VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

ALGORITMY PRO SHLUKOVÁNÍ TEXTOVÝCH DAT

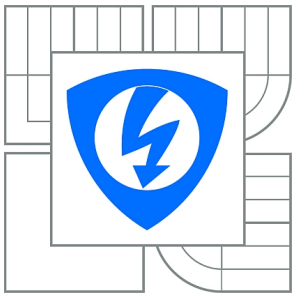
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JOSEF SEDLÁČEK
AUTHOR

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

ALGORITMY PRO SHLUKOVÁNÍ TEXTOVÝCH DAT

TEXT DATA CLUSTERING ALGORITHMS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

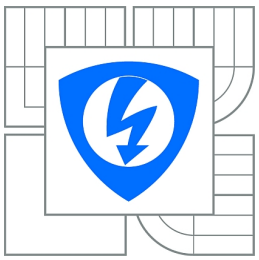
AUTOR PRÁCE — Bc. JOSEF SEDLÁČEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN KARÁSEK

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Josef Sedláček	***ID:***	84159
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Algoritmy pro shlukování textových dat

POKYNY PRO VYPRACOVÁNÍ:

Úkolem studenta bude seznámit se s problematikou dolování dat, se zaměřením na textová data. Student prostuduje teorii týkající se metod používaných pro shlukování textových dat a sepíše rešerši. Na základě prostudované teorie navrhne aplikaci, která bude provádět shlukování textových dokumentů. Součástí diplomové práce bude vytvoření testovací množiny dat a implementace alespoň dvou metod shlukování dokumentu v jazyce JAVA a jejich srovnání.

DOPORUČENÁ LITERATURA:

[1] Feldman, R., Sanger, J.: The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data, Cambridge University Press, 2007, ISBN 0521836573.
[2] Beil, F., Ester, M. a Xu, X.: Frequent Term-Based Text Clustering. Edmonton, Canada, Proceedings of SIGKDD02, 2002.
[3] Vester, L. K. a Martiny, C. M.: Information Retrieval in Document Spaces Using Clustering. Denmark, Technical University of Denmark, Informatics and Mathematical Modelling, 2005.

Termín zadání:	7.2.2011	***Termín odevzdání:***	26.5.2011

Vedoucí práce: Ing. Jan Karásek

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato diplomová práce se zabývá problematikou dolování textových dat. Je zde popsána teorie potřebná ke shlukování textových dokumentů a také algoritmy, které se při shlukování využívají. Podle této teorie je pak vytvořena aplikace pro shlukování textových dat. Aplikace je vytvořena v programovacím jazyku Java a obsahuje tři metody používané při shlukování. Uživatel si tak sám může vybrat metodu, podle které chce kolekci dokumentů shlukovat. Implementované metody jsou K medoids, BiSec K medoids a SOM (self organization map). Součástí aplikace je také vytvoření validační množiny, pomocí které jsou algoritmy testovány. V závěru jsou pak algoritmy porovnány podle dosažených výsledků.

KLÍČOVÁ SLOVA

data mining, shlukováni, kolekce dokumentu, term, shlukovací algoritmy, SOM(self organization map), K means, K medoids, BiSec K means, BiSec K medoids

ABSTRACT

The thesis deals with text mining. It describes the theory of text document clustering as well as algorithms used for clustering. This theory serves as a basis for developing an application for clustering text data. The application is developed in Java programming language and contains three methods used for clustering. The user can choose which method will be used for clustering the collection of documents. The implemented methods are K medoids, BiSec K medoids, and SOM (self-organization maps). The application also includes a validation set, which was specially created for the diploma thesis and it is used for testing the algorithms. Finally, the algorithms are compared according to obtained results.

KEYWORDS

data mining, clustering, document collection, term, clustering algorithms, SOM (self-organization map), K means, K medoids, BiSec K means, BiSec K medoids

SEDLÁČEK, Josef *Algoritmy pro shlukování textových dat*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2010/2011. 77 s. Vedoucí práce byl prof. Ing. Jan Karásek,

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Algoritmy pro shlukování textových dat“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

Poděkování

Dekuji Ing. Janu Karáskovi za poskytnutí teoretických informací i praktických rad, pro napsání této diplomové práce a vytvoření aplikace na shlukování textových dokumentů.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V diplomové práci je rozebrána problematika dolování textových dat se zaměřením na shlukování dokumentů. V rámci práce je podrobně popsána teorie zabývající se zpracováním textu. Na základě prostudované teorie byly vybrány nejpoužívanější algoritmy pro shlukování dokumentů. Jsou to algoritmy K medoids, BiSec K medoids a Kohonenova samoorganizační mapa (SOM). Na prostudovanou teorii navazuje implementace vybraných algoritmů, které byly následně testovány a porovnávány. Pro testování byla vytvořena speciální validační množina souborů, kterou je možné poskytnout k volnému použití komunitě zabývající se touto problematikou. Validační množina je rozdělena do deseti kategorií z nichž každá čítá deset dokumentů. Porovnání algoritmů se zaměřuje na časovou a paměťovou náročnost a také na přesnost dosažených výsledků. Na konci diplomové práce jsou dosažené výsledky shrnuty a vyhodnoceny. Diplomová práce se skládá ze tří hlavních částí, které jsou popsány v následujících odstavcích.

V první části je popsaná teorie týkající se dolování informací z textových dat. V textu jsou popsány základní pojmy používající se při dolování dat. Také je v této části popsán rozdíl mezi strukturovanými a nestrukturovanými daty a modely, které se používají při zpracovávání dat. Mezi nejčastěji používané modely patří Booleovský, Vektorový a Fuzzy Booleovský model. Jsou zde také popsány jednotlivé postupy při zpracovávání dat, a to převod dat na jednotlivé termy, převod termů na kanonický tvar či vymazání stop slov. Také zde jsou popsány jednotlivé metody pro zpracovávání textových dat. Jednou z těchto metod je shlukování, kterému je věnována celá následující kapitola.

Druhá část diplomové práce se zabývá metodami shlukování textových dat. Tato část zmiňuje typy dat a metod používajících se při shlukovávání. Mezi takové metody patří například hiearchické metody, dělící metody, metody založené na hustotě, metody založené na mřížce či neuronové sítě. U těchto metod jsou popsány také algoritmy, které se při shlukovávání využívají. V mé diplomové praci se zabývám především algoritmy K medoids, BiSec K medoids a SOM. Tyto tři algoritmy jsou zde také podrobně popsány.

Třetí část se pak věnuje samotné aplikaci, která byla implementována v rámci diplomové práce. Popisuje jednotlivé balíčky a třídy, které aplikace obsahuje. Dále také popisuje jednotlivé atributy a metody. Součástí této kapitoly je podrobný návod s obrázky jak aplikaci používat. V poslední části jsou pak srovnány klady a zápory jednotlivých algoritmů a vyhodnoceny výsledky.

1 DOLOVÁNÍ INFORMACÍ Z TEXTOVÝCH DAT

V dnešní době internetu je velké množství dat, které nejsme schopni v reálném čase zpracovat, a každý den vnikají nová a nová data. Abychom tato data mohli efektivně zpracovávat, vznikl vědní obor zvaný Data mining neboli dolování dat. Tento obor se zabývá získáváním informací z velkého množství dat.

Jedním z podoborů dolování dat je dolování textových dat. Při dolování dat se většinou využívají strukturovaná data, například z databázových tabulek. Při dolování z textových dat se většinou nepoužívají data strukturovaná, ale data nestrukturovaná či semistrukturovaná. Tato data představují většinou textové dokumenty, pdf dokumenty, různé články, emailová komunikace či html stránky. Tato data mají svoji pevnou strukturu či šablonu, ale nejsou plně strukturovaná.

Problémem dolování textových dat je velké množství souborů, ve kterých lze data hledat. Ne všechny soubory jsou však pro nás obsahově zajímavé. Starší vyhledávací metody většinou vracely malé množství relevantních dokumentů a poměrně velké množství irelevantních dokumentů. Často jsme při hledání informací zahlceni velkým počtem dokumentů, ale z časových důvodů můžeme prohlédnou jen několik. Z tohoto důvodu vzrůstá poptávka po nástrojích pro vyhledávání a zpracovávání textu.

1.1 Základní pojmy

Na začátku této práce je potřeba ujasnit si základní používané pojmy používané při hledání v textových datech.[2]

1.1.1 Znaky

Znaky jsou základní komponenty, ze kterých jsou složena slova neboli termy. Znak může mít podobu mezery, písmene, číslice, nebo se jedná o některý ze speciálních znaků. Samostatný znak nemá moc velký význam při hledání informací z textových dokumentů. Až při seskupení více znaků do slov a termů se jejich význam zvětšuje.

1.1.2 Slova a termy

Každý dokument je tvořen ze slov, která mají svůj vlastní specifický význam. Termy jsou pak tvořeny těmito jednotlivými slovy či víceslovnými výrazy. U těchto více slovných termů nesmí být žádné slovo vyjmuto, jelikož by tento term pak ztratil smysl. Každý term musí také obsahovat určení slovního druhu.

Jako příklad je uvedena věta „Navštívili jsme vánoční trhy na náměstí Svobody.“ U této věty může obsahovat seznam termů jak samostatná slova jako „trhy“ či „náměstí“, ale také víceslovné termy jako „náměstí Svobody“ či „vánoční trhy“.

Některé termy mohou mít podobný či skoro stejný význam, což může způsobit zkreslené výsledky při vyhledávání informací. Aby se toto zkreslení co nejvíce zmenšilo, používají se různé techniky na snížení počtu termů a ponechaní jen termů s vyšším významem.

1.1.3 Dokument

Dokument je základní jednotka pro vyhledávání textových dat. Může být v mnoha podobách například jako novinový článek, e-mail, kniha či zpráva z burzy. Každý dokument může být obsažen ve více kolekcích dokumentů a každá kolekce dokumentů může obsahovat více druhů dokumentů.

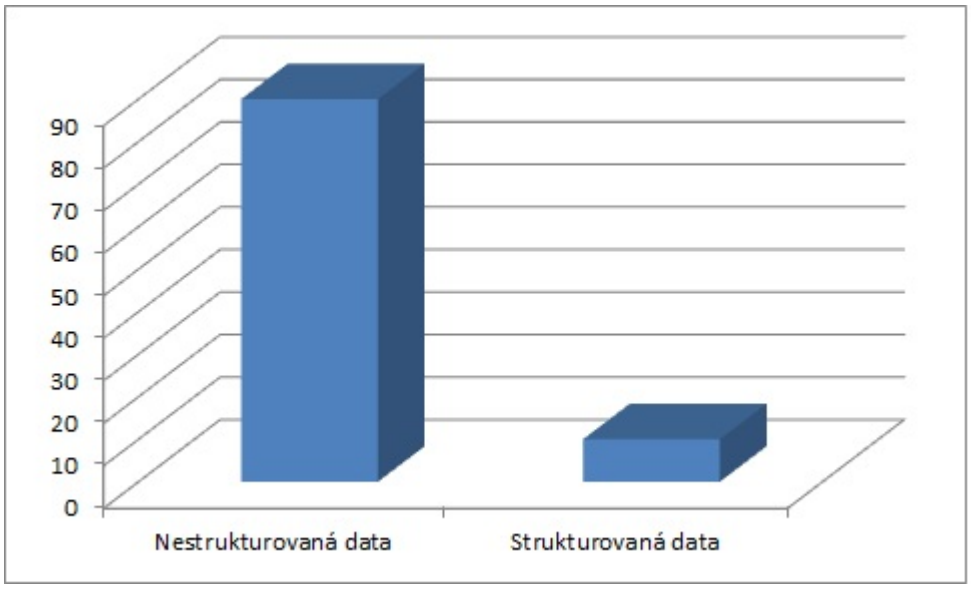
1.1.4 Kolekce dokumentů

Kolekce dokumentů obsahuje různé typy a počty dokumentů. Kolekce může obsahovat dokumenty v řádu set až tisíců nebo milionů. Kolekce mohou být také statické nebo dynamické. U statických kolekcí se počet obsažených dokumentů nemění, oproti tomu v dynamických kolekcích se počet dokumentů neustále mění. Kolekce dokumentů je základní prvek, na kterém se zkoušejí různé dolovací metody.

1.2 Dokumentografické informační systémy

Dokumentografické informační systémy (zkráceně DIS) slouží k práci s textovými daty, jako je například ukládání a výběr dokumentů odpovídajících uživatelskému dotazu. První DIS vznikly už před více než padesáti lety. Vyvinuly se v knihovnictví, kde se tvořily různé listy popisující knihy, které se následně třídily a postupnou automatizací těchto kroků se první DIS vytvořily. Tyto první systémy byly založeny na sekvenčním prohledávání dat. V dnešní době se však už více používají různé metody indexace, jako jsou například hašovací funkce či signaturové metody. Informační systémy lze rozdělit do dvou hlavních částí (faktografické a dokumentografické).[3]

Faktografické systémy používají strukturovaná data uložená většinou v relačních databázích. Jako příklad lze uvést personalistické databáze u firem, kdy každá taková databáze má svoji pevnou strukturu. Mohou v ní být uvedeny informace o zaměstnanci, jako například jméno, příjmení, datum narození, bydliště a mnoho dalších informací.[1]



Obr. 1.1: Poměrné uložení strukturovaných a nestrukturovaných dat.[14]

Dokumentografické informační systémy uchovávají nestrukturovaná data, jako jsou u firem například životopisy nebo příspěvky v různých elektronických konferencích. Následující obrázek ukazuje poměr uložených dat v textové podobě a dat strukturovaných.[2]

DIS se skládají ze dvou hlavních částí. Těmito částmi jsou subsystém zpřístupnění dokumentu a subsystém dodání dokumentů. Subsystém zpřístupnění dokumentů obsahuje například název dokumentu, autora, rok vydání, vydavatele, stručný obsah a podobné údaje. Tyto informace mohou být nalezeny například při internetovém vyhledávání pomocí internetového vyhledavače Google[1] či Seznam[2]. Naopak subsystémy dodání dokumentů neobsahují žádné tyto informace, ale slouží ke zprostředkování a předložení celého dokumentu uživateli. V DIS jsou použity většinou dva hlavní modely ukládání dat: Booleovský a Vektorový model.[2]

1.2.1 Booleovský model

Booleovský model je nejstarší model používaný v DIS, přesto je stále poměrně velice často používaný. Teoretické základy tohoto modelu byly popsány už v 50. letech minulého století a lze jej vidět například u internetového vyhledávače AltaVista[3]. Každý dokument v tomto modelu je reprezentován množinou termů, které jej co nejlépe popisují. Dotaz na takový dokument se pak skládá právě z těchto termů a logických spojek. Takový dotaz může mít například tvar: (term1 or term2) and term3.

[1] http://www.google.cz/
[2] http://www.seznam.cz/
[3] http://www.altavista.com/

U těchto dotazů jsou nejvíce používané spojky: [2]

- A AND B: tato spojka znamená logický součin neboli konjunkci. Výsledkem takového dotazu budou všechny dokumenty, které obsahují jak term A, tak také term B.
- A OR B: tato spojka znamená logický součet neboli disjunkci. Výsledkem takového dotazu budou všechny dokumenty, které obsahují term A nebo B.
- A XOR B: tato spojka značí exkluzivní součin. Výsledek tohoto dotazu bude podobný jako u OR, jen s tím rozdílem, že zde nebudou dokumenty, které obsahují oba termy současně.
- NOT A: tato spojka značí negaci. Výsledkem bude dokument, který neobsahuje term A.

Tyto spojky jsou však jen základní, a kdyby systém používal jen tyto čtyři, tak by příliš kvalitní nebyl. Proto existují další rozšiřující značky a dotazy. Jedním z nich je možnost dotazování se na sekundární informace. Další možností je třeba používání zástupních znaků. U některých systémů je však povoleno tyto zástupky vkládat až na konec dotazů. V Booleovském modelu lze také omezit vzdálenost hledaných se termů pomocí speciálních dotazů, které jsou uvedeny zde:[2]

- A (m,n) B: term A je minimálně vzdálen od termu B o m a maximálně o vzdálenost n.
- A chapter B: term A se musí vyskytovat ve stejné kapitole jako term B.
- A paragraph B: term A musí být ve stejném odstavci jako term B.
- A sentence B: term A se musí vyskytovat ve stejné větě jako term B.

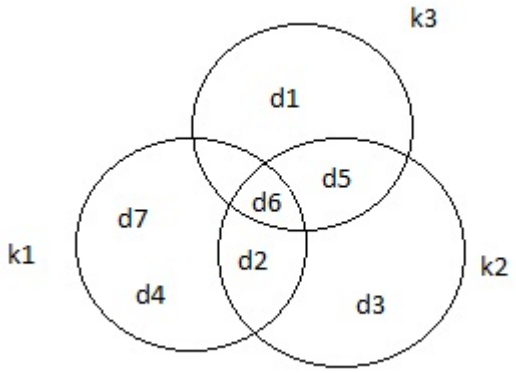
Při indexaci v Booleovském modelu nejsou k souboru přiřazeny termy, které soubor obsahuje, ale naopak k jednotlivým termům je přiřazen soubor, který obsahuje názvy souborů, ve kterých se daný term nachází. Tento soubor se jmenuje invertovaný indexový soubor.[2]

V dotazech lze také vyhledávat nadřazené či podřazené termy k termu hledanému. Toto lze provést pomocí speciálních operátorů Oracle SQL*Text. Tyto znaky jsou uvedeny v Tab.1.1

Velkým nedostatkem Booleovského modelu je neschopnost seřadit výsledky v relevantním pořadí. Další nevýhodou je fakt, že při některých dotazech dostaneme naprosto neočekávané výsledy. Booleovský model má své výhody i nevýhody a je na uživateli, zda si tento model zvolí či nikoli.[2]

Operátor	Význam	Popis
NT (A)	NARROWED TERM	užší term k termu A
BT (A)	BROADER TERM	širší term k termu A
TT (A)	TOP TERM	nejširší term k termu A
RT (A)	RELATED TERM	příbuzný term k termu A
PT (A)	PREFERRED TERM	preferovaný term k termu A
SYN (A)	SYNONYMUM	synonyma k termu A

Tab. 1.1: Operátory Oracle SQL*Text



Obr. 1.2: Vektorový model

1.2.2 Vektorový model

Jak již název napovídá Vektorový model je založen na vektorech. Každý dokument v tomto modelu je indexovaný pomocí n termů. Vektorový model bere v úvahu počet výskytů termů v dokumentech. Tento výskyt je znázorněn na Obr.1.2, kde *k1,k2 a k3* znázorňují shluky dokumentů a *d1 - d7* jednotlivé dokumenty.[2]

Z tab.1.2 je pak dobře patrné, že nezáleží jen na tom, zda se hledaný term v dokumentu nachází, ale také kolikrát je v dokumentu obsažen a jakou má váhu. Obecná matice dokumentů by pak měla vypadat takto:[2]

$$D = \begin{pmatrix} w_{11} & w_{12} & w_{13}... & w_{1n} \\ w_{21} & w_{22} & w_{23}... & w_{1n} \\ .... & .... & .... & .... \\ w_{31} & w_{32} & w_{33}... & w_{1n} \end{pmatrix}$$

$\mathrm{D}_m = (w_{m1}, w_{m2}, \ldots, w_{mn})$
$w_{mn} \in < 0, 1 >$ - váhy termů v dokumentu D_m

Shluky	Shluk k1	Shluk k2	Shluk k3	Ohodnocení (q*j)
dokument 1	0	0	1	3
dokument 2	1	1	0	3
dokument 3	0	1	0	2
dokument 4	1	0	0	1
dokument 5	0	1	1	5
dokument 6	1	1	1	6
dokument 7	1	0	0	1
váha	1	2	3	

Tab. 1.2: Ohodnocování dokumentů

$w_n \in < 0, 1 >$ - vyhledávané termy

Podobnost dokumentu s dotazem pak lze vyjádřit vzorcem 1.1

$$Sim\left(Q, D_i\right) = \sum \left(q_k * w_{ik}\right). \tag{1.1}$$

Tato rovnice však nerespektuje závislost vah termů na délce dokumentu, a proto lze podobnost zapsat přesněji pomocí kosínové míry. Kosínovou míru popisuje rovnice 1.2.

$$Sim\left(Q, D_i\right) = \frac{\sum \left(q_k * w_{jk}\right)}{\sqrt{\sum \left(w_{ik}\right)^2}} * \sum \left(q_k\right)^2 = \cos\left(\varrho\right). \tag{1.2}$$

Podobnost dokumentu s dotazem pro binární hodnoty lze zapsat pomocí Jaccardova koeficientu (vzorec 1.3) nebo Czekanowskeho-Diceova koeficientu (vzorec 1.4).

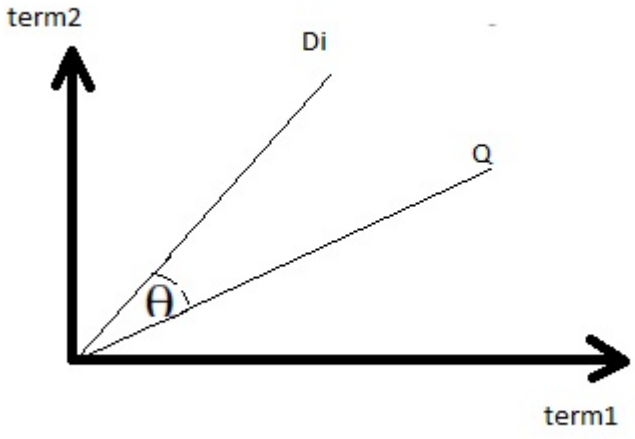
$$Sim\left(Q, D_i\right) = \frac{\sum_{k=1\ldots n} \left(q_k * w_{jk}\right)}{\sum_{k=1,\ldots,n} \left(q_k * w_{ik}\right) + \left(\sum_{k=1,\ldots,n} \left|q_k - w_{ik}\right|\right)}. \tag{1.3}$$

$$Sim\left(Q, D_i\right) = 2 * \frac{\sum_{k=1\ldots n} \left(q_k * w_{jk}\right)}{2 * \sum_{k=1,\ldots,n} \left(q_k * w_{ik}\right) + \left(\sum_{k=1,\ldots,n} \left|q_k - w_{ik}\right|\right)}. \tag{1.4}$$

Díky funkci Sim je odstraněn jeden z největších problémů v Booleovském modelu, a to řazení dokumentů ve výsledku. To se zde řadí sestupně podle výsledků této funkce.[2] Váhování zpracovávaných dokumentů se provádí dvěma způsoby. První ze způsobů využívá rovnici 1.5

$$w_{ij} = TF_{ij} * IDF_j, \tag{1.5}$$

kde TF_{ij} je frekvence fermu j v dokumentu i. $IDF_j = \log\left(m/DF_j\right) + 1$, kde IDF_j je inverzní frekvence výrazu a DF_j je počet dokumentů s termem j.



Obr. 1.3: Geometrická interpolace měření podobnosti

Druhý způsob pak využívá rovnici 1.6

$$w_{ij} = NTF_{ij} * IDF_j, \tag{1.6}$$

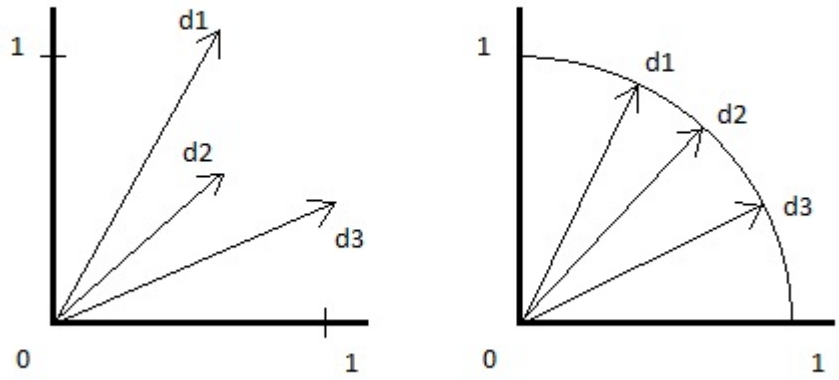
kde NTF_{ij} je normalizovaná frekvence. $NTF_{ij} = \frac{\left(\frac{TF_{ij}}{max_{tk}}+1\right)}{2}$ a $maxTF_{ij}$ je maximální frekvence termu v řádku *i*. IDF_j se pak počíta stejně jako u vzorce 1.5
Když se term vyskytuje ve všech dokumentech pak se $\log(1) = 0$, jelikož term patří mezi nevýznamová slova. Naopak když se term v dokumentu vyskytuje pouze jednou pak se $\Rightarrow IDF$=$\log(m+1)$. Geometrická interpretace měření podobnosti je znázorněna na obrázku 1.3

Při vyhledávání ve více dokumentech je dobré vektory těchto dokumentů normalizovat, aby nebyly bezdůvodně zvýhodněny delší vektory. Na obrázku 1.4 je ukázka normalizace vektoru na jednotkovou velikost.[2]

1.2.3 Fuzzy booleovský model

Tento model odstraňuje nevýhody Booleovského modelu pomocí fuzzy logiky. Každému dokumentu je přiřazen m-rozměrný vektor, kde každá složka určuje důležitost termu v daném dokumentu. Dotaz se tvoří podobně jako u normálního Booleovského modelu, jen s tím rozdílem, že lze k jednotlivým termům přidělit i jejich váhu.[2]

Při vyhodnocování dotazu se rozlišuje konjunkce (operátor AND) a disjunkce (operátor OR). Při konjunkci se bere v potaz vzdálenost od jednotkového vektoru



Obr. 1.4: Normalizace vektoru na jednotkovou velikost

a u disjunkce vzdálenost od nuly. Tento model tak spojuje výhody Booleovského a Vektorového modelu.[2]

1.3 Předzpracovávání dat

Dříve než začneme z určitého množství dat dolovat užitečné informace různými metodami, musíme si zpracovávaná data předpřipravit.[4] Předzpracovávání lze rozdělit na základní čtyři části, a to: převod dat na jednotlivé termy, převod termu na kanonický tvar, vynechání stop slov a méně významových slov a ohodnocování termů. Po předzpracovávání dat je jejich objem mnohonásobně menší, což vede k rychlejšímu zpracovávání dat a k přesnějším výsledkům.

1.3.1 Převod dat na jednotlivé termy

V této části jsou všechny zpracovávané dokumenty načteny a jejich věty jsou převedeny na jednotlivé termy. O termech jsem se již zmiňoval v podkapitole Slova a Termy. Při tomto zpracovávání jsou odstraněny interpunkční znaky jako například tečka, čárka, vykřičník, otazník a mnoho dalších. Jsou zde všechna velká písmena převedena na písmena malá, aby se v paměti neuchovávaly termy se stejným výrazem.[6]

1.3.2 Převod termu na kanonický tvar slov

Tento bod předzpracovávání informací je poměrně obsáhlý a důležitý. Jednotlivá slova se zde nahrazují jen jejich základem a tím se výrazně zmenšuje počet termů v kolekci dat. Tento krok se nazývá stemming. Jako příklad lze uvést slova jako plavání, plavčík a plavecký. Tyto slova po stemmingu nahradí jen jedno slovo plav.

Pro stemming anglického jazyka jsou používány různé speciální algoritmy. Jeden z nejznámějších je Porterův stemmovací algoritmus. Tento algoritmus vznikl již v roce 1979 a byl napsán v typovacím jazyce BPCL. Dnes se již v této podobě nepoužívá, ale je přepsaný do všech dnes známých programovacích jazyků. Porterův algoritmus je založen pouze na zpracovávání přípon. Předpon u anglického jazyka není tolik, a tak výrazně neovlivní vyhledávaný výsledek. Přípon je v angličtině něco přes 1000 a většina z nich je tvořena kombinací menších a jednodušších přípon. Z tohoto předpokladu pak celý Porterův algoritmus vychází.[5]

Většina stemmovacích algoritmů se zaměřuje jen na angličtinu nebo na skupinu indoevropských jazyků, což vede ke stále větší potřebě vytvořit nástroje pro lemmatizaci češtiny. Čeština je velice složitý jazyk, a proto není snadné takovýto nástroj vytvořit.

1.3.3 Stop-slova

V této části jsou odstraněna nevýznamová slova, jako jsou například spojky předložky a některá další nevýznamová slova.[4] Opět jsou různá stop-slova pro různé jazyky. Většinu slovníků s těmito stop-slovy však lze najít na internetu. V příloze je většina českých stop-slov.

1.3.4 Ohodnocování slov

Tato část předzpracovávání dat se aplikuje jen u některých dolovacích úloh. Jedná se o přiřazení číselné hodnoty k jednotlivým termům, a tím nastavení důležitosti. K takovémuto ohodnocování slov se používají tzv. váhové metody. Tyto váhové metody využívají například vektorové modely, kde každý dokument D_j z kolekce n dokumentů je reprezentován vektorem.[9]

Mezi nejznámější a nejpoužívanější váhovou metodu patří metoda TF-IDF. Tato metoda zohledňuje výskyt slov v dokumentu a současně snižuje jeho důležitost podle množství výskytů ohodnocovaných slov v celé kolekci dokumentů. [10]

Frekvence výrazu (TF) určuje počet výskytů ohodnocovaných slov v daném dokumentu a vypočítá se pomocí vzorce 1.7

$$TF_{ij}=\frac{n_{ij}}{\sum_k n_{kj}}, \tag{1.7}$$

kde n_{ij} určuje počet výskytů ohodnocovaného slova v dokumentu a $\sum_k n_{kj}$ počet všech slov v daném dokumentu. Inverzní frekvence dokumentů (IDF) určuje počet výskytů ohodnocovaných slov v celé kolekci dokumentů a lze ji vypočítat podle vzorce 1.8.

$$IDF_i = \log \frac{|D|}{|d_j : t \in d_j|}, \tag{1.8}$$

kde $|D|$ je celkový počet dokumentů v kolekci a $|d_j : t \in d_j|$ je počet dokumentů, v kterých se ohodnocované slovo vyskytuje. Váhu jednotlivých slov pak lze vypočítat podle vzorce 1.9

$$TFIDF_{ij} = TF_{ij} * IDF_i. \tag{1.9}$$

Ohodnocování slov váhou se využívá například v modelech pro vyhledávání informací. Také se využívá u klasifikaci či shlukování.

1.4 Metody pro dolování textových informací

Dolování v textech se v některých věcech podobá dolování v datech, avšak současně se i v mnoha věcech liší. Mezi klasické metody v dolování v datech patří například předvídání hodnot nebo rozhodovací strom. Tyto metody však nemohou být použity pro dolování v textech, jelikož jsou založeny na strukturovaných datech. Metod pro dolování v textech je mnoho, proto zde zmíním jen ty základní.

1.4.1 Klasifikace

Při klasifikaci jsou dokumenty řazeny do předem daných kategorií. Lze toho využít například při filtrování nevyžádané pošty, kdy do kategorie spam jsou řazeny nevyžádané dokumenty. Počet a druh kategorií pro klasifikaci je dán vždy předem. Každá kategorie je popsána pomocí požadavků nebo profilu.

Požadavek tvoří uživatel manuálně a může vypadat třeba takto: Do kategorie sportovní aktivity zařaď každý dokument, který bude obsahovat aspoň 8x slovo „sport“. Pro lepší přesnost je pak finální požadavek rozdělen do několika konjunkcí a disjunkcí jednodušších požadavků. Takovéto manuální tvoření požadavků je však velice náročné a náchylné na chyby.[22]

Profil dokumentu vznikne tak, že se z dokumentu extrahuje množina slov. Profil kategorie se pak tvoří právě ze všech profilů dokumentů v dané kategorii.

Podle způsobu výběru kategorií lze klasifikaci dále dělit na mono-klasifikace a multi-klasifikace. Při mono klasifikaci patří každý dokument jen do jedné třídy, zatímco u multi-klasifikace může každý dokument patřit do více tříd.

Klasifikační metody lze také rozdělit podle přístupu k hledání a trénování odpovídajících kategorií. Dělení je následující:

- Metody s klasifikačními pravidly hledají pravidla, která popisují danou kategorii. Tato pravidla pak mohou být použita při tvorbě rozhodovacích stromů.
- Lineární klasifikátory pak popisují lineární vektory k popisu dokumentů i kategorií. Vektory kategorií se při trénování nastaví tak, aby odpovídaly vektorům trénovacích dokumentů. Při řazení dalšího dokumentu mezi už známé kategorie se pomocí algoritmu zjistí, jaká kategorie je danému dokumentu nejblíže, a do té se dokument přiřadí.
- Metody založené na příkladech zase najdou nejprve dokumenty podobné k řazenému dokumentu a pak se podle jejich zařazení řadí i nový dokument.
- Poslední skupinou jsou algoritmy pro strojové učení jako například genetické algoritmy nebo Fuzzy logika.

1.4.2 Extrakce

Cílem extrakce informací z dokumentu je převést nestrukturovaná data na data strukturovaná. Tato metoda se používá například při zpracovávání HTML a XML souborů. Tyto soubory obsahují značky, pomocí kterých se dají data extrahovat. Například lze naučit automat, jaké údaje má extrahovat mezi zadanými značkami. Po natrénování pak automat data zpracovává mezi značkami z dané stránky.[22]

1.4.3 Selekce

Selekce informací je jednou z metod, kterou používají internetové vyhledávací servery, jako je například Google nebo Seznam. Do vyhledávače zadá uživatel klíčová slova a vyhledávač během pár sekund najde seznam relevantních dokumentů. Někdy však uživatel není spokojen s výsledkem hledání. Při posuzování kvality vyhledávacího systému hraje hlavní roli relevance klíčových slov a seznamu vyhledaných dokumentů. Na základě posuzování relevance výsledku se pak vyhledávání hodnotí dvěma hodnotami. Jednou je přesnost vyhledávání (precision) a druhou úplnost vyhledávání (recall).[7]

1.4.4 Predikce

Při predikci se odhaduje neznámá hodnota ze spojité funkce. Používá se především při dolování dat z transakčních databází. Nejznámější metodou predikce je tzv. lineárně jednoduchá a lineárně nenásobná regrese.[8]

U regrese se data prokládají přímkou tak, aby jim co nejvíce odpovídala. U jednoduché regrese má přímka tvar:

$$Y = aX + b, \tag{1.10}$$

kde Y jsou výstupní parametry a X parametry vstupní. Parametry a a b jsou hledané parametry. U lineární vícenásobné regrese mohou být použity dva i více parametrů. Rovnice se tedy modifikuje na tvar:

$$Y = a_0 + a_1X_1 + a_2X_2 + \ldots + a_vX_v. \tag{1.11}$$

1.4.5 Sumarizace

Sumarizace pomáhá uživateli porozumět dokumentu během velice krátké doby. Při sumarizaci vzniká krátký souhrn všech důležitých částí dokumentu. Sumarizaci lze provádět dvěma způsoby.

První ze způsobů je obsahová sumarizace. Při této metodě vzniká souhrn implementací původního textu. V textu jsou dlouhé věty redukovány na krátké, ale jejich význam je zachován. Například věta „Babička doma chová králíky, kačeny, slepice, a má dokonce i psa.“, je nahrazena větou „Babička doma chová zvířata.“

Druhý způsob je sumarizace extrakcí. U tohoto způsobu vzniká souhrn vyjmutím důležitých částí dokumentu a jejich následným vložením do souhrnu. Toto vyjmutí lze provést pomocí statických principů nebo heuristických metod.[22]

1.4.6 Shlukování

Při shlukování se velké množiny dokumentů dělí na menší podmnožiny tak, že jsou v každé podmnožině podobné dokumenty a naopak mezi jednotlivými podmnožinami jsou co největší rozdíly. Při shlukování neznáme ani vlastnosti, ani počet podmnožin, které vzniknou. Při učení nepotřebujeme trénovací množinu, ale pouze množinu validační, a proto se jedná o učení bez učitele. Algoritmům používaným při shlukování se bude zabývat celá další kapitola.[22]

1.4.7 Vizualizace

Tato metoda je doplňková ke všem předchozím metodám. Každá ze zde zmíněných metod by měla být graficky znázorněna pro lepší orientaci uživatele. Existuje mnoho způsobů vizualizací pro každou ze zmíněných metod dolování.

2 SHLUKOVÁNÍ TEXTOVÝCH DAT

Jak už bylo napsáno v kapitole 1.4.6, shlukování je jednou z metod dolování textových dat. Má za cíl vytvoření neznámého počtu shluků. Shluky se musí co nejvíce lišit, ale dokumenty ve shlucích musí mít co největší podobnost. Jedná se o učení bez učitele, jelikož při trénování není potřeba žádná trénovací množina. Shlukování má určitou podobnost s klasifikací, u které se také snažíme řadit neznámá data do shluků. U klasifikace jsou však shluky předem popsány a data se pak řadí přímo do těchto shluků, zatímco u shlukování jsou shluky tvořeny přímo při učení.

Podobnost u jednotlivých dokumentů je určena podle tzv. měr podobnosti, mezi které patří například vzdálenostní a podobnostní funkce. Vzdálenostní funkce se používá především u numerických typů proměnných. Čím je hodnota vzdálenosti větší, tím si jsou objekty méně podobné. U podobnostní funkce je to přesně naopak. Čím má podobnostní funkce větší hodnotu, tím si jsou objekty podobnější. Aby nedocházelo k záměnám těchto dvou funkcí, používá se u podobnostní funkce tzv. koeficient odlišnosti. Ten je velice snadno vypočitatelný a je to vlastně odečet podobnosti od čísla jedna. Vzorec 2.1 znázorňuje tento koeficient:

$$d\left(x_i, x_j\right) = 1 - S\left(x_i, x_j\right). \tag{2.1}$$

Odlišnost pak nabývá hodnot v intervalu (0,1).

Když bychom chtěli vytvořit nejprve všechny možné kombinace shluků a pak z nich vybrat ten nejlepší, trvalo by nám to třeba i několik let. Například všech možných kombinací 100 datových objektů řazených do 5 shluků je 10^{67}. Proto se shlukování zabývá různými heuristickými metodami a ne hrubou silou. Vzorec 2.2 ukazuje výpočet všech kombinací shluků:

$$\frac{1}{c!}\sum_{i=1}^{c}(-1)^{c-i}\binom{c}{i}i^N, \tag{2.2}$$

kde N je počet datových objektů a c je počet shluků.

2.1 Typy dat používající se při shlukování

Při shlukování můžeme pracovat s více druhy proměnných. Například můžeme shlukovat textové dokumenty, ale také dokumenty binární a jiné. Při výpočtu vzdálenosti (podobnosti) mezi objekty tak záleží právě na typu proměnných, které shlukujeme, jelikož se vzdálenost mezi jednotlivými typy dat počítá odlišně. Každá vzdálenostní

funkce však musí splňovat tyto dvě podmínky:[23]
symetrie:

$$D\left(x_i, x_j\right) = D\left(x_j, x_i\right), \tag{2.3}$$

nezápornost:

$$D\left(x_i, x_j\right) \geq 0. \tag{2.4}$$

Existují i další podmínky, ale ty být splněny nemusí. Je to například:
totožnost:

$$D\left(x_i, x_j\right) = 0 \Leftrightarrow x_i = x_j. \tag{2.5}$$

trojúhelníková nerovnost:

$$D\left(x_i, x_j\right) \leq D\left(x_i, x_k\right) + D\left(x_k, x_j\right). \tag{2.6}$$

Při splnění i těchto dvou podmínek lze vzdálenostní funkci označit jako metriku a vzorce 2.3 a 2.4 upravit do následujících tvarů:
symetrie:

$$S\left(x_i, x_j\right) = S\left(x_j, x_i\right), \tag{2.7}$$

nezápornost:

$$0 \leq S\left(x_i, x_j\right) \leq 1. \tag{2.8}$$

2.1.1 Intervalové proměnné

Jsou proměnné, které nabývají číselných hodnot v lineárním měřítku. Příkladem může být teplota. Problém však může nastat s měrnou jednotkou. Teplotu lze uvádět jak ve stupních Celsia, tak i v Kelvinech. Hodnoty těchto rozdílných stupnic pak nemůžeme porovnávat, jelikož by nám vyšel neobjektivní výsledek. U těchto proměnných tak musíme zavést standardizované bezjednotkové hodnoty.[23] Těmito hodnotami pak může být:

střední odchylka:

$$S_f = \frac{1}{n}\left(|x_{1f} - m_f| - |x_{2f} - m_f| - |x_{nf} - m_f|\right), \tag{2.9}$$

kde $x_{1f}, \ldots x_{nf}$ proměnné f a m_f je střední hodnota f,

		$\mathbf{x}_i$		
		1	**0**	**Suma**
$\mathbf{x}_j$	1	n_{11}	n_{10}	$n_{11} + n_{10}$
	0	n_{01}	n_{00}	$n_{01} + n_{00}$

Tab. 2.1: Kontingenční tabulka[9]

výpočet z-score:

$$z_{if} = \frac{x_{if} - m_f}{s_f}. \tag{2.10}$$

Vzdálenost objektů u tohoto typu proměnných počítáme podle těchto měrných vzdáleností:

euklidovská vzdálenost:

$$D\left(x_i, x_j\right) = \sqrt{\sum_{i=1}^{d} \left|x_{il} - x_{jl}\right|^2}. \tag{2.11}$$

manhattanova vzdálensot:

$$D\left(x_i, x_j\right) = \sum_{i=1}^{d} \left|x_{il} - x_{jl}\right|. \tag{2.12}$$

minkowského vzdálenost:

$$D\left(x_i, x_j\right) = \left(\sum_{i=1}^{d} \left|x_{il} - x_{jl}\right|^p\right)^{\frac{1}{p}}, \tag{2.13}$$

kde d je počet proměnných (dimenzí) objektu x_k. Po dosazení čísla 1 za parametr p dostaneme Manhattanskou vzdálenost a po dosazení čísla 2 vzdálenost Euklidovskou.[9]

2.1.2 Binární proměnné

Tyto proměnné se používají ke značení výskytu některého atributu v objektu. Jako příklad lze uvést dotaz, zda venku sněží nebo ne. Výsledkem je vždy odpověď ANO nebo NE, tedy hodnota 1, nebo 0.[9] To popisuje kontingenční tabulka 2.1, ve které jsou popsány objekty *A a B*.

n_{11} počet 1 u objektu x_i a x_j
n_{00} počet 0 u objektu x_i a x_j
n_{10} počet atributů, kde se u objektu x_i vyskytne 1 a u objektu x_j 0

n_{10} počet atributů, kde se u objektu x_i vyskytne 0 a u objektu x_j 1

Binární proměnné se dělí na symetrické a asymetrické. U symetrických proměnných mají oba stavy, tedy 0 i 1, stejnou hodnotu a tedy i váhu.[23] Jako příklad lze uvést pohlaví žena - muž. U těchto proměnných se dá shoda vyjádřit jednoduchým koeficientem, který popisuje vzorec 2.14:

$$S(x_i, x_j) = \frac{n_{11} + n_{00}}{n_{11} + n_{00} + n_{10} + n_{01}} = \frac{n_{11} + n_{00}}{d}. \tag{2.14}$$

U asymetrických proměnných má naopak jeden stav větší váhu, a tedy i hodnotu, než stav druhý. Jako příklad lze uvést testy na nemoc. U těchto testů nás více zajímá kladný výsledek, jelikož hledáme nemoc, kterou pacient trpí.[9] Shodu u asymetrických proměnných znázorňuje Jaccardův koeficient shody, který popisuje vzorec 2.15:

$$S(x_i, x_j) = \frac{n_{11}}{n_{11} + n_{10} + n_{01}}. \tag{2.15}$$

2.1.3 Nominální, ordinální a poměrové proměnné

Nominální proměnné představují zobecnění binárních proměnných. U těchto proměnných mohou datové objekty nabývat více než dvou stavů. Jako příklad lze uvést výčet barev. Vzdálenost pak vyjadřujeme pomocí koeficientu shody podle vzorce 2.16.

$$d(i, j) = \frac{p - m}{p}, \tag{2.16}$$

kde m je počet shodných barev u obou objektů a p je počet všech barev u obou objektů.

Ordinální proměnné se shodují s nominálními jen s tím rozdílem, že nad nimi lze vytvářet uspořádání dle významnosti. Příkladem je věk člověka, který lze dělit na nízký, střední a vysoký. Tyto hodnoty jdou také nahradit skutečným věkem, tedy například hodnotami 1-100. Obecně pak hodnotami 1-M_l. Hodnoty M_l jsou však pro jednotlivé atributy různé.[9] Z tohoto důvodu je vhodné je transformovat do intervalu pomocí vzorce 2.17:

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}. \tag{2.17}$$

Po této transformaci už lze s proměnnými z_{if} zacházet jako s intervalovými a použít na ně některou ze vzdálenostních funkcí.

Poměrové proměnné nejsou znázorněny v lineárním měřítku, ale mohou být znázorněny například v měřítku exponenciálním. Existuje více možností jak s takovými proměnnými zacházet:

- Můžeme je například zpracovávat stejně jako intervalové proměnné.
- Můžeme je zpracovávat jako ordinální proměnné.
- Lze na ně také aplikovat některou z transformací a pak s nimi pracovat jako s intervalovými proměnnými. Například když jsou proměnné v exponenciálním měřítku, použijeme na ně logaritmickou transformaci: $y_{il} = \log x_{il}$.

2.1.4 Proměnné různého typu

Při dolování dat většinou nepracujeme jen s jedním typem dat, ale s více typy zároveň. Pro výpočet podobnosti různých typů dat lze použít vzorec 2.18:

$$S\left(x_i, x_j\right) = \frac{\sum_{i=1}^{d} \delta_{ijl} S_{ijl}}{\sum_{i=1}^{d} \delta_{ijl}}, \tag{2.18}$$

kde

$$\delta_{ijl} \left(\begin{array}{c} 0 \textit{ pokud hodnota } \mathrm{x}_{il} \text{ nebo } x_{jl} \text{ chybí} \\ 1 \textit{ pokud se hodnoty } \mathrm{x}_{il} \text{ nebo } x_{jl} \text{ vyskytují} \end{array} \right)$$

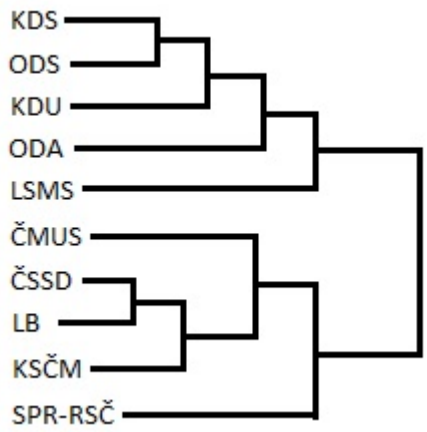
Výpočet podobnosti se pak u tohoto vztahu liší podle typu proměnných.[23] Pro kategorické a binární proměnné platí:

$$\mathrm{S}_{ijl} \left(\begin{array}{l} 1 \textit{ pro } \mathrm{x}_{il} = x_{jl} \\ 0 \textit{ pro } \mathrm{x}_{il} \neq x_{jl} \end{array} \right)$$

Pro numerické proměnné platí rovnice 2.19:

$$S_{ijl} = 1 - \frac{|x_{il} - x_{jl}|}{max x_{nl} - min x_{nl}}, \tag{2.19}$$

kde n jde přes neprázdné datové objekty proměnné l, tzn., že se spočtou minimální a maximální hodnoty dané proměnné l.[9]



Obr. 2.1: Dendrogram hlasování parlamentu v letech 1995-96.[20]

2.2 Metody používané při shlukování

Při shlukování dokumentů lze použít mnoho shlukovacích metod. Tyto metody se dělí na hierarchické, dělící, metody založené na hustotě, mřížkové metody a některé další. Následující podkapitoly tyto metody popisují.

2.2.1 Hierarchické metody

Tyto metody pracují na základě hierarchického rozdělování, kdy je kolekce dokumentů dělena na stromovou strukturu. Dělení může probíhat zespodu nahoru (aglomerativní algoritmy) nebo shora dolů (rozdělovací algoritmy).

U aglomerativních algoritmů je na počátku vytvořeno mnoho shluků. Tyto shluky se pak postupně spojují podle jejich podobnosti. Algoritmus probíhá do té doby, dokud nevznikne jeden velký shluk, nebo než se vytvoří počet shluků zadaných uživatelem. Mezi nejznámější algoritmy patří například Single-Link, Complete-Link nebo Average-Link.

U rozdělovacích algoritmů je naopak na počátku jeden velký shluk a ten se postupně dělí do menších a menších shluků. Algoritmus probíhá do té doby, než není každý dokument ve vlastním shluku, nebo než se vytvoří uživatelem zadaný počet shluků. Tyto metody jsou většinou výrazně rychlejší, ale mají nevýhodu v tom, že po rozdělení do shluků už toto rozdělení nelze vrátit zpět. Jedním z používaných algoritmů je PDDP (Principal Direction Divisive Parttitioning) algoritmus.[21]

Výsledkem všech těchto algoritmů jsou shluky, které se většinou znázorňují pomocí binárních stromů nebo dendrogramů. Na obrázku 2.1 je ukázka takového dendrogramu.

Výhodou všech hierarchických metod je možnost aplikovat tyto metody na libovolné typy dat a jejich snadné zpracování. Další velkou výhodou je možnost zpracovávat data na jakékoli úrovni dělení shluků. Mezi nevýhodu patří ukončovací kritérium, po kterém je algoritmus zastaven, a také neschopnost zpracovávat již rozdělená data u většiny algoritmů.

2.2.2 Dělící metody

U těchto metod se kolekce dokumentů dělí do určitého počtu shluků. Tento počet je předem zadán uživatelem. Každý shluk je na počátku vytvořen náhodným vložením dokumentu z kolekce. Poté se ostatní dokumenty řadí do těchto shluků podle podobnosti. V dalších opakováních se pak jednotlivé dokumenty přesouvají z jednoho shluku do druhého a tím se získávají přesnější výsledky. Mezi nejpoužívanější dělící algoritmy patří k-mean, k-medoids a BiSec-k-medoids algoritmus. Největším problémem u těchto dělících metod je určení optimálního počtu shluků.[21]

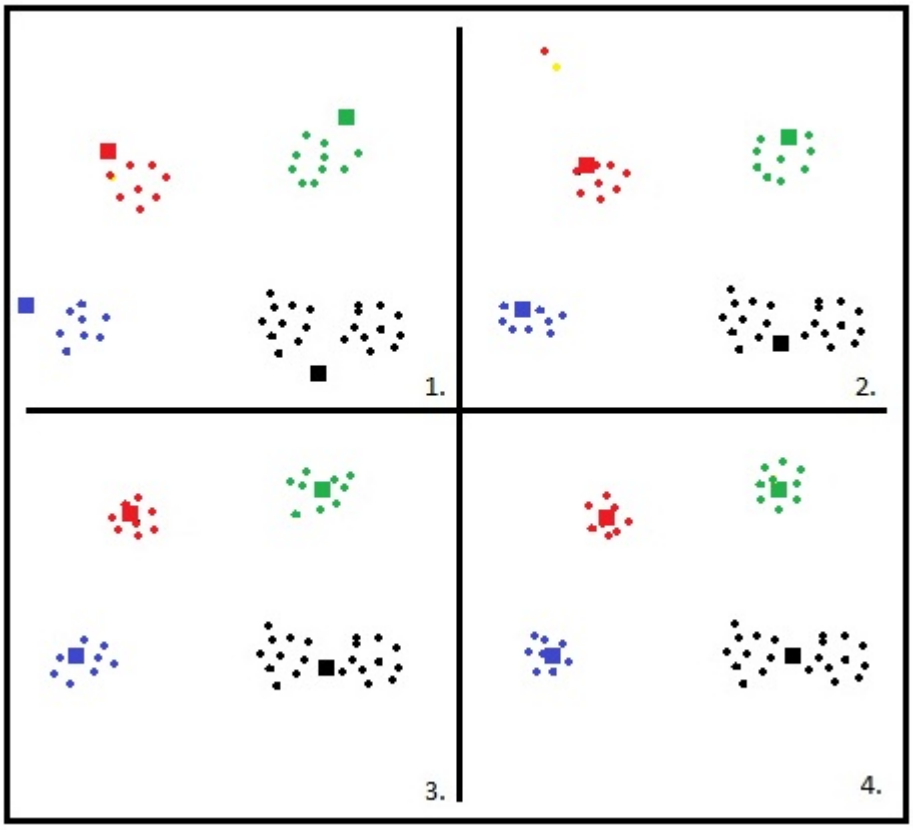
Určení optimálního počtu shluků

Existují dva základní přístupy jak určovat počet shluků pro kolekce dokumentů, a to heuristické metody a formální testy. Heuristické metody jsou používány častěji. Více o této metodě se lze dočíst v literatuře [25]. U formálních testů uživatel testuje kolekci dokumentů a mění požadovaný počet shluků. Po více opakováních pak z výsledků určí, které rozložení shluků je optimální.

K-Means algoritmus

Tento algoritmus je jeden z nejjednodušších a byl vytvořen v roce 1967 J. B. MacQueenem. Algoritmus dělí množinu objektů do podmnožin (shluků) tak, aby vzdálenost mezi shluky byla co největší. Celý algoritmus je složen ze 4 základních kroků, a to:

1. Jsou zvoleny náhodně středy shluků (centroids). Podle zvolených středů jsou ovlivněny všechny shluky. Při různém zvolení středů jsou i různé výsledky algoritmu. Mělo by platit pravidlo, že budou od sebe středy vzdáleny stejně.
2. Algoritmus přiřazuje ostatní dokumenty do shluků podle toho, jak jsou podobné ze středem shluku. Podobnost je vypočítaná většinou pomocí Euklidovy vzdálenosti.
3. Po rozřazení se přepočítá střed shluků pomocí průměrné Euklidovy vzdálenosti dokumentů v daném shluku.
4. Algoritmus probíhá do té doby, než se ustálí středy tvořených shluků.



Obr. 2.2: Iterace metody K-means

Tento algoritmus je sice nejlehčí, ale má řadu nevýhod. Na počátku shlukování musíme předem zadat počet požadovaných shluků. Tato skutečnost vede k nepřesným výsledkům, jelikož kolekce dokumentů je složena například z pěti tematických okruhů, ale my budeme požadovat jen čtyři shluky. Tento příklad je znázorněn na obrázku 2.2. Jednotlivé shluky jsou zde barevně rozděleny.

Obrázek 2.2 znázorňuje čtyři opakování algoritmu. Tečky na obrázku představují dokumenty a čtverečky středy shluků. Z obrázku lze poznat pět tematických okruhů. Uživatel však zadal na začátku algoritmu jen čtyři shluky, což způsobí nepřesnosti ve výsledku. Z těchto důvodů je dobře algoritmus K-means vícekrát opakovat s různým počtem požadovaných shluků a pak si vybrat pro nás nejlepší výsledek.

K-medoids

Tento algoritmus je velice podobný algoritmu K-means. Rozdíl je jen v určování středu shluků. Ten je zde určen pomocí reálného objektu, v našem případě dokumentu z dané kolekce. Ostatní kroky už jsou pak stejné jako u algoritmu K-means.

Dokumenty z kolekce	Hodnota	Euclidovská vzdálenost pro medoid $c_1 = 2$
2. soubor	7	$D(x_i, x_j) = \sqrt{\sum_{i=1}^{d} \lvert x_{il} - x_{jl} \rvert^2} = \sqrt{\lvert 7-2 \rvert^2} = 5$
3. soubor	5	$D(x_i, x_j) = \sqrt{\sum_{i=1}^{d} \lvert x_{il} - x_{jl} \rvert^2} = \sqrt{\lvert 5-2 \rvert^2} = 3$
4. soubor	12	$D(x_i, x_j) = \sqrt{\sum_{i=1}^{d} \lvert x_{il} - x_{jl} \rvert^2} = \sqrt{\lvert 12-2 \rvert^2} = 10$
6. soubor	10	$D(x_i, x_j) = \sqrt{\sum_{i=1}^{d} \lvert x_{il} - x_{jl} \rvert^2} = \sqrt{\lvert 10-2 \rvert^2} = 8$

Tab. 2.2: Výpočty Euclidovské vzdálenosti pro shluk $c_1 = 2$

Dokumenty z kolekce	Hodnota	Euclidovská vzdálenost pro medoid $c_2 = 9$
2. soubor	7	$D(x_i, x_j) = \sqrt{\sum_{i=1}^{d} \lvert x_{il} - x_{jl} \rvert^2} = \sqrt{\lvert 7-9 \rvert^2} = 2$
3. soubor	5	$D(x_i, x_j) = \sqrt{\sum_{i=1}^{d} \lvert x_{il} - x_{jl} \rvert^2} = \sqrt{\lvert 5-9 \rvert^2} = 4$
4. soubor	12	$D(x_i, x_j) = \sqrt{\sum_{i=1}^{d} \lvert x_{il} - x_{jl} \rvert^2} = \sqrt{\lvert 12-9 \rvert^2} = 3$
6. soubor	10	$D(x_i, x_j) = \sqrt{\sum_{i=1}^{d} \lvert x_{il} - x_{jl} \rvert^2} = \sqrt{\lvert 10-9 \rvert^2} = 1$

Tab. 2.3: Výpočty Euclidovské vzdálenosti pro shluk $c_2 = 9$

Jako příklad lze uvést kolekci 6 dokumentů. Tyto dokumenty jsou pro zjednodušení příkladu reprezentovány pouze jednou hodnotou a ne vektorem. Dokumenty mají hodnoty 2, 7, 5, 12, 9 a 10. Uživatel chce tyto dokumenty rozdělit na dva shluky, tedy *k=2*. Algoritmus K-medoids pracuje v těchto krocích:

1. Náhodně jsou vybrány reprezentanti dvou požadovaných shluků, takzvané medoidy. V našem případě například *c1 = 2* a *c2 = 9*.
2. K těmto dvěma medoidum se přiřadí ostatní dokumenty z kolekce. Ke zjištění podobnosti dokumentů s medoidy použijeme tzv. Euklidovskou vzdálenost podle vzorce 2.11. Výpočty jsou provedeny v tabulkách 2.2 a 2.3.
3. Určení nových medoidů ze shluků. Pro shluk *c1* je to 3. soubor, jelikož má v tabulce 2.2 nejmenší Euklidovskou vzdálenost a pro shluk *c2* je to 6. soubor, jelikož má nejmenší Euklidovskou vzdálenost v tabulce 2.3.
4. Krok 2 a 3 opakujeme do té doby, než se dokumenty přestanou přesouvat mezi shluky, nebo až dosáhneme prahové hodnoty.

BiSec-k-medoids

Tento algoritmus vychází z algoritmu K-medoids. Na počátku algoritmu je jeden velký shluk objektů, který se dále dělí. Algoritmus lze popsat ve třech bodech a to:

1. Algoritmus vybere shluk pro rozdělení. Tento výběr může být na základě jeho velikosti, míry podobnosti uvnitř shluků nebo obojího.
2. Pomocí algoritmu K-medoids se rozdělí vybraný shluk na dva nové shluky.
3. Tyto dva body se opakují do té doby, než dostaneme požadovaný počet shluků, nebo než klesne suma vzdálenosti pod určitou mez.

2.2.3 Metody založené na hustotě

Tyto metody považují objekty za body s n-rozměrem, kde je počet atributů daného objektu. Objekty se pak umísťují podle jejich atributů. Za shluky jsou považovány místa s velkou hustotou objektů, naopak za šum nebo odlehlé hodnoty považujeme místa s malou hustotou. Mezi metody založené na hustotě patří například metoda DENCLUE nebo DBSCAN.[11]

Metoda DENCLUE (Density-based Clustering)

Tato metoda provádí shlukování na základě funkce hustoty. Vychází z předpokladu, že objekty kolem sebe na sebe mají vliv. Vliv objektu na své okolí je pak modelován pomocí matematické funkce a funkce hustoty. Hodnota funkce hustoty je pak určena součtem funkcí vlivu jednotlivých objektů v daném bodě. Shluky jsou pak místa, kde funkce hustoty dosahuje lokálního maxima.

Funkce vlivu na okolní objekty je určena ze vzdáleností okolních objektů. V nejlehčím případě je definována tak, že má hodnotu 1 v menší vzdálenosti než τ a hodnotu 0 ve větší vzdálenosti. Nevýhodou této metody je ovšem určení parametru τ.[11]

Metoda DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

Tato metoda vznikla v roce 1996 a je asi nejznámějším zástupcem metod založených na hustotě. Pracuje s tzv. okolím bodu o poloměru. Když okolí bodu obsahuje stanovený minimální počet objektů, je tento objekt označen jako centrum shluku. Pak do shluku algoritmus přidá všechny objekty, které jsou v jeho okolí. Tím většinou dojde ke sloučení více shluků.

Mezi nevýhody této metody patří hlavně problém s nastavením okolí objektu ξ. Tento parametr pak ovlivňuje výsledky celé metody.[11]

2.2.4 Metody založené na mřížce

Tyto metody mapují objekty do n-rozměrné mřížky. Tato mřížka pak rozděluje prostor na konečný počet buněk a shlukování pak probíhá už jen nad touto mřížkou. Z tohoto důvodu algoritmy používané u těchto metod nejsou závislé na počtu vstupních datových objektů, ale pouze na velikosti mřížky. Mezi přednosti těchto metod patří rychlost zpracování dat. Mezi nejznámější metody založené na mřížce patří STING nebo WaveCluster.[11]

Metoda STING (Statistical Information Grid)

Tato metoda je určena pro zpracovávání kolekcí dokumentů s velkými soubory. Pracuje se statistickými informacemi uloženými v mřížce a byla vytvořena profesorem Wangem a kolektivem roku 1997.

Metoda WaveCluster

Tato metoda používá vlnovou transformaci k transformaci vstupního datového prostoru. Všechny tyto transformace jsou prováděny v mřížce. Hlavní kroky algoritmu probíhají následovně:

1. Rozdělí datový prostor pomocí mřížky a přiřadí objekty do buněk rozděleného prostoru.
2. Využije vlnové tranformace na rozdělení prostoru mřížky.
3. Po transformaci prostoru najde připojené komponenty v dílčích pásmech.
4. Připojí je k buňkám a namapuje objekty do shluků.

Tato metoda je velice rychlá a lze ji využít pro zpracování velkého množství dat.[26]

2.2.5 Metody založené na shlukování frekvenčních množin

Existuje několik přístupů, které využívají shlukování na základě frekvenčních množin. Tyto přístupy nejsou závislé na vektorovém modelu, a tím odpadá problém vysoké dimenzionality dat. Všechny tyto přístupy vycházejí z předpokladu, že každá podmnožina frekventované množiny musí být také frekventovaná, a tím lze dimenzi vstupních dat výrazně zmenšit. Mezi nejznámější metody založené na shlukování frekvenčních množin patří algoritmus FTC (Frequent Term-based Clustering) a HFTC (Hierarchical Frequent Term-based Clustering).[13]

FTC (Frequent Term-based Clustering)

Tento algoritmus vytváří pouze jednovrstvé shlukování s minimálním přesahem. Shluky vytvořené tímto algoritmem jsou nestrukturované a pokrývají celou vstupní databázi. Shluky, které jsou tvořeny, jsou známé již při jejich vytvoření a není tak nutné jejich popisy dodatečně generovat. Algoritmus pracuje odspodu vzhůru, takže jsou na počátku vytvořeny všechny frekventované množiny termů.[13]

HFTC (Hierarchical Frequent Term-based Clustering)

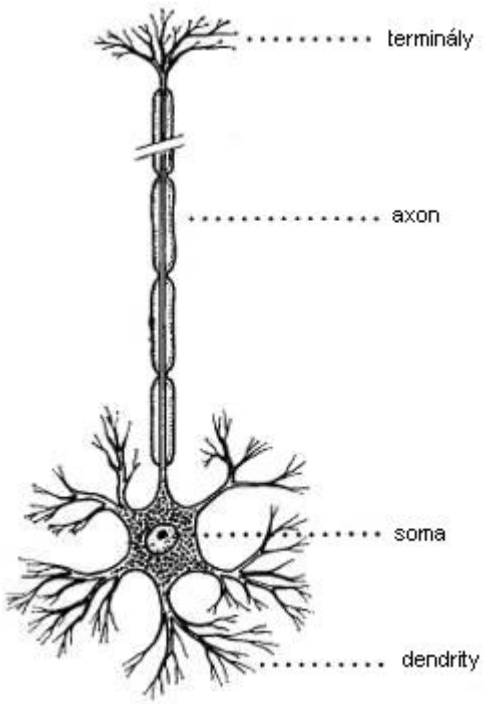
Tento algoritmus vytváří hierarchickou neboli stromovou strukturu shluků. Výsledkem je strukturované shlukování se vztahy mezi každým shlukem a jeho předchůdci. Tento algoritmus pak může být založen na opakovaném FTC algoritmu, kdy FTC algoritmus není aplikován na celou množinu frekventovaných množin, ale jen na množinu frekventovaných množin jedné úrovně. Přesnost tohoto algoritmu je srovnatelná s algoritmem Bisecting K-Medoids. Není však vhodné tento algoritmus použít pro kolekci více než 6000 dokumentů.[13]

2.2.6 Neuronové sítě

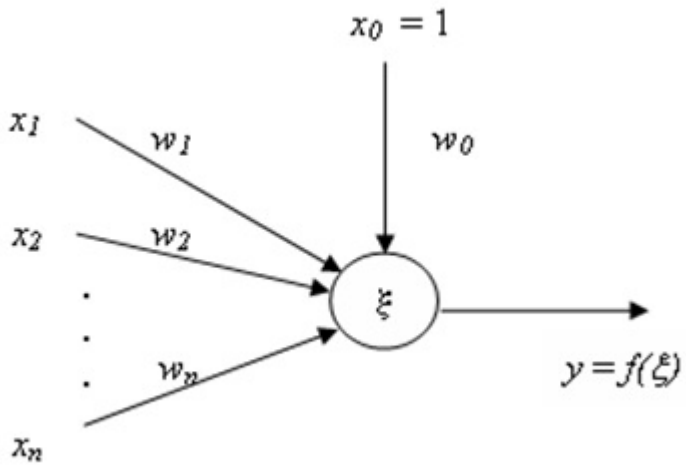
Neuronové sítě vycházejí z principů sítě v lidském mozku, kde je základním stavebním kamenem neuron. Do něj je veden velký počet vstupů (dendritů) a pouze jeden výstup (axon). Ten se může ještě větvit do tzv. terminálů. Jednotlivé neurony jsou pak propojeny synapsí. Biologický neuron v mozku je ukazán na obrázku 2.3. Na základě tohoto modelu byl vytvořen formální neuron, který je zobrazen na obrázku 2.4. Tento formální neuron je základním stavebním kamenem všech neuronových sítí.[24]

$x_1, \ldots x_n$ jsou vstupy neuronu
x_0 je formální vstup
$w_1, \ldots w_n$ jsou váhy spojů
ε je vnitřní potenciál
$y = f(\varepsilon)$ je výstup neuronu

Neuronová síť je tvořena těmito formálními neurony, které jsou mezi sebou propojeny spoji. Tyto spoje mají určitou váhu a změnami těchto vah se neuronové sítě učí. Další důležitou vlastností u neuronových sítí je, že umí nacházet souvislosti v trénovacích datech a tyto souvislosti pak vyjádřit pomocí vah. Díky této vlastnosti pak mohou neuronové sítě zpracovávat i informace, na které nebyly trénovány. Nejznámější metodou založenou na těchto principech je metoda SOM (Self organizing feature maps) neboli Kohonenova samoorganizující neuronová síť.[15]



Obr. 2.3: Biologický neuron[28]



Obr. 2.4: Formální neuron

SOM (Self organizing feature maps)

Tuto metodu vytvořil v 80. letech Teuvo Kohonen. Základní myšlenkou této metody je poznatek, že mozek používá vnitřní prostorovou reprezentaci dat k uchovávání informací. Data přicházející do takové sítě se mohou transformovat do vektorů a zakódovat do neuronové sítě. Převádí se zde vlastně vícerozměrný prostor vstupních dat do prostoru sítě s menší dimenzí. Většina spojů mezi neurony je vedena pouze mezi sousedními neurony, což značně ulehčuje tuto metodu. Metoda SOM v průběhu zpracovávání dokumentů vytváří v Kohonenově mapě shluky, ke kterým jsou následně přiřazeny vstupní dokumenty.[17]

Problémem SOM je rozměr vektorů vstupních dokumentů, kdy příliš velké vektory metodu zpomalují a dělají méně efektivní. Aby se těmto nepříjemnostem předcházelo, používá metoda SOM různé předzpracující techniky redukce dimenze vstupních vektoru.[16] Mezi tyto techniky patří:

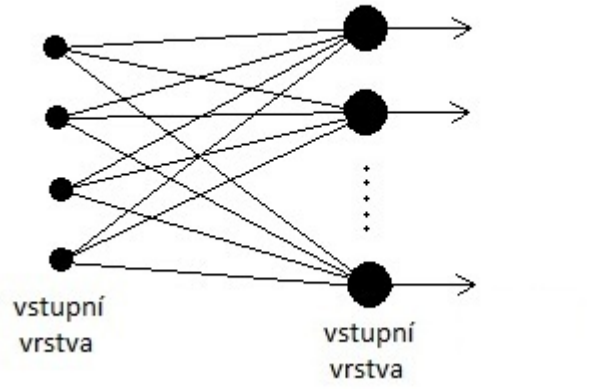
- Latentně-sémantická indexace (LSI). Tato metoda redukuje dimenzi tak, že pomocí SVD (singular-value decomposition) rozkladu je vytvořen předem určený (malý) počet faktorů, které charakterizují daný dokument.
- Náhodná projekce histogramů. U této metody bylo experimentálně zjištěno, že náhodné promítnutí vektorů histogramu do prostoru s menší dimenzí vede k dobrým výsledkům bez ztráty informací odlišujících jednotlivé dokumenty.
- Mapy pro slovní kategorie. U této metody jsou shlukována slova do slovních kategorií, přičemž metoda slučuje synonyma a různé tvary stejných slov. Do původní Kohonenovy mapy pak vstupují vektory s dimenzí rovnou počtu slovních kategorií.[16]

Metoda SOM je složena jen ze vstupní a výstupní vrstvy. Počet vstupů je roven počtu vstupních dat, tzv. dimenzi vstupního prostoru. Každý vstup je propojen s každým neuronem v mřížce, který je zároveň výstupem. Počet výstupů je pak roven počtu neuronů v mřížce. Čím je počet neuronů větší, tím je lepší i pokrytí vstupního prostoru, ale roste časová náročnost.[12]

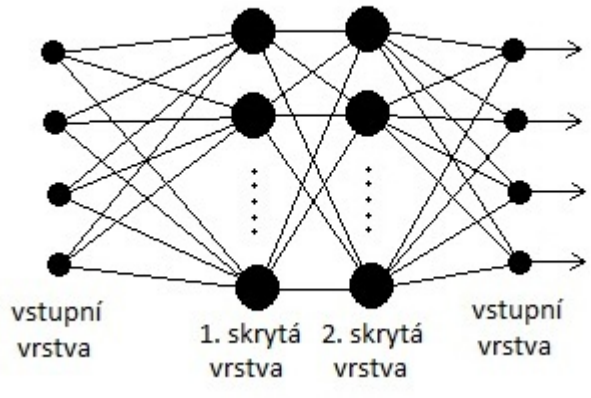
V metodě SOM lze využít dvou druhů sítí, a to jednovrstvé a vícevrstvé sítě.[17] Obě sítě jsou ukázány na obrázku 2.5 a 2.6.

Metoda SOM se skládá z šesti základních kroků:[18]

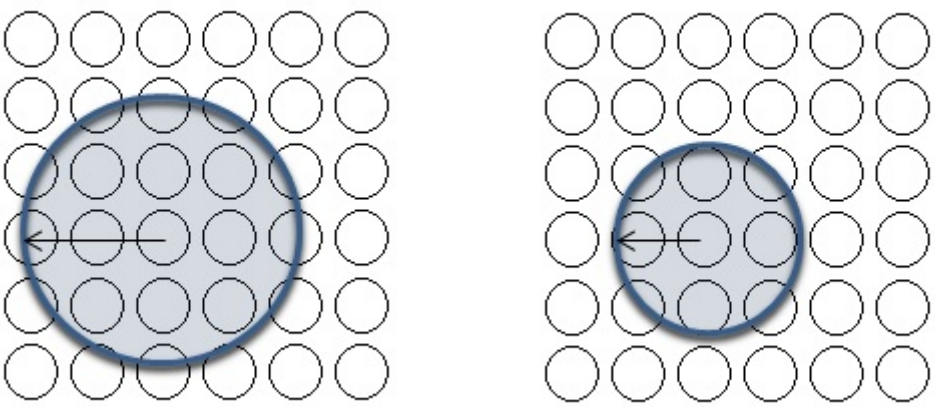
1. Inicializace sítě: V tomto kroku je vytvořena síť neuronů. Každý neuron má svoji pozici v mřížce a svůj vektor vah. Vektor vah je pak náhodně nastaven na hodnoty 0 až 1 a má normalizovanou délku. Je zde také inicializován parametr učení η, který určuje velikost změn při adaptaci vah. Tento parametr je nastaven na hodnotu blízkou jedné a během učení se snižuje k nule. S tímto



Obr. 2.5: Jednovrstvá Kohonenova síť



Obr. 2.6: Vícevrstvá Kohonenova síť



Obr. 2.7: Změna okolí vítězného neuronu

parametrem lze experimentovat a zkoušet nastavit takový počáteční parametr, který bude vykazovat nejlepší výsledky. Tento parametr také způsobuje, že se neuronová síť mění více na začátku učení. Parametr učení se počítá pomocí vzorce 2.20:

$$\eta(t) = \eta_0 exp\left(-\frac{t}{\gamma}\right), \tag{2.20}$$

kde η_0 je počáteční rychlost učení, $t = 1, 2, 3, \ldots, n$ je probíhající iterace a γ je celkový počet iterací.

2. Předložení vstupního vektoru: V tomto kroku je neuronové síti předložen vstupní vektor $x_1(t), x_2(t), \ldots x_n(t)$, kde $x_i(t)$ je vstup uzlu i v čase t. Tento vektor je náhodně vybraný normalizovaný vektor z kolekce dokumentů.
3. Výpočet vzdálenosti okolí: Kolem každého neuronu v síti je definováno okolí, ve kterém se bude provádět změna vah. Toto okolí lze vypočítat pomocí vzorce 2.21:

$$\tau(t) = \tau_0 exp\left(-\frac{t}{\lambda}\right), \tag{2.21}$$

kde t nabývá hodnot $1, 2, 3, \ldots n$, τ_0 značí průměrnou velikost mřížky počítanou podle vzorce $(dlka + vka)/2$ a $\lambda = \gamma / \log \tau_0$.

Určit, zda sousední neuron do okolí patří či nikoliv, lze pomocí Pythagorovy věty. Graficky je toto okolí znázorněno na obrázku 2.7.

4. V tomto kroku vypočteme pro každý neuron v síti Euklidovskou vzdálenost a vybereme ze sítě neuron s nejmenší vzdáleností. Euklidovská vzdálenost je

počítána pomocí vzorce 2.11.

5. Úprava vah: Váhy vítězného neuronu a jeho okolí jsou upraveny podle vztahu 2.22:

$$m_{ij}(t+1) = m_{ij}(t) + \eta(t)\, h(v,t)\,(x_i(t) - m_{ij}(t)), \tag{2.22}$$

kde i a j značí pozici neuronu v mřížce a funkce $h(v,t)$ vliv vzdálenosti na učení. Lze vypočítat podle vzorce 2.23:

$$h(t) = exp\left(-\frac{dist}{2 * \tau(t)^2}\right), \tag{2.23}$$

kde *dist* je vzálenost uzlu od vítěze a $\tau(t)$ je okolí kolem vítěze.

6. Návrat k bodu 2 a provádění další iterace.

Aby metoda SOM měla nejlepší výsledky, je na počátku voleno velké okolí neuronů a velký vliv učeného vzoru na změnu vah neuronů. Po několika iteracích se vytvářejí jednotlivé shluky a pak se okolí neuronu i vliv učeného vzoru pomalu zmenšuje.[12]

3 APLIKACE PRO SHLUKOVÁNÍ TEXTOVÝCH DAT

3.1 Úvod

V minulé kapitole byl popsán teoretický základ potřebný k implementaci aplikace pro shlukování textových dat. V této kapitole je vytvořená aplikace podrobně popsána a vysvětlena. Aplikace je naprográmovaná v rozšířeném programovacím jazyce Java a bylo použito vývojové prostředí Eclipse. Toto vývojové prostředí je šířeno pod open-source licencí a je v něm tvořena většina javovských aplikací na VUT v Brně. Aplikace vytvořena v rámci diplomové práce se skládá ze šesti základních balíčků, a to:

1. Core: Tento balíček slouží ke zpracování textových dat a převodu těchto dat na vektorizovaný tvar. Obsahuje dvě třídy, a to třídu *Document* a *DocumentSet*. Třída *Document* představuje datový objekt, který má atributy obecného dokumentu. Ve třídě *DocumentSet* pak jsou veškeré metody potřebné k převodu textového dokumentu na datový objekt a k následné vektorizaci.
2. Execute: Tento balíček obsahuje stejnojmenou třídu, ve které jsou spouštěny všechny metody daných algoritmů.
3. Kmethods: V tomto balíčku jsou obsaženy dvě třídy s algoritmy na shlukování dokumentů, a to K medoids a BiSec K medoids.
4. Kohonen: V tomto balíčku je zpracována metoda SOM. Balíček obsahuje dvě třídy, a to Neuron a KohonenMap. Třída Neuron představuje základní stavební prvek neuronové sítě, na které je metoda SOM založena. V třídě KohonenMap pak jsou jednotlivé metody potřebné pro SOM algoritmus.
5. SampleSet: Tento balíček obsahuje validační množinu dokumentů, které byly používány pro testování aplikace.
6. ClusterSet: V tomto posledním balíčku je soubor obsahující slova, podle kterých se na počátku provádí vektorizace. Tyto slova lze v grafickém rozhraní do souboru clusterSet.txt ukládat.

V následujících podkapitolách budou popsány jednotlivé metody a atributy, které jsou v aplikaci použity. Budou zde také prezentovány printscreeny z grafického rozhraní aplikace a stručný navod k použití této aplikace.

3.2 Balíček core

Tento balíček je jádrem celého programu. Jsou v něm načítány kolekce dokumentů, které jsou pak zvektorizovány a ukladány do pole dokumentů. Základním stavebním prvkem této aplikace je právě objekt zvaný dokument.

3.2.1 Třída Document

Každý dokument obsahuje atributy jako:

- name: tento atribut uchovává jméno dokumentu.
- noOfWeights: tento atribut značí počet vah každého dokumentu.
- noOfClusters: hodnota tohoto atributu udává počet shluků, které chce uživatel vytvořit. Tento atribut však není použit pro všechny tři shlukovací metody. Je použit pouze pro metodu K medoids a BiSec k medoids. U těchto dvou metod je nutné totiž předem zadat počet výsledných shluků.
- fileContent: v tomto atributu jsou uložena všechna slova v daném dokumentu.
- weights: v tomto atributu jsou uloženy váhy souborů po převedení na vektorový tvar.
- euclidDistances: tento atribut ukladá Euclidovskou vzdálenost pro porovnávání a dělení dokumentů u metody K medoids a BiSec k medoids.

Třídu *Document* tvoří také metody. Jednou z hlavních metod je metoda *calculateDistance*. Tato metoda počíta Euclidovskou vzálenost daného dokumentu od dokumentu, který je jí předložen. Ostatní metody v tomto dokumentu jsou už jen nastavování atributů a získávání těchto atributů.

3.2.2 Třída DocumentSet

V této třídě je vše důležité týkající se zpracovávání dokumentů. Třída obsahuje tyto atributy:

- pathClusterSet: v tomto atributu je uložena cesta k dokumentu, podle kterého bude prováděna vektorizace.
- pathTrainingSet: tento atribut označuje cestu ke kolekci souborů, které chceme shlukovat.
- documents: v tomto atributu jsou uloženy všechny dokumenty v kolekci.
- clusterSet: do tohoto atributu se načtou všechna slova ze souboru clusterSet.txt.
- sampleSet: do tohoto atributu jsou uloženy všechny načtené soubory z kolekce.

- occurenceOfWordInFileSet: tento atribut ukládá počty výskytů slov v kolekci dokumentů, podle kterých se provádí vektorizace.
- filesFrequency: zde jsou uložena jednotlivá slova ze všech souborů z kolekce. Zároveň je v tomto atributu uložena i četnost slova v kolekci dokumentů.

Metody použité v této třídě jsou:

- readClusterSet: tato metoda načítá soubor, podle kterého bude kolekce souborů převáděna na vektory.
- readSampleSett: v této metodě jsou načteny všechny soubory z kolekce dokumentů do pole dokumentů.
- createContentVectors: tato metoda zjištuje počet výskytů slov, podle kterých je prováděna vektorizace v dokumentech.
- countWordFrequencyInFile: metoda pro výpočet četnosti slov v jednom dokumentu.
- countWordFrequencyInFileSet: metoda pro výpočet četnosti slov v celé kolekci dokumentů.
- createWeightVectors: metoda která převádí dokumenty do vektorového tvaru.

3.3 Balíček execute

Tento balíček obsahuje dvě třídy, a to třídu Aplication a třídu Execute.

3.3.1 Třída Aplication

Tato třída je hlavní třídou celé aplikace. Je tvořena podle návrhového vzoru singleton, takže všechny metody a proměnné obsažené v této třídě jsou vždy dostupné. Mezi nejdůležitější atributy této třídy patří:

- KMedoids_array3D: v tomto atributu je uložen průběh shlukování u metody K medoids. Tento atribut je na konci také graficky znázorněn.
- BiSec_array3D: tento atribut je podobný pro metodu BiSec K medoids.

Metody použité v této třídě jsou například:

- getBiSec_array3D: metoda pro přístup k atributu BiSec_array3D.
- getKMedoids_array3D: metoda pro přístup k atributu KMedoids_array3D.
- doKohonenMap: metoda pro vytvoření a volání třídy KohonenMap.
- doBiSecK_medoids: metoda pro vytvoření a volání třídy BiSec_K_medoids.
- doK_medoids: metoda pro vytvoření a volání třídy K_medoids.

3.3.2 Třída Execute

Pomocí této třídy se celá aplikace spouští.

3.4 Balíček kohonen

V tomto balíčku je zpracována metoda SOM. Balíček obsahuje dvě třídy a to třídu *Neuron* a *KohonenMap*.

3.4.1 Třída Neuron

Tato třída znázorňuje základní stavební prvek neuronové sítě. Z objektů tohoto typu se skládá celá neuronová mřížka použita v metodě SOM. Každý neuron musí mít tyto atributy:

- position_x: atribut označující pozici v X-ové ose.
- position_y: atribut označující pozici v Y-ové ose.
- noOfWeights: atribut ukládající počet vah.
- euclidDistance: atribut ukládající Euclidovskou vzdálenost k předloženému dokumentu.
- weights: atribut ukládající váhy neuronu.

Třída *Neuron* dále obsahuje dvě hlavní metody a to metodu *calculateDistance* a *adjustWeights*. V metodě *calculateDistance* je počítana Euclidovská vzdálenost od předloženého dokumentu a v metodě *adjustWeights* jsou upravovány váhy neuronu.

3.4.2 Třída KohonenMap

V této třídě je naprogramována celá metoda SOM. Jsou v ní použity tyto atributy:

- position_x: v tomto atributu je uložena X-ová pozice právě zpracovávaného neuronu.
- position_y: v tomto atributu je uložena Y-ová pozice právě zpracovávaného neuronu.
- noOfWeights: atribut ukládající počet vah.
- noOfIterations: atribut, ve kterém je uložen počet iterací.
- learningRate: atribut, ve kterém je nastavena počáteční rychlost učení.
- learningRateConst: do tohoto atributu se ukládá aktuální rychlost učení. Rychlost učení se mění podle počtu iterací.
- sizeOfNeighborhood: atribut udávající vzdálenost sousedů od vítěze. Podle tohoto atributu se pak určuje jestli se budou měnit váhy okolních neuronů.

- winnerNeuron: v tomto atributu je uložen vítězný neuron pro danou iteraci.
- clustersOfDoc: v tomto atibutu jsou uloženy všechny dokumenty výsledných shluků.
- neurons: atribut, ve kterém jsou uloženy všechny neurony použity v metodě SOM.

Metody použity ve třídě *KohonenMap* jsou:

- findWinerNeuron: tato metoda vyhledává vítězný neuron pro danou iteraci.
- countSizeOfNeighborhood: v této metodě je počítána velikost okolí vítězného neuronu.
- countSpeedOfLearning: tato metoda počíta rychlost učení pro danou iteraci.
- changeWeights: v této metodě jsou měněny váhy neuronu. Tím probíhá učení Kohonenovi sítě.
- makeIterations: v této metodě jsou prováděny všechny zadané iterace.
- createCluster: tato metoda předkládá naučené neuronové síti dokumenty v kolekci. Tyto dokumenty se pak podle podobnosti dělí do jednotlivých shluků.

3.5 Balíček kMethods

V tomto balíčku jsou zpracovávány další dvě shlukovací metody, a to K medoids a metoda BiSec k medoids. V balíčku jsou dvě třídy se stejným názvem, ve kterých jsou metody naprogramovány.

3.5.1 Třída K_medoids

V této třídě je naprogramována celá shlukovací metoda K_medoids, která obsahuje tyto atributy:

- noOfClusters: atribut, ve kterém je uložen počet požadovaných shluků.
- noOfWeights: atribut s počtem vah.
- documents: v tomto atributu jsou uloženy všechny zpracovávané dokumenty.
- winnerDocs: tento atribut ukládá vítězné dokumenty jednotlivých shluků.
- clusters: atribut, který ukladá dokumenty do jednotlivých shluků.
- randNumbers: atribut, do kterého jsou generovány náhodná čísla. Tyto čísla se neopakují a generují se z množiny o velikosti počtu dokumentů v kolekci.

Dále obsahuje třída *K_medoids* také tyto metody:

- splitToClusters: pomocí této metody se nastaví počet vah a shluků pro každý z dokumentů. Také se zde počíta Euklidovská vzdálenost k vítěznému dokumentu.
- fillArray: tato metoda vytváří pole typu integer, ve kterém jsou generovány neopakující se celá čísla v rozmezí od nuly do počtu souborů v kolekci.
- find: metoda pro zjištění, zda se generované číslo už v poli nenachází.
- makeClusters: v této metodě se dokumenty dělí podle podobnosti k *winerDoc* do jednotlivých shluků.
- winerDocs: tato metoda určuje vítězné dokumenty z vytvořených shluků.
- doCalculation: v této metodě se provádí dělení na shluky a následný výpočet vítězných dokumentů v cyklu. Cyklus končí, když se žádný dokument nepřesune z jednotlivých shluků nebo když je dosaženo tzv. prahové hodnoty.

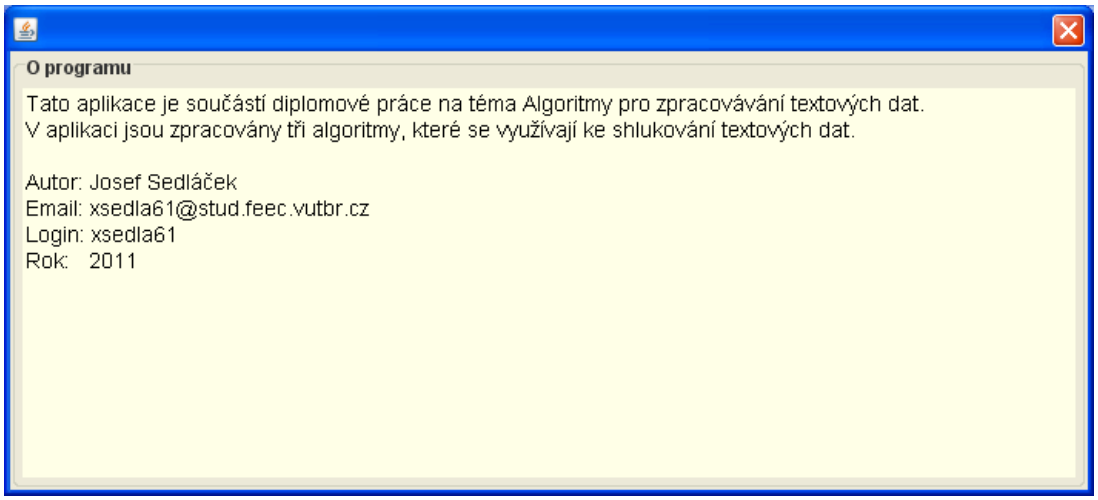
3.5.2 Třída BiSec_K_medoids

Tato třída je podobná třídě K_medoids. Rozdíl je zde v tom, že se kolekce dokumentů rozdělí vždy na dva shluky. Jeden se dále nedělí a druhý se hiearchicky dělí na další a další až do té doby, než se vytvoří požadovaný počet shluků. Výběr shluku, který budeme dělit, se může provádět buď podle velikosti shluku nebo podle odlišnosti dokumentů v daném shluku. Třída obsahuje stejné atributy jako třída K_medoids. Dále také obsahuje některé atributy navíc, jako:

- index: atribut, ve kterém je počítán počet vytvářených shluků.
- twoClusters: v tomto atributu jsou ukládány dokumenty, které jsou rozděleny do dvou shluků.

Metody, které jsou u tohoto algoritmu použity, jsou také podobné jako u K_medoids. Zmíněny jsou zde tedy jen ty odlišné:

- winerDoc: tato metoda je odlišná od stejnojmené metody ve třídě K_medoids v tom, že zde jsou dva vítězné dokumenty náhodně vybrány ze shluku, který se bude dále dělit.
- selectionOfCluster: v této metodě se určuje, jaký ze dvou vytvořených shluků se bude dále dělit. Shluk lze určit podle velikosti nebo odlišnosti.
- doCalculation: v této metodě se dělí kolekce vždy na dva shluky, určí se shluk, který bude dále dělen a vítězné dokumenty těchto shluků. Hlídá se zde počet vytvořených shluků a po vytvoření tohoto počtu shluků se metoda ukončí.



Obr. 3.1: Grafické uživatelské rozhraní AppAbout

3.6 Balíček gui

V tomto balíčku je zpracováváno grafické rozhraní celé aplikace. Balíček tvoří šest grafických rozhraní, a to, SOM, K_medoids, BiSecK_medoids, AppAbout, AppDescription a hlavní grafické rozhraní MainJFrame. V následujících podkapitolách budou jednotlivá grafická rozhraní popsány podrobněji.

3.6.1 Grafické uživatelské rozhraní AppAbout

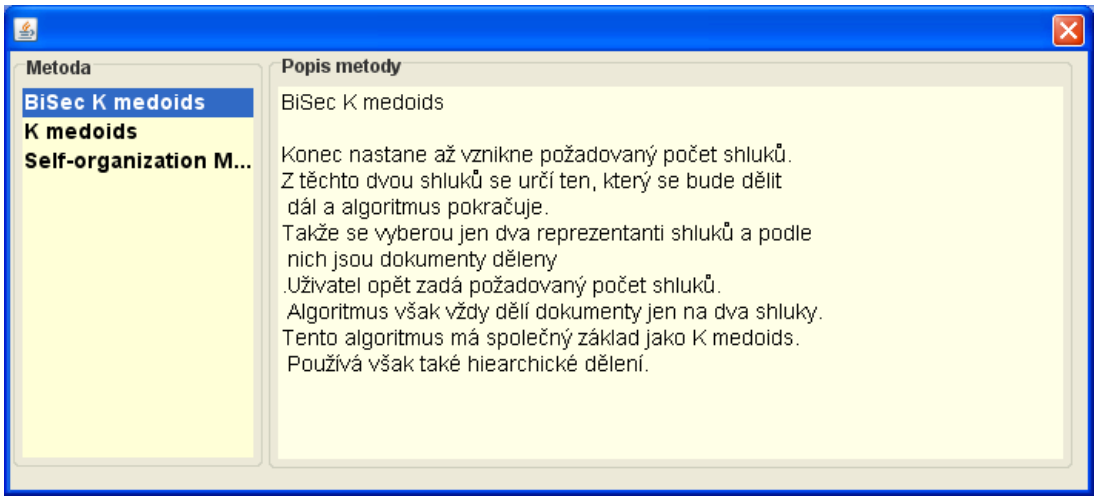
V tomto grafickém rozhraní je popsáno, proč byla aplikace vytvořena a kým. Je tvořeno dvěmi konponentami, a to jPanel a jTextArea. Na obrázku 3.1 je tato třída ukázána.

3.6.2 Grafické uživatelské rozhraní AppDescription

V tomto grafickém rozhraní jsou popsány jednotlivé algoritmy implementované v aplikaci. Je tvořeno dvěma jPaneli. První obsahuje AlgList, ve kterém se dá vybírat mezi jednotlivými algoritmy, a druhý obsahuje jTextArea, v kterém jsou informace o daném algoritmu. Na obrázku 3.2 je tato grafická třída ukázána.

3.6.3 Grafické uživatelské rozhraní k třídám K_medoids, BiSecK_medoids a SOM

Grafická rozhraní tříd K_medoids a BiSecK_medoids jsou téměř totožná, jelikož u těchto metod uživatel zadává stejný parametr, a to počet shluků, které chce vytvořit. U grafického rozhraní K_medoids se pak ještě zadává počet iterací. Mezi



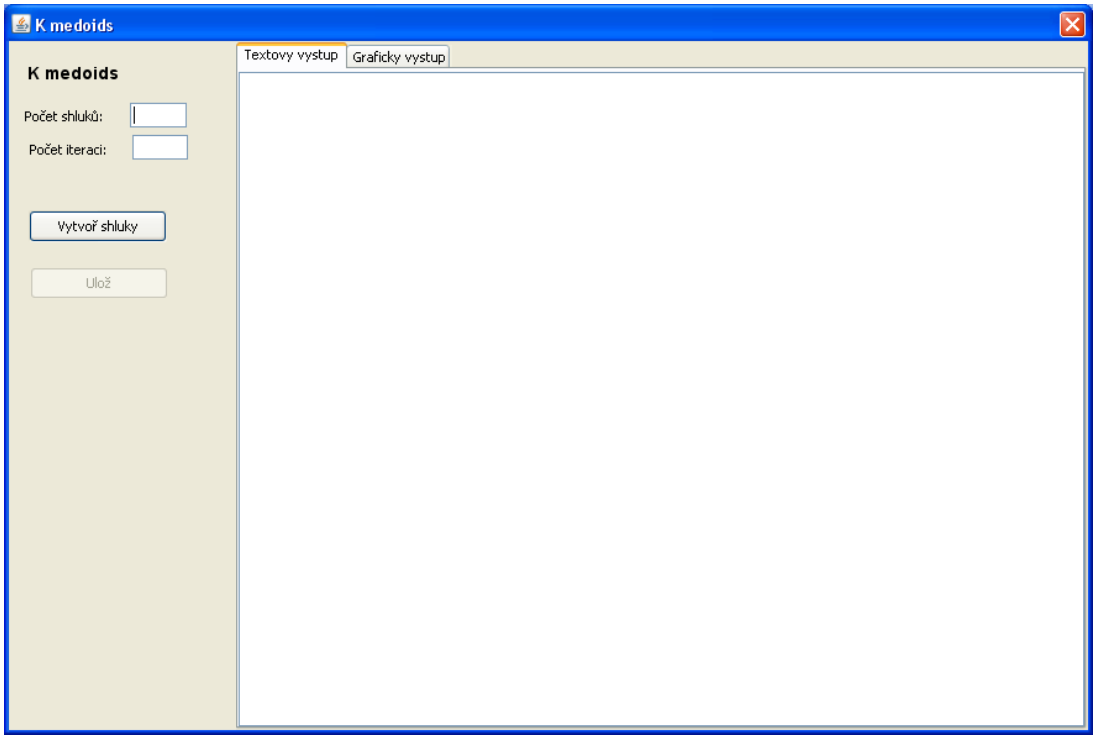
Obr. 3.2: Grafické uživatelské rozhraní AppDescription

jednu z hlavních komponent u těchto grafických rozhraní patří jTabbedPane. Pomocí této komponenty si může uživatel přepínat mezi textovým a grafickým znázorněním výsledků daných metod. Grafická rozhraní tříd K_medoids a BiSecK_medoids jsou ukázány na obrázku 3.3 a 3.4.

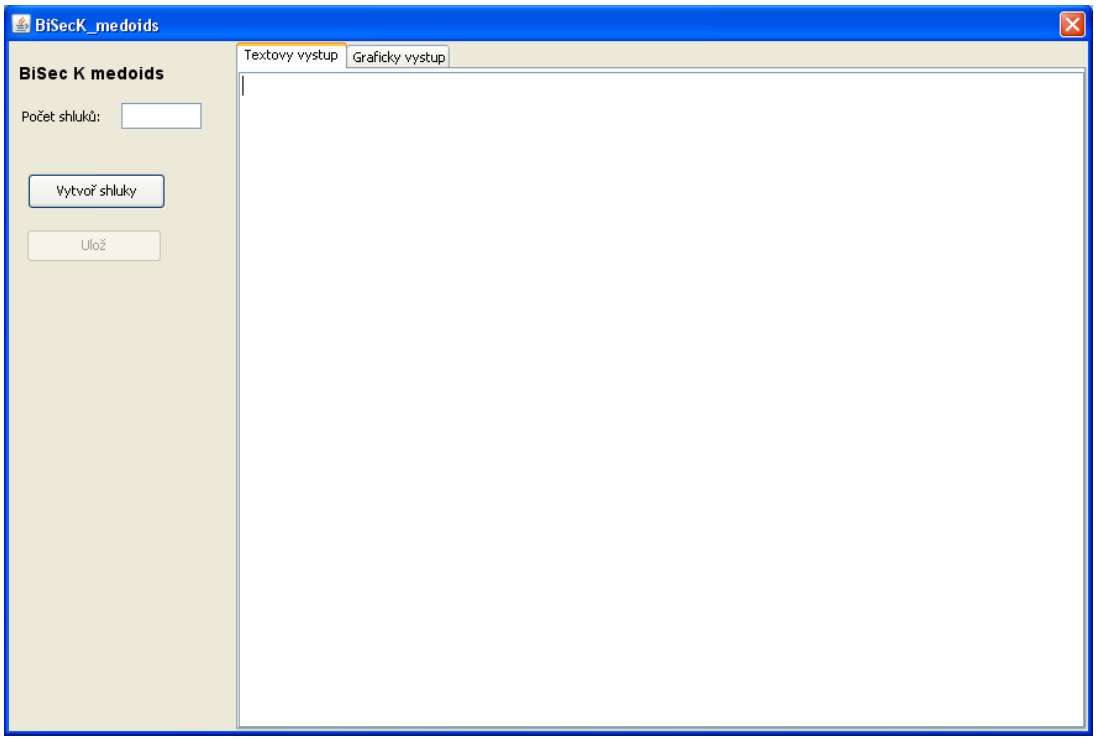
Grafické rozhraní k třídě SOM je úplně jiné než u předchozích metod. U tohoto algoritmu zadává uživatel velikost mřížky a také počet iterací, po kterých se učení sítě zastaví. Může se zde také zadávát například počáteční rychlost učení či velikost okolí vítězného neuronu. Tyto parametry však v mé aplikaci může měnit jen programátor nikoliv běžný uživatel. Toto opatření je vytvořeno kvůli lepším výsledkům algoritmu. Běžný uživatel, který neví, jak SOM algiritmus pracuje, by mohl zadávat hodnoty, které by vedly k horším výsledkům. Na obrázku 3.5 je ukázka grafického rozhraní třídy SOM.

3.6.4 Grafické uživatelské rozhraní MainJFrame

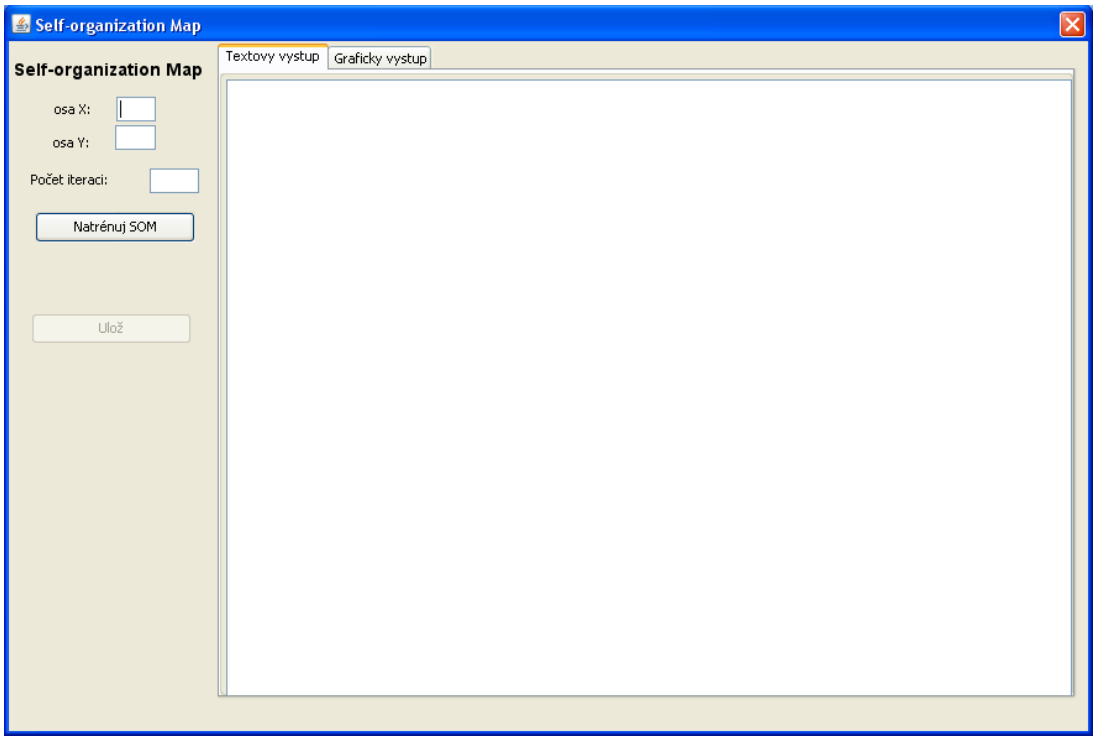
Toto grafické rozhraní je hlavním rozhraním celé aplikace. Po spuštění aplikace se uživateli zobrazí právě toto rozhraní. Základním kamenem je komponenta jFrame, na které jsou umístěny všechny ostatní. Je zde například menu pro otevření kolekce dokumentů či zjištění informací o aplikaci. Dále jsou zde tlačítka pro načtení souborů a převod dokumentů na vektory podle slov, které zaškrtne uživatel. Mezi poslední významné komponenty patří radiobuttony, pomocí kterých si uživatel vybere jaký algoritmus chce pro dělení dokumentů na shluky použít. Na počátku je u tlačítek i radiobuttonů nastaveno enable na false, a to z toho důvodu, aby byl uživatel částečně veden samotnou aplikací a dělal akce ve správném pořadí. Po správné akci se vždy některá komponenta povolí a uživatel může pokračovat v dalších akcích. Na



Obr. 3.3: Grafické uživatelské rozhraní k třídě K_medoids



Obr. 3.4: Grafické uživatelské rozhraní k třídě BiSecK_medoids



Obr. 3.5: Grafické uživatelské rozhraní k třídě SOM

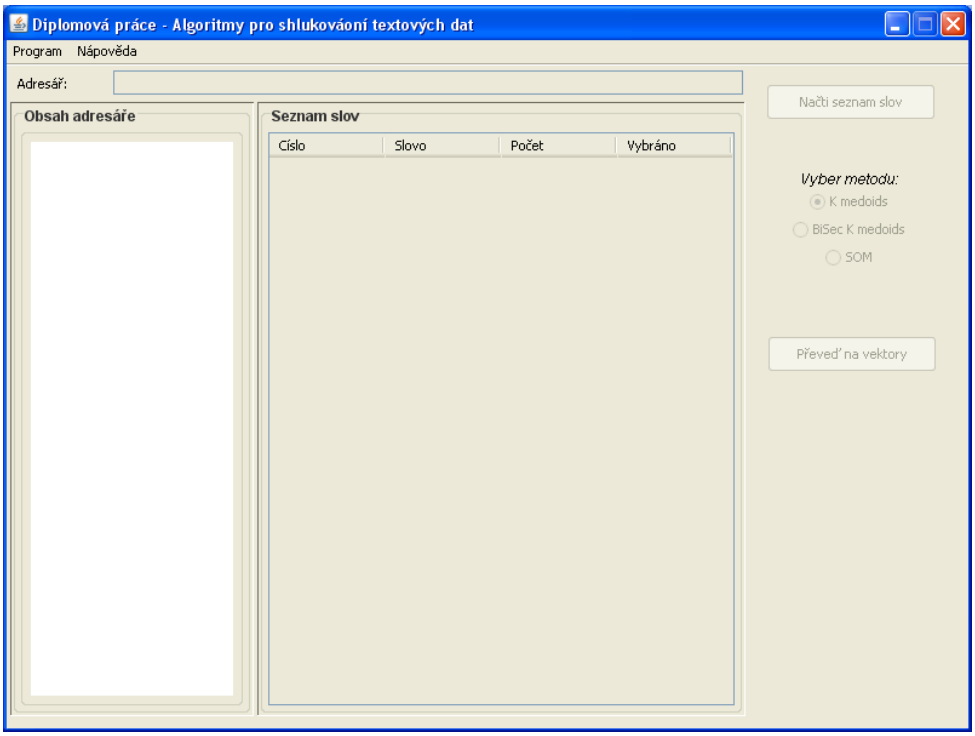
obrázku 3.6 je pak celá tato grafická třída ukázána.

3.7 Validační množina SampleSet

Tato validační množina dokumentů byla vytvořena pro testování implementovaných algoritmů. V množině je obsaženo sto dokumentů, které jsou řazeny do deseti tématických skupin. Podle toho do jaké skupiny dokument patří, je dokument pojmenován. Dokumenty jsou tedy děleny na auta, barvy, cisla, hokej, jmena, mesice, prvky, sport, zeme a zvirata. V každém dokumentu je vždy přibližně deset významových slov. Tato slova jsou vybírána z množiny asi třiceti slov. Při testování algoritmů tak hned vidíme, jak algoritmus dokumenty rozdělil a jestli toto dělení odpovídá zařazení do skupin.

3.8 Návod k aplikaci

Aplikaci lze spustit dvěma způsoby. První z nich je spuštění aplikace přímo v prostředí Eclipse pomocí třídy Execute. Druhým a snažším způsobem je spuštění souboru xsedla61.jar. Tento soubor je samostatně spustitelný a lze s ním pracovat na

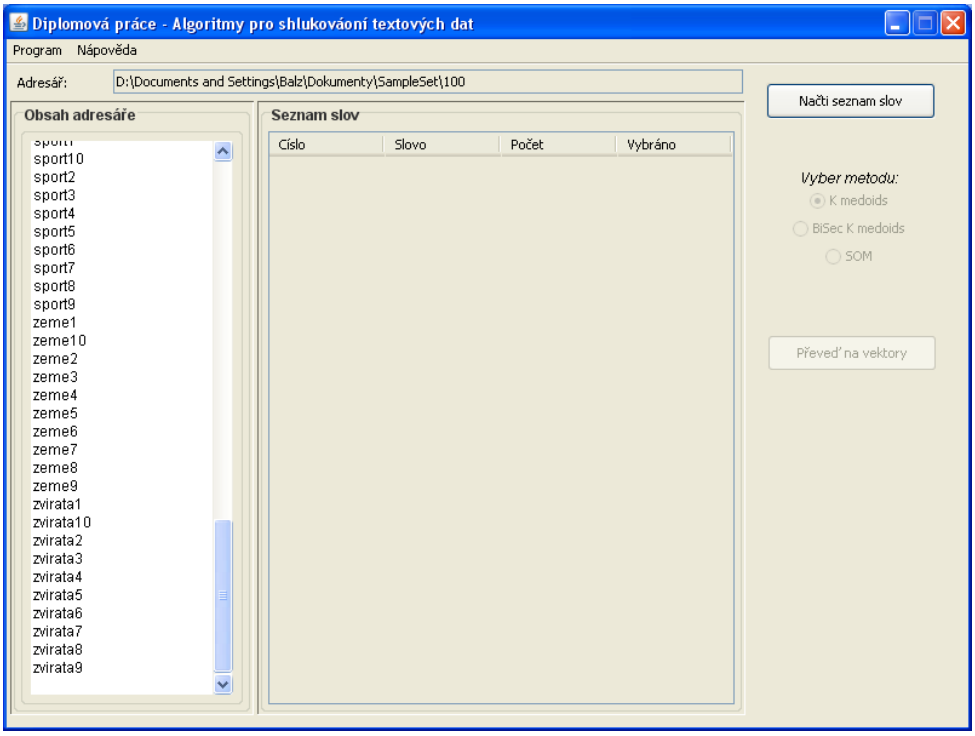


Obr. 3.6: Hlavní grafické uživatelské rozhraní aplikace

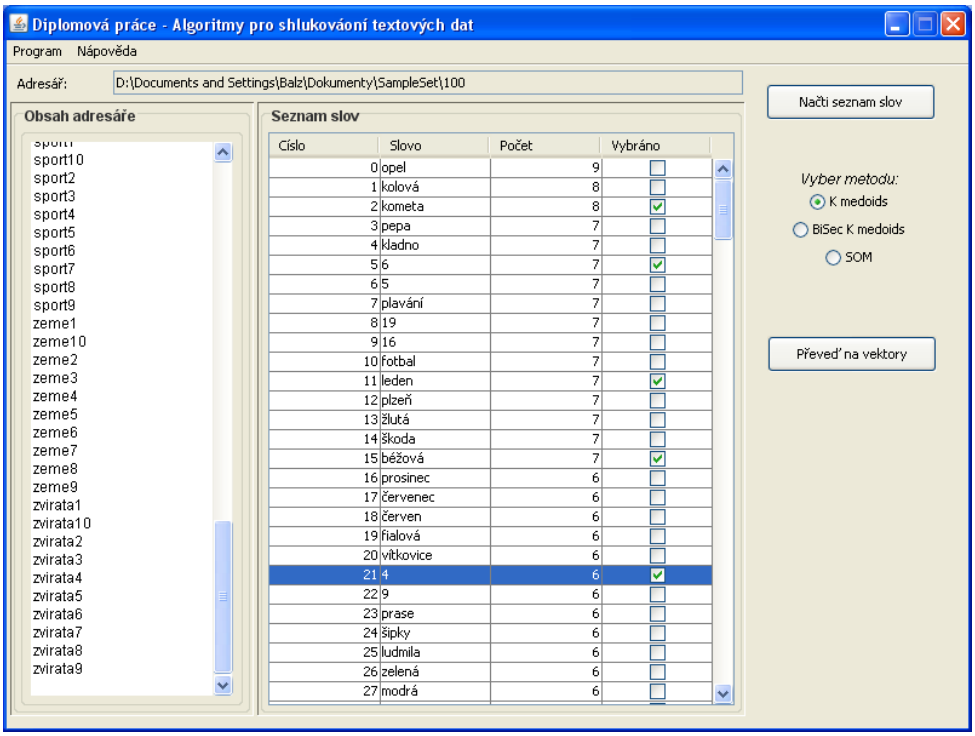
kterémkoli počítači, kde je nainstalováno prostředí pro běh aplikací imlpementovaných v jazyce Java. Soubor je přiložen na CD v diplomové práci. Po spuštění se zobrazí hlavní okno programu. Toto okno je zobrazeno na obrázku 3.6. Nyní si může uživatel například pomocí menu zobrazit, které algoritmy aplikace obsahuje či kdo aplikaci vytvořil. Může si zde také otevřít adresář s dokumenty, které chce shlukovat. Na obrázku 3.7 je zobrazeno, jak vypadá hlavní okno po otevření adresáře.

V horní části je zobrazena cesta k souborům, které chceme shlukovat, a v levé části jména souborů, které vybraný adresář obsahuje. Po otevření adresáře uživatel použije volbu "Načti seznam slov". Po této akci se načtou všechny soubory v adresáři a zpracuje se jejich obsah. Jednotlivá slova jsou pak uložena a v tabulce na pravé části okna vypsána, podle četnosti výskytů slov ve všech dokumentech. Zde si může uživatel také vybrat významová slova, podle kterých se bude provádět vektorizace a následné dělení do shluků. Jak tato část aplikace vypadá, je ukázáno na obrázku 3.8.

Po vybrání významových slov si uživatel vybere také algoritmus, který chce použít. Na výběr má ze tří, a to K medoids, BiSec K medoids a SOM. Po ukončení výběru uživatel použije volbu "Převeď na vektory". Po této akci se otevře další okno s algoritmem, který si uživatel zvolil. Okna jednotlivých algoritmů jsou ukázána na obrázcích 3.3, 3.4 a 3.5. Po zadání potřebných hodnot uživatel zmáčkne tlačítko "Vytvoř shluky". Po této akci se spustí algoritmus a vytvoří se požadovaný počet



Obr. 3.7: Hlavní grafické rozhraní aplikace po otevření adresáře s dokumenty



Obr. 3.8: Hlavní grafické rozhraní po načtení adresáře s dokumenty

shluků.

V textovém výstupu se pak vypíše průběh dělení a konečné shluky vytvořené algoritmem.

U algoritmu K medoids se vypíší vždy čtyři stavy shuků v průběhu dělení. Vyjímka může nastat tehdy, když se žádny ze souborů již nepřesunuje. To algoritmus zjistí a automaticky se zastaví. V této chvíli nemusí být všechny čtyři stavy shluků naplněny a vypíší se jen ty, které naplněny jsou. Stavy shuků se ukládájí vždy na začátku, v 1/3 počtu iterací, ve 2/3 počtu iterací a na konci.

U algoritmu BiSec K medoids se zase vypisuje průběh dělení algoritmu. V tomto případě se však dokumenty dělí pouze na dva shluky a ten který je větší, se dělí dále. Algoritmus se ukončí ve chvíli, kdy je dosaženo požadovaného počtu shluků.

U algoritmu SOM se vždy vypíší shluky podle velikosti mřížky, kterou zadal uživatel. Některé shluky mohou být prázné.

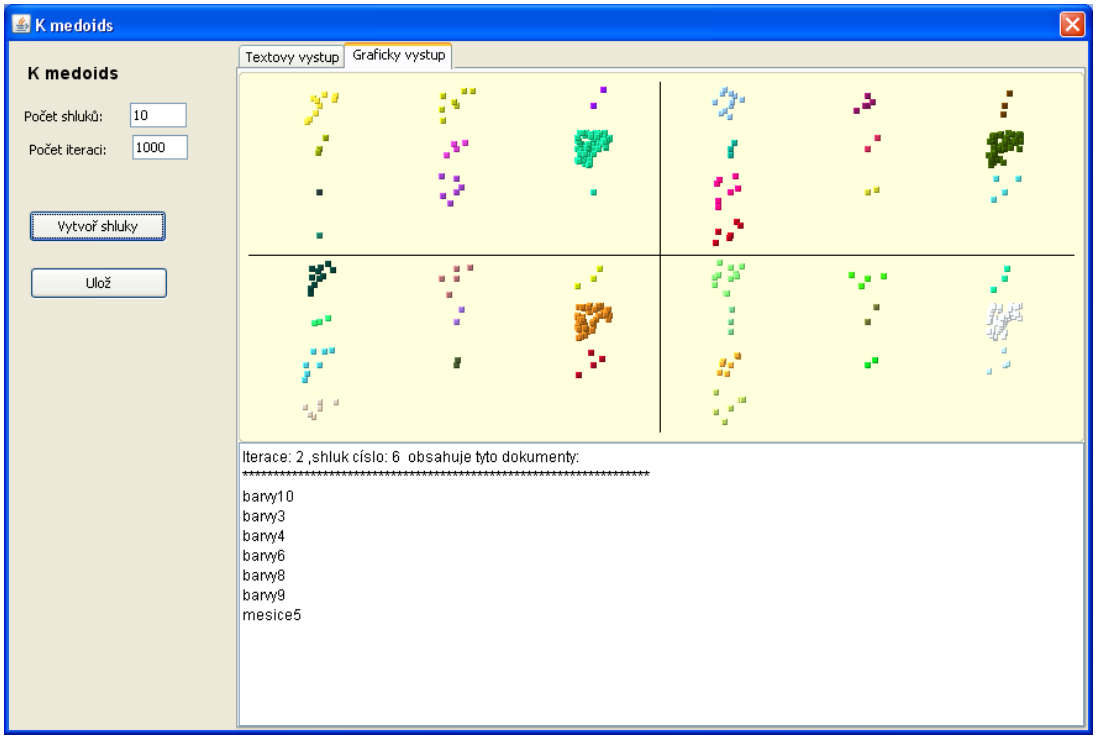
Uživatel si nyní může přečíst výsledky algoritmu, uložit tyto výsledky pomocí tlačítka "Ulož" a nebo zobrazit výsledky v grafické podobě. K tomu stačí kliknout myší na záložku "Grafický výstup". Grafický výstup se u jednotlivých algoritmů také liší.

U algoritmu K medoids se zobrazí shluky tvořené v průběhu dělení. Jednotlivé dokumenty ve shluku jsou zobrazeny tečkou stejné barvy. Po kliknutí na shluk se pak v dolní části okna vypíší dokumenty, které daný shluk obsahuje. Jak tento grafický výstup vypadá, znázorňuje obrázek 3.9.

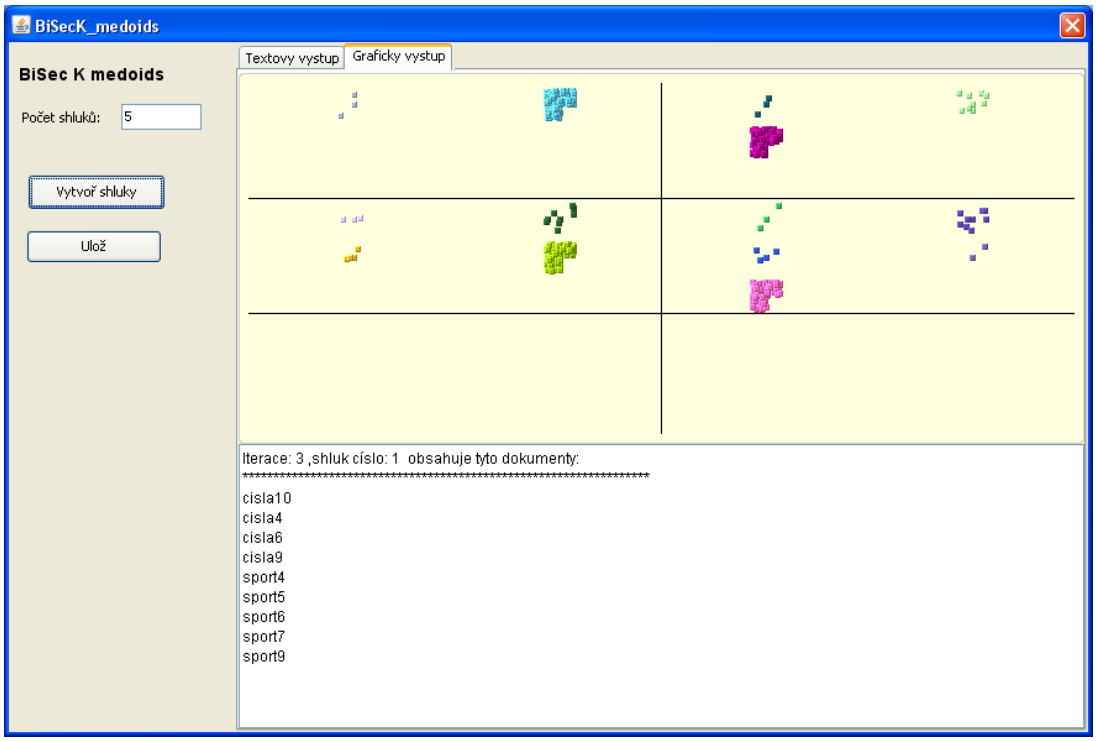
U algoritmu BiSec K medoids je v grafickém výstupu zobrazeno postupné dělení na jednotlivé shluky. Zobrazení je stejné jako u algoritmu K medoids. Opět po kliknutí na nekterý z vytvořených shluků se v dolní části vypíší dokumenty obsažené ve shluku. Ukázka je zobrazena na obrázku 3.10.

U algoritmu SOM je grafický výstup odlišný od předchozích dvou algoritmů. Jsou zde vytvořeny ovály, které reprezentují jednotlivé buňky neuronové sítě (naše shluky). Uvnitř shluků je číslo s počtem obsažených dokumentů. Po kliknutí na toto číslo se v dolní části okna opět vypíší dokumenty, které obsahuje daný shluk. Tento grafický výstup ukazuje obrázek 3.11.

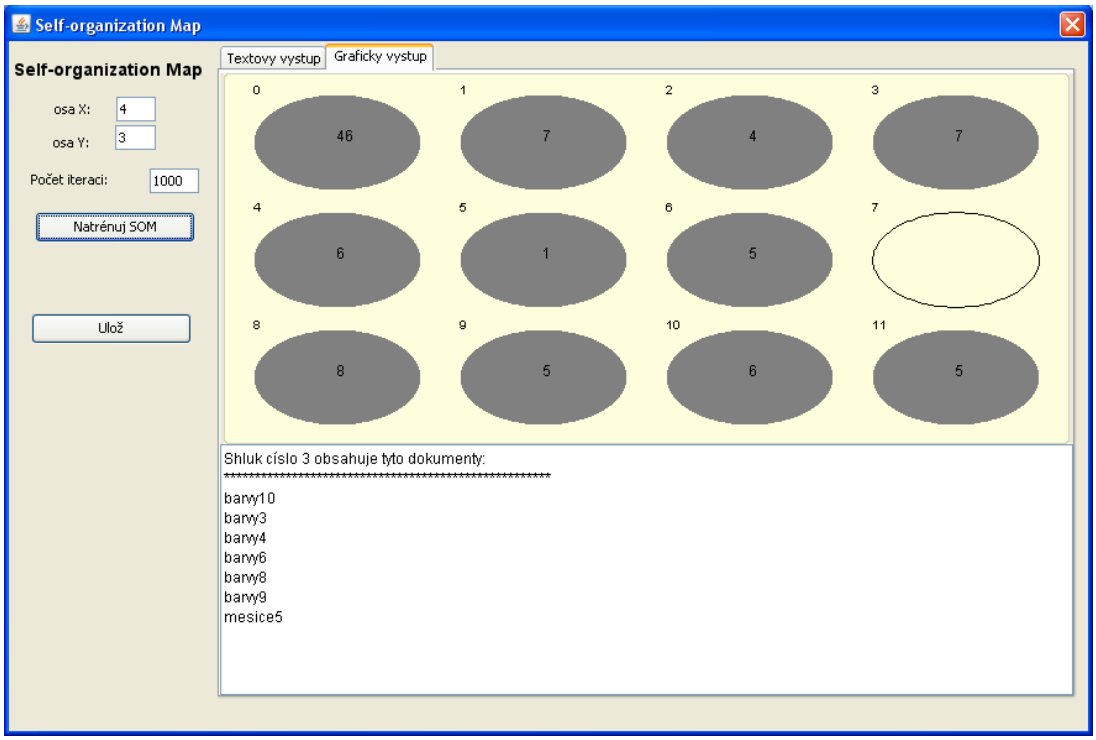
Po vytvoření shluků, prohlédnutí grafického výstupu a uložení výsledku si uživatel může jednotlivé algoritmy zavřít. Po této akci se zobrází opět hlavní okno aplikace. Uživatel zde může pokračovat v práci s aplikací nebo aplikaci ukončit. Zavřít aplikaci lze z menu nebo po kliknutí na křížek v pravé horní části okna.



Obr. 3.9: Grafický výstup metody K medoids



Obr. 3.10: Grafický výstup metody BiSec K medoids



Obr. 3.11: Grafický výstup metody SOM

3.9 Srovnání implementovaných algoritmů

Po implementaci shlukovacích algoritmů v této diplomové práci je také potřeba tyto algoritmy porovnat. Aby byly algoritmy objektivně porovnávány, byl vytvořen pojem asymptotická složitost algoritmu. Tuto složitost lze vyjádřit jako funkci vyjadřující počet kroků algoritmu (velikost paměti) v závislosti na velikosti vstupních dat. Krok algoritmu je operace, která se provede v konstantním (tj. na velikosti dat nezávislém) čase. Operací může být sčítání, odčítání, násobení, porovnání dvou hodnot či přiřazení. Složitost algoritmu se často používá jako kritérium pro porovnání kvality algoritmů a zjištění praktické použitelnosti.[27]

Složitost algoritmů rozdělujeme na dva typy, a to:

- Časovou - tato složitost odpovídá počtu vykonaných operací (resp. rychlosti výpočtu).
- Prostorovou - tato složitost odpovídá velikosti datových struktur využívaných algoritmem.

Obě tato kritéria si navzájem odporují, jelikož není možné najednou optimalizovat paměť i čas. Z těchto důvodů si musí programátor zvolit podle jaké složitosti bude algoritmus optimalizovat. V dnešní době se obvykle preferuje časová složitost, jelikož pamětové prostředky jsou už poměrně velké.

N	20	40	60	80	100	500	1000
n	$20\mu s$	$40\mu s$	$60\mu s$	$80\mu s$	$0.1ms$	$0.5ms$	$1ms$
$n \log n$	$86\mu s$	$0.2ms$	$0.35ms$	$0.5ms$	$0.7ms$	$4.5ms$	$10ms$
n^2	$0.4ms$	$1.6ms$	$3.6ms$	$6.4ms$	$10ms$	$0.25s$	$1s$
n^3	$8ms$	$64ms$	$0.22s$	$0.5s$	$1s$	$125s$	$17min$
2^n	$1s$	$11.7dn$	$36tis.let$				
$n!$	$77tis.let$						

Tab. 3.1: Obecná tabulka s druhy asymptotické časové náročnosti

Dalším důležitým parametrem při porovnávání algoritmů je také přesnost. Ta se určuje z mnoha testování algoritmů a z porovnávaní výsledků s výsledky předpokládanými.

3.9.1 Asymptotická časová složitost

Přesné vyjádření funkce pro časovou složitost je obtížné a většinou zbytečné. Proto se používá jen řádová rychlost růstu této funkce pro rostoucí N. Tomuto popisu se říká asymptotická časová složitost. U této složitosti zanedbáme pomaleji rostoucí členy a konstanty. Asymptotická složitost může být například $O(N^2)$. Tato složitost značí, že funkce $f(n)$ je asymptoticky menší nebo rovna funkci $g(n) = N^2$.[27]

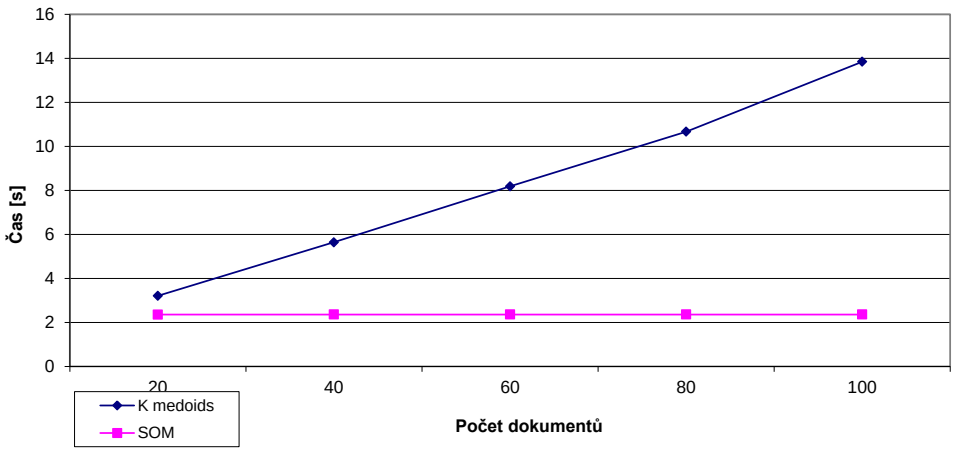
Cílem programátora je implementace algoritmů s co nejmenší asymptotickou časovou složitostí, tedy aby byl co nejrychlejší pro velká N. Pro malá N může být tato složitost větší v porovnání s jinými algoritmy, jelikož pro malá N je časová náročnost vždy dostatečně krátká. Tabulka 3.1 ukazuje dobu, po kterou trvá provádění operací $f(n)$ pro vstupní data velikosti N za předpokladu, že použitý hardware je schopen vykonat 1 milion operací za sekundu.[27]

Podle tabulky 3.1 bylo v této diplomové práci provedeno měření časové náročnosti implementovaných algoritmů pro různé počty dokumentů (N). U algoritmu K medoids byl nastaven počet shluků na 2 a počet iterací na 10 000. Vysoký počet iterací zaručí delší časovou náročnost, a tedy i přesnější výsledky. U algoritmu SOM pak byly nastaveny parametry rozměrů mřížky na 4x1 a počty iterací na 1 000. Toto nastavení dává optimální časový úsek pro měření. U algoritmu BiSec K medoids byl nastaven počet shluků na 10. Algoritmy SOM a K medoids jsou měřeny v sekundách a algoritmus BiSec K medoids v milisekundách,a to z důvodu jeho velice malé časové náročnosti. Celé měření je popsáno v tabulce 3.2.

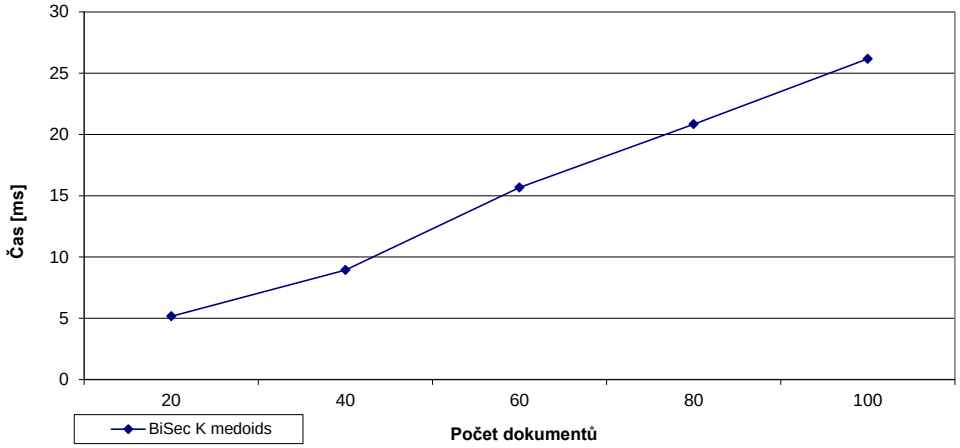
Z grafu 3.12 a 3.13 je patrné, že mají implementované algoritmy K medoids a BiSec K medoids časovou složitost lineární $O(n)$ nebo maximálně lineárně logarit-

N	20	40	60	80	100
K medoids	$3.2113s$	$5.6432s$	$8.1823s$	$10.6690s$	$13.8478s$
BiSec K medoids	$5.1667ms$	$8.9333ms$	$15.6667ms$	$20.8333ms$	$26.16667ms$
SOM	$2.3596s$	$2.3598s$	$2.3599s$	$2.3601s$	$2.3602s$

Tab. 3.2: Tabulka měření časové náročnosti implementovaných algoritmů zaměřující se na počet prvků N



Obr. 3.12: Graf časové náročnosti algoritmu K medoids a SOM na počtu dokumentů



Obr. 3.13: Graf časové náročnosti algoritmu BiSec K medoids na počtu dokumentů

x*y	4*1	4*2	4*3	4*4
SOM	2.3022s	4.5757s	7.0308s	9.3018s

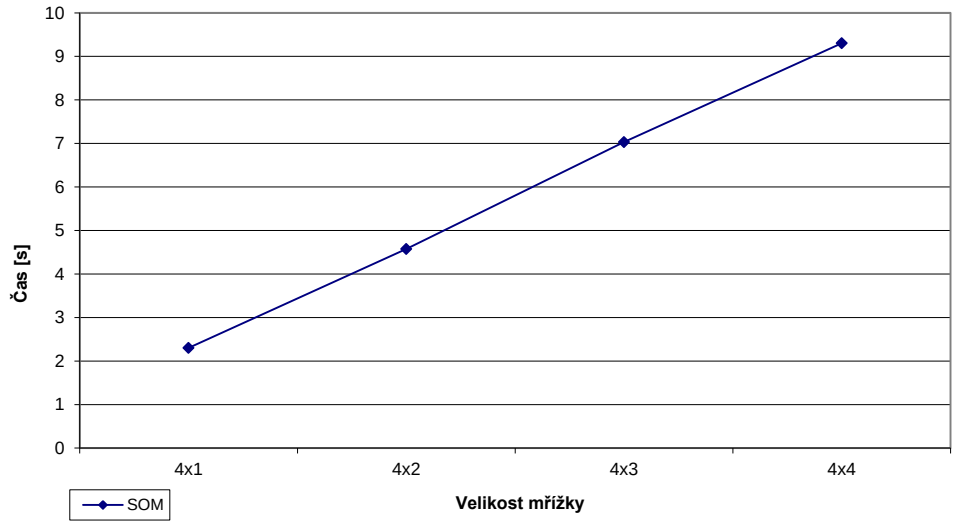
Tab. 3.3: Tabulka meření časové náročnosti algoritmů SOM zaměřující se na velikost mřížky

mickou $O(n \log n)$. Algoritmus SOM má časovou složitost konstantní $O(1)$ a jeho časová náročnost na počtu prvků je zanedbatelná. Všechna měření kromě tabulky 3.2 jsou prováděna pro celou validační množinu (100 dokumentů).

V tabulce 3.3 je provedeno měření časové náročnosti algoritmu SOM se zaměřením na velikos vstupní mřížky. U tohoto měření byl počet iterací nastaven na hodnotu 1 000.

V tabulce 3.4 je provedeno měření časové náročnosti algoritmů K medoids a SOM se zaměřením na počet iterací. U algoritmu K medoids byl parametr počet shluků nastaven na hodnotu 2 a u algoritmu SOM parametr velikosti mřížky na 4x1.

V tabulce 3.7 a 3.8 je provedeno měření časové náročnosti algoritmu K medoids a BiSec K medoids se zaměřením na počet požadovaných shluků. U těchto měření byl nastaven parametr počtu iterací na hodnotu 1 000.



Obr. 3.14: Graf časové náročnosti algoritmu SOM na velikosti vstupní mřížky

Počet iterací	1000	2000	3000	4000
K medoids	$1.4292s$	$2.8593s$	$4.2238s$	$5.6435s$
SOM	$2.3022s$	$4.627s$	$7.0128s$	$9.6027s$

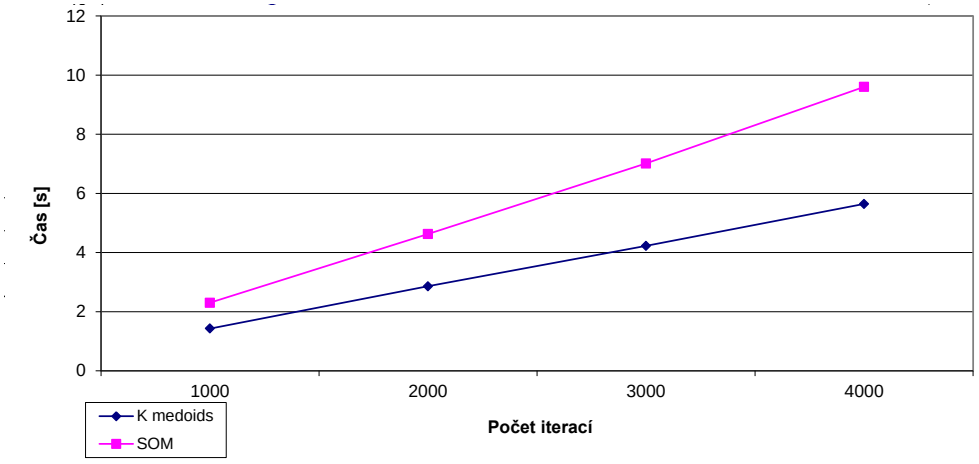
Tab. 3.4: Tabulka meření časové náročnosti algoritmů K medoids a SOM zaměřující se na počet iterací

Počet shluků	2	4	6	8
K medoids	$5.6435s$	$7.046s$	$8.3755s$	$9.7817s$

Tab. 3.5: Tabulka meření časové náročnosti algoritmu K medoids zaměřující se na počet shluků

Počet shluků	10	20	30	40
BiSec K medoids	$29.0125ms$	$57.5024ms$	$75.6667ms$	$101.6667ms$

Tab. 3.6: Tabulka meření časové náročnosti algoritmu BiSec K medoids zaměřující se na počet shluků



Obr. 3.15: Graf časové náročnosti algoritmu K medoids a SOM na počtu iterací

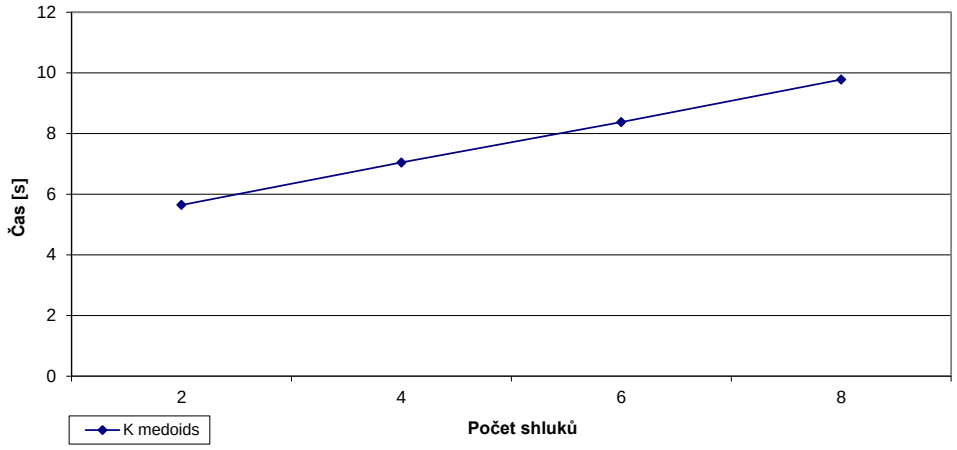
Z grafu 3.12, 3.14 a 3.15 lze zjistit, že implementovaný algoritmus SOM je lineárně časově závislý na velikosti mřížky a počtu iterací a jeho časová náročnost na počtu prvků je zanedbatelná. Z grafu 3.12 a 3.15 naopak vyplívá, že implementovaný algoritmus K medoids je lineárně závislí jak na počtu zpracovávaných prvků tak i na počtu iterací.

Z grafu 3.16 a 3.17 a patrné, že algoritmy K medoids a BiSec K medoids jsou závislé na počtu shluků, které vytváří.

Všechna měření byla prováděna 10x a výsledek byl následně zprůměrován a zaznamenán do tabulek.

3.9.2 Přesnost implementovaných algoritmů

Pro srovnání přesnosti algoritmů bylo provedeno mnoho testování implementovaných algoritmů. K testování byla vytvořena validační množina sta dokumentů, rozdělená do deseti tématických okruhů. Výsledky jednotlivých algoritmů jsou uloženy v textových dokumentech v adresáři s názvem „Testy“ na přiloženém cd. Adresář se dále dělí na dvě skupyny a to:



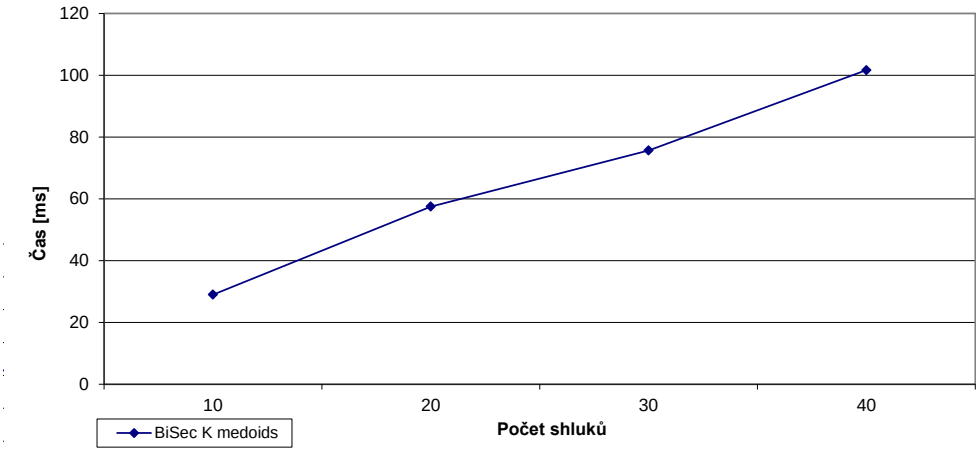
Obr. 3.16: Graf časové náročnosti algoritmu K medoids na počtu shluků

- Dělení do tématických okruhů: zde jsou dokumenty rovnoměrně děleny podle tématických okruhů.
- Vyhledávání jednoho tématické okruhu uvauta: zde jsou vyhledávány jen dokumenty patřící do tématického okruhu auta.

Dělení do tématických okruhů

V tomto adresáři jsou změřeny všechny tři implementované algoritmy. Významová slova, kéra byla u měření vybrána, byla ze všech deseti tématických okruhů. Z každého okruhu bylo vybráno pět slov s nejčastějším výskytem. U algoritmu K medoids a BiSec K medoids byla provedena měření pro 4,8,12 a 16 shluků. U algoritmu SOM pro velikost vstupní mřížky 4x1, 4x2, 4x3 a 4x4. V tabulce 3.7 je ukázka výsledku měření algoritmu SOM s velikostí vstupní mřížky 4x2. Tabulka obsahuje 8 shluků, jelikož má vstupní mřížka rozměr 4x2 (4x2=8). Každý ze shluků reprezentuje neuron v mřížce. Natrénovaným neuronům pak jsou přiřazovány dokumenty, které jim jsou podobné. Některé shluky obsahují dokumenty z více tématických okruhů, protože těchto okruhů je deset a vtupní mřížka obsahuje jen osum neuronů.

V tabulce 3.8 je pak ukázka měření pro algoritmus K medoids s počtem 8 shluků.



Obr. 3.17: Graf časové náročnosti algoritmu BiSec K medoids na počtu shluků

Shluk 0	Shluk 1	Shluk 2	Shluk 3	Shluk 4	Shluk 5	Shluk 6	Shluk 7
barvy10	barvy1	barvy7	auta1	hokej1	cisla1	hokej2	sport1
barvy3	barvy5	cisla5	auta10	hokej10	cisla2	hokej3	sport10
barvy4	jmena1	jmena10	auta2	hokej7	cisla3	hokej4	sport3
barvy6	jmena2	jmena6	auta3	prvky4	cisla4	hokej5	sport4
barvy8	jmena3	jmena7	auta4	prvky6	cisla6	hokej6	sport5
barvy9	jmena8	prvky1	auta5	prvky9	cisla7	hokej8	sport6
mesice2	jmena9	prvky5	auta6	zvirata1	cisla8	hokej9	sport8
mesice6	mesice10	prvky7	auta7	zvirata5	cisla9		
mesice7	mesice5	prvky8	auta8	zvirata7	jmena4		
mesice9	mesice8	sport7	auta9	zvirata8	jmena5		
zeme10		sport9	barvy2	zvirata9	mesice1		
zeme2		zeme1	cisla10		mesice3		
zeme3		zeme5	prvky10		mesice4		
zeme4		zeme7	prvky2		sport2		
.....		.....	.....				

Tab. 3.7: Tabulka s výsledky měření algoritmu SOM s velikostí vstupní mřížky 4x2

Shluk 0	Shluk 1	Shluk 2	Shluk 3	Shluk 4	Shluk 5	Shluk 6	Shluk 7
barvy1	auta7	hokej1	auta1	cisla1	auta2	barvy10	cisla2
barvy2	jmena10	hokej10	auta10	cisla10	prvky10	barvy3	cisla4
barvy5	jmena6	hokej2	auta3	cisla3	prvky2	barvy4	cisla6
jmena1	jmena7	hokej3	auta4	cisla5	prvky4	barvy6	cisla8
jmena2	prvky1	hokej4	auta5	cisla7	prvky6	barvy7	mesice1
jmena3	sport2	hokej5	auta6	hokej7	prvky7	barvy8	mesice10
	sport3	sport1	auta8	zvirata1	prvky9	barvy9	mesice3
	sport5	sport10	auta9	zvirata10	zeme5	hokej8	mesice4
		sport6	cisla9	zvirata3	zvirata2	hokej9	mesice6
		sport8	hokej3	zvirata5	zvirata4	mesice2	mesice8
			jmena4	zvirata6	mesice1	mesice5	mesice9
			jmena5	zvirata7	mesice3	mesice7	
			jmena8		mesice4	zeme10	
			jmena9		sport2	zeme2	
			.....			.....	

Tab. 3.8: Tabulka s výsledky měření pro algoritmus K medoids s počtem 8 shluků

Opět je zde tabulka dělena do osmi shluků. Validační množina však obsahuje dokumenty, které jsou děleny do deseti tematických okruhů. Z tohoto důvodu jsou v některých shlucích i dokumenty z více tematických okruhů. Druhým duvodem pro různé dokumenty v jednom shluku je nepřesnost algoritmu. Veškeré výsleky testování všech tří implementovaných algoritmů jsou obsaženy v adresáři „testy“ na přiloženém CD.

Vyhledávání jednoho tematického okruhu „auta“

U těchto měření byla vybrána jen významová slova z tematického okruhu „auta“. U algoritmu K medoids a BiSec K medoids je měření provedeno po 2,4,6 a 8 shluků. U algoritmu SOM pak pro vstupní mřížku o velikosti 4x1 a 4x2. V tabulce 3.9 je ukázka měření algoritmu BiSec K medoids s 8 shluky. Z této tabulky lze rozpoznat jak algoritmus BiSec K medoids dělil dokumenty do shluků. Výtezný dokument nultého shluku byl některý z dokumentů z tématického okruhu „auta“. K němu se hned přiřadily dokumenty stejného typu. Ve shluku jedna pak byl vítězný dokument „cisla4“. K tomuto dokumentu se žádný další dokument nepřiřadil, jelikož při výběru významových slov byly vybrána jen slova z tematického okruhu „auta“. V ostatních shlucích byl postup stejný. V sedmém shluku pak zůstaly všechny zbylé dokumenty a

Shluk 0	Shluk 1	Shluk 2	Shluk 3	Shluk 4	Shluk 5	Shluk 6	Shluk 7
auta1	cisla4	sport4	mesice7	zeme3	barvy1	cisla2	auta9
auta10							barvy10
auta2							barvy2
auta3							barvy3
auta4							barvy4
auta5							barvy5
auta6							barvy6
auta7							barvy7
auta8							barvy8
							barvy9
							. . .

Tab. 3.9: Tabulka s výsledky měření pro algoritmus BiSec K medoids s počtem 8 shluků

dělení se zastavilo, jelikož byl dosažen požadovaný počet shluků. U hledání jednoho tematického okruhu je ukázán jen výsledek algoritmu BiSec K medoids. Všechny ostatní výsledky jsou opět přiloženy na CD.

Po mnoha měřeních a testech uložených na přiloženém cd vyplývá, že z hlediska přesnosti výsledků je nejlepší algoritmus SOM. Pro větší přesnost tohoto algoritmu může uživatel zvětšit velikost mřížky nebo zvýšit počet iterací. To však způsobí také větší časovou náročnost. Nevýhodou tohoto algoritmu je jen to, že při zadaní velké vstupní mřížky uživatel neví, kolik shluků vlastně dostane. Výhodu pak je především přesnost a rychlost algoritmu při zpracovávání velkého množství dat.

Algoritmus K medoids je z hlediska přesnosti druhý ve srovnání s ostatními implementovanými algoritmy. U tohoto algoritmu velice záleží na počtu shluků, které zadá uživatel. Z tohoto důvodu se doporučuje provést více shlukování pro různé počty shluků a z výsledků si pak vybrat optimální řešení. Přesnost algoritmu se dá opět zvýšit pomocí počtu iterací, avšak i při velkém počtu iterací nedává tak přesné výsledky jako SOM. Tento algoritmus jde poměrně dobře využívat na menší kolekce dokumentů, při známem počtu tématických okruhů.

Algoritmus BiSec K medoids je nejméně přesný ze tří implementovaných algoritmů. Opět zde jako u algoritmu K medoids, záleží na počtu shluků, které zadá uživatel. Jinak se přesnost algoritmu nedá ovlivnit a záleží na výběru vítězných dokumentů, podle kterých pak probíhá dělení do shluků. Algoritmus je ovšem velice rychlí, a proto ho lze využít na základní dělení velkých kolekcí dat.

Přesnost všech tří algoritmů je také hodně ovlivněna výběrem významových

slov. Tato slova uživatel může volit dvěma způsoby. Jedním z nich je vybrat co nejvíce těchto slov z různých odvětví. Tím se dosáhne rozdělení do více shluků podle podobnosti. Druhým způsobem je vybrání jen těch významových slov, která mohou obsahovat pouze dokumenty hledané uživatelem. Každý z těchto tří algoritmů má své uplatnění a je jen na uživateli, jaký si vybere.

4 ZÁVĚR

V této diplomové práci je popsána teorie týkající se algoritmů na shlukování textových dokumentů. Teorie je zaměřena jak na druhy používaných algoritmů, tak i na postupy, podle kterých se zpracovávájí textové dokumenty. Součástí diplomové práce je též aplikace, která slouží ke shlukování textových dokumentů. V aplikaci jsou implementovány tři často používané aplgoritmy, a to K medoids, BiSec K medoids a SOM.

Aplikace je implementována v programovacím jazyce Java za pomocí vývojového prostředí Eclipse. K diplomové práci je také poskytnut samostatně spustitelný soubor xsedla61.jar, pomocí kterého lze aplikace spustit na jakémkoli počítači, kde je nainstalováno prostředí pro běh aplikací implementovaných v programovacím jazyce Java. Součástí diplomové práce je také vytvoření dokumentace pro programátora, který by chtěl aplikaci dále vyvíjet. Tato podrobná dokumentace je vytvořena pomocí dokumentačního nástroje JavaDoc. Pro aplikaci je vytvořena validační množina dokumentů, která sloužila k testování implementovaných algoritmů. Tato množina obsahuje 100 dokumentů rozdělených do 10 tématických skupin.

Aplikace v diplomové praci byla vytvořena pro projekt textového zpracování, kde na základě kumulace dat, kontroly pravopisu a překlepů a lemmatizace, bude možné jednotlivé dokumenty také třídit pomocí implementovaných shlukovacích algoritmů. K aplikaci je implementováno přívětivé grafické rozhraní, které mohou využít i běžní uživatelé zpracovávající textová data. Z praktické části diplomové práce lze jednoduchou úpravou vytvořit knihovnu použitelnou v jiných projektech.

Na konci diplomové práce jsou porovnány implementované algoritmy z hlediska časové náročnosti a přesnosti výsledků.

LITERATURA

[1] ANANIADOU, S. - MCNAUGHT, J.: *Text Mining for Biology and Biomedicine.* Artech House Publishers, 2006, 286 s., ISBN 1-58053-984-X

[2] RYDVAL, Slávek.: *Dokumentografické informační systémy. [online]* Softwarové noviny, 2002. Dokument dostupný na URL: `http://www.rydval.cz/phprs/view.php?cisloclanku=1980010101`

[3] MARKOV, Z. - LAROSE, D. T.: *Data-mining the Web: uncovering patterns in Web content, structure, and usage.* John Wiley Sons, NJ, 2007, 218 s., ISBN 978-0-471-66655-4.

[4] ZHU, X.: . *Advanced Natural Language Processing. [online]* CS769 Spring 2010. Dokument dostupný na URL: `http://pages.cs.wisc.edu/ jerryzhu/cs769/text_preprocessing.pdf`

[5] PORTER, Martin.: *The Porter Stemming Algorithm. [online]* 2006. Dokument dostupný na URL: `http://tartarus.org/ martin/PorterStemmer/`

[6] DORRE J. - GERSTL P. - SEIFFERT, R.: *Text Mining: Finding Nuggets in Mountains of Textual Data. [online]* 2003. Dokument dostupný na URL: `http://www.cs.uvn.ude/ xwu/kdd/TextMining-3.ppt`

[7] YANG, Y.: *An evaluation of statistical approaches to text categorization.* Journal of Information Retrieval, 1999, 294 s.

[8] ZENDULKA, Jaroslav. - BARTÍK, V. - LUKÁŠ, R. - RUDOLFOVÁ, I.: *Získávání znalostí z databází - Studijní opora.* Fakulta informačních technologií VUT v Brně, 2006.

[9] HAN, J. - KAMBER, M.: *Data Mining: Concepts and Techniques. second edition.* Morgan Kaufmann Publishers, 2006, 770 s., ISBN 1-55860-901-6.

[10] FELDMAN, R. - SANGER, J.: *The Text Mining Handbook.* New York, Cambridge University Press, 2007, 410 s.

[11] HAN J. - KAMBER M.: *Data Mining: Concepts and Techniques. Second Edition.* San Francisco, Elsevier Inc., 2006, 770 s., ISBN: 978-1-55860-901-3.

[12] VONDRÁK, Ivo.: *Umelá inteligence a neuronové síte.* VŠB - Technická univerzita Ostrava, 2009, 139 s., ISBN 80-2481-981-3

[13] BEIL F. - ESTER, M. - XU, X.: *Frequent Term-Based Text Clustering.* Canada, Proceedings of SIGKDD02, 2002.

[14] DRAŽIL, Jiří.: *Hitachi Content Archive platform. [online]* 2009. Dokument dostupný na URL: http:/www.storage.cz/index.php

[15] BISKUP, Roman. - ČERMÁKOVÁ, Anna.: *Genetické učení neuronových sítí.* Jihočeská univerzita v Českých Budějovicích, Zemědělská fakulta, Katedra aplikované matematiky a informatiky, 2004.

[16] SKOPAL, Tomáš.: *Neuronové sítě a Information Retrieval. [online]* VŠB-Technical University Ostrava, 2000. Dokument dostupný na URL: http://siret.ms.mff.cuni.cz/skopal/pub/neural.pdf

[17] ŠTEPÁNOVSKÝ, Jan.: *Úvod do neuronových sítí. [online]* Dokument dostupný na URL: http://www.lopikus.cz/temp/BMI-NAN.ppt

[18] DELISTE, Robert.: *Kohonen's Self Organizing Feature Maps. [online]* Dokument dostupný na URL: http://www.ai-junkie.com/ann/som/som1.html

[19] HANUS, Jan.: *Shluková analýza a její aplikace.* Západočeská univerzita v Plzni, Fakulta aplikovaných věd, 2009, 42 s.

[20] ŠARMANOVÁ, Jana.: *Metody dolování znalostí z dat. [online]* Dokument dostupný na URL: http://www.datakon.cz/datakon08/d02_sarmanova.pdf

[21] ZEMAN, David.: *Získávání znalostí z databází. [online]* Vysoké učení technické v Brně. Fakulta informatiky, 2009. Dokument dostupný na URL: http://www.datakon.cz/datakon08/d02_sarmanova.pdf

[22] KOPÁČKOVÁ, Hana. - MÁCHOVÁ, Renáta.: *Manažérské rozhodování za využití metod pro zpracování dokumentů. [online]* Ústav systémového inženýrství a informatiky, Fakulta ekonomicko-správní, UPA , 2006. Dokument dostupný na URL: http://hdl.handle.net/10195/35140

[23] ŘEZANKOVÁ, Hana.: *Klasifikace pomocí shlukové analýzy. [online]* Vysoká škola ekonomická v Praze, 2008. Dokument dostupný na URL: http://nb.vse.cz/ rezanka/shlukova_analyza2003.pdf

[24] MAŘÍK V. - ŠTĚPÁNKOVÁ O. - LAŽANSKÝ J. a kolektiv: *Umělá inteligence.* Academia, Praha 2003, ISBN 80-200-1044-0.

[25] HEBÁK, P. - HUSTOPECKÝ, J. - MALÁ, I.: *Vicerozmirné statistické metody 3.* Informatorium, Praha, 2005.

[26] SHEIKHOLESLAMI, G. - CHATTERJEE, S. - HANG, A.: *A multi-resolution clustering approach for very large spatial databases.* In Proceedings of the 24th Conference on VLDB, pages 428-439, 1998. New York, NY.

[27] KLEIN, Radek.: *Asymptotická časová složitost algoritmů. [online]* Dokument dostupný na URL: `http://radekklein.cz/asymptoticka-casova-slozitost-algoritmu/`

[28] HERMAN, B., ZIMMERMAN.: *Geoinformatika. [online]* 2001. Dokument dostupný na URL: `http://geologie.vsb.cz/geoinformatika/`

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

$D_m = (w_{m1}, w_{m2}, \ldots, w_{mn})$ obecná matice dokumentů

$w_{mn} \in < 0, 1 >$ váhy termů v dokumentu D_m

$w_n \in < 0, 1 >$ vyhledávané termy

TF_{ij} frekvence fermu j v dokumentu i

NTF_{ij} normalizovaná frekvence

$maxTF_{ij}$ max. frekvence termu

DF_j počet dokumentů s termem j

n_{ij} určuje počet výskytů ohodnocovaného slova v dokumentu

$\sum_k n_{kj}$ počet všech slov v daném dokumentu

$|D|$ celkový počet dokumentů v kolekci

$|d_j : t \in d_j|$ počet dokumentů v kterých se ohodnocované slovo vyskytuje

m_f střední hodnota

S_+ počet konkordantních srovnán

S_- počet diskordantních srovnání

(k) součet všech vnitroskupinových nepodobností při rozdělování objektů do k shluků

$x_1, \ldots x_n$ vstupy neuronu

x_0 formální vstup

$w_1, \ldots w_n$ váhy spojů

ε vnitřní potenciál

$y = f(\varepsilon)$ výstup neuronu

η parametr učení

η_0 počáteční rychlost učení

γ celkový počet iterací

$x_i(t)$ vstup uzlu i v čase t

τ_0 průměrnou velikost mřížky

$h(v,t)$ vliv vzdálenosti na učení

$\tau(t)$ okolí kolem vítěze

DIS dokumentografické informační systémy – Documentographic Information Systems

BPCL základní kombinovaný programovací jazyk – Basic Combined Programming Language

TF frekvence výrazu – Term Frequency

IDF inverzní frekvence výrazu – Inverse Document Frequency

HTML hypertextový značkovací jazyk – HyperText Markup Language

XML rozšiřitelný značkovací jazyk – Extensible Markup Language

PDDP Principal Direction Divisive Parttitioning

DENCLUE Density-based Clustering

DBSCAN Density-Based Spatial Clustering of Applications with Noise

FTC Frequent Term-based Clustering

HFTC Hierarchical Frequent Term-based Clustering

SOM samoorganizujicí mapa – Self Organizing Map

LSI latentně-sémantická indexase – Latent Semantic Indexase

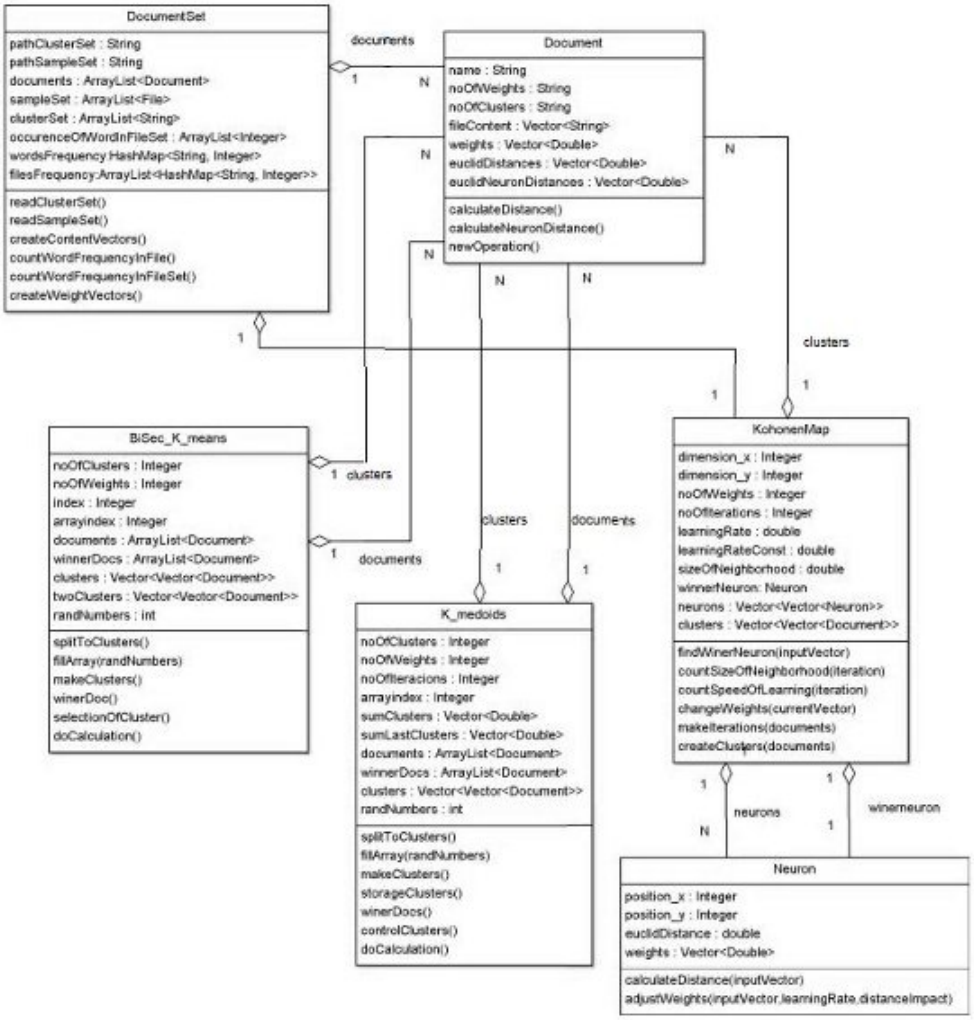
SVD singulární rozklad – Singular-value decomposition

SEZNAM PŘÍLOH

A OBSAH PŘILOŽENÉHO CD

Adresář	Popis
DP-xsedla61	Adresář obsahující aplikaci s implementovanými shlukovacími algoritmy.
DP-pdf	Adresář obsahující soubor s diplomovou prací xsedla61.pdf.
javaDoc	Dokumentace k aplikaci.
testy	Adresář obsahuje testy všech tří implementovaných algoritmů v aplikaci.
sampleSet	Validační množina dokumentů potřebná k testování aplikace.
jarSoubor	Samostatně spustitelný soubor s aplikací.

B DIAGRAM TŘÍD VYTVOŘENÉ APLIKACE



C STOP SLOVA PRO ČESKÝ JAZYK

a	aby	aj	ale	anebo	ani	aniz	ano	asi
az	ba	bez	bude	budes	by	byl	byla	byli
byt	ci	clanek	clanku	clanky	co	com	coz	cz
design	dnes	do	email	ho	i	jak	jake	jako
jeji	jeho	jej	jejich	jen	jeste	ji	jiz	jsem
jsi	jsme	jsou	jste	k	knam	kde	kdo	kdyz
ktere	kteri	kterou	ktery	ku	ma	mate	me	mezi
mit	mne	mne	mnou	muj	muze	my	na	nad
napiste	nas	nasi	ne	nebo	nebot	necht	nejsou	neni
nez	ni	nic	nove	novy	nybrz	o	od	ode
org	pak	po	pod	podle	pokud	pouze	prave	prave
pres	pri	pro	proc	proto	protoze	prvni	pta	re
se	si	sice	spol	strana	sve	svuj	svych	svym
ta	tak	take	takze	tamhle	tato	tedy	tema	te
tedy	tento	teto	tim	timto	tipy	to	tohle	toto
tom	tomto	tomuto	totiz	tu	tudiz	tuto	tvuj	ty
u	uz	v	vam	vas	vase	ve	vedle	vice
vsechen	vy	vzdyt	z	za	zde	zda	ze	zpet