini index a další (Quinlan, 1979).

3.3 Shlukování

Při aplikaci text miningových metod se zpravidla zpracovávají zdrojová data z jedné konkrétní domény, například novinové články, lékařské záznamy o pacientech a další. Tato zdrojová data lze určitým způsobem roztřídit do skupin s jistými společnými vlastnostmi. Člověk, který by měl o zdrojových datech přehled, by je mohl roztřídit do samostatných skupin podle témat, která se v jednotlivých dokumentech vyskytují. Pokud je však obsah a struktura dokumentů zdrojových dat neznámá, je vhodné pro jejich roztřízení použít jednu z text miningových metod zvanou shlukování (Weiss, Indurkhya, Zhang, Damerau, 2010).
Ve své podstatě jde o proces rozložení množiny dokumentů zdrojových dat na disjunktní, případně částečně se překrývající, podmnožiny, z nichž každá má následující vlastnosti:

- dokumenty v ní obsažené jsou si obsahově velmi podobné
- dokumenty v ní neobsažené jsou málo podobné dokumentům v ní obsaženým

Získané podmnožiny by bylo možné opět rozdělit na podmnožiny se stejnými vlastnostmi tolikrát, až by se došlo k dostatečně homogenním podmnožinám. Vyhledávání shluků by následně probíhalo hierarchicky metodou shora dolů (top-down) případně zespoda nahoru (bottom-up).

Pro vytvoření shluků je potřeba spočítat podobnost mezi všemi dokumenty. To může být velice náročné, pokud je dokument reprezentován pomocí několika tisíc dimenzí, což je případ numerické reprezentace textových v klasickém vektorovém modelu. Redukci problému a snížení výpočetní složitosti lze dostáhnout redukcí prostoru například pomocí SVD. Aplikací singulárního rozkladu na zdrojová data je výrazně zmenšen prostor, ve kterém budou prováděny shlukovací metody. Je však nutné při redukci prostoru zvolit vhodný počet dimenzí. Bylo dokázáno, že ideálních výsledků lze při klasifikaci dosáhnout při výběru poměrně malého počtu (20–100) dimenzí. (Landauer, McNamara, Dennis, Kintsch, 2014).

Zásadním při shlukování je zejména určení optimálního počtu shluků, do kterých mají být dokumenty rozděleny. V zásadě lze počet shluků stanovit na základě testování nebo s využitím heuristických metod, případně kombinací obou. Při využití přístupu na základě testování se po několikanásobném opakování procesu shlukování s různými počty shluků volí jako výsledek takový počet, kde je rozložení shluků optimální (Hebák, Hustopecký, Malá, 2000).

3.3.1 Shlukovací metody

Pro shlukování textových dat lze využít několik různých shlukovacích metod. Mezi typy shlukovačích metod patří například hierarchické a dělící metody, metody založené na hustotě a další. Vybrané typy shlukovacích metod jsou stručně popsány v následující části textu.

3.3.1.1 Dělící metody

Dělící metody rozdělují dokumenty do určitého počtu shluků, který je předem zadán uživatelem. Nejznámějším zástupcem dělících metod je algoritmus K-means, jakožto zástupce klasických nehierarchických metod shlukování, je jedním z nejjednodušších algoritmů shlukovací analýzy (Ševčík, 2010). Při inicializaci algoritmu se ze zdrojových dat vybírá k reprezentantů, představujících středy shluků. Principem algoritmu je posun středů shluků, označovaných jako centroidy, takovým způsobem, aby dokumenty uvnitř stejného shluku byly co nejvíce homogenní. Podobnost dokumentů je posuzována podle zvolené metriky podobnosti (euklidovská vzdálenost, kosinová vzdálenost, apod.). Základní algoritmus K-means pracuje v následujících krocích (Konchady, 2006):

4. Z kolekce dokumentů vyber k reprezentantů
5. Opakuj následující podkroky dokud nejsou splněny ukončovací podmínky
   5.1. Pro každý dokument d najdi shluk i, jehož centroid je nejblíže a přiřaď dokument d do shluku i
   5.2. Pro každý shluk i přepočítej jeho aktuální střed na základě dokumentů, které aktuálně obsahuje
   5.3. Zkontroluj ukončovací podmínky
6. Vrať seznam shluků společně s dokumenty, které do nich byly přiřazeny

Kromě nutného určení počtu shluků před aplikací algoritmu je další nevýhodou K-means také závislost na inicializačním nastavení středů shluků. Metodou, která může vést k dosažení lepších výsledků, je vícenásobné spuštění inicializační části a následný výběr nejlepšího rozložení centroidů, které bude použito pro fázi zařazování dokumentů do jednotlivých shluků.

3.3.1.2 Hierarchické metody

U hierarchických metod dochází k rozdělování dokumentů do stromové struktury metodou zespoda nahoru (aglomerativní algoritmy) nebo shora dolů (rozdělovací algoritmy). Výsledné hierarchických algoritmů se zpravidla znázorňují pomocí binárních stromů či dendrogramů. Jejich výhodou je schopnost zpracovávat dat různého typu na libovolné úrovni dělení shluků.

Aglomerativní metody na počátku řadí každý dokument do samostatného shluku. Na základě podobnosti dokumentů shluky postupně spojují až do té doby, než vznikne jeden shluk obsahující všechny dokumenty. V některých případech je spojování zastaveno ve chvíli, kdy je vytvořeno *k* shluků zadaných uživatelem.

U rozdělovacích algoritmů se naopak začíná s jedním shlukem obsahujícím všechny dokumenty, který se dělí na menší části tak dlouho až je vytvořen uživatelem zadaný počet shluků, případně do té doby, než je každý dokument v samostatném shluku (Srivastava, Sahami, 2009).

3.4 Sémantické modely

Jedním ze základních modelů využitelných pro numerickou reprezentaci textových dat je Booleovský model. V tomto modelu se dokumenty porovnávají na základě lexikálního výskytu termů ve srovnávaných dokumentech. Problémem při využití tohoto modelu je zejména fakt, že jako relevantní dokumenty jsou vráceny pouze ty, které lexikálně obsahují vyhledávané termy.

Pokročilejším modelem tradičně využívaným v oblasti IR je vektorový model (Salton, McGill, 1983). Ve vektorovém modelu jsou dokumenty a jejich termy reprezentovány v n-rozměrném prostoru, kde hodnota n odpovídá počtu unikátních termů. Souřadnice n-tého termu pro i-tý dokument označuje, jak moc je daný dokumenty pro tento term relevantní. Hodnoty vektorů zde již neoznačují binární přítomnost termu v dokumentu, ale častěji jejich počet výskytů, označovaného jako term frequency (TF). Hodnoty vektorů však mohou zkreslovat termy s vysokou frekvencí výskytu nesoucí minimální množství informace, které se v přirozeném jazyce přirozeně objevují. Příkladem takovýchto termů v českém jazyce jsou například předložky a spojky, v anglickém jazyce je typickým term *the*, nesoucí v podstatě nulovou informaci. Toto možné nežádoucí zkreslení je odstraňováno pomocí váhování důležitosti slov. Často využívanou hodnotou při váhování je inverzní četnost výskytu termu ve všech dokumentech (Inverse document frequency, IDF). Tato hodnota ohodnocuje jako důležité termy takové, jež se v sadě dokumentů objevují méně a naopak jako méně dů-

žité termy ty, které se vyskytují velice často. Kombinací TF a IDF lze získat schéma pro vážení důležitosti termu t v dokumentu d daného vzorcem

$$TF\text{-}IDF_{t,d} = TF_{t,d} \times IDF_t \quad (1)$$

kde $TF_{t,d}$, lze vypočítat jako

$$TF_{t,d} = n_{td} / \sum n_d \quad (2)$$

pro n_{td} označující počet výskytů termu t v dokumentu d. $\sum n_d$ určuje počet termů v dokumentu d. Hodnotu IDF_t, lze získat jako

$$IDF_t = \log N / n_t \quad (3)$$

kde N označuje počet dokumentů a n_t počet výskytů (document frequency) termu t napříč všemi dokumenty. Po normalizaci metodou TF-IDF je k dispozici zpravidla velice řídká matice, jejíž počet dimenzí pro reálné soubory dokumentů dosahuje až desítek tisíc. Velikost prostoru je jednou z hlavních nevýhod vektorového prostoru, která navíc s přibývajícími dokumenty exponenciálně roste. (Salton, McGill, 1983)

Modernější metoda LSA – metoda latentní analýzy, podrobněji popsaná v kapitole 3.6 – je klasickému vektorovému modelu podobná s tou výjimkou, že dimenze prostoru neurčují termy, ale jsou předem přesně definovány. Počet dimenzí je výrazně menší než počet termů a zpravidla se tento počet dimenzí stanovuje experimentem.

LSA analýza se skládá ze čtyř základních kroků, z nichž první dva jsou společné i pro tradiční modely jako například vektorový model nebo pravděpodobnostní modely. Prvním krokem je sestavení tzv. matice výskytů termů v dokumentech (Term-Document Matrix, Term-by-document matrix) označované A. Kolekce dokumentů jsou převedeny do matice, kde řádky reprezentují termy a sloupce dokumenty. Hodnoty matice vyjadřují počet výskytů termů v jednotlivých dokumentech. Pořadí slov v této matici není podstatné a často se lze setkat s označením matice názvem „bag of words“.

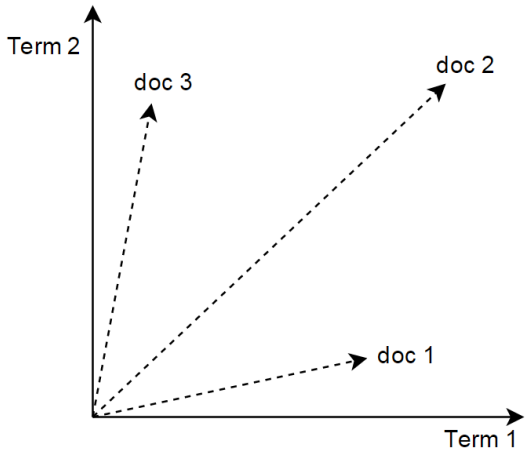
Po získání hodnot matice A je druhým krokem jejich určitá transformace. Jen velmi zřídka se pracuje přímo s hodnotami počtů výskytů slov, častější je, že jsou hodnoty určitým způsobem normalizovány, například dříve zmíněnou metodou TF-IDF.

Pro modely latentní analýzy je dalším krokem redukce dimenzí. Jde o převedení problému do menšího k-dimenzionálního prostoru aproximací matice A. Metod využitelných pro redukci prostoru existuje celá řada, například analýza hlavních komponent (Principal Component Analysis, PCA), faktorová nalýza (Factor Analysis), analýza vlastních vektorů (Eigenvector Analysis) nebo nejčastěji používaná metoda singulárního rozkladu hodnot (Singular Value Decomposition, SVD), která bude popsána v části textu týkající se nalezení reprezentativních konceptů originálních dat.

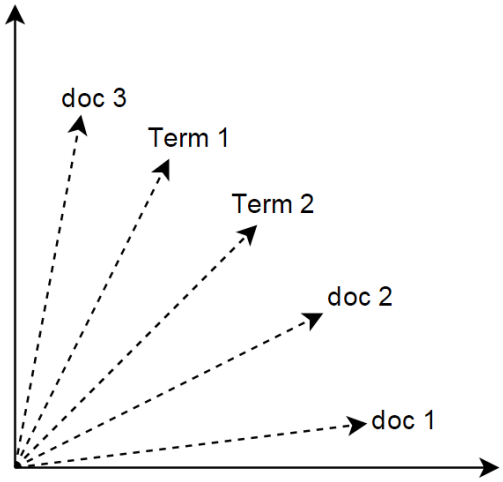
Posledním krokem je nalezení podobností v redukovaném prostoru, namísto vyhledávání v originální matici *A*. Podobnost se měří jako vzdálenost vektorů v redukovaném prostoru. Nejčastěji používanou metodou pro měření vzdáleností vektorů je kosinová vzdálenost, jelikož bylo dokázáno, že jde o nejefektivnější způsob pro mnoho problémů z oblasti IR. Výhodou redukovaného prostoru je to, že termy i dokumenty jsou reprezentovány ve stejném prostoru a lze tak jednoduše hledat podobnosti typu term – term, term – dokument, nebo dokument – dokument. Pomocí nalezených podobností mohou být odhaleny určité skryté (latentní) vazby mezi termy a dokumenty.

Rozdíl mezi tradičním vektorovým modelem, kde je každý dokument reprezentován vektorem vah termů v dokumentu a modelem využívajícím redukovaný dimenzionální LSA prostor je o to patrnější pokud se vyjádří graficky. Grafická interpretace vektorového modelu je zobrazena na Obrázku 1. Termy určují dimenze, případně osy prostoru. Dokumenty jsou reprezentovány jako vektory v tomto prostoru, jejichž délku a směr určují hodnoty výskytů termů v dokumentu z matice *A*. Důležitým faktem v tomto modelu je skutečnost, že pokud dokument neobsahuje určitý term, podobnost s dotazem obsahujícím pouze tento term se rovná nule. Pokud by byl položen dotaz obsahující slovo *bicykl*, dokumenty obsahující term *kolo* by například nebyly uvažovány jako podobné.

Grafická reprezentace LSA prostoru je zobrazena na Obrázku 2. Osy prostoru jsou odvozeny od SVD jako lineární kombinace termů. Jak dokumenty, tak termy jsou reprezentovány jako vektory v tomto *k*-dimenzionálním LSA prostoru. Pozice vektorů termů odrážejí propojení s jejich výskytem napříč dokumenty. Jako podobné mohou být na dotaz následně získány i dokumenty přímo neobsahující termy z položeného dotazu.



Obr. 1 Schéma prostoru klasického vektorového modelu



Obr. 2 Schéma prostoru LSA

V oblasti Information retrieval se již ve svých začátcích objevovali pokusy odhalit určité latentní vazby v datech. Využívalo se shlukování dokumentů a termů, analýza latentních tříd nebo například faktorová analýza. Společným pro tyto přístupy byla snaha representovat do společného prostoru termy nebo dokumenty. Výjimkou byl návrh Metthew B. Kolla (1979) reprezentovat obojí, jak termy, tak i dokumenty, do stejného prostoru konceptů. Ačkoli byla tato myšlenka velice podobná LSA, prostor konceptů využíval pouze velice málo dimenzí, které byly navíc voleny ručně a nebyly ortogonální, jako je tomu u SVD. Dříve byly všechny zmíněné pokusy limitovány nedostatky výpočetního výkonu a dostupností velkých textových kolekcí ve strojově čitelné formě. Spolu s vývojem na poli informatiky jsou tyto problémy a nedostatky postupně řešeny.

Dvojice vědců Dumais a Deerwester popisovala okolo doku 1990 aplikaci LSA na oblast Information retrieval. Využívali několik testovacích kolekcí dat a srovnávali LSA model s klasickým vektorovým modelem. Testovací kolekce se typicky skládá ze sady dokumentů, uživatelských dotazů a usouzení popisujících, které dokumenty jsou relevantní na konkrétní dotazy. Každý dokument byl ohodnocen podle podobnosti na položený dotaz. Výkonnost Information retrieval systémů se obvykle měří pomocí dvou ukazatelů *precision* a *recall* (Landauer, McNamara, Dennis, Kintsch, 2014).

$$precision = \frac{|\{relevantní\ dokumenty\} \cap \{vybrané\ dokumenty\}|}{|\{vybrané\ dokumenty\}|} \tag{4}$$

$$recall = \frac{|\{relevantní\ dokumenty\} \cap \{vybrané\ dokumenty\}|}{|\{relevantní\ dokumenty\}|} \tag{5}$$

Recall (koeficient úplnosti) značí poměr relevantních dokumentů v kolekci vybraných systémem jako relevantní na pokládaný dotaz. Precision (koeficient přesnosti) lze chápat jako pravděpodobnost, že vybraný dokument je relevantní. Při vyhledávání je cílem maximalizovat oba tyto ukazatele. Ukazuje se však, že

jsou na sobě závislé, tedy se zvyšující se přesností klesá úplnost a naopak. Je tedy potřeba hledat určitou rovnováhu mezi precision a recall. Dalším ukazatelem možným k měření výkonosti text miningových modelu je F-measure. Hodnota tohoto ukazatele kombinuje precison a recall a je vypočítána jako jejich harmonický průměr (Zhu, 2004).

Z pokusů na testovacích datech vyplynul také fakt, že je poměrně jednoduché najít několik málo relevantních dokumentů, ale k nalezení všech relevantních dokumentů odpovídajících položenému dotazu je potřeba prozkoumat veliké množství irelevantních dokumentů. Výkon LSA modelu byl podle testování vždy lepší než standartní vektorový model (Landauer, McNamara, Dennis, Kintsch, 2014).

Pro úlohy v oblasti text miningu je z předchozích informací patrné, že využití redukce dimenzí je vhodné nejen z důvodu snížení paměťových a výpočetních nároků, ale dokáže také odhalit jisté skryté (latentní) vazby mezi daty a poskytnout cenné informace. V rámci práce byl pro úlohy z oblasti IR vytvořen program pro vyhledávání podobností v datech redukovaných metodou SVD, jehož popisu se věnuje kapitola 4.3.

3.5 Metody předzpracování textových dat

Textová data, ukládaná za účelem dalšího zpracování, nemají zpravidla žádnou ucelenou podobu. Pro možnost aplikace data miningových metod, zvýšení jejich efektivity a dosahování lepších výsledků je vhodné data před zpracováním nejdříve určitým způsobem předzpracovat. Jak již bylo dříve zmíněno, textová data je před zpracováním nutné převést do určité numerické reprezentace. Velikost výsledného, obvykle vektorového, prostoru je dána počtem zpracovávaných dokumentů, ale zejména počtem unikátních termů v těchto dokumentech. Snížením počtu těchto termů lze redukovat řešené problémy, odstranit z dat přebytečný šum a prvky nesoucí minimální množství informace. Do dat lze také zanést určité statistické vazby mezi termy a dokumenty, které mohou výsledek taktéž ovlivnit. V následující části bude zmíněno několik použitelných technik předzpracování dat.

3.5.1 Case folding

Stejná slova se napříč dokumenty nemusejí vyskytovat vždy ve stejné podobě. Ze stejného slova, vyskytujícího se jednou na začátku a podruhé uprostřed věty, vznikají prakticky dva unikátní termy[1], což následně zvyšuje prostor, ve kterém se bude s daty pracovat, zároveň tím mohou být ztraceny určité skryté (latentní) vazby mezi termy a dokumenty. Aby bylo této možné ztrátě či nárůstu dimenzionality prostoru zamezeno, využívá se metody case folding, což je metoda pře-

[1] V případě správného využití verzálek a minusek v textových datech, popřípadě při výskytu překlepů. Příklad: „Car“, „car“, „CaR“, „caR“ apod.

vodu veškerého textu do upper-case případně lower-case podoby (Neto, Santos, Kaestner, Freitas, 2000).

3.5.2 Odstranění stop slov

Stop slova jsou slova, která se v textu vyskytují ve velkém počtu a příliš nesouvisí s obsahem dokumentu. Typickým příkladem v anglickém jazyce mohou být slova „the“, „can“ nebo „will“. Odstranění stop slov vede ke snížení velikosti dokumentu a tím zvýšení rychlosti a efektivnosti zpracovávajícího algoritmu (Neto, Santos, Kaestner, Freitas, 2000). Seznamy stop slov pro různé jazyky lze získat z veřejných databází a na konkrétních datech poté testovat změny efektivnosti, na základě různě obsáhlých seznamů stop slov.

3.5.3 Odstranění slov s nízkou četností

Slova vyskytující se v textu s velice malou četností většinou na výsledky dolování znalostí z textových dat mají pouze malý či vůbec žádný vliv. Odstraněním slov vyskytujících se v rozsáhlých kolekcích dokumentů pouze jedenkrát lze docílit až poloviční redukce unikátních termů bez jakéhokoliv negativního vlivu na výsledky metod text miningu (Žižka, Dařena, 2011).

3.5.4 Stemming

S využitím metody stemmingu se z původních slov vytváří nová tak, že dochází k odstranění částí slov vzniklých skloňováním nebo časováním, předpon přípon a morfologických koncovek. Cílem je získat termy v unifikované formě v jejich kořenovém tvaru (angl. stem). Existuje několik typů algoritmů pro stemming, které se liší v přesnosti a schopnosti překonávat překážky vyskytující se v nestrukturovaných textových datech (Řezníček, Dařena, 2013):

- Porterův algoritmus – nejznámějším stemmovací algoritmus původně vyvinutým pro stemming anglického jazyka, jenž se stal postupem času v oblasti stemmingu určitým standardem.
- Vyhledávací algoritmy – používají vyhledávání skloňovaných tvarů termů ve vyhledávací tabulce. Výhodou těchto algoritmů je jednoduchost a rychlost. Nevýhodou je nutný výčet všech skloňovaných forem termů uvedený ve vyhledávací tabulce. Neúplný výčet poté způsobuje nepřesnosti při zpracování neznámých slov.
- Produkční techniky – vyhledávací tabulky jsou zde vytvářeny poloautomaticky a dokáží si lépe poradit s neznámými slovy.
- Algoritmy pro odstraňování přípony (suffix-stripping) – jejich součástí není vyhledávací tabulka, ale list pravidel, podle kterých algoritmus postupuje. Tento typ algoritmu je založen zejména na oddělování přípon termů a pro jeho použití nejsou nutné příliš veliké znalosti lingvistiky. Problémy s tímto algoritmem nastávají v případě, že kořen slova nelze získat pouze odstraněním přípony popřípadě předpony (např. nepravidelná anglická slovesa).

- Stochastické algoritmy – využívají k identifikaci kořenového tvaru termu pravděpodobnost a jsou trénovány pomocí tabulky kořenových tvarů, která je v průběhu zpracování ovlivňována a jejich pravděpodobnostní model je postupem času vyvíjen.
- Hybridní přístupy – k dosažení výsledků kombinují dva nebo více přístupů popsaných výše. Příkladem je kombinace vyhledávacích tabulek a odstraňování přípon, kde pokud není slovo ve vyhledávací tabulce, uplatní se algoritmus pro odstranění přípony.
- Porovnávací algoritmy – používají databázi stem slov, které nemusí být nutně validními slovy, ale spíše určitými běžnými podřetězci slov (např. „brows“ ve slově „browse“ a ve slově „browsing“).

Většina stemmovacích algoritmů byla vytvořena pro zpracování anglického jazyka, v oblasti text miningu se však zpracovávají textová data nejen v anglickém jazyce, což bylo jedním z důvodu vzniku jazyka Snowball jako nadstavby Porterova algoritmu, umožňujícího vytvářet vlastní algoritmy pro stemming a pomocí těchto algoritmů předzpracovávat zdrojová data a získávat z nich další informace (TARTARUS, 2012).

3.6 Nalezení reprezentativních konceptů

3.6.1 Latentní sémantická analýza

Latentní sémantická analýza (Latent semantic analysis, LSA), někdy označována jako Indexace latentní sémantiky (Latent semantic indexing, LSI), byla ve svých začátcích využívána jako nástroj pro efektivní vyhledávání textových dokumentů. Jde o variantu vektorového modelu, jehož originální data byla aproximována pomocí metody singulárního rozkladu (SVD) případně jiných metod jako například analýza hlavních komponent (PCA). Na základě experimentů bylo již v minulosti dokázáno, že metody redukce dimenzionality dat mohou přinášet jisté výhody. Jednou z těchto výhod je odstranění šumu vyskytujícího se v originálních datech. Další výhodou je mnohdy velice výrazné zmenšení prostoru výsledné matice. Metodou redukce dimenzionality lze také v dokumentech objevit určité skryté (latentní) vztahy mezi slovy (Berry, Drmač, Jessup, 2009) (Landauer, McNamara, Dennis, Kintsch, 2014).

3.6.1.1 Rozklad Term-Document matice metodou SVD

Nechť symbol A označuje matici $m \times n$. Cílem SVD je poté nalézt rozklad podle následujícího vzorce

$$A = USV^T \tag{6}$$

takový, že S je nezáporná diagonální matice $m \times n$, na jejíž diagonále se nacházejí tzv. singulární čísla. Matice U označuje $m \times m$ orthogonální matici, kde $U^T = U^{-1}$ a sloupce matice U jsou nazývány levými singulárními vektory. Matice

V poté označuje $n \times n$ orghogonální matici, kde $V^T = V^{-1}$ a její sloupce jsou nazývány pravými singulárními vektory. Výpočet rozkladu je zpravidla prováděn tak, že singulární hodnoty $\sigma_1, \sigma_2, \ldots, \sigma_{\min(m,n)}$ na diagonále matice S jsou seřazeny sestupně, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{\min(m,n)}$. Pro rozsáhlé matice je výpočet rozkladu časově velice náročnou operací. Výpočet vlastních čísel pomocí naivního algoritmu vede k faktoriálové časové složitosti, proto jej nelze pro reálně velké matice použít a využívá se jiných numerických metod. Navíc se ukazuje, že výpočet singulárních čísel okolo nuly může zhoršovat kvalitu vyhledávání. Pro reálné výpočty je tedy vhodné využívat pouze prvních k singulárních hodnot a jejich příslušných singulárních vektorů. Provádí se tedy převod do k-dimenzionálního redukovaného prostoru jako

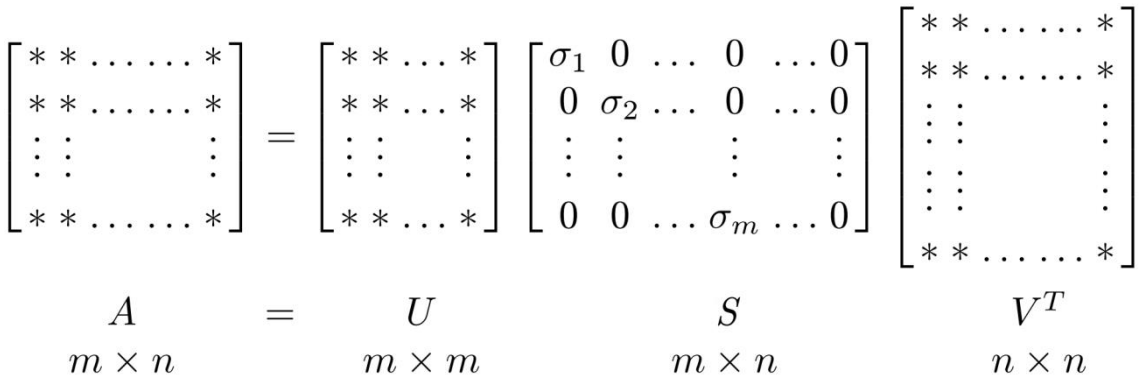
$$A_k = U_k S_k V_k^T \tag{7}$$

kde symboly U_k, resp. V_k^T, označují $m \times k$, resp. $n \times k$, matice obsahující prvních $k$sloupců matice U, resp. matice V. Symbol S_k označuje matici obsahující prvních $k$největších singulárních hodnot původní matice S. Vybrané sloupce matice V_k^T reprezentují dokumenty původní databáze, řádky matice U_k poté reprezentují termy vyskytující se v této databázi. Pomocí SVD lze vzhledem ke sloupcovým vektorům matici A aproximovat a získat tak tři samostatné matice. Následně pak k-tá aproximace (A_k) matice A je sestavena na základě výběru prvních k singulárních hodnot matice S, přičemž ostatní hodnoty této matice jsou zanedbány. Schématické zobrazení singulárního rozkladu pro matici řádu $m \times n$ kde $m > n$, lze vidět na Obrázku 3, případně na Obrázku 4 pro matici kde $m < n$.

$$\begin{bmatrix} * & * & \ldots & * \\ * & * & \ldots & * \\ \vdots & \vdots & & \vdots \\ * & * & \ldots & * \end{bmatrix} = \begin{bmatrix} * & * & \ldots & * \\ * & * & \ldots & * \\ \vdots & \vdots & & \vdots \\ * & * & \ldots & * \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \ldots & 0 \\ 0 & \sigma_2 & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \ldots & \sigma_n \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix} \begin{bmatrix} * & * & \ldots & * \\ * & * & \ldots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \ldots & * \end{bmatrix}$$

$$\underset{m \times n}{A} = \underset{m \times m}{U} \quad \underset{m \times n}{S} \quad \underset{n \times n}{V^T}$$

Obr. 3 Schéma transformace matice metodou SVD pro m > n



Obr. 4 Schéma transformace matice metodou SVD pro m < n

Algoritmus pro výběr optimálního počtu singulárních hodnot není dosud znám, a proto je nutné hodnotu k stanovovat experimentem. Značný dopad na výsledek vyhledávání má volba správného k zejména proto, že aproximací matice A maticí A_k vzniká aproximační chyba vedoucí k následujícím skutečnostem. Chybu aproximace A maticí A_k lze podle věty Eckarta a Younga vypočítat následovně:

$$\|A - A_k\| = \min_{rank(B)\leq k}\|A - B\| = \sqrt{\sigma_{k+1}^2 + \sigma_{k+2}^2 + \ \ldots + \sigma_{r_A}^2} \qquad (8)$$

Čím méně singulárních hodnot se při aproximaci využije, tím více chyba roste. Naopak, čím větší hodnota k je použita pro aproximaci, tím menší je následná chyba. Zásadním, při aproximacích maticemi A_k nízkých hodností, je fakt, že se do sloupcových vektorů mohou promítat hodnoty z jiných sloupců. Děje se tak na základě možné existence skrytých (latentních) vazeb mezi nimi. Získaný model navíc rozšířený o jisté vazby mezi dokumenty a termy lze následně využít pro efektivní vyhledávání podobností. Podobnost vektorů latentního prostoru se u LSA měří pomocí kosinové vzdálenosti podle následujícího vzorce (Krátký, 2002):

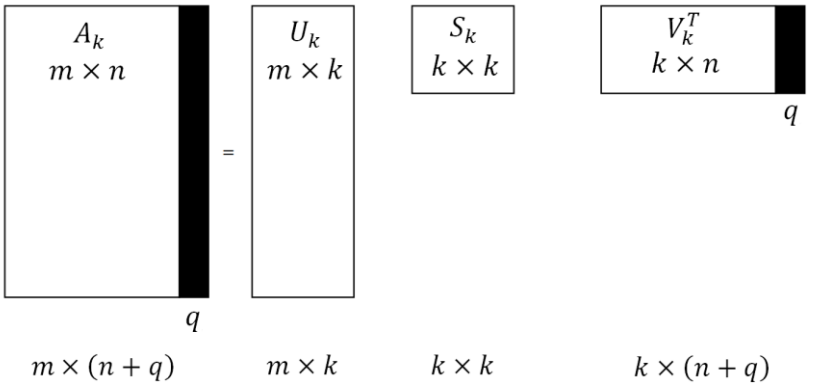
$$similarity = \cos(\theta) = \frac{A*B}{\|A\|*\|B\|} = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} * \sqrt{\sum_{i=1}^{n}(B_i)^2}} \qquad (9)$$

Ačkoli je výpočet singulárního rozkladu náročnou operací, provádí se zpravidla pouze jednou a to při indexaci. Při dalších operacích s daty je využito již uloženého vypočítaného rozkladu někdy nazývaného též LSI databáze. Ne vždy je však v oblasti text miningu výkonnost LSA modelu lepší než využití klasického vektorového modelu. K horším výsledkům vede zejména použití velkého počtu termů ve vyhledávaném dotazu. Negativní vliv může na výsledky mít také nesprávný počet použitých singulárních hodnot, ať už počet příliš nízký nebo naopak příliš vysoký.

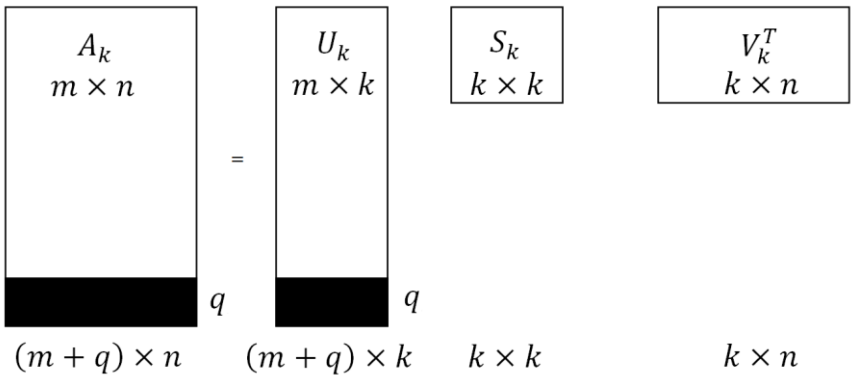
3.6.1.2 Aktualizace LSI databáze

Pokud je k dispozici již předem získaná LSI databáze je potřeba do ní zařadit další dokumenty, které mohou obsahovat nové termy, existují dvě možnosti: kompletní přepočet SVD nové matice výskytů termů v dokumentech nebo SVD-updating (O'Brien, 1994).

Metoda přepočítání SVD je implementačně nejjednodušší avšak výpočetně nejnáročnější, zejména při zpracování reálných sad dokumentů v řádech desítek tisíců a větších může narážet na omezení paměťového prostoru. Metodou, které se vyhýbá nutnosti přepočtu celého SVD s nově vloženým dokumentem je SVD-updating. Metoda spočívá v přidání nových termů a dokumentů do již existující LSI databáze. Každý nový dokument je reprezentován jako vektor, který je následně zařazen do matice V (znázorněno na Obrázku 5). Stejně tak každý nový term je reprezentován vektorem a je zařazen do matice U (znázorněno na Obrázku 6).



Obr. 5 Schéma přiřazení nového dokumentu do LSI databáze



Obr. 6 Schéma přiřazení nového termu do LSI databáze

Pro zařazení nového $m \times 1$ vektoru dokumentu, d, do již existující LSI databáze, je potřeba nejdříve vypočítat příslušný vektor termů, d_t, jako vážený vektor termů v dokumentu. A následně pak vytvořit vektor d_p vypočítaný jako:

$$d_p = d_t V_k S_k^{-1} \tag{10}$$

Podobně je potřeba pro zařazení nového $n \times 1$ vektoru termu, t, do již existující LSI databáze nejdříve vypočítat příslušný vektor dokumentů, t_d, jako vážený vektor vážených dokumentů. A následně pak vytvořit vektor t_q vypočítaný jako:

$$t_q = t_d U_k^T S_k^{-1} \tag{11}$$

3.6.1.3 Možná vylepšení SVD

Singulární rozklad matice je operace výpočetně poměrně náročná a pro zpracování rozsáhlých databází dokumentů je potřeba dostatečný výpočetní výkon. Existují však i jiné metody, případně vylepšení metody SVD, které jsou méně náročné a dosahují stejných, někdy i lepších výsledků aproximace.

Přírůstková metoda výpočtu SVD

Tradiční SVD algoritmus využívající tzv. „in-core“ výpočet je zpravidla velice náročný na paměťové zdroje. Matice *výskytů* A musí být před samotným výpočtem nejdříve celá nahrána do paměti. Při zpracovávání reálných databází dokumentů (až stovky tisíc dokumentů), kde po předzpracování vstupu vznikají obrovské řídké matice, se nahrání jejich celého obsahu do paměti může stát prakticky nemožným. Omezením je zde zejména množství paměti počítače. Byla však již popsána a vyvinuta technika zvaná „out-of-core“ SVD, ve které jsou zdrojová data z disku čtena sekvenčně po menších částech a výpočty hodnot rozkladu jsou prováděny přírůstkově. Přírůstkový způsob vede k určitému snížení přesnosti výpočtu, nicméně dají se stanovit metriky definující množství ztráty v přesnosti výpočtu. Podle těchto metrik se lze rozhodnout, kdy nahradit přírůstkový výpočet klasickým SVD.

Menší paměťová náročnost je v případě přírůstkového výpočtu spojena s delší dobou výpočtu singulárního rozkladu. Při zpracování rozsáhlých sad dokumentů lze akceptovat délku výpočtu rozkladu v řádech jednotek až desítek minut, stále se však metoda považuje za vhodnou i přes určité nepřesnosti vznikající během výpočtu. S jejím využitím lze teoreticky zpracovávat matice o řádech až 5,000,000 (Lin, 2000).

3.6.2 Pravděpodobnostní latentní sémantická analýza

Jedním z dalších možných způsobů jak dosáhnout redukce prostoru je pravděpodobnostní latentní sémantická analýza (Probabilistic Latent Semantic Analysis, pLSA), která se vyvinula z již dříve popisované LSA. Tato metoda řeší stejný problém, ale místo lineární algebry používá jako svůj základ statistiku a pravděpodobnost. Pro identifikaci T témat v sadě dokumentů je potřeba nejprve pro

každé téma i vypočítat pravděpodobnost výskytu pro všechny termy t jako $P(t|i)$. Téma lze chápat jako skupinu termů, které se v dokumentech často vyskytují společně. Za předpokladu, že pro každý dokument existuje pravděpodobnost výskytu tématu i v dokumentu D, jako $P(i|D)$, lze odvodit pravděpodobnost výskytu termu t v dokumentu D jako $\sum_{i=1}^{k} P(t|i)P(i|D)$. Pravděpodobnosti $P(t|i)$ a $P(i|D)$ jsou však na počátku výpočtu neznámé a je potřeba je identifikovat. Standartní metodou je maximalizace věrohodnosti dat, což je případ optimalizační úlohy. Výsledkem je reprezentace každého dokumentu pomocí vektoru témat, kde každá položka vektoru reprezentuje pravděpodobnost výskytu tématu i v dokumentu D. Pro měření podobnosti dokumentů se namísto kosinové vzdálenosti vektorů používá spíše Hellingerova vzdálenost[2], případně KL-divergence[3]. Nevýhodou tohoto modelu je jeho sklon k přeučování a také nemožnost predikce vektoru témat pro dokumenty, jež nebyly obsaženy v trénovací množině dokumentů (Landauer, McNamara, Dennis, Kintsch, 2014; Materna, 2011).

3.6.3 Dirichletova alokace

Pokročilejší metodou pravděpodobnostní sémantické analýzy je metoda s názvem Latentní Dirichletova alokace (Latent Dirichlet Allocation, LDA). Tato metoda řeší problémy modelu pLSA jako je tendence k přeučování a neschopnost identifikace témat pro dokumenty, které nebyly součástí trénovací množiny. LDA umožňuje úsporným způsobem popsat kolekci dokumentů se zachováním cenných informací statistické povahy. Základem LDA jsou pro identifikaci k témat dvě základní rozdělení pravděpodobnosti – binomické a beta rozdělení, respektive jejich vícerozměrné varianty multinomické rozdělení[4] a Dirichletovo rozdělení[5]. Pro identifikaci témat je potřeba vyjádřit vektor témat pro každý dokument a rozdělení pravděpodobnosti termů pro každé téma. Stejně jako u pLSA se i zde jedná o maximalizační úlohu, která se řeší s využitím aproximativních metod, jako jsou variační řady[6] případně Gibbsovo vzorkování[7] (Materna, 2011).

[2] http://en.wikipedia.org/wiki/Hellinger_distance

[3] http://en.wikipedia.org/wiki/Kullback–Leibler_divergence

[4] http://en.wikipedia.org/wiki/Multinomial_distribution

[5] http://en.wikipedia.org/wiki/Dirichlet_distribution

[6] http://en.wikipedia.org/wiki/Variational_Bayesian_methods

[7] http://en.wikipedia.org/wiki/Gibbs_sampling

4 Metodika

V rámci práce nejdříve získám sady nestrukturovaných textových dat a provedu jejich předzpracování. Předzpracování bude provedeno několika různými metodami a jejich kombinacemi. Textová data budou následně převedena do vektorového modelu, ze kterého bude pomocí metody SVD vytvořena LSI databáze. Provedu experimenty dokazující určité výhody využití latentní sémantické analýzy při vyhledávání podobných termů a dokumentů na základě pokládaných dotazů. Pomocí třech různých klasifikátorů budu zkoumat vliv různých metod předzpracování dat na výsledky klasifikace, jejichž efektivita bude měřena na základě metrik precision, recall a F-measure. Rozdíly mezi klasickým vektorovým modelem a LSI databází budu zkoumat také při procesu shlukování pro různě předzpracovaná data, kde se zaměřím zejména na metriku silhouette. Výsledky experimentů a měření budou v práci představeny a následně zhodnoceny v samostatných kapitolách.

4.1 Zdrojová data

Textová data, použitá v práci jako zdroj pro dolování znalostí, jsou data popsaná v práci Žižky a Dařeny. Textová data představují recenze zákazníků, jež si prostřednictvím internetové služby objednali ubytování v mnoha různých hotelech a zemích po celém světě. Recenze jsou psány zákazníky, kteří si udělali rezervaci online a skutečně v hotelu pobývali, jsou psány v mnoha jazycích a lze je rozdělit do dvou skupin – pozitivní a negativní zkušenosti s hotelem. Všechny recenze jsou psány v přirozeném jazyce a velká část z nich obsahuje nedostatky, typické pro texty psané v přirozených jazycích (překlepy, zpřeházená písmena, chybějící písmena, gramatické chyby, kombinace dvou jazyků v jedné recenzi a další).

K dispozici jsem měl několik milionů recenzí v celkem 18 jazycích, z nichž jsem si pro podrobnější zpracování vybral 4 jazyky – angličtina, němčina, španělština a francouzština. Ve zdrojových datech se jako nejkratší vyskytují recenze obsahující pouze jedno slovo, naopak nejdelší recenze obsahují více než 300 slov (389 pro angličtinu, 308 pro němčinu, 346 pro francouzštinu a 352 pro španělštinu). Průměrná délka recenze pro vybrané jazyky je přibližně 20 slov (24 pro angličtinu, 18 pro němčinu, 20 pro španělštinu a francouzštinu) (Žižka, Dařena, 2011). Pro měření byla z množiny recenzí pro každý ze 4 zvolených jazyků vybrána pouze určitá část dat obsahující 15 000 recenzí.

4.2 Předzpracování

Jelikož byla zdrojová data sesbírána pomocí online dotazníku formou otevřených otázek, nelze v těchto datech předpokládat jakoukoliv dodrženou strukturu. V recenzích zákazníků hotelů se navíc kromě alfanumerických znaků mohou vyskytovat téměř jakékoliv další znaky, které je nutné před zahájením procesu dolování informací z dat odstranit. Jednou z částí práce je srovnání dopadu růz-

ných forem předzpracování dat na úspěšnost text miningových metod. Kromě odstranění nežádoucích znaků, jimiž jsou prakticky veškeré znaky nealfanumerické, v práci proto jako jeden ze způsobů předzpracování dat využívám metodu odstraňování stop-slov. Z dat jsou tímto způsobem odstraněny termy nesoucí pouze minimální množství informace a pro úlohy klasifikace, shlukování či vyhledávání dalších informací v datech nemají větší význam. Druhým možným způsobem předzpracování dat, který v práci využívám je proces stemmingu. Jeho aplikací je získán menší počet unikátních termů, díky čemuž se zmenšuje prostor, do kterého jsou následně data převáděna.

Jak již bylo dříve v práci zmíněno, textová data je nutné pro možnost aplikace text miningových metod nejdříve převést do určité numerické reprezentace. Booleovský model byl z důvodů omezení s ním spojených jako numerická reprezentace dat zamítnut a v práci využívám modelu vektorového. Metodou SVD jsou následně data redukována do vhodného počtu dimenzí a výsledky text miningových metod v tomto latentním sémantickém prostoru jsou porovnávány s výsledky dosaženými s využitím klasického vektorového modelu.

4.2.1 Odstranění stop-slov

Odstranění stop slov, operace odstranění často se vyskytujících termů, nesoucí minimální množství informace vede k očištění dat, kde následně zůstávají pouze termy určitým způsobem zajímavé. Slovníky stop slov pro celkem 29 jazyků jsem získal z veřejné databáze projektu stop-words[8]. Získaný archiv obsahuje pro každý z dostupných jazyků několik různě rozsáhlých souborů s výčtem stop-slov, které byly získány z několika zdrojů a jsou dostupné pod GNU GPL v3 licencí. Slovníky jsou v rámci práce využívány ve fázi předzpracování jako jeden ze zdrojových souborů programu textmining.pl, jenž disponuje funkcí odstranění stop-slov ze zdrojového textu. K jejich dalšímu využití dochází při vyhledávání podobností vůči položenému dotazu v dostupné sadě dokumentů. Pro dosažení přesnějších výsledků jsou z dotazu před jeho další potřebnou transformací stop-slova odstraněna a jeho součástí jsou pouze termy vyskytující se v předzpracované sadě dokumentů.

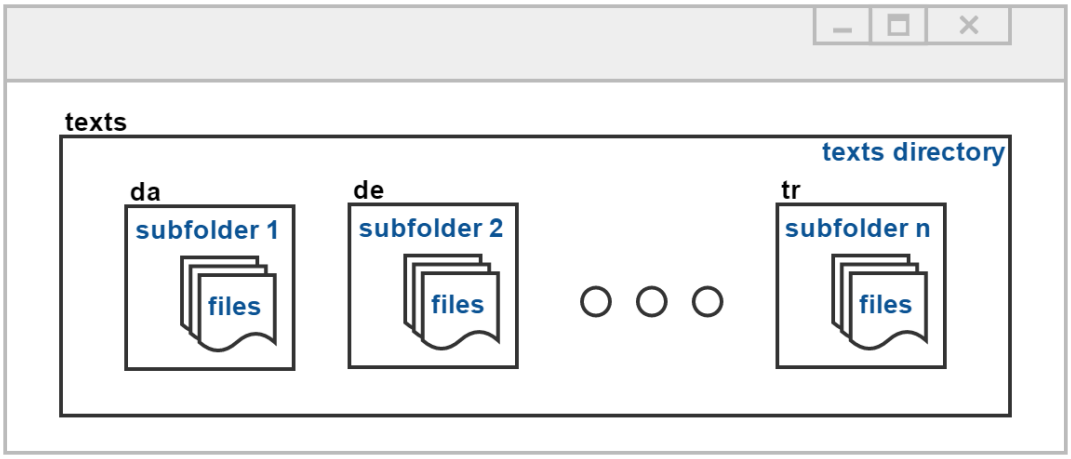
4.2.2 Stemming

Stemming, jakožto proces převedení slov na jejich kořenový tvar, je v práci řešen pomocí samostatného programu. Pro převedení slov zdrojového textu do jejich kořenového tvaru jsem využil program SnowballStemmer.pl, který jsem v minulosti vytvářel v rámci projektu IGA. Program lze spouštět z příkazové řádky a pracuje v několika uživatelských rovinách. Pomocí programu lze proces stemmingu aplikovat na text samostatného souboru, na řetězec zadaný parametrem, případně na adresář se specifickou adresářovou strukturou obsahující libovolné množství textových souborů. Pro první dva uvedené typy použití je nut-

[8] https://code.google.com/p/stop-words/

né vždy definovat jazyk zdrojových dat, podle kterého je následně vybrán vhodný stemmovací algoritmus.

Při zachování určitých konvencí v adresářové struktuře, nastíněných na Obrázku 7, lze jediným příkazem jednoduše zpracovat libovolné množství souborů. To je vhodné zejména při fázi testování, kdy je potřeba z důvodu úspory času pokud možno co nejvíce procesů automatizovat. Program prochází adresářovou strukturou, názvy podadresářů chápe jako názvy jazyků, podle kterých se má vybrat stemmovací algoritmus aplikovaný na sobory v adresářích obsažené. Souborů může být v každém podadresáři libovolný počet, program zpracovává jeden po druhém a postupně na standartní výstup zobrazuje informace o aktuálním stavu.



Obr. 7 Hierarchie adresářové struktury použitelné pro hromadný stemming více souborů

Ačkoliv jde program použít několika různými způsoby, proces stemmingu zůstává pro všechny stejný. V programu pro proces stemmingu využívám jeden z modulů programovacího jazyka Perl, konkrétně modul *Lingua::Stem:Snowball*. Tento modul umožňuje spouštět stemmery jazyka Snowball, zmiňovaného v kapitole 3.5.4, původně určené pro použití v jazyce C. Při inicializaci stemmeru se nejprve definuje kódování vstupních dat a také jejich jazyk. Dále je v programu provedena kontrola, zda je pro daný jazyk dostupný stemmovací algoritmus. Následně jsou načtena zdrojová data a iterativním způsobem je poté řádek po řádku očištěn o nežádoucí znaky jako například html značky, entity, bílé znaky a speciální znaky určitých jazyků. Na očištěná data je následně aplikován proces stemmingu jehož výsledkem je pole obsahující slova ve svém kořenovém tvaru. Tato data jsou na konci zpracování uložena do samostatného souboru (v případě použití pro stemming řetězce zadaného z parametru příkazové řádky je výsledek vypsán na standartní výstup a případně zpracován dalším programem). Aplikací stemovacího algoritmu je oproti původním datům docíleno několika výhod. Zaprvé je docíleno zmenšení velikosti

zdrojových dat, což vede obecně ke zrychlení jejich dalšího načítání a zpracování. Důležitějším faktem však je skutečnost, že po tomto typu předzpracování lze výrazně snížit počet unikátních termů, čímž se se zmenšuje prostor potřebný pro reprezentaci dat.

4.3 Nalezení reprezentativních konceptů

4.3.1 Převod textových dat do vektorové reprezentace

V rámci výzkumu na Ústavu informatiky PEF MENDELU je ve vývoji aplikace určená pro předzpracování textových dat a jejich převedení do numerické reprezentace ve formě matice. Aplikace je implementovaná v jazyce Perl a disponuje vlastním uživatelským rozhraním, lze ji však spouštět také z příkazové řádky. Samotný program disponuje celou řadou funkcí jako například odstranění slov s nízkou nebo naopak příliš vysokou četností výskytu, odstranění nežádoucích symbolů, html tagů a entit či odstranění stop slov. Výstup programu lze získat v několika formátech jako například CSV, CLUTO, ARFF, SVMlight a při převodu dat do vektorové reprezentace lze využít několik různých nastavení jako: typ globálních a lokálních vah, minimální a maximální délka slova, minimální a maximální lokální/globální četnost výskytu slova v dokumentu/dokumentech, definice výstupního souboru a další (Novák, Dařena, 2012).

Pro předzpracování souborů recenzí vybraných jazyků jsem využíval následujícího nastavení zmíněného programu:

- Minimální globální četnost výskytu slova – 5
- Minimální délka slova – 5
- Maximální délka slova – 12
- Lokální váhy vektorů – Term Frequency (TF)
- Globální váhy vektorů – Inverse Document Frequency (IDF)
- Metoda normalizace hodnot – Sum of weights
- Formát výstupního souboru – CSV

4.3.2 Singulární rozklad

4.3.2.1 Volba implementačního jazyka

Algoritmus singulárního rozkladu je již implementován v několika jazycích jako například Perl, C++, MATLAB, Octave či Python. Jelikož jsem, pro celý proces předzpracování dat, využíval skripty napsané v jazyce Perl, zvolil jsem původně tento jazyk i pro výpočet singulárního rozkladu. Základem pro výpočet byla normalizovaná *matice výskytů termů v dokumentech A* interpretovaná jako rozsáhlé řídké pole. Pro práci s n-dimensionálními datovými strukturami lze v jazyce Perl vyžít modulu *PDL*[9], který je pro práci s těmito datovými struktu-

[9] http://pdl.perl.org/

rami optimalizován. Modul obsahuje funkce pro maticové operace jako například inverze matice, výpočet determinantu či singulární rozklad matice. Výpočet singulárního rozkladu s využitím modulu *PDL* je možné paralelizovat a využít tak celý výpočetní potenciál zpracovávajícího stroje.

Výsledkem singulárního rozkladu jsou tři matice U, S a V obsahující levé ortogonální, singulární a pravé ortogonální vektory. Pro menší kolekce dokumentů (do 1000 dokumentů) byly jak paměťové tak časové nároky pro výpočet rozkladu přijatelné (výpočet SVD pro 1000 dokumentů – xxx min yyy s). Se vzrůstajícím počtem zpracovávaných dokumentů (10 – 100 tisíc dokumentů) však doba výpočtu dosahovala až desítek hodin, což bylo pro testování ale zejména pro budoucí reálné použití výpočtu neúnosné.

Z důvodů velké časové ale i prostorové složitosti řešení algoritmu SVD implementovaného v jazyce Perl jsem byl nucen využít alternativní řešení. Tímto řešením byl výpočet singulárního rozkladu s využitím knihovny *SciPy*[10] jazyka Python. *SciPy* je kolekce matematických algoritmů a funkcí umožňující provádět složité manipulace a výpočty nad rozsáhlými daty. Svými funkcemi se knihovna dokáže v úlohách zpracování dat vyrovnat systémům jako MATLAB či Octave. Výhodou této knihovny je také možnost vizualizace dosažených výsledků výpočtů. Pro práci se *SciPy* je potřeba importovat pouze již zmíněné rozšíření *Numpy*, na kterém je knihovna postavena.

Pro výpočet singulárního rozkladu lze využít jednu ze dvou dostupných matod z balíčků *scipy.linalg* a *scipy.sparse.linalg*, které poskytují stejné výsledky, ale liší se způsobem samotného výpočtu. Metoda *svd* z balíčku *scipy.linalg* je implementací klasického algoritmu SVD. Umožňuje definovat, zda mají být kromě singulárních hodnot vypočítány také matice U, V a zahrnuje i kontrolu hodnot vstupní matice. Druhý zmíněný balíček *scipy.sparse.linalg* je optimalizován pro výpočty s tzv. řídkými maticemi, což je přesně typ matice, který je potřeba zpracovat. Balíček obsahuje metodu *svds*, jenž je implementací přírůstkového SVD algoritmu popsaného v kapitole 3.6.1. Výpočet je prováděn v několika iteracích, jejichž počet je možné definovat parametrem funkce. S využitím této metody se oproti metodě *svd* snižují paměťové nároky a je možné se stejným množstvím dostupných prostředků zpracovat řádově až několikrát větší datové kolekce. Metoda produkuje tři matice U, S a V^Tjejichž singulární hodnoty jsou seřazeny vzestupně. Je tedy nejdříve nutné definovat, že požadovaným výsledkem rozkladu jsou největší singulární hodnoty a po výpočtu prohodit pořadí sloupců/řádků matic U, S a V^T.

4.3.2.2 Vytvořený program

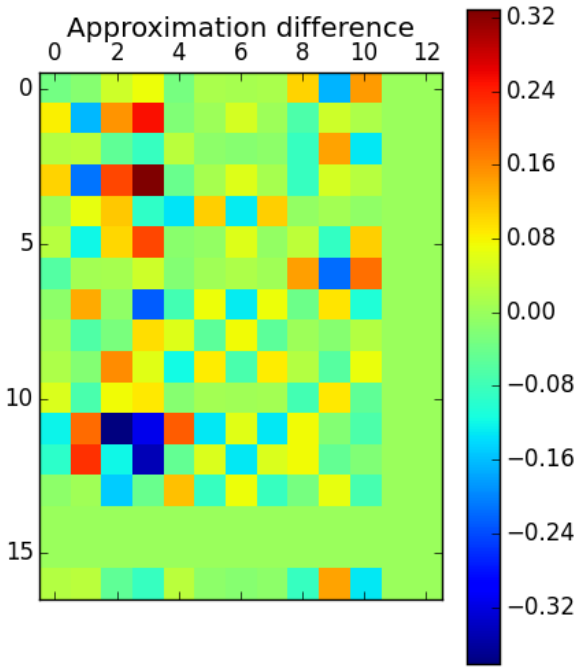
Program, vytvořený pro výpočet singulárního rozkladu, zahrnuje obě dvě zmíněné metody. Metoda *svd* byla využívána zejména pro testování menších kolekcí, kdy bylo potřeba získat výsledky co nejrychleji bez ohledu na paměťové nároky. Pro reálné kolekce dokumentů je pak využita metoda *svds*, s kterou roste

[10] http://docs.scipy.org/

doba výpočtu avšak pro provedení výpočtu je potřeba méně paměťového prostoru. Program lze spouštět z příkazového řádku následujícím příkazem:

```
python SVD.py --input=[OccurancesMatrixFile]
--outputpostfix=[postfixForOutputFiles]
--dimensions=[NumberOfDimensions]
--delimiter=[StringDelimiter]
--method=[SVD | SVDS]
```

Pomocí parametru `--dimensions` lze nastavit do kolika dimenzí bude původní prostor redukován. Parametrem `--method` se volí mezi dvěma popisovanými metodami pro výpočet singulárního rozkladu. Součástí programu pro výpočet SVD je také možnost replikovat z redukovaného prostoru zpět původní matici A aproximovanou (A_approx). Porovnáním původní matice A s maticí A_approx lze určit, jak velké odchylky v hodnotách vznikly s využitím k–dimenzionální redukce a při překročení určité prahové hodnoty použít pro redukci jiný počet dimenzí. Odečtením matice A_approx od původní matice A vzniká matice pojmenované $approximationDifference$. Vizualizací této matice vznikne paleta, jejíž barvy odpovídají odchylkám hodnot aproximace. Pro zpracovávané matice o řádech desítek tisíc je tato vizualizace nereálná, na Obrázku 8 uvádím alespoň příklad toho, jak tato matice vypadá pro testovací sadu 13 dokumentů. V ideálním případě odchylky aproximační matice nabývají hodnot okolo nuly a vzniká matice, jejíž celá plocha je světle zelená.



Obr. 8 Vizualizace odchylek aproximace matice A

4.3.2.3 Výstup programu

I přes různé optimalizace a výpočet pouze k singulárních hodnot, kde je hodnota k zpravidla menší než řády matice A, je singulární rozklad stále poměrně časově náročnou úlohou. Je proto vhodné provést na zpracovávaných datech tento rozklad pouze jednou, výsledek výpočtu si uložit a dále pracovat už pouze s tímto výsledkem. Jednou ze součástí skriptu řešícího problémy latentní sémantické analýzy je proto také možnost uložení výsledků singulárního rozkladu do tří samostatných CSV souborů a možnost jejich následného znovupoužití. Jednotlivé soubory mají následující strukturu:

- Soubor singulárních hodnot matice S

$$[\sigma_1 \quad \dots \quad \sigma_k]$$

- Soubor hodnot pravých ortogonálních vektorů obohacený o označení dokumentů

$$\begin{bmatrix} dokument_1 & V_{1,1} & \cdots & V_{1,k} \\ \vdots & & \ddots & \vdots \\ dokument_m & V_{m,1} & \cdots & V_{m,k} \end{bmatrix}$$

- Soubor hodnot levých ortogonálních vektorů obohacený o označení termů

$$\begin{bmatrix} term_1 & U_{1,1} & \cdots & U_{1,k} \\ \vdots & & \ddots & \vdots \\ term_n & U_{n,1} & \cdots & U_{n,k} \end{bmatrix}$$

Tyto vytvořené soubory se dají považovat za LSI databázi, nad níž je možné provádět metody shlukování, klasifikace nebo vyhledávání nejpodobnějších dokumentů jakožto úlohu z oblasti IR. Hodnoty LSI databáze jsou však vztaženy pouze k dokumentům, které byly k dispozici v době výpočtu SVD. Pro přidání dokumentů do již existující LSI databáze je potřeba nový výpočet SVD, zahrnující staré i nově dostupné dokumenty. Počet dimenzí využívaných při singulárním rozkladu byl stanovován na základě experimentů a zpravidla vždy se pohyboval okolo hodnoty 100.

4.4 Information retrieval

Jednou ze zásadních operací v různých metodách text miningu je porovnávání podobností mezi jednotlivými dokumenty na základě zvolené metriky. V rámci práce jsem implementoval program umožňující v prostoru redukovaném metodou SVD vyhledávat na základě kosinové vzdálenosti dvou vektorů nejpodobnější dokumenty (případně slova – termy) na základě zadaného dotazu.

4.4.1 Vytvořený program

Pro úlohu vyhledání nejpodobnějších dokumentů či termů vůči zadanému dotazu je v programu připravena třída SimilaritySearch. Uvádím strukturu příkazu, jehož spuštěním lze nalézt definovaný počet nejpodobnějších dokumentů (případně termů) v poskytnutých datech:

```
python InformationRetrieval.py
--input=[OccurrencesMatrixFile]
--delimiter=[DelimiterString]
--svdU=[LeftOrthogonalVectorsFile]
--svdS=[SingularValuesFile]
--svdV=[RightOrthogonalVectorsFile]
--dimensions=[NumberOfDimensionsToUse]
--resultsnumber=[NumberOfResults]
--context=[DOCUMENTS | TERMS]
--stopwordsfile=[FileOfStopWords]
--stemmer=[ProgramUsedForStemming]
--query=[QueryString]
--lang=[LanguageCode]
```

Vyhledávání podobností lze pomocí programu provádět ve dvou různých kontextech. Prvním je kontext dokumentů, ve kterém lze na zadaný dotaz hledat mezi zdrojovými daty nejpodobnější dokumenty. Druhým kontextem jsou termy. Je-li pomocí parametru zvolen tento kontext, je na zadaný dotaz jako výsledek vrácen určitý počet nejpodobnějších slov definovaný pomocí tru `--resultsnumber`. Výchozím kontextem při hledání podobností jsou dokumenty a lze jej změnit pomocí parametru `--context`.

Samotnému vyhledání nejpodobnějších dokumentů případně termů předchází ještě několik nutných operací a transformací dat. Celý proces začíná načtením zdrojových dat. Tyto data představují tři soubory definované pomocí parametrů `--svdU`, `--svdS` a `--svdV` jakožto produkty singulárního rozkladu matice výskytů termů v dokumentech prováděného programem SVD. Další nedílnou částí zdrojových dat je slovník termů, které se ve zpracovávaných dokumentech vyskytovaly. Tento slovník lze načíst ze samostatného souboru, případně ze záhlaví souboru obsahujícího matici výskytů termů v dokumentech. Pro dosahování správných výsledků je nutné, aby byla data ve slovníku uspořádána ve stejném pořadí, v jaké byla zpracována při vytváření matice, jinými slovy n-tý term Term-Document matice musí být ve slovníku na n-té pozici. Pokud jsou načtena všechna zdrojová data, lze přejít k další fázi.

Vyhledávaný dotaz je zadán jako prostý řetězec a pro možnosti porovnávání s ostatními dokumenty (resp. termy) je nutné jej nejdříve převést do stejně rozměrného prostoru. Počet dimenzí prostoru, ve kterém bude vyhledávání prováděno, stejně tak jako počet dimenzí, do kterých bude pokládaný dotaz převáděn je definován pomocí parametru `--dimensions`. Dimenze jsou seřazeny sestupně od nejdůležitější až po tu nejméně důležitou, vždy je proto pro výpočty

vybráno prvních k nejdůležitějších dimenzí. Před převedením řetězce reprezentujícího zadaný dotaz do n dimenzionálního prostoru je pro zadaný dotaz nutné nejdříve na základě načteného slovníku termů vytvořit vektor výskytů termů.

Aby bylo dosahováno, co nejlepších výsledků, je potřeba před sestavením vektoru na zadaný dotaz aplikovat stejné metody předzpracování dat, jaké byli použity pro zdrojová data. K odstranění stop slov slouží metoda cleanFromStopWords využívající stejný slovník stop slov, který byl použit pro předzpracování původních dat. Není-li slovník stop slov definován, lze využít Python knihovnu *stop_words* obsahující seznam 2400 stop slov pro 11 podporovaných jazyků. Po výběru jazyku jsou z dotazu stop slova jednou z uvedených metod odstraněna. Pokud byl na zdrojová data aplikován proces stemmingu, tedy převodu slov na jejich kořenový tvar, je potřeba tento proces provést i pro vyhledávaný dotaz. K tomuto účelu slouží ve třídě SimilaritySearch metoda *stemQuery*. Pokud je při spouštění programu definován pomocí parametru `--stemmer` program řešící proces stemmingu, jehož návratovou hodnotou je transformovaný řetězec, metoda spouští tento program a jeho výsledek ukládá místo původního dotazu. Příkladem může být již dříve zmiňovaný program SnowballStemmer.pl, kterému se jako parametr předá jazyk hledaného dotazu a řetězec reprezentující samotný dotaz. Program SnowballStemmer.pl provede proces stemmingu a vrací řetězec s termy v jejich kořenovém tvaru, který je následně uložen namísto originálního vyhledávaného dotazu. Po těchto úpravách původního dotazu je následně vytvořen vektor výskytů termů q.

V této fázi je získaný vektor termů reprezentován jednorozměrným polem o délce odpovídající počtu položek nahraného slovníku termů. Následuje převod do k-rozměrného prostoru pomocí vzorce 12. Převodem je získáno jednorozměrné pole hodnot reprezentující vektor ve stejném prostoru, jako jsou zdrojová data. V této fázi lze zahájit proces výpočtu podobnosti.

$$q = q^T * U_k * {S_k}^{-1} \tag{12}$$

4.4.2 Výsledek programu

Po transformaci dotazu do požadovaného tvaru využívám pro nalezení nejpodobnějších položek ve zdrojových datech metriku kosinové podobnosti (cosine similarity) vhodnou pro porovnání dokumentů reprezentovaných vektory. Dotaz je postupně porovnáván s každou položkou zdrojových dat, pro kterou následně uchovávám hodnotu podobnosti vůči položenému dotazu. Po průchodu všemi položkami mám k dispozici slovník (asociativní pole) obsahující hodnoty podobností, po jehož seřazení jsem schopen nabídnout m nejpodobnějších položek. Klíče tohoto slovníku jsou indexy řádků v souboru předaného pomocí parametru `--svdV` pracujeme-li v kontextu dokumentů, případně tru `--svdU` pracujeme-li v kontextu termů. Na základě výběru m klíčů s největší hodnotou podobnosti vůči položenému dotazu lze pak snadno ve zdrojových datech dohledat odpovídající položky. Uvádím příklad výstupu programu vyhledávajícího 5 nejpodobnějších termů na dotaz „parking“: zdrojových datech:

```
1) TERMS 2211 (PARKING) with similarity [[ 0.99284789]]
2) TERMS 2379 (SECURE) with similarity [[ 0.76063927]]
3) TERMS 1962 (GARAGE) with similarity [[ 0.7355081]]
4) TERMS 2286 (VALET) with similarity [[ 0.7021411]]
5) TERMS 2223 (AMPLE) with similarity [[ 0.55194167]]
```

Výsledkem jsou termy, které se v recenzích vyskytovaly často spolu se slovem „parking“. Při hledání v kontextu dokumentů program na výstupu nabízí již konkrétní recenze, ve kterých se hledaný dotaz řeší. Výsledek programu zde však nelze chápat jako odpověď na položenou otázku (dotaz), ale spíše jako poskytnutí vybraných informací ve vhodné formě, které mohou k zodpovězení otázky vést. Výhodou a přidanou hodnotou vytvořeného programu je, že pro nelezení informací například o zmíněném parkování, není nutné procházet a číst všechny recenze jednu po druhé. Stačí si pouze nechat zobrazit a přečíst si ty recenze, ve kterých se toto téma řeší pomocí zobrazení nejpodobnějších dokumentů vzhledem k zadanému dotazu. Není potřeba předávat jako hodnotu dotazu konkrétní téma, ke kterému je potřeba získat informace. Účelnější je spíše zadat dotaz jako výčet slov, podobně jako v libovolném internetovém vyhledávači, nechat program dotaz transformovat a získat recenze kontextově odpovídající pokládanému dotazu. Výhodou využití LSI databáze je také fakt, že jako nejpodobnější nemusejí být označeny pouze dokumenty obsahující termy dotazu, ale také dokumenty, v nichž je řešeno stejné téma jako téma položeného dotazu.

4.5 Klasifikace

S využitím zdrojových dat, kde každá zákaznická recenze na hotel spadá do třídy P (positive, pozitivní) nebo N (negative, negativní), lze natrénovat model, schopný klasifikovat stávající i nové recenze do jedné z příslušných tříd. Jedním z možných praktických využití tohoto modelu je například různá reakce na zákazníkovu recenzi bezprostředně po jejím zadání. Hotely se snaží budovat si dobré jméno a jejich cílem je získávat stále více a více nových zákazníků a přimět zákazníky, kteří již v hotelu bydleli k jejich opětovnému návratu. Pokud zákazník po konci ubytování vyplní recenzi, kterou vytvořený model klasifikuje jako pozitivní, lze zákazníka například zdvořile vyzvat o kladné ohodnocení hotelu na sociálních sítích nebo mu poskytnout slevu na příští možný pobyt. Zmíněné metody mohou vést ke zlepšení hodnocení hotelu a udržení si spokojenosti zákazníků. Pokud je recenze naopak klasifikována jako negativní, je možné zákazníka kontaktovat a snažit se zjistit detailnější příčiny jeho nespokojenosti a využít je jako možné podklady pro zlepšení hotelových služeb. Následující podkapitoly popisují způsob dosažení výsledků použitelných například pro zmíněné rozhodnutí o reakci na zákazníkovu reakci, případně pro další jiné účely.

4.5.1 Volba implementačního jazyka

Klasifikační algoritmy popisované v kapitole 3.2 jsou již v jazycích obecně používaných pro úlohy text miningu úspěšně implementovány a odladěny. V rámci

řešení problému klasifikace tedy nebylo potřeba vyvíjet vlastní řešení, ale pouze vybrat vhodné z již dostupných. Bylo potřeba vybrat nejen jazyk, ale také správné knihovny, pomocí kterých bude možné provádět klasifikační úlohy s různými nastaveními, nad různými daty a ideálně pomocí co nejvíce algoritmů pro možnost následného srovnání. Mezi uvažované jazyky jsem si vybral jazyk Python a MATLAB.

Jazyk MATLAB původně určený pro čistě matematické účely prošel od svého vzniku řadou úprav a dnes je z něj jazyk umožňující práci s obrovskými maticemi, vykreslování 2D a 3D grafiky, modelování a simulaci procesů, analýzy nejrůznějších typů dat a jiné. Pro potřeby klasifikace dat existuje v tomto jazyce řada knihoven implementujících proces učení s učitelem s využitím konkrétního klasifikátoru. Využitelná je také například aplikace Classification Leaner[11] sloužící k trénování modelů metodou učení s učitelem za využití několika klasifikátorů. Mezi tyto klasifikátory patří například rozhodovací stromy, SVM (Support Vector Machines) klasifikátory či KNN (K-nearest neighborst). Na webu aplikace je uvedeno srovnání rychlosti učení a predikce při použití různých klasifikátorů, spolu s jejich přesností predikce a požadavky na paměť.

Tab. 1 Srovnání časové náročnosti procesu predikce pro různé typy dat u tří vybraných klasifikátorů

Algoritmus	Přesnost predikce	Rychlost učení	Rychlost predikce	Požadavky na paměť	Snadnost interpretace
Rozhodovací strom	Průměrná	Vysoká	Vysoká	Nízké	Ano
SVM	Vysoká	Průměrná	Vysoká pro nízký počet support vektorů	Nízké pro nízký počet support vektorů	Ano ale pouze u lineárního SVM
KNN	Vysoká pro malý počet dimenzí, nízká pro velký počet dimenzí	Vysoká	Vysoká pro malý počet dimenzí (<10), nízká pro větší počet dimenzí (>20)	Vysoké	Ne

Zdroj: MathWorks, 2015.

Na základě tohoto srovnání lze říci, že v případě kdy vyžadujeme klasifikovat data v omezeném čase a při určitých limitech týkajících se dostupné paměti, je vhodné využít jako klasifikátor rozhodovací strom. Pokud však nejsme limitováni pamětí ani časem a požadujeme přesnější výsledky klasifikace, je vhodné využít klasifikátory SVM či KNN. Každý z uvedených klasifikátorů má navíc řadu

[11] http://www.mathworks.com/help/stats/classificationlearner-app.html

parametrů, které je možné upravovat pro dosažení lepších výsledků při trénování modelu a následné predikci. Aplikace nabízí řadu možností vizualizace výsledků trénování, jako například zobrazení rozhodovacího stromu, ale také výsledků predikce. Na stejná data je možné aplikovat různá nastavení s různými klasifikátory a porovnávat získané výsledky. V situaci, kdy je potřeba provést trénování modelů určitým způsobem dávkově pro různá data s využitím různých klasifikátorů je tato aplikace již méně vhodná.

Za určitou alternativu aplikace Classification Leaner se dá považovat modul *scikit-learn* pro jazyk Python. Tento modul je postavený nad moduly *NumPy*, *SciPy* a *Matplotlib* pro vizualizaci výsledků. *Scikit-learn* zahrnuje metody pro klasifikaci, shlukování či regresní úlohy. Co se týče procesu klasifikace, je zde dostupná řada klasifikátorů jako například SVM, KNN, Decision Trees nebo Naive Bayes. Ke všem klasifikátorům je dostupná podrobná dokumentace a díky poměrně velké komunitě i řada příkladů. Díky modulům *NumPy* a *SciPy*, které jsou optimalizovány pro práci s rozsáhlými maticemi, jsou klasifikační úlohy poměrně paměťově i časově nenáročné. Další výhodou je možnost vizualizace výsledů, kterou lze libovolně customizovat a zobrazovat data v takové podobě v jaké je požadováno. Zásadní výhodu pro mě představovala možnost spouštět procesy klasifikování s různými nastaveními z příkazové řádky. Po napsání obslužného skriptu je možné jednoduše spouštět klasifikaci s různými algoritmy pro poskytnutá data a sledovat rozdíly ve výsledcích. Jako implementační jazyk pro proces klasifikace jsem po srovnání dostupných možností zvolil Python. Následující kapitola popisuje proces klasifikace s využitím modulu *scikit-learn* a možnosti customizace tohoto procesu pomocí implementovaného skriptu.

4.5.2 Vytvořený program

Proces klasifikace s využitím modelu *scikit-learn* se dá rozdělit prakticky do následujících pěti fází:

- Příprava zdrojových dat
- Inicializace klasifikátoru
- Natrénování klasifikátoru
- Validace klasifikátoru
- Vizualizace a interpretace výsledků

V rámci práce byl vytvořen program Classification.py, jehož součástí je třída Classificator, pomocí které se provádí pět dříve zmíněných fází klasifikace. Jednotlivé fáze budou podrobněji popsány v následujícím textu. Uvádím příkaz, jenž spouští vytvořený program, pomocí kterého je provedena klasifikace zdrojových dat s využitím jednoho z dostupných klasifikátorů (Decission Tree, KNN, SVM):

```
python Classification.py
--omsourcefile=[OccurrencesMatrixFile]
--svdVsourcefile=[RightOrthogonalVectorsFile]
--svdSsourcefile=[SingularValuesFile]
--classificator=[KNN | SVM | DECISSION_TREE]
--datatype=[SVD_REDUCED | OCCURANCES_MATRIX]
--lang=[languageCode]
--preprocessingtype=[NONE | STOP_WORDS | STEMMING]
```

4.5.2.1 Příprava zdrojových dat

Fáze přípravy zdrojových dat zahrnuje načtení dat ze zdrojového souboru, jejich případnou transformaci, převedení na požadovaný datový typ a rozdělení dat na trénovací a testovací část. Vytvořený program využívá jako zdrojové celkem tři soubory. Prvním z nich je CSV soubor obsahující normované hodnoty výskytů termů v dokumentech (*podoba souboru popsána jinde*) spolu s třídami, jenž přísluší jednotlivým dokumentům. Druhým souborem je opět CSV soubor, jehož hodnoty jsou tentokrát orthogonální vektory k-redukovaného dimenzionálního prostoru vzniklé metodou SVD. Při načítání obou těchto souborů je provedeno rozdělení dat na trénovací a testovací kolekci dokumentů. Pro natrénování modelu je ze zdrojových dat náhodně vybráno 90% dokumentů a zbylých 10% je v další fázi použito pro testování. Třídy dokumentů pro obě částí je potřeba uložit do samostatných proměnných pro jejich budoucí využití. Třetím zdrojovým souborem je taktéž soubor ve formátu CSV obsahující singulární hodnoty redukovaného prostoru využité pro reprezentaci redukovaného prostoru dokumentů. Trénovací i testovací kolekce dat jsou očištěny o přebytečné informace vyskytující se v záhlaví souborů a jejich hodnoty jsou převedeny na datový typ *numpy.ndarray*, což je požadovaný datový typ pro zdrojová data klasifikátorů modulu *scikit-learn*.

4.5.2.2 Inicializace klasifikátoru

Na základě hodnoty parametru `--classificator` je v konstruktoru třídy Classificator definován klasifikátor, který bude využit pro natrénování modelu. Lze volit mezi klasifikátory KNN, SVM a Decission Tree. Každý z klasifikátorů má několik vstupních parametrů, které jsou v programu nastaveny na hodnoty, jež poskytovaly při testování nejlepší výsledky.

U klasifikátoru SVM je možné volit z několika jádrových funkcí, jako například lineární, sigmoidální, polynomiální či RBF (Radial Basis Function). Při testování s rozsáhlými kolekcemi dat se jako nejefektivnější jevila poslední zmíněná funkce RBF. Při jejím použití je zásadní správné nastavení hodnot parametrů *C* a *gamma* u používaného klasifikátoru. Nízké hodnoty parametru *C*, činí rozhodování při klasifikaci snadnější, při vyšších hodnotách je kladen větší důraz na správné klasifikování všech subjektů správně. Parametr *gamma* definuje, jako moc velký vliv na výsledek bude mít jeden trénovací subjekt. Čím je hodnota parametru vyšší, tím více musí být ostatní subjekty blíže, aby byly ovlivněny.

Při práci s velkými kolekcemi dat je také povolit heuristiku shrinking. Jejím využitím se řešený problém zmenšuje tím, že hraniční hodnoty, jež dosáhly své horní či spodní hranice, nejsou do dalšího výpočtu zahrnovány. Užitečným je také parametr *cache_size* umožňující ukládání posledních vypočítaných hodnot ukládat do speciálního uložiště pro jejich budoucí znovupoužití bez nutnosti opětovného přepočítání hodnot.

Pro klasifikátor Desicison Tree lze definovat například minimální počet položek v listech stromu, maximální výšku vytvářeného stromu, nebo také kritérium pro další rozdělení uzlů stromu. Kritériem může být hodnota získané informace (enthropy) nebo Gini impurity – měřítko toho, jak často by byl náhodně vybraný prvek z trénovací množiny nesprávně označen, pokud by byl označován náhodně.

Klasifikátor KNN nabízí nastavení pro počet sousedů použitých při dotazech, metriku použitou pro výpočet vzdáleností prvků a také algoritmus použitý pro výpočet nejbližších sousedů. Dostupnými algoritmy jsou BallTree, KDTree a brute-force neboli přístup hrubou silou. Stromové metody jsou vhodné pro své nízké paměťové nároky a jejich využitím se zvyšuje i rychlost výpočtu nejbližších sousedů. U klasifikátoru KNN lze zvolit hodnotu parametru *algorithm* na 'auto', což způsobí, že klasifikátor si na základě poskytnutých dat při trénování vybere nejoptimálnější ze zmíněných možností výpočtu sousedů.

Před zahájením trénování klasifikátoru je vhodné zkontrolovat tréninkový set dat a pokusit se vyvážit počty trénovacích případů pro jednotlivé třídy. K tomuto účelu slouží ve třídě Classificator metoda *balanceTrainingDataSet*. Při jejím využití dochází nejprve k detekci počtu trénovacích případů pro jednotlivé třídy, následně je určena třída, ve které je obsaženo více trénovacích případů než ve třídě druhé. Poté je náhodným výběrem z tréninkové sady dat odstraněno x prvků příslušící třídě s větším počtem případů, kde hodnota x je stanovena jako rozdíl v počtu trénovacích případů jednotlivých tříd. Po tomto vyvážení obsahuje trénovací set dat stejný počet trénovacích případů pro obě klasifikační třídy (negativní a pozitivní).

Samotný klasifikátor je ve třídě Classificator ukládán do proměnné *clf*, uvádím příklad inicializačního nastavení pro všechny tři klasifikátory používané pro úlohu klasifikace:

```
    self.clf = svm.SVC (C=15, cache_size=300, gamma=0.000,
class_weight=None, kernel='rbf', max_iter=-1, probabili-
ty=False, random_state=None, shrinking=True, tol=0.001,
verbose=False)

    self.clf = tree.DecisionTreeClassifier(splitter='best',
criterion='gini', max_depth=8, min_samples_split=2,
min_samples_leaf=1, max_features=None, random_state=None,
min_density=None, max_leaf_nodes=None)
```

```
    self.clf = neighbors.KNeighborsClassifier(leaf_size=20,
n_neighbors=19, weights='distance', algorithm='auto')
```

Nastavení parametrů klasifikátorů, díky kterému bylo dosahováno nejlepších výsledků, bylo stanovováno experimentem. Vycházel jsem z výchozích hodnot při inicializaci klasifikátoru, které jsem upravoval v závislosti na výsledcích metody GridSearchCV. Této metodě je pro definované parametry klasifikátorů předáno pole hodnot, kterých by mohli parametry nabývat a metodou krosvalidace jsou následně na testovacích datech zkoušeny jejich možné kombinace. Po vyzkoušení všech kombinací jsou metodou nabídnuty hodnoty parametrů, při kterých dosáhl klasifikátor nejlepšího skóre. Klasifikátory lze tedy na datech nejdříve otestovat, hodnoty parametrů klasifikátorů podle potřeby přenastavit a poté přejít k další fázi procesu klasifikace.

4.5.2.3 Natrénování klasifikátoru

Další, již časově náročnější, fáze je fáze trénování klasifikačního modelu pomocí zvoleného klasifikátoru. Trénování se pro všechny klasifikátory modulu *scikit-learn* provádí pomocí metody *fit*, která přebírá dva parametry. Prvním parametrem je pole trénovacích dat – Term-Document matice nebo k-dimenzí redukovaného prostoru této matice, druhým parametrem je pole cílových hodnot – třídy dokumentů (P, N) trénovacích dat získané ve fázi přípravy zdrojových dat. Povoleným datovým typem položek předávaného pole cílových hodnot jsou pouze hodnoty integer, proto je potřeba nejdříve textové hodnoty 'P' a 'N' označující třídu dokumentu převést na celočíselné hodnoty a až poté použít v procesu trénování.

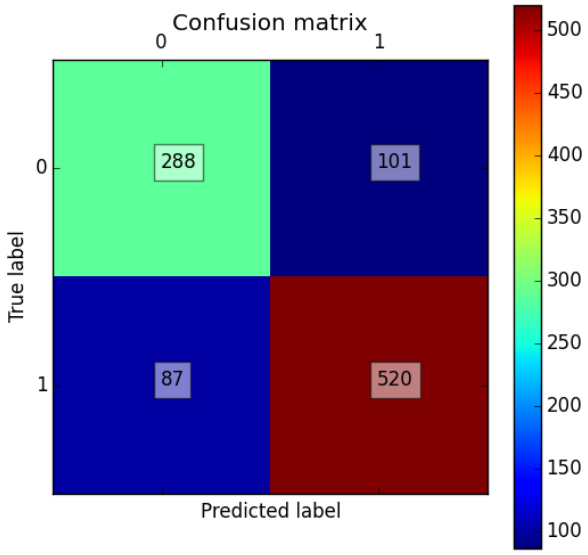
4.5.2.1 Testování klasifikátoru

Fáze testování (validace klasifikátoru) zahrnuje proces předání testovacích dat a následnou predikci hodnot pomocí natrénovaného modelu. Predikci lze v modulu *scikit-learn* provádět pomocí metody *predict* přebírající jediný parametr a to dvourozměrné pole testovacích dat. Návratovou hodnotou funkce *predict* je pole predikovaných hodnot – třídy dokumentů.

4.5.2.2 Metody hodnocení výsledků a jejich vizualizace

Poslední fází je proces srovnání testovacích dat s daty predikovanými a případná vizualizace výsledků. Úspěšnost klasifikátorů lze měřit podle dvou, respektive tří v textu již dříve zmíněných metrik. První z nich, precision, zde udává schopnost klasifikátoru neoznačovat za pozitivní položky, které jsou negativní (a naopak). Druhou metriku, recall, zde lze chápat jako schopnost nalezení všech pozitivních položek. Poslední metrika F-measure určuje hodnotu váženého průměru precision a recall. Tyto metriky jsou vypočítávány pro každou třídu zvlášť a na výstup programu, který je také ukládán do souboru, jsou zobrazeny v přehledné, zformátované formě spolu s jejich průměrnými hodnotami pro všechny klasifikované třídy.

Pro grafické zobrazení úspěšnosti klasifikace využívám tzv. Confusion matrix, jejíž příklad lze vidět na Obrázku 9. Sloupce této matice reprezentují predikované třídy, zatímco řádky prezentují třídy skutečné. V matici jsou zobrazeny počty úspěšných i neúspěšných predikcí, jejichž hodnota je zvýrazněna pomocí automaticky generované barevné škály.



Obr. 9 Confusion matrix

Hodnoty na Obrázku 9 zobrazují, že 808 testovacích dokumentů bylo pro určitá data klasifikováno správně, dále také, že 87 dokumentů s třídou P (odpovídá hodnotě True label = 1) bylo klasifikováno chybně jako třída N (odpovídá hodnotě True label = 0) a také, že 101 dokumentů s třídou N bylo chybně klasifikováno do třídy P. Z těchto hodnot lze pomocí vzorců V14 a V15 vypočítat hodnoty metrik precision a recall uvedené v ukázce části reportu níže. Je využito anglického značení pro správně zařazené výsledky – TP (true positives), nesprávně zařazené výsledky – FP (false positives) a pro relevantní dokumenty zařazené do nesprávné třídy – FN (false negative).

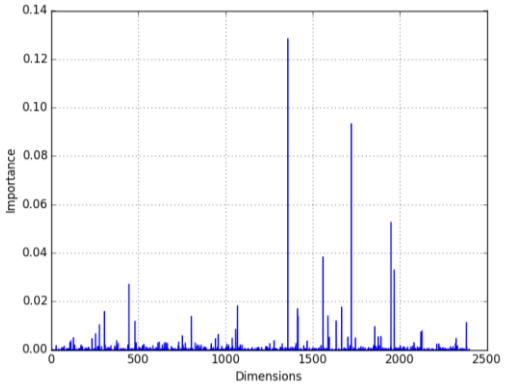
$$precision = TP/(TP + FP) \tag{13}$$

$$recall = TP/(TP + FN) \tag{14}$$

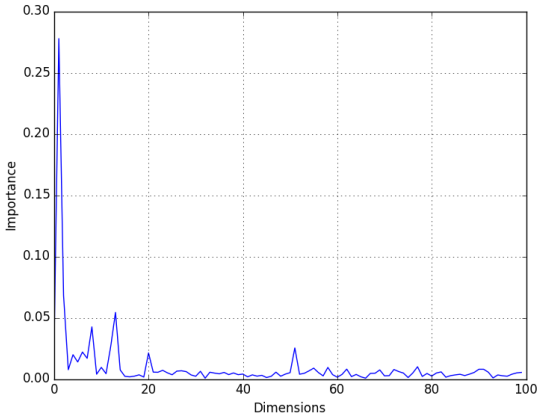
	precision	recall	f1-score	support
class N	0.74	0.77	0.75	375
class P	0.86	0.84	0.85	621
avg / total	0.81	0.81	0.81	996

Při použití klasifikátoru Decission tree lze po procesu natrénování získat navíc informace o důležitosti jednotlivých dimenzí, případně termů při použití

Term-Document matice jako zdroje dat pro klasifikaci. Informace o důležitosti prvků (angl. feature importance) jsou uloženy jako jednorozměrné pole. Čím vyšší je hodnota *n*-tého prvku výsledného pole, tím větší význam měla *n*-tá dimezne při trénování klasifikátoru a budoucí predikci tříd. Příklad vizualizace důležitosti prvků pro oba využívané typy zdrojových dat uvádím na Obrázku 10 a Obrázku 11.

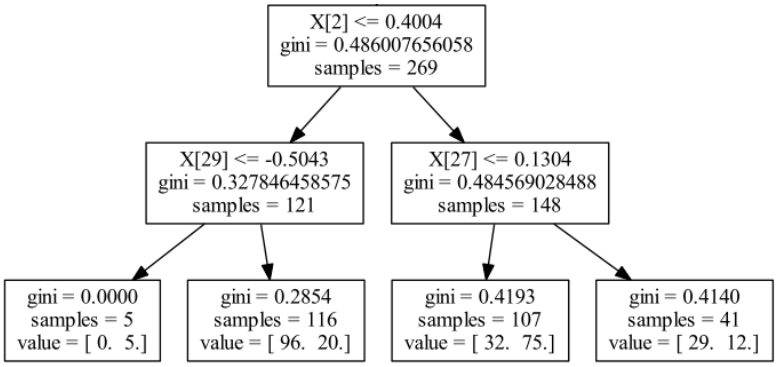


Obr. 10 Feature importances – Term-Document matrix



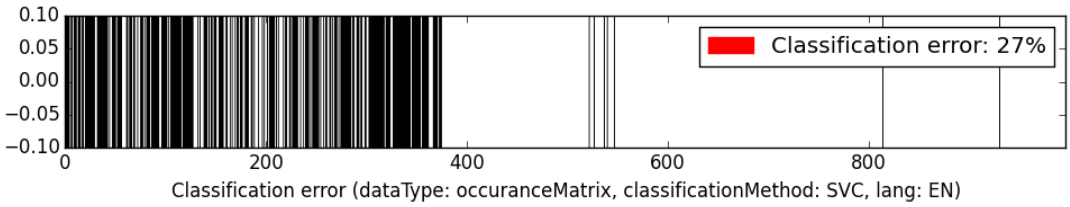
Obr. 11 Feature importances – SVD reduced data

Pro klasifikátor Decission tree je také možné získat celý rozhodovací strom, který byl při trénování vygenerován, jeho ukázku (při použití maximální výšky stromu 3) uvádím na Obrázku 12. Přehledně zobrazuje, podle jakých hodnot byly jednotlivé dokumenty do tříd klasifikovány. Pomocí balíku *tree* je následně možné získat reprezentaci stromu ve formě použitelné pro uložení do pdf souboru. Po uložení lze dále podrobněji zkoumat konkrétní uzly grafu.



Obr. 12 Decission tree example

Jednou z dalších možností vizualizace získaných výsledků je zobrazení procentuálního vyjádření chybně klasifikovaných dokumentů do grafu spolu se svým procentuálním vyjádřením. Ukázku lze vidět na Obrázku 13. Spolu s metrikami precision a recall tento graf poskytuje důležité informace o schopnosti klasifikátoru správně určovat cílové třídy. Úspěšně klasifikované dokumenty jsou v grafu zobrazovány jako bílé plochy, naopak neúspěšně klasifikované dokumenty jsou zobrazeny jako plochy černé.



Obr. 13 Classification error plot

4.6 Shlukování

Často používaným nástrojem umožňujícím shlukování data je software Cluto. Autorem tohoto softwaru je George Karypis a i přesto, že jeho poslední verze pochází z roku 2006, je Cluto stále často využívaným nástrojem, který jsem se na základě kladných referencí rozhodl pro úlohy v oblasti shlukování dat v rámci práce využít. Cluto je možné spustit na operačních systémech Microsoft Windows, Linux i OS X a jeho obsluha je prováděna pomocí příkazové řádky. Pro program existují také dvě nadstavby – gCluto a wCluto. gCluto je grafická nadstavba poskytující grafické uživatelské rozhraní avšak bez jakékoliv další funkcionality. wCluto je webové rozhraní programu Cluto hostované na jedné z universit státu Minesota, umožňující provádět shlukování dat prostřednictvím internetového prohlížeče (Karypis, 2006).

Program umožňuje úlohu shlukování parametrizovat pomocí volby hodnot parametrů jako například:

- shlukovací algoritmus (*repeated bisection, direct k-way clustering, agglomerative, graph partitionaing-based a další*)
- metrika podobnosti (cosine, euclidean distance, ext. Jaccard coeficient)
- kriteriální funkce

Vstupní data předaná programu musejí odpovídat požadovanému formátu. První řádek obsahuje informace o počtu položek zpracovávaného souboru. Další řádky obsahují dvojice sloupec–hodnota oddělené mezerou. Uvádím ukázku části souboru odpovídající požadavkům programu:

```
14839 487 199483
2 1,184 5 0,486 89 1,161
35 1,121 188 1,753 322 0,221
```

Tento zápis popisuje soubor s celkem 14839 záznamy (řádky) a 487 atributy (sloupce) obsahující celkem 199483 hodnot. Hodnota prvního záznamu ve 2. sloupci je 1,184 a 0,486 je hodnota v 5. sloupci pro první záznam. Touto úspornou formou je definován celý soubor a je velice efektivní při aplikaci shlukovacích algoritmů. Dříve popisovaný program TextMining.pl umožňuje textová data do formátu požadovaného programem Cluto převést automaticky. Možnost využití programu Cluto pro shlukování záznamů LSI databáze jsem v jazyce Python implementoval jednoduchý konvertor, převádějící data do požadovaného formátu.

Pro 4 vybrané jazyky jsem pro jejich různě předzpracované části sad recenzí provedl shlukování s využitím algoritmu repeated bisection, jakožto zástupce hierarchických metod shlukování, a metody direct k-way clustering (obdoba K-menas algoritmu). Data jsem shlukoval nejdříve do 2 a následně do 10 a 20 shluků s využitím dvou kriteriálních funkcí I1 a H2, které se v oblasti shlukování textových dat ukazují jako vhodné (Žižka, 2012).

4.6.1.1 Metodika hodnocení

Metod či ukazatelů, pomocí kterých lze hodnotit a měřit kvalitu výsledků shlukování existuje celá řada. Mezi nejznámější patří například (Dařena, 2011)

- entropie (angl. entropy)
- čistota (purity)
- přesnost (accuracy)
- F-measure

z nichž první dvě jsou po provedení procesu shlukování s využitím programu Cluto k dispozici. *Entropii* lze chápat jako množství neurčitosti a hodnocení kvality s jejím využitím je ve většině případů komplexnější, než použití čistoty, z toho důvodu, že má větší vypovídající hodnotu (Wang, 2009; Strehl, Ghosh, Mooney, 2000). Pro hodnocení shlukování je v rámci práce využita vážená en-

tropie definovaná jako vážená suma etropií konkrétních shluků. Entropie $E(S_r)$ pro jednotlivé shluky lze vypočítat na základě vzorce (Zhao, Karypis, 2001)

$$E(E_s) = -\frac{1}{\log q}\sum_{i=1}^{q}\frac{n_r^i}{n_r}\log\frac{n_r^i}{n_r} \tag{15}$$

kde S_r představuje konkrétní shluk, q udává celkový počet tříd v sadě dokumentů, n_r udává počet dokumentů v r-tém shluku, i reprezentuje i-tý dokument a n_r^i je počet dokumentů i-té třída přiřazených do r-tého shluku.

Váženou entropii E lze získat jako sumu entropií pro jednotlivé shluky podle vzorce (Zhao, Karypis, 2001)

$$E = \sum_{r=1}^{k}\frac{n_r}{n}E(S_r) \tag{16}$$

kde k označuje počet shluků a n reprezentuje počet dokumentů v celé sadě. Na základě stejného formálního značení lze získat na základě vzorců 17 a 18 také hodnoty vážené čistoty poskytované programem Cluto.

$$Pu(S_r) = \frac{1}{n_r}\max_i n_r^i \tag{17}$$

$$P = \sum_{r=1}^{k}\frac{n_r}{n}Pu(S_r) \tag{18}$$

Rozsah hodnot entropie i čistoty je dán intervalem $\langle 0, 1\rangle$ a zatím co pro entropii je ideální dosaženou hodnotou 1 značící, že shluk obsahuje dokumenty pouze z jedné třídy, pro čistotu je to obráceně. Ideální hodnota čistoty dosažená při shlukování je 1, uvádí se také, že kvalitní výsledek shlukování se vyznačuje shluky s nízkou etropií a vysokou čistotou (Deepa, Ravarthy, Student, 2012).

Pro různé kombinace typů předzpracovaných dat, využitých algoritmů a kriteriálních funkcí jsem hodnoty metrik *entropy* a *purity* měřil, porovnával a z dostupných dat vytvořil grafickou reprezentaci dat v podobě grafů.

5 Výsledky

V rámci práce bylo v dříve popisovaných oblastech text miningu provedeno celkově na stovky měření. Výsledná presentovaná data byla získána jako průměr vždy alespoň deseti měření se stejným nastavením parametrů algoritmů, na určitém typu dat zvolenou metodou[12]. V rámci práce jsou srovnávány čtyři typy dat vzniklé kombinací různých metod předzpracování (hodnoty v hranatých závorkách udávají označení, které bude pro jednotlivá data používáno v rámci následujícího textu):

- Originální data [ORIGINAL]
- Data po odstranění stop-slov [NO STOPWORDS]
- Data po procesu stemmingu [STEMMED]
- Data po odstranění stop-slov a následné aplikaci algoritmů stemmingu [NO STOPWORDS STEMMED]

Kromě vlivu různých typů předzpracování je v rámci práce srovnán také vliv využití různé reprezentace dat. Měření byla prováděna s využitím zdrojových dat v podobě Term-Document matice (označované v práci jako OCCURRENCES_MATRIX nebo OM) a také redukovaného latentního prostoru metodou SVD (v práci označováno také jako LSI databáze). Původní data byla metodou SVD redukována do 100 dimenzí, což se v průběhu experimentů ukázalo jako nejvhodnější volba. Výrazně menší počet dimenzí vedl k horším výsledkům a naopak větší počet dimenzí zpravidla nevedl k výrazně lepším výsledkům. Ze zákaznických recenzí hotelů ve 21 jazycích, které jsem měl k dispozici, jsem pro detailnější prozkoumání zvolil následující 4 jazyky:

- de – němčina
- en – angličtina
- es – španělština
- fr – francouzština

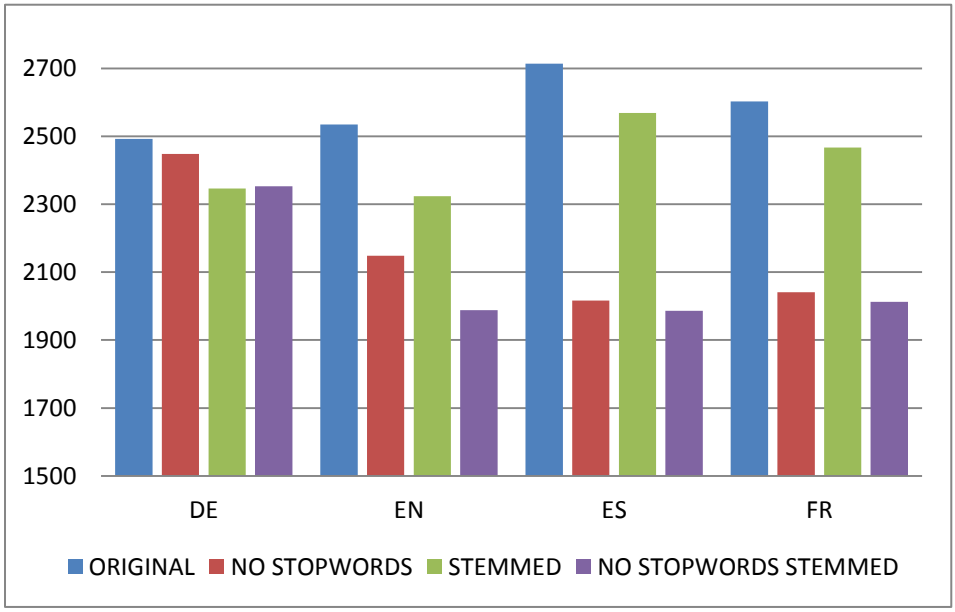
Předzpracování

Z původních dat jsem kombinací různých metod předzpracování pro vybrané jazyky získal různě rozsáhlé kolekce dokumentů. Tabulka 2 spolu s Obrázkem 14 zobrazují vliv různých metod předzpracování dat na výsledný počet unikátních termů v sadě dokumentů.

[12] Měření bylo prováděno za identických podmínek na stroji s procesorem Intel® Core™ i5-3230M CPU 2,60 GHz, se 4,00 GB RAM a operačním systémem Windows 7 Ultimate 64bit

Tab. 2 Počty unikátních termů po aplikaci různých metod předzpracování dat

	ORIGINAL	NO STOPWORDS	STEMMED	NO STOPWORDS STEMMED
DE	2492	2448	2346	2353
EN	2535	2148	2324	1988
ES	2714	2016	2569	1986
FR	2603	2041	2467	2013



Obr. 14 Počet unikátních termů při různých způsobech předzpracování dat

S výjimkou německého jazyka měla metoda odstranění stop slov na výsledný počet unikátních termů výrazný vliv. V případě španělštiny došlo aplikací této metody až téměř k 26%-ní redukci počtu unikátních termů. V anglickém jazyce bylo kombinací metody odstranění stop-slov s metodou stemmingu dosaženo redukce termů okolo 22%. Následující kapitoly popisují výsledky měření pro různé typy text miningových metod, způsoby předzpracování dat a jejich různé reprezentace pro 4 vybrané jazyky.

5.1 Information Retrieval

V oblasti Information Retrieval jsem se zaměřoval na hledání podobných dokumentů a zkoumání vlivu různého předzpracování dat na výsledky tohoto hledání. K výpočtu podobnosti dokumentů jsem využil jako metriku podobnosti kosinovu vzdálenost. Jelikož bylo při hledání podobností a hodnocení kvality výsledků nutné rozumět obsahu jednotlivých dokumentů, musel jsem se v této

oblasti omezit pouze na zpracování anglického jazyka. Ze souboru recenzí v anglickém jazyce jsem nejdříve vybral pět zástupců typicky kladných recenzí:

- Friendly staff
- Good location
- Delicious food
- Clean rooms
- Accomodation for good price

a pět zástupců záporných recenzí:

- Noise in rooms
- Bad TV channels
- Smell in rooms
- Dirty bathrooms
- Confusion of parking

S pomocí vytvořeného programu, využívajícího jako zdroj dat podle volby buď Term-Document matici nebo LSI databázi, jsem pro každou ze zmíněných recenzí hledal určitý počet nejpodobnějších dokumentů. Každý z takto nalezených nejpodobnějších dokumentů jsem následně hodnotil z těchto dvou hledisek:

- Dokument spadá do stejné třídy (pozitivní, negativní) jako zadaný dotaz
- Dokument je sémanticky podobný zadanému dotazu

Předpokládal jsem, že pro pozitivní recenzi o určitém tématu budou jako nejpodobnější ohodnoceny opět pozitivní recenze ze stejného tématu a naopak, pro negativní recenzi z určitého tématu budou jako nejpodobnější ohodnoceny opět negativní recenze týkající se stejného tématu. Tuto domněnku jsem ověřil experimenty, kde jsem pro dříve zmíněných 10 recenzí hledal nejpodobnější recenze v různě předzpracované sadě dokumentů.

Informace zda nalezený dokument spadá do stejné třídy, jako hledaná recenze je z dat jednoduše zjistitelná, jelikož všechny dokumenty jsou již předem do jedné ze tříd pozitivní či negativní zařazeny. Určení zda nalezený podobný dokument spadá také do stejného tématu, jako hledaná recenze, je již částečně subjektivní záležitostí. Proto byly do souboru zkoumaných recenzí zařazeny takové recenze, u nichž je téma poměrně jednoznačné a lze jej snadno identifikovat.

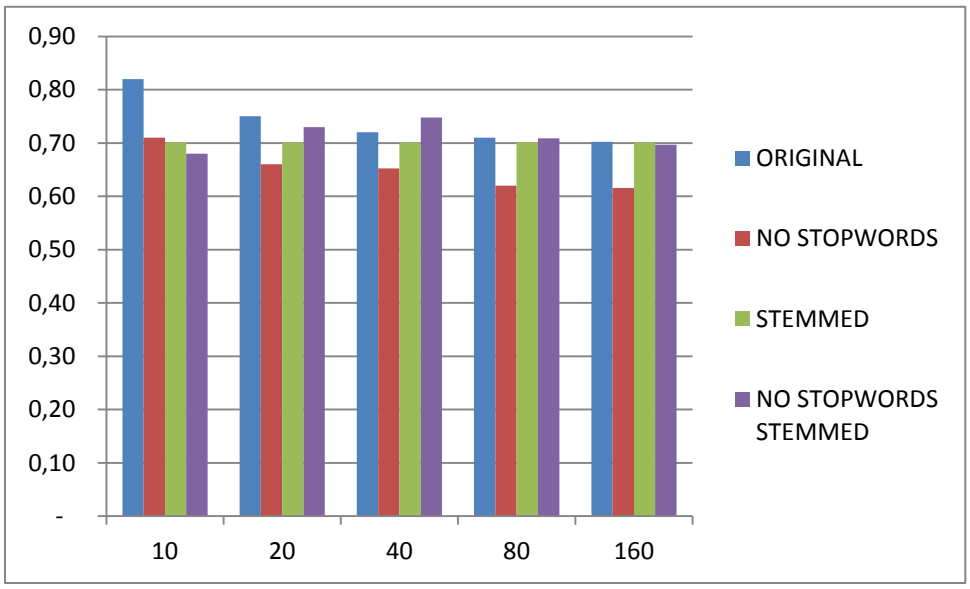
V rámci práce jsem úlohy pro zkoumání zařazení dokumentu do stejné třídy a zařazení do stejného tématu oddělil a prováděl měření zvlášť pro zdrojová data v podobě Term-Document matice a v podobě LSI databáze.

5.1.1 Vyhledání dokumentů ze stejné třídy

Pomocí experimentů a měřením jsem se snažil odhalit rozdíly ve schopnosti nalezení určitého počtu nejpodobnějších dokumentů ze stejné třídy s využitím Term-Document matice a LSI databáze pro různé typy předzpracování dat. V rámci experimentů byla pro 10 dříve zmíněných vybraných recenzí měřena hodnota precision pro prvních 10, 20, 40, 80 a 160 nejpodobnějších dokumentů, vypočítaná jako

$$precision = \frac{počet\ nalezených\ dokumentů\ ze\ stejné\ třídy}{počet\ hledaných\ dokumentů} \tag{19}$$

Pro Term-Document matice využité jako zdroj dat byly výsledky měření zprůměrovány a vyneseny do grafu na Obrázku 15.

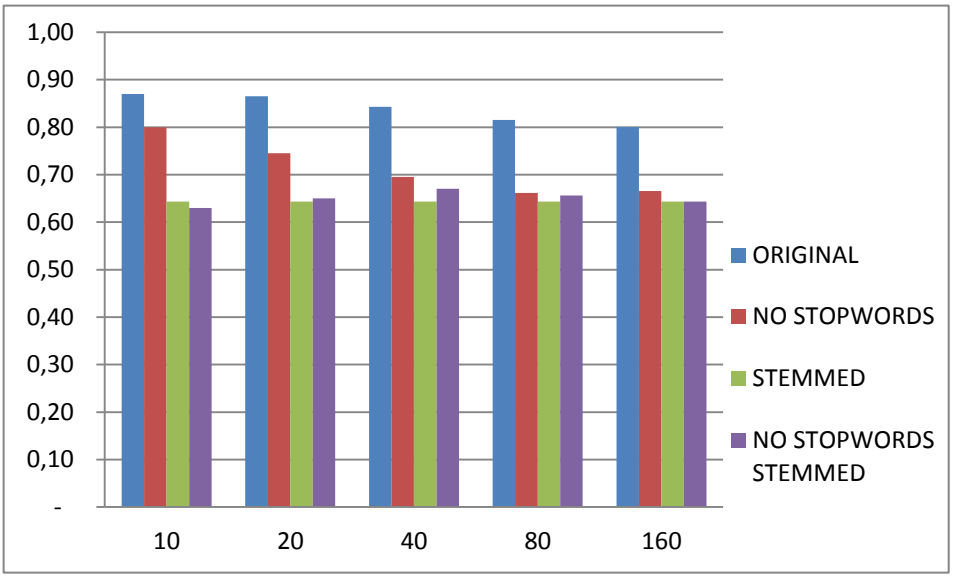


Obr. 15 Průměrné hodnoty precision pro různý počet vyhledávaných dokumentů s využitím Term-Document matice

Lze vidět, že s rostoucím počtem hledaných nejpodobnějších dokumentů, průměrná hodnota precision pro vybrané recenze postupně klesala. Kombinací metod odstranění stop-slov a stemmingu bylo pro hledání dvaceti a více nejpodobnějších dokumentů dosaženo lepších výsledků než v případě využití originálních dat. S využitím těchto metod lze pracovat v menším prostoru (díky menšímu počtu unikátních termů) a dosahovat alespoň stejných nebo dokonce i lepších výsledků v oblasti IR.

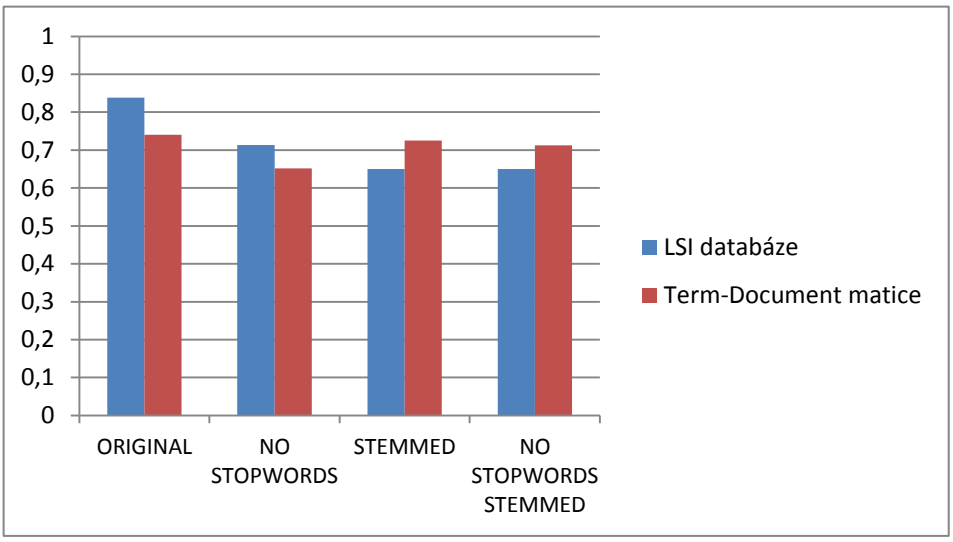
Pro LSI databázi jako zdroj dat bylo pro hledání nejpodobnějších dokumentů využito prvních 20 dimenzí. Z výsledků lze opět sledovat mírný pokles hodnoty precision s rostoucím počtem hledaných podobných dokumentů. Využití metod předzpracování mělo na hodnoty precison zde však mělo již zásadnější vliv.

Zejména metoda stemmingu vedla k výrazně horším výsledkům presentovaným na Obrázku 16.



Obr. 16 Průměrné hodnoty precision pro různý počet vyhledávaných dokumentů s využitím LSI databáze

Na Obrázku 17 jsou srovnány výsledky měření hodnot precision s využitím dvou srovnávaných metod reprezentace dat, které byly pro jednotlivé typy předzpracování získány jako průměr z měřených hodnot pro hledání 10, 20, 40, 80 a 160 nejpodobnějších dokumentů.



Obr. 17 Průměrné hodnoty precision pro srovnávané metody reprezentace dat

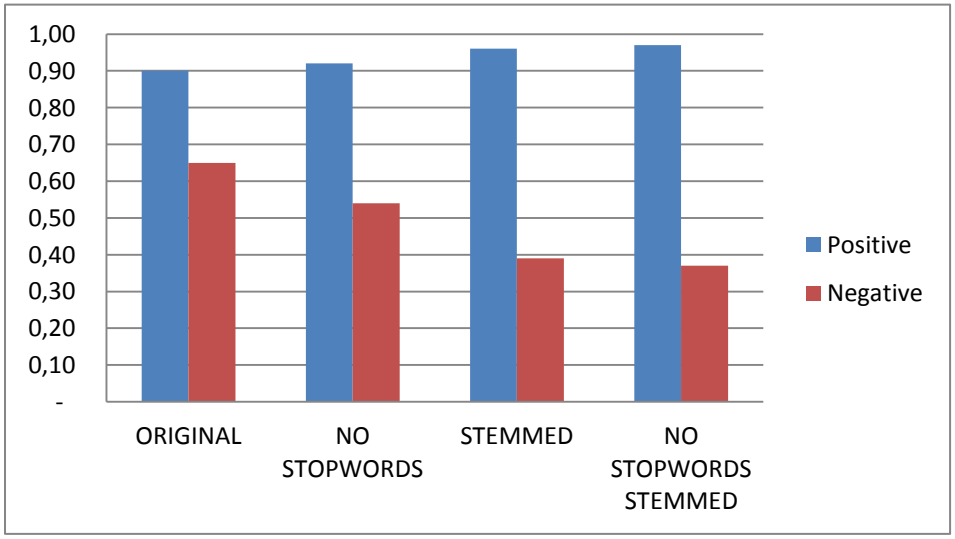
Pro originální data došlo využitím LSI databáze oproti využití Term-Document matici jako zdroje dat k poměrně výraznému zlepšení výsledků vyhledávaní. Jako nejpodobnější byly při využití tohoto modelu častěji vybírány dokumenty ze stejné třídy jako hledaná recenze a díky menšímu prostoru reprezentujícímu data bylo nalezení nejpodobnějších dokumentů navíc také rychlejší. Pro kombinaci metod stemmingu a odstranění stop-slov se naopak jako efektivnější reprezentace dat jevila Term-Document matice, avšak použitým předzpracováním došlo k určité ztrátě sémantiky a nebylo dosahováno tak dobrých výsledků v průměrných hodnotách precision jako při využití originálních dat.

5.1.2 Vyhledání tematicky podobných dokumentů

Hodnocení správného tématu nalezených dokumentů jsem hodnotil manuálně na základě vzorce

$$precision = \frac{počet\ nalezených\ dokumentů\ ze\ stejného\ tématu}{počet\ hledaných\ dokumentů} \tag{20}$$

Na základě prvních 20 nejpodobnějších dokumentů jsem po jejich přečtení stanovoval hodnotu precision. V případě využití Term-Document matice spadalo 20 nejpodobnějších dokumentů s 99% pravděpodobností vždy do stejného tématu jako hledaná recenze. Při využití LSI databáze již výsledky tak přesné nebyly. Mezi nejpodobnější byly občas zařazeny i dokumenty z tématu jiného, avšak s požadovaným tématem často určitým způsobem příbuzného. Pro konkrétní vybrané recenze měl způsob předzpracování různý vliv v souvislosti s tím, zda je recenze pozitivní nebo negativní, což znázorňuje Obrázek 18.

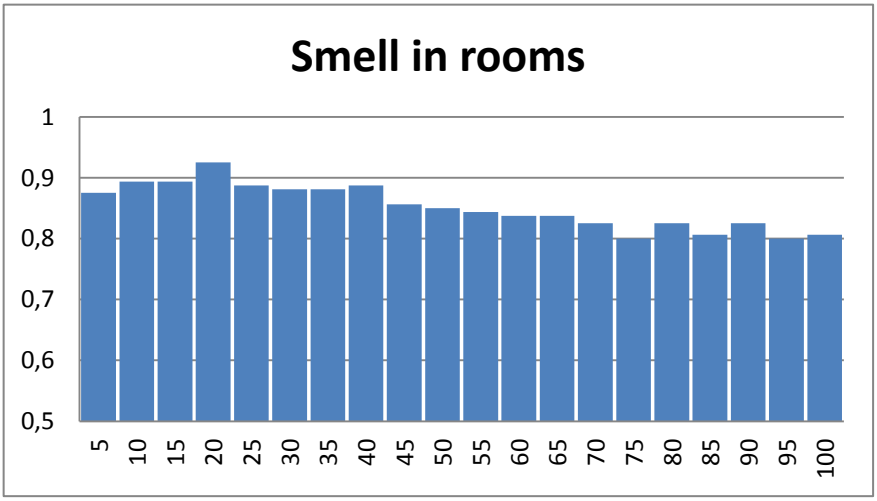


Obr. 18 Průměrné hodnoty precision pro skupiny pozitivních a negativních recenzí

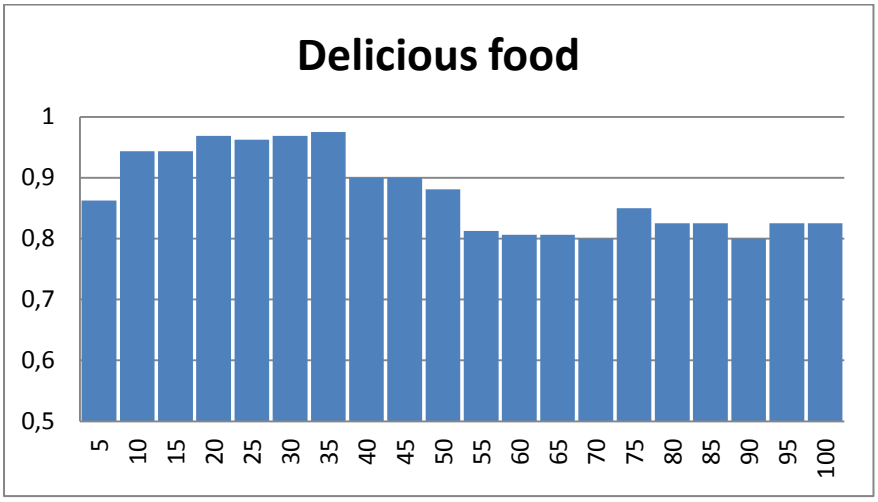
U záporných může být negativní dopad předzpracování způsoben větším počtem stop-slov vyskytujících se v těchto recenzích. Zároveň byl ve zpracovávaných datech počet negativních recenzí menší, což také mohlo ovlivnit naměřené výsledky. Ve srovnání s využitím Term-Document matice se LSI databáze pro nalezení dokumentů ze stejného tématu jevila jako méně efektivní (zejména pro negativní recenze), avšak s její pomocí byly jako podobné nalezeny také dokumenty, které přímo neobsahují slova hledané recenze, avšak dané téma se v nich řeší. Pro recenzi ‚Delicious food' byly mezi podobnými zahrnuty například recenze o nových domácích specialitách (‚NEW BREAKFAST MENU WHICH INCLUD THEIR OWN HOME-MAD JAM, WHICH WE ENJOYED') nebo o excelentních snídaních (‚THE BUFFET BREAKFAST IS EXCELLENT'). V určitém směru mohou být tyto recenze mnohem zajímavější než recenze obsahující slova DELICIOUS, FOOD nebo jejich kombinaci.

5.1.3 Volba vhodného počtu dimenzí LSI databáze

Při využití LSI databáze jako zdroje dat, jsem se kromě zkoumání tématu a tříd nejpodobnějších dokumentů zaměřil také na vliv počtu využitých dimenzí LSI databáze. Počet dimenzí může v některých případech celkem zásadně ovlivnit výsledek hledání a to jak pozitivně, tak i negativně. Z kladných a záporných recenzí, které byly zkoumány dříve, jsem vybral vždy jednoho konkrétního zástupce. Z kladných to byla recenze ‚Delicious food' a ze záporných ‚Smell in rooms'. Pro tyto recenze jsem následně měřil výsledky hledání 160 nejpodobnějších dokumentů měřené na základě hodnoty precision počítané podle vzorce 17. Pro názornější zobrazení rozdílů mezi hodnotami precision při různém počtu vybraných dimenzí LSI databáze je na grafech odrážejících výsledky měření (Obrázek 19 a 20) minimum osy *y* nastaveno na hodnotu 0,5.



Obr. 19 Hodnoty precision při různém počtu využitých dimenzí LSI databáze pro zástupce záporných recenzí



Obr. 20 Hodnoty precision při různém počtu využitých dimenzí LSI databáze pro zástupce kladných recenzí

Rozdíly v počtu využitých dimenzí z intervalu <5;100> s krokem 5 vedly pro konkrétní recenze k téměř totožným výsledkům. Příliš nízký počet využitých dimenzí (5 – 15) nebo naopak počet příliš vysoký (60 – 100) nebyl tou nejefektivnější volbou. Nejlepších výsledků pro konkrétní recenze bylo dosahováno při využití 20 – 35 dimenzí LSI databáze. Výsledky dokazují, že při využití LSI databáze je nutné se na počet využitých dimenzí zaměřit a určitým způsobem stanovit jejich vhodný počet pro konkrétní aplikaci.

5.2 Klasifikace

Při klasifikaci dokumentů jsem zkoumal vliv různého způsobu předzpracování dat na výsledky dosažené třemi vybranými klasifikátory – Decission Tree, KNN a SVM.

5.2.1.1 Natrénování klasifikátoru

Různé způsoby předzpracování dat měly na délku trénování v případě využití Term-Document matic poměrně značný vliv. V Tabulce 3 uvádím srovnání naměřených časů pro různě předzpracovaná data španělského jazyka u vybraných klasifikátorů. Ze všech tří použitých klasifikátorů byla fáze trénování vždy nejdelší pro klasifikátor SVM a to i při využití heuristiky shrinking a cachování výpočtů v průběhu trénování, což oproti původním hodnotám výrazně snížilo délku trénování. Časově nejméně náročným ve fázi trénování byl klasifikátor KNN. Z naměřených hodnot lze sledovat rozdíl mezi dobou zpracování dat reprezentovaných pomocí různých modelů. Při využití redukovaného prostoru do 100 dimenzí metodou SVD byly časy pro trénování až několikanásobně menší než při využití původních Term-Document matic. Při výpočtech nad Term-Document maticemi bylo zapotřebí zpracovat větší počet dat a kromě narůstajícího času trénování rostlo i množství využité paměti. Tabulka 3 zároveň

zobrazuje pozitivní vliv využití metod (případně kombinací metod) předzpracování dat na délku trénování klasifikátoru. Ke snížení času potřebného k natrénování dochází zejména redukcí počtu unikátních termů, čímž se zmenšuje prostor dat, která jsou klasifikátorem zpracovávána.

Tab. 3 Srovnání časové náročnosti procesu trénování klasifikátoru pro různé typy předzpracování dat španělského jazyka a jejich reprezentace u tří vybraných klasifikátorů

	OCCURRENCES MATRIX			SVD		
	Decission tree	KNN	SVM	Decission tree	KNN	SVM
ORIGINAL	12 s	4 s	270 s	4 s	1 s	19 s
NO STOPWORDS	11 s	4 s	255 s	4 s	1 s	19 s
STEMMED	9 s	3 s	260 s	4 s	1 s	19 s
NO STOPWORDS STEMMED	8 s	3 s	250 s	4 s	1 s	19 s

5.2.1.2 Validace klasifikátoru

Jak je možné vidět v Tabulce 4, proces predikce hodnot testovací sady dokumentů byl nejrychlejší s využitím klasifikátoru Decission Tree, kde výsledky byly získány prakticky okamžitě. Predikce při využití klasifikátoru KNN trvala naopak nejdéle, zejména při predikci hodnot s využitím dat z Term-Document matic. Podobně jako při procesu trénování modelu byl i zde patrný rozdíl mezi délkou zpracování různých typů dat. Z uvedených hodnot je zřejmé, že využití metody redukce dimenzí na data určená ke klasifikaci má významný pozitivní dopad na dobu potřebnou k natrénování modelu i následné predikci hodnot, kde je tento rozdíl ještě patrnější. Využitím procesu stemmingu a odstraněním stop slov lze proces predikce u určitých typů klasifikátorů opět až několikanásobně zkrátit.

Tab. 4 Srovnání časové náročnosti procesu predikce hodnot klasifikátoru pro různé typy předzpracování dat španělského jazyka a jejich reprezentace u tří vybraných klasifikátorů

	OCCURRENCES MATRIX			SVD		
	Decission tree	KNN	SVM	Decission tree	KNN	SVM
ORIGINAL	0 s	110 s	40 s	0 s	7 s	2 s
NO STOPWORDS	0 s	105 s	27 s	0 s	6 s	2 s
STEMMED	0 s	80 s	35 s	0 s	6 s	2 s
NO STOPWORDS STEMMED	0 s	75 s	32 s	0 s	6 s	2 s

5.2.1.3 Výsledky klasifikace

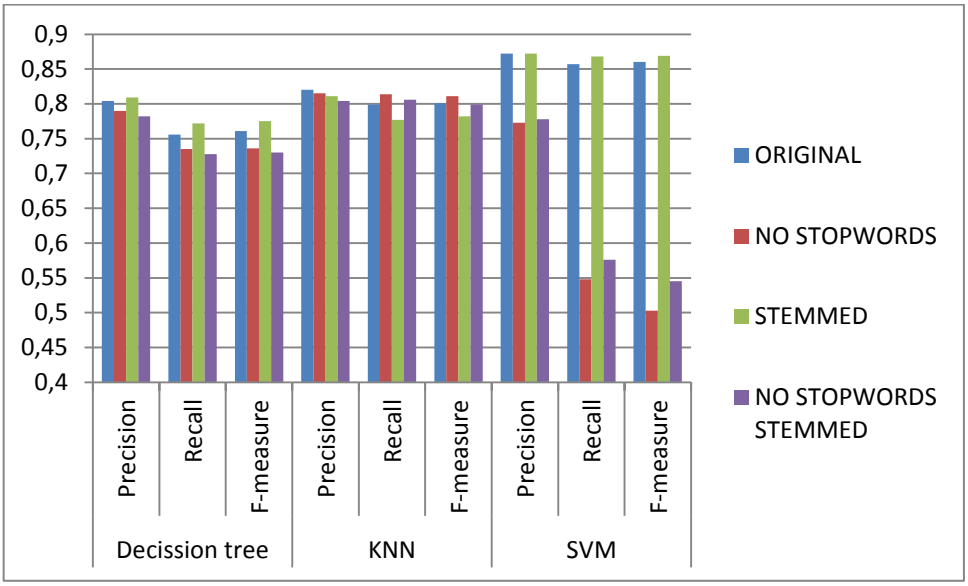
V Term-Document matici jsou dimenze vektorového prostoru reprezentovány jednotlivými termy. Při použití klasifikátoru Decission Tree lze získat seznamy termů, jež jsou pro trénování klasifikátoru nejvýznamnější. Pro vybrané zkoumané jazyky tyto nejvýznamnější termy uvádím v Tabulce 5.

Tab. 5 Seznamy nejdůležitějších termů v procesu klasifikace pro vybrané jazyky

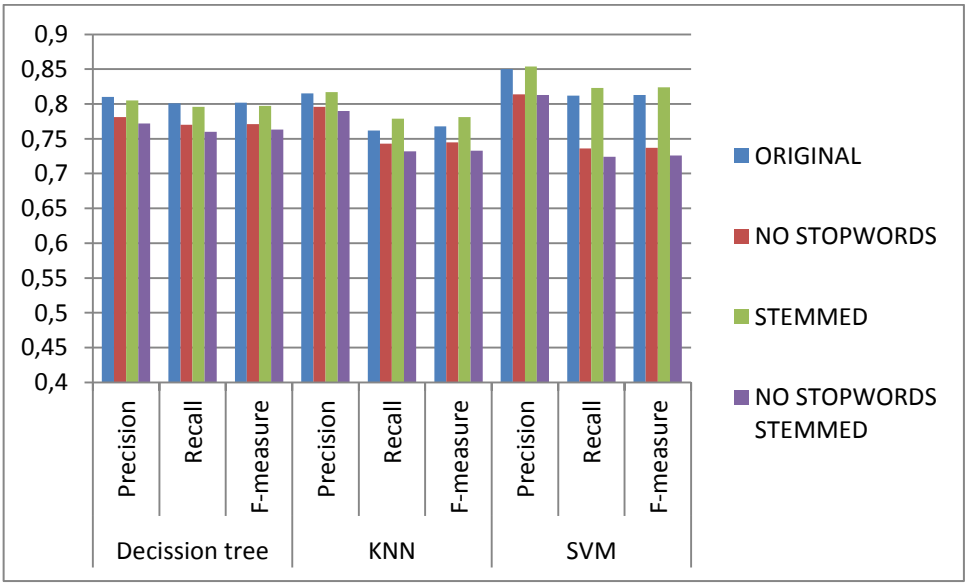
DE	EN	ES	FR
FREUNDLICH	STAFF	PERSONAL	ACCUEIL
RUHIG	LOCAT	UBICACION	PERSONNEL
ZENTRAL	LOCATION	TRANQUIL	SITUAT
PERSONAL	CLEAN	SITUACION	PROXIM
HOTEL	EXCEL	EXCELENT	CALME
KLEIN	COMFORT	RELACION	EXCELLENT
ANGENEHM	FRIEND	COMOD	CENTR
SERVICE	GREAT	TRANQUILO	ACCEUIL
MITARBEITER	CLOSE	TRANQUILIDAD	MANQU
AUSSICHT	BEAUTI	CENTR	SPACIEUX
GEPFLEGT	CONVENI	AMPLI	CONFORT
PREIS	QUIET	CONFORT	EMPLAC
WUNDERSCHON	EXCELLENT	CENTRICO	SITUE
EINGERICHTET	SPACIOUS	AMPLITUD	GRAND
BLICK	WONDER	CENTRIC	COPIEUX

Z těchto seznamů důležitých termů lze odhalit termy, které jsou označeny jako důležité ve více jazycích společně. Například termy EXCELLENT, STAFF nebo CENTR. Po hlubší analýze by na základě dostatečného množství dat mohl být sestaven multi-jazyčný klasifikátor nezávislý na jazyku zkoumaného dokumentu klasifikující dokumenty do tříd pozitivní a negativní, což by však bylo nad rámec této práce.

Výsledky klasifikace hodnocené na základě metrik precision a recall se lišili v závislosti na zpracovávaném jazyku. Na následující sérii obrázků uvádím naměřené hodnoty vynesené do grafů následované jejich stručný zhodnocením vztahujícímu se k příslušnému jazyku.

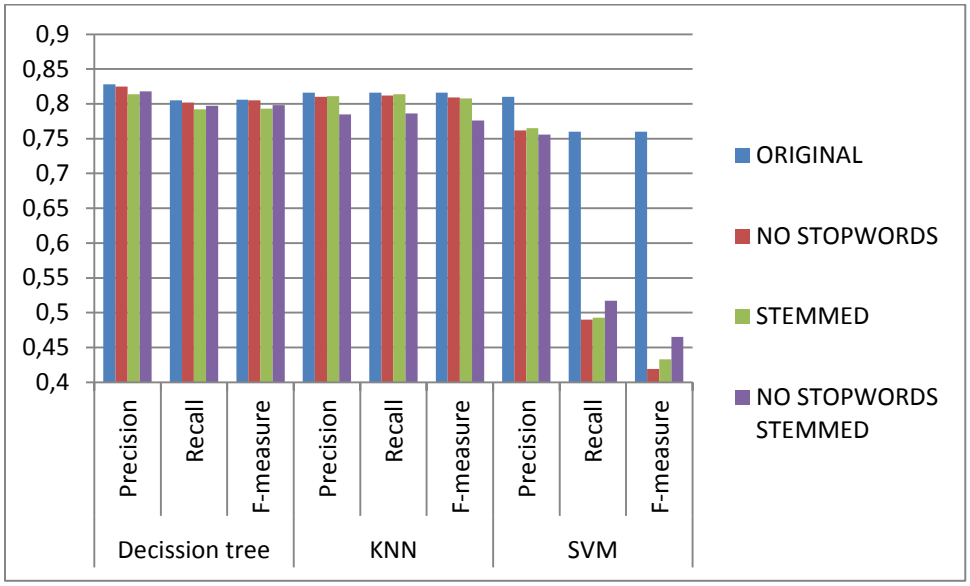


Obr. 21 Klasifikace [DE] OCCURRENCES_MATRIX - hodnoty sledovaných metrik

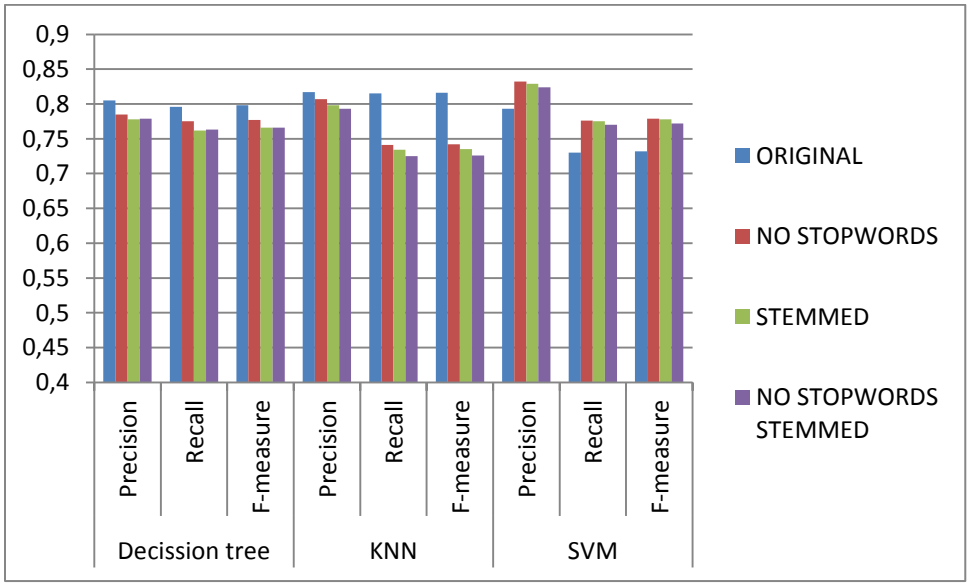


Obr. 22 Klasifikace [DE] SVD - hodnoty sledovaných metrik

V případě německého jazyka se jako nejvhodnější forma předzpracování osvědčila metoda stemmingu. Oproti originálním datům a dalším typům předzpracování bylo pro takto předzpracovaná data dosahováno téměř vždy nejlepších výsledků. Metoda odstranění stop-slov naopak vedla k horším výsledkům než při zpracování originálních dat. Kombinací těchto metod bylo dosahováno oproti originálním datům vždy pouze horších výsledků, avšak proces trénování a predikce lze pomocí těchto metod výrazně urychlit a v praktických využitích by mohl být tento ušetřený čas velice ceněn. Vliv redukce dat neměl pro tento jazyk na sledované metriky větší vliv.

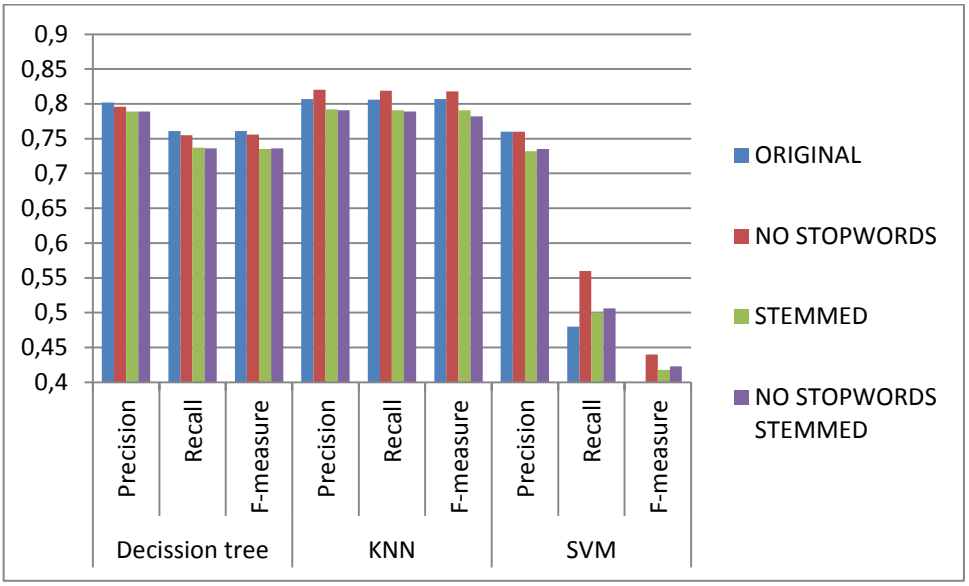


Obr. 23 Klasifikace [EN] OCCURRENCES_MATRIX - hodnoty sledovaných metrik

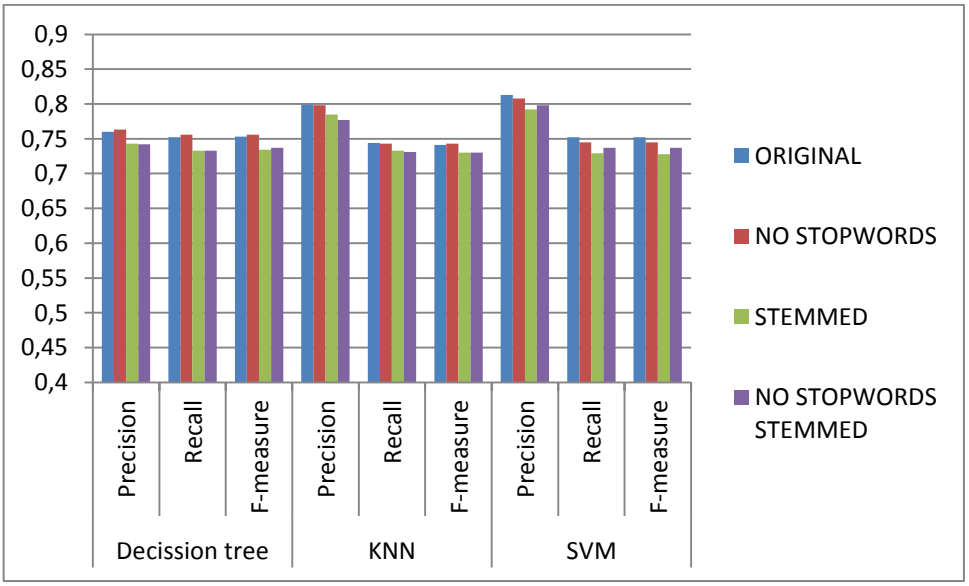


Obr. 24 Klasifikace [EN] SVD - hodnoty sledovaných metrik

V anglickém jazyce lze sledovat nepatrné rozdíly v efektivnosti klasifikování různě předzpracovaných dat, kde metody předzpracování vedly (s výjimkou klasifikátoru SVM u redukovaných dat) pouze k horším výsledkům. S využitím redukovaných dat se průměrné hodnoty sledovaných metrik podobně jako pro Term-Document matice držely okolo hodnoty 0,8 a jako nejvýhodnější se zde jevil klasifikátor SVM použitý pro redukovaná data.

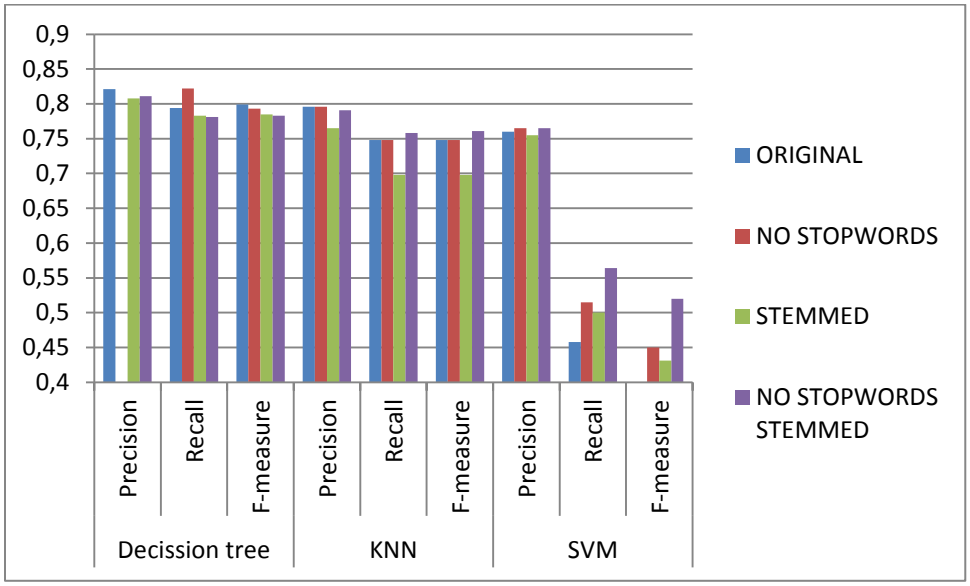


Obr. 25 Klasifikace [ES] OCCURRENCES_MATRIX - hodnoty sledovaných metrik

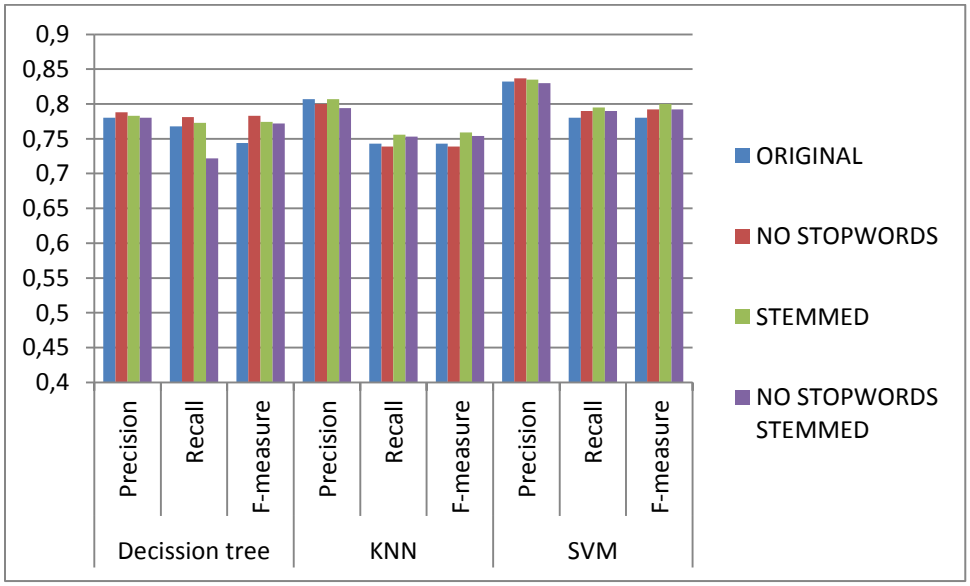


Obr. 26 Klasifikace [ES] SVD - hodnoty sledovaných metrik

Ve španělském jazyce lze pozorovat pozitivní vliv odstranění stop-slov na dosažené výsledky. I zde jsou hodnoty F-measure a recall pro klasifikátory KNN a Decision Tree v případě redukovaného prostoru o něco málo nižší než v případě klasifikace na základě Term-Document matic. Metoda stemmingu pro tento jazyk vede oproti originálním datům k výsledkům klasifikace nepatrně horším a to pro redukovaná data i data s využitím Term-Document matic.



Obr. 27 Klasifikace [FR] OCCURRENCES_MATRIX - hodnoty sledovaných metrik

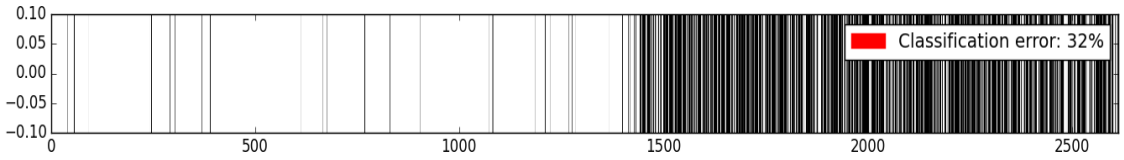


Obr. 28 Klasifikace [FR] SVD - hodnoty sledovaných metrik

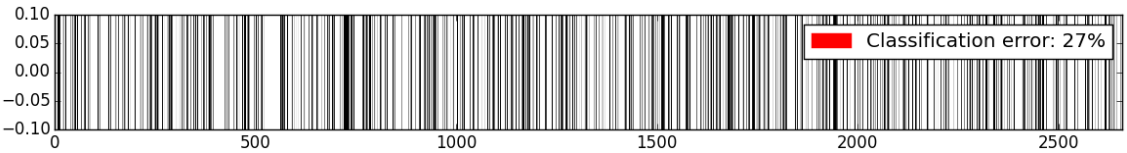
Klasifikace dat ve francouzském jazyce dosahovala lepších výsledků, pokud byla na data aplikována metoda odstranění stop-slov, případně její kombinace s metodou stemmingu. Sledované metriky byly výrazně vyšší s využitím redukovaného prostoru pouze u klasifikátoru SVM. U ostatních klasifikátorů však nenastal oproti Term-Document matici výrazný pokles v efektivnosti klasifikace a redukovaný prostor zde lze stále hodnotit jako efektivní a vhodná forma reprezentace dat.

Při počátečních měřeních výkonosti klasifikátorů jsem v případě využití SVM a zpracování dat v podobě Term-Document matice často získával výsledky,

kde jedna ze tříd byla rozpoznávána s téměř stoprocentní úspěšnosti a dokumenty třídy druhé nebyl klasifikátor schopen správně určit zpravidla nikdy, což znázorňuje Obrázek 29. Důvodem byla nevyváženost trénovacích sad dokumentů, kde jedna ze tříd měla v počtu dokumentů vždy výraznou převahu. Před zahájením procesu trénování klasifikátoru byly proto sady dokumentů jednotlivých tříd nejdříve vybalancovány tak, aby obsahovaly stejný počet dokumentů. Tato metoda byla použita pro všechny dříve demonstrované měření efektivnosti klasifikátoru a obecně vedla k lepším výsledkům klasifikace pro všechny tři typy vybraných klasifikátorů. Porovnáním Obrázku 29 s Obrázkem 30 lze demonstrovat rozdíl vlivu vybalancování sad dokumentů, kterým došlo jednak ke snížení chybovosti klasifikace a zároveň byly lépe klasifikovány dokumenty nacházející se v druhé polovině datového souboru patřící do stejné třídy (N, negativní).



Obr. 29 Classification error nevyvážených trénovacích dat



Obr. 30 Classification error po vybalancování trénovacích dat

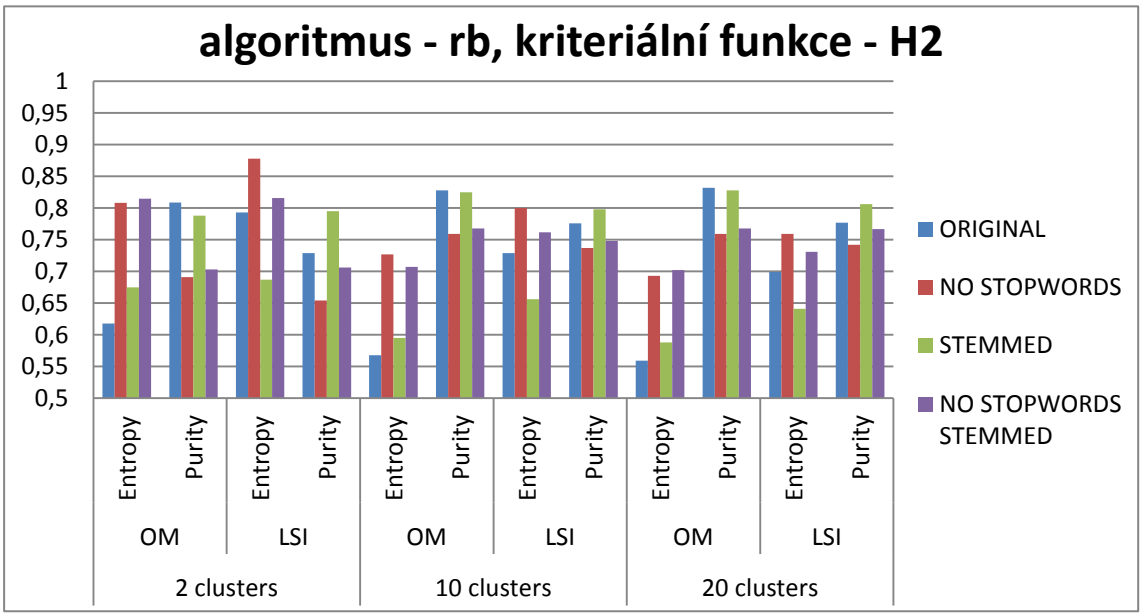
5.3 Shlukování

Na základě hodnot metrik *entropy* a *purity*, poskytnutých programem Cluto, jsem hodnotil shlukování různě předzpracovaných dat. Nejlepších výsledků bylo při procesu shlukování dosahováno s využitím shlukovacího algoritmu Repeated bisection (rb). Pro shlukování do 2, 10 i 20 shluků bylo s využitím tohoto algoritmu dosahováno průměrně nejnižších hodnot *entropy* (DE – 0,706; EN – 0,668; ES – 0,787; FR – 0,711) a naopak nejvyšších hodnot pro *purity* (DE – 0,768; EN – 0,784; ES – 0,728; FR – 0,775). Kriteriální funkce vedoucí nejčastěji k nejlepším hodnotám metrik byla funkce H2.

Srovnání vlivu způsobu předzpracování dat vedlo pro různé jazyky k výsledkům poměrně odlišným, a proto budou tyto výsledky pro jednotlivé jazyky rozebrány samostatně. Současně s vlivem předzpracování dat byly srovnávány také výsledky s využitím LSI databáze (data byla redukována do 100 dimenzí) oproti využití Term-Document matici a pro jednotlivé jazyky budou tyto výsledky také stručně shodnoceny.

Pro německý jazyk se při shlukování do většího počtu shluků (více než 2) s využitím kriteriální funkce H2 jevila jako nejefektivnější metoda předzpraco-

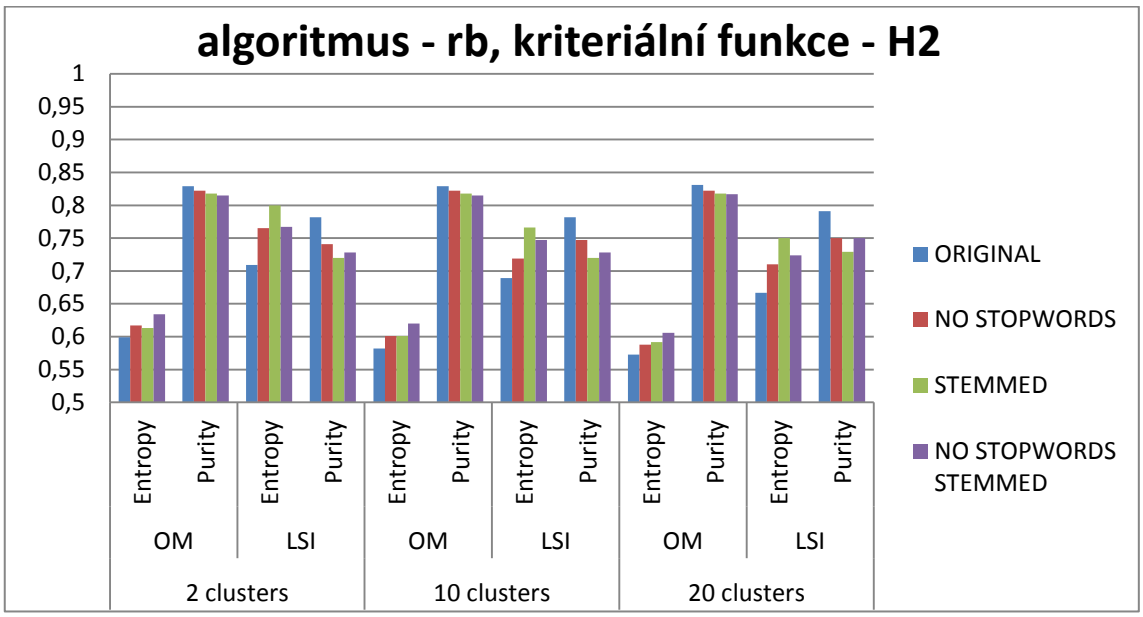
vání dat metoda stemmingu. Jeho aplikací bylo dosaženo rozdělení dokumentů do shluků s menší hodnotou průměrné entropie a občas také vyšší hodnoty čistoty shluků. Metoda stemmingu měla pozitivní dopad při shlukování s využitím Term-Document matice ale také při využití LSI databáze. Naměřené výsledky shlukování pro německý jazyk jsou zobrazeny na Obrázku 31. Metoda odstranění stop-slov a její kombinace s metodou stemingu vedla pouze k výsledkům horším a to jak při využití LSI databáze tak pro Term-Document matici.



Obr. 31 Hodnoty sledovaných metrik shlukování německého jazyka

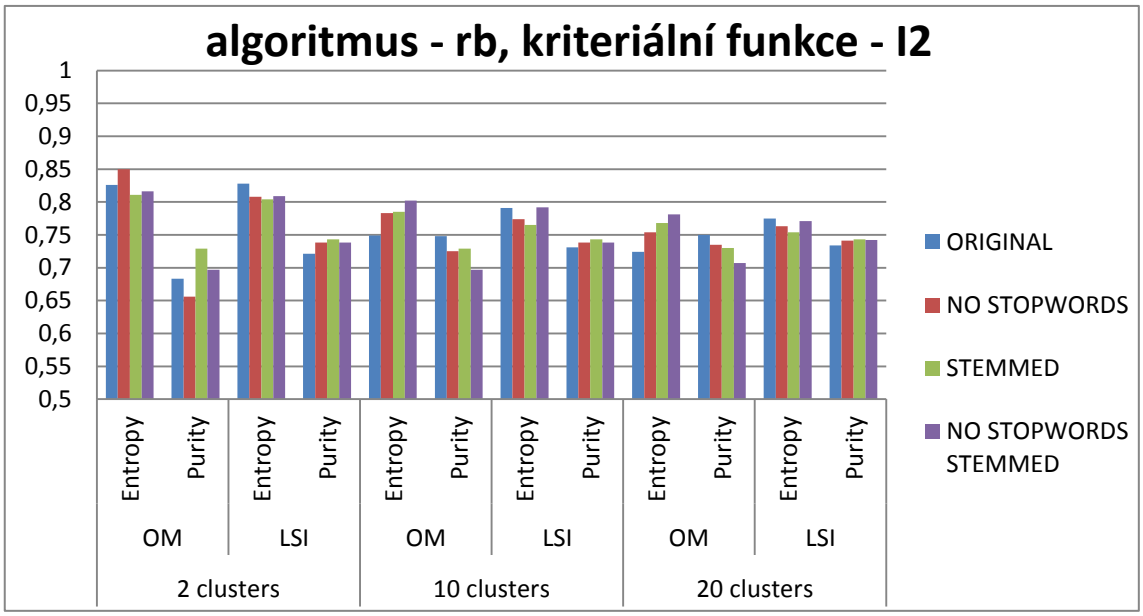
Při zprůměrování hodnot se využití LSI databáze v německém jazyce oproti Term-Document matici jevilo jako lepší pouze při shlukování do 2 shluků a pro větší počet shluků vykazovalo pouze horší hodnoty sledovaných metrik.

Podobně tomu s efektivitou využití LSI databáze bylo i v anglickém jazyce. Zde její využití vedlo vždy pouze k horším výsledkům, což bylo společné i pro použité metody předzpracování. Aplikace jakékoliv z porovnávaných metod předzpracování vedla oproti shlukování originálních dat pouze ke zhoršením hodnot sledovaných metrik, jejichž pokles je znázorněn na Obrázku 32.



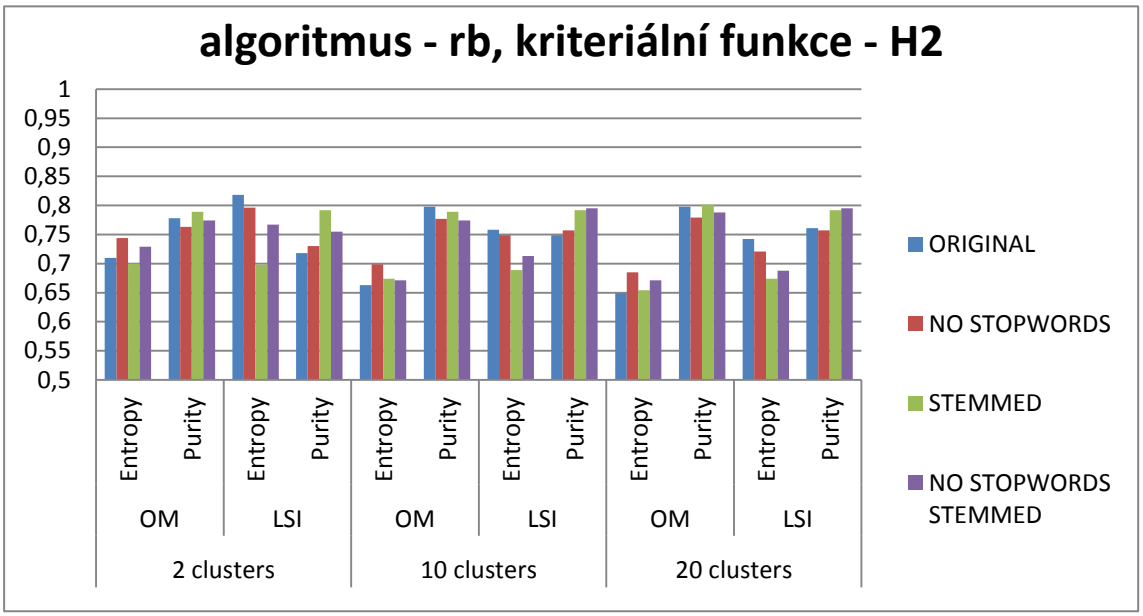
Obr. 32 Hodnoty sledovaných metrik shlukování anglického jazyka

U španělského jazyka bylo možné sledovat pozitivní vliv aplikace metody stemmingu na výsledky shlukování a to zejména při shlukování do 2 shluků s využitím LSI databáze. Pro větší počet shluků bylo využití LSI databáze přínosné zejména s ohledem na hodnoty metriky *purity*, jejíž hodnoty byly vyšší než při využití Term-Document matice, což znázorňuje Obrázek 33.



Obr. 33 Hodnoty sledovaných metrik shlukování španělského jazyka

Ve francouzském jazyce měla na výsledky shlukování opět významný vliv metoda stemmingu. Jejím využitím bylo oproti originálním datům dosahováno vyšší čistoty shluků a také menších hodnot metriky entropie. Podobně jako pro španělský jazyk se i zde využití LSI databáze osvědčilo především při shlukování do menšího počtu shluků. Při shlukování do 10 a 20 shluků se jako efektivnější jevilo využití Term-Document matice. Pro francouzský jazyk jsou hodnoty sledovaných metrik vyneseny v grafu na Obrázku 34.



Obr. 34 Hodnoty sledovaných metrik shlukování francouzského jazyka

6 Diskuse

Vzhledem k tomu, že měření byla prováděna pouze na části dostupných dat, nelze jejich výsledky příliš zobecňovat. Nicméně byly díky provedeným měřením odhaleny určité pravidelnosti a vazby na metody předzpracování a také modely reprezentace dat, které lze brát jako podklad pro další analýzy dat.

Při hodnocení schopnosti nalezení určitého počtu podobných dokumentů ze stejné třídy vedlo hledání většího počtu dokumentů ke snížení hodnoty precision. Nejvýznamnější vliv na hodnotu precision a její snížení měla metoda stemmingu následovaná metodou odstranění stop-slov. Při využití LSI databáze bylo s originálními daty bez aplikace metod stemmingu a odstranění stop-slov dosahováno nejlepších výsledků, z čehož lze odvozovat, že je tento model v oblasti IR jistě vhodný a díky práci s menším prostorem reprezentujícím data jsou následné výpočty i rychlejší. V případě hledání nejpodobnějších dokumentů ze stejného tématu již využití LSI databáze tak efektivní oproti Term-Document matici nebylo, avšak alespoň v případě pozitivních recenzí vedlo k nalezení dokumentů velice často tematicky podobných a navíc takových, jež přesně neobsahují slova hledané recenze, což by v reálném využití mohlo být přínosné.

Pro využití LSI databáze jako zdroje dat se v oblasti IR pro konkrétní případy anglického jazyka jevilo jako nejefektivnější využití poměrně nízkého počtu dimenzí (20 – 30). To je oproti řádově až tisícům dimenzí využívaných při využití Term-Document matic výrazné snížení prohledávaného prostoru, vedoucí k možnému urychlení a úspoře paměti bez výrazné ztráty v přesnosti vyhledávání. Metody předzpracování zdrojových dat a jejich kombinace vedly při hledání nejpodobnějších dokumentů zpravidla pouze k horším výsledkům, což si lze vysvětlit nutností odstranění nebo transformace termů při tomto předzpracování vedoucí ke ztrátě určité sémantiky v textu

V části týkající se klasifikace textových dat jsem na základě měření odhalil pro vybrané jazyky vhodné metody předzpracování dat, vedoucí k výsledkům lepším nežli v případě využití originálních dat. Pro různě předzpracované textové sady každého ze 4 zvolených jazyků bylo provedeno vždy minimálně 10 měření, ze kterých jsou představené výsledky získány jako průměr. Pro německý jazyk se jako ideální metoda předzpracování jevila metoda stemmingu, pro španělštinu a francouztinu bylo nejvýhodnější odstranění stop-slov. Měření také odhalilo přínos využití latentní sémantické analýzy, kde v případě využití dat redukovaných pomocí metody SVD byla doba učení klasifikátorů až jedenkrát menší, snížil se také čas predikce hodnot a metriky precision, recall a F-measure, na základě kterých jsem hodnotil efektivitu klasifikace, dosahovali vyšších hodnot nebo alespoň stejných hodnot jako v případě zpracování Term-Document matic. Data byla oproti původním redukována do pouhých 100 dimenzí, což nevedlo k výrazným ztrátám informací, ale naopak byly metodou SVD mezi daty odhaleny určité vazby, vedoucí k dosažení lepších výsledků zejména s využitím klasifikátoru SVM. Vytvořené programy v jazyce Python mohou být navíc bez větších úprav integrovány do již existujících nástrojů vyu-

žívaných například na ÚI Mendelovy university, případně mohou být rozšířeny o další funkcionalitu.

V oblasti shlukování jsem na základě metrik entropy a purity měřil efektivitu shlukování. Pozitivní vliv na výsledky shlukování části dostupné sady dat německého, francouzského a španělského jazyka měla zejména metoda stemmingu. Pro anglický jazyk bylo využití metod předzpracování pro účely shlukování dat spíše neefektivní. Využití LSI databáze vedlo při shlukování do menšího počtu shluků oproti využití Term-Document maticím k lepším výsledkům a bylo by zajímavé sledovat výsledky zpracování ještě větších sad dokumentů, kde by mohly být rozdíly mezi jednotlivými metodami ještě výraznější.

Presentované výsledky všech popisovaných text miningových metod vychází z naměřených hodnot přehledně zpracovaných do samostatných reportů v podobě textových souborů případně tabulek, jež jsou spolu s implementovanými programy součástí přiloženého CD.

6.1 Možná vylepšení

Určitým nedostatkem práce je využití poměrně malého množství dostupných dokumentů, na základě jejichž zpracování jsou získávány a následně prezentovány dosažené výsledky. Zpracované dokumenty pocházejí navíc pouze z jedné sledované domény, a proto nemohou být získané výsledky příliš zobecňovány. Jedním z možných vylepšení stávajícího řešení by mohlo být využití pokročilejších metod latentní analýzy, jako jsou pravděpodobnostní latentní sémantická analýza případně Dirichletova alokace, překonávající určité nedostatky použitého LSA modelu. V oblasti předzpracování dat by bylo jistě také zajímavé sledovat vliv aplikace nahrazení termů jejich synonymy, například s využitím slovníku synonym projektu Tartarus[13].

[13] http://www.powerthesaurus.org/

7 Závěr

Práce byla zaměřena na dolování znalostí z textových dat s cílem srovnat vliv využití různého způsobu předzpracování dat a možný přínos využití latentní analýzy. Popisem text miningových metod se zabývá kapitola 3, ve které jsou také popsány srovnávané metody předzpracování dat jako stemming, odstranění stop-slov a další.

Pro oblast klasifikace a Information retrieval byly představeny dílčí vytvořené programy sloužící k dolování znalostí, využitelné v dalším výzkumu sledované oblasti. Při klasifikaci zdrojových dat představujících recenze hotelových zákazníků byly využity algoritmy KNN, SVM a DecissionTree, na základě jejichž výsledků byl následně hodnocen vliv různého předzpracování textových dat (podkapitola 5.2). V oblasti Information retrieval byl srovnán vliv předpracování a zejména použití latentní analýzy na schopnost identifikovat pro vyhledávaný dokument podobné dokumenty ze stejného tématu a třídy. S využitím latentní analýzy byl také pro vybrané dokumenty anglického jazyka srovnán vliv počtu využitých dimenzí na výsledky hledání.

Měření úspěšnosti shlukování bylo prováděno na základě metrik entropy a purity poskytované softwarem Cluto. Pro čtyři vybrané jazyky, jejichž textové sady byly zpracovávány, bylo provedeno na stovky měření. Výsledky těchto experimentů jsou představeny v podkapitole 5.3 spolu s jejich grafickou interpretací.

Vliv využití různých metod předzpracování dat na zkoumané metody text miningu je diskutován v kapitole 6 spolu s přínosem využití latentní analýzy. Výsledky se zpravidla lišili v závislosti na jazyku zpracovávaných dat, avšak byly odhaleny i určité společné rysy použitelné jako podklad pro další projekty v oblasti text miningu. Práce se ve svých jednotlivých částech zabývala všemi stanovenými cíli, k jejichž dosažení bylo provedeno stovky experimentů a měření. Jejich presentované výsledky implikují dosažení původně stanovených cílů.

8 Literatura

BERRY, W. M, DRMAČ, Z., JESSUP, J. R. 1999. *Matrices, Vector Spaces, and Information Retrieval*. SIAM Review. 1999, 41, s. 336–362.

BOTTOU, L., LIN CH-L. *Support Vector Machine Solvers*. s. 27. [online] 2006. Dostupné z: http://leon.bottou.org/publications/pdf/lin-2006.pdf

CRISTIANINI, N., Shawe-Taylor, J. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Vyd. 1. New York: Cambridge Press, 2000, 189 s. ISBN 05-217-8019-5.

DAŘENA, F. *Vybrané přístupy k dolování znalostí ze sociálních médií*. Habilitační práce. Brno: MENDELU, 2011.

DEEPA, M., REVATHY, P., STUDENT, P. G. *Validation of Document Clustering-based on Purity and Entropy measures*. International Journal of Advanced Re-search in Computer and Communication Engineering. 2012, s. 147–152. ISSN2278-1021.

HAN, J., KAMBER, M. *Data mining concepts and techniques*. 1st ed. San Francisco: Morgan Kaufmann, 2000, 549 s. ISBN 15-586-0489-8.

HEBÁK, P., HUSTOPECKÝ, J. MALÁ, I. *Vícerozměrné statistické metody*. Vyd 1. Praha: Informatorium, 2005, 239 s. ISBN 80-733-3036-9.

HOWLAND, P., PARK, H. *Generalizing discriminant analysis using the generalized singular value decomposition, IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol.26, no. 8, pp. 995-1006, August 2004, doi:10.1109/TPAMI.2004.46.

KARYPIS, G. *Cluto – A Clustering Toolkit*. Minnesota. [online] 2006. Dostupné z: http://glaros.dtc.umn.edu/gkhome/fetch/sw/cluto/manual.pdf.

KONCHADY, M. *Text mining application programming*. Boston: Charles River Media, 2006, xix, 412 s. ISBN 978-1-58450-460-3.

KRÁTKÝ, M. *Využití SVD pro indexování latentní sémantiky*. Department of Computer Science, VŠB-Technical University of Ostrava, Czech Republic [online]. 2002. Dostupné z:http://www.cs.vsb.cz/kratky/publications/lsi-svd_ma.pdf

LANDAUER, T. K., MCNAMARA, D. S., DENNIS, S., KINTSCH, W. *Handbook of Latent Semantic Analysis*. New-York: Routledge, 2014. ISBN 978-113-8004-191.

LIN, M.-H., 2000. *Out-of-Core singular value decomposition*. Technical Report TR-83. New-York. Sate University of New York Stony Brook, Experimental Computer Systems Laboratory.

MATERNA, J. *Sémantická analýza textů: Latentní Dirichletova Alokace*. [online]. 2011. Dostupné z: http://fulltext.sblog.cz/2011/12/22/semanticka-analyza-textu-6/

MATERNA, J. *Sémantická analýza textů: Probabilistic Latent Semantic Analysis*. [online]. 2011. Dostupné z: http://fulltext.sblog.cz/2011/12/04/semanticka-analyza-textu-5/

NETO, J. L., SANTOS, A. D., KAESTNER, C. A. A., FREITAS, A. A. *Document clustering and text summarization*, In Proceedings of the 4th International Conference Practical Applications of Knowledge Discovery and Data Mining, London, 2000.

NOVÁK, Z., DAŘENA, F. *Aplikace pro přípravu textových dat*. In PEFnet 2012. ISBN 978-80-7375-669-7.

O'BRIEN, G. W., 1994. *Information Management Tools for Updating an SVD-Encoded Indexing Scheme*. Knoxville, 80 s. Disertační práce. The University of Tenessee.

QUINLAN, J. R. *Discovering rules by induction from large collections of examples: Expert systems in the micro electronic age*. Edinburgh: University Press, 1979.

ŘEZNÍČEK, P., DAŘENA, F. *Předzpracování nestrukturovaných dat pomocí jazyka Snowball*. In PEFnet 2013. 1. vyd. Brno: MENDELU Publishing centre, 2013. ISBN 978-80-7375-906-3

SALTON, G., MCGILL, M. J. *Introduction to modern information retrieval*. New York: McGraw-Hill, c1983, xv, 448 p. ISBN 00-705-4484-0.

SILVA, C., RIBEIRO, B. *Inductive inference for large scale text classification: kernel approaches and techniques*. Berlin: Springer, 2010, xx, 155 p. Studies in computational intelligence, v. 255. ISBN 3642045324.

SRIVASTAVA, A., SAHAMI, M. *Text mining: classification, clustering, and applications*. Boca Raton, FL: CRC Press, 2009, xxx, 290 p. ISBN 978-142-0059-403.

STEINWART, I., CRISTMANN, A. *Support vector machines*. 1st ed. New York: Springer, c2008, xvi, 601 p. ISBN 9780387772424.

STREHL, A., GHOSH, J., MOONEY, R., 2000. *Impact of Similarity Measures on Web-page Clustering*. The University of Texas at Austin.

ŠEVČÍK, R., 2010. *Klasifikace elektronických dokumentů s využitím shlukové analýzy*. Praha, 78 s. Diplomová práce. Vysoká škola ekonomická v Praze.

TARTARUS. *Snowball – Quick introduction* [online]. 2012. Dostupné z: http://snowball.tartarus.org/texts/quickintro.html

WANG, Y. ZHANG, J., VESSILEVA, J. *Artificial Intelligence: methodology, systems, and applications: Towards Effective Recommendation of Social Data across Social Networking Sites*. Editor Stefano A Cerri, Danail Dochev. Berlin: Springer, 2010, xii, 366 s. Lecture notes in computer science, 6304. ISBN 35-404-1044-9.

WEISS, S. M., INDURKHYA, N., ZHANG, T., DAMERAU, F. *Text Mining: Predictive Methods for Analyzing Unstructured Information*. Berlin: Springer Science & Business Media, 2010. ISBN 0387345558.

WANG Y., 2009. *Novel approaches in cognitive informatics and natural intelligence*. Hershey: IGI Global. ISBN 978-1-60566-171-1.

WU, J. A *Summary of Support Vector Machine*. [online] 2007. Dostupné z: http://dsec.pku.edu.cn/~jinlong/pksvm/An_Summary_of_svm.pdf

ZHANG, X., ZHU, X. *Pattern recognition and image analysis: Extended Bi-gram Features in Text Categorization*. New York: Springer, 2005. ISBN 978-3-540-26154-4.

ZHAO, Y., KARYPIS, G., 2001. *Criterion Functions for Document Clustering: Experiments and analysis*. Technical report. Department of Computer Science University of Minnesota.

ZHU, M., 2004. *Recall, precision and average precision*. Working Paper 2004-09. University of Waterloo - Department of Statistics & Actuarial Science, 2004.

ŽIŽKA, J., BURDA, K., DAŘENA, F. *Clustering a Very Large Number of Textual Unstructured Customers' Reviews in English*. [online]. 2012. DOI: 10.1007/978-3-642-33185-5_5. Dostupné z: http://link.springer.com/10.1007/978-3-642-33185-5_5

ŽIŽKA, J., DAŘENA, F. *Mining Significant Words from Customer Opinions Written in Different Natural Languages*. Lecture Notes in Artificial Intelligence, 2011, sv. 6836, s. 211–218. ISSN 0302-9743.

ŽIŽKA, J., DAŘENA, F. *Mining Significant Words from Customer Opinions Written in Different Natural Languages*. In *Text, Speech and Dialogue*. Heidelberg Dordrecht London New York: Springer, 2011, s. 211-218. ISBN 978-3-642-23537-5.

9 Seznam obrázků

10Seznam tabulek

Přílohy

A Seznamy stop-slov

Německý jazyk

aber	die	ihr	müßt	vor
als	dies	ihre	nach	wann
am	dieser	im	nachdem	warum
an	dieses	in	nein	was
auch	doch	ist	nicht	weiter
auf	dort	ja	nun	weitere
aus	du	jede	oder	wenn
bei	durch	jedem	seid	wer
bin	ein	jeden	sein	werde
bis	eine	jeder	seine	werden
bist	einem	jedes	sich	werdet
da	einen	jener	sie	weshalb
dadurch	einer	jenes	sind	wie
daher	eines	jetzt	soll	wieder
darum	er	kann	sollen	wieso
das	es	kannst	sollst	wir
daß	euer	können	sollt	wird
dass	eure	könnt	sonst	wirst
dein	für	machen	soweit	wo
deine	hatte	mein	sowie	woher
dem	hatten	meine	und	wohin
den	hattest	mit	unser	zu
der	hattet	muß	unsere	zum
des	hier	mußt	unter	zur
dessen	hinter	musst	vom	über
deshalb	ich	müssen	von	

Anglický jazyk

a	added	all	an	anyways
able	adj	almost	and	anywhere
about	adopted	alone	announce	apparently
above	affected	along	another	approxi-
abst	affecting	already	any	mately
accordance	affects	also	anybody	are
according	after	although	anyhow	aren
accordingly	afterwards	always	anymore	arent
across	again	am	anyone	arise
act	against	among	anything	around
actually	ah	amongst	anyway	as

aside
ask
asking
at
auth
available
away
awfully
b
back
be
became
because
become
becomes
becoming
been
before
beforehand
begin
beginning
beginnings
begins
behind
being
believe
below
beside
besides
between
beyond
biol
both
brief
briefly
but
by
c
ca
came
can
cannot
can't
cause
causes
certain
certainly
co
com
come
comes
contain
containing
contains
could
couldnt
d
date
did
didn't
different
do
does
doesn't
doing
done
don't
down
downwards
due
during
e
each
ed
edu
effect
eg
eight
eighty
either
else
elsewhere
end
ending
enough
especially
et
et-al
etc
even
ever
every
everybody
everyone
everything
everywhere
ex
except
f
far
few
ff
fifth
first
five
fix
followed
following
follows
for
former
formerly
forth
found
four
from
further
furthermo-
re
g
gave
get
gets
getting
give
given
gives
giving
go
goes
gone
got
gotten
h
had
happens
hardly
has
hasn't
have
haven't
having
he
hed
hence
her
here
hereafter
hereby
herein
heres
hereupon
hers
herself
hes
hi
hid
him
himself
his
hither
home
how
howbeit
however
hundred
i
id
ie
if
i'll
im
immediate
immediate-
ly
importance
important
in
inc
indeed
index
informati-
on
instead
into
invention
inward
is
isn't
it
itd
it'll
its
itself
i've
j
just
k
keep
keeps
kept
keys
kg
km
know
known
knows
l
largely
last
lately
later
latter
latterly
least
less
lest
let
lets
like
liked
likely
line
little
'll

look
looking
looks
ltd
m
made
mainly
make
makes
many
may
maybe
me
mean
means
meantime
meanwhile
merely
mg
might
million
miss
ml
more
moreover
most
mostly
mr
mrs
much
mug
must
my
myself
n
na
name
namely
nay
nd
near
nearly
necessarily
necessary
need
needs
neither
never
nevertheless
new
next
nine
ninety
no
nobody
non
none
nonetheless
noone
nor
normally
nos
not
noted
nothing
now
nowhere
o
obtain
obtained
obviously
of
off
often
oh
ok
okay
old
omitted
on
once
one
ones
only
onto
or
ord
other
others
otherwise
ought
our
ours
ourselves
out
outside
over
overall
owing
own
p
page
pages
part
particular
particularly
past
per
perhaps
placed
please
plus
poorly
possible
possibly
potentially
pp
predominantly
present
previously
primarily
probably
promptly
proud
provides
put
q
que
quickly
quite
qv
r
ran
rather
rd
re
readily
really
recent
recently
ref
refs
regarding
regardless
regards
related
relatively
research
respectively
resulted
resulting
results
right
run
s
said
same
saw
say
saying
says
sec
section
see
seeing
seem
seemed
seeming
seems
seen
self
selves
sent
seven
several
shall
she
shed
she'll
shes
should
shouldn't
show
showed
shown
showns
shows
significant
significantly
similar
similarly
since
six
slightly
so
some
somebody
somehow
someone
somethan
something
sometime
sometimes
somewhat
somewhere
soon
sorry
specifically
specified
specify
specifying
state
states
still
stop
strongly
sub
substantially

success-
fully
such
sufficiently
suggest
sup
sure
t
take
taken
taking
tell
tends
th
than
thank
thanks
thanx
that
that'll
thats
that've
the
their
theirs
them
themselves
then
thence
there
thereafter
thereby
thered
therefore
therein
there'll
thereof
therere
theres
thereto
thereupon
there've
these
they
theyd
they'll
theyre
they've
think
this
those
thou
though
thoughh
thousand
throug
through
throughout
thru
thus
til
tip
to
together
too
took
toward
towards
tried
tries
truly
try
trying
ts
twice
two
u
un
under
unfortuna-
tely
unless
unlike
unlikely
until
unto
up
upon
ups
us
use
used
useful
usefully
usefulness
uses
using
usually
v
value
various
've
very
via
viz
vol
vols
vs
w
want
wants
was
wasn't
way
we
wed
welcome
we'll
went
were
weren't
we've
what
whatever
what'll
whats
when
whence
whenever
where
whereafter
whereas
whereby
wherein
wheres
whereupon
wherever
whether
which
while
whim
whither
who
whod
whoever
whole
who'll
whom
whomever
whos
whose
why
widely
willing
wish
with
within
without
won't
words
world
would
wouldn't
www
x
y
yes
yet
you
youd
you'll
your
youre
yours
yourself
yourselves
you've
z
zero

Španělský jazyk

él
ésta
éstas
éste
éstos
última
últimas
último
últimos
a
añadió
aún
actualmen-
te
adelante
además
afirmó
agregó
ahí
ahora
al
algún
algo
alguna
algunas
alguno
algunos
alrededor
ambos
ante

anterior
antes
apenas
aproxi-
madamen-
te
aquí
así
aseguró
aunque
ayer
bajo
bien
buen
buena
buenas
bueno
buenos
cómo
cada
casi
cerca
cierto
cinco
comentó
como
con
conocer
consideró
considera
contra
cosas
creo
cual
cuales
cualquier
cuando
cuanto
cuatro
cuenta
da
dado
dan
dar
de
debe
deben
debido
decir
dejó
del
demás
dentro
desde
después
dice
dicen
dicho
dieron
diferente
diferentes
dijeron
dijo
dio
donde
dos
durante
e
ejemplo
el
ella
ellas
ello
ellos
embargo
en
encuentra
entonces
entre
era
eran
es
esa
esas
ese
eso
esos
está
están
esta
estaba
estaban
estamos
estar
estará
estas
este
esto
estos
estoy
estuvo
ex
existe
existen
explicó
expresó
fin
fue
fuera
fueron
gran
grandes
ha
había
habían
haber
habrá
hace
hacen
hacer
hacerlo
hacia
haciendo
han
hasta
hay
haya
he
hecho
hemos
hicieron
hizo
hoy
hubo
igual
incluso
indicó
informó
junto
la
lado
las
le
les
llegó
lleva
llevar
lo
los
luego
lugar
más
manera
manifestó
mayor
me
mediante
mejor
mencionó
menos
mi
mientras
misma
mismas
mismo
mismos
momento
mucha
muchas
mucho
muchos
muy
nada
nadie
ni
ningún
ninguna
ningunas
ninguno
ningunos
no
nos
nosotras
nosotros
nuestra
nuestras
nuestro
nuestros
nueva
nuevas
nuevo
nuevos
nunca
o
ocho
otra
otras
otro
otros
para
parece
parte
partir
pasada
pasado
pero
pesar
poca
pocas
poco
pocos
podemos
podrá
podrán
podría
podrían
poner
por
porque
posible
próximo
próximos
primer
primera
primero

primeros
principal-
mente
propia
propias
propio
propios
pudo
pueda
puede
pueden
pues
qué
que
quedó
queremos
quién
quien
quienes
quiere
realizó
realizado
realizar
respecto
sí
sólo
se
señaló
sea
sean
según
segunda
segundo
seis
ser
será
serán
sería
si
sido
siempre
siendo
siete
sigue
siguiente
sin
sino
sobre
sola
solamente
solas
solo
solos
son
su
sus
tal
también
tampoco
tan
tanto
tenía
tendrá
tendrán
tenemos
tener
tenga
tengo
tenido
tercera
tiene
tienen
toda
todas
todavía
todo
todos
total
tras
trata
través
tres
tuvo
un
una
unas
uno
unos
usted
va
vamos
van
varias
varios
veces
ver
vez
y
ya
yo

Francouzský jazyk

a
à
â
abord
afin
ah
ai
aie
ainsi
allaient
allo
allô
allons
après
assez
attendu
au
aucun
aucune
aujourd
aujourd'hui
auquel
aura
auront
aussi
autre
autres
aux
auxquelles
auxquels
avaient
avais
avait
avant
avec
avoir
ayant
b
bah
beaucoup
bien
bigre
boum
bravo
brrr
c
ça
car
ce
ceci
cela
celle
celle-ci
celle-là
celles
celles-ci
celles-là
celui
celui-ci
celui-là
cent
cependant
certain
certaine
certaines
certains
certes
ces
cet
cette
ceux
ceux-ci
ceux-là
chacun
chaque
cher
chère
chères
chers
chez
chiche
chut
ci
cinq
cinquanta-
ine
cinquante
cinquan-
tième
cinquième
clac
clic
combien
comme
comment
compris
concernant
contre
couic
crac
d
da
dans
de
debout
dedans
dehors
delà
depuis

derrière
des
dès
désormais
desquelles
desquels
dessous
dessus
deux
deuxième
deuxi-
èmement
devant
devers
devra
différent
différente
différentes
différents
dire
divers
diverse
diverses
dix
dix-huit
dixième
dix-neuf
dix-sept
doit
doivent
donc
dont
douze
douzième
dring
du
duquel
durant
e
effet
eh
elle
elle-même
elles
elles-
mêmes
en
encore
entre
envers
environ
es
ès
est
et
etant
étaient
étais
était
étant
etc
été
etre
être
eu
euh
eux
eux-mêmes
excepté
f
façon
fais
faisaient
faisant
fait
feront
fi
flac
floc
font
g
gens
h
ha
hé
hein
hélas
hem
hep
hi
ho
holà
hop
hormis
hors
hou
houp
hue
hui
huit
huitième
hum
hurrah
i
il
ils
importe
j
je
jusqu
jusque
k
l
la
là
laquelle
las
le
lequel
les
lès
lesquelles
lesquels
leur
leurs
longtemps
lorsque
lui
lui-même
m
ma
maint
mais
malgré
me
même
mêmes
merci
mes
mien
mienne
miennes
miens
mille
mince
moi
moi-même
moins
mon
moyennant
n
na
ne
néanmoins
neuf
neuvième
ni
nombreu-
ses
nombreux
non
nos
notre
nôtre
nôtres
nous
nous-
mêmes
nul
o
o|
ô
oh
ohé
olé
ollé
on
ont
onze
onzième
ore
ou
où
ouf
ouias
oust
ouste
outre
p
paf
pan
par
parmi
partant
particulier
particulière
particulière
ticulière-
ment
pas
passé
pendant
personne
peu
peut
peuvent
peux
pff
pfft
pfut
pif
plein
plouf
plus
plusieurs
plutôt
pouah
pour
pourquoi
premier
première
première-
ment
près

proche
psitt
puisque
q
qu
quand
quant
quanta
quant-à-soi
quarante
quatorze
quatre
quatre-vingt
quatrième
quatrième ment
que
quel
quelconque
quelle
quelles
quelque
quelques
quelqu'un
quels
qui
quiconque
quinze
quoi
quoique
r
revoici
revoilà
rien
s
sa
sacrebleu
sans
sapristi
sauf
se
seize
selon
sept
septième
sera
seront
ses
si
sien
sienne
siennes
siens
sinon
six
sixième
soi
soi-même
soit
soixante
son
sont
sous
stop
suis
suivant
sur
surtout
t
ta
tac
tant
te
té
tel
telle
tellement
telles
tels
tenant
tes
tic
tien
tienne
tiennes
tiens
toc
toi
toi-même
ton
touchant
toujours
tous
tout
toute
toutes
treize
trente
très
trois
troisième
trois-ièmement
trop
tsoin
tsouin
tu
u
un
une
unes
uns
v
va
vais
vas
vé
vers
via
vif
vifs
vingt
vivat
vive
vives
vlan
voici
voilà
vont
vos
votre
vôtre
vôtres
vous
vous-mêmes
vu
w
x
y
z
zut