Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

DIPLOMOVÁ PRÁCE

PLZEŇ, 2013 **VÁCLAV VOLDŘICH**

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni 30. srpna 2013 .....

vlastnoruční podpis

Anotace

Tato práce se zabývá shlukovacími metodami v oblasti textu, které jsou testovány za účelem použití jako příprava dat pro adaptaci jazykového modelu. Zároveň se shlukovacími metodami jsou probrány možnosti různých typů příznaků, a následně jsou diskutovány výsledky při jejich kombinování. Součástí práce je i rozhraní pro klasifikaci dokumentů.

Klíčová slova: shlukovací metody, příznaky, K-means, klasifikace

Abstract

In this thesis is discussed a range of clustering methods tested for use in language adaptation. Along with clustering algorithms there are introduced feature space models and the combination with clustering of documents is discussed afterwards. A part of project is a classification framework.

Key words: topic based clustering, feature space model, classification

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Cílem této práce je otestování různých metod v různých podmínkách za účelem jejich použití v adaptaci modelu. S použitím korpusu dat a jednoho cílového dokumentu, jehož tématu je potřeba přizpůsobit jazykový model. Za pomoci vybrané metody se korpus rozdělí na několik shluků, a shluk, ve kterém je zahrnut daný cílový dokument, bude poskytnut k dalšímu zpracování.

Dříve vypracovaná bakalářská práce udala směr této diplomové práci a poskytla i základy k jejímu vypracování. Pro dosažení vytyčených cílů bylo využito několika algoritmů, včetně základního prostředí pro testování, který bude popsán v části 3.

Hlavním tématem této práce jsou shlukovací algoritmy. Shlukování pochází původně z vědního odvětví antropologie, ve kterém jej představili Driver a Kroeber v roce 1932, avšak velké proslavení přišlo až v roce 1943, kdy jej Cattell použil na klasifikaci psychologických osobností v oblasti teorie charakteristik.

Obecně je shlukování statistický přístup pro vytvoření skupin, tzv. *shluků*, objektů, které mohou být určeny na základě vyšší hustoty bodů v daném prostoru, mající dané rozložení, nebo nacházející se v daném intervalu. Prostředkem shlukové analýze jsou například centroidově orientované algoritmy, založené na spojování do celků, hustotní či distribuční modely, nebo grafové algoritmy. Účelem pak může být analýza obrazu, statistických dat, *data mining*, rozpoznávání obrazu či klasifikace.

Částečné shodné cíle mají například i v [1], kde je pojednáno o využití latentní sémantické analýzy, která dokáže najít informace, které jsou obsaženy ve významu, ale nemusí být obsaženy ve formě. Zároveň zde používají LSA v kombinaci s n-gramy, které jsou v oblasti zpracování textu oblíbené, především pro svoji schopnost využít vyskytující se sousloví. Zmíněné metody jsou popsány v sekcích 1.2 a 1.3.

V [2] se pak zaměřili na ICA, která je popsána v sekci 2.2.3, a na SOM, *Self-Organizing Map*, nebo též *Kohonenova mapa*, což je typ neuronové sítě, která plní účel snižování mnoho dimenzionálního prostoru na obyčejně dvě dimenze.

V následujících kapitolách budou nejdříve představeny příznkové prostory, které následují použité metody pro shlukování. Implementace jednotlivých metod je popsána v kapitole 3, kde je popsáno testovací prostředí. Na závěr jsou porovnány výsledky jednotlivých metod v kombinaci s příznaky, a diskuze problematiky napojení této práce na stávající projekt adaptace jazykového modelu. V přílohách se pak nacházejí výsledky algoritmů uspořádané do tabulek.

Kapitola 1

Předzpracování

Abychom mohli vybrat vhodné dokumenty k adaptaci modelu, musíme vybrat dokumenty tématicky podobné. Člověk přirozenou inteligencí při výběru rozhoduje podle obsahu, kontextu a významu. Význam je prozatím v oblasti umělé inteligenci bohužel stále velmi složité získat. Samotný obsah nám zase nemusí prozradit celou informaci o autorově myšlence.

Analýza menších jednotek, než jsou slova, nemá valný smysl. Proto prvotní analýza bude probíhat na úrovni slov. Protože je čeština bohatá na tvary slov a obsahuje nemalé množství slov, tzv *stop-slov*, která nenesou podstatnou informaci, lze množství slov redukovat. Slova surového korpusu jsou tedy filtrována. Tedy ta, jenž se nachází na seznamu předem vybraných stop-slov.

Pro další snížení roztříštění informace napříč různými tvary slov lze použít *lemmatizaci*. Ta převede všechna slova na základní tvar, tedy na první pád jednotného čísla pro subjektiva, infinitiv pro slovesa, a první stupeň pro příslovce.

Před samotným shlukováním je třeba text připravit pro výpočty. To je možné převedením do vhodného vektorového modelu. Nejjednodušším je pak počítání slov v daném dokumentu. Tato informace je ovšem znehodnocená v případě různě dlouhých dokumentů.

1.1 TF-IDF

Ani po zprůměrování (tato forma se nazývá *Term frequency*, *TF* a vypočítá se podle vzorce 1.1) nejsou data zcela vhodná pro použití, především z toho důvodu, že obecné slovo se vyskytuje ve všech dokumentech, není proto pro klasifikaci důležité. Naopak slova, která jsou požívané pouze v určité oblasti jsou méně zastoupená. Tento problém řeší *TF-IDF*, tedy *Term frequency - Inverse document frequency*. Ta je jedním ze základních používaných vektorových modelů dokumentů, a skládá se z váhy slova *IDF* a samotné *TF*, viz vzorce

1.2, 1.3.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{1.1}$$

$$idf_i = \log \frac{|D|}{|\{d : t_i \in d\}|}, \tag{1.2}$$

$$(TF\text{–}IDF)_{i,j} = idf_i \cdot tf_{i,j}. \tag{1.3}$$

Pokud jsou vypočtené vektory, resp. matici $t \times d$, s hodnotami TF-IDF, je připravený prostor pro klasifikaci. Dokument, který ale bude klasifikován bude muset být také převeden do tohoto prostoru. Nejprve se tedy tedy musí jednotlivé termy spočítat, a přepočítat na frekvence za pomocí slovníku převádějící termy na index $0..t$.

Slova, která nebyla ve slovníku můžeme vypustit, protože pokud se slovo nevyskytovalo v původním prostoru, sloupce matice by pak měly hodnotu na dané pozici nulovou, a při klasifikaci by se klasifikovaný dokument promítal do nižšího prostoru.

Vektor frekvencí je pak po složkách vynásoben vektorem *idf*, který byl uchován z procesu shlukování trénování množiny.

1.2 N-gram

Použijeme-li základní myšlenku *TF-IDF*, a rozšíříme-li jí o kontext, dostaneme *n-gram*. Ten, narozdíl od původní myšlenky nehodnotí jen jedno slovo, ale n za sebou jdoucích slov. Zaznamenají se tak i slovní spojení, která mohou nést další informaci.

Ovšem při sledování n-tic slov se projeví exponenciální zvyšování dimenze při zvyšování n. Platí totiž závislost 1.4.

$$\text{počet n-gramů} \leq t^n, \tag{1.4}$$

kde t je počet termů. Při modelování pak také můžeme využít kombinace různých, potom dimenze nadále vzrůstá. Využijeme-li n v rozsahu 1 až N, cílový počet sledovaných subjektů se chová dle závislosti 1.5.

$$\text{počet n-gramů} \leq \sum_{n=1}^{N} t^n \tag{1.5}$$

V tomto případě byly spočteny unigramy, bigramy a trigramy, se kterými se zacházelo jako s termy, a byly tedy vyděleny velikostí dokumentu a následně převáženy *IDF*. Informace obsažená v bigramech a trigramech rozšiřuje obsah informace a dokumenty obsahující stejná slovní spojení si budou více podobné.

N-gramy mohou sloužit nejenom pro extrakci příznaků, ale i jako prediktivní jazykový model, který se využívá například ve zpracování přirozené řeči, a to jak v analýze, tak syntéze. Velmi rozšířen je pak jako skrytý Markovův model (HMM).

Obecně se však nepoužívají pouze ve zpracování textu či modelování přirozeného jazyka. Své uplatnění našly v laboratořích zabývajících se sekvencováním genomů či proteinů. Svoji implementaci má například i společnost Google, která analýzou Google Books získala jazykový model, a nabízí k nahlédnutí vývoj procentuální četnosti zadaných frází v knihách v průběhu let.

1.3 LSA

Ještě o krok dál pak jde *LSA*, tedy *Latentní Sémantická Analýza*. Jak název naznačuje, je to metoda extrakce příznaků, která svou podstatou zachycuje význam za slovy a tedy i váží slovní spojení, která nemusí být v blízkém sledu.

Princip této metody spočívá v rozložení *matice výskytů*, tedy matice složená z řádků, které popisují rozložení termů v dokumentech, sloupce pak termy, které jsou v dokumentu zastoupeny. Tato matice je převážena *TF-IDF*.

1.3.1 SVD

Rozklad probíhá pomocí *SVD* (*Singular Value Decompositon - Rozklad na singulární hodnoty*). Při transformaci dochází k natočení celého prostoru a tím i ke změně pohledu na data, a zároveň se model transformuje do nižší dimenze velikosti počtu dokumentů.

Samotný rozklad pak zachovává v patrnosti následující rovnici 1.6:

$$M = USV^T, \tag{1.6}$$

kde M je rozkládaná matice výskytů rozměru $t \times d$, U je unitární matice dimenze $t \times t$, pro něž obecně platí $U = U^{-1}$, S je diagonální matice singulárních hodnot dimenze $t \times d$, a V^T je transponovaná unitární matice dimenze $d \times d$. Rozklad je implementován využitím knihovny, která se řídí metodou popsanou v [3].

Při použití LSA se pak podobnost počítá za využitím vzorce 1.7, kde je původní vektor příznaků nahrazen násobkem, který je popsáno ve vzorci 1.8. Násobek se skládá z diagonální matice singulárních hodnot S, a řádku pravé matice V, který představuje daný dokument.

$$s(d_i, d_j) = s(\tilde{d}_i, \tilde{d}_j) \tag{1.7}$$

$$\tilde{d}_i = d^T \dot{U} \tag{1.8}$$

Snížením dimenze se sníží náročnost výpočtů algoritmů, za cenu složitějšího předzpracování. Je pak nasnadě zamyšlení nad potřebou opakovaných výpočtů, například při určování počtu shluků, nebo eliminaci vlivu náhodných počátečních podmínek různých metod.

LSA je díky zobecňování vztahů mezi slovy používaná v různých oblastech textu. Lze ji využít k vyhledávání a získávání informací, kde je známa spíše pod názvem *latentní sémantické indexování*. Zvládá jak i synonyma a polysémii, tak i po analýze přeložených dokumentů i vyhledávat v jiném jazyce, než v jakém byl dotaz.

1.4 Kombinace N-gramu a LSA

Poslední zmíněnou metodou předzpracování je pak kombinace *LSA* a *n-gramů*. Spojuje se tím výhoda n-gramu modelování vztahů slov v po sobě jdoucí sekvenci a modelování vztahů slov napříč dokumentem vyplývající z LSA. Úprava spočívá v záměně termů za n-gramy. Dochází tedy v přiřazení $|t| :=$ počet n-gramů.

Původní modelování vzájemné závislosti jednotlivých slov se tak nyní modelují závislosti i slovních spojení. To dodává další sílu pro analýzu významu slov a slovních spojení.

Tímto spojením se mimo jiné i eliminuje dříve zmíněná nepříjemná vlastnost n-gramů daná závislostmi 1.4 a 1.5.

Kapitola 2

Shlukovací algoritmy

2.1 Metody s volbou K

2.1.1 K-means

Jedním ze základních shlukovacích algoritmů je *K-means* (*MacQueen, 1967*) a mimo zpracování textu slouží i k vektorové kvantizaci. Na základě výběru K náhodných dokumentů a podobnostní míry jsou vytvořeny shluky, a jejich průměrem je vytvořeno K *centroidů*. V následující iteraci jsou všechny dokumenty přiřazeny k nejbližšímu centroidu, čímž jsou opět vytvořeny shluky, a následně nové i centroidy tak, jak je popsáno ve vzorci 2.1.

$$c_i = \frac{1}{|S_i|} \sum_{x \in S_i} x \tag{2.1}$$

Algoritmus se zastaví v okamžiku, kdy se jednotlivé centroidy neposunou více, než je daný práh (`epsilon`), nebo počet iterací nepřesáhne maximální počet (`iter`). Tyto parametry jsou společné pro všechny v této práci zmíněné varianty algoritmu K-means.

Metoda K-means je velmi oblíbená především z důvodu jednoduchosti algoritmu, a skládá se z jednoduchých operací. Hlavní nevýhodou je pak nutnost znalosti počtu shluků. Další je pak náhodné počáteční podmínky, z čehož vyplývá, že dva po sobě jdoucí běhy nemusí mít kvalitativně stejný výsledek. Výsledek také neblaze ovlivňují body mimo přirozený shluk, tedy shluk, který odpovídá realitě. Takové body se nazývají *outliners* a právě takové body mohou vychylovat střed shluku, a ovlivnit tak klasifikaci.

2.1.2 Vylepšené K-means

Úpravou předchozího algoritmu získáme *vylepšené K-means*. Úprava spočívá v přepočítání centroidu po každém přiřazení k danému shluku a to tak, že se počítá postupný

průměr ve smyslu 2.2.

$$c_i = \frac{1}{|S_i| + 1} \left(|S_i| \cdot c_i + x\right) = \frac{|S_i|}{|S_i| + 1} \left(c_i + \frac{x}{|S_i|}\right), \tag{2.2}$$

kde $|S_i|$ značí velikost daného shluku před přidáním, a x je přidávaný dokument.

2.1.3 Bisecting K-means

Trochu jiný přístup má pak *Bisecting K-means*. Ten bere celou množinu dat a postupně jí dělí na menší shluky, dokud jich nezbyde požadovaný počet. Nejprve rozdělí celou množinu na dva shluky, a následně se vždy vybere nezvětší z nich a ten se znovu rozdělí. K dělení se používá buď K-means, nebo Vylepšené K-means.

Oba tyto algoritmy mají náhodné počáteční podmínky, a při některých dělení může být nepřesné. Proto se používá více iterací pro jednotlivá dělení. Počet iterací je dán (stejně jako počet iterací podalgoritmu), parametrem `iter`.

Z takovýchto iterací je pak potřeba vybrat tu nejlepší. K tomu slouží kriteriální funkce, jíž může být například celková podobnost iterace. Ta se spočítá podle vzorce 2.3.

$$S = \sum_{i=1}^{K} c_i^T c_i, \tag{2.3}$$

kde c_i je i-tý centroid.

Další nevýhodou bisecting k-means je předpoklad, že jsou všechny shluky stejně velké. To ovšem v reálném světě nemusí být vždy pravda, a tak může nastat situace, kdy při výběru se při maximalizaci celkové podobnosti vybere největší shluk, který ale už může být přirozeným shlukem s velkým počtem dokumentů.

2.1.4 Refined bisecting K-means

Vzhledem k tomu, že další dělení prováděná v bisecting K-means mají omezený rozsah dělením předchozím, mohou tak vznikat nepřesnosti. Ty se snaží odstranit *Refined bisecting K-means*. Výsledek Bisecting K-means předá jako počáteční podmínky jedné z metod K-means, respektive Vylepšené K-means.

2.1.5 Aglomerativní K-means

Opačným směrem než bisecting K-means jde aglomerativní K-means. Narozdíl od něj nerozděluje data na menší části, ale naopak jednotlivé dokumenty slučuje do shluků podle toho, jak moc jsou si podobné. Ovšem narozdíl od něj bere v úvahu pouze lokální data, a nemá tak přehled o globálním rozložení dat. Metoda se zastaví, jsou li data sloučena do přesně K shluků.

2.2 Metody volící K

Algoritmy prezentované v předchozí podkapitole požadovali po uživateli a priori informaci o počtu shluků, které se ve vektorovém prostoru objevují. Naproti tomu v metodách, které budou následovat, probíhá kontrola daného kritéria či jiný odhad tohoto parametru. V podkapitole 2.4 je pak nabídnuto několik možností, jak využít předchozí algoritmy bez informace o počtu shluků.

2.2.1 Mean-shift

Metodu *Shift-mean*, tedy posouvání průměry lze obecně použít například pro vektorovou kvantizaci, vyhledávání míst s vyšší hustotou prvků v prostoru, nebo pro počítačové vidění, kde ji lze využít k detekci pohybu. Byla poprvé představena v roce 1975 autory Fukunaga, Hostetler.

Jejím základem je *Kernel function* (ve volném překladu *jádrová funkce*) a pro data dle normálního rozložení ji lze zapsat jak je popsáno ve vzorci 2.4. Tou se váží každý okolní průměr, který přispívá do nově počítaného průměru. Výpočet nových průměrů je vidět ve vzorci 2.5.

$$K(m,n) = e^{-c*||m-n||^2}, \tag{2.4}$$

kde c je konstanta útlumu vlivu. Ta především ovlivní, jak moc se posune průměr, jsouli od sebe body více vzdálené. Při hodnotách $c > 0$ jsou vzdálenější body penalizovány, a to tím víc, čím větší hodnota c je.

$$s(m) = \frac{\sum_{n \in N(m)} K(m,n) \cdot n}{\sum_{n \in N(m)} K(m,n)}, \tag{2.5}$$

kde $||d||$ představuje normu, a funkce $N(m)$ značí okolní bodu m. Jako inicializační podmínky platí $m^{(0)} := x$, kde x jsou veškeré dokumenty. Pro přechod do další iterace se pak přiřadí $m^{(i+1)} := s(m^{(i)})$.

Algoritmus je upraven tak, že se po každé iteraci zkontrolují všechny středy, a pokud jsou libovolné dva blíže, než je stanovená mez (parametr `filter`), jeden z nich je odstraněn. Byl by totiž ovlivněn stejným okolím, a postupem iterací by měl stejnou dráhu prostorem. Protože by pak ale bylo ovlivněno okolí, jsou uchovávány váhy. Ta odpovídá počtu středů, které byly postupem algoritmu sloučeny do daného středu, a dala by se tedy popsat *násobnost středu*. Tj. sloučí-li se dva středy, a následně pak s třetím, váha je tedy postupně nejdříve 2, a po druhém sloučení 3.

Ve výchozím stavu je tato filtrace zapnuta. Změnit chování pak lze nastavením parametru `tofilter` na jednu z následujících hodnot:

`in` Bude se provádět filtrace po každé iteraci.

`both` Bude se provádět filtrace po každé iteraci a na konci.

`end` Bude se provádět filtrace pouze po dokončení algoritmu.

`none` Nebude se provádět filtrace.

Po skončení algoritmu jsou každému středu přiřazeny dokumenty. Může se ale stát, že bude existovat takový střed, který nebude mít přiřazen žádný dokument, a v takovém případě není užitečný. K tomuto jevu může například nastat, překrývají-li se dva středy při vypnuté filtraci. Takové středy odfiltrovány nezávisle na nastavení parametru `tofilter`.

Jednou z ukončovacích podmínek je velikost sumy vzdáleností jednotlivých středů od minulé iterace tak, jak je popsáno ve vzorci 2.6, kde mez ϵ_{iter} je v algoritmu značena jako `iter`. Druhou je pak počet iterací, který je značen jako v předchozích případech `iter`.

$$\epsilon_{\text{iter}} > d(m^{(i+1)} - m^{(i)}), \tag{2.6}$$

kde $d(,)$ značí vzdálenosní míru.

k - Nearest Neighbour

Další důležitou částí je funkce $N(m)$, která představuje okolí bodu m. V tomto případě je použita metoda *k-nejbližšího souseda*, tedy *k-Nearest Neighbours*, obecně známa také jako *k-NN*. Platí tak že pokud seřadíme všechny body dle vzdálenosti k x jak je naznačeno ve posloupnosti 2.7, metoda nabídne prvních k členů poslouponosti 2.8, tedy $\{i \in 1 \dots k : p_i\}$.

$$d(p_i, x) <= d(p_{i+1}, x) : \tag{2.7}$$

$$(p_i)_1^N : p_1, p_2, \cdots, p_{k-1}, p_k, p_{k+1}, \cdots, p_N \tag{2.8}$$

V programu se tak prochází všechny body a měří se vzdálenost od cílového bodu, a K nejbližších výsledků se udržuje v paměti. k tedy definuje velikost okolí, které ovlivní daný střed, a stává se tím parametrem metody (v algoritmu označen jako `knn`. Nabízí se i jiný přístup, a to nabídnout takové body v prostoru, které jsou blíže, než je daná hranice.

k-nejbližší soused lze také využít pro klasifikaci s učitelem, kde se klasifikovaný objekt převede do příznakového prostoru, který byl předem natrénován, a následně podle k-nejbližšího souseda (nejčastěji je voleno liché k, protože to odbourává nejistotu výběru třídy) rozhodne, k jaké třídě objekt patří.

Z důvodu úpravy algoritmu Mean-shift bylo nutné upravit i metodu k-NN. Ta spočívá v držení informace o váze, tedy násobnosti bodu. Nejprve se udělá standardní k-NN, následně se kontroluje násobnost. Označíme-li násobnost $w(x)$, pak pro P-násobný bod p_i platí 2.9, kde $(p^i)_1^P$ je posloupnost P bodů p_i.

$$(p_j^i)_{j=1}^P : p_i, p_i, \cdots, p_i \tag{2.9}$$

Pro všechny body pak musí platit 2.10.

$$\sum_{i=1}^{M} w(p^i) = N \tag{2.10}$$

Seřazenou posloupnost 2.8 lze tedy přepsat jako posloupnost 2.11, tedy jako posloupnost posloupností 2.9.

$$\begin{aligned}(p^i)_{i=1}^{M} : p^1, p^2, \cdots, p^M = \\ = \quad (p_j^1)_{j=1}^{w(p_1)}, (p_j^2)_{j=1}^{w(p_2)}, \cdots, (p_j^M)_{j=1}^{w(p_M)} = \\ = \quad p_1, \cdots, p_1, p_2, \cdots, p_2, \cdots, p_M, \cdots, p_M\end{aligned} \tag{2.11}$$

Je-li násobnost nejbližšího středu n, vrátí se tedy ve výsledné posloupnosti n krát. Je-li součet násobností prvních m bodů posloupnosti 2.8 menší než k, a součet prvních $m+1$ bodů větší, pak se ve výsledku objeví prvních m bodů v počtu jejich násobnosti, a $m+1$. v násobnosti zbylé dostupné. Matematicky je proces popsán ve vzorci 2.12

$$\sigma = \sum_{i=1}^{m} w(p^i) < k < \sum_{i=1}^{m+1} w(p^i) : [(p^i)_{i=1}^{m}, (p_i^m)_{i=1}^{k-\sigma}] \tag{2.12}$$

Lze poznamenat, že metoda pak nevrací k bodů p^i, ale $m = 1 - k$ s tím, že platí $\sum_{i=1}^{m} w(p_i) = k$. Toto pravidlo bráno v potaz, metoda k-NN v této implementaci vrací sdružené body p^i spolu s jejich násobností $w(p^i)$. Toho lze s výhodou použít v upravené verzi funkce 2.5, která je popsána ve vzorci 2.13.

$$s(m) = \frac{\sum_{n \in N(m)} K(m,n) \cdot w(n) \cdot n}{\sum_{n \in N(m)} w(n) \cdot K(m,n)} \tag{2.13}$$

2.2.2 PCA

Rozdílný přístup volí *Principal Component Analysis*, tedy *PCA* (v překladu *analýza hlavních komponent*), založena na hledání nekorelovaných, nezávislých složek dat. Prostředkem pro dosažení je opět maticový rozklad, například SVD nebo rozklad na vlastní čísla. Historie sahá až do roku 1901, kdy byla tato metoda poprvé použita v mechanice, a od té doby se používá v nejrůznějších vědních oblastech a disciplínách.

Cílem PCA je najít matice T a W takové, aby platilo $T = XW$; W jsou vlastní vektory $X^T X$; T je potom matice vah komponent (z orig. *component score*).

Principiálně se nejprve najde první hlavní komponenta a to maximalizací výrazu 2.14. Následně se itarativním procesem získávají další komponenty tak, že se rozkládaná matice upraví odečtením předchozích komponent dle vzorce 2.15, a následně probíhá maximalizace dle výrazu 2.16.

$$w_{(1)} = \arg\max_{||w||=1} \{||Xw||^2\} \tag{2.14}$$

$$\hat{X}_{k-1} = X - \sum_{s=1}^{k-1} X w_{(s)} w_s^T \tag{2.15}$$

$$w_{(k)} = \arg\max_{||w||=1} \{||\hat{X}_{k-1} w||^2\} \tag{2.16}$$

2.2.3 ICA

Na podobných principech funguje *Independent Component Analysis* (dále jen *ICA*), neboli *Analýza nezávislých komponent*. Ta má původ ve zpracování signálů, kdy jsou pozorovány vzájemně ovlivněné signály, které byly původně nezávislé. ICA však v zásadě neumožňuje najít správný počet komponent ani správné pořadí zdrojů.

Prostředkem pro dosažení stanoveného cíle metoda využívá minimalizaci vzájemné informace, nebo maximalizace *Non-Gaussianity*. To jsou dva různé přístupy vycházející z různých definic nezávislosti.

Před použitím algoritmu se data musí nejdříve vystředit (vz. 2.17) a vybělit (tak aby platilo 2.18).

$$x := x - E\{x\} \tag{2.17}$$

$$E\{xx^T\} = I \tag{2.18}$$

Pro bělení dat lze využít například rozklad na vlastní čísla analýzou hlavních komponent nebo SVD (viz sekce 1.3.1). Metody PCA a ICA jsou implementovány v knihovně `scikit-learn` [4] .

2.3 Výběr míry

Předložené algoritmy využívají míru, ať už podobnostní, nebo vzdálenostní. Proto je důležité správně vybrat. Jako podobnostní míra byla zvolena *cosinová podobnost*. Ta vyjadřuje úhlovou vzdálenost mezi dvěma vektory ve vektorovém prostoru a lze jí spočítat dle vz. 2.19.

$$s_{cos}(x, y) = \frac{(x^T y)}{||x|| \cdot ||y||}, \tag{2.19}$$

Mezi další podobností míry patří například *Tanimoto measure*, nebo podobnost založená na trojúhelníkové podobnosti. Naopak jako vzdálenostní míra byla zvolena *Eukleidovská vzdálenost*. Ta je jako tradičně počítána dle vzorce 2.20.

$$d_2(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \tag{2.20}$$

Zobecněním euklidovské vzdálenosti dostaneme *Minkovského* normu (vz. 2.21). V případě $p = 2$ pak dostaneme výše zmíněnou Eukleidovskou normu.

$$d_p(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}, \tag{2.21}$$

2.4 Výběr K

Pro metody uvedené v podkapitole 2.1, které ze své podstaty potřebují znát počet témat pro vytvoření shluků. Tato problematika velmi ošemetná, protože K je závislé na datech. Ty mohou mít různé rozdělení a/nebo různý rozptyl. Jeden extrém je, že se vytvoří jeden velký shluk, což ovšem popírá podstatu shlukování za účelem klasifikace, druhým pak, že každý jednotlivý bod prostoru bude samostatným shlukem, což má stejný efekt. Logicky tedy bude optimální hodnota někde mezi. Empirické pravidlo vypozorované časem je udáno vzorcem 2.22, kde $||t||$ je počet dokumentů.

$$K \approx \sqrt{\frac{1}{2}||d||} \tag{2.22}$$

2.4.1 Metoda ohybu

Metoda s originálním názvem *Elbow method* je heuristická metoda, pomocí níž lze nalézt parametr K. Jejím principem je sledovat chování kritéria, zvyšujeme-li postupně K. Dochází tak k iterativnímu procesu, kdy pomocí zvoleného shlukovacího algoritmu dostáváme výsledky. Předpokládá se, že Kritérium bude monotónně závislé na K, a při překročení skutečné hodnoty se kritérium razantně skokově změní. Cílem metody je tak najít v posloupnosti tuto změnu a tak usoudit, jaké K je nejvhodnější.

Největší překážkou této metody je nejasná formulace takovéhoto kritéria. Nabízí se například celková podobnost shluků nebo rozptyl. Druhou je pak pojem „razantní změna“, která nemusí být zcela jednoznačně zachytitelná. Je totiž jasné, že budeme-li počet shluků postupně snižovat, bude se například rozptyl neustále snižovat a v konečném případě bude rozptyl nulový.

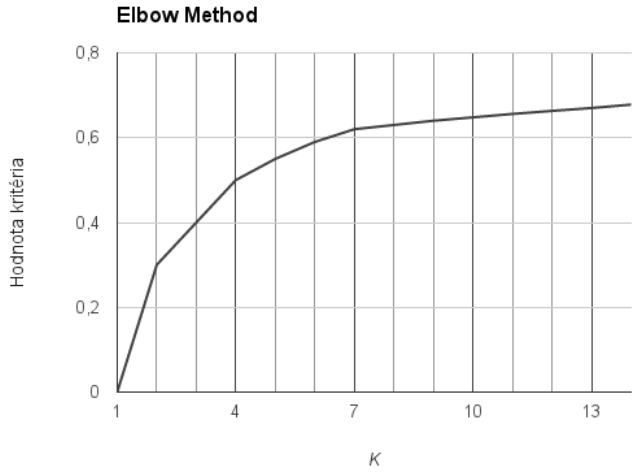
Další nepříjemnou vlastností je opět náhodný charakter metod K-means. Budou-li vybrány nevhodné počáteční podmínky, bude hodnota kritéria ovlivněna. Toto lze ovlivnit například opakováním pokusů (parametr `eiter`) a vybráním nejvhodnějšího výsledku, podobně, jako v metodě bisecting K-means.

Implementované metodě pak lze poskytnout informaci o odhadovaném intervalu, kde se má vyskytovat K: od `kestlow` do `kesthi`. Při vyhodnocování podmínky na kritéria iterací volby K se vytvoří posloupnost rozdílů dvou po sobě jdoucích hodnot. Poté se rozdíl porovná s parametrem `varkrit` a vybere se takové nejmenší možné K takové, aby kritérium vyhovovalo podmínce. Pokud ani jeden výsledek podmínce nevyhoví, `varkrit` se sníží o hodnotu parametru `varstep` a vyhodnocení se opakuje.

Protože se o parametru K rozhoduje během některé ze shlukovacích metod, uchovává se kromě hodnoty kritéria také výsledek, tedy vytvořené shluky. Počet těchto shluků je pak vybrané K.

Příklad průběhu sledovaného kritéria si lze představit jako na ilustračním grafu 2.1. Z obrázku je patrné, že změna je relativně plynulá, lze ale vyčíst, že v bodě $K = 4$ dochází ke změně a bude pravděpodobně odpovídat skutečnému parametru K. Lze poznamenat,

že optimální K bude ležet v inflexním bodě funkce kriteriální funkce, ovšem vzhledem k charakteru (diskrétnosti) měření, tento bod je jen velmi těžké odhadnout.



Obr. 2.1: Ilustrační průběh kritéria

Neblahý vliv má také případná nemonotónnost kriteriální funkce, která může být zapříčiněna šumem v podobě náhodných podmínek, či nevhodnou volbou kritéria

Možným vylepšením by mohlo být například postupné kontrolování hodnoty kritéria a při překročení hlídané hodnoty by se metoda zastavila, narozdíl od stávající verze, kdy se předpočítá vývoj kritéria, a z něj se pak určí nejlepší možný výsledek.

2.4.2 Hierarchické metody

Hierarchické metody už z principu nepotřebují a priori znalost počtu shluků, pokud je navrženo kritérium, při kterém se hierarchie rozdělí. V této části budou popsány dva algoritmy, které lze podobným způsobem navrhnout. Právě návrh kritéria je velmi důležitý, protože právě na něm stojí, jak kvalitní bude odhad K. Ovšem je ho nutné navrhnout na charakter dat, který může být u každé množiny jiný. Přirozené shluky tak můžou mít jiné parametry nebo rozložení. Nevýhody z předchozí sekce týkající se sledování vývoje kritéria tak zůstávají přítomné.

Bisecting K-means

Algoritmus bisecting K-means ze sekce 2.1.3 lze upravit tak, že se nezastaví na daném počtu shluků, ale až po dosažení požadovaného kritéria. Je nutné pomocí sekundárního

určit, který shluk se se bude dělit. Takovým sekundárním kritériem může být například počet dokumentů ve shluku, rozptyl shluk, hodnota střední kvadratické chyby, nebo celkovou podobnost shluků.

Aglomerativní K-means

I metoda popsaná v s sekci 2.1.5 je upravitelná do podoby, kdy si sama určí počet shluků. Při sledování hodnoty kritéria pak lze opět usoudit, kdy se je vhodné slučování prvků zastavit.

Kapitola 3

Testovací prostředí

K testování a zkoušení jsem testovací prostředí, které má kořeny v předchozí bakalářské práci. V té se počítalo pouze s jedním typem příznaků, naproti tomu v této práci se počítá s testováním hned několika. Proto bylo třeba vytvořit unifikované prostředí, které bude počítat s více metodami stejně jako s více typy příznaků.

Zásadní změnou pak byl přechod od původního modelu, kde příznak byl vektor zaobalený do třídy, která měla metody pro manipulaci s ním k modelu stávajícímu - vektor je zaobalený do menší třídy nesoucí pouze přídavné informace a poskytující přístupové metody, množina příznaků je však zaobalená do větší třídy, která poskytuje metody pro manipulaci nad množinou příznaků. Metody nezávislé na typu příznaků byly přesunuty do knihovního modulu.

Následně byly parametry metod upraveny tak, aby více vyhovovaly běhu ve skriptu, který připravuje inicializační data, sbírá informace o průběhu a provádí vyhodnocování každého běhu. Tyto informace jsou pak zapsány do souboru.

Nad tímto skriptem pak běží nadřazený skript, který připravuje příznaky, a zaznamenává trvání přípravy, stará se vstupní soubory a vytváří hierarchickou strukturu adresářů pro přehlednost výstupů.

Celkově je tedy projekt rozdělen do několika balíků:

`TextFeatures` obsahuje moduly týkající se příznaků.

`TextClustering` obsahuje moduly týkající se shlukovacích metod.

`TextLibrary` který nabízí podpůrné metody ostatním balíkům modulů.

`Scripts` skýtá testovací skripty a vytváří tak prostředí pro běh metod.

Postup dat programem je znázorněn na schématu A.1, který je přiložen. Zde je vidět jak jsou data postupně zpracovávána. Modul `Preprocessing` zpracuje data od uživatele a to buď ze souboru, či polem (více v sekci 3.2) a slouží tak i k částečné validaci. Dále se data, která jsou stále v textovém formátu, rozdělena po slovech, případně přefiltrována, zpracují na zvolený typ příznaků. Jednotlivé typy příznaků jsou nabídnuty v balíku `TextFeatues`.

Balík dokumentů lze nyní uložit do úložiště k pozdějšímu použití, aby byl ušetřen výpočetní výkon a především čas. Plnou čarou jsou znázorněny datové toky, které postupují od začátku včetně zpracování, přerušované pak toky, které využívají uložených dat z předchozího zpracování. K ukládání a načítání dat slouží knihovna `FeaturePickle`, která je popsána v části 3.1.

Vytvořené nebo načtené příznaky pak lze shlukovat jedním z představených algoritmů, které se nachází v balíku `TextClustering`. Ten vrátí seznamy celočíselných indexů, které odpovídají původním dokumentům, kde každý seznam odpovídá jednomu shluku. Takovéto výslekdy pak lze uložit na disk, opět s využitím knihovny `FeaturePickle`.

Pro klasifikaci dokumentu je nutno ho nejdříve převést do daného příznakového prostoru. Tak lze u učinit metodou `Encode`, kterou poskytují všechny příznakové objekty. Jsou-li připravené příznaky, výsledek shlukování a převedený dokument, klasifikace podle nejbližšího středu shluku je poskytnuta modulem `Processing`, který nabízí metodu `Classify`. Ta vrátí pouze jeden shluk ve stejném formátu, jako shlukovací metody, tedy jako seznam indexů.

3.1 Knihovna FeaturePickle

Jednou knihovnou poskytovanou modulem `TextLibrary` je `FeaturePickle`. Ten poskytuje soubor metod, které mají za úkol načítat a ukládat dokumenty převedené do jednoho z vektorových prostorů. Tyto uložené soubory jsou generovány pomocí knihovny Pythonu `pickle` a neprobíhá žádná validace obsahu dat, kromě správného formátu.

V případě metody `tryLoadCreate` se nejdříve pokusí načíst dané soubory, které jsou potřebné ke kompletní sadě dat. Při jakémkoli selhání se pak data načtou ze vstupního souboru, patřičně se zpracujou, a následně výsledek uloží na disk, aby při dalším běhu nebylo nutné data opět zpracovávat.

3.2 Knihovna Processing

Ústřední metodou knihovny `Processing` z balíku `TextLibrary` je `PrepareData`, jejíž účelem je kontrola a zpracování vstupních parametrů a příprava dat pro jednotlivé metody. Variabilita parametrů je široká, a jejich popis následuje:

`data` Tento parametr by měl obsahovat data samotná, či odkaz na ně. Daty se v tomto případě myslí posloupnost slov v `list`. Možností, jak zadat tento parametr je několik:

- Textem: parametr je `str`, pro získání termů je použito rozdělení oddělovačem mezerou. Jednotlivé dokumenty jsou rozděleny podle escape sekvence `\n`.
- Seznamem: parametr je seznam, jejíž obsahem jsou jednotlivé dokumenty ve formátu `str`.

- Souborem: Soubor obsahuje jeden text na každém řádku. Soubor může obsahovat i seskupení řádků, viz parametr `dataformat`. Parametr je typu `str` a obsahuje cestu k souboru, či přímo typu `file`.

`filter` Parametr `filter` slouží k načtení stoplistu, respektive seznamu termů, které mají být vynechány při zpracování. První možností, jak zadat parametr je souborem typu `file`, nebo cestou. Soubor samotný by měl mít na každém řádku jeden term. Druhým způsobem je předání přímo seznamu termů. Výchozí hodnotou je `None`, a tedy žádná filtrace neprobíhá.

`dataformat` V případě, že je formát data typu `str`, potom je nutné určit, zda se jedná o text nebo o soubor s textem. Pro text je třeba zadat `"text"`, v opačném případě je třeba dále specifikovat rozložení v n-tici řádků. N-ticí je myšleno, že pro $i = 1..||D||, j = 1..N$ je každý (i)tý řádek obsahuje stejný charakter dat, $(i + j)$ pak obsahjí jiný charakter dat. Pro $N = 2$ sudé řádky obsahují např. texty, liché ID dokumentu.

Řádky souboru můžou mít dva charaktery - `"data"`, kdy řádek obsahuje dokument; a `"topic"`, kdy řádek obsahuje téma sloužící pro vyhodnocení pro testování. Parametr `dataformat` pak musí obsahovat seznam těchto kódů, např. `("data","topic")`.

`filterformat` Slouží především k rozlišení textu obsahujícího filtrované termy od názvu souboru. Pro první případ nastavte `"file"`, v opačném případě `"text"`. Výchozí hodnotou je `"text"`.

`topics|` Slouží k označení dokumentů k testování výsledků. Délka seznamu `topics` musí být být shodné délky. Kontrola formátu parametrů je kontrolována funkcí `assert` a při jakékoli nesrovnalosti ve formátu je vyhozena výjimka `AssertionError`. shlukování.

Kapitola 4

Výsledky

V této kapitole budou shrnuty dosažené výsledky různých metod s různým typem přípravy příznaků. Budou zde porovnány nejen průměrné F-míry, ale také průměrné trvání běhu programu v testovacím prostředí, to vše na různých testovacích množinách dokumentů. Měřeny byly dvě veličiny, a to F-míra popsaná v části 4.3, která vypovídá, jak moc jsou vytvořené shluky odpovídají těm přirozeným.

Druhou veličinou je pak trvání běhu algoritmu. Trvání lze rozdělit na dvě části: na tvorbu příznaků, kdy se převádí text na číselné vektory, ať už pomocí TF-IDF, nebo je následně přepočítán pomocí SVD na LSA. Druhá část je trvání samotného shlukování. Důvod k rozdělení na dvě části je, že s jednou vypočítanými příznaky lze spustit běh několikrát, takže při dávkovém režimu je ulehčen výpočet.

Při sledování těchto dvou veličin pak lze sledovat výkon a při podobných výsledcích v oblasti F-míry se lze rozhodnout podle času a obráceně.

Kromě shlukování je třeba také vyhodnotit úspěšnost klasifikace dokumentů. To probíhá také pomocí F-míry s drobným rozdílem, který je též popsán v sekci 4.3.

4.1 Testovací dokumenty

Dokumenty použité k testování příznaků a metod byly vyexportovány z projektu JMZW a původně pochází z internetových zpravodajských serverů. Pro testovací účely byly rozděleny do několika množin. Ty mají různý počet témat v rozsahu 3-7, dokumentů 60-525, rozsah počtu slov je pak přibližně 6-15 tisíc, detaily jsou rozepsány v tabulce **??**. Pro účely klasifikace jsou, jak bylo naznačeno, množiny rozděleny na *trénovací množinu* a *testovací množinu*.

K dokumentům byla za účelem testování také přiložena témata. Pro množiny 1 a 2 bylo každému dokumentu přiřazeno právě jedno téma, ve zbylých dvou množinách to bylo alespoň jedno téma.

Pro názornost nahlédněme do množiny 4. Témata obsažená jsou následující: akcie, silnice, vláda, školy, hudba, tenis a počasí. Z těch je vidět, že některá témata k sobě mají

Množina	K	$\|S\|$	Velikost prostoru	Rovnoměrné rozložení	Fuzzy	Způsob předzpracování	Trén.	Test.
množina 1	3	60	5993	ano	není	filtrace termů	70	10
množina 2	4	129	11801	ne	je	filtrace termů	109	20
množina 3	5	145	9254	ne	není	filtrace termů	54	6
množina 4	7	525	15105	ano	je	lemmatizace i filtrace termů	471	54

Tab. 4.1: Přehled vstupních množin dokumentů

sobě velmi blízko. Lze snadno nalézt příklad, kdy lze dokument zařadit do více než jednoho z témat. Větě „Ministerstvo financí schválilo nový obchvat měst Plzně.“ by zřejmě mělo být přiřazeno téma silnice, stejně tak, jako vláda.

4.2 Klasifikace

Cílem této práce je najít nejvhodnější skupinu dokumentů, které by co nejlépe posloužily k adaptaci jazykového modelu. Natrénované množiny jsou připravené ke klasifikaci. Po obdržení dokumentu ke klasifikaci ho musíme převést do daného vektorového prostoru, ve kterém byly shluky natrénovány. Teprve potom lze přistoupit k samotné klasifikaci:

V zásadě jsou možné dva přístupy. První je, že se využije centroidů, a je vybrána třída dokumentů, která k němu spadá. Druhým je pak použití metody k-nejbližšího souseda, která by vzhledem k charakteru shlukovacích metod měla vracet shodné výsledky.

4.3 Způsob vyhodnocení

K vyhodnocování výsledných shluků slouží F-míra, která se v oboru hodnot $< 0,1 >$ vyjadřuje, jak moc vytvořené shluky odpovídají shlukům přirozeným. Toho se dosahuje s pomocí tabulky výskytu, příkladem je 4.2.

	Téma 1	Téma 2
Shluk 1	14	4
Shluk 2	5	16
Shluk 3	1	2

Tab. 4.2: Ukázka tabulky výskytu

V tabulce 4.2 je vidět, že nebylo správě zvolené K, protože jsou zde pouze dvě témata, ale shluky jsou tři. Dále je z tabulky vidět, že shluk 1 nejlépe odpovídá tématu 1, a shluk 2 tématu 2. Shluk 3 je pak chybou.

F-míra se harmonickým průměrem (vz. 4.1) dvou veličin, a to přesnosti a úplnosti. Přesnost vyjadřuje, jak moc shluk odpovídá přiřazenému tématu, úplnost pak jak moc shluk postihl veškeré dokumenty odpovídající danému tématu. Přesnost P a úplnost R jsou popsány ve vzorcích 4.2 a 4.3.

$$F = 2 \cdot \frac{P + R}{P \cdot R} \tag{4.1}$$

$$P = \frac{\text{správě zařazené}}{\text{správně zařazené} + \text{nesprávně zařazené}} \tag{4.2}$$

$$R = \frac{\text{správně zařazené}}{\text{správně zařazené} + \text{nesprávně nezařazené}} \tag{4.3}$$

Pro vysvětlení lze pak dodat, že v ukázkové tabulce 4.2 pro shluk 1 správně zařazených 14 dokumentů, nesprávně zařazené jsou 4 a nesprávně nezařazených je 6 (5 bylo zařazeno do shluku 2; jeden pak do shluku 3).

Výsledky klasifikace jsou vyhodnocovány podnebným způsobem, jako vyhodnocení shluků, s drobnou úpravou: Při vyhodnocování shluků jako výsledku shlukování je k vyhodnocení předloženo zpravidla více shluků, a je tak nutné přiřadit pro testovací účely poskytnutá témata k jednotlivým shlukům a pomocí tohoto přiřazení se počítá přesnost a úplnost.

V případě klasifikace se předpokládá, vybraný shluk nejbližší klasifikovanému dokumentu je stejného tématu, a přesnost se vypočítá se zařazeným klasifikovaným dokumentem.

4.4 Parametry zvolených metod

Pro testování typů příznaků byly parametry metod, u nichž je možné zvolit počet shluků (K-means) byl zadán explicitně. U ostatních, které dokáží navrhnout počet shluků samy (Mean-Shift), jim byla tato výsada ponechána. V tabulce C.1 jsou popsány jednotlivé parametry a hodnoty. Většina metod při předání parametru `debug` nastaveného na `True` vypisuje do konzole průběh často s průběhem kritéria napříč iteracemi. Jedinou výjimkou oproti standardním parametrům byla metoda ohybu, kde byl nastaven parametr `kesthi` na hodnotu 20.

4.5 Interpretace výsledků

Tabulka D.1 je uskupená po částech, které spolu více souvisí, tedy jednotlivé metody jsou v řádkách přehledně u sebe v rámci testovacích množin. Sloupce pak odpovídají jednotlivým příznakovým prostorům využitých ke shlukování. Hodnoty jsou pak průměrnými

F-mírami jednotlivých shluků zprůměrované přes více běhů, jak shlukovacích, tak příznakových.

Jednotlivé řádky obsahují jednopísmenné zkratky metod: K pro K-means, A pro vylepšené K-means, B pro bisecting K-means, R pro Refined K-means, F pro ICA (FastICA), N pro ICA bez explicitně zadaného parametru K, M pro mean-shift, E pro metodu ohybu (Elbow Method).

4.5.1 Příznakové metody

Nejprve porovnejme výsledky jednotlivých příznakových prostorů. Srovnáme-li F-míry získané metodami na první množině, mezi TF-IDF a n-gramy jsou velmi malé rozdíly, a ani jedna z příznakových metod není výrazně lepší. Metody B a R nad trénovací množinou vynikají nad if-idf. V prostoru LSA pak vynikají především výsledky metody A, a to na obou množinách. Zbylé metody lehce se zdají být spíš podprůměrné.

Na množině číslo 2 nejsou vidět rozdíly v hodnotách mezi jednotlivými příznakovými prostory, zatímco na množině 3 jsou patrné rozdíly především v metodě A, která oproti ostatním příznakovým prostorům vyčnívá. LSA ve většině výsledků alespoň dosahuje výsledků, které vrací TF-IDF.

4.5.2 Shlukovací metody

V rámci první množiny dosahuje dobrých výsledků metoda A při použití LSA. Kromě oblasti n-gramů je metoda A kvalitativně lepší, než K. Naopak B při použití s n-gramy dosahuje dobrých výsledků.

Ve druhé množině zcela jednohlasně vyčnívá metoda N, která ve většině případů přesahuje výsledky vesměs o dvojnásobek a to nad hranicí $0,8$, této hodnoty však nabývá z důvodu, že identifikoval K jako 1. Jako jediný shluk má tedy tato metoda menší chybu, než při zadání správné hodnoty.

Ze zbylých metod pak nad ostatní vyčnívá pouze R spolu s A, které se pohybují kolem hranice $0,3$. Obecně nízké hodnoty pak naznačují špatnou separovatelnost přírodních shluků, což odpovídá tématům v dané množině: počasí, doprava, nehoda a zdraví. Z těch je cítit, že jsou velmi příbuzná.

V množině 3 dosahuje velmi dobrých výsledků M. Pouze A při použití s variantami LSA dosahuje podobných výsledků.

4.5.3 Volba K

V tabulce D.3 je zobrazen přehled průměrné volby K, které provedly metody, které tuto schopnost mají. Je vidět, že metoda N velmi často volí K=1. V případech, kdy zvolí K větší, vybere takové K, které přibližně odpovídá skutečnému K s rozdílem přibližně 1. Metoda M volí K zpravidla vyšší, než je skutečné K, bylo by tedy vhodné ladit parametry

`c`, případně zvolit jiné `knn`. I metoda E parametr K nadhodnocuje, a pro dosažení lepších výsledků by bylo vhodné parametry lépe naladit.

4.5.4 Klasifikace

V tabulce D.4 je přehled, jak úspěšná byla klasifikace samotných testovacích dokumentů. Tabulka obsahuje průměrnou F-míru, která byla zprůměrována přes různé běhy a přes celou testovací množinu pro každý běh. Ze zde uvedených F-hodnot je vidět, že na množině 1 klasifikace proběhla v příznakovém prostoru TF-IDF lépe

4.5.5 Časová náročnost

V tabulce D.2 je přehled průměrného trvání běhu algoritmu. Z tabulky je vidět, že n-gramový model je o řád pomalejší, než TF-IDF. Tato závislost vychází z dimenze vektoru, která je u n-gramového modelu vyšší. TF-IDF je hned o několik řádů pomalejší, než LSA. Rychlost LSA a její kombinace s n-gramem je shodná, protože je prostor pomocí SVD redukován na stejnou dimenzi.

Co se týče jednotlivých metod, v rámci TF-IDF a n-gramů jsou nejrychlejší F, respektive N, což je patrně zapříčiněno optimalizací v implementaci scikit. Ze zbylých metod je nejrychejší K pro svoji jednoduchost, A jako rozšířená verze K je pomalejší. Metoda B je iterativně používaná metoda K, proto je téměř o řád pomalejší (závisí na K a na parametru `iter`). Metoda R je upravenou verzí B s jedním během K navíc, což v součtu odpovídá skutečnosti. Metoda M je pomalejší z důvodu iterativního procházení středů, a hlednání pomocí k-NN. Na závěr E používá iterativně K v hojném počtu. proto je od dva řády pomalejší.

4.5.6 Časová náročnost přípravy

Obsahem tabulky D.5 jsou průměrné časy zpracování textového souboru a jeho převedení na vektory daného příznakového prostoru. Je vidět, že vytvoření TF-IDF je triviální úlohou. Vytvoření n-gramů a následné převedení do TF-IDF je časově náročnější, ale stále ne tak, jako vytvoření LSA vektorového prostoru. Trvání vytvoření kombinace n-gramů a LSA je mnohem náročnější, protože SVD rozkládá matici mnohem vyššího řádu.

V průběhu výpočtů se projevily neduhy některých algoritmů. Především paměťová náročnost maticových algoritmů způsobila v hardwarovém testovacím prostředí nestabilitu a nebylo tak možné některé kombinace algoritmů otestovat. Především se jednalo o LSA na velkých testovacích skupinách, jmenovitě 2 a 4. O to více byl algoritmus náročnější při použití s n-gramy, které počet termů rozšiřují exponenciálně, maximálně dle vztahu **??**. Nestabilita se projevila pády, které nebyly vysvětlena.

Možným důvodem může být i poněkud nestálá hardwarová testovací sestava, na které se v nepravidelných intervalech při větším zatížení snižuje stabilita a to má podobné projevy, jako pozorované pády.

4.5.7 Celkové zhodnocení

Příznaky

Příznakové metody založené mají nespornou výhodu v nízké náročnosti výpočtu samotného shlukování. Metody jsou pak vyhodnoceny za zlomek času oproti TF-IDF. Tato výhoda je ale vyvážena vysokou náročností přípravy prostoru. Tyto vektorové prostory mohou být s výhodou použity v případě offline zpracování, kde je možnost je uchovat pro pozdější použití.

Kombinace n-gram + LSA je zajímavá v teoretické rovině, v praktických experimentech však vesměs nedosahuje výsledků samotné LSA. Pro potvrzení výsledků by bylo zřejmě třeba podrobnější testování na větších množinách a pokus opakovat vícekrát. Běh metody je stejně náročný, jako v případě LSA, a to z toho důvodu, že je prostor redukován pomocí SVD na stejnou dimenzi. Náročnost přípravy je vyšší pro vyšší počet příznaků k redukci.

TF-IDF má oproti předchozím metodám velmi rychlou přípravu, ale náročnost výpočtu shlukování se zvyšuje s počtem unikátních slov v množině. Ve srovnání s LDA je výpočet o několik řádů náročnější. Kvalita shlukování je nepatrně vyšší, než v případě LDA. Proto výběr metody záleží na aplikaci.

N-gramový model, ačkoliv obsahuje oproti TF-IDF informace navíc, dosahuje výsledků pouze o něco nižších, než TF-IDF. A to i přesto, že příprava trvá podstatně déle.

Shlukovací metody

V první řadě je třeba rozlišovat mezi metodami, které umí samy zvolit počet shluků, a které nikoliv. Tato zásadní vlastnost uvolňuje předpoklad o znalosti shlukovaných dat.

Z metod, které nejsou schopny určit počet shluků samostatně vynikají především metody refined bisecting K-means a vylepšené K-means. Vesměs dosahují podobných výsledků, avšak náročnost vylepšených K-means je mnohem nižší. Na druhou stranu je ale mnohem náchylnější na počáteční podmínky, zatímco refined bisecting K-means je robustnější díky opakování pokusů v průběhu shlukování.

Metody, které volí K bez vnějšího zásahu vyniká spíše mean-shift, ačkoliv na menších množinách byly výsledky srovnatelné.

Kapitola 5

Napojení na stávající projekt

Tato práce je připravena na napojení na projekt týkající se adaptace jazykového modelu tak, že při poskytnutí textových dat spolu se vzorovým dokumentem, je schopna za pomocí vstupní knihovny připravit vektorové příznaky, z nichž za pomocí zvolené metody vytvoří shluky, a vrátí indexy dokumentů, které byly ve stejném shluku jako vzorový dokument.

Vzhledem k charakteru úlohy, kdy budou dokumenty předem předshlukované, časová náročnost přípravy příznaků nebude mít na volbu vhodného shluku vliv, protože budou moci být předpočítány v offline trénovacím režimu. Zbylá úloha by byla pouze klasifikace.

Byla-li by úloha koncipovaná jinak, například on-line přepočítávání jazykového modelu, bly by kladen požadavek na rychlost algoritmu. Zde proti sobě stojí dva protiklady: v případě příznaků *TF-IDF* je rychlá příprava, ale značně pomalejší shlukování, naproti tomu pak stojí příznaky ve formě *LSA*, kdy je příprava dat časově náročnější, výpočet samotný je však nesrovnatelně rychlejší díky redukci dimenze.

Budeme-li předpokládat, že se některé výpočty budou provádět vícekrát, například kvůli eliminaci vlivu počátečních podmínek nebo pro detekci počtu shluků, můžeme se tak spíše přiklonit k druhé verzi. V opačném případě můžeme spoléhat na jeden běh, a časová náročnost je přibližně v podobném rozmezí a můžeme se tedy rozhodnout dle posouzené kvality metod.

Klasifikace je prováděna tak, že se klasifikovaný dokument převede do shodného příznakového prostoru, jako připravená data, a porovná se se středy daných shluků. U toho, u kterého bude podobnost nejvyšší, do toho shluku s nejvyšší pravděpodobností dokument patří. Jiným způsobem klasifikace pak může být metoda k-NN, která je popsána v části 2.2.1.

Postup pro napojení je popsán na schématu A.1, které je popsáno v části 3.

Závěr

Vybrané metody byly otestovány v kombinaci s různými příznakovými prostory. Bohužel výzkum byl tak intenzivní, jak mohl být, ovšem ani základní algoritmy, které byly v této práci testované přinesly výsledky. Byly porovnány v různých prostředích a následně testovány pomocí F-míry. Poté byly množiny shluokvaných dokumentů rozděleny na trénovací a testovací. Následně proběhla klasifikace.

Vzhledem k charakteru cílové úlohy, tedy offline shlukování a online klasifikace, nemá náročnost přípravy takovou váhu, jako kdyby byla aplikace cílená na online zpracování včetně shlukování.

Protože příznakový prostor LSA, ať už samotný, či v kombinaci s n-gramem, redukuje dimenzi úlohy na velikost počtu dokumentů, urychluje tak samotné shlukování, a je možné provést více pokusů, a vybrat ten nejlepší dle stanoveného kritéria. Snížení dimenze prostoru pomáhá i rychlosti klasifikaci.

Testované shlukovací algoritmy, které mají schopnost odhadnout počet shluků, jsou vesměs schopny tak učinit a jisté toleranci tak určí správně. Ostatní metody potřebují znalost o datech, která nemusí být dostupná, a proto jejich síla spočívá spíše v podpoře ostatních algoritmů, jako například metodě ohybu, či hierarchickým metodám.

Při použití LSA příznakového prostoru se pak nadějně jeví i metoda ohybu, protože jsou jednotlivé výpočty značně usnadněny. To samé lze říci i o metodě mean-shift, protože i pro tu je zjednodušeno vyčerpávající hledání nejbližšího souseda.

Z metod, které nejsou schopny odhadnou K se nejlépe jeví vylepšené K-means spolu s refined K-means. První z nich má velmi rychlý průběh, je však náchylná na počáteční podmínky, druhá jmenovaná je robustnější, protože se v rámci výpočtů vybírá z vhodných iterací, a výsledek je navíc přepracován pomocí K-means, které vyhladí případné chyby na rozhraní shluků.

Pro napojení na stávající projekt adaptace jazykového modelu je tak nabídnutý nástroj, který je schopný poskytnuté dokumenty převést do příznakového prostoru, nashlukovat je, a pak klasifikovat předložený vzorový dokument a poskytnout dokumenty s podobným tématem.

Práci by bylo možno obohatit především ve směru hlubšího výzkumu shlukovacích metod, které nepotřebují a priori znalost o počtu shluků, případně o intenzivnější testování předložených metod.

Literatura

[1] Exploiting Latent Semantic Information in Statistical Language Modelling; Jerome R. Bellegarda, IEEE, 2000

[2] ICA and SOM in Text Document Analysis; Ella Bingham, Jukka Kuusisto, Krista Lagus; Neural Networks Research Centre, Helsinki University of Technology, Finland, 2002

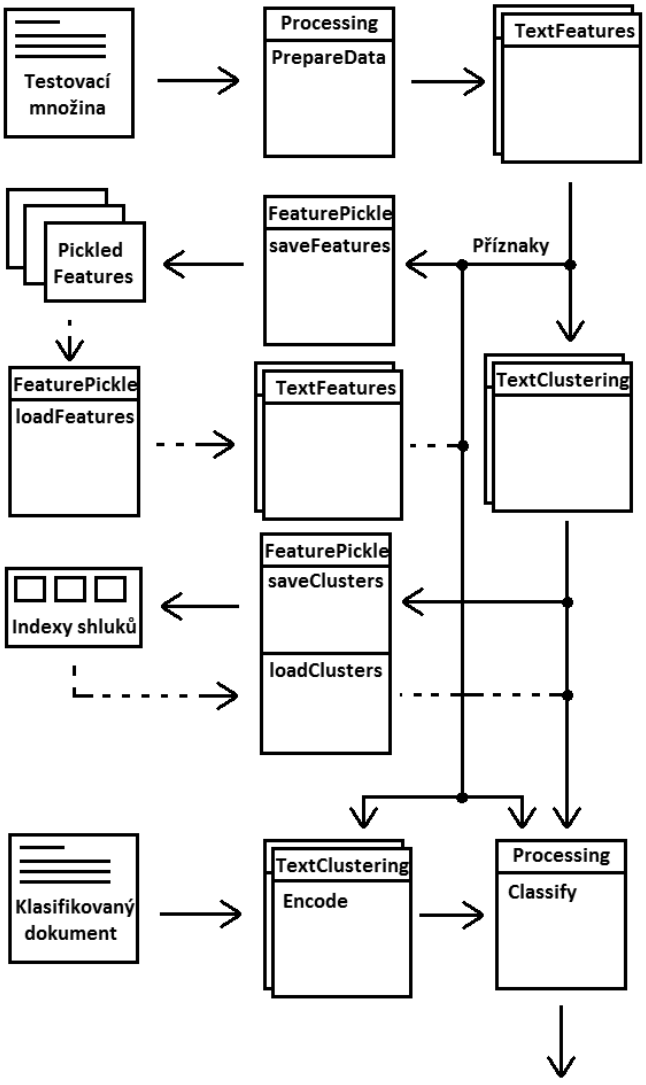
[3] Numerical methods for computing angles between linear subspaces; Ake Björck and Gene H. Golub; Math. Comp. 27 (1973), 579-594

[4] Projekt scikit-learn <http://scikit-learn.org/stable/auto_examples/decomposition/plot_ica_blind_source_separation.html>

[5] <http://en.wikipedia.org/wiki/Principal_component_analysis>

Příloha A

Schéma modulů



Obr. A.1: Schéma předávání dat mezi moduly

Příloha B

Obsah přiloženého datového média

Datový nosič obsahuje zdrojové soubory v modulech popsaných v sekci 3, které se nachází ve složce `src`. Vstupní a výstupní soubory jsou uloženy ve složkách `in`, respektive `out`, výstupní soubory jsou uspořádány do stromové hierarchie. Dále nosič obsahuje digitální verzi této práce ve formátu PDF.

Příloha C

Výchozí parametry metod

Variace K-means			
`epsilon`	=	0,03	Minimální posun centroidů
`iter`	=	5	Maximální počet iterací algoritmu

Variace bisecting K-means			
`epsilon`	=	0,03	
`iter`	=	3	
`method`	=	`None`	Podmetoda, která se použije při dělení

Mean-shift			
`epsilon`	=	0,03	Minimální posun středů
`iter`	=	20	
`knn`	=	3	Počet sousedů, podle kterého se počítá $N()$
`kepsilon`	=	0,001	Hodnota kernel function, pod níž se nehýbe středy
`filter`	=	0,0001	Min. vzdálenost pro odstraňování shodných středů

Mean-shift			
`eiter`	=	5	Počet iterací podmetody při výpočtu každého K
`method`	=	`None`	Podmetoda, jejíž se použije hodnota kritéria a výsledek
`kestlow`	=	2	Odhadovaná minimální hodnota K
`kesthi`	=		Odhadovaná[1] minimální maximální K

Tab. C.1: Přehled parametrů metod

Příloha D

Výsledky algoritmů

height	Příznaky	tf-idf		n-gram		LSA		n-gram + LSA	
Trénovací		Ne	Ano	Ne	Ano	Ne	Ano	Ne	Ano
Mn. 1	K	0.622	0.637	0.621	0.589	0.664	0.604	0.635	0.595
	A	0.631	0.64	0.557	0.58	0.782	0.761	0.741	0.783
	B	0.539	0.562	0.519	0.793	0.531	0.502	0.474	0.494
	R	0.698	0.681	0.616	0.841	0.598	0.673	0.65	0.58
	F	0.52	0.545	0.329	0.384	0.325	0.335	0.292	0.291
	N	0.453	0.541	0.5	0.541	0.5	0.541	0.5	0.541
	M	0.317	0.426	-	-	-	0.384	-	0.445
	E	0.341	0.514	-	-	-	0.597	-	0.434
Mn. 2	K	0.274	0.263	0.279	0.28	0.25	0.25	0.255	0.257
	A	0.308	0.292	0.33	0.324	0.287	0.308	0.326	0.299
	B	0.244	0.241	0.241	0.217	0.223	0.227	0.23	0.236
	R	0.32	0.268	0.239	0.26	0.277	0.242	0.27	0.266
	F	0.283	0.263	0.257	0.275	0.193	0.206	0.185	0.185
	N	0.816	0.832	0.816	0.832	0.816	0.832	0.816	0.832
	M	0.176	0.141	-	-	-	-	0.153	0.184
	E	0.166	0.309	-	-	-	-	0.148	0.162
Mn. 3	K	0.455	0.426	0.394	0.446	0.45	0.46	0.461	0.451
	A	0.422	0.391	0.399	0.439	0.638	0.603	0.584	0.584
	B	0.237	0.259	0.23	-	0.228	0.211	0.241	0.23
	R	0.453	0.445	0.413	0.378	0.457	0.332	0.429	0.438
	F	0.312	0.289	0.32	0.242	0.248	0.272	0.318	0.275
	N	0.333	0.353	0.352	0.353	0.352	0.353	0.352	0.353
	M	0.605	0.638	0.623	0.628	-	0.632	-	0.519
	E	0.317	0.337	-	-	-	0.392	0.3	0.326
Mn. 4	K	0.552	-	-	-	-	-	-	-
	A	0.398	-	-	-	-	-	-	-
	B	0.298	-	-	-	-	-	-	-
	R	0.571	-	-	-	-	-	-	-
	F	0.562	-	-	-	-	-	-	-
	N	0.32	-	-	-	-	-	-	-
	M	-	-	-	-	-	-	-	-
	E	-	-	-	-	-	-	-	-

Tab. D.1: Přehled průměrné F-míry

height	Příznaky	tf-idf		n-gram		LSA		n-gram + LSA	
Trénovací		Ne	Ano	Ne	Ano	Ne	Ano	Ne	Ano
Mn. 1	K	2.83	2.33	15.41	14.728	0.036	0.028	0.035	0.028
	A	4.296	3.678	23.87	22.227	0.12	0.088	0.111	0.09
	B	9.051	8.352	54.814	44.868	0.093	0.084	0.09	0.07
	R	11.994	10.432	68.072	59.675	0.113	0.091	0.109	0.089
	F	0.205	0.197	1.159	0.921	0.016	0.023	0.035	0.086
	N	0.778	0.651	1.862	1.373	0.445	0.393	0.49	18.425
	M	48.185	38.169	-	-	-	1.819	-	2.215
	E	226.98	199.895	-	-	-	4.894	-	4.501
Mn. 2	K	15.263	12.438	105.077	74.658	0.172	0.124	0.209	0.13
	A	22.751	20.679	247.052	102.11	0.547	0.41	0.557	0.401
	B	49.072	44.478	389.335	246.06	0.456	0.326	0.51	0.348
	R	65.313	52.827	458.271	316.045	0.564	0.41	0.655	0.463
	F	0.958	4.43	7.08	8.442	0.024	0.08	0.603	0.328
	N	3.317	6.386	10.467	9.657	2.57	1.36	115.631	3.288
	M	547.719	392.479	-	-	-	-	29.255	18.865
	E	952.225	721.377	-	-	-	-	12.54	9.687
Mn. 3	K	9.027	7.447	48.619	50.197	0.08	0.086	0.082	0.086
	A	11.618	10.771	64.332	62.56	0.255	0.225	0.232	0.243
	B	26.368	24.862	161.525	-	0.265	0.237	0.196	0.209
	R	35.883	30.491	215.687	205.293	0.292	0.263	0.281	0.625
	F	3.858	0.388	2.989	5.667	0.017	0.024	0.012	0.246
	N	17.031	1.092	4.141	123.655	0.685	0.67	0.814	65.211
	M	154.475	121.406	983.759	863.342	-	4.328	-	8.705
	E	435.504	385.604	-	-	-	6.401	5.778	7.6
Mn. 4	K	121.435	-	-	-	-	-	-	-
	A	176.657	-	-	-	-	-	-	-
	B	355.197	-	-	-	-	-	-	-
	R	470.823	-	-	-	-	-	-	-
	F	6.407	-	-	-	-	-	-	-
	N	88.706	-	-	-	-	-	-	-
	M	-	-	-	-	-	-	-	-
	E	-	-	-	-	-	-	-	-

Tab. D.2: Přehled průměrného trvání běhu

height	Příznaky	tf-idf		n-gram		LSA		n-gram + LSA	
Trénovací		Ne	Ano	Ne	Ano	Ne	Ano	Ne	Ano
Mn. 1	N	4.2	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	M	6.0	5.0	-	-	-	5.0	-	5.0
	E	7.0	4.4	-	-	-	3.375	-	3.5
Mn. 2	N	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	M	8.0	9.0	-	-	-	-	9.0	7.0
	E	6.0	5.5	-	-	-	-	8.833	7.5
Mn. 3	N	3.059	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	M	6.0	6.0	6.0	6.0	-	6.0	-	7.0
	E	3.0	5.3	-	-	-	5.2	4.333	4.538
Mn. 4	N	1.0	-	-	-	-	-	-	-
	M	-	-	-	-	-	-	-	-
	E	-	-	-	-	-	-	-	-

Tab. D.3: Přehled průměru odhadnutého parametru K

height	Příznaky	tf-idf	n-gram	LSA	n-gram + LSA
Mn. 1	K	0.716	-	0.64	-
	A	0.895	-	0.867	-
	B	0.617	-	0.526	-
	R	0.733	-	0.728	-
	F	0.575	-	0.434	-
	N	0.553	-	0.553	-
	M	0.738	-	0.615	-
	E	0.659	-	0.717	-
Mn. 2	K	0.393	-	0.385	-
	A	0.765	-	0.366	-
	B	0.382	-	0.662	-
	R	0.406	-	0.331	-
	F	0.434	-	0.248	-
	N	0.831	-	0.831	-
	M	0.302	-	-	-
	E	0.457	-	-	-
Mn. 3	K	0.505	-	0.512	-
	A	0.495	-	0.658	-
	B	0.351	-	0.352	-
	R	0.513	-	0.344	-
	F	0.403	-	0.355	-
	N	0.353	-	0.353	-
	M	0.683	-	0.612	-
	E	0.484	-	0.482	-
Mn. 4	K	0.563	-	-	-
	A	0.594	-	-	-
	B	-	-	-	-
	R	-	-	-	-
	F	-	-	-	-
	N	-	-	-	-
	M	-	-	-	-
	E	-	-	-	-

Tab. D.4: Přehled průměru F-míry klasifikace

Příznaky	TF-IDF		n-gram		LSA		n-gram + LSA	
Trénovací	Ne	Ano	Ne	Ano	Ne	Ano	Ne	Ano
Mn. 1	-	2.258	-	83.702	-	34.293	-	263.28
Mn. 2	-	14.382	-	904.886	-	319.187	-	2566.832
Mn. 3	-	8.567	-	157.077	-	100.749	-	700.918
Mn. 4	-	124.961	-	-	-	-	-	-

Tab. D.5: Přehled trvání převedení do vektorového prostoru