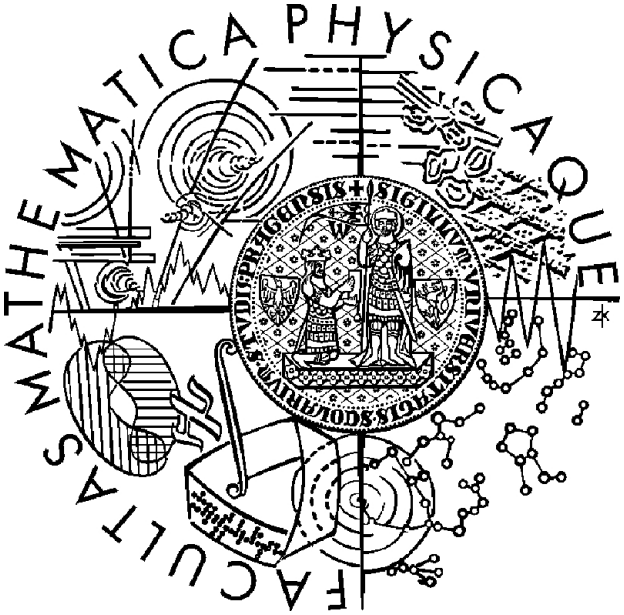
UNIVERZITA KARLOVA V PRAZE
MATEMATICKO-FYZIKÁLNÍ FAKULTA

DIPLOMOVÁ PRÁCE



Jiří Diviš

Shluky silně podobných textů

Ústav formální a aplikované lingvistiky

Vedoucí práce:	RNDr. Martin Holub Ph.D.
Studijní program:	Informatika
Studijní obor:	Počítačová lingvistika

Poděkování

Na tomto místě bych rád poděkoval svému školiteli RNDr. Martinu Holubovi Ph.D. za cenné rady při psaní této diplomové práce. Dále děkuji RNDr. Aleně Piroutkové za její trpělivost a podporu.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 5.12.2006 Jiří Diviš

Obsah

Název práce: Shluky silně podobných textů
Autor: Jiří Diviš
Katedra (ústav): Ústav formální a aplikované lingvistiky
Vedoucí práce: RNDr. Martin Holub Ph.D.
E-mail vedoucího: holub@ufal.mff.cuni.cz
Abstrakt: Práce se věnuje automatizovanému hledání shluků tématicky podobných textových dokumentů v rozsáhlých textových kolekcích. V práci je navržen algoritmus pro nalezení těchto shluků a metoda pro optimalizaci jeho parametrů pomocí strojového učení. Byla provedena implementace a experimentální ověření funkčnosti navrženého postupu. Pro evaluaci je využita ručně anotovaná kolekce českých dokumentů obsahující množinu vzorových shluků a dále obsáhlá kolekce novinových článků. Provedené experimenty ukazují, že výstupem navrženého algoritmu jsou požadované shluky tématicky podobných textů.
Klíčová slova: konceptotvorný shluk, metody shlukové analýzy, kvalita shlukování, porovnání shlukování

Title: Clusters of strongly similar documents
Author: Jiří Diviš
Department: Institute of Formal and Applied Linguistics
Supervisor: RNDr. Martin Holub Ph.D.
Supervisor's e-mail address: holub@ufal.mff.cuni.cz
Abstract: This thesis focuses on automatic searching for clusters of topically similar texts in large text collection. We introduce an algorithm for finding the clusters and a method of optimizing its parameters using machine learning techniques. The algorithm is implemented and experimentaly evaluated. For evaluation we use a manually annotated collection of Czech documents, which contains a set of sample clusters chosen and tagged by a human annotator, and a huge collection of newspaper arcticles. Experiments show that the output of our algorithm fulfils our expectation and gives clusters of topically similar texts.
Keywords: concept-formative cluster, cluster analysis methods, clustering quality, clusters comparing

Kapitola 1

Úvod

Jednou z metod, jak z dat získat požadovanou informaci, je nad danými daty (objekty) provést shlukování. Shlukování dokumentů je důležitou oblastí z teorie získávání informací. Shlukování je cestou, jak objevit vnitřní strukturu kolekce objektů, případně cestou jednoduššího a kratšího popisu objektů, z nichž je kolekce složena. Kvůli velkým objemům dat a obyčejně velkému počtu objektů v kolekci je snahou nalézt rychlý a efektivní algoritmus pro provedení shlukování.

Tato práce vychází z článku [1], který se věnuje automatickému hledání shluků silně podobných textových dokumentů v rozsáhlé kolekci textů. Dokumenty jsou nestrukturalizované texty v přirozeném jazyce. V článku byly definovány požadované vlastnosti hledaného shluku. V této práci navrhneme algoritmus pro nalezení takto definovaného shluku a určení míry incidence každého dokumentu k tomuto nalezenému shluku. Tento algoritmus implementujeme a experimentálně ověříme, zda takto automaticky nalezené shluky jsou shluky silně podobných dokumentů. Pro tyto experimenty použijeme dvě kolekce dokumentů. Menší z těchto kolekcí je ručně anotovaná a jsou v ní anotátorem vyznačeny shluky, které budeme považovat za vzorové. Naším cílem bude dosáhnout maximálního možného přiblížení výstupu navržného algoritmu k těmto vzorovým shlukům. Pro porovnání účinnosti nadefinované metody porovnáme výsledek navrženého algoritmu s výsledkem známého algoritmu K-means. Druhá kolekce je obsáhlá databáze novinových článků. Na této rozsáhlé kolekci ověříme, zda navržený algoritmus a jeho výstup má pro velká vstupní data stejné vlastnosti jako na malé anotované kolekci.

Práce je rozdělena do pěti základních částí:

- Základní přehled a rozdělení metod používaných pro shlukování.

- Definice požadovaných vlastností hledaného shluku a definice postupu pro jeho automatické nalezení v kolekci textů.
- Analýza vstupních kolekcí a jejich vzájemné porovnání. Předzpracování těchto kolekcí a spočítání vzájemných podobností všech dokumentů.
- Natrénování nejlepších možných parametrů navrženého algoritmu.
- Experimetální nalezení shluků na daných kolekcích a ověření jejich vlastností.

Práce je ukončena závěrem, v němž jsou shrnuty výsledky experimentů a nastíněny další možné cesty vývoje, zlepšení a použití navrženého algoritmu pro hledání shluků daných vlastností. Všechny použité programy a data (kolekce českých textů) jsou přiloženy na DVD. Jeho obsah je popsán v příloze. Tam se také nachází stručná uživatelská dokumentace k programům potřebných pro provedení experimentů.

Kapitola 2

Shluková analýza

Shlukování má za úkol roztřídit množinu objektů do menších celků podle vhodných charakteristických rysů. Hlavní obtíží, na kterou narazíme, budeme-li chtít takovou situaci matematicky modelovat, je mimo jiné nutnost klasifikovat zkoumané věci a jevy na základě jejich podobnosti. Protože posuzování vzájemných podobností bylo a je záležitostí diskutabilní, začaly se s rozvojem výpočetní techniky objevovat pokusy o matematické podchycení a vyhodnocení míry těchto podobností. Hledaly se vhodné metody číselného vyjádření vlastností, jimiž se zkoumané objekty vyznačovaly, metody kvantitativního vyjádření podobností takto zakódovaných objektů a konečně metody seskupování podobných objektů do „shluků“. Tak začaly pokusy s řadou metod, které dnes podle společného cíle zahrnujeme pod název **shluková analýza** nebo zkráceně **shlukování**.

Známý příklad využití shlukové analýzy je nalezení bílých trpaslíků a červených obrů v astronomii. Hvězdy byly popsány roku 1913 astronomy Hertzsprungem a Russellem pomocí dvou hodnot - svítivosti a teploty. Následně byly pomocí shlukové analýzy roztříděny. Vznikl tak v astronimii známý H-R diagram zobrazující tři shluky: bílé trpaslíky, červené obry a hlavní sekvenci hvězd mezi nimi (viz tab. 2.1).

HR diagram	**Teplota, T**	**Svítivost**
Bílý trpaslík	střední	nízká
Hlavní sekvence	široký rozsah hodnot	nízká pokud je T vysoká, vysoká pokud je T nízká
Červený obr	středně nízká	vysoká

Tab. 2.1: Rozdělení hvězd shlukovou analýzou

2.1 Metody shlukové analýzy

O metodách shlukové analýzy bylo napsáno mnoho monografií. V této kapitole vycházíme zejména z [2], [3].
Procedura shlukování by měla v ideálním případě splňovat dva cíle: korektnost a efektivitu. Hlavním kritériem efektivity je čas potřebný k shlukování. Kritéria pro korektnost jsou:

- stabilita při růstu kolekce, tj. rozdělení do shluků by se nemělo příliš měnit při přidávání nových dokumentů
- malé chyb při popisu dokumentů by měly vést k malým změnám v rozdělení do shluků
- nezávislost na počátečním uspořádáni dokumentů

V literatuře se objevuje takové množství shlukovacích metod, že je obtížné je nějak rozumně utřídit. Protože shluková analýza má sloužit jako prostředek k získání klasifikace, nabízí se možnost rozlišovat shlukovací metody nikoli podle použitých matematických prostředků, ale podle cílů, k nimž směřují. Takto rozlišujeme dvě základní skupiny metod shlukové analýzy, a to **hierarchické** a **nehierarchické metody**, z nichž první směřují k hierarchické klasifikaci, druhé k nehierarchické klasifikaci. Další významnou etapou v rozvoji shlukové analýzy je uplatnění fuzzy množin ve shlukovacím procesu. Tyto metody nazýváme fuzzy shlukovací metody nebo zkráceně **fuzzy shlukování**. Nejznámější a nejpoužívanější je fuzzy shlukování definované J.C. Bezdekem [4].

2.1.1 Hierarchické metody

Hierarchické shlukování se dělí na dva typy metod: **Aglomerativní metody shlukování** začínají s N shluky, kde každý shluk je tvořen právě jedním z N objektů. Na konci výpočtu je celá kolekce obsažena v jednom shluku. **Divizivní metody shlukování** pracují přesně opačně než aglomerativní metody. Začínají s jedním shlukem, který obsahuje všechny objekty z kolekce. Postupným dělením se tento shluk rozpadá na menší a menší shluky až na konci je každý objekt samostatným shlukem.

U obou druhů metod se vychází z toho, že dokážeme spočítat jak daleko je bod od nějakého shluku nebo dokonce spočítat vzdálenost mezi dvěma shluky. I tyto vzdálenosti lze definovat více způsoby a záleží jen na konkrétní aplikaci jaká definice je nejvýhodnější. Rozdělení hierarchických metod podle definice vzdálenosti mezi dvěma shluky:

1. single linkage ... nejkratší vzdálenost mezi objekty shluků
2. complete linkage ... největší vzdálenost mezi objekty shluků
3. average linkage ... průměrná vzdálenost mezi objekty shluků
4. median ... vzdálenost mezi centroidy shluků

K zobrazení výsledku hierarchického shlukování je možno použít **dendrogram**. Dendogram je strom zobrazující spojení jednotlivých shluků v průběhu shlukování. Každý uzel tohoto stromu představuje shluk.

Nevýhodou hierarchických metod je nevratnost kroků. Poté, co se dva shluky spojí v jeden (resp. rozdělí na dva), nedají se už opět rozdělit (resp. spojit), i přestože se struktura v kolekci mohla změnit tak, že by to bylo vhodnější. Výhodou je, že není nutná žádná znalost o struktuře nebo složení vstupní kolekce.

2.1.2 Nehierarchické metody

Nehierarchické procedury pracují s předem určeným počtem skupin, do nichž chceme objekty roztřídit. Jednotlivé kroky algoritmu pak spočívají ve změnách obsahu skupin, až se docílí optimálního stavu. Pracuje se přitom se zástupci jednotlivých skupin, které se nazývají centroidy.

Nevýhodou nehierarchických metod oproti hierarchickým je nutnost znát alespoň částečně strukturu vstupující kolekce - počet shluků, která budou výstupem.

Algoritmus K-means

Algoritmus k-means je nejznámější algoritmus shlukové analýzy. Má mnoho obměn, ale poprvé byl uveřejněn v roce 1967 MacQeenem [5]. Než algoritmus popíšeme, musíme nadefinovat několik základních pojmů.

Pro libovolný shluk C_i definujeme **střední hodnotu shluku** μ_i jako:

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x \tag{2.1}$$

Dále definujeme **rozptyl uvnitř shluku** C_i:

$$J_i = \sum_{x \in C_i} \|x - \mu_i\|^2 \tag{2.2}$$

Nechť c označuje počet shluků, pak pro množinu shluků $\{C_i\}_{i=1}^{c}$ definujeme **kvalitu rozdělení** do shluků:

$$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{x \in C_i} \|x - \mu_i\|^2 \tag{2.3}$$

Algoritmus K-means spočívá v hledání minima kritéria (2.3) střídavě vzhledem k vektorům x a k vektorům středních hodnot μ_i. Předpokládáme, že známe počet shluků c. Vycházíme z počáteční volby vektorů středních hodnot $\mu_i(0)$. Kritérium J bude nabývat minima, jestliže každý vektor x je zařazen do shluku C_i, jehož vektor střední hodnoty $\mu_i(k)$ dává nejmenší hodnotu $\|x - \mu_i(k)\|^2$. Postup opakujeme dokud neplatí $\mu_i(k+1) = \mu_i(k)$. Neformálně řečeno K-means algoritmus aproximuje množinu dat vhodně zvolenými vektory.

Algoritmus K-means je popsán schématem 2.1.

Problémů algoritmu K-means je několik:

1. Končí v lokálním extrému.
2. Je závislý na počtu a volbě počátečních odhadů.
3. Prvek je přiřazen vždy pouze do jednoho shluku.

Právě poslední nevýhodu algoritmu K-means se snaží napravit algoritmus fuzzy K-means, který je také nazýván C-means.

Vstup:
Kolekce C, počet shluků c
Výstup:
Prvky z C rozdělené do shluků

Algoritmus:

1. Zvol c iniciálních středních hodnot $\mu_i(0)$, $k = 0$
2. Každý dokument x z kolekce C přidej do shluku C_i s minimální hodnotou $\|x - \mu_i(k)\|^2$
3. Spočítej nové střední hodnoty shluků $\mu_i(k+1)$
4. Pokud $\mu_i(k+1) <> \mu_i(k)$ pokračuj bodem 2

Obr. 2.1: Algoritmus K-means

Algoritmus C-means

Algoritmus K-means rozděluje vstupní data na pevný počet disjunktních shluků. Každý prvek přitom může být přiřazen pouze do jednoho shluku. Fuzzy shlukování umožňuje přiřadit jeden prvek do více shluků a tomuto přiřazení určit stupeň náležení. Fuzzy c-means (FCM) je metoda, kterou poprvé popsal Dunn v roce 1973 [6] a kterou zlepšil Bezdek v roce 1981 [4].
Stejně jako alg. K-means je i algoritmus C-means založen na minimalizaci kriteriální funkce (2.4), ovšem v té nyní přibyl člen určující míru náležení do shluku.

$$J_m = \sum_{i=1}^{n}\sum_{j=1}^{c} u_{ij}^m \|x_i - \mu_j\|^2 \quad , \quad 1 < m < \infty \tag{2.4}$$

m je libovolné reálné číslo větší než 1 určující míru fuzzy rozdělení [7] a u_{ij} je stupeň náležení prvku x_i do shluku C_j. Fuzzy rozdělení množiny prvků je pak získáno iterativní

optimalizací kriteriální funkce (2.4) s postupným zlepšováním hodnot náležení u_{ij} a center shluků c_j:

$$u_{ij} = \left[\sum_{k=1}^{c} \left(\frac{\|x_i - \mu_j\|}{\|x_i - \mu_k\|} \right)^{\frac{2}{m-1}} \right]^{-1} \quad , \quad \sum_{i=1}^{c} u_{ij} = 1 \tag{2.5}$$

$$\mu_j = \frac{\sum_{i=1}^{n} (u_{ij})^m x_i}{\sum_{i=1}^{n} (u_{ij})^m} \tag{2.6}$$

Tato iterace je ukončena pokud $\sum_{i=1}^{c} \mu_i(k) - \mu_i(k-1) \ < \epsilon$, kde ϵ je kritérium pro ukončení s hodnotou mezi 0 a 1 a k je označuje číslo kroku algoritmu. Procedura c-means, stejně jako k-means konverguje do lokálního minima. Formální zápis algoritmu je popsán na schématu 2.2.

Vstup:
kolekce C, c určující počet shluků a $\epsilon \in [0, 1]$
Výstup:
prvky z C rozdělené do shluků

Algoritmus:

0. $k = 0,\ U^{(k)} = [u_i j],\ u_{ij} = 0$
1. Pokud $k = 0$ zvol c iniciálních středních hodnot $\mu_i(0)$, jinak spočítej nová $\mu_i(k)$ podle vzorce (2.6)
2. Spočítej nové hodnoty $U^(k + 1)$ podle vzorce (2.5)
3. Pokud $\sum_{i=1}^{c} |\mu_i(k + 1) - \mu_i(k)| > \epsilon$ pokračuj krokem 1.

Obr. 2.2: Algoritmus C-means

Kapitola 3

Hledání konceptotvorného shluku

Definice konceptotorného shluku i použité značení bylo poprvé publikováno v článku [1]. Cílem této kapitoly je použít tuto definici a navrhnout postup pro automatizované nalezení takto definovaného shluku. Dříve než přistoupíme k vlastní definici konceptotvorného shluku, nadefinujeme si používané značení.

3.1 Značení

Nechť $G = (V, E)$ je hranově ohodnocený graf, kde V je množina vrcholů, $|V| = N$, $E = \{e_{ij} = (v_i, v_j) | v_i, v_j \in V\}$ je množina hran a $w(e_{ij}) \in < 0, 1 >$ je ohodnocení hrany e_{ij}. Pak pro libovolnou množinu vrcholů $A, B \subset V$, libovolný vrchol $v \in V$ a libovolnou množinu hran $H \subset E$ zavádíme následující značení:

$e(A) = \{e_{ij} \,|\, e_{ij} \in E \wedge v_i, v_j \in A\}$
– množina všech hran mezi vrcholy z A

$Con(A, B) = e(A \cup B) \setminus (e(A) \cup e(B))$
– množina hran mezi vrcholy z A a B (propojení množin A a B)

$Y(A) = \{v \in (V \setminus A) \,|\, deg(v, A) > 0\}$
– množina vrcholů z doplňku A ve V, ze kterých vede alespoň jedna hrana do A

$R(A) = V \setminus (A \cup Y(A))$
– množina vrcholů z doplňku A ve V, ze kterých nevede žádná hrana do A

$w(H) = \sum_{e \in H} w(e)$
– součet ohodnocení všech hran množiny H (váha H)

$w(A) = w(e(A))$
– součet ohodnocení všech hran mezi vrcholy z A (váha A)

$\overline{w} = w(V) / \binom{N}{2}$
– očekávaná velikost ohodnocení hrany v grafu G

$\overline{w}(A) = \overline{w} \cdot \binom{|A|}{2}$
– očekávaná váha množiny A

$\overline{w}(A, \overline{A}) = \overline{w} \cdot |A| \cdot (N - |A|)$
– očekávaná váha propojení množiny A a zbytku grafu G

$Neigh(v) = \{x \in V | (v, x) \in E\}$
– množina vrcholů, z nichž vede hrana do v (okolí vrcholu v)

Mějme kolekci C obsahující N dokumentů. Pro každou dvojici dokumentů d_1, d_2 z C definujeme hodnotu $sim(d_i, d_j) \in < 0, 1 >$. Potom graf $G = (V, E)$ je **graf kolekce** C, kde $V = \{d_i | d_i \in C\}$ a $E = \{(d_i, d_j) | d_i, d_j \in C \ sim(d_i, d_j) \geq PP\}$, kde PP je **prahová podobnost** udávající minimální akceptovatelnou váhu hrany v grafu.

3.2 Konceptotvorný shluk

V definici vycházíme z práce [1]. Libovolnou podmnožinu X dokumentů kolekce C nazveme shlukem. Nechť $Q(X)$ je nezáporná funkce, která pro shluk X definuje jeho kvalitu. Čím vyšší nabývá hodnoty, tím je shluk X kvalitnější. Shluk X nazveme **lokálně optimální shluk** vzhledem k funkci kvality Q, pokud každý shluk X', který vznikl z X pouze malou modifikací nemá větší kvalitu, tj. $Q(X') \leq Q(X)$.

Zvolme prahovou kvalitu shluku Q_t určující minimální akceptovatelnou kvalitu přijímaného shluku. Shluk X nazveme **konceptotvorný shluk** pokud $Q(X) \geq Q_t$ a X je lokálně optimální shluk.

Mějme nezáporné funkce $F_i(X)$ definující jednotlivé požadované vlastnosti shluku X. Čím vyšší nabývá funkce F_i hodnoty, tím více splňuje shluk danou vlastnost. Potom definujeme míru konceptotvornosti nebo-li kvalitu Q shluku X jako

$$Q(X) = \prod_i F_i^{\lambda_i} \tag{3.1}$$

kde λ_i jsou váhy jednotlivých vlastností požadovaných od konceptotvorného shluku.
Jak se ukáže v kapitolách s výsledky kde jsou natrénovány ideální λ_i zvlášť pro každý vzorový shluk (viz kapitola 5.4.1), jsou ideální hodnoty λ_i lineárně závislé na velikosti shluku. Zvolme proto $\lambda_i = \lambda_{i0} + \lambda_{i1} \cdot |X|$. Upravený vzorec pro kvalitu shluku potom je

$$Q(X) = \prod_i F_i^{\lambda_{i0}+\lambda_{i1}|X|} \tag{3.2}$$

Dále mějme funkci $P(d, X) : C \times \mathcal{P}(C) \mapsto \mathbf{R}$, která definuje **potenciál dokumentu** d ke shluku X jako

$$P(d, X) = Q(X \cup \{d\}) - Q(X \backslash \{d\}) \tag{3.3}$$

Potenciál dokumentu d ke shluku X je kladný, pokud se po přidání dokumentu do shluku kvalita shluku zvýší. Potenciál je záporný, pokud se po přidání do shluku kvalita shluku sníží. Potenciál nám tedy pro libovolný dokument d určí jeho příspěvek k míře konceptotvornosti shluku X.

Definujme si nyní požadované vlastnosti konceptotvorného shluku. V článku [1] byly definovány dvě základní vlastnosti shluku, které lze volně popsat jako požadavek na velkou váhu shluku (velké hrany uvnitř shluku) a malou váhu propojení shluku a zbytku kolekce (ze shluku vedou pouze malé hrany). Tyto dvě vlastnosti byly nazvány **kompaktnost** a **exhaustivita** shluku.

3.2.1 Kompaktnost shluku

Kompaktnost shluku X (zn. $Com(X)$) je definována jako poměr váhy shluku X ku očekávané váze shluku X. Čím má shluk větší váhu (čím je $Com(X)$ větší), tím je kompaktnější (dokumenty jsou si více podobné). Poměr ku očekávané váze shluku nám umožňuje porovnat kompaktnosti dvou různě velkých shluků z různých kolekcí. Formální definice:

$$Com(X) = \frac{w(X)}{\overline{w}(X)} = \frac{2 \cdot w(X)}{\overline{w} \cdot |X| \cdot (|X| - 1)} \tag{3.4}$$

Zřejmě pro každý shluk X platí $Com(X) \geq 0$.

3.2.2 Exhaustivita shluku

Exhaustivita shluku X (zn. $Exh(X)$) je definována jako poměr očekávané váhy propojení X a $\overline{X}$ ku váze propojení shluku X a $\overline{X}$. Čím má shluk menší váhu propojení se zbytkem kolekce, tím je $Exh(X)$ větší (dokumenty ve shluku jsou od zbývajících dokumentů v kolekci zřetelněji tématicky oddělené). Poměr ku očekávané váze propojení nám stejně jako u kompaktnosti umožňuje porovnat exhaustivitu dvou různě velkých shluků z různých kolekcí. Formální definice:

$$Exh(X) = \frac{\overline{w}(X, \overline{X})}{w(X, \overline{X})} = \frac{\overline{w} \cdot |X| \cdot (N - |X|)}{w(Con(X, \overline{X}))} \tag{3.5}$$

Zřejmě pro každý shluk X platí $Exh(X) \geq 0$.

3.3 Algoritmus hledání konceptotvorného shluku

Řekneme, že Shluk X je **vhodný iniciální shluk**, pokud jeho malou modifikací vznikne konceptotvorný shluk.

Protože výpočet kvalit všech shluků z $\mathcal{P}(C)$ a z nich následné vybrání konceptotvorných shluků je výpočetně nezvládnutelné, budeme postupovat ve dvou krocích. Prvním krokem je nalezení co možná největší množiny vhodných iniciálních shluků z $\mathcal{P}(C)$. To je výpočetně zvládnutelné díky omezení počtu hran v grafu kolekce prahovou podobností a vhodným heuristikám. Druhým krokem je provedení lokální optimalizace nalezených vhodných iniciálních shluků podle potenciálu dokumentů a vybrání konceptotvorných shluků z výstupu této optimalizace.

Nechť $G = (V, E)$ je graf kolekce, S a S_{Init} jsou množiny shluků, X je shluk a X^* je konceptotvorný shluk vzniklý ze shluku X pouze malou modifikací, pak alg. hledání konceptotvorného shluku je popsán schématem 3.1.

Vstup:
graf kolekce $G = (V, E)$, procedura `FindInitClusters(k)` pro nalezení k iniciálních shluků a procedura `LocalSearch(`X`)` pro lokální optimalizaci iniciálního shluku X

Výstup:
konceptotvorné shluky C_1, C_2, $\cdots$, C_m

Algoritmus:

1. $S = \emptyset$
2. S_{Init} = výstup z `FindInitClusters(`k`)`
3. Pro každé $X \in S_{Init}$ spočítej X^* = `LocalSearch(`X`)`, $S = S \cup \{X^*\}$
4. Setřiď S podle kvality, tzn. $S = \{X_i^*\}_{i=1}^k$ tak, že $Q(X_i^*) \geq X_{i+1}^*$ pro $i = 1 \cdots k$
5. Zvol prahovou kvalitu Q_t a spočítej $m = max\{i|Q(X_i^*) \geq Q_t\}$
6. $C_i = X_i^*$ pro $i = 1, \cdots m$

Obr. 3.1: Algoritmus hledání konceptotvorného shluku

Volba prahové kvality Q_t:

- Q_t je zvolena předem jako konstanta
- Q_t je zvolena tak, aby výstupem bylo max. m konceptotvorných shluků
- Q_t je zvolena tak, aby se dosáhlo předem daného pokrytí kolekce konceptotvornými shluky

3.3.1 Heuristiky pro nalezení iniciálního shluku

Pro nalezení vhodných iniciálních shluků jsou implementovány dvě metody. První metoda zohledňuje pouze počet hran vedoucích z vrcholu v do shluku X. Nazvěme tento postup jako „Metoda max. zapojení“. Druhá metoda, ozn. „Metodou max. kompaktnosti“, počítá kompaktnost shluku X po přidání vrcholu v.

Metoda max. zapojení

Pro přidání vrcholu v do shluku X rozhoduje hodnota $|Con(\{e\}, X)|$, tj. jeho zapojenost do X. Přitom nezáleží na váze hran. Důležité je vhodné zvolení prahové podobnosti PP (minimální akceptovaná váha hrany v grafu - viz následující kapitola analýzy dat) tak, aby pro libovolné $v \in V : deg(v) \ll |N - 1|$.
Nechť $G = (V, E)$ je graf kolekce, v, z jsou vrcholy, K_i, O jsou množiny vrcholů, pak schéma 3.2 popisuje algoritmus hledání iniciálních shluků metodou maximálního zapojení.

Vstup:
graf kolekce $G = (V, E)$ pro prahovou podobnost PP.
Výstup:
k iniciálních shluků $C_i \subset V$, $i = 1 \dots k$.

Algoritmus:

1. i=1, zvol práh I_t
2. Najdi vrchol $v \in V$ s největším stupněm. $K_i = \{v\}$, $O = Neigh(v)$
3. Najdi vrchol $z \in O$ s největším stupněm.
4. Pokud $|Con(\{z\}, K_i)| > I_t$, pak $K_i = K_i \cup \{z\}$, $O = O \cup Neigh(z)$, $V = V \backslash \{z\}$, jinak $O = O \backslash \{z\}$.
5. Pokud $O \neq \emptyset$ pokračuj krokem 3.
6. Zapiš K_i na výstup, $i = i + 1$, pokud $V \neq \emptyset$ pokračuj krokem 2.

Obr. 3.2: Algoritmus metody max. zapojení

Volba prahu I_t:

- I_t je volena jako konstanta. Zaručíme tak nejmenší stupeň vrcholu v iniciálním shluku.
- $I_t = |K_i| \cdot p$, kde konstanta p určuje minimální přijímaný poměr počtu hran, které přibudou s novým prvkem, ku počtu prvků ve shluku X. Můžeme tak zaručit, že

libovolný vrchol shluku je spojen hranami o min. váze P_t s např. polovinou zbývajícího shluku pro $p = 12$.

Metoda max. kompaktnosti

Pro přidání prvku v do shluku X rozhoduje jeho přínos pro kompaktnost shluku X. Narozdíl od metody max. zapojení není tato metoda příliš závislá (až na rychlost výpočtu) na zvolené prahové podobnosti.

Nechť $G = (V, E)$ je graf kolekce, v, z jsou vrcholy, K_i, O jsou množiny vrcholů, $FirstCom$ je proměnná, pak schéma 3.3 popisuje algoritmus hledání iniciálních shluků metodou maximální kompaktnosti.

Vstup:
graf kolekce $G = (V, E)$ pro prahovou podobnost P_t.
Výstup:
k iniciálních shluků $C_i \subset V$, $i = 1 \dots k$.

Algoritmus:

1. $i = 1$, zvol práh I_t
2. Najdi vrchol $v \in V$ s max. kompaktností $Com(Neigh(v))$.
3. $K_i = \{v\}$, $O = Neigh(v)$, $FirstCom = Com(Neigh(z))$
4. Najdi vrchol $z \in O$ s max. kompaktností $Com(K_i \cup z)$.
5. Pokud $Com(K_i \cup z) > I_t$ pokračuj krokem 6, jinak pokračuj krokem 7.
6. $K_i = K_i \cup \{z\}$, $O = O \cup Okoli(z)$, $V = V \backslash \{z\}$. Pokračuj krokem 4.
7. Zapiš K_i na výstup, $i = i + 1$, pokud $V \neq \emptyset$ pokračuj krokem 2.

Obr. 3.3: Algoritmus metody max. kompaktnosti

Volba prahu I_t:

- I_t je volena jako konstanta. Zaručíme tak nejmenší akceptovanou kompaktnost iniciálního shluku.
- $I_t = FirstCom \cdot p$, kde konstanta p určuje minimální poměr kompaktnosti shluku X po přidání prvku v ku iniciální kompaktnosti shluku X.

3.3.2 Lokální optimalizace shluku

Lokální optimalizace navazuje na hledání iniciálních shluků. Jejím cílem je z iniciálního shluku vytvořit přidáváním nebo ubíráním dokumentů výsledný konceptotvorný shluk. Proceduru, která provádí lokální optimalizaci nazveme **LocalSearch (LS)**.

Zvolme **prahový potenciál** $P_t \geq 0$. Prahový potenciál určuje nejmenší přijímaný příspěvek dokumentu ke kvalitě shluku. Volba nenulového P_t je nutná z důvodu rychlejšího průběhu a ukončení výpočtu.

Základní algoritmus, který je popsán schématem 3.4 a jehož obměny jsou popsány níže, nejprve hledá dokument d z X s nejmenším potenciálem. Pokud je $P(d, X) < -P_t \cdot Q(X)$, je dokument ze shluku ubrán. Toto se opakuje dokud existuje dokument se záporným potenciálem. Poté se hledá dokument d z $Y(X)$ s největším potenciálem. Pokud je $P(d, X) \geq P_t \cdot Q(X)$, dokument se do shluku přidá a začne se opět od začátku hledat dokument $d \in X$ s nejnižším potenciálem. Potenciály dokumentů $d \in X$ se totiž mohou změnit po každém jednom ubrání nebo přidání dokumentu do shluku. Toto celé se opakuje dokud se během cyklu provede alespoň jedna operace přidání/ubrání prvku z/do X.

Do výpočtu vstupují pouze dokumenty z X a $Y(X)$. Nutné je ovšem poznamenat, že množina Y se může během výpočtu výrazně měnit (pokud dokument z $Y(X)$ má hodně sousedů v $R(X)$, pak po přidání tohoto dokumentu do X se všichni jeho sousedi z $R(X)$ přidají do $Y(X)$).

Procedura `LocalSearch` generuje během výpočtu posloupnost shluků $X^{(0)}$, $X^{(1)}$, $X^{(2)}$, $\ldots X^{(k)}$ takovou, že platí:

1. $Q(X^{(i)}) > Q(X^{(i-1)})$ pro $i \geq 1$
2. shluk $X^{(i)}$ vždy vznikne ubráním nebo přidáním dokumentu z/do shluku $X^{(i-1)}$

Vstup:
graf kolekce dokumentů, iniciální shluk $X = X^{(0)} \subset \mathcal{C}$
Výstup:
lokálně optimální shluk X^*

Algoritmus:

1. Najdi vrchol $z \in X^{(i)}$ s min. potenciálem $P(z, X^{(i)})$.
2. Pokud $P(z) < -P_t \cdot Q(X)$, pak $X^{(i+1)} = X^{(i)} \backslash z$, $i = i + 1$, pokračuj krokem 1.
3. Najdi vrchol $z \in Y(X^{(i)})$ s max. potenciálem $P(z, X^{(i)})$.
4. Pokud $P(z) > P_t \cdot Q(X)$, pak $X^{(i+1)} = X^{(i)} + z$, $i = i + 1$, pokračuj krokem 1.
5. $X^* = X^{(i)}$

Obr. 3.4: Algoritmus lokální optimalizace shluku

Protože úpravami shluku nemůže jeho kvalita růst do nekonečna, musí existovat $k \geq 0$ tak, že $X^{(k)}$ je lokálně optimální a program `LocalSearch` se po k-tém průchodu cyklem zastaví.

Možné obměny algoritmu:

- Nejdříve prvky do X přidáváme, potom ubíráme. Tzn. záměna kroků 1 a 2 za 3 a 4.
- Nevybíráme prvek s minimálním (resp. maximálním) potenciálem, ale první prvek s potenciálem menším (resp. větším) nule.
- Místo ubrání a přidání vždy pouze jednoho prvku zkoušíme provádět naráz ubrání i přidání prvku.
- Místo ubrání a přidání vždy pouze jednoho prvku zkoušíme tyto operace provádět s dvojicemi, trojicemi,... prvků.

3.3.3 Natrénování λ-parametrů procedury LocalSearch

Pro natrénování λ parametrů procedury LocalSearch je navržena komparativní metoda.

Metoda vychází z předpokladu, že na trénovací kolekci známe nějaké konceptotvorné shluky určené anotátorem - takový shluk nazveme **vzorový shluk** a že dokážeme porovnat kvalitu dvou shluků. Tzn. můžeme určit: $Q(X_1) \geq Q(X_2)$. Pro vzorový shluk předpokládáme, že jeho kvalita je maximální možná, tzn. neexistuje shluk, který by vznikl z tohoto shluku libovolnou posloupností přidávání a ubírání prvku, a jehož kvalita je větší než kvalita vzorového shluku a dále je pro vzorový shluk definována funkce $\Phi_i : d \rightarrow 0, 1, 2, 3$, kde $d \in C600$. Hodnota Φ udává míru náležení dokumentu do shluku (3 je nejvíce).

Shluk, který vznikne malou modifikací vzorového shluku nazveme **srovnávací shluk**. Komparativní metoda je založena na vyřešení soustavy nerovnic, které porovnávají kvalitu vzorových a srovnávacích shluků. Nerovnice soustavy jsou dané systémem komparativních podmínek pro vzorový shluk C_i:

(a1)	$Q(C_i) \geq Q(C_i \backslash \{d\})$	pro $\forall d \in C_i$
(a2)	$Q(C_i) \geq Q(C_i \cup \{\overline{d}\})$	pro $\forall \overline{d} \in \overline{C}_i$
(b1)	$Q(C_i \backslash \{d\}) \leq Q(C_i \backslash \{d'\})$	pro $\forall d, d' \in C_i$, $\Phi_i(d) > \Phi_i(d')$
(b2)	$Q(C_i \cup \{\overline{d}\}) \geq Q(C_i \cup \{\overline{d'}\})$	pro $\forall \overline{d}, \overline{d'} \in \overline{C}_i$, $\Phi_i(\overline{d}) > \Phi_i(\overline{d'})$
(b3)	$Q(C_i \backslash \{d\}) \leq Q(C_i \cup \{\overline{d'}\})$	pro $\forall d \in C_i, \overline{d'} \in \overline{C}_i$, $\sqrt{\Phi_i(d)\Phi_i(\overline{d'})} > \Phi_t$

Kvalita vzorového shluku se sníží při ubrání libovolného prvku (podmínka a1) i při přidání libovolného prvku (podmínka a2). Dále platí, že kvalita vzorového shluku je nižší při ubrání prvku s větším Φ_i než při ubrání prvku s Φ_i menším (podmínka b1) a podobně je kvalita shluku vyšší při přidání prvku s větším Φ_i než při přidání prvku s Φ_i menším (podmínka b2). Poslední podmínka (b3) porovnává kvality shluku po přidání a ubrání prvků. Φ_t udává mezní míru náležení dokumentu do shluku.

Pro vzorový shluk C_i a srovnávací shluk C_i' potom jedna rovnice komparativní podmínky vypadá:

$$Q(C_i) - Q(C_i') > P_t \cdot Q(C_i) \tag{3.6}$$

Rozepišme si nyní rovnici 3.6 podle vzorce 3.1. Pokud bysme chtěli trénovat hodnoty podle vzorce 3.2, stačí vždy rozepsat λ_i jako $\lambda_i = \lambda_{i0} + \lambda_{i1} \cdot |X|$.

$$Q(C_i') < Q(C_i)(1 - P_t)$$
$$\log\left(\frac{Q(C_i')}{Q(C_i)}\right) < \log(1 - P_t)$$
$$\log(Com^{\lambda_1}(C_i) \cdot Exh^{\lambda_2}(C_i)) - \log(Com^{\lambda_1}(C_i') \cdot Exh^{\lambda_2}(C_i')) < \log(1 - P_t)$$
$$\lambda_1 \log(Com(C_i)) + \lambda_2 \log(Exh(C_i)) - \lambda_1 \log(Com(C_i')) - \lambda_2 \log(Exh(C_i')) < \log(1 - P_t)$$
$$\lambda_1(\log(Com(C_i)) - \log(Com(C_i')) + \lambda_2(\log(Exh(C_i)) - \log(Exh(C_i')) < \log(1 - P_t)$$

Dostáváme tak soustavu nerovnic $Ax \geq 0$, $A \in R^{n \times 2}$, kde n je počet komparativních podmínek, $a_1^i = \log(Com(C_i)) - \log(Com(C_i')$ a $a_2^i = \log(Exh(C_i)) - \log(Exh(C_i')$:

$$\begin{aligned} a_1^1\lambda_1 + a_2^1\lambda_2 &< \log(1 - P_t) \\ a_1^2\lambda_1 + a_2^2\lambda_2 &< \log(1 - P_t) \\ &\vdots \\ a_1^n\lambda_1 + a_2^n\lambda_2 &< \log(1 - P_t) \end{aligned}$$

Řešení této soustavy nemusí existovat, hledáme tedy řešení s nejmenší chybou. Jedna z možných metod, pro řešení soustavy nerovnic je převedení na úlohu **lineárního programování**.

Převod na úlohu lineárního programování:

Nechť $\sum_{i=1}^{2} a_{ji}\lambda_i \geq 0$ pro $j = 1 \ldots n$ je soustava nerovnic. Zvolme $d_j, j = 1 \ldots n, d_j \in \mathcal{R}$ takové, že platí $d_j \geq 0$ a $d_j \geq -\sum_{i=1}^{2} a_{ji}\lambda_i$. Dále rozepišme $\lambda_i = \lambda_{i1} - \lambda_{i2}$, kde $\lambda_i \in \mathcal{R}, \lambda_{i1}, \lambda_{i2} \geq 0$. Potom úloha nalézt $min(\sum_{j=1}^{n} d_j)$ s podmínkami $d_j + \sum_{i=1}^{2} a_i^j(\lambda_{i1} - \lambda_{i2}) \geq 0$ je standardní úlohou lineárního programování.

Soustava nerovnic po rozepsání má tvar:

$$\begin{aligned} d_1 + a_1^1\lambda_{11} - a_1^1\lambda_{12} + a_2^1\lambda_{21} - a_2^1\lambda_{22} &< \log(1 - P_t) \\ d_2 + a_1^2\lambda_{11} - a_1^2\lambda_{12} + a_2^2\lambda_{21} - a_2^2\lambda_{22} &< \log(1 - P_t) \end{aligned}$$

$$\begin{array}{rcl} & \vdots & \\ d_n + a_1^n\lambda_{11} - a_1^n\lambda_{12} + a_2^n\lambda_{21} - a_2^n\lambda_{22} & < & \log(1 - P_t) \end{array}$$

Označíme-li ch_j velikost chyby j-té nerovnice pro nejlepší možné řešení původní soustavy rovnic, potom platí, že $\sum_{j=1}^{n} d_j \geq \sum_{j=1}^{n} ch_j$.

Kapitola 4

Analýza a zpracování vstupních dat

Pro ověření postupů pro hledání konceptotvorných shluků použijeme českou kolekci dokumentů (ČKD). Tyto textové dokumenty jsou tvořeny zejména články z novin a časopisů a pocházejí ze tří zdrojů: Lidové noviny (LN), časopis Computer World (CW) a časopis PC World (PCW). LN jsou články, které poskytla redakce deníku Lidové noviny a pokrývají poměrně široké spektrum témat. Články z CW a PCW jsou převážně tématicky zaměřeny na oblast výpočetní techniky a elektroniky.

Při experimentech budeme pracovat se dvěma kolekcemi, které jsou vybrány z ČKD . S kolekcí obsahující 10000 náhodných dokumentů (ozn. C10000) a s anotovanou trénovací kolekcí 600 dokumentů (ozn. C600), která byla vytvořena a popsána v práci [11].

4.1 Kolekce C10000

Kolekce C10000 je množina 10000 dokumentů (článků) z ČKD. Tyto dokumenty jsou převedeny na sjednocený textový formát vhodný jako vstup pro další strojové zpracování a jsou vhodné svým obsahem (vypustily se např. rozhovory) i rozsahem (vypustily se dokumenty, které měly méně než 100 slov).
Dále jsou texty označkovány pomocí standardu SGML. Značky jsou trojího druhu: správní (vnější anotace), strukturní a lingvistické (vnitřní anotace). Správní značky obsahují především informace o původu, autorství, typu a zdroji textu, případně způsobu značkování. Nakonec prošly texty morfologickou analýzou. Podrobnější informace o značkování lze nalézt např. v [8].

Postup označkování je vzhledem ke sjednocenému formátu vstupních textů jednotný a

probíhá ve 3 krocích:

1. tokenizace textů
2. morfologické značkování textů
3. filtrace obsahově nevhodných souborů (počet nerozpoznaných slov je příliš velký)

Tokenizace textů

Vlastní text je nejprve procesem tokenizace hierarchicky členěn strukturními značkami na menší celky např. kapitoly, strany, odstavce, ty potom dále na věty, které jsou formálně tvořeny posloupností tzv. textových slov (tokenů). Tokeny mohou být slovní tvary, čísla, zkratky, interpunkční znaménka a další zvláštní znaky (symboly měn, fyzikálních jednotek, matematické symboly, atd.). Tokenizace je provedena programem `tokenizer` [9].

Morfologické značkování textů

V této fázi se prostřednictvím lingvistických značek přiřazují slovním tvarům jejich možné lingvistické atributy. Konkrétně se jedná o lemmatizaci a vytvoření morfologické značky. Lematizací je danému slovu (tokenu) přiřazena informace o jeho slovníkovém tvaru zvaném lemma. V případě nejednoznačnosti je přiřazeno více možných základních tvarů. každému lemmatu jsou pak přiřazeny všechny jeho potencionální morfologické interpretace, tj. informace o slovním druhu a dalších morfologických vlastnostech (rodu, čísle, pádu podstatných jmen, apod.). Každá morfologická interpretace je vyjádřena morfologickou značkou tvořenou 15 údaji, z nichž každý je reprezentován jedním znakem na příslušné předem pevně dané pozici. Nás bude zajímat zejména první znak, který reprezentuje slovní druh.
Ke každému slovnímu tvaru je tedy přiřazeno jedno či více lemmat a ke každému lemmatu jedna či více morfologických interpretací. Z nich je na základě kontextu metodou disambiguace (zjednoznačnění) vybrána ta nejpravděpodobnější. Bližší informace o způsobech disambiguace jsou např. v [9] nebo [10].

Filtrace obsahově nevhodných souborů

V posledním kroku došlo podle předem nastavených parametrů k filtraci nevhodných dokumentů z kolekce. Vyřazeny jsou dokumenty, u kterých tagger nerozeznal více než předem daný podíl slov (jedná se o nečeské texty nebo tabulky sportovních výsledků a pod.) Kritéria jsou různá pro LN, CW i PCW. V případě zdroje LN je tato hranice nastavena na 4% všech nerozpoznaných slov ze všech slov dokumentu. V případě CW a PCW je to 10%

(z povahy textů, které obsahují zvýšený počet cizích (anglických) slov, která český tagger nerozezná).

4.2 Anotovaná trénovací kolekce C600

V této kapitole je popsána anotovaná trénovací kolekce C600, která byla vytvořena v rámci diplomové práce [11].
Anotovaná trénovací kolekce C600 je množina 600 dokumentů náhodně vybraná z kolekce ČKD, převedená na stejný formát jako kolekce C10000 a anotátorem ručně doplněná o další níže popsané informace. Cílem ruční anotace bylo vytvoření množiny shluků pokrývající kolekci C600 a určení míry náležení dokumentů z C600 do těchto shluků. V neposlední řadě také porovnání a označení kvality vzniklých shluků.

Metoda vytvoření a anotování kolekce C600 je poněkud odlišná od budování některých známých kolekcí (např. TREC). Postup budování většiny testovacích kolekcí většinou začíná formulací informačních potřeb (testovacích dotazů) a následně se k těmto dotazům hledají a přiřazují relevantní dokumenty (např. podílovou metodou). Vzhledem k velikosti kolekce C600 byl zvolen víceméně opačný postup. Testovací dotazy (témata dokumentů) byly vytvářeny postupným procházením a čtením dokumentů (manuální víceúrovňovou indexací dokumentů). Tím se dosáhlo tématického pokrytí celé kolekce C600 a poměrně přesného přiřazení stupňů incidence dokumentů k nalezeným tématům.

Tento postup sestavení kolekce C600 lze rozdělit do 4 kroků:

1. výběr dokumentů kolekce C600 z ČKD
2. vytváření slovníku konceptů
3. víceúrovňová manuální indexace (anotace) dokumentů
4. ohodnocení koherence množin relevantních dokumentů

Výběr dokumentů kolekce C600

Velikost kolekce C600 byla předem stanovena na 600 dokumentů. Tyto dokumenty jsou vybrány z ČKD následujícím způsobem:

- všechny dokumenty ČKD byly zaindexovány do vyhledávacího systému BTRS (Bool Text Retrieval System - jednoduchý boolský systém) podporující pouze konjunkce a negace literálů [11]

- pomocí několika boolských dotazů z různých oblastí a manuálním výběrem z výsledků dotazů je vybráno celkem 200 dokumentů (tím bylo zajištěno, že v kolekci existují alespoň některé dvojice dokumentů s více či méně podobným obsahem - tématem).
- zbylých 400 dokumentů je ze zbytku kolekce vybráno zcela náhodně

Vytváření slovníku konceptů

Konceptem budeme nazývat množinu lemmat nebo slovních spojení, které charakterizují téma (obsah) dokumentu na dané úrovni abstrakce. Koncept je tedy jakési označení shluku dokumentů, které jsou si obsahově podobné.

Nejprve bylo vybráno (vytipováno) 26 konceptů, které velmi pravděpodobně pokrývají témata obsažená v dokumentech kolekce. Další koncepty (podrobnější) vznikaly v průběhu indexace dokumentů postupným zařazováním nových konceptů na základě zjištění nového tématu v právě anotovaném dokumentu.

Víceúrovňová manuální indexace

Množinu dokumentů z C600 odpovídající nějakému nalezenému konceptu c_i budeme nazývat **vzorový shluk** a značit jej C_i. Ke každému konceptu c_i je manuálně definována funkce $\Phi_i : d \rightarrow 0, 1, 2, 3$, kde $d \in$ C600.

Hodnota Φ_i je přiřazena každému dokumentu z C600 podle následujících pravidel:

0 ... přiřazena automaticky všem konceptům, které nebyly vybrány jako incidentní
1 ... c_i charakterizuje pouze téma, o kterém pojednává pouze malá část dokumentu (c_i charakterizuje okrajové téma dokumentu)
2 ... c_i charakterizuje téma, o kterém pojednává významná část dokumentu (c_i charakterizuje významné téma dokumentu)
3 ... c_i charakterizuje téma, o kterém pojednává celý dokument (c_i charakterizuje hlavní téma celého dokumentu)

Z popisu stupnice hodnot Φ_i a výsledku anotace je zřejmé, že přiřazení hodnoty incidentu konceptu a dokumentů není lineární, protože hodnoty 2 a 3 jsou si blíže než hodnoty 1 a 2.

Dále je určena prahová hodnota náležení do vzorového shluku Φ_t, která určuje velikost vzorových shluků:

$$C_i = \{d \in C600 | \Phi_i(d) \geq \Phi_t\} \tag{4.1}$$

Ohodnocení koherence shluků dokumentů

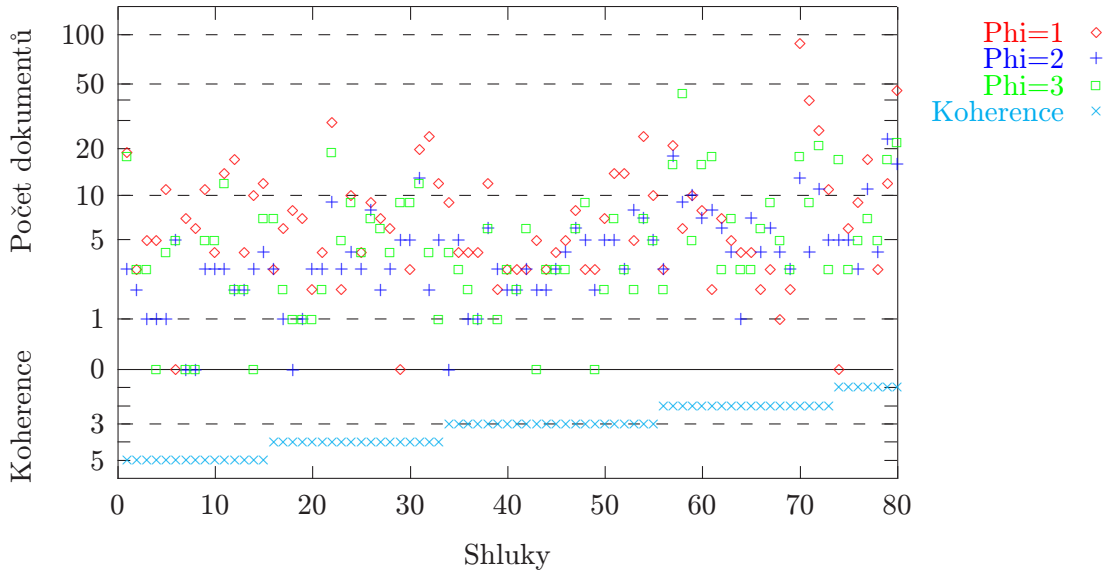
V posledním kroku anotátor ručně prošel (přečetl) všechny shluky a určil jejich míru koherence. Shluk dokumentů má větší míru koherence (je více koherentní) v případě, že bez ohledu na téma nebo počet dokumentů daného shluku je více zřejmé, proč právě tyto dokumenty jsou v jednom shluku. Míru (stupeň) koherence shluku dokumentů C_i budeme značit κ_i. Míra koherence může nabývat pouze 5 hodnot (4.1):

Míra koherence	**Název kategorie**	**Popis kategorie**
1	slabě koherentní	společné téma je pouze okrajovým tématem všech dokumentů ve shluku
2	málo koherentní	společné téma je pouze okrajovým tématem velké většiny dokumentů ve shluku, u ostatních (menšiny) je významné nebo hlavní
3	středně koherentní	společné téma je okrajovým tématem většího množství dokumentů ve shluku, u ostatních je významné nebo hlavní
4	hodně koherentní	společné téma je významným nebo hlavním tématem většiny dokumentů ve shluku, ale existuje několik málo dokumentů, u kterých je jen okrajové
5	silně koherentní	společné téma je významným nebo hlavním tématem všech dokumentů ve shluku

Tab. 4.1: Ohodnocení míry koherence shluku dokumentů.

Výsledná trénovací množina vzorových shluků

Celkem je v C600 pro $\Phi_t = 2$ určeno 80 vzorových shluků s průměrnou velikostí 14,31 dokumentu. Informace o počtu dokumentů ve shlucích a jejich koherencích je obsažena v grafu na obr. 4.1. V horní části grafu (nad osou x) je pro každý vzorový shluk určen počet dokumentů pro jedn. Φ. Pozor - tato část má logaritmickou osu y. V dolní části grafu (pod osou x) je pro každý vzorový shluk určena jeho koherence κ_i.



Obr. 4.1: Počet dokumentů shluků pro různá Φ a Koherence κ shluků

Jak vidíme na obr. 4.1, nezáleží míra koherence na velikosti shluků. Co je ale více zajímavé, neexistuje ani souvislost mezi koherencí a poměrem dokumentů s různou Φ_i.

Další podrobnější informace o struktuře a způsobu anotace trénovací kolekce C600 lze nalézt v [11]. K anotované kolekci se ještě vrátíme v kapitole trénování lambda parametrů.

4.3 Porovnání kolekcí a vytvoření vektorů dokumentů

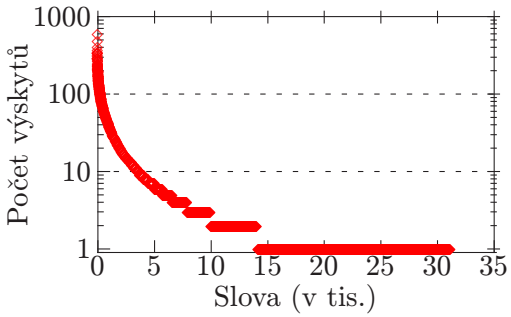
Kolekce C600 narozdíl od C10000 obsahuje informaci dodanou anotátorem. Tato informace je využita pro natrénování parametrů modelu LocalSearch a následnému ověření a vyhodnocení výsledků LS s natrénovanými parametry na této kolekci. Takto natrénované parametry chceme dále použít na kolekci C10000 a vyhodnotit dosažené výsledky. Aby

práce měla alespoň nějakou šanci na úspěch, musí být struktura kolekce C10000 alespoň částečně podobná C600. Podobností se myslí např. podobná frekvence slov v textech (ozn. DF z angl. document frequency), velikost dokumentů před a po filtraci slov a následná distribuce stupňů a velikosti hran v kolekci.
Dále je analýza vstupních dat potřeba pro určení základních parametrů pro filtraci slov a tím určení velikosti vektorů dokumentů a nalezení ideální prahové podobnosti (PP) pro výpočet iniciálních shluků.

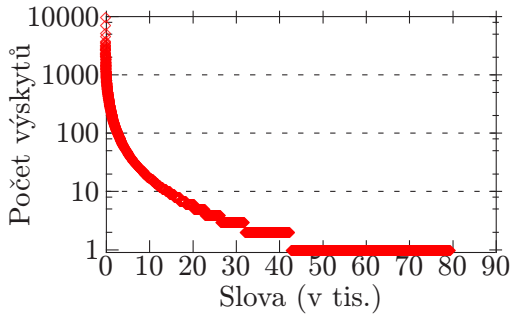
Popis	**C600**	**C10000**
Velikost kolekce (počet dokumentů)	600	10000
Velikost kolekce (MB)	28	174
Celkový počet slov v kolekci	390 712	2 107 670
Počet různých slov v kolekci	31 801	80 894
Průměrný počet slov v dokumentu	651,186	210,767

Tab. 4.2: Základní popis kolekcí C600 a C10000.

První informace o kolekcích dává tabulka 4.2. Podle tabulky je zřejmé, že počet různých slov je dokonce i v kolekci C600 příliš velký. Při použití všech slov by vznikly příliš velké vektory, které by byli hodně řídké. Mnoho slov se navíc vyskytuje pouze v několika málo dokumentech a jejich význam pro určení podobnosti dokumentů je minimální nebo dokonce nulový (slova s DF rovno 1). Frekvence slov v dokumentech je patrná z grafů na obr. 4.2 a obr. 4.3.



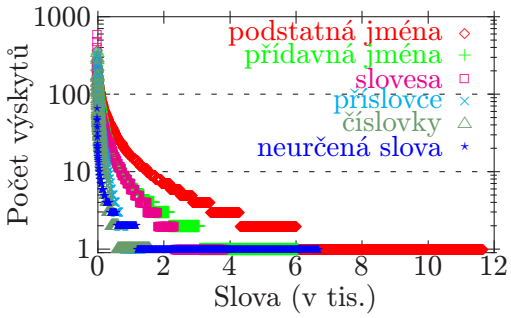
Obr. 4.2: Slova podle DF pro C600



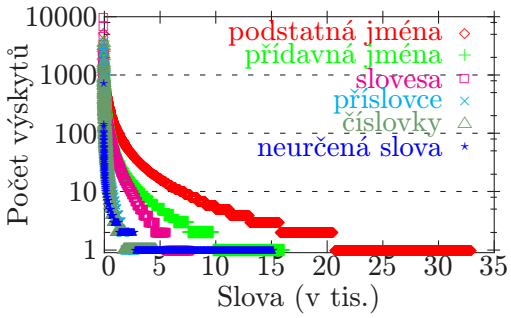
Obr. 4.3: Slova podle DF pro C10000

Minimální akceptovaná velikost shluku je 5 dokumentů. Vyfiltrujeme tedy slova, která se vyskytují v méně než 4 dokumentech. Neočekáváme , že vzniklé shluky budou větší než

cca maximálně polovina kolekce. A i této hranice nejspíše nedosáhneme. Proto můžeme vyfiltrovat slova, která se vyskytují ve více než v polovině dokumentů kolekce.



Obr. 4.4: Slovní druhy podle DF pro C600



Obr. 4.5: Slovní druhy podle DF pro C10000

Pokud bychom vyfiltrovaly všechna slova, která se vyskytují v méně než 4 dokumentech a více než v polovině dokumentů kolekce, získáme vektory dokumentů dlouhé 7931 pro C600 a 26591 pro C10000. I tahle filtrace je ale příliš hrubá a získané vektory jsou velké a řídké. Je zřejmé, že ne všechny slovní druhy přenášejí stejně hodnotnou informaci o podobnosti dokumentů. Podrobný pohled na zastoupení jednotlivých slovních druhů zobrazují grafy na obr. 4.4 a obr. 4.5. Mezi slovními druhy je zařazena i množina slov, pro která automat nedokázal slovní druh určit. Tato slova ale nemůžeme ignorovat pro jejich velký počet (viz obr. 4.4 a obr. 4.5) a informační hodnotu. Typickým příkladem takového neurčeného slova je „Macintosh“, „Network“ nebo „T602“.

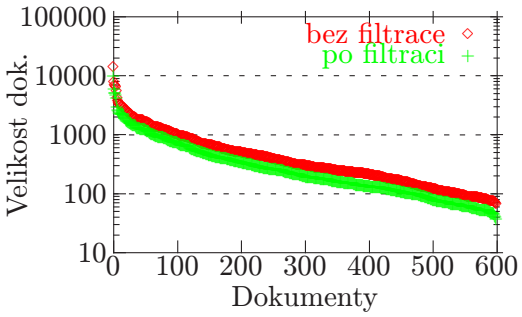
	#C600		**#C10000**	
Slovní druh	dolní mez	horní mez	dolní mez	horní mez
podstatná jména	4	300	4	5000
přídavná jména	4	300	4	5000
slovesa	4	300	4	5000
příslovce	0	0	0	0
číslovky	0	0	0	0
neurčený druh	4	300	4	5000

Tab. 4.3: Podmínky na DF slov zadané ve filtru.

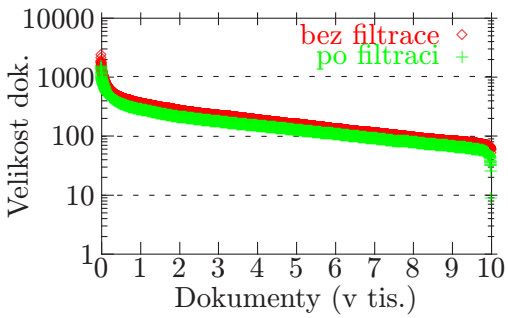
Informaci použitelnou pro podobnost dokumentů určitě nepřináší příslovce. Minimum in-

formací nesou také číslovky (výjimkou však může být např. datum, ale to často se nevyskytuje). Oba slovní druhy proto ze slovníku vypustíme. Finální nastavení filtru je vypsáno v tab. 4.3.

Počet různých slov po použití filtru a tedy i velikost vektoru dokumentů je 7046 pro C600 a 24333 pro C10000. Velikost vektorů dokumentů se tedy zmenšila cca pětkrát. Změnu velikosti dokumentů před a po filtraci zachycují grafy na obr. 4.6 a obr. 4.7.



Obr. 4.6: Dokumenty podle velikosti pro C600



Obr. 4.7: Dokumenty podle velikosti pro C10000

Z finálního slovníku, který jsme získali použitím filtru s nastavením z tab. 4.3, vypočítáme vektory všech dokumentů v kolekci. Každý dokument d_i je potom reprezentován jako m-složkový vektor, kde m je počet termů ve slovníku. Formálně:

$$d_i = [w_{i1}, w_{i2}, \ldots, w_{im}] \quad ; \quad w_{ik} \in < 0, 1 > \tag{4.2}$$

kde jedn. složky vektoru w_{ik} spočítáme:

$$w_{ik} = \frac{tf_{ik} log(N/n_k)}{\sqrt{\sum_{k=1}^{t} (tf_{ik})^2 (idf_k)^2}} \tag{4.3}$$

při značení:
w_{ik} = k-tá složka vektoru w dokumentu d_i
tf_{ik} = frekvence termu k v dokumentu d_i
n_k = počet dokumentů z kolekce obsahující term k
N = počet dokumentů v kolekci
$idf_k = log(n_k/N)$, tj. inverzní DF termu k v kolekci

4.4 Podobnost dokumentů a struktura grafu kolekce

Dalším krokem po filtraci slov a sestavení vektorů dokumentů je výpočet podobnosti dokumentů d_i a d_j. Podobnost dvou dokumentů spočítáme jako skalární součin jejich vektorů:

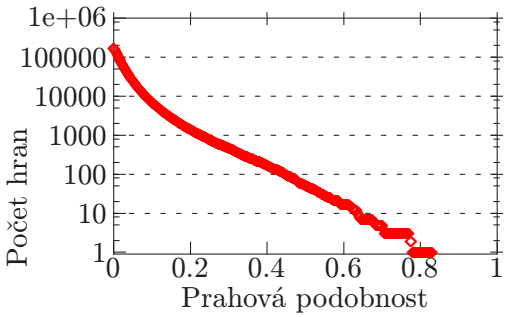
$$sim(d_i, d_j) = \sum_{k=1}^{t} w_{ik} \cdot w_{jk} \tag{4.4}$$

Po provedení výpočtu podobnosti vznikne 175 104 nenulových podobností mezi dokumenty kolekce C600 a 47 448 733 nenulových podobností mezi dokumenty kolekce C10000. Počet nulových podobností mezi dokumenty může být překvapující, ale po prozkoumání kolekce se zjistilo, že existují na první pohled „normální“ novinové články, ve kterých se nevyskytuje např. slovo „být“. Kontrolou se potvrdila nulová podobnost mezi nalezenými dokumenty.

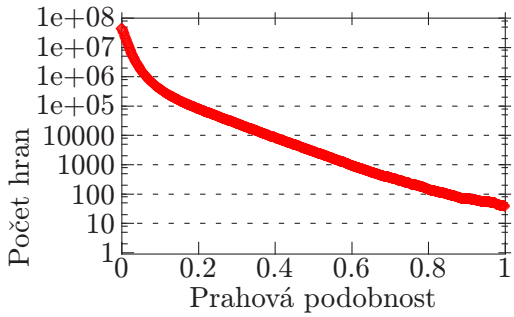
Výpočtem podobnosti dokumentů z kolekce C jsme vytvořili **vážený graf G** kolekce C, kde dokumenty jsou vrcholy grafu G a podobnost dvou dokumentů je váha hrany mezi nimi. Dále definujeme prahovou podobnost (PP) jako nejnižší možné ohodnocení hrany v grafu. Formálně:

Nechť V je množina vrcholů grafu a $E = \{(d_i, d_j) : sim(d_i, d_j) \geq PP; d_i, d_j \in V\}$ je množina hran, pak $G = (V, E)$ je graf kolekce V při prahové podobnosti PP.

Co nás dále bude zajímat je struktura grafu **G**, zejména počet hran, distribuce velikostí hran a distribuce stupňů vrcholů při různé PP.



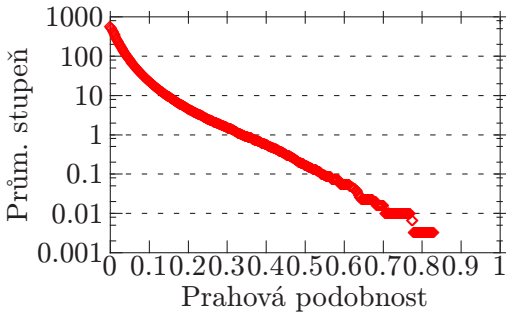
Obr. 4.8: Počet hran při PP pro C600



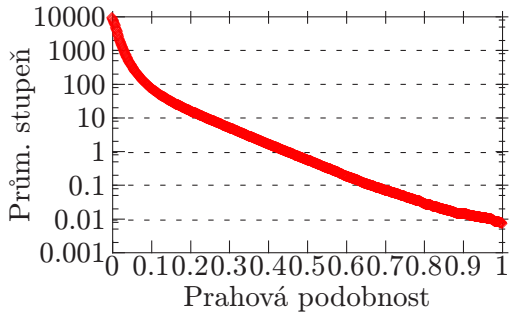
Obr. 4.9: Počet hran při PP pro C10000

Při zvyšování prahové podobnosti samozřejmě ubývají v grafu G hrany - viz graf na obr.

4.8 a obr. 4.9. Zajímavé je, že zatímco v kolekci C600 má hrana s nejvyšším ohodnocením váhu 0,823, tak v kolekci C10000 existují hrany s ohodnocením 1. Vysvětlení je ve vzniku kolekce C10000. Jak je uvedeno na začátku kapitoly, kolekce C10000 jsou novinové články z Lidových novin. Kupodivu některé články se v nepozměněné podobě vyskytují v kolekci několikrát. Nějaký novinář si nejspíše „ulehčil“ práci a článek vydal dvakrát v různých letech. I přesto, že nalézt tyto články a odstranit je z kolekce je snadné, ponecháme je v kolekci pro zachování autentičnosti. Z grafů na obr. 4.8 a obr. 4.9 lze snadno spočítat průměrný stupeň vrcholů při dané prahové podobnosti, což je zachyceno na obr. 4.10 a obr. 4.11.



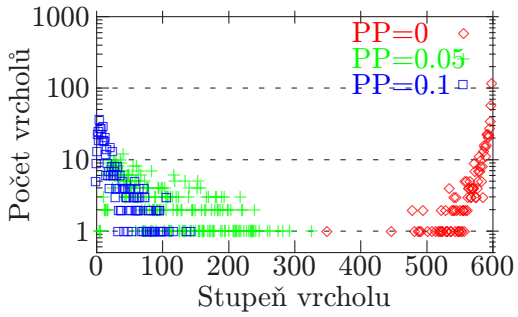
Obr. 4.10: Průměrný stupeň vrcholu při PP pro C600



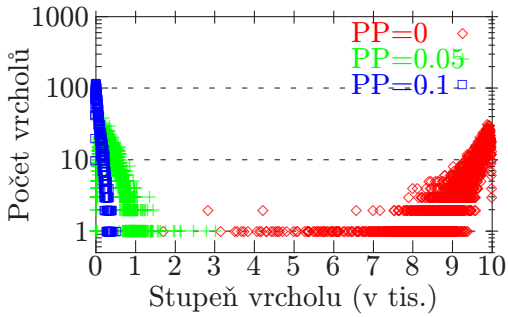
Obr. 4.11: Průměrný stupeň vrcholu při PP pro C10000

Úbytek hran zvyšuje rychlost programů pracujících s grafem kolekce a dále výrazně snižuje paměťové nároky. Je ovšem důležité, aby se snížením počtu hran nezměnila výrazně struktura celé kolekce. Např. pokud by z poloviny vrcholů kolekce vycházely pouze slabé hrany a z poloviny pouze silné, při zvýšení prahové podobnosti by se graf rozpadl na jednu komponentu s vrcholy se silnými hranami a spoustou samostatných vrcholů. Ztratili bychom tedy informace o celé jedné polovině kolekce, protože nový graf by měl úplně jinou strukturu než graf původní. Bližší pohled na rozložení hran (počet vrcholů pro daného stupně) v grafech kolekcí C600 a C10000 ukazují grafy na obr. 4.12 až 4.15.

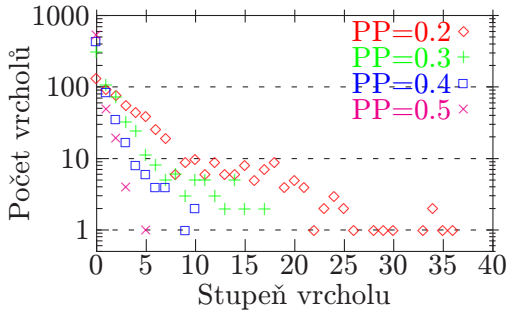
Z grafů na obr. 4.12 až 4.15 je patrné, že PP=0 oproti PP=0.05 nepřináší příliš nových informací, pouze graf rovnoměrně „zašumí“, tj. vyplní graf malými hranami (nejnižší stupeň je pak 350 u C600 a 1700 u C10000). To je důležité zejména pro optimalizaci poměru „rychlost/výkon“ programů pro hledání shluků, protože už při přechodu z PP=0 na PP=0,05 se sníží počet hran u kolekce C600 10x a u C10000 dokonce cca. 40x, což výrazně zvýší rychlost programů a sníží paměťovou náročnost.



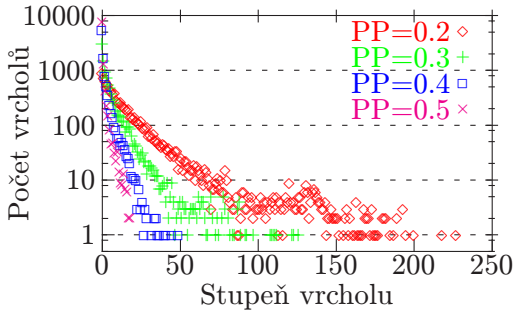
Obr. 4.12: Počet vrcholů daného stupně pro C600



Obr. 4.13: Počet vrcholů daného stupně pro C10000



Obr. 4.14: Počet vrcholů daného stupně pro C600



Obr. 4.15: Počet vrcholů daného stupně pro C10000

4.5 Shrnutí porovnání kolekcí C600 a C10000

Kolekce C600 se od C10000 liší ve dvou vlastnostech (samozřejmě kromě velikosti kolekcí a s tím souvisejících vlastností jako např. počet hran).
První je rozdílná velikost dokumentů v kolekci. V C600 je několik dokumentů velikosti přes 5000 slov, které způsobí velikou průměrnou délku dokumentu v kolekci. V C10000 jsou ovšem mnohem více zastoupeny menší dokumenty, které průměrnou velikost dokumentu v kolekci sníží.
Druhý rozdíl mezi kolekcemi je v existenci stejných dokumentů v kolekci C10000, zatímco v C600 jsou všechny dokumenty pouze v jednom exempláři.

Pro další výpočty nad kolekcemi ovšem není důležitá velikost souborů, ale pouze rozložení a váha hran mezi vrcholy grafu kolekcí. Vyšší velikost několika dokumentů nemá na rozložení a váhu hran velký vliv, jak je zřejmé z grafů v předchozích sekcích srovnávající obě kolekce. Diference způsobená stejnými dokumenty v kolekci je velmi malá (jde pouze o cca

60 hran z 47 448 733 v kolekci) a navíc díky způsobu dalšího výpočtu nepodstatná.

Po zanedbání dvou výše zmíněných rozdílů můžeme říci, že kolekce jsou si podobné „až na měřítko“, viz 4.2 až 4.7. Důležité také je, že obě kolekce mají podobné rozložení hran i při libovolně zvolené PP, což je vidět z obr. 4.12 až 4.15. Toho využijeme zejména při hledání iniciálních shluků.

Kapitola 5

Experimenty

Roztřídění textů do kategorií, případně navrácení relevantních textů k dotazu nad množinou dokumentů, je příliš náročné pro ruční anotaci. Pokud by výstup shlukovacího algoritmu byl „podobný“ výsledku ruční anotace, ušetřil by mnoho času a peněz. Je však nesnadné určit, co znamená ona podobnost dvou výstupů shlukovacích algoritmů. A také je nejednoznačné, co vlastně znamená dobré rozdělení kolekce dokumentů do shluků. Pro různé účely mohou být vhodná jiná rozdělení dokumentů do shluků. Jako příklad můžeme uvést např. rozdělení novinových článků – podle obsahu nebo podle data novin, ve kterých byly otištěny.

V této kapitole porovnáme výsledky alg. pro nalezení iniciálních shluků, natrénujeme nejlepší λ-parametry pro LocalSearch (LS) a ověříme výsledky LS na C600 a C10000. Pro větší spolehlivost výsledků můžeme vybrat z několika měr [12]. Každé nové měřítko (míra) pro porovnání kvality shlukování má přínos pro jeho lepší porozumění. Podle způsobu měření kvality shlukování můžeme tuto míru rozdělit na vnitřní a vnější. Vnitřní míra je založena na reprezentaci shluků a může to být např. podobnost. Vnější míra porovnává libovolné shlukování s referenčním shlukováním (typicky shlukování pomocí ruční anotace).

V našem případě máme dány dvě kolekce: C600 a C10000. V kolekci C600 máme anotátorem nalezené vzorové shluky. Tyto vzorové shluky budeme považovat za ideální a budeme je porovnávat s výstupem shlukovacích algoritmů LS a C-means (v textu budou jejich výstupy nazývány pouze jako shluky). Protože algoritmus LocalSearch i algoritmus C-means vyžaduje jako vstup nějaké iniciální shluky (stačí i náhodně generované) a kvalita (a výpočtová rychlost) jejich výstupů se zlepší při použití kvalitních iniciálních shluků, porovnáme se vzorovými shluky z kolekce C600 i výstupy algoritmů pro iniciální shluky. Pro porov-

nání alg. pro iniciální shluky s nějakým jiným známým algoritmem použijeme algoritmus K-means.
V kolekci C10000 nemáme vzorové shluky k dispozici. Pro porovnání výsledků a alespoň částečné ověření hypotéz, které formulujeme díky výsledkům na C600, použijeme míry vnitřní.

5.1 Míry pro evaluaci

Přesnost a úplnost

V souvislosti se shlukováním se v [12] definují pojmy přesnost (precision, ozn. p) a úplnost (recall, ozn. r) porovnávající shluk i se vzorovým shlukem j následovně:

$$p_{ij} = \frac{n_{ij}}{n_i}, \qquad r_{ij} = \frac{n_{ij}}{n_j}, \tag{5.1}$$

kde n_{ij} je počet dokumentů ze vzorového shluku j ve shluku i (tj. velikost jejich průniku), n_i je počet textů ve shluku i a n_j je počet textů ve vzorovém shluku j. Pomocí přesnosti p nadefinujeme **čistotu** (z ang. purity) shluku jako $\rho_i = \max\{p_{ij}\}$. Nyní můžeme použít vážený průměr čistoty ρ přes všechny shluky jako míru kvality celého shlukování:

$$\rho = \sum_i \frac{n_i}{n_{ci}} \rho_i = \frac{n_{max}}{n}, \tag{5.2}$$

kde $n_{ci} = \sum_i |X_i|$ je počet dokumentů v nalezených shlucích a n_{max} je počet dokumentů v celé kolekci, které jsou součástí shluku, v němž je počet dokumentů z jedné třídy větší než počet všech zbývajících dokumentů.
Čistota shluku reflektuje, jak přesně se v nejlepším případě zobrazí shluk na nějaký vzorový shluk. Nabývá hodnot z intervalu $<0, 1>$. Větší hodnota znamená přesnější rozdělení do shluků vzhledem ke vzorovým shlukům. Čistota shluku je rovna jedné, pokud všechny shluky jsou nalezeny ve vzorových shlucích (stačí, pokud je shluk podmnožinou lib. vzorového shluku).

Entropie

Přesnost se dá také interpretovat [12] jako pravděpodobnost, že text náhodně vybraný ze shluku i patří do vzorového shluku j. Proto můžeme spočítat entropii shluku jako:

$$E_i = -\sum_j p_{ij} \log(p_{ij}) \tag{5.3}$$

Entropie pro celé shlukování o N shlucích je průměr přes všechny shluky:

$$E = \sum_i \frac{E_i}{N} \tag{5.4}$$

Entropie je nezáporné číslo. Měří homogenitu shluků vzhledem ke vzorovým shlukům. Čím nižší je entropie, tím více homogenní a kvalitnější jsou shluky. Nuly nabývá pouze v případě, že ve vzorových shlucích je lib. dokument z kolekce pouze jednou a nalezené shluky přesně odpovídají shlukům vzorovým.

F-míra

F-míra pro shluk i a vzorový shluk j je:

$$F_{ij} = \frac{2r_{ij}p_{ij}}{r_{ij} + p_{ij}} \tag{5.5}$$

V [13] je F-míra pro hierarchické shlukování definována následovně: F-míra pro každý shluk v kolekci je $F_j = \max_i\{F_{ij}\}$, kde maximum je přes všechny shluky v kolekci na všech úrovních hierarchie shlukování. Pokud označíme $n_{cj} = \sum_j |C_j|$, pak F-míra pro celé shlukování je definována jako:

$$F = \sum_j \frac{n_j}{n_{cj}F_j} \tag{5.6}$$

Průměr je počítán přes všechny vzorové shluky narozdíl od přesnosti a entropie.
F-míra kombinuje přesnost a úplnost z teorie získávání informací. Každý shluk je uvažován jako výsledek dotazu nad kolekcí a vzorové shluky jako ideální odpovědi. Hodnota F-míry je z intervalu $<0, 1>$. Větší hodnota koresponduje s větší kvalitou shluků.

Odchylka incidence

Protože předcházející míry počítají pouze s binárním rozdělením dokumentů do shluků (dokument ve shluku je nebo není), hodí se nám tyto míry pro porovnání algoritmů pro vytvoření iniciálních shluků a algoritmu LocalSearch, u kterého převedeme fuzzy rozdělení na binární. Pro přesnější porovnání algoritmu LocalSearch se zachováním fuzzy rozdělení dokumentů do shluků, definujeme míru $I_\triangle$ (odchylka incidence), která s tímto fuzzy dělením počítá. U alg. ale musíme určit mez m, která určuje hranici shluku ve fuzzy rozdělení. Hodnotu m si můžeme představit jako "zarážku", které dokumenty se vrátí jako odpověď na dotaz a které už ne.

Kladnou odchylku potenciálu $I^+_\triangle(X_i, C_j)$ shluku X_i od vzorového shluku C_j a zápornou odchylku potenciálu $I^-_\triangle(X_i, C_j)$ definujeme jako:

$$I^+_\triangle(X_i, C_j) = \frac{\sum_{d \in X_i \setminus (X_i \cap C_j)} [\mu(d, X_i) - m]^2}{n_i} \tag{5.7}$$

$$I^-_\triangle(X_i, C_j) = \frac{\sum_{d \in C_j \setminus (X_i \cap C_j)} [m - \mu(d, X_i)]^2}{n_j} \tag{5.8}$$

$I^+_\triangle$ započítává dokumenty, které jsou vzhledem ke vzorovému shluku ve výsledném shluku navíc. $I^-_\triangle$ naopak započítává dokumenty, které ve výsledku chybí. Pomocí kladné a záporné odchylky definujeme odchylku incidence $I_{\triangle_i}$ pro shluk i jako $I_{\triangle_i} = min_j\{I^+_\triangle(X_i, C_j) + I^-_\triangle(X_i, C_j)\}$. Odchylka potenciálu pro celé shlukování s N shluky je potom definována jako:

$$I_\triangle = \frac{1}{N} \sum_i I_{\triangle_i} \tag{5.9}$$

Čím nižší je celková odchylka, tím přesnější je shlukování vzhledem ke vzorovým shlukům. Odchylka narozdíl od předcházejících měr využívá fuzzy rozdělení do shluků. Lze ji ale samozřejmě počítat i pro binární rozdělení.

Zatímco čistota a entropie shlukování se zaměřují spíše na kvalitu zobrazení nalezených shluků na shluky vzorové, F-míra a odchylka zachycují navíc i kvalitu zobrazení vzorových shluků na shluky nalezené. Jak uvidíme dále, jsou to často dva protichůdné požadavky, kdy zvyšujeme čistotu a entropii zmenšením nalezených shluků, ale tím se zhoršuje F-míra a odchylka.

Inter- a intra- shluková podobnost

Všechny předcházející míry porovnávaly výsledný shluk se vzorovým shlukem. V případě kolekce C10000 ale žádné vzorové shluky k dispozici nemáme. Zavedeme si proto míry P_{intra} a P_{inter}. První z nich odpovídá kompaktnosti shluků, zatímco druhá reflektuje jejich izolaci od ostatních dokumentů v kolekci. Tato koncepce byla zveřejněna v článku [14] pro demonstraci "fraktálové podstaty" dat a rozšířena o pravděpodobnostní interpretaci v [15]. Mějme k shluků nad n dokumenty z kolekce C. Tyto shluky bude reprezentovat k vektorů středních hodnot μ_i. Potom intra shluková podobnost pro mez x je definována jako počet

dokumentů z C, které jsou v x-okolí střední hodnoty μ_i shluku X_i, do nějž dokument patří, ku počtu všech dokumentů z C obsažených v k shlucích. Formálně

$$P_{intra}(x) = \frac{1}{|C| \cdot k} \cdot |\{d \in C \mid d \in X_i,\ sim(d, \mu_i) \geq x\}| \tag{5.10}$$

Podobně budeme definovat inter shlukovou podobnost jako podíl všech dvojic typu dokument d z C patřící do X_i a střední hodnota μ_j shluku X_j, pro něž je d z x-okolí μ_j. Formálně

$$P_{inter}(x) = \frac{1}{|C|(k-1)} \cdot |\{(d, \mu_j) \mid d \in X_i,\ j \neq i,\ sim(d, \mu_j) \geq x\}| \tag{5.11}$$

Intra a inter shluková podobnost znázorňují kompaktnost a izolaci shluků a využívají pouze středních hodnot nalezených shluků a jejich podobnost s dokumenty.

Náhodné shlukování

Abychom nahlédli, jak složité je shlukování nad nějakou kolekcí, použijeme srovnání s náhodným shlukováním. Kvalita libovolného rozdělení kolekce shlukovacím algoritmem je potom porovnávána s kvalitou náhodného rozdělení. V textu budeme míru nad náhodným shlukováním označovat horním indexem „rand“ (např. ρ^{rand}).

5.2 Výsledky pro iniciální shluky

Výsledky algoritmů pro nalezení iniciálních shluků budeme na kolekci C600 porovnávat se vzorovými ručně anotovanými shluky. Pro posouzení, jak „dobré“ tyto algoritmy jsou, je navíc srovnáme s výsledky algoritmu K-means (popis viz předcházející sekce Nehierarchické metody) a náhodným shlukování. Naším cílem ovšem není natrénovat parametry algoritmů a vybrat ten nejlepší, ale pouze určit, které parametry dávají „použitelné“ výsledky. Všechny výstupy mohou totiž v krajním případě sloužit jako vstup pro následné zpracování programem LS. Ovšem zpracováním „nevhodných“ respektive málo kvalitních shluků se zvyšuje časová náročnost výpočtu. Proto je naším cílem vybrat pouze relevantní

iniciální shluky. Tabulka 5.1 sumarizuje použité algoritmy, jejich vstupy a potřebné parametry.

Metoda	**Parametry**	**Vstup**
K-means [KM]	Počet požadovaných iniciálních shluků	Vektory dokumentů
Max. zapojení [MZ]	Prahová podobnost + nejmenší stupeň vrcholu v inic. shluku (případně poměr ku počtu dokumentů ve shluku)	Podobnosti dokumentů
Max. kompaktnost [MK]	Prahová podobnost + nejmenší kompaktnost inic. shluku (nebo poměr k první komp.)	Podobnosti dokumentů

Tab. 5.1: Metody a jejich potřebné parametry

5.2.1 Porovnání algoritmů na C600

Na kolekci C600 máme k dispozici vzorové shluky určené anotátorem, proto pro porovnání můžeme použít míry vnitřní i vnější. Nejdříve proto provedeme porovnání výstupu algoritmů se vzorovými shluky pomocí měr čistota, entropie, F-míra a odchylka. Následně porovnáme výstupy mezi sebou pomocí míry P-intra a P-inter.

Míry založené na porovnání nalezených a vzorových shluků

V tabulkách 5.2 až 5.3 jsou uvedeny výsledky algoritmů KM, MK a MZ pro jednotlivé prahové podobnosti. Na závěr uvedeme tabulku s nejlepšími výsledky u každého alg. V tabulkách výsledků je červenou barvou zvýrazněn nejhorší výsledek algoritmu pro každý parametr I_t programů MZ a MK resp. pro různý počet shluků alg. KM. Zelenou je naopak označen výsledek nejlepší.

Nepříjemné omezení algoritmu KM je nutnost zadat počet požadovaných shluků. Výhodou je naopak rychlost a jednoduchá implementace algoritmu. Z tabulky 5.2 lze vypozorovat, že s klesajícím počtem shluků stoupá přesnost a entropie, ale klesá F-míra a odchylka. Protože dále vlastností alg. KM je, že pro každý dokument určí shluk (žádný nevynechá), můžeme i

Míra	$c = \frac{\|C\|}{20}$	$c = \frac{\|C\|}{10}$	$c = \frac{\|C\|}{7}$	$c = \frac{\|C\|}{5}$
Počet shluků	28	43	49	53
Čistota	0.363	0.408	0.427	0.442
Entropie	3.792	3.035	2.978	2.813
F-míra	0.277	0.289	0.271	0.260
Odchylka	0.707	0.717	0.726	0.735

Tab. 5.2: Výsledky alg. KM na C600 pro různý počet inic. shluků

bez znalosti vzorových shluků říct, že je mezi nimi několik velkých shluků, které pokrývají velkou část kolekce C600 a zbytek je rozdělen mezi neznámý počet menších shluků, což je potvrzeno v kapitole o analýze dat v sekci Anotovaná trénovací kolekce C600.

PP	**Míra**	$I_t = 0.1$	$I_t = 0.3$	$I_t = 0.5$	$I_t = 0.7$
0.1	Počet shluků	17	28	24	20
	Čistota	0.397	0.585	0.661	0.684
	Entropie	2.966	2.426	2.129	2.149
	F-míra	0.282	0.356	0.333	0.283
	Odchylka	0.622	0.552	0.501	0.517
0.2	Počet shluků	25	16	9	5
	Čistota	0.655	0.717	0.791	0.769
	Entropie	2.127	2.028	1.911	2.069
	F-míra	0.345	0.259	0.174	0.094
	Odchylka	0.484	0.510	0.414	0.526
0.3	Počet shluků	12	6	4	3
	Čistota	0.751	0.873	0.897	0.928
	Entropie	2.360	1.747	1.544	1.664
	F-míra	0.221	0.128	0.085	0.073
	Odchylka	0.415	0.399	0.378	0.352

Tab. 5.3: Výsledky alg. MZ na C600 pro prahovou podobnost PP

Algoritmy MK a MZ využívají prahové podobnosti PP k odfiltrování hran s nízkou vahou. Z tabulek 5.3 a 5.4 lze vypozorovat, že počet nalezených shluků je nepřímoúměrný k počtu odfiltrován hran, což není nijak překvapivé, protože průměrný stupeň vrcholu při $PP = 0.3$ je 1.1 (viz kapitola o analýze a zpracování vstupních dat). U alg. MK je zřejmá závislost

parametrů PP a I_t, kdy parametr I_t „vyvažuje“ počet hran v kolekci. Kvalita nalezených shluků algoritmem MZ je zato závislá pouze na parametru I_t.

PP	**Míra**	$I_t = 0.1$	$I_t = 0.3$	$I_t = 0.5$	$I_t = 0.7$
0.1	Počet shluků	30	35	31	31
	Čistota	0.603	0.629	0.677	0.739
	Entropie	2.414	2.287	2.221	1.994
	F-míra	0.346	0.354	0.328	0.321
	Odchylka	0.548	0.576	0.522	0.516
0.2	Počet shluků	16	16	16	13
	Čistota	0.803	0.803	0.806	0.781
	Entropie	1.856	1.856	1.704	1.801
	F-míra	0.190	0.190	0.185	0.157
	Odchylka	0.497	0.497	0.505	0.509
0.3	Počet shluků	5	5	5	4
	Čistota	0.972	0.972	0.972	0.966
	Entropie	0.899	0.899	0.907	0.994
	F-míra	0.076	0.076	0.075	0.064
	Odchylka	0.471	0.471	0.479	0.426

Tab. 5.4: Výsledky alg. MK na C600 pro prahovou podobnost PP

Tabulka 5.5 obsahuje nejlepší výsledek pro daný algoritmus. Všechny algoritmy dopadly mnohem lépe než náhodné shlukování. Znamená to tedy, že v každém případě všechny shlukovací algoritmy přináší do shlukování nějakou přidanou hodnotu. KM dosáhl nejlepších výsledků se shlukováním o 43 shlucích. Výhoda KM je ve větším pokrytí kolekce, ale snižuje se čistota a odchylka shlukování. Naproti tomu algoritmy MK a MZ mají nejlepších výsledků u 16 resp. 9 shluků, které lépe odpovídají svým protějškům ve vzorových shlucích, ale nepokrývají je všechny, což se projevilo v lepší čistotě a odchylce, ale horší entropii a F-míře. Projevila se také převaha MK se sofistikovanějším výpočtem kompaktnosti oproti MZ, který počítá pouze binární zapojení (hrana existuje / neexistuje). Výsledky MK jsou proto ve většině případů lepší než výsledky MZ.

Pro další použití algoritmů pro hledání iniciálních shluků už použijeme pouze výběr parametrů, které dosáhly nadprůměrných výsledků. Všechny výsledky jednoho alg. potom

Míra	**MK**	**MZ**	**KM**	**Rand**
Počet shluků	16	9	43	rand
Čistota	0.806	0.791	0.408	0.268
Entropie	1.704	1.911	3.035	3.543
F-míra	0.185	0.174	0.289	0.133
Odchylka	0.505	0.414	0.717	0.829

Tab. 5.5: Porovnání nejlepších výsledků alg. pro inic. shluky s náhodným shlukováním

sloučíme do jednoho shlukování a prohlásíme ho iniciálním shlukováním daného algoritmu. Pro KM použijeme výsledky pro $c = \frac{|C|}{10}$, $c = \frac{|C|}{7}$, $c = \frac{|C|}{5}$ a shluky obsahující více než tři dokumenty. MZ ponecháme s parametrem I_t rovným 0.3 a 0.5 (pro všechny PP) a pro MK dvojice parametrů $< PP, I_t > < 0.1; 0.7 >, < 0.2; 0.5 >$ a $< 0.3; 0.3 >$. Výstupem MZ i MK už standardně jsou shluky o více než třech dokumentech.

Tabulka 5.6 obsahuje výsledky pro takto sloučené shlukování. I při sloučených shlucích má nejlepší výsledky MK. Pro následný vstup programu LS použijeme shluky z MK a MZ. Pro porovnání budeme nadále používat sloučené shluky KM.

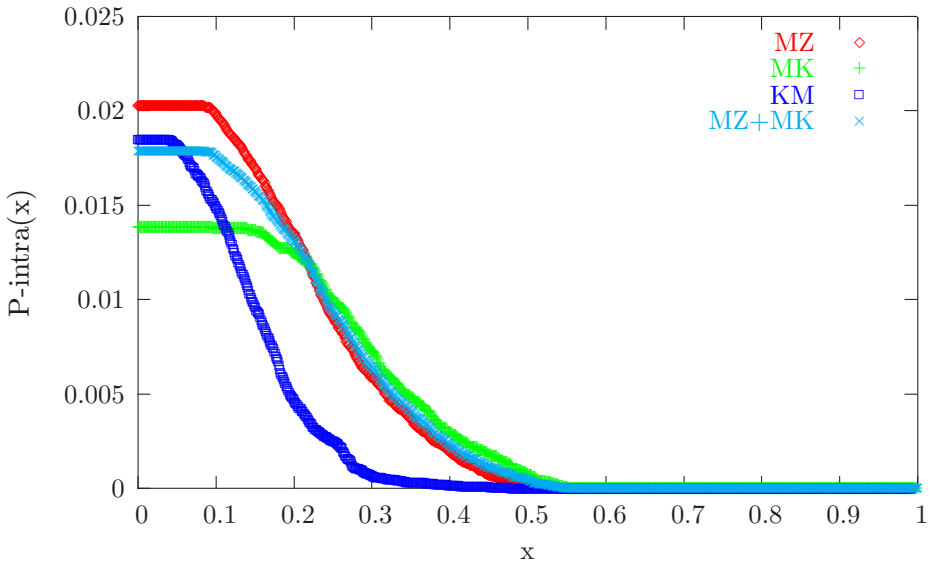
Míra	**MK**	**MZ**	**KM**	**MK+MZ**
Počet shluků	52	87	144	139
Čistota	0.778	0.675	0.426	0.704
Entropie	1.799	2.130	2.944	2.006
F-míra	0.331	0.410	0.318	0.431
Odchylka	0.508	0.497	0.726	0.501

Tab. 5.6: Porovnání výsledků alg. pro inic. shluky pro sloučené nejlepší výsledky

Míry založené na porovnání vnitřní struktury nalezených shluků

Míry P-intra a P-inter zachycují vnitřní strukturu shlukování na kolekci bez potřeby porovnání s nějakými vzorovými shluky. Graf na obr. 5.1 zachycuje výsledky pro míru P-intra, na obr. 5.2 míru P-inter. Obě s krokem 0.001.

Na grafu v obr. 5.1 lze vidět rozdíly v kompaktnosti shlukování pro jednotlivé algoritmy.



Obr. 5.1: Výsledky míry P-intra pro iniciální shluky na C600

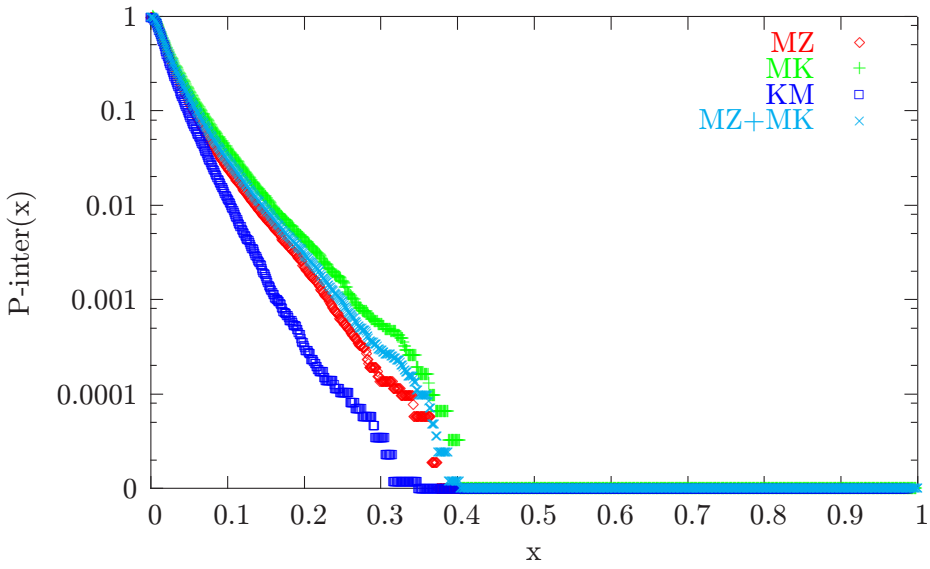
Pro porovnání není příliš důležité, jakou maximální hodnotu má vnitřní míra. Důležitý je tvar křivky. Z grafu lze vyčíst, že KM má v průměru dokument s nejmenší podobností od středu shluku cca 0.05 a největší 0.35. Naproti tomu alg. MK, MZ i jejich sloučení mají ve shlucích nejmenší průměrnou podobnost cca 0.1 (u MK dokonce 1.5) a největší cca 0.5, což jsou hodnoty příznivější (dokumenty ve shluku jsou si podobnější).

Graf na obr. 5.2 zachycuje „překrytí“ shluků, neboli počet dokumentů, jejichž podobnost s jiným shlukem, než ke kterému byl přiřazen, je vyšší než daná mez. Osa y má logaritmické měřítko. Lze vidět, že v tomto pohledu není příliš velký rozdíl mezi jednotlivými algoritmy. Algoritmus KM zde však vychází nejlépe.

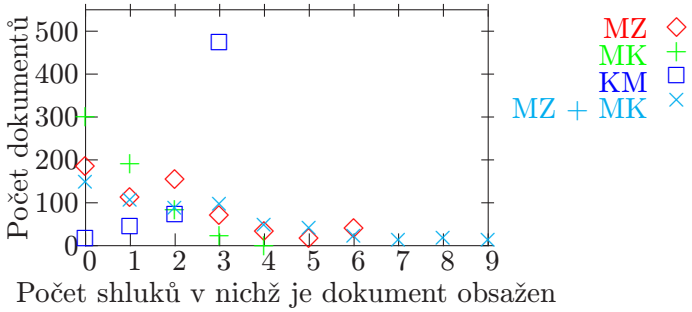
Pokrytí kolekce C600

Graf na obr. 5.3 zobrazuje počty dokumentů, které byly shlukovacími algoritmy přiřazeny do daného počtu shluků. Tj. počet dokumentů (osa y), které byly přiřazeny do právě x shluků (osa x).

Nejméně dokumentů nezařazených do nějakého shluků bylo ve shlukování KM. Naopak nejvíce jich nezařadil MK. KM má většinu dokumentů zařazenu ve třech shlucích, což koresponduje s tím, že toto shlukování je vytvořeno sloučením třech výstupů KM. Každý



Obr. 5.2: Výsledky míry P-inter pro iniciální shluky na C600



Obr. 5.3: Pokrytí kolekce C600

dokument je u MZ průměrně zastoupen v 1.77 shlucích, u MK v 0.72 shlucích a u KM v 2.67 shlucích.

5.2.2 Porovnání algoritmů na C10000

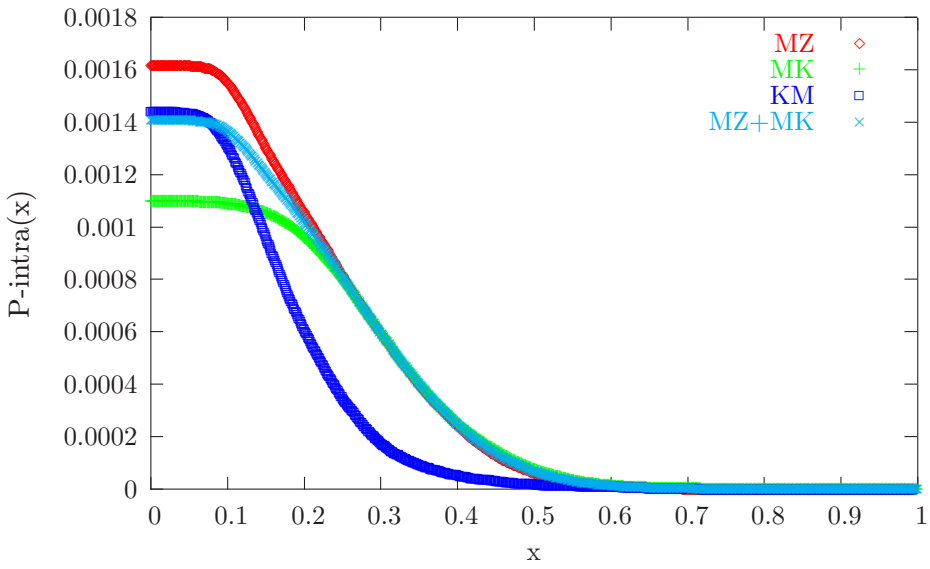
Na C10000 nejsou k dispozici žádné vzorové shluky. Pro porovnání výstupů algoritmů jsou použity vnitřní míry inter- a intra- shluková podobnost. Programy byly spuštěny s nalezenými nejlepšími parametry, které jsou popsány v předcházející kapitole. Počty nalezených shluků ve výstupech jednotlivých algoritmů jsou uvedeny v tab. 5.7.

Metoda		Počet shluků
K-means	[KM]	1910
Max. zapojení	[MZ]	1899
Max. kompaktnost	[MK]	1923

Tab. 5.7: Počet shluků pro shlukování na C10000

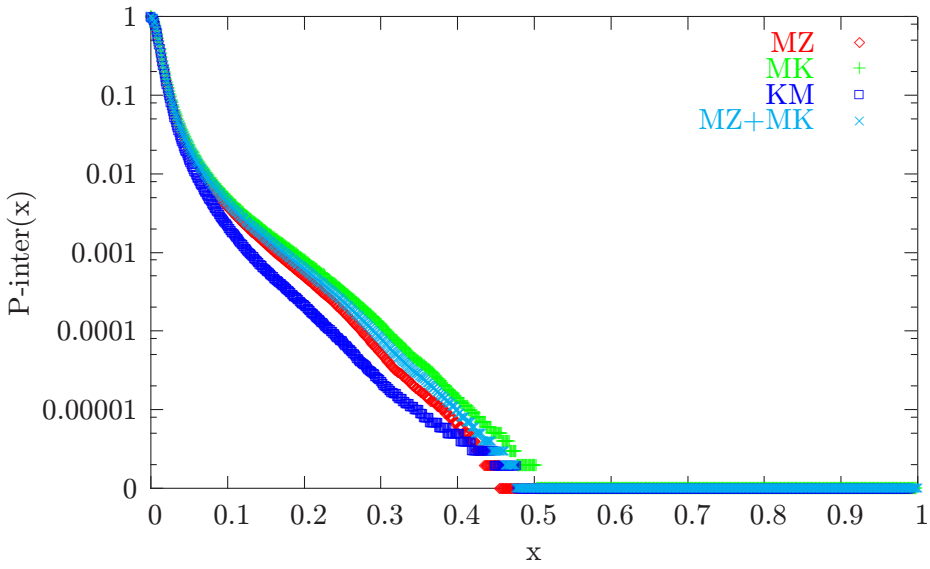
Míry založené na porovnání vnitřní struktury nalezených shluků

Na obr. 5.4 jsou zobrazeny výstupy míry P-intra pro shlukování s krokem 0.001. Všechny algoritmy dosáhly mírně lepšího výsledku než na C600. A stejně jako na C600 je vidět, že i na kolekci C10000 jsou z hlediska kompaktnosti shluků lepší algoritmy MZ i MK včetně jejich sloučení.



Obr. 5.4: Výsledky míry P-intra pro iniciální shluky na C10000

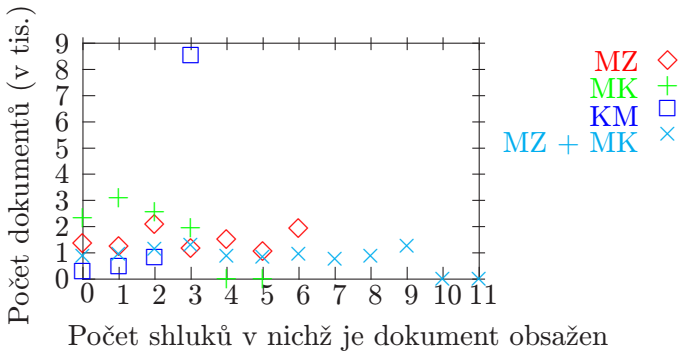
Míra P-inter pro shlukování je zachycena v grafu na obr. 5.5. Rozdíl mezi jednotlivými shlukováními je minimální.



Obr. 5.5: Výsledky míry P-inter pro iniciální shluky na C10000

Pokrytí kolekce C10000

Graf na obr. 5.6 zachycuje pokrytí kolekce C10000.



Obr. 5.6: Pokrytí kolekce C10000

Každý dokument je na C10000 u MZ průměrně zastoupen v 2.40 shlucích, u MK v 1.42 shlucích a u KM v 2.75 shlucích. Průměrné hodnoty jsou pro KM podobné jako na kolekci C600. U MK a MZ se průměrné zastoupení dokumentů ve shluku zvýšilo. Příčinou je menší počet dokumentů, které nebyly zařazeny do žádného shluku, díky většímu počtu dokumentů v kolekci. Minimální velikost přijímaného shluku se totiž nezměnila a stále jsou požadovány alespoň 4 dokumenty ve shluku.

5.2.3 Shrnutí výsledků algoritmů pro nalezení iniciálních shluků

Bylo ukázáno, že navržené algoritmy MK i MZ pro nalezení vhodných iniciálních shluků, mají jako výstup shluky tématicky podobných dokumentů, jejichž kvalita je v porovnání se vzorovými shluky vyšší než u výstupu standardního algoritmu K-means. Projevila se také převaha MK se sofistikovanějším výpočtem kompaktnosti oproti MZ, který počítá pouze binární zapojení (hrana existuje / neexistuje). Výsledky MK jsou proto ve většině případů lepší než výsledky MZ.

Dále bylo pomocí vnitřních měr ukázáno, že výstupy navržených algoritmů mají na kolekci C10000 i C600 podobnou distribuci velikosti hran, tj. že na obou kolekcích byly algoritmy MK i MZ vytvořeny podobně kvalitní shluky z hlediska váhy shluků a jejich zapojení do zbytku kolekce.

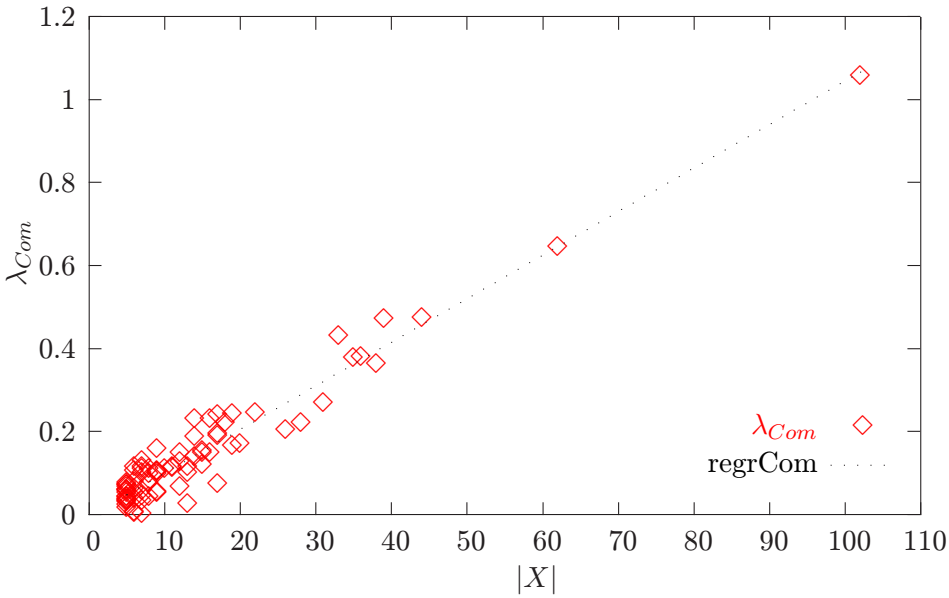
5.3 Natrénování a evaluace λ-parametrů pro LocalSearch

Standardní postup pro trénování parametrů programu je rozdělit trénovací množinu na dvě části. Na jedné podmnožině natrénovat parametry, které dávají nejlepší výstup a na druhé podmnožině tyto natrénované parametry ověřit. My ovšem využijeme toho, že ověření kvality můžeme provést (a provedeme v následující kapitole) na shlucích nalezených algoritmy pro iniciální shluky. Očekávaný výsledek je potom zlepšení kvality původního shlukování. Zbývá tedy ověřit stabilitu nalezených parametrů pro různé trénovací množiny a trénovací podmínky, což provedeme v této kapitole. Nejdříve ale ukážeme, že optimální λ-parametry jsou zhruba lineárně závislé na velikosti shluků, tj. ověříme správnost rovnice $\lambda_i = \lambda_{i0} + \lambda_{i1}|X|$.

5.3.1 Lineární závislost λ-parametrů na velikosti hluku

Základní rovnice pro výpočet kvality konceptotvorného shluku tvar $Q(X) = \prod_i F_i^{\lambda_i}$. Pokud natrénujeme nejlepší hodnoty vždy pouze pro jeden každý trénovací shluk, dostaneme hodnoty λ-parametrů tak, jak jsou zakresleny v grafech na obr. 5.7 a obr. 5.8.

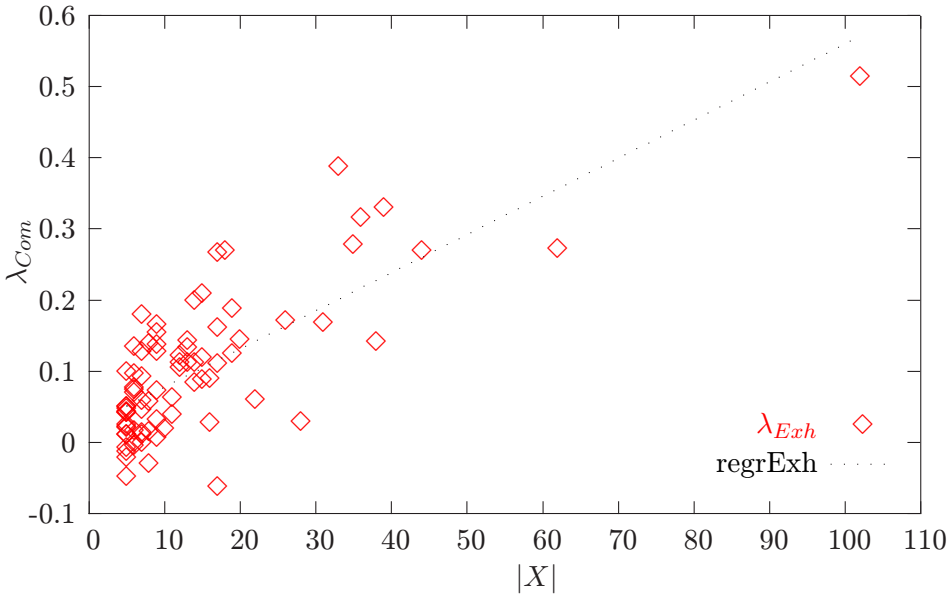
V grafu na obr. 5.7 jsou zobrazeny ideální hodnoty λ_{Com} pro jednotlivé vzorové shluky. Jde vidět lineární závislost těchto parametrů na velikosti shluku, která je znázorněna proložením bodů přímkou metodou lineární regrese. Platí tedy $\lambda_{Com} = \lambda_{Com0} + \lambda_{Com1}|X|$. Nalezené



Obr. 5.7: Závislost λ_{Com} na velikosti shluku

nejlepší hodnoty pro partikulární parametry λ_{Com} jsou: $\lambda_{Com0} = -0.0047$ a $\lambda_{Com1} = 0.0105$.

V grafu na obr. 5.8 jsou zobrazeny ideální hodnoty λ_{Exh} pro jednotlivé vzorové shluky.



Obr. 5.8: Závislost λ_{Exh} na velikosti shluku

Jde vidět lineární závislost těchto parametrů na velikosti shluku stejně jako u λ_{Com}. Nalezené optimální λ-parametry mají ale větší rozptyl než u λ_{Com}. Body je také proložena přímka metodou lineární regrese. Platí tedy $\lambda_{Exh} = \lambda_{Exh0} + \lambda_{Exh1}|X|$. Nalezené ideální hodnoty pro dílčí parametry λ_{Exh} jsou: $\lambda_{Exh0} = 0.0242$ a $\lambda_{Exh1} = 0.0054$.

λ-parametry nalezené trénováním na samostatných shlucích metodou lineární regrese můžeme již použít jako finální. Jde vlastně o použití principu rozděl a panuj, kdy velkou soustavu nerovnic pro trénink λ-parametrů rozdělíme na několik partikulárních úloh a z jejich výsledků vypočítáme výsledek celého problému. Tento postup je mnohem méně časově náročný (řešíme pouze malé soustavy nerovnic). Pokud by rozdíl mezi takto nalezenými λ-parametry oproti řešení přes celou soustavu byl malý, znamenalo by to významné ušetření potřebného výpočetního výkonu.

5.3.2 Optimální počet a typ trénovacích podmínek

Idea natrénování λ-parametrů spočívá ve vytvoření soustavy nerovnic a nalezení výsledku, který ji řeší s nejmenší chybou. Čím více je nerovnic a proměnných, tím více výpočetně náročná je tato úloha. V našem případě máme 4 neznámé a počet rovnic je volitelný. Soustavu nerovnic řešíme programem `lp_solve`, který soustavu o 27699 podmínkách řešil na PC Athlon 1700 MHz cca 29 hodin. Proto se pokusíme snížit potřebný počet nerovnic na minimum, případně zjistit, zda mají všechny podmínky pro akceptovatelné řešení stejnou váhu, tj. zda nemůžeme omezit počet podmínek určitého typu, případně je zcela vynechat.

Závislost natrénovaných λ-parametrů na jedn. typech podmínek je zachycena v tab. 5.8. Měníme pouze podmínky typu (b1) až (b3), protože podmínky (a1) a (a2) jsou nezbytný základní typ podmínek pro tvorbu nerovnic a podmínky typu (bx) jsou jejich vylepšení využívající hlubší znalost trénovacích shluků (určení stupně incidence dokumentu do vzorového shluku ruční anotací). Nejlepší výsledek z hlediska průměrné hodnoty chyby na jednu rovnici má trénování s vynecháním všech podmínek typu b. Největší vliv na zvětšení průměrné chyby mají podmínky typu (b2) a (b3).

Stabilitu natrénovaných parametrů a jejich nezávislost na vybraných trénovacích shlucích můžeme vysledovat v tab. 5.9. V této tabulce jsou natrénovány λ-parametry pro náhodně vybrané podmnožiny trénovacích shluků o stejné mohutnosti 20 shluků. Vidíme, že rozdíl mezi takto natrénovanými parametry je malý.

#a1	#a2	#b1	#b2	#b3	#nerovnic	∅error[1]	λ_{Com1}	λ_{Com2}	λ_{Exh1}	λ_{Exh2}
50	50	50	50	50	14883	0,00387	0.0065	0.0035	-0.0265	0.0107
50	50	0	50	50	12781	0,00391	0.0007	0.0046	-0.0118	0.0063
50	50	50	0	50	11183	0,00336	-0.0015	0.0062	-0.0243	0.0059
50	50	50	50	0	10883	0,00327	-0.0034	0.0074	-0.0066	0.0098
50	50	50	0	0	7183	0,00197	0.0043	0.0094	-0.0450	0.0167
50	50	0	50	0	8781	0,00312	-0.0055	0.0078	-0.0196	0.0100
50	50	0	0	50	9082	0,00322	-0.0020	0.0063	-0.0167	0.0045
50	50	0	0	0	5081	0,00109	0.0020	0.0133	-0.0995	0.0272
200	200	0	0	0	17145	0,00052	0.0103	0.0165	-0.0484	0.0251

Tab. 5.8: Porovnání natrénovaných λ-parametrů pro různý typ nerovnice

č.[2]	#a1	#a2	#b1	#b2	#b3	#nerov.	∅error	λ_{Com1}	λ_{Com2}	λ_{Exh1}	λ_{Exh2}
1	200	200	200	200	200	13345	0.00302	-0.0043	0.0069	-0.0118	0.0040
2	200	200	200	200	200	13712	0.00269	-0.0077	0.0085	-0.0391	0.0039
3	200	200	200	200	200	14216	0.00327	-0.0031	0.0068	-0.0199	0.0035
4	200	200	200	200	200	13257	0.00286	-0.0075	0.0077	-0.0055	0.0019
5	200	200	200	200	200	13638	0.00312	-0.0034	0.0071	0.0008	0.0033

Tab. 5.9: Porovnání stability natrénovaných λ-parametrů pro náhodné trénovací shluky

Další možností, kterou musíme zvážit, je závislost trénovaných parametrů na mohutnosti trénovacích shluků. Proto rozdělíme trénovací shluky na tři množiny $M1$ až $M3$ následovně:

$M1$ množina shluků, pro které platí: $|C_i| \leq 6$ ... celkem 24 shluků
$M2$ množina shluků, pro které platí: $6 \leq |C_i| \leq 14$ ... celkem 31 shluků
$M3$ množina shluků, pro které platí: $14 \leq |C_i|$ ... celkem 25 shluků

λ-parametry natrénované na množinách $M1$, $M2$ a $M3$ jsou zobrazeny v tabulce 5.10. Velká diferenciace hodnot natrénovaných parametrů je způsobena trénováním parametrů na shlucích omezené velikosti. Jak jsme ukázali v sekci 5.3.1, jsou hledané λ-parametry závislé na velkosti shluků. Při rozdělení trénovacích shluků na množiny stejně velkých shluků

[1] Průměrná chyba na jednu nerovnici.
[2] Číslo pokusu.

hladina	#a1	#a2	#b1	#b2	#b3	#nerov.	∅error	λ_{Com1}	λ_{Com2}	λ_{Exh1}	λ_{Exh2}
$M1$	100	100	100	100	100	7164	0.00197	-0.0006	0.0089	0.0724	0.0128
$M2$	100	100	100	100	100	10180	0.00301	-0.0126	0.0081	-0.0484	0.0097
$M3$	100	100	100	100	100	10355	0.00376	-0.0197	0.0065	-0.0518	0.0089

Tab. 5.10: Porovnání stability natrénovaných λ-parametrů pro různě velké trénovací shluky

vlastně napodobujeme důkaz lineární závislosti ze sekce 5.3.1. Proto musí λ-parametry natrénované na těchto množinách vycházet různé.

5.3.3 Natrénování nejlepších λ-parametrů

V předchozí sekci 5.3.2 jsme ukázali, že navržený postup trénování parametrů programu LS je stabilní a nezávislý na náhodně vybrané množině trénovacích shluků. Také jsme ukázali vliv jednotlivých trénovacích podmínek na výsledné parametry a potvrdili lineární závislost λ-parametrů na velikosti shluků.

#a1	#a2	#b1	#b2	#b3	#nerovnic	∅error	λ_{Com1}	λ_{Com2}	λ_{Exh1}	λ_{Exh2}
50	50	50	50	50	14883	0.00387	0.0065	0.0035	-0.0265	0.0107
100	100	100	100	100	27699	0.00384	0.0032	0.0041	-0.0225	0.0093
200	200	200	200	200	52581	0.00371	0.0022	0.0044	-0.0215	0.0084

Tab. 5.11: Porovnání natrénovaných λ-parametrů pro zvyšující se počet nerovnic

V tabulce 5.11 jsou výsledky pro přibývající počet všech podmínek. Hypotéza je, že soustava s největším počtem trénovacích nerovnic podá nejlepší výsledky.

Typ natrénování		λ_{Com1}	λ_{Com2}	λ_{Exh1}	λ_{Exh2}
Lineární regrese	$[LS_{LR}]$	-0.0047	0.0105	0.0242	0.0054
Pouze podmínky (a1) a (a2)	$[LS_{AA}]$	0.0103	0.0165	-0.0484	0.0251
Všechny druhy podmínek	$[LS_{LB}]$	0.0022	0.0044	-0.0215	0.0084

Tab. 5.12: Nejlepší natrénované λ-parametry pro různý typ výpočtu

V předchozí kapitole jsem ukázali, že největší chybu soustavy způsobují trénovací pod-

mínky typu (b). Soustava bez těchto podmínek dosahuje nejmenší průměrné chyby na jednu nerovnici. Pro další výpočty proto pro porovnání použijeme skupiny parametrů natrénovaných s i bez podmínek typu (b). Dále také vyzkoušíme λ-parametry získané lineární regresí. Hodnoty nejlepších dále použitých natrénovaných λ-parametrů jsou v tab. 5.12.

5.4 Výsledky algoritmu LocalSearch

V této kapitole vytvoříme výstupy programu LocalSearch na kolekci C600 a C10000 s natrénovanými λ-parametry. Na kolekci C600 porovnáme výstupy LocalSearch s výstupy alg. pro iniciální shluky pomocí vnějších měr. Na kolekci C10000 použijeme pro porovnání vnitřní míry P-intra a P-inter a popíšeme fuzzy přiřazení dokumentů ke shlukům.

5.4.1 Porovnání alg. na C600

Vstupem LS je shlukování získané algoritmy pro nalezení inic. shluků. Výstupem LS je potenciál každého dokumentu ke shlukům. Potenciál určuje přínos dokumentu ke kvalitě shluku. Řekneme, že dokument patří do shluku, pokud je potenciál kladný. Dokument nepatří do shluku, pokud je jeho potenciál záporný. Pro porovnání alg. pomocí měr čistota, entropie a F-míra použijeme toto jednoduché binární dělení. Pro výpočet odchylky využijeme potenciál jako fuzzy rozdělení s mezí 0.

Alg.	Míra	inic.	LS_{LR}	LS_{AA}	LS_{AB}
MK	Čistota	0.778	0.823	0.872	0.787
	Entropie	1.799	1.037	0.707	1.315
	F-míra	0.331	0.261	0.238	0.345
	Odchylka	0.508	0.001	0.009	0.00014
MZ	Čistota	0.675	0.672	0.657	0.688
	Entropie	2.130	1.309	0.901	1.643
	F-míra	0.410	0.344	0.276	0.397
	Odchylka	0.497	0.002	0.056	0.00011
KM	Čistota	0.426	0.549	0.709	0.517
	Entropie	2.944	1.468	0.757	2.952
	F-míra	0.318	0.342	0.313	0.439
	Odchylka	0.726	0.008	0.473	0.00092
MZ+MK	Čistota	0.704	0.769	0.716	0.718
	Entropie	2.006	1.207	0.828	1.520
	F-míra	0.431	0.356	0.297	0.476
	Odchylka	0.501	0.001	0.039	0.00012

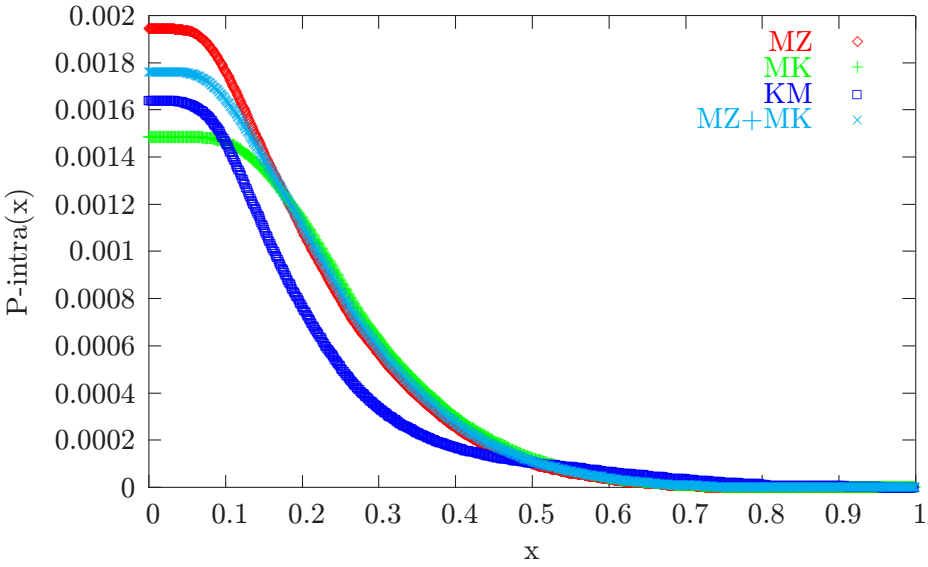
Tab. 5.13: Porovnání sloučených výsledků alg. pro inic. shluky s LS

V tab. 5.13 jsou porovnány výsledky programů MK, MZ a KM pro nalezení iniciálních shluků s výstupem programu LS s různými λ-parametry. Zelenou barvou jsou označeny nejlepší výsledky pro danou množinu shluků. Malé hodnoty u odchylky LS jsou způsobeny malým rozptylem potenciálu kolem 0 - bližší vysvětlení a přiblížení tvaru křivky potenciálů bude v následující kapitole. Je zřejmé, že ve většině případů byl výstup z LS lepší než vstup. LS tedy dělá shlukování kvalitnější. Navíc ve většině případů je lepší výstup LS s parametry natrénovanými všemi podmínkami (a) a (b). Má tedy smysl je trénovat jako velkou soustavu nerovnic i za cenu větší výpočetní náročnosti. Na kolekci C10000 už tedy použijeme pouze LS s λ-parametry natrénovanými ze všech podmínek.

Největší zvýšení kvality je u vstupu z programu K-means. Je to logické vzhledem k lepším výsledkům ostatních algoritmů při hledání iniciálních shluků (což bylo ukázáno v předešlých kapitolách) a vzhledem k předcházejícímu závěru, že LS zkvalitňuje vstupní shluky.

5.4.2 Porovnání alg. na C10000

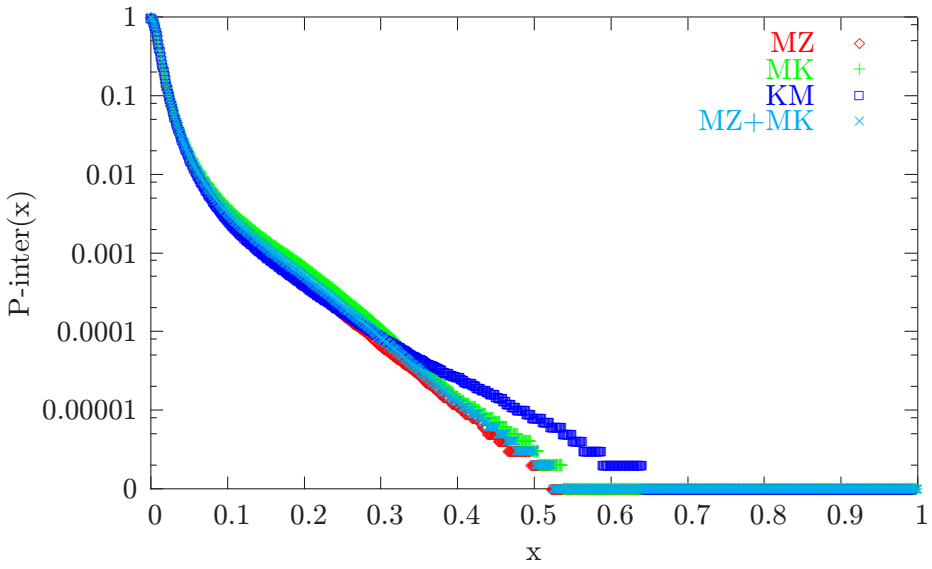
Pro porovnání výsledků na kolekci C10000 použijeme míry P-inter a P-intra.



Obr. 5.9: Výsledky míry P-intra pro výstupní shluky LS na C10000

Na obr. 5.9 je graf míry P-intra pro výsledné shlukování programu LocalSearch na vstupních iniciálních shlucích popsaných v předcházejících sekcích s natrénovanými λ-parametry. Je

patrné, že kvalita všech shluků se zvýšila. Zvýšil se počet dokumentů ve shlucích s vysokou podobností (s podobností 0.5 až 0.6) - tj. upevnila se centra shluků. LS také zlepšil výstup algoritmu K-means, který se přiblížil k hodnotám z ostatních algoritmů pro iniciální shluky. Toto výrazné zlepšení je ale na úkor mírnému zhoršení míry P-inter. Jak můžeme sledovat v grafu na obr. 5.10 zvětšila se váha spojení shluků s dokumenty mimo shluk. Zhoršení je však pouze slabé v porovnání se zlepšením míry P-intra.



Obr. 5.10: Výsledky míry P-inter pro výstupní shluky LS na C10000

5.4.3 Shrnutí výsledků algoritmu LocalSearch

Bylo zjištěno, že LocalSearch dává nejlepší výsledky pro λ-parametry natrénované komparativní metodou pomocí všech typů podmínek. Pomocí vnitřních i vnějších měr bylo ukázáno, že výstupem algoritmu LocalSearch je shlukování, které je kvalitnější než vstupní iniciální shlukování. LocalSearch tedy provádí optimalizaci vstupních shluků pro zvýšení kvality shlukování. Navíc se zlepšila charakteristika distribuce velikosti hran v kolekci, což je patrné z obrázků s grafy míry P-inter a P-intra.

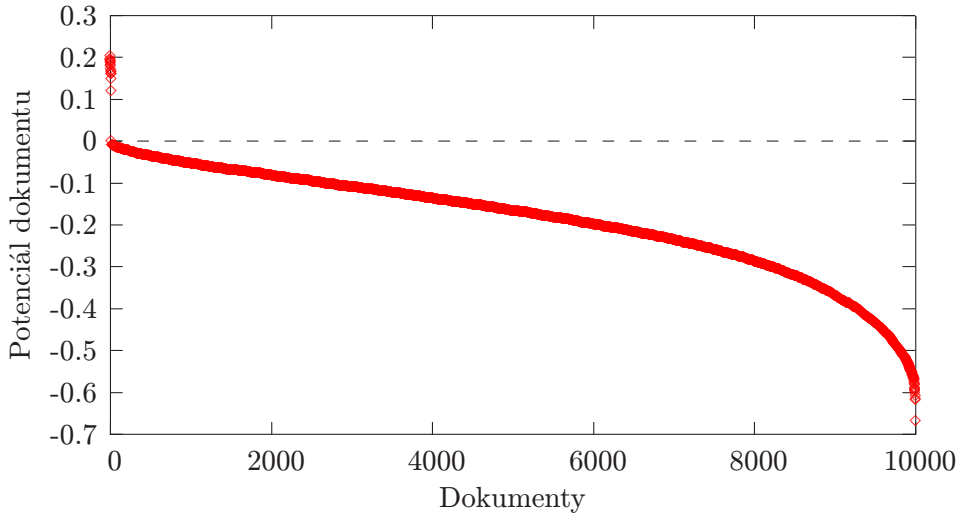
5.4.4 Křivka potenciálu shluku

Program LocalSearch (LS) dává na výstup množinu shluků a pro každý dokument kolekce určí jeho míru náležení do každého ze shluků – určí potenciál dokumentu pro shluk. Výstupem LS je tedy fuzzy množina, podobně jako u algoritmu C-means, který byl uveden v kapitole o nehierarchickém shlukování. Setřidme dokumenty v kolekci podle potenciálu do nerostoucí posloupnosti. Za předpokladu, že do shluku nemůže patřit dokument se záporným potenciálem (což je splněno z definice), jsou v této posloupnosti vždy na začátku dokumenty patřící do shluku a poté následuje zbytek kolekce. Příkladem jsou např. grafy na obr. 5.11, obr. 5.13 a 5.15. Tuto posloupnost bodů proložíme křivkou, kterou nazveme **křivka potenciálu shluku**. U algoritmu C-means tuto křivku nazýváme křivkou incidence. její tvar je blíže popsán v [16]. Od křivky potenciálu LS se však výrazně liší.

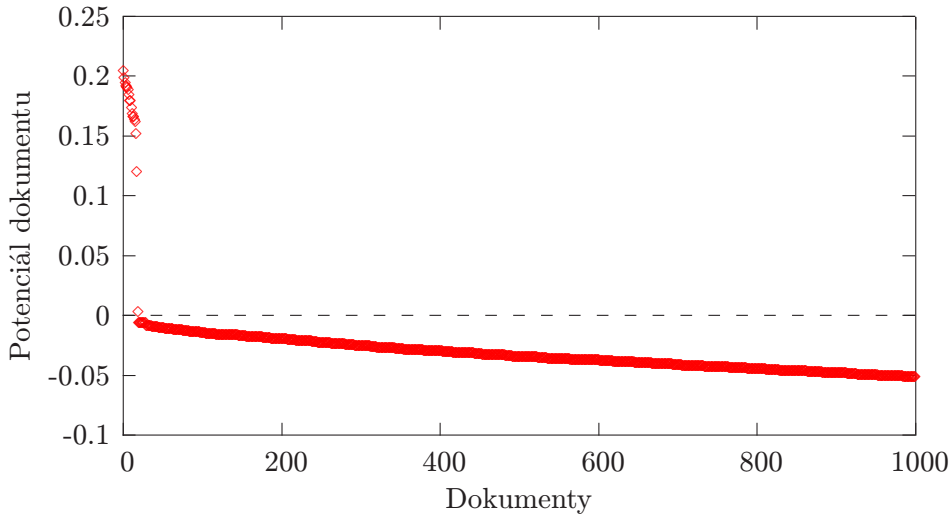
Křivka potenciálu má pro všechny shluky podobný tvar. Nejprve má velký záporný gradient pro dokumenty s kladným potenciálem patřící do shluku. Následuje konec shluku a křivka potenciálu dále klesá pouze pozvolna. Na závěr se záporný gradient křivky opět zvýší. Kvalita shluku se neprojeví na velikosti potenciálu jeho jednotlivých dokumentů, ale na tvaru křivky potenciálu a na vzdálenosti sousedních bodů na této křivce. Pokud do shluku patří mnoho dokumentů a není jasně oddělen od zbylých dokumentů kolekce, je rozdíl potenciálů dvou libovolných sousedních dokumentů přibližně stejný pro všechny sousedící dvojice). Navíc křivka potenciálu na hranici shluku klesá pouze mírně. Pokud však shluk obsahuje pouze několik dokumentů a je ostře oddělen od zbytku kolekce, je patrný skok v potenciálu mezi dokumenty shluku a zbývajícími dokumenty. Vztah shluku ke zbývajícím dokumentům v kolekci se dále odráží ve velikosti gradientu zbývající části křivky. Blíže nám to osvětlí pohled na tři shluky nalezené v kolekci C10000.

Příkladem malého jasně ohraničeného shluku je shluk č. 1 na obr. 5.11. Na obr. 5.12 je výřez z křivky pro prvních 1000 dokumentů. Shluk obsahuje 19 dokumentů vztahujících se k předpovědi počasí. Všechny s výjimkou posledního jsou krátké předpovědi, pouze poslední dokument s názvem *Změna nebo opakování historie* je obsáhlejší a pojednává o vývoji počasí od 17. století do současnosti. Následuje dokument, který již nebyl zařazen do shluku a to báseň Jana Nerudy *Modlitba*, ve které „noc se mění v denní jas“, který již opravdu nemá s počasím mnoho společného.

Je tedy zřejmé, že v tomto případě se jedná o přesně definovaný shluk, který již nemůže být dále rozšířen. Otázkou zůstává zařazení posledního dokumentu do shluku, tj. zařazení dokumentu o historii počasí do shluku o předpovědi počasí. Nabízí se hypotéza, zda by



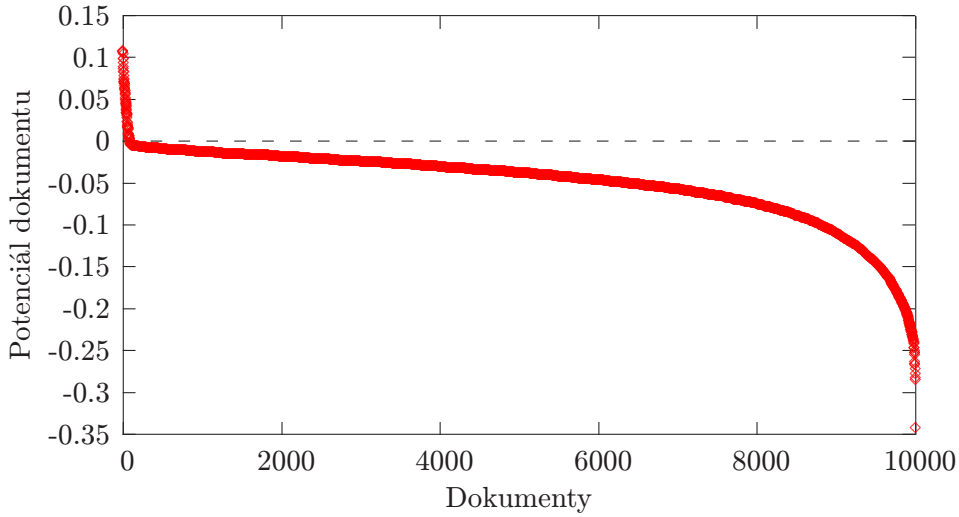
Obr. 5.11: Křivka potenciálu výsledného shluku 1.



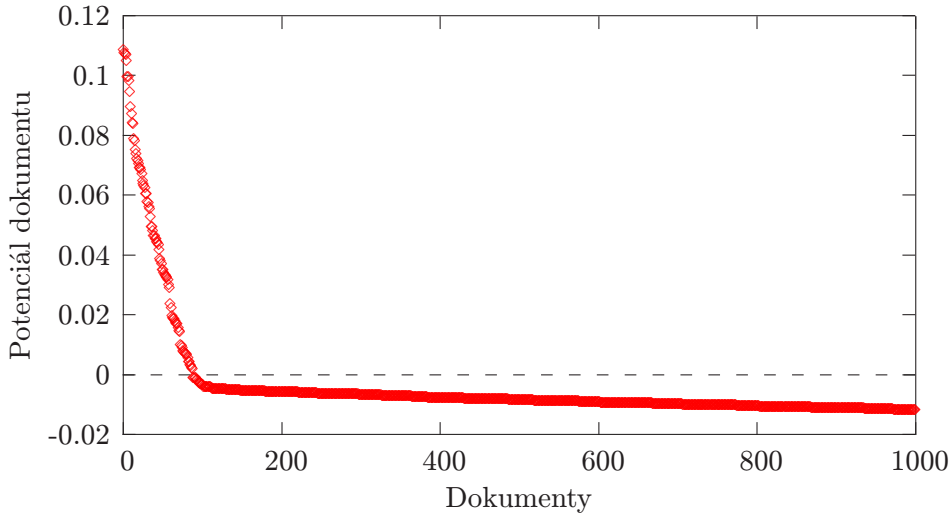
Obr. 5.12: Prvních tisíc dokumentů ze shluku 1.

nebylo lepší určovat hranici výsledného shluku podle křivky potenciálu shluku. Např. zvolit hranici shluku v místě s největším rozdílem dvou sousedních potenciálů.

Shluk č. 2 na obr. 5.13 již není tak jasně oddělen od zbývající kolekce, přesto je oddělení ještě patrné. Velikost shluku je 88 dokumentů. Všechny dokumenty jsou relativně obsáhlé,



Obr. 5.13: Křivka potenciálu výsledného shluku 2.

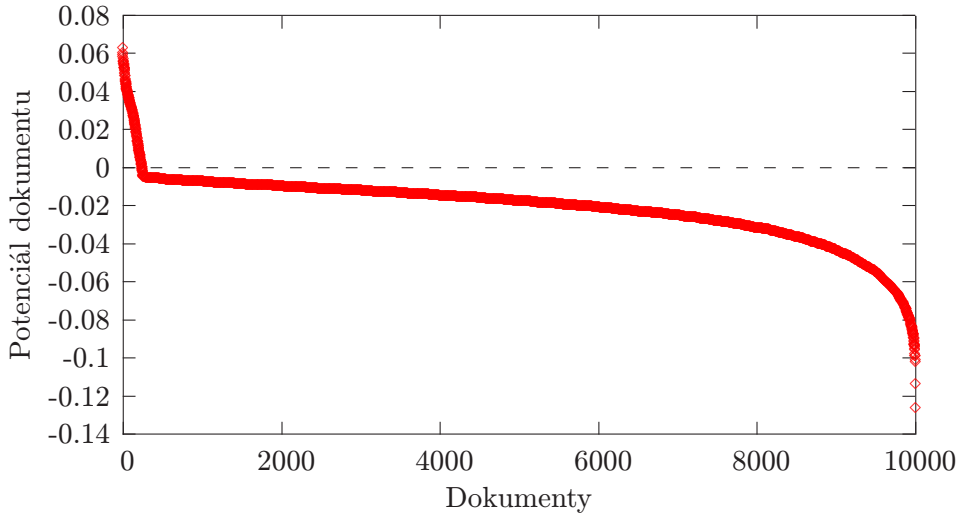


Obr. 5.14: Prvních tisíc dokumentů ze shluku 2.

je tedy snadnější nalézt nějakou podobnost s dokumenty zbývající v kolekci. Názvy prvních dokumentů ve shluku jako *Převody družstevních bytů v etapách*, *Návrh zákona družstvům vyhovuje*, *Otazníky kolem převodu vlastnictví k družstevním bytům zatím zůstávají* definují obsah shluku na dokumenty vztahující se k bytové otázce, přesněji k družstevnímu vlastnic-

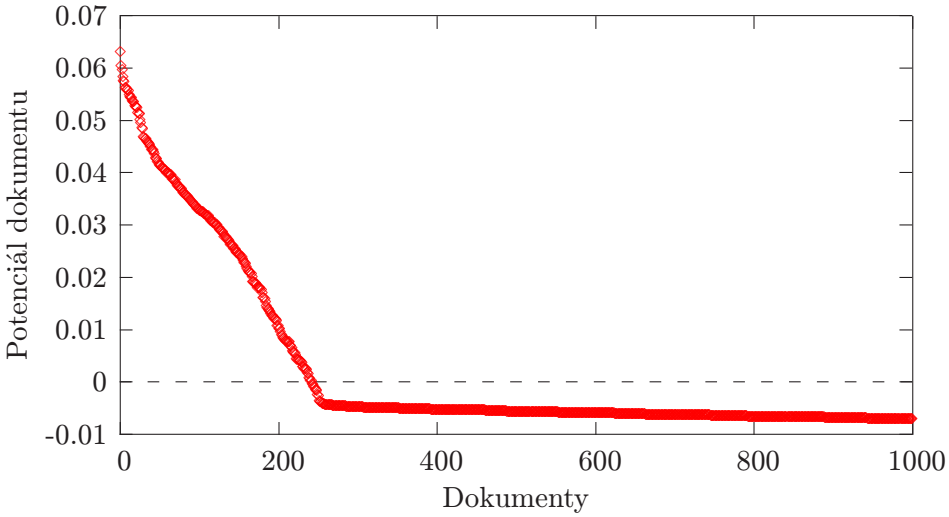
tví. Shluk končí dokumenty o deregulaci nájemného. Obsahuje také dokumenty týkající se cen bytů na trhu. Také pokud se podíváme na podrobnější vykreslení křivky potenciálu na obr. 5.14, zjistíme, že dokumenty zařazené do shluku tvoří dále několik menších viditelně oddělených podshluků. Opět se tedy nabízí otázka o využití informace o rozdílu sousedních potenciálů pro jiné určení hranice shluku, případně rozdělení shluku na více podshluků.

Poslední třetí uvedený shluk, je příkladem velkého ne příliš jasně odděleného shluku. Celkem je do shluku zařazeno 242 dokumentů. Dokumenty jsou relativně krátké, vztahují se však k nejdiskutovanějšímu tématu doby, ze které jsou dokumenty této kolekce. Hlavním tématem je válka v bývalé Jugoslávii.
Je zajímavé, že i přes konec shluku křivka potenciálu pokračuje beze změny dál a změna nastává až po následujících 15 dokumentech. Tyto dokumenty s názvy *Umělci se obávají o Sarajevany*, *Sarajevské děti mají za sebou jednadvacet měsíců války* nebo *Bída a utrpení sousedí na Balkáně s luxusem* se také vztahují k válce v bývalé Jugoslávii, ale spíše „okrajově“. Je otázkou, zda v tomto případě je do shluku také nepřiřadit – tj. zvolit hranici shluku až v prudké změně sklonu křivky.



Obr. 5.15: Křivka potenciálu výsledného shluku 3.

Výše jsme uvedli tři typické příklady shluků a jejich křivky potenciálu. Ukázali jsme, že existují i jiné způsoby určení hranice shluku (podle křivky potenciálu) a nastínili v jakých situacích je použít. Nabízí se tedy cesta dalšího vývoje algoritmu, a to poslední



Obr. 5.16: Prvních tisíc dokumentů ze shluku 3.

optimalizační krok, ve kterém by se zkoumal tvar křivky a vzdálenosti sousedních bodů a určila se konečná hranice shluku. To budiž předmětem dalšího bádání.

Kapitola 6

Závěr

V této práci jsme rozšířili definici konceptotvorného shluku z [1] a navrhli algoritmus pro nalezení tohoto shluku pomocí nalezení iniciálních shluků a jejich následnou optimalizací programem LocalSearch. Pro nalezení iniciálních shluků byly definovány dva algoritmy a jejich výsledky porovnány s algoritmem K-means. Bylo zjištěno, že oba navržené algoritmy dávají kvalitnější výsledky v porovnání se vzorovými shluky než algoritmus K-means. Pro ověření byly použity vnější a vnitřní míry z teorie shlukování. Dále byly natrénovány nejlepší možné λ-parametry pro LocalSearch a ověřena hypotéza jejich lineární závislosti na velikosti shluku. Toto natrénovaní soustavou nerovnic bylo porovnáno s natrénováním pomocí lineární regrese. Bylo ukázáno, že celá soustava nerovnic přináší lepší výsledky než aproximované řešení. Ovšem za cenu velké výpočtové náročnosti. V neposlední řadě bylo ukázáno, že algoritmus LocalSearch optimalizuje vstupní shluky pro zvýšení kvality shlukování a to na malé anotované kolekci i na rozsáhlé kolekci českých novinových článků. Navržený postup tedy vede k získání koherentních konceptotvorných shluků. Na závěr byl popsán výstup programu LocalSearch – křivka potenciálu pro výstupní shluky LocalSearch a byl ukázán vliv tvaru křivky potenciálu shluku na kvalitu shluku.

Během práce bylo otevřeno mnoho možností v pokračování této práce. Jednou z nich je navrhnout nové algoritmy na hledání iniciálních shluků, případně porovnat jako vstup pro LocalSearch více různých standardních shlukovacích algoritmů. Další možností je zlepšovat model konceptotvorného shluku, tzn. upravit funkci pro výpočet kvality shluku přidáním dalších požadovaných vlastností shluku F_i. Velkou oblastí pro možné změny je program LocalSearch. Např. přidání více typů podmínek pro trénování λ-parametrů, případně zkoumat vlastnosti této soustavy nerovnic. Také je možné vyzkoušení úprav algoritmu LocalSearch, jak bylo nastíněno při jeho definici, a jejich vzájemné porovnání. Další možnou oblastí

je i zkoumání a využití křivky potenciálu pro přesnější určení hranice shluku, případně její transformace na matematicky snáze definovatelnou křivku. V neposlední řadě je to pak aplikace navrženého postupu hledání konceptotvorných shluků pro účely dotazovacích systémů a vyhledávání informací v rozsáhlých textových kolekcích.

Příloha A

Obsah přiloženého DVD

Tato kapitola slouží pro popis dat na přiloženém DVD a rychlou orientaci na médiu.

Data

Všechna vstupní data potřebná k provedení experimentu jsou uložena v adresáři \DATA\. Kolekce 10 000 českých dokumentů C10000 je v \DATA\C10000\, kolekce C600 v \DATA\C600\. Soubory jsou ve formátu tagged, tak jak jsme je dostali k dispozici. Po přeparsování dokumentů do tvaru potřebném pro další zpracování programem `parse` je možné použít program `gettext` s parametrem texttttext_id pro získání dokumentu s identifikátorem `text_id` v pro člověka čitelném formátu. Informace o anotaci kolekce C600 – přiřazení dokumentů do shluků a hodnota Φ – je v adresáři \DATA\C600\ANOTACE\.

Dokumentace a texty

Veškeré dokumenty jsou uloženy v adresáři \DOKUMENTY\. Jde zejména o stručnou uživatelskou dokumentaci k programům a text diplomové práce. Formát dokumentů je pdf. Tato diplomová práce byla psána v systému LaTeX. Zdrojové soubory textu diplomové práce jsou uloženy v adresáři \DOKUMENTY\LATEX\.

Zkompilované programy

Programy potřebné pro hlavní výpočet jsou uloženy v adresáři \PROGRAMY\. Programy jsou zkompilovány a jsou spustitelné na platformě `WINDOWS`. Všechny programy při spuštění se

špatnými (případně žádnými) parametry vypíší krátkou nápovědu s popisem správných parametrů.

Zdrojové kódy programů

Zdrojové soubory programů se nalézají v adresáři \ZDROJ\. Všechny programy jsou psány v jazyce C++ a překládány ve WINDOWS kompilátorem GCC. To umožnilo psát zdrojové soubory kompilovatelné a spustitelné jak na platformě WINDOWS, tak na platformě UNIX (překlad např. pomocí kompilátoru CC).

Příloha B

Uživatelská dokumentace k programům

V této kapitole jsou stručně popsány veškeré potřebné programy pro provedení všech experimentů a jejich parametry. Programy jsou rozděleny do sekcí podle jejich určení.

Programy pro hlavní výpočet

V této sekci jsou popsány programy, které jsou nutné pro provedení základního experimentu, tzn. získat ze vstupních souborů konceptotvorné shluky a určit potenciál každého dokumentu ke všem shlukům.

Na začátku je k dispozici adresář `DataIn` s tagged soubory a souborem `seznam.txt`, ve kterém je seznam všech souborů v adresáři (typicky je získán příkazem „`ls > seznam.txt`“ na platformě UNIX / Linux nebo příkazem „`dir /B > seznam.txt`“ na platformě Windows). Dále je vytvořen prázdný adresář `DataOut`. Programy jsou řazeny v pořadí, ve kterém se musí spustit za sebou. Všechny jsou uváděny i s příkladem parametrů, jenž (pokud je dodržen sled programů) zaručují správný průběh celého výpočtu.

1. `parse <vstupní_adresář> <výstupní_adresář>`
    - Příklad: `parse DataIn DataOut`
    - Funkce: přeparsuje soubory z adresáře `DataIn` a uloží je do adresáře `DataOut` a vytvoří slovník.

- Výstupní soubor: `wordBook.txt`
- Formát výstupního souboru: pro každé slovo řádek ve formátu: slovní druh # hodnota CF (collect. freq.) # slovo #
- Poznámka: ve vstupním adresáři DataIn musí být soubor "seznam.txt" se jmény všech souborů v adresáři, které vstupují do výpočtu.

2. `order <soubor_se_slovníkem>`

   - Příklad: `order wordBook.txt`
   - Funkce: Setřídí slovník podle CF a přiřadí slovům jednoznačné ID (= číslo řádky)
   - Výstupní soubor: `orderedWordBook.txt`
   - Formát výstupního souboru: id slova (čísluje se od nuly)# CF# slovní druh# slovo#

3. `DForder <adresář_s_přeparsovanými_texty> <setříděný_slovník>`

   - Příklad: `DForder DataOut orderedwordBook.txt`
   - Funkce: do setříděného slovníku podle CF přidá DF (doc. frequency).
   - Výstupní soubor: `DFwordBook.txt`
   - Formát výstupního souboru: id slova# CF (collection freq.)# DF (document freq.)# slovní druh# slovo#
   - Poznámka: ve vstupním adresáři `DataOut` musí být soubor `seznam.txt` se jmény všech souborů v adresáři, které vstupují do výpočtu.

4. `wordFilter [DF options] <soubor_se_slovníkem>`

   - Příklad: `wordFilter -n5 -N2000 -a2 -K40 -v3 -V20000 DFwordBook.txt`
   - Funkce: vyfiltruje slova ze slovníku podle DF. Ponechá slova odpovídajícího slovního druhu s DF ze zadaného intervalu (interval včetně mezí). Výstup setřídí podle DF.
   - „DF options" parametry: pro každý slovní druh je jedno písmeno. Malé písmeno značí dolní mez pro DF, velké písmeno horní mez.
     Slovní druhy:
     - n,N . . . podstatná jména

  - a,A ... přídavná jména
  - v,V ... slovesa
  - d,D ... příslovce
  - c,C ... číslovky
  - k,K ... kolokace
  - x,X ... lematizátorem nerozpoznané
- Výstupní soubor: `finalWordBook.txt`
- Formát výstupního souboru: číslo_řádku;DF;CF;slovo
- Poznámka: pokud není pro nějaký slovní druh zadána žádná mez, slova tohoto slovního druhu zůstají ve slovníku všechna. Vypuštění slovního druhu je možné zadáním nulové horní meze (např. N0).

5. `vectors <adresář_s_přeparsovanými_texty> <profiltrovaný_slovník>`
   - Příklad: `vectors DataOut finalWordBook.txt`
   - Funkce: vytvoří soubor s vektory incidence slov a dokumentů.
   - Výstupní soubory: `vectors.txt` (soubor s vektory) a `docID.txt` (soubor se jmény souborů a přiřazenými ID)
   - Formát výstupního souboru `vectors.txt`: <číslo_vektoru>; ID_slova1 # incidence1; ID_slova2 # incidence2;... (jen nenulové složky)
   - Formát výstupního souboru `docID.txt`: ID_dokumentu_ve_vectors.txt;jméno_souboru
   - Poznámka: ve vstupním adresáři `DataOut` musí být soubor `seznam.txt` se jmény všech souborů v adresáři, které vstupují do výpočtu.
6. `similarity <počet_slov> <prahová_podobnost> <jméno_souboru_s_vektory>`
   - Příklad: `similarity 25000 0.2 vectors.txt`
   - Funkce: spočítá podobnosti dokumentů (vytvoří graf kolekce) jako normovaný skalární součin vektorů dokumentů.
   - Výstupní soubor: `sim_PP.txt` , kde PP je prahová podobnost (napr. `sim_0.2.txt`)
   - Formát výstupního souboru: ID_vektoru1;ID_vektoru2;hodnota_podobnosti
7. `initClusterCom <počet_dokumentů> <prahová_podobnost> <min_velikost_shluku> <poměr> <soubor_s_podobnostmi>`

- Příklad: `initClusterCom 20000 0.3 10 0.5 sim_0.2.txt`
- Funkce: vypočítá iniciální shluky. Parametr „poměr“ určuje minimální povolený poměr kompaktnosti prvního shluku a kompaktnosti tohoto shluku po rozšíření.
- Výstupní soubor: `iClustersCom_0.3_10_0.5.txt` (slození jména ze zadaných parametrů)
- Formát výstupního souboru: <číslo_shluku>;číslo_dokumentu1;číslo_dokumentu2;...

8. `LocalSearch [options]`
   - Příklad: `LocalSearch -pit -c ls.cfg`
   - Funkce: lokálně optimalizuje iniciální shluky. Protože možnosti a nastavení parametrů nejsou triviální, je programu věnována samostatná kapitola.

Program LocalSearch (LS)

Funkce programu LocalSeach

LS má 4 základní funkce pro libovolný zadaný shluk X:

1. Vypočítat $Com(X), Ext(X), Exh(X), Hom(X)$ a $Q(X)$.
2. Najít lokálně optimální $X^\star$.
3. Vypočítat potenciál, kvalitu resp. hodnotu μ všech dokumentů. Tj. $P(d, X)$ a $P(d, X^\star)$, $Q(X \cup \{d\})$ a $P(X^\star \cup \{d\})$.
4. Poskytnout údaje potřebné pro sestavení komparativních podmínek pro trénování λ-parametrů.

Volba základní činnosti programu (co zapíše do výstupního souboru) je pomocí přepínačů.

Přepínače:

- **p** dává na výstup potenciály dokumentů;
- **q** dává na výstup kvalitu shluku po přidání (resp. ubrání) dokumentů;
- **o** lokálně optimalizuje vstupní shluky;
- **t** dává na výstup údaje pro tvorbu komparativních podmínek;

– **i** vypíše na výstup množinu X;

– **s** vypíše na výstup hodnoty $Com(X), Ext(X), Exh(X), Hom(X)$ a $Q(X)$;

– **c <filename>** použije zadaný konfigurační soubor

Přepínače jsou ve standardu UNIXu, mohou se sdružovat. S výjimkou parametru t je pro každý parametr vypsán jeden řádek do výstupního souboru. Např. při zadání parametru -pq se vypíše nejdříve řádek s potenciály a následně řádek s kvalitou. Každý řádek začíná řetězcem „<id_shluku>[xx];“, kde xx je p, q, o, t, i nebo s podle parametru, pro který je řádek vypsán. Příklad spouštění programu: „`LocalSearch -p -i -t -c soubor.cfg`“ nebo „`LocalSearch -pit -c soubor.cfg`“

Vstupy a výstupy

Vstupy:

1. *Soubor s podobnostmi dokumentů* (tj. ohodnocení hran v kolekci)
   Na každém řádku je jedna hrana. Nulové hrany jsou vynechány.
   Formát řádky: id_dokumentu1;id_dokumentu2;váha_hrany

2. *Soubor s iniciálními shluky*
   Na každém řádku je jeden shluk. ID shluku je libovolné číslo $\in N_0$.
   Formát řádky: <id_shluku>;id_dokument1;id_dokument2;id_dokument3; ...

3. *Konfigurační soubor*
   Popis a struktura konfiguračního souboru a klíčových slov je vysvětlena v samostatné kapitole.

Výstupy:

1. *Logovací soubor s ladícími hláškami*
   Podrobnost výpisů lze nastavit v konfiguračním souboru klíčovým slovem tunnig_level.

2. *Soubor celkové statistiky výpočtu LS*
   Statistika výpočtů LS. Pro každý shluk je zde uvedeno:
   - seznam id dokumentů z množiny X po načtení
   - počet provedených operací move_in
   - počet provedených operací move_out
   - seznam id dokumentů z množiny X po skončení výpočtu

3. *Hlavní výstupní soubor.*

Formát výstupu se liší podle zadaných přepínačů. Při více zadaných přepínačech se nejdříve vypíší všecha data pro jeden shluk a všechny přepínače, pak následuje další shluk. Každý přepínač vypisuje data na samostatný řádek. Výjimka je přepínač -t, který pro každou komparativní podmínku vygeneruje 2 řádky.

Formát výstupu pro jednotlivé přepínače:

- **Přepínač -i**
  - Vypíše X. Pokud je současně zadán přepínač -o, vypíše $X^\star$
  - Formát: <id_shluku>[i];id_dokument1;id_dokument2; ...
- **Přepínač -s**
  - Vypíše hodnoty $Hom(X)$, $Exh(X)$, $Com(X)$, $Ext(X)$ a kvalitu shluku X.
  - Formát: <id_shluku>[s];Ext: hodnota; Com: hodnota; Exh: hodnota; Hom: hodnota; Q(X): hodnota
- **Přepínač -o**
  - Nalezne $X^\star$ a vypíše hodnoty $Hom(X^\star)$, $Exh(X^\star)$, $Com(X^\star)$, $Ext(X^\star)$ a kvalitu shluku $X^\star$.
  - Formát: <id_shluku>[o];Ext: hodnota; Com: hodnota; Exh: hodnota; Hom: hodnota; Q(X): hodnota
- **Přepínač -p**
  - Vypíše $P(d, X)$ pro všechny dokumenty. Pokud je současně zadáno -o, pak vypíše $P(d, X^\star)$.
  - Formát: <id_shluku>[p];id_dokument1#membership1;id_dokument2#membership2; ...
- **Přepínač -q**
  - Vypíše $Q(X \cup \{d\})$ pro všechny dokumenty. Pokud je současně zadáno -o, pak vypíše $Q(X^\star \cup \{d\})$.
  - Formát: <id_shluku>[q];id_dokument1#kvalita1;id_dokument2#kvalita2; ...
- **Přepínač -t**

– Vypíše data pro vytvoření komparativních podmínek. Každá podmínka je na dvou řádcích začínající '*' a '+'.
– Formát:
<id_shluku1>hodnota_Com;hodnota_Exh;hodnota_Ext;hodnota_Hom;
⋆a1[75];hodnota_Com;hodnota_Exh;hodnota_Ext;hodnota_Hom;
+a1[75];hodnota_Com;hodnota_Exh;hodnota_Ext;hodnota_Hom;
⋆b2[15];hodnota_Com;hodnota_Exh;hodnota_Ext;hodnota_Hom;
+b2[34];hodnota_Com;hodnota_Exh;hodnota_Ext;hodnota_Hom;
⋮
<id_shluku2>hodnota_Com;hodnota_Exh;hodnota_Ext;hodnota_Hom;
⋆a1[49];hodnota_Com;hodnota_Exh;hodnota_Ext;hodnota_Hom;
+a1[49];hodnota_Com;hodnota_Exh;hodnota_Ext;hodnota_Hom;
⋮

Popis konfiguračního souboru

Formát:

Komentáře jsou vždy celořádkové a začínají '//'. Řádky s klíčovým slovem začínají znakem '#'. Následující řádek je vždy hodnota pro předcházející klíčové slovo. Mezi klíčovým slovem a hodnotou nesmí být komentář.

Příklad:

// komentář 1
// komentář 2
#klíčové_slovo1
hodnota1
// komentář 3
#klíčové_slovo2
hodnota2
⋮

Klíčová slova

- **similarity_file_name**
  jméno vstupního souboru s podobnostmi dokumentů
- **log_file_name**
  jméno výstupního logovacího souboru

- **out_file_name**
  jméno výstupního souboru

- **init_cluster_file_name**
  jméno vstupního souboru s inicialními shluky

- **annotated_cluster_file_name**
  jméno vstupního souboru s anotovanými dokumenty

- **number_of_documents**
  počet dokumentů v kolekci (označ. N)

- **min_edge_weight**
  prahová podobnost (označ. w_t)

- **potential_threshold**
  prahový potenciál (označ. P_t)

- **exh_type**
  hodnota 1,2, určuje typ výpočtu Exh

- **com_type**
  hodnota 1,2, určuje typ výpočtu Com

- **logBaseExt_constant**
  konstanta pro výpočet Ext

- **logBaseExh_constant**
  konstanta pro výpočet Exh

- **kappaExh_constant**
  konstanta pro výpočet Exh

- **lCom_constant_0**
  konstanta λ_{10} pro výpočet Com

- **lCom_constant_1**
  konstanta λ_{11} pro výpočet Com

- **lExh_constant_0**
  konstanta λ_{20} pro výpočet Exh

- **lExh_constant_1**
  konstanta λ_{21} pro výpočet Exh

- **move_out_first**
  pokud se rovná 1, pak LS nejdříve ubírá prvky z X, a až následně přidává z Y, jinak naopak

- **move_first_good**
  pokud se rovná 1, pak LS vybere pro přesun (operace move_in a move_out) první prvek s potenciálem menším/větším 0, jinak vybírá min/max

- **tunnig_level**
  hodnota 0..4, určuje počet ladicích hlášek

- **needed_quality**
  0..standardní výpočet kvality shluku, jinak výpočet kvality podle absolutních vzorů

- **max_Y_count**
  maximální velikost Y, při které LS končí výpočet

- **lambda_count_a1**
  maximální počet vygenerovaných podmínek typu a1

- **lambda_count_a2**
  maximální počet vygenerovaných podmínek typu a2

- **lambda_count_b1**
  maximální počet vygenerovaných podmínek typu b1

- **lambda_count_b2**
  maximální počet vygenerovaných podmínek typu b2

- **lambda_count_b3**
  maximální počet vygenerovaných podmínek typu b3

Programy pro trénování lambda parametrů

Natrénování lambda parametrů vychází z komparativních podmínek, které vytvoří LocalSearch. Proces natrénování dále probíhá ve dvou krocích:

1. `convert_LStoLPSolve.exe <použití_homogenity(0/1)> <prahové_$\Phi_t$> <PP> <výstup_LS> <ručně_anotované_shluky>`
   - Příklad: `convert_LStoLPSolve.exe 1 2 0.01 LocalSearchOut.txt anot.txt`
   - Funkce: z výstupního souboru LS vytvoří podmínky (nerovnice) pro program `lp_solve`.
   - Výstupní soubor: `lp_solve.lp`
   - Formát výstupního souboru: viz specifikace programu `lp_solve`
2. `lp_solve <soubor_s_podmínkami_lp>`
   - Příklad: `lp_solve lp_solve.lp`
   - Funkce: GNU program, který spočítá úlohu lineárního programování zapsanou v `lp_solve.lp`.
   - Výstup: na STDOUT vypíše hodnoty proměnných
   - Formát výstupu: viz specifikace programu `lp_solve`

Programy pro vykreslení grafů křivky potenciálu

Pro kreslení grafů je využíván GNU program GNUPlot. Potřebujeme tedy pouze přetransformovat data z výstupu LS na vstupní data pro GNUPlot.

`Incidence_GNUPlot <způsob_třízení(0|1|2)> <soubor_s_potenciály>`

- Příklad: `Incidence_GNUPlot 0 LocalSearchOut.txt`
- Funkce: překonvertuje výstupní soubor z LS na datové vstupy pro program `GNUPlot` - pro každý shluk jeden soubor, dále vytvoří soubory s makry pro `GNUPlot` a dávkový soubor pro spuštění všech maker a vytvoření grafů.
- Výstupní soubory: `dataX.txt, macroX.txt, GNUPlot_incidence.bat`, kde X je číslo shluku.

- Formát výstupních souborů: viz dokumentace programu `GNUPlot` a dávkových souborů pro platformu Windows.

Pro názornost uvedeme posloupnost příkazů s parametry pro vykreslení grafu křivky potenciálu:

```
LocalSearch -op -c ls.cfg
Incidence_GNUPlot 0 LocalSearchOut.txt
GNUPlot_incidence.bat
```

Programy pro evaluaci výsledků

Programy pro evaluaci jsou implementací vnitřních a vnějších měr.

1. `eval <velikost_kolekce> <počet_shluků> <soubor_shluků> <soubor_vzorových_shluků> <fuzzy> <mez>`
   - Příklad: `eval eval 600 200 iclustersCom.txt anot_clusters.txt 0 0.5`
   - Funkce: Spočítá hodnotu vnějších měr pro shlukování ze souboru `soubor_shluků` v porovnání se vzorovými shluky ze souboru `soubor_vzorových_shluků`. Parametr fuzzy [0|1] určuje, zda je vstupní soubor shluků výstupem z LS (tj. obsahuje hodnoty potenciálů). Pokud ano, parametr `mez` určuje, které shluky náleží do shluku a které nikoliv. Výstup se uloží ve formátu vhodném pro GNUPlot.
2. `pii <velikost_kolekce> <velikost_vektorů> <počet_shluků> <soubor_vektorů> <soubor_shluků> <krok>`
   - Příklad: `pii 600 7050 87 vectors.txt clustersCom.txt 0.001`
   - Funkce: Vypočte hodnoty měr P-intra a P-inter na intervalu $<0,1>$s krokem `krok`. Výstup uloží ve formátu vhodném pro GNUPlot.

Pomocné programy

V této kapitole jsou popsány programy, které mají pouze pomocný a informační charakter.

1. `parseTextBack <vstupní_adresář> <výstupní_adresář>`
    - Příklad: `parseTextBack DataIn FullTexts`
    - Funkce: do zadaného výstupního adresáře překonvertuje vstupní označkované soubory do čitelné formy.
    - Poznámka: ve vstupním adresáři `DataIn` musí být soubor `seznam.txt` se jmény všech souborů v adresáři, které chceme překonvertovat.
2. `getText <ID_dokumentu>`
    - Příklad: `getText 10398`
    - Funkce: na STDOUT vypíše soubor s požadovaným ID.
3. `viewer <maxDocs> <dataXX.txt>` [`udataXX.txt`]
    - Příklad: `viewer 250 data19.txt` [`udata19.txt`]
    - Funkce: na STDOUT vypíše soubory jednoho shluku a to podle vstupního souboru pro `GNUPlot` na kreslení grafu.

Literatura

[1] M. Holub, J. Semecký, J. Diviš. Searching for Topics in a Large Collection of Texts. Proceedings of the ACL, 2004 Student Research Workshop, pp. 43-48. Association for Computational Linguistic (ACL), Barcelona, 2004.

[2] Christis Faloutsos, Douglas Oard. A Survey of Information Retrieval and Filtering Methods. University of Maryland, College Park, MD 20742., 1995.

[3] J. Pokorný, V. Snášel, D. Húsek. Dokumentografické informační systémy. Karolinum, Skriptum MFF UK Praha, 1998.

[4] J. C. Bezdek. Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York, 1981.

[5] J. MacQueen. Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability, pp. 281-297. University of California Press, Berkeley, 1967.

[6] J. C. Dunn. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. Journal of Cybernetics 3, pp. 32-57, 1973.

[7] L. Kaufman and P. Rousseuw. Finding Groups in Data – An Introduction to Cluster Analysis. Wiley Series in Probability and Mathematical Sciences, 1990.

[8] P. Pecina, M. Holub. Sémanticky signifikantní kolokace. ÚFAL/CKL technical report TR-2002-13, Praha, 2002.

[9] J. Hajič. Morphological Tagging: Data vs. Dictionaries, strana 94-101. Proceedings of the 6th ANLP Conference, 1st NAACL Meeting. Seattle, Washington 2000.

[10] J. Hajič, B. Vidová-Hladká. Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured Tagset, Proceedings of the Conference COLING-ACL 98. Montreal, Canada, 1998.

[11] J. Pávek. Testovací kolekce českých dokumentů a dotazů. Diplomová práce. MFF UK, Praha, 2004.

[12] M. Steinbach, G. Karapis, V. Kumar. A comparison of document clustering techniques. Proc. Workshop on Text Mining, 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2000.

[13] B. Larsen, C. Aone. Fast and effective text mining using linear-time document clustering. Proc. 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 1999.

[14] S. Dhillon and D.S. Modha. Concept decompositions for large sparse text data using clustering. Machine learning, 42(1/2), pp. 143-175, 2001

[15] M. Červenka. Clusters of terms and documents in document retrieval systems, Diplomová práce. MFF UK, Praha, 2004.

[16] S.Albayrak, F. Amasyali. Fuzzy c-means clustering on medical diagnostic systems. XII. international Turkish symposium on artificial inteligence and neural network. TAINN, 2003.