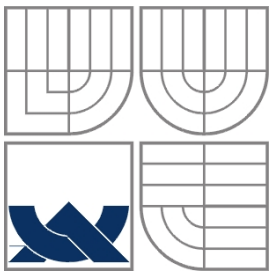
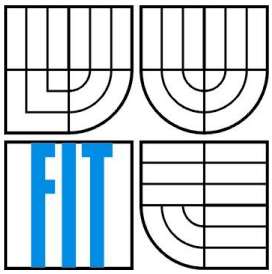


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# ZÍSKÁVÁNÍ ZNALOSTÍ NA WEBU - SHLUKOVÁNÍ

WEB MINING - CLUSTERING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Martin Rychnovský

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Vladimír Bartík, Ph.D.

BRNO 2008

# Abstrakt

Práce se zabývá problematikou získávání znalostí na webu. Cílem bylo prostudovat metody shlukovaní a realizovat shlukování pomocí algoritmu k-means. Potom algoritmus testovat na množině dokumentů a datech získaných z webu a následně vyhodnotit dosažené výsledky této metody. Shlukování bylo implementováno pomocí technologie Java.



# Klíčová slova

Získávání znalostí, dolování dat na webu, předzpracování, shluková analýza, výpočet vzdálenosti, k-means, centrální bod shluku.



# Abstract

This work presents the topic of data mining on the web. It is focused on clustering. The aim of this project was to study the field of clustering and to implement clustering through the k-means algorithm. Then, the algorithm was tested on a dataset of text documents and on data extracted from web. This clustering method was implemented by means of Java technologies.



# Keywords

Data Mining, web mining, preprocessing, cluster analysis, compute distance, k-means, cluster center.


# Citace

Rychnovský Martin: Získávání znalostí na webu - Shlukování. Brno, 2008, diplomová práce, FIT VUT v Brně.

# Získávání znalostí na webu - shlukování

## Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Vladimíra Bartíka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jméno Příjmení
Datum

## Poděkování

Děkuji vedoucímu své práce, Ing. Vladimíru Bartíkovi, za námět, směrování a cenné připomínky.

# Obsah

# 1 Úvod

Rychle se rozvíjející informační a komunikační technologie umožňují komunikovat a ukládat velké množství dat nejrůznějšího charakteru. Přesto často nejsou schopny splnit všechny požadavky, se kterými se na ně obracíme. Velké množství dat v sobě totiž nese potřebu vyhledat v nich jen to, co je pro nás nejdůležitější, tedy užitečnou informaci. Ta je často ve velkém objemu dat skryta. Proto obor, zvaný souhrnně získávání znalostí z dat, získává v posledních letech čím dál větší pozornost. Definici, historii a typické úlohy tohoto oboru shrnuje druhá kapitola.

Původně byla poptávka po přeměně rozsáhlých dat na znalost motivována zejména snahou managementu obchodních společností opřít svá strategická i operativní rozhodnutí o argumenty skryté v obrovském množství dat shromažďovaných po dlouhá léta v podnikových informačních systémech. V datech uchovávaných v podnikových databázích, které byly instalovány původně z důvodů vedení účetnictví, řízení skladového hospodářství a automatizace dalších ekonomických či technologických agend, jsou skutečně zaznamenávány projevy chování daného ekonomického, technologického nebo sociálního systému. Jejich analýza pak může napomoci přijetí důležitých strategických rozhodnutí.

Své opodstatnění našlo získávání znalostí z dat i v jiných oborech lidské činnosti, např. se může jednat o segmentaci a klasifikaci klientů banky (rozpoznávání problémových a rizikových klientů), predikci vývoje různých veličin, analýzu příčin poruch v telekomunikačních sítích, segmentaci a klasifikaci klientů pojišťovny, určení příčin poruch automobilů, rozbor databáze pacientů, analýzu nákupního košíku, aj..

Zajímavé využití získávání znalostí z dat skýtá i web. Drtivá většina volně dostupných informací je totiž rozptýlena po miliardách webových stránek, přičemž každá z nich má svou vlastní strukturu a formát. To vše dosti ztěžuje vyhledávání pro nás užitečných informací. Proto jsou pochopitelné snahy o ulehčení procesu vyhledávání pomocí metod zpracování obsahu stránek, získávání informací ze struktury stránek a analýzy chování uživatele. Informace o získávání znalostí na webu zahrnuje třetí kapitola.

Proces získávání znalostí není jednoduchou záležitostí. Důležitou částí tohoto procesu, mající významný vliv na výsledek celého procesu je předzpracování vstupních dat. Tomuto předzpracování se věnuje další kapitola. Všímá si především předzpracování textových a webových dat pro následné použití při shlukování.

Jedním z typů dolovacích úloh vhodných pro dolování na webu je shluková analýza, která rozděluje objekty na základě podobnosti do tříd, takzvaných shluků. Přehled shlukovacích metod uvádí pátá kapitola.

Kapitola šestá se zabývá problematikou reprezentace dokumentu pro dolovací úlohy a zjišťování míry podobnosti dvou dokumentů.

Pro realizaci shlukování by zvolen algoritmus k-means a proto je mu věnována další část, kde je popsán jeho princip, vlastnosti a možná rozšíření, které se snaží eliminovat hlavní nevýhody.

Další kapitola se věnuje návrhu a implementaci aplikace, využívající algoritmus z předchozí kapitoly. Jsou zde probrány všechny důležité třídy, popis nastavení a spuštění aplikace.

Devátá kapitola se věnuje zhodnocení výsledků a experimentů. Snaží se shrnout kvalitu výsledků nad různými daty.

# 2 Základní pojmy získávání znalostí

## 2.1 Definice získávání znalostí z dat

Tato kapitola čerpá z [1, 2, 3]. Do vědeckého povědomí se pojem „Získávání znalostí z databází“ dostal na přelomu 80. a 90. let minulého století a předcházel mu především technologický vývoj databázových technologií. V 60. letech minulého století, začaly vznikat datově intenzivní aplikace na podporu činnosti firem. Podpora pro práci se soubory pro tyto účely z řady důvodů nevyhovovala. Toto období je obdobím vzniku databázové technologie. Nejdříve to byly hierarchické a síťové databáze. V roce 1970 pracovník firmy IBM Corporation E.F. Codd publikoval základy teorie relačního modelu dat, na nichž byl vybudován dosud neúspěšnější typ databází – relační databáze a odpovídající systémy řízení báze dat. 80. léta minulého století znamenala konečné vítězství relačních systémů na trhu. To bylo podpořeno výrazným rozvojem metod a nástrojů na podporu návrhu relačních databází. Postupně se začaly objevovat pokročilejší modely dat, které se snažily rozšířit relační model dat nebo poskytnout specifický způsob práce s daty jako objektově orientovaný model dat nebo model pro deduktivní databáze. Další rozvoj databázové technologie v tomto období a s ním související výrazný nárůst objemu dat v databázích uložených byl počátkem tlaku na dostupnost technologií a nástrojů pro analýzy těchto dat.

Proto se v dalších letech objevují technologie a nástroje pro budování datových skladů. Ty vyplňují prostor, vedle produkčních databází, vzniklý z potřeby analyzovat data, které sice typicky poskytují jednoduchou a základní podporu pro operativní řízení, avšak jejich primární účel je jiný. A to zajištění databázových transakcí zvaných OLTP (Online Transaction Processing). Pro taktické či strategické rozhodování by bylo nepraktické a mnohdy i neúnosné provádět analýzy nad produkčními daty. Často by jsme mohli potřebovat i starší data, která naopak již v produkční databázi uchovávat nepotřebujeme. Datové sklady tak oddělují prostředí pro provádění analýz od produkční databáze a umožňují shromažďovat a současně připravit data získaná z různých datových zdrojů pro následnou analýzu. Model dat v datovém skladu pak lze definovat jako vícerozměrnou datovou kostku, nad kterou jsou definovány určité operace. Toto oddělení analýzy s odpovídajícími operacemi se označuje jako OLAP (Online Analytic Procesing). Kromě možnosti analýzy podporované přímo OLAP operacemi se v dalších letech jako reakce na potřebu analyzovat velké objemy dat objevuje právě získávání znalostí z databází. Narozdíl od OLAP operací, které používá uživatel interaktivně, jádrem procesu získávání znalostí je automatizované nalezení potenciálně užitečných modelů a vzorů v datech.

Podívejme se tedy na přesnější definici získávání znalostí z dat podle [2]. Můžeme jej definovat jako extrakci (neboli „dolování“) zajímavých (netriviálních, skrytých, dříve neznámých a potenciálně užitečných) modelů dat a vzorů z velkých objemů dat. Tyto modely a vzory reprezentují znalosti získané z dat. Zajímavost získaných znalostí charakterizují především netriviálnost, skrytost a potenciální užitečnost. Netriviálnost získané znalosti lze vnímat tak, že ji nejsme schopni získat nějakým SQL dotazem, ale použitím sofistikovanějšího postupu.Podobně skrytost říká, že musí jít o modely a vzory, které nejsou v datech na první pohled vidět, musíme je v datech nalézt a to netriviálním způsobem. Produkční databáze nebyla navrhována s ohledem na to, abychom takový typ informace ukládali. Potenciální užitečnost je vlastně význam získaných znalostí pro nějaké rozhodnutí. Může jít o různé rozhodnutí, například o udělení půjčky klientovi banky, analýzu nákupního košíku nebo analýzu příčin poruch v telekomunikačních sítích. Trochu odlišným případem je upozornění na podezřelé bankovní operace.Je zřejmé z definice potenciální užitečnosti, že pokud získáme jako výsledek získávání znalosti nějakou již známou informaci, potom se užitečnost takové znalosti blíží nule.

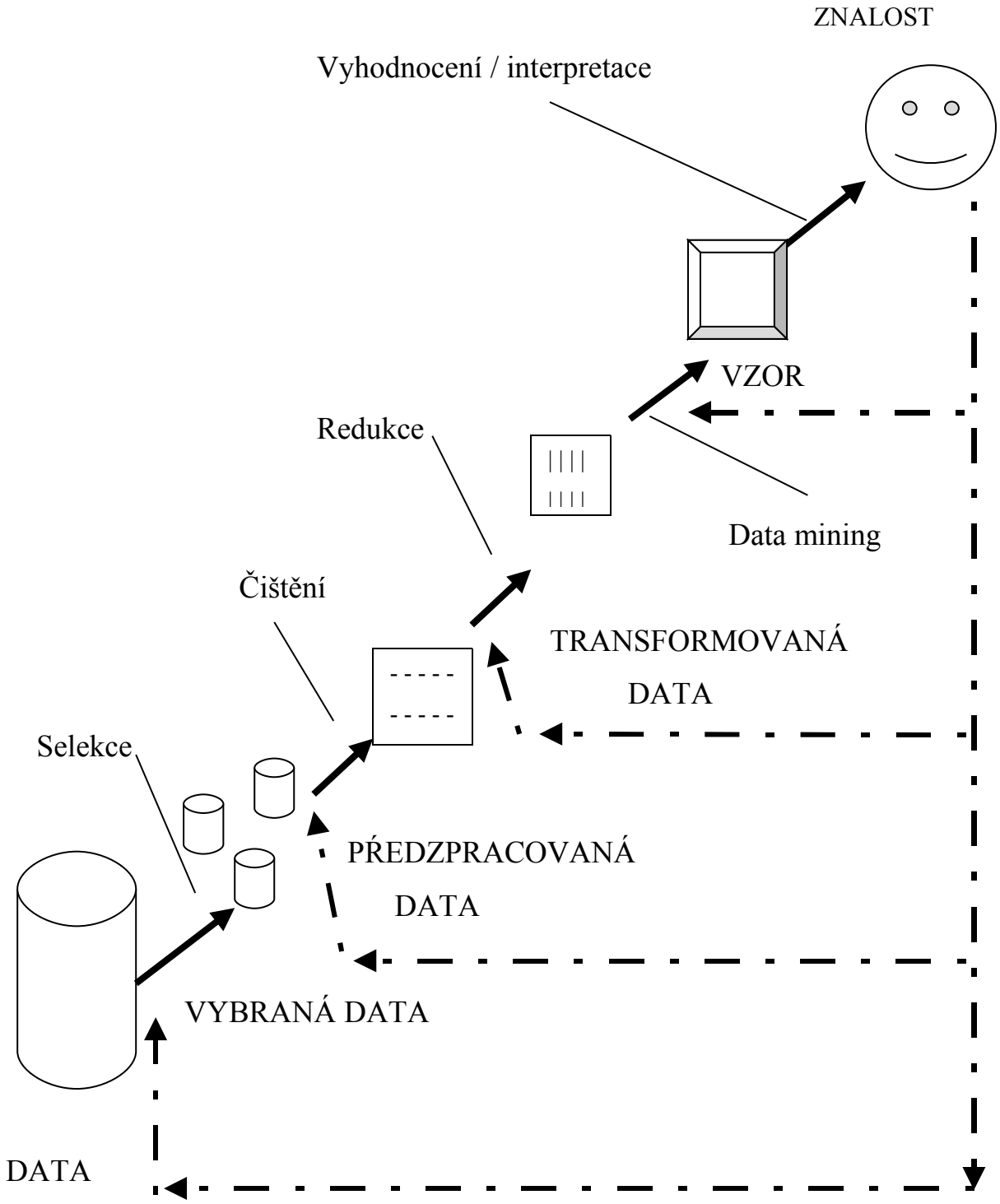
## 2.2 Proces získávání znalostí

Cílem procesu získávání znalostí je získat co nejvíce relevantních informací vhodných k řešení daného problému. Nyní se podíváme na získávání znalostí z databází jako na proces sestávající z řady kroků, které se zpravidla v určitých iteracích opakují podle [2]:
Jde o následující kroky:

1. **Čištění dat** – cílem je vypořádání se s chybějícími daty, odstranění šumu a vyřešení nekonzistence dat.
2. **Integrace dat** – cílem je integrace dat pocházejících z několika datových zdrojů. Často se integrace a čištění dat provádí společně. Jednak proto, že vyčištěná data potřebujeme někam ukládat, jednak proto, že jedním ze zdrojů nekonzistence jsou typicky data pocházející z více zdrojů. V takovém případě jsou data ukládána do datového skladu, jak ukazuje obrázek.
3. **Výběr dat** – cílem je vybrat data, která jsou pro řešení dané analytické úlohy relevantní. Pokud získáváme znalosti z dat uložených v relační databázi, pracujeme typicky s jednou tabulkou. V tomto kroku tedy vybereme z tabulky relevantní sloupce. V případě datového skladu můžeme analogicky vybírat dimenze.
4. **Transformace dat** – cílem je transformovat data do konsolidované podoby vhodné pro dolování. Může jít například o sumarizaci nebo agregaci. V případě použití datového skladu pro dolování může transformace předcházet výběru dat, protože může být součástí tvorby datového skladu.
5. **Dolování dat** – jádro procesu získávání znalostí, jehož cílem je aplikace určité metody a konkrétního algoritmu extrahovat z dat vzory, resp. vytvořit model dat.

6. **Hodnocení modelů a vzorů** – cílem je identifikovat skutečně zajímavé vzory pomocí mír užitečnosti.

7. **Prezentace znalostí** – cílem je prezentovat výsledky dolování uživateli využitím technik vizualizace a reprezentace znalostí.




**Obrázek 2.1** Proces získávání znalostí

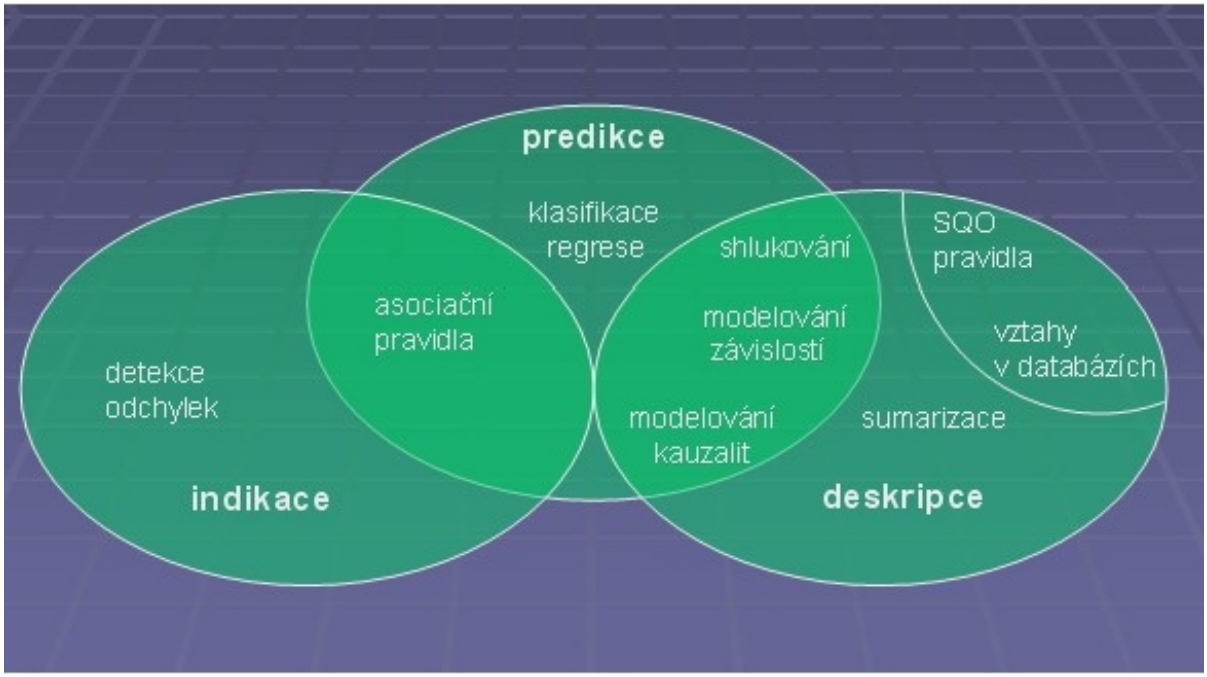
Kroky 1 až 4 výše uvedeného procesu jsou různou formou *předzpracování dat*. Protože reálné databáze či jiná data, které používáme jako datový zdroj, zpravidla nejsou v takovém stavu, abychom

jejich data mohli bezprostředně postoupit dolovacímu algoritmu k dolování. Obsahují velice často data, která jsou zašuměná, nekonzistentní a některé hodnoty mohou chybět. Taková data označujeme jako „nečistá“. Nízká kvalita vstupních dat by mohla vést k nepřesným nebo dokonce nesprávným a zavádějícím závěrům vyplývajícím z vydolovaných znalostí.

V kroku dolování může docházet k interakci s uživatelem nebo může algoritmus využívat nějakou znalost o datech, která může být uložena v databázi znalostí.

## 2.3 Dolovací úlohy

Řada metod používaných v problematice získávání znalostí vychází z umělé inteligence. Úlohy se podle [4] rozdělují na 3 typy: deskriptivní, prediktivní a indikační. Deskriptivní charakterizují a popisují data podle jejich vlastností uložených v databázi. Prediktivní pracují tak, že na základě dat v databázi jsou schopny předpovědět vlastnosti dat nových. Posledním typem jsou indikační a jejich cílem je rozpoznat neobvyklé vzory chování daného systému např. včasná identifikace poruchy technologického systému a vyslání alarmu obsluze. Avšak pro účely deskripce a predikce můžou být požity i metody shlukování, modelování závislostí a modelování kauzalit jak naznačuje obrázek 2.2.




**Obrázek 2.2** Rozdělení dolovacích úloh (převzato z [4])

### 2.3.1 Charakterizace a diskriminace dat

Podle [2] je jedním ze základních typů dolovacích úloh **popis konceptu/třídy** (concept/class desrciption). Data mohou být asociována s určitou třídou nebo konceptem. Takový popis můžeme získat jedním ze dvou následujících způsobů a to charakterizací dat a diskriminací dat.

Charakterizace dat provádí sumarizaci obecných vlastností analyzované třídy. Data, která odpovídají třídě, lze z databáze vybrat jednoduchým dotazem. Výsledkem může být například charakteristika určitého zboží a zjištění druhu zákazníka, který si jej kupuje.

Diskriminace naopak porovnává atributy cílové třídy s odpovídajícími atributy jiné třídy a hledá, v čem se nejvíce liší.

### 2.3.2 Frekventované vzory

Podle [2] plyne z názvu, že jde o vzory, které se vyskytují v datech často. Jedná se vlastně o nalezení vzorů, které se objevují v datech pravidelně. Úloha vede k odhalení zajímavých korelací nebo použití asociační analýzy a vyhledávání asociačních pravidel ve tvaru X => Y, kde X a Y jsou tvrzení týkající se hodnot atributů. Asociační pravidlo nám říká, jaká je pravděpodobnost výskytu prvků v transakci. Například následující asociační pravidlo:

LYŽE ∧ LYŽAŘSKOU OBUV => LYŽAŘSKÝ OBLEK

Nám říká, že zákazníci, kteří si koupí lyže a lyžařskou obuv si s velkou pravděpodobností koupí i lyžařský oblek. V praxi se nicméně nemusí jednat jenom o nákupy a košíky, ale lze dolovat souvislosti mezi událostmi, hodnotami v různých procesech a podobně.

### 2.3.3 Klasifikace a predikce

Jedná se o prediktivní dolovací úlohy. Cílem klasifikace je nalezení pravidel, která rozlišují a zároveň popisují třídy dat. Tato pravidla se pak použijí k predikci třídy objektu, jehož zařazení neznáme. Model je sestavován pomocí podmínkových pravidel, rozhodovacích stromů nebo jiných prostředků. Podle [2] se proces klasifikace se sestává ze tří kroků:

1. **Trénování** – na základě trénovací množiny je vytvořen model pro klasifikaci. Tato fáze se označuje také jako učení.
2. **Testování** – ověření kvality modelu testováním pomocí testovací množiny.
3. **Aplikace** – použití modelu ke klasifikaci dat, jejichž třídu neznáme.

Klasifikace se používá k predikci diskrétních tříd. Oproti tomu predikce předpovídá hodnoty spojitých atributů. V tomto případě předpovídáme numerickou nedostupnou hodnotu. Nejčastější metodou predikce je regresní analýza.

### 2.3.4 Analýza odlehlých objektů

Jde o nalezení objektů, které se nějakým způsobem významně odlišují od ostatních. Takové datové objekty se nazývají odlehlé (outlier). Tato analýza může například odhalit podvodné zneužití kreditních karet, extrémně velké nebo podezřelé nákupy.

### 2.3.5 Evoluční analýza

Jedná se o analýzu dat v čase, která hledá modely trendů a jejich vývoj. Zkoumá se vývoj a rychlost změn v čase anebo pravidelnost dat v jistých časových intervalech. Efektivní využití této metody může být při analýze průběhu hodnot akcií a rozhodování o investicích.

### 2.3.6 Shluková analýza

Podle [2] se dá říci, že shluková analýza (Cluster Analysis) na rozdíl od klasifikace a predikce analyzuje objekty bez znalosti přiřazení do tříd. A cílem je nalézt třídy objektů, které mají co nejvíce společného tak, aby se objekty různých tříd co nejvíce lišily. Nalezené třídy mají podobu tzv. shluků. Přesná definice dle [4] se s výše uvedeným ztotožňuje a říká nám, že shluková analýza je postup formulovaný jako procedura, pomocí níž objektivně seskupujeme jedince do skupin na základě jejich podobnosti a odlišnosti.(zkráceně R.C. Tryon, 1939). Více o vlastnostech a metodách shlukové analýzy se dozvíme v kapitole 4.

# 3 Dolování dat na webu

## 3.1 Úvod

Vyhledávání informací v prostředí webu se stalo rutinní součástí života. Nyní už můžeme jen vzpomínat na dřívější procházení stromů kategorií prvních předmětových katalogů nebo příchod prvních fulltextově orientovaných vyhledávácích strojů. Dnes nejznámější z nich Google, indexuje více než 8 miliard dokumentů. Spolu s rostoucím množstvím informací na webu se jejich sledování a využívání stává čím dál tím obtížnějším. Drtivá většina volně dostupných informací je rozptýlena po mnoha webových stránkách, přičemž každá z nich má svou vlastní strukturu a formát. Jak tedy vyhledat rychle a snadno požadované informace v užitečném formátu? Také proto se podle [6] dostávají úvahy a myšlenky, proč nevyzkoušet metody získávání znalostí. Proč třeba u vyhledávání nebrat v potaz také chování uživatele? A proto se dostávají obory jako text mining a web mining v této oblasti do popředí zájmu.

## 3.2 Co je web mining?

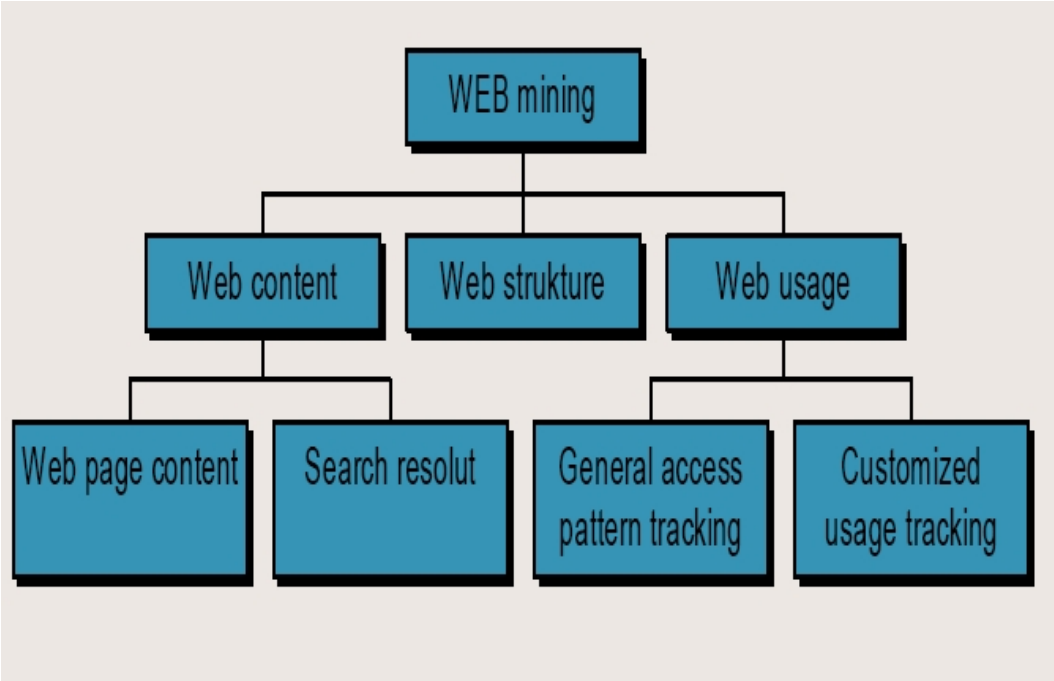
Jen obtížně lze polemizovat s tvrzením, že web je nejrozsáhlejším úložištěm informací, které kdy existovalo. Během pouhého desetiletí se z kuriozity stal nezbytný nástroj pro výzkum, marketing a komunikaci, jenž ovlivňuje každodenní život většiny obyvatel vyspělého světa. Co si tedy představit pod pojmem web mining? Web mining zahrnuje širokou oblast problémů primárně zaměřenou aplikaci technik data miningu k získání znalostí z internetových dat.

Definice podle [7] zmiňuje, že web mining (nebo také web farming, web harvesting či web scraping) označuje proces shromažďování a uspořádávání nestrukturovaných informací z webových stránek a obecně dat obsažených na webu. Z pohledu dat pro dolování lze podle [2] chápat web jako velkou heterogenní databázi s velkým potenciálem pro dolování.

## 3.3 Oblasti zájmu web miningu

Podle [7] se web mining zabývá třemi širokými oblastmi:

- Dolování **obsahu internetu** (Web Content mining) – zpracování obsahu stránek
- Dolování **struktury internetu** (Web Structure mining) – získávání informací ze struktury stránek
- Dolování **o používání internetu** (Web usage mining) - analýza chování uživatele




**Obrázek 3.1** Rozdělení Web miningu (převzato z [7])

### 3.3.1 Web Content mining

Web Content mining je aplikace data miningových technik nad obsahem stránek zveřejněných na internetu, obvykle jako HTML (polostrukturované), holé (nestrukturované) nebo XML (strukturované) dokumenty. Jde tedy o analýzu textové složky (obsah a meta popis) stránek zaměřenou na detekci sémanticky významných termů a jejich další užití. Často založený na vektorovém modelu dokumentu.

### 3.3.2 Web Structure mining

Web Structure mining operuje nad strukturou internetových odkazů. Tato grafová struktura může poskytnout informace o ohodnocení nebo autoritě stránky a vylepšit výsledky vyhledávaní pomocí filtrů. Často analýza vzájemného propojení WWW stránek vede na transformaci webového prostoru do orientovaného grafu.

### 3.3.3 Web Usage mining

Web usage mining analyzuje výsledky interakce uživatele s web serverem, včetně web logů, sekvence klikání a databázových transakcí na serveru nebo odhaluje zájmové komunity. Pro analýzu a detekci vzorů v datech generovaných v průběhu spojení mezi klientem a WWW serverem se užívá shlukové analýzy nebo asociačních pravidel. S těmi jsme se již seznámili v kapitole 2.3.

Například 68% uživatelů, kteří navštívili URL A, navštíví také URL B. Za použití shlukové analýzy můžeme třeba seskupovat uživatele s podobnými vzory chování či stránky navštěvovaných stejnou skupinou.

# 4 Předzpracování

## 4.1 Proč předzpracování dat?

Hlavní důvod spočívá v rozličnosti zpracovávaných dat. Mohou být získány od lidí, senzorů, jiným předchozím výpočtem apod. Avšak ani jeden z těchto zdrojů není v reálném světě úplně přesný. Tento nepříjemný fakt způsobuje, že data zpravidla nejsou v takovém stavu, abychom je mohli bezprostředně postoupit dolovacímu algoritmu. V případě, že tyto chyby a nepřesnosti nějakým způsobem neeliminujeme, můžeme dostat velmi rozdílné výsledky než v případě předzpracovaných dat. Dalším dobrým důvodem může být například i to, že data nejsou v takové podobě, kterou dolovací systém vyžaduje, proto se je snažíme určitým způsobem upravit a přizpůsobit. Výhodná může být i úprava dat, ve které data redukujeme. Tím můžeme výrazně urychlit dolování z takovýchto dat. Redukce samozřejmě musí zachovat charakter původních neredukovaných dat.

Problémem ale může být, jak poznat, že data potřebují pro dobré výsledky dolovací úlohy předzpracování. Dle [2] se dají projevy nekvality dat rozdělit na tyto skupiny:

*Data jsou nekompletní* – Chybějí některé atributy či jejich hodnoty. Problémem může být i celkově malé množství dat.

*Data jsou zašuměná* – Nekvalita se projevuje nesprávnými nebo odlehlými hodnotami, nekompatibilními daty nebo i různou úrovní dat.

*Data jsou nekonzistentní* – Zde nám dolování stěžují nepodstatná data vzniklá redundancí dat, případně nekonzistence v pojmenování, kódování, formátech apod.

## 4.2 Předzpracování webových a textových dat

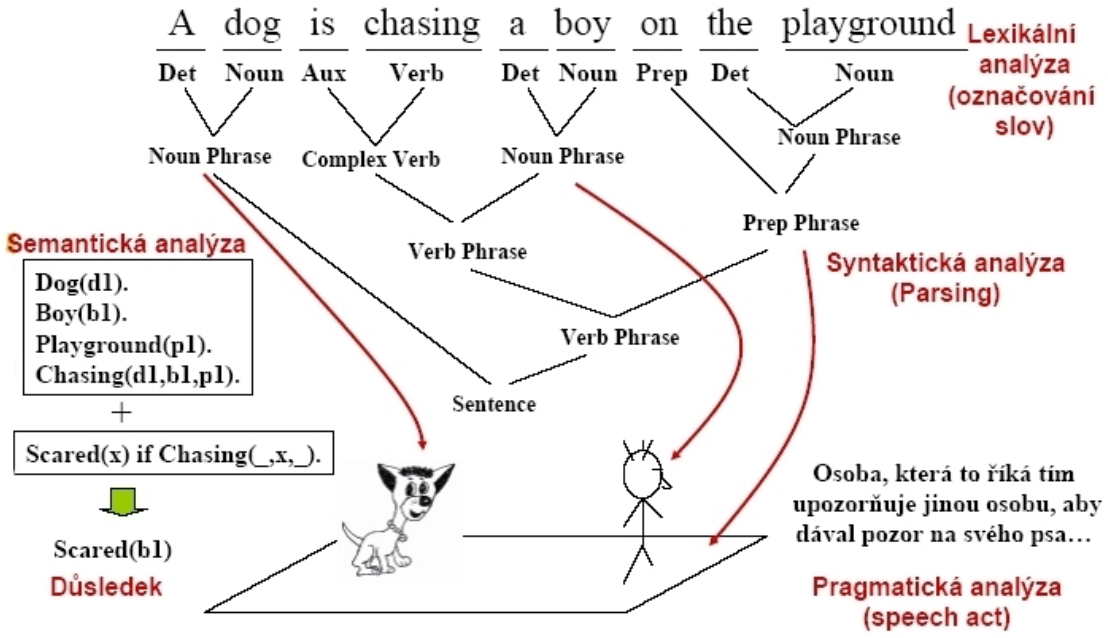
### 4.2.1 Problémy textových dat

Metody získávání znalostí z textu pracují s daty, které mají díky své struktuře zcela odlišné vlastnosti než jiná data používaná v data miningu. Musíme se často vypořádat s velkými soubory dat, častými změnami dat, velkým šumem a jinými problémy. Texty často nemají ani přesnou a jasnou strukturu. Největší komplikaci, ale přináší počítačové zpracování textu. Počítač je sice schopný zpracovávat lidmi napsaný text, ale pouze jako nic neříkající posloupnost znaků. Výsledkem toho všeho je úplně odlišný přístup dolování v textech než dolování v ostatních datech. Především využitím technik vědního oboru zabývajícího se zpracování přirozeného jazyka.

### 4.2.2 Zpracování přirozeného jazyka

Zpracování přirozeného jazyka (Natural language processing – NLP) je efektivním nástrojem pro předzpracování dat při dolování v textu. Tato disciplína zkoumá problémy analýzy či generování textů nebo mluveného slova, které vyžadují určitou (ne absolutní) míru porozumění přirozenému jazyku strojem. Text lze analyzovat podle [9] ve čtyřech základních rovinách, které na sebe postupně navazují:

1. morfologie
2. syntaxe
3. sémantika
4. pragmatika




**Obrázek 4.1** Zpracování přirozeného jazyka (převzato z [9])

Výstup z nižší roviny je vstup do následující vyšší roviny. Na úrovni morfologie se zajímáme o ohýbání a odvozování slov pomocí předpon a přípon (též tvarosloví). Třeba ze slova „*dívkami*“ nám vznikne term „*dívka*“. Rovina syntaktická zkoumá větnou strukturu. Syntaktické kategorie jsou podmět, přísudek, předmět apod. Sémantická rovina se snaží zachytit význam věty v závislosti na její syntaktické struktuře. Například vztah mezi povrchovými kategoriemi jako „podmět“, „předmět“ a hloubkovými kategoriemi jako „konatel“, „trpitel“. Na pragmatické úrovni, které se někdy říká logická či textová, probíhá přiřazení objektů reálného světa uzlům větné struktury. Pro předzpracování textových dat jsou využívány nejčastěji tyto techniky: Lemmatizace a derivace, Morfologická desambiguace, Syntaktická analýza (Případně parciální syntaktická analýza).

### 4.2.3 Lemmatizace a derivace

Lemmatizace vytvoří k určitému tvaru slova základní tvar, tzv. lemma. Toto se většinou děje na morfologické úrovni a vykonávají ji lemmatizátory. Většinou jsou ještě schopny určit slovní druh a další gramatické kategorie spojené s tvarem slova (osoba, číslo, pád apod.). S případnou mnohoznačností se většinou vypořádávají přiřazením všech tvarů. Takže při lemmatizaci výrazu „vesel“ si slovo převede na oba základní tvary „veslo“ i „veselý“, aniž by zkoumal, o který případ se konkrétně jedná.

### 4.2.4 Morfologická desambiguace

Morfologická analýza značně ulehčuje analýzu na vyšších úrovních. Problémem ale je, že nebere v potaz textový kontext analyzovaného slova, dokonce i více slovné pojmy jsou vyhodnoceny jednotlivě.

Třeba čeština je syntetický jazyk a k vyjádření určitého jazykového jevu zpravidla používá vhodný tvar slova narozdíl od jazyků analytických, které ke stejnému účelu využívají pomocná slova. Takovýmto jazykem je například angličtina. Výsledkem tohoto faktu je podle [24], že mnoho slov má více možných interpretací a tak morfologická analýza poskytuje nejednoznačný výsledek. Několik možných gramatických interpretací a někdy i několik možných základních tvarů. Příklad nejednoznačnosti na úrovni lemmatu najdeme ve větách „Jez do polosyta.“ a „Berounský jez je opravdu velký.“ Kde základní tvar slova „jez“ je v prvním případě „jíst“ a v druhém „jez“. Desambiguace může vznikat i na jiných úrovních v závislosti na jazyku. Například nejednoznačné tvary, kdy nevíme o jaký jde slovní druh apod. Zajímavá je i desambiguace spojovacích výrazů. Třeba ve větách: „Celou sobotu a neděli budeme doma. Budeme doma celou sobotu a neděli strávíme na chatě.“, v jednom případě budeme oba dny doma a v druhém budeme doma jen v sobotu.

Určit správnou gramatickou interpretaci slova v daném kontextu je pro počítač velmi složitý problém, vyžadující inteligenci blízkou člověku. Morfologická desambiguace se zaobírá právě tímto problémem.

### 4.2.5 Syntaktická analýza

Úkolem syntaktické analýzy je rozpoznat, zda vstupní textový řetězec je větou v daném přirozeném jazyce. V kladném případě je výsledkem analýzy syntaktická struktura věty, například v podobě derivačního stromu. Cílem syntaktické analýzy je, aby počítač, například na základě gramatických pravidel, „porozuměl“ vztahům mezi jednotlivými slovy (a nepřímo tedy i mezi zmiňovanými lidmi, věcmi a činnostmi). Podle [24] se v syntaktické rovině hledají a formulují pravidla, podle kterých se dají kombinovat slova do skupin a skupiny do větných vztahů. Skupiny (složky) jsou řetězce slov patřící k sobě podle řídícího slova, které není možné vynechat. Z toho plyne, že celá skupina se dá redukovat právě na toto slovo. Analyzovat strukturu věty z tohoto hlediska znamená najít ve větě

skladební dvojice, tedy dvojice řídící prvek a závislý prvek. Problémem při tomto zpracování je mnohoznačnost, kterou se přirozené jazyky vyznačují. Tu lze ukázat například na větě „Sledoval ho s čepicí na hlavě“, kde přesně nevíme, zda jej sledoval s čepicí na hlavě či sledoval někoho, kdo měl čepici na hlavě.

#### 4.2.5.1 Parciální syntaktická analýza

Cílem standardních syntaktických analyzátorů je provést pokud možno co nejpřesnější a nejúplnější analýzu vstupní věty. Předpokládají, že gramatika pokrývá celý zpracovávaný jazyk a potom analyzátor vyhledá nejlepší analýzu. Takovýto přístup ale nelze použít pro texty obsahující chyby. Avšak texty, které jsou určené k automatickému zpracování, bývají málokdy bezchybné. Řešením tohoto problému je parciální syntaktická analýza. Ta se podle [25] snaží získat syntaktické informace z libovolného a potencionálně chybného textu na úkor úplnosti a hloubky analýzy. Hlavní myšlenkou parciální syntaktické analýzy je nalezení větných skupin, které vyžaduje pro minimální rozpoznání syntaktické znalosti a stačí k nim poměrně jednoduchá gramatika. Zpravidla jde o vyhledání nerekurzivních jader substantivních (podstatných) skupin, respektive vyhledání nerekurzivních jader všech významných větných skupin.

### 4.2.6 Filtrace

Filtrace může hrát důležitou roli i při předzpracování textových či webových dat. Většinou je totiž dat až příliš. Většinou se v této oblasti zabýváme filtrací termů. Používá se několik přístupů, avšak základem všech je redukce množství termů výběrem jen několika nejfrekventovanějších nebo zahození termů, vyskytujících se ve všech dokumentech s konstantním rozložením.

### 4.2.7 Korpus

V počítačové lingvistice je jazykový korpus většinou rozsáhlý soubor textů, které jsou v různé míře opatřeny metajazykovými značkami vypovídajícími o samotném textu (autor, rok vydání, žánr apod.), zařazení jednotlivých slov do kategorie slovních druhů, o frekvenci slova v korpusu, případně dalších lingvistických a frekvenčních aspektech. Korpusy však mohou být i bez těchto metainformací. Speciální programy, tzv. korpusový manažeři, umožňují vyhledávání slov a slovních spojení v kontextu, zjištění frekvence výskytu v korpusu i zjištění původního zdroje textu. Pro formátování textů a vkládání značek se používá zejména standardizovaného jazyka XML, případně staršího SGML.

### 4.2.8 Thesauri, Soundex

Princip tezaurus se používá především pokud nám jde o vyhledání. Tezaurus nám umožňuje rozšíření původního dotazu pomocí podobných nebo souvisejících termínů. Dle [26] se dá říci, že jde o slovník, který umožňuje nabízet shodný nebo podobný seznam slov. To zajišťuje shodné „vnímání“ určitého tématu popsaného textem do jazyka systému. Vyjadřuje pojmy, které jsou v přirozeném jazyce těžko postižitelné a pomocí složených termínů a dalších nástrojů překonává problémy týkající se umělého jazyka. Při předzpracování dat pro dolovací úlohy webu či textu se příliš nevyužívá, jelikož lokální kontext dokumentu nemusí odpovídat názoru stroje.

Soundex transformuje termy na „zvukovou základnu“. Většinou se tento převod realizuje pomocí heuristiky převádějící slovo na fonetický ekvivalent. To znamená, že slova vyslovovaná podobně vnímáme stejně. Například slovo „Euler“ a slovo „Ellery“ může mít stejný soundex.

### 4.2.9 Stemming

Stemming můžeme vnímat jako metodu, která ke každému slovu umožňuje určit jeho kořen. Přesněji můžeme pomocí [12] definovat stemming jako proces redukování skloňovaných, časovaných nebo odvozených slov na jejich kořen případně slovní základ. Stem (slovní základ) nemusí být shodný s morfologickým kořenem slova, obyčejně postačuje, když související slova jsou reprezentovány stejným základem, i když není právoplatným kořenem.

Algoritmus stemmeingu je dlouhodobým problémem v počítačové vědě, první dokument na téma stemming byl publikovaný v roce 1968 a sepsal jej Julie Beth Lovins. Proces stemmingu je hojně využívaný ve vyhledávacích algoritmech, rozšiřování dotazů, indexování a ostatních problémech zpracování přirozeného jazyka. Stemmovací algoritmy, případně programy se někdy nazývají jako stemmery.

Stemmer například pro angličtinu by měl řetězce „cats“, „catlike“, „catty“ apod. reprezentovat základem, chcete-li kořenem v podobě řetězce „cat“ a například slovo „stemmer“, „stemming“, „stemmed“ jako slova odvozená od základu „stem“. Existuje celá řada stemmovacích algoritmů. Od jednoduchých jako je S-stemmer, ve kterém je odstraněno jen několik běžných ukončení slov:„ies“, „es“ „s“ (s několika málo výjimkami). Jeho výsledky však jsou jednoduché a nedostačující. Dalším [12] stemmovacím algoritmem je Lovinův. Tento algoritmus je pouze jednoprůchodový a z toho plyne jeho citlivost na kontext. I proto je dosti nespolehlivý a často chybuje. Jeho vylepšením je algoritmus Dawson, který opravuje chybující transformace předchozího algoritmu.

Stemmer, který se stal prakticky standardním algoritmem na stemming anglického jazyka, vymyslel a publikoval v roce 1980 Martin Porter. Tento algoritmus v porovnání s ostatními vykazuje nejlepší výsledky co se týká úspěšnosti 97% a 90% pokrytí spojení. Proto je v aplikaci použit právě Porterův stemmovací algoritmus.

## 4.3 Proces předzpracování v aplikaci

Abychom mohli provést shlukovou analýzu nad texty, potřebujeme dokumenty upravit do vhodné podoby. Nejdříve je předzpracovat a následně reprezentovat pomocí vhodného modelu. Proces předzpracování v aplikaci je zaměřen na kolekci dat Reuters-21578. Nejdříve vytvoříme soubor obsahující termy jednotlivých dokumentů. Tento soubor má na začátku znak „D“, následovaný číslem dokumentu (např. D51). Za tímto označením už následují termy dokumentu. Tyto termy již prošly procesem předzpracování. Výstupem je i soubor se stejným značením dokumentů ("D číslo" ) obsahující termy z titulku jednotlivých dokumentů. Nad těmito soubory je pak vykonána shluková analýza. V případě použití vstupních dat z databáze, předzpracování v tradičním smyslu dolovacích úloh provádět nemusíme.

### 4.3.1 Kolekce dat Reuters-21578

Jedná se o novinové články agentury Reuters publikované v roce 1987. V roce 1990 byla celá sada uvolněna pro vědecké účely univerzitě v Massachusetts. Zde byly dokumenty převedeny Davidem D. Lewisem a Stephenem Hardingem do formátu SGML. Kolekce se skládá z dvaadvaceti souborů a v každém se nachází tisíc dokumentů.

Přesná specifikace formátu SGML DTD je součástí balíčku těchto dokumentů. Dostupné jsou na URL http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html (květen 2008). Vysvětlení značek je uvedeno v souboru README, který je rovněž v tomto balíčku.

Z množství značek použitých v každém dokumentu jsou pro naše účely potřebné a použité pouze značky:

- `<REUTERS>` která určuje začátek a značka `</REUTERS>` která značí konec dokumentu
- `<TITLE>` a `</TITLE>` vymezuje titulek dokumentu
- `<TEXT TYPE="BRIEF"` která značí, že tento dokument je stručný
- `<BODY>` a `</BODY>` vymezuje textový obsah dokumentu

Zbylé značky jsou ignorovány, protože jejich význam pro účely shlukové analýzy je malý a také proto, že se zaměřujeme na shlukování přímo textu nikoliv jeho atributů jako v případě dat z databáze.

```
<REUTERS  TOPICS="YES"  LEWISSPLIT="TRAIN"  CGISPLIT="TRAINING-SET"
          OLDID="15530" NEWID="10009">
    <DATE>26-MAR-1987 12:22:10.06</DATE>
    <TOPICS></TOPICS>
    <PLACES><D>usa</D></PLACES>
    <PEOPLE></PEOPLE>
    <ORGS></ORGS>
```

```
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN>
&#5;&#5;&#5;RM
&#22;&#22;&#1;f1766&#31;reute
u f BC-ASLK-CGER-FINANCE-ISS   03-26 0094
</UNKNOWN>
<TEXT>
<TITLE>ASLK-CGER FINANCE ISSUES 10 BILLION YEN BOND</TITLE>
<DATELINE>LONDON, March 26 - </DATELINE>
    <BODY>
     ASLK-CGER Finance NV is issuing a 10
     billion yen eurobond due April 10, 1994 with a 5-1/2 pct
     couponand priced at 101-1/2 pct, lead manager IBJ
     International Ltd said. The bonds are guaranteed by
     Belgian savings bank ASLK-CGER Bank and have all been pre-
     placed. They will be issued in denominations of one mln
     yen and listed in Luxembourg. Fees comprise 5/8 pct for
     management and underwriting combined, with a 1/8 pct
     praecipuum, and 1-1/4 pct for selling. Pay date is April
     10.
           REUTER
    </BODY>
  </TEXT>
</REUTERS>
```

### 4.3.2 Tokenizace

Na nejnižší úrovni jsou dokumenty reprezentovány jako posloupnosti znaků a přestože jde o úplnou a nejpřesnější reprezentaci není příliš vhodná ani výhodná. Proto se snažíme z dokumentu získat termy. To jsou slova případně slovní spojení. Dokument postupně načítáme po řádcích a ty rozdělujeme na menší kusy - na jednotlivé lexikální jednotky - tokeny (Z toho název Tokenizace). Token je elementární syntaktická jednotka, tedy posloupnost znaků, která má nějaký význam. Nejčastěji se jedná o samotná slova, která známe z přirozeného jazyka. Tyto lexikální jednotky se musí převést na výsledné termy, následně použité při reprezentaci dokumentu. Převod může být jednoduchý, například lexikální jednotky budou převedeny na termy konvertováním všech znaků na malá písmena, odstraněním interpunkce, případně se může skládat z více složitějších kroků. Tokeny jsou postupně zpracovávané a podle jejich obsahu se vykonávají příslušné akce.

### 4.3.3 Odstranění značkování

Dokumenty Reuters obsahují formátovací tagy, které musí být správně zpracované. Pokud se totiž na řádku vyskytuje sekvence znaků `<TEXT TYPE="BRIEF"` říká nám tato značka, že tento dokument obsahuje sice i tagy `<TITLE>` a `</TITLE>` avšak neobsahuje `tagy <BODY>` a `</BODY>`, což by mohlo vést k nepřesnostem a špatné interpretaci mezi dvěma soubory. Proto je do souboru, který uchovává obsah dokumentu vloženo označení dokumentu (např. D45) následováno znakem konce řádku. Tagy `<TITLE>` a `<BODY>` (pokud se vyskytuje) jsou postupně odstraňovány a na jejich vnitřní obsah (vlastní text) jsou aplikovány kroky předzpracování.

### 4.3.4 Převod na termy

Funkce předzpracování, která je volaná na každý token dokumentu vyskytující se mezi tagy <BODY> a </BODY>, <TITLE> a </TITLE> se skládá z několika částí. Nejdříve dojde k nahrazení HTML tagů „<“ za „<“ a „&“ za „&“. Poté jsou všechny znaky, které reprezentují písmena, převedeny na písmena malá. V dalším kroku dojde k odstranění interpunkčních znamének (háčky, čárky, tečky...). Ještě před aplikací stemmingu se odstraní znaky, které nejsou alfabetické („\“, „(“, „<“, „\“, „–“). Nejtěžší fází je aplikování Porterova stemmovacího algoritmu na každý token. Poté již máme takové tokeny, které stačí porovnat se slovníkem nepotřebných slov. V případě výskytu ve slovníku, není toto slovo zahrnuto do výsledné množiny termů, která slouží pro následnou reprezentaci dokumentu.

### 4.3.5 Stemmovací algoritmus

Porterův algoritmus pro stemming je podle [13,14] proces odstranění prostých morfologických a skloňovaných koncovek z anglických slov. Hlavní využití je v normalizaci termů, v procesu automatického vyhledávání informace.

V [15] uvádí, že algoritmus je založený na myšlence, že všechny přípony v angličtině (přibližně 1200) jsou většinou složené z kombinace menších a jednodušších přípon. Jeho Stemmer pracuje lineárním způsobem a skládá se z pěti kroků, kde jsou postupně aplikována pravidla. V každém kroku se zjišťuje, zda příponové (suffix) pravidlo je aplikovatelné na dané slovo. Pokud tomu tak je, tak jsou testovány podmínky vztahující se k tomuto pravidlu a zjišťuje se, jaký by byl výsledný stem, pokud by byla přípona (suffix) odstraněna. Příkladem takovéhoto pravidla může být, kritérium splnění počtu samohlásek, které jsou následované souhláskou apod. Pokud jsou splněny podmínky stanovené pravidlem, akce spojena s tímto pravidlem je vykonána. To většinou znamená, že se daná přípona odstraní a pokračujeme do dalšího kroku algoritmu. Řízení, tedy může být předáno do dalšího kroku buď po aplikaci jednoho z pravidel nebo až po vyčerpání všech

možných.Tento proces pokračuje pro všech pět kroků. Za výsledný stem se považuje stem po posledním pátém kroku.

Dle [13,15] byl původní originální Porterův stemmovací algoritmus poněkud agresivní k některým typům slov. Zatímco například slova „connect“, „connected“, „connecting“, „connection“, connestions“ upraví správně na základ „connect“, tak například slovo „adding“ změní na „ad“ „security“ na „secur“, atd. Z těchto důvodů bylo nutné originální algoritmus vylepšit, a to především řešeními pro jednotlivé různé případy. Algoritmus lze pro různé programovací jazyky, včetně použité technologie Java, stáhnout z [14].

### 4.3.6 Odstranění nepotřebných slov

Velmi mnoho slov v dokumentu vůbec nepopisuje jeho obsah. Proto je velmi efektivní v transformaci lexikálních jednotek na termy tyto nevýznamové termy odstranit. Odstraní se i termy, které se vyskytují příliš často a konstantně v celé množině dokumentů, protože tyto termy jsou buď nevýznamné nebo málo významné a neovlivní nám celkový výsledek shlukování. Slova, která se mají odstranit sepisujeme do seznamu. Tento seznam se nazývá stoplist (stopwords). Podle [19] je stoplist seznam slov, které díky své vysoké frekvenci v textu ztrácejí význam pro sémantickou analýzu věty. Protože mají tato slova ve větě význam spíše gramatický, lze je tedy bez větší újmy na srozumitelnosti sdělení z věty vypustit. Jedná se především o spojky, částice a předložky, tedy slova velmi krátká. Díky jejich frekvenci se totiž na nich výrazněji projevila asimilace a redukce.

Metoda výběru slov do stoplistu podle frekvence pochopitelně může produkovat různě zkreslené výsledky. Hlavní příčinou je to, že frekvenční hranice těchto slov není a také nikdy nemůže být přesně určena. Míra srozumitelnosti a jednoznačnosti sdělení bez daného slova je totiž do velké míry subjektivní. Navíc na frekvenci závisí „pouze statisticky“. Existuje totiž celá řada slov, která se používají z globálního hlediska poměrně zřídka a přesto nemají pro analýzu věty žádný význam, slova „vlastně“, „prostě“ apod. Ve všeobecnosti se dá pomocí stoplistu odstranit 40% - 50% z celkového počtu slov. Tvorba slovníku není jednoduchou záležitostí, a proto byl použít víceméně upravený z [18] a můžeme si jej představit zhruba takto:

a
about
above
across
after
afterwards
again
against
all
almost

…

Další používané stoplisty pro anglický jazyk nalezené z internetových zdrojů, lze nalézt v v části příloh.

### 4.3.7 Předzpracování dat z databáze

Při shlukování dat z databáze předzpracování v tradičním smyslu dolovacích úloh provádět nemusíme. Již extrahovaná data, obsahující informace především o značkování stránek, jsou uložena v databázi MySQL. Ukázku, jak by taková data mohla vypadat, ukazuje obrázek 4.2.

| class | fontsize | weight | style | aabove | abelow | aleft | aright | tlength | tdigits | tlower | tupper | tspaces | textbtns | bgbtns | contrast |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| none | 94 | bold | normal | 0 | 0 | 0 | 1 | 5 | 4 | 0 | 0 | 0 | 0.30787 | 1.00000 | 2.93402 |
| none | 94 | bold | normal | 0 | 0 | 0 | 1 | 5 | 4 | 0 | 0 | 0 | 0.30787 | 1.00000 | 2.93402 |
| none | 94 | bold | normal | 0 | 0 | 0 | 1 | 5 | 4 | 0 | 0 | 0 | 0.30787 | 1.00000 | 2.93402 |
| none | 94 | bold | normal | 0 | 0 | 0 | 1 | 5 | 4 | 0 | 0 | 0 | 0.30787 | 1.00000 | 2.93402 |
| none | 128 | bold | normal | 1 | 2 | 0 | 0 | 44 | 0 | 36 | 3 | 5 | 0.13012 | 0.93054 | 5.44363 |
| none | 202 | bold | normal | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.00000 | 0.93054 | 19.61069 |
| none | 91 | normal | normal | 0 | 1 | 1 | 0 | 21 | 0 | 19 | 1 | 1 | 0.13012 | 1.00000 | 5.82928 |
| none | 91 | normal | normal | 0 | 0 | 0 | 0 | 21 | 0 | 19 | 1 | 1 | 0.13012 | 1.00000 | 5.82928 |
| none | 91 | bold | normal | 1 | 0 | 0 | 0 | 21 | 0 | 17 | 2 | 2 | 0.13012 | 0.93054 | 5.44363 |
| none | 91 | bold | normal | 0 | 0 | 0 | 0 | 21 | 0 | 17 | 2 | 2 | 0.13012 | 0.93054 | 5.44363 |
| none | 111 | normal | normal | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.00000 | 1.00000 | 21.00000 |

**Obrázek 4.2** Extrahovaná webová data

O splnění úloh předzpracování se postará až podobnostní funkce, která odstraní nejvýznamnější nepřesnosti dat, jež by nám mohly výrazně zkreslit výsledky shlukování. Jde především o snížení váhy odlehlých hodnot a vynechání chybějících hodnot. O použité funkci se dozvíme v kapitole 6.2. Pro funkci je potřeba připravit některé informace, jako je minimum a maximum daného atributu ve všech shlukovaných datech apod. Namísto předzpracování, si tedy nachystáme tyto informace.

# 5 Shlukování

Shlukování už vlastně využíváme v každodenním životě, aniž bychom si to uvědomovali. Všechny osoby můžeme rozdělit třeba do dvou tříd na kuřáky a nekuřáky, podobně rostliny a živočichové představují dvě třídy žijících organismů. Jako nástroj pro získávání znalostí z databází umožňuje shluková analýza zjistit distribuci dat a nalezení charakteristik pro jednotlivé třídy. Při shlukování se můžeme zaměřit na některé třídy objektů a dále je analyzovat. Shlukování může být také využito jako předzpracování dat pro další algoritmy, především pro algoritmy pro klasifikaci a charakterizaci, které potom pracují nad vytvořenými třídami. Z hlediska strojového učení představuje shlukování příklad učení bez učitele. Na rozdíl od klasifikace shlukování nevyžaduje žádné předdefinované třídy ani žádnou trénovací množinu příkladů (objekty, o nichž víme, do které třídy patří).

## 5.1 Vlastnosti shlukovacích metod

Již jsme se zmínili, že shlukování je proces rozdělování objektů do tříd (clusterů) na základě podobnosti objektů. Z hlediska využití shlukování pro získávání znalostí nás zajímají takové metody, které jsou schopné účinně a efektivně zpracovávat rozsáhlé databáze. Jednotlivé aplikace shlukové analýzy vyžadují metody s různými vlastnostmi. Podle [2] jsou na shlukové metody kladeny nejčastěji následující požadavky:

- **Škálovatelnost -** Schopnost metody dobře zpracovat rozsáhlé databáze. Nahrazení všech dat pouze vzorkem a jejich následná analýza totiž může vést ke zkreslení výsledků.
- **Schopnost zpracovávat různé typy atributů –** Je známo o řadě algoritmů pro shlukování numerických dat, avšak v praxi je třeba zpracování např. ordinálních dat nebo dokonce dat různého typu.
- **Vytváření shluků různého tvaru -** Nejběžnější shlukovací metody vytvářejí třídy na základě Euklidovské příp. Manhattanovské vzdálenostní funkce. Tyto metody obvykle směřují k vytváření kulovitých shluků podobné velikosti a hustoty a neumožňují vytvářet shluky libovolného tvaru, které mohou lépe odpovídat hledaným třídám dat.
- **Minimální požadavky na znalost problému při určování –** Některé metody požadují určení vstupních parametrů, a to může mít velký vliv na kvalitu nalezených shluků.
- **Schopnost vyrovnat se s daty obsahujícími šum -** Většina databází obsahuje určité procento záznamů, které obsahují chybná, neznámá nebo chybějící data. Tyto záznamy mohou výrazně snížit kvalitu nalezených shluků.
- **Necitlivost na pořadí vstupních záznamů -** Některé algoritmy mohou pro stejná data nalézt zcela různé shluky, jestliže jsou záznamy zpracovávané databáze jinak uspořádány. Proto nás zajímají takové algoritmy, které nejsou citlivé na pořadí záznamů v databázi.

- **Schopnost zpracovávat vysokodimenzionální data** - Běžné shlukovací metody zpracovávají dobře nízkodimenzionální data (datové položky se 2–3 atributy). Tato data (do 3 dimenzí) je schopné analyzovat i lidské oko. Proto jsou významné ty algoritmy, které jsou schopné zpracovávat datové položky s větším počtem atributů.
- **Schopnost shlukování na základě omezování -** Aplikace běžně vyžadují shlukování na základě různých omezení. Úkolem je nalézt třídy dat, které splňují požadovaná omezení.
- **Interpretovatelné a použitelné shluky -** Výsledkem shlukové analýzy musí být interpretovatelné, srozumitelné a použitelné shluky. Součástí nástrojů pro shlukování musí být modul, který umožní srozumitelnou reprezentaci dosažených výsledků.

## 5.2 Datové struktury při shlukování

Nejčastěji shlukovací algoritmy využívají dva druhy datových struktur a to datové matice a podobnostní matice. Datová matice podle [2] reprezentuje *n* objektů (např. osob) pomocí *p* proměnných (atributů – např. věk, výška, váha apod.). Tato struktura má podobu relační tabulky, nebo matice *n x p*. Podobnostní matice zase obsahuje vzdálenosti pro všechny dvojice objektů. Nejčastěji se reprezentuje jako *n x n* tabulka. Problémem zůstává, jak zjistit vzdálenost jednotlivých objektů. Tato problematika je odvislá od typu dat jednotlivých atributů.

### 5.2.1 Intervalové a binární proměnné

Intervalové proměnné jsou spojitá měřítka, která jsou rozdělená přibližně lineárně. Mezi typické proměnné tohoto typu patří výška, váha nebo tělesná teplota. Pro vyjádření podobnosti objektů popsaných těmito proměnnými se nejčastěji využívají vzdálenostní funkce (Euklidovská, Manhattanovská, Minkowského vzdálenost). Důležité je podle [2], vhodně zvolit použité jednotky, protože hodnoty těchto proměnných mohou ovlivnit celý výsledek shlukování. Například změna délkových jednotek z metrů na palce může vést k úplně jinému rozdělení objektů do tříd. Z těchto důvodů je vhodné data nejprve standardizovat, aby všechna data měla stejnou váhu.

Binární proměnné jsou proměnné, které mohou nabývat pouze dvou hodnot - *0* a *1*. Příkladem binární proměnné může být atribut pracující, hodnota *1* vyjadřuje, že daná osoba je zaměstnána, pro ostatní osoby bude mít atribut hodnotu *0*. Dále můžeme binární proměnné rozdělit na symetrické a asymetrické. O symetrických hovoříme, když oba stavy jsou stejně pravděpodobné, mají stejnou hodnotu a váhu např. atribut pohlaví. Naopak asymetrické binární proměnné jsou takové, jejichž obě hodnoty nejsou stejně významné. Příkladem této proměnné může být test na určité onemocnění, kdy pozitivní výsledek je významnější než negativní výsledek.

### 5.2.2 Nominální a ordinální proměnné

Nominální proměnné jsou zobecněním binárních proměnných, mohou nabývat více než dvou hodnot, ale hodnoty jsou předem definovány a jejich počet je omezen. Něco podobného jako výčtový typ v programovacích jazycích. Příkladem nominální proměnné může být například atribut typ převzetí v nějaké databázi prodejce. Tento atribut může nabývat hodnot osobní převzetí, na dobírku či expresní přepravu.

Podobné nominálním proměnným jsou proměnné ordinální, ale hodnoty, kterých mohou nabývat, jsou uspořádány v určitém pořadí. Podle [2] je toto pořadí hodnot významné a umožňuje vyjádřit subjektivní ohodnocení různých vlastností, které nelze měřit objektivně. Příkladem ordinální proměnné může být například atribut *obtížnost_projektu*, který může nabývat hodnot: *velmi_lehký, lehký, přiměřený, těžký, velmi_těžký*. Ordinální proměnné lze potom zpracovávat podobně jako intervalové proměnné, nejprve ovšem jednotlivým hodnotám musíme přiřadit číselnou hodnotu.

### 5.2.3 Zpracování proměnných různého typu

Výše jsme si uvedli, jak zpracovat jednotlivé typy proměnných. V běžných aplikacích však obvykle pracujeme s objekty, které jsou popsány pomocí atributů různých typů, jak uvádí [1, 2]. Otázkou tedy zůstává, jak určit vzdálenost objektů na základě atributů různých druhů. Jednou možností je seskupit proměnné stejných typů a provést zvláštní shlukovou analýzu pro jednotlivé skupiny proměnných. Výsledky těchto analýz jsou však obvykle různé a nevypovídají nic o struktuře dat, což je nežádoucí. Schopnost shlukové analýzy, nalézt nějaké struktury v datech, je založena na tom, že při analýze se kombinují různé atributy objektů. Tyto atributy jsou obvykle různého typu, a proto shlukování podle jednotlivých typů proměnných většinou není dostačující. Lepší je zpracovat všechny atributy vhodným způsobem dohromady a provést jednu shlukovou analýzu.

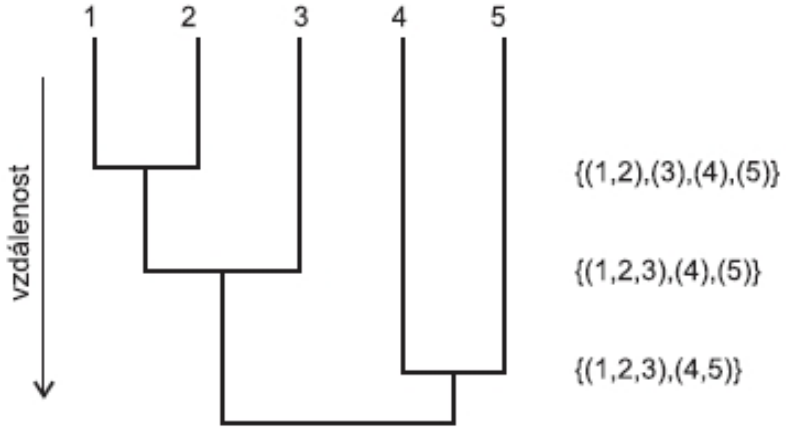
## 5.3 Typy shlukovacích metod

Shlukovací metody můžeme v zásadě rozdělit podle cílů, k nimž směřují, na hierarchické a nehierarchické. Hierarchické shlukování je systém navzájem různých neprázdných podmnožin množiny, v němž průnikem každých dvou podmnožin je buď jedna z nich nebo prázdná množina a v němž existuje alespoň jedna dvojice podmnožin, jejichž průnikem je jedna z nich. Naopak nehierarchické shlukování je takový systém, kde je průnik shluků prázdný, jedná se o disjunktní množiny.

My však můžeme podle [2] rozdělení do skupin zjemnit na hierarchické metody, metody založené na rozdělování, metody založené na mřížce, metody založené na hustotě, metody založené na modelech, metody pro shlukování vysoce-dimensionálních dat.

### 5.3.1 Hierarchické metody

Tyto metody vytvářejí hierarchický rozklad dané množiny objektů, který je v literatuře [8] znám jako dendrogram. Dendrogram je binární strom znázorňující hierarchické shlukování. Každý uzel tohoto stromu představuje shluk. Horizontální řezy dendrogramem obrázek 4.1 jsou rozklady ze shlukovací sekvence. Vertikální směr v dendrogramu představuje „vzdálenost“ mezi shluky (rozklady).




**Obrázek 4.1** Dendrogram (převzato z [8])

Podle směru postupu při shlukování dělíme metody hierarchického shlukování podle některých pramenů [8] na aglomerativní a divizivní. Podle [2] srozumitelněji na shlukující a rozdělující. Shlukující hierarchické metody (metody shlukování zdola-nahoru) nejprve umístí každý objekt do zvláštní třídy. Následně dochází ke slučování nejpodobnějších tříd, dokud všechny třídy nejsou spojeny do jedné třídy nebo nedosáhneme požadovanou úroveň shlukování. Většina hierarchických shlukovacích metod patří do této kategorie. Rozdělující hierarchické metody (metody shlukování shora-dolů) nejprve umístí všechny objekty do jedné třídy. Následně se jednotlivé třídy rozdělují na menší třídy, dokud každý objekt není ve zvláštní třídě nebo nedosáhneme požadované úrovně rozdělování.

Základní nevýhodou těchto metod je to, že pokud některé třídy sloučíme nebo rozdělíme, není už možné nikdy tyto třídy znovu rozdělit nebo spojit. Výpočetní složitost těchto metod je menší než složitost metod založených na rozdělování, ale hierarchické metody nemusí být dostatečně přesné.

### 5.3.2 Metody založené na rozdělování

Tyto metody rozdělují $n$ objektů do $k$ tříd, kde $n \leq k$. Jednotlivé třídy musí splňovat určité podmínky: Každá třída musí obsahovat alespoň jeden objekt a každý objekt patří pouze do jedné třídy. Podle [2] v prvním kroku shlukování se náhodně vybere $k$ objektů, které reprezentují jednotlivé třídy, a ostatní prvky se na základě podobnosti (vzdálenostní funkce) rozdělí do jednotlivých tříd.

Následně se iterativně hledají objekty, které nejlépe reprezentují jednotlivé třídy, a objekty se přesunují mezi třídami tak, aby podobnost prvků uvnitř jedné třídy byla maximální a podobnost prvků z různých tříd minimální. Pro dosažení optimálního rozdělení by bylo třeba provést důkladný výpočet všech možných rozdělení. V praxi se však využívají různé heuristiky zejména pak k-means, která je založena na centrálním bodu a k-medoids, která je založena na reprezentujícím objektu. Liší se tím jak určují příslušnost objektu do jednotlivých tříd. Každá metoda využívá jiný bod, který reprezentuje třídu, a příslušnost objektu do tříd se určuje na základě vzdálenosti objektu od tohoto bodu. Za použití těchto heuristik rozdělovací metody dobře naleznou shluky kulovitého tvaru pro menší a středně velké množiny objektů. Pro nalezení shluků složitějšího tvaru je nutné tyto metody rozšířit, případně upravit. Jelikož tato metoda byla implementována seznámíme se s ní podrobněji, a to v sedmé kapitole.

### 5.3.3 Metody založené na mřížce

Tyto metody využívají víceúrovňovou mřížkovou datovou strukturu. Prostor objektů rozdělují na mřížku. Všechny operace shlukování probíhají nad touto mřížkovou strukturou. Hlavní výhodou těchto metod je rychlá doba zpracování datové množiny.

### 5.3.4 Metody založené na hustotě

Metody založené na hustotě podle [2] považují za shluky oblasti s velkou hustotou objektů v prostoru dat, které jsou od sebe oddělené oblastmi s malou hustotou vyskytujících se objektů. Objekty, které se vyskytují v oblastech s malou hustotou objektů, se považují za šum. Tyto metody tak umožňují nacházet shluky různých tvarů a navíc odolné vůči šumu v datech a výstředním hodnotám.

### 5.3.5 Metody založené na modelech

Podle [2] se metody založené na modelech snaží optimalizovat shodu mezi datovou množinou a nějakým matematickým modelem, tzn. snaží se nalézt takové shluky, které by co nejvíce odpovídaly danému modelu. Tyto metody jsou nejčastěji založeny na tom, že data jsou generována na základě nějaké složené pravděpodobnostní distribuční funkce. Mezi tyto metody patří například metoda Expectation-Maximization, která představuje rozšíření algoritmu k-means, dále konceptuální shlukování a metody neuronových sítí.

### 5.3.6 Metody shlukování vysoce-dimensionálních dat

Většina výše zmíněných shlukovacích metod byla navržena pro shlukování dat s malým počtem dimenzí (atributů) a často se potýkají s řadou problémů při shlukování vysoce-dimensionálních dat. Hlavním důvodem jak uvádí [1,2] je to, že s rostoucím počtem dimenzí je pouze malé množství dimenzí, které jsou relevantní pro jednotlivé shluky. Data v ostatních dimenzích mohou produkovat

příliš mnoho šumu a znemožnit objevení shluků. Navíc s rostoucím počtem dimenzí dochází k většímu rozptýlení dat.

Data, která se nacházejí v různých dimenzích, potom mohou být považována za stejně vzdálená a nelze využít vzdálenostní funkci. Řešením těchto problémů je využít metodu transformace rysů nebo metodu výběru atributů.

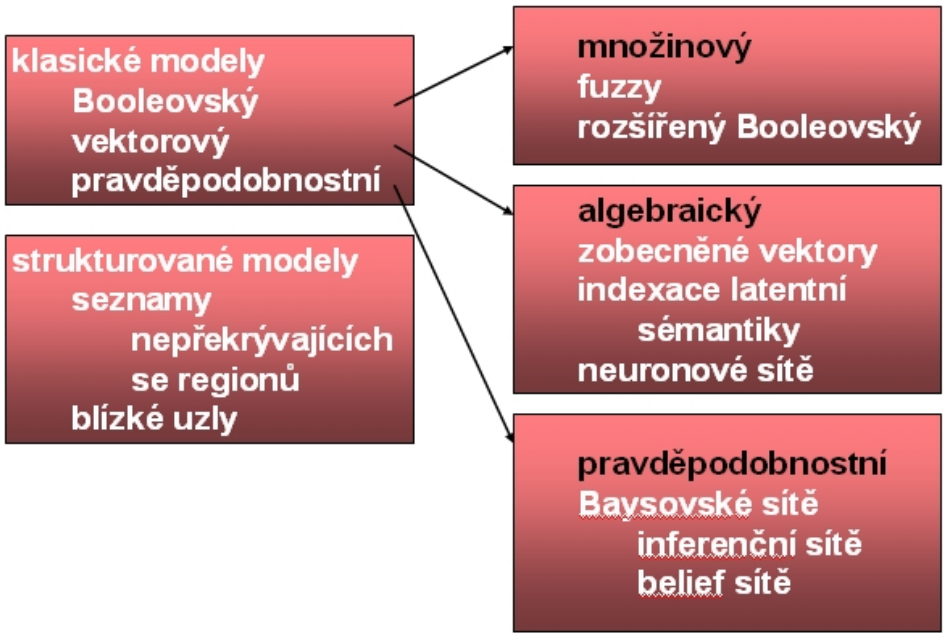
Metoda transformace rysů transformuje data do prostoru s menším počtem dimenzí při zachování relativních vzdáleností mezi objekty. Dochází k sumarizaci dat vytvářením lineárních kombinací atributů (rysů) a může dojít k odhalení struktur ukrytých v datech. Při této transformaci nedochází k odstranění žádných atributů. Tento fakt může způsobovat problémy při velkém počtu irelevantních atributů, které mohou i po transformaci maskovat skutečné shluky. Navíc vytvořené rysy je těžké interpretovat, což snižuje užitečnost výsledku shlukování. Metoda je vhodná pouze pro datové množiny, kde většina dimenzí je relevantní.

Metoda výběru atributů je založena na odstranění irelevantních dimenzí (atributů). Dochází tak k nalezení atributů, které jsou pro danou úlohu nejvíce relevantní. Provádí se prohledávání různých podmnožin atributů, jejich vyhodnocování pomocí různých kritérií. Nejčastěji se využívá strojového učení s učitelem, kdy nejvíce relevantní atributy jsou nalezeny podle ohodnocení jednotlivých tříd objektů. Analýza podmnožin atributů může být provedena také na základě entropie.

# 6 Reprezentace dokumentu a míry podobnosti

## 6.1 Modely dokumentů

Problémem jak reprezentovat a porovnávat mezi sebou dva dokumenty, případně jak efektivně vyhledávat relevantní dokumenty podle nějakého složitějšího výrazu, řeší dokumentografické informační systémy již od raných počátků své existence. Prvním modelem byl model booleovský, pak se postupně rozšířil na komplexnější model vektorový či model pravděpodobnostní. Kompletní rozdělení modelů nám shrnuje obrázek 6.1.




**Obrázek 6.1** Přehled modelů (převzato z [10])

### 6.1.1 Booleovský model dokumentů

Nejstarším a v současnosti nejrozšířenějším modelem dokumentů databáze je model Booleovský. Teoretické základy byly u dokumentografických informačních systémů navrženy již v 50. letech. V současné době jsou hlavním nástrojem i známých "vyhledávacích strojů" na Internetu, jako je AlstaVista, HotBot a další. Je zajímavé, že v těchto nástrojích se Booleovský model zahrnuje pod tzv. pokročilé vyhledávání.

Podle [9, 10] můžeme booleovský model vnímat jako reprezentaci dokumentu pomocí množin termů. To znamená, že máme kolekci dokumentů D obsahující *m* dokumentů $D = \{d_1, d_2, \ldots d_m\}$ a dokument složený z *n* termů $T = \{t_1, t_2, \ldots t_n\}$, kde $t_i$ je term, tedy slovo či sousloví. Dokument pak reprezentujeme pomocí množiny termů obsažených v dokumentu $d_1 \subseteq T$. Uchováváme pouze informaci o přítomnosti či nepřítomnosti termu v daném dokumentu, nejsou tedy důležité počty opakování termů v dokumentu jako v jiných modelech. Dotazování nebo porovnání pak probíhá pomocí Booleovských výrazů. Dotazem je tedy např. logický výraz „databáze“ AND NOT „obchod“ OR „SQL“. Odpovědí na dotaz jsou ty dokumenty, jejichž popis splňuje zadanou podmínku. Častým rozšířením booleovského modelu je povolení použití zástupných znaků a regulárních výrazů v termech dotazu.

Ač jde o jednoduchý princip, naráží tento model při praktickém použití na řadu problémů. Pokládání efektivních dotazů, tj. dotazů, které vyberou z databáze co nejvíce relevantních dokumentů při potlačení výstupu dokumentů nerelevantních, vyžaduje jisté zkušenosti, často i znalost databáze. Některé zdroje [11] dokonce uvádějí, že formulace dotazů je spíše uměním než vědou. Další nevýhodou je, že při konjunktivním dotazu jsou zamítnuty dokumenty, které neobsahují jeden z termů, stejně jako dokumenty, které neobsahují žádný z uvedených termů. Stejně tak při disjunktivním dotazu jsou vybrány dokumenty, které obsahují jeden z termů, stejně jako dokumenty, které obsahují všechny uvedené termy. Problémem je i celková sémantika dokumentu. Dokument se něčeho týká, ale přímo to slovo v něm není. Jako výhodné se nedá považovat ani fakt, že všechny termy v dotazu i v identifikaci dokumentu jsou chápány jako stejně důležité. Tento poslední problém se dá eliminovat pomocí rozšířeného Booleovského modelu, který zavádí váhy termů dokumentu i dotazu.

### 6.1.2 Pravděpodobnostní model dokumentů

Pravděpodobností model podle [10] se na daný dotaz snaží vrátit dokumenty v pořadí s klesající pravděpodobností relevance. Základní předpoklad je, že je-li dán dotaz, pak existuje množina dokumentů, která obsahuje pouze relevantní dokumenty a žádné jiné (ideální odpověď). Dotazovací proces je proces specifikace vlastností ideální odpovědi. Jelikož tyto vlastnosti v době dotazu nejsou známy, provádíme většinou první odhad. První odhad zahrnuje vygenerování předběžného pravděpodobnostního popisu ideální odpovědi, která je použita k vyhledání první množiny dokumentů. Pomocí interakce s uživatelem je pak dosaženo vylepšení pravděpodobnostního popisu výsledku dotazu. Pravděpodobnostní model se nejčastěji reprezentuje pomocí Bayesova vzorce pravděpodobnosti: $P(a \mid b) = \frac{P(b \mid a)\,P(a)}{P(b)}$

### 6.1.3 Vektorový model dokumentů

Metoda reprezentace dokumentu pomocí vektorového modelu je přibližně o 20 let mladší než booleovský model a snaží se minimalizovat nebo odstranit jeho nevýhody. Hlavním rysem vektorového modelu je reprezentace dokumentů pomocí vektoru vah termů.. Každý jeden term definuje jednu dimenzi. Podle [9,10] mějme kolekci dokumentů D obsahující *m* dokumentů $D = \{d_1,d_2, \ldots d_m\}$ a dokument složený z *n* termů $T = \{t_1,t_2, \ldots t_n\}$, kde $t_i$ je term, tedy slovo či sousloví. Dokument je pak reprezentován vektorem $D_i = (w_{i1}, w_{i2}, \ldots, w_{im})$, kde $w_{ij}$ je váha náležející termu $t_j$ v reprezentaci dokumentu $D_i$.

Jelikož tento model byl použit pro implementaci, přibližme si jeho detaily na příkladě. Představme si, že se pohybujeme v kolekci dokumentů, kde množina termů T, podle kterých se vyhledává, se skládá z termů "databáze", "term", "koeficient přesnosti", "koeficient úplnosti", "zpracování textu". Nechť obsahuje tři dokumenty obsahující termy:

(1) databáze, term, koeficient úplnosti,

(2) koeficient úplnosti, koeficient přesnosti,

(3) databáze, zpracování textu

Reprezentace těchto záznamů pomocí vektorů je:

(1,1,1,0,0)

(0,0,1,1,0)

(1,0,0,0,1)

kde 1, resp. 0 sděluje fakt, že daný term z T existuje, resp. neexistuje v záznamu. Uvažujme dotaz na dokumenty s termy "databáze" a "zpracování textu". Jeho vektor bude (1,0,0,0,1). Vyzkoušíme-li v Booleovském modelu dotaz (konjunkce) vzhledem k záznamům, zjistíme, že dokument (3) je hitem. Zajímavý však může být i dokument (1), obsahuje term "databáze", dokument (2) je zřejmě nerelevantní. Tato fakta lze též zjistit ekvivalentním způsobem, vynásobíme-li vektor dotazu skalárně s vektory databáze, tj. např. $(1,1,1,0,0) \cdot (1,0,0,0,1) = 1$. Postupným provedením tří součinů obdržíme čísla 1, 0, 2 a můžeme tedy seřadit dokumenty do pořadí (3), (1), (2).

Když budeme udržovat pro jednotlivé dokumenty počty výskytů jejich termů v rámci dokumentu, získáme jednoduchý systém vážení. Např. pro dokument (1) by mohl vypadat vektor vah jako (3,0,0,0,2). Aplikací skalárního součinu s vektorem dotazu bychom obdrželi číslo 5. Je vidět, že tato čísla již nevyjadřují jednoduše shodu termů dotazu s termem v dokumentu, ale něco víc, jaké-si kvantitativní ohodnocení míry podobnosti dokumentu s dotazem. Pro relevanci by totiž mohlo být významné, že daný term se vyskytuje v dokumentu vícekrát než druhý term. Tato idea je základem složených heuristik založených na frekvenci. Připomínám, že dotaz je formou vektoru jiného dokumentu, jelikož se jedná o shlukování podobných dokumentů. Přesto, že vektorový model patří

k nejstarším, není dosud známá efektivní implementace. V literatuře je nejčastěji zmiňován problém, jak přesně a matematicky správně postupovat při výpočtu podobnosti dokumentů.

### 6.1.4 Určování vah termů

Většina způsobů určování vah je založena na pozorování, že významnost termů přímo souvisí s frekvencí výskytu termu v dokumentu. Frekvence termu v dokumentu je číslo, které udává počet výskytů termu v dokumentu. Frekvenci termu $t_j$ v dokumentu $D_i$ dle [10] označíme symbolem TF. Ta označuje tzv. prostou frekvenci. Tato váha nám však nebere v úvahu lišící se délky dokumentů, což by mohl být problém, kdyby se dokumenty příliš lišily rozsahem. Proto se častěji používá normalizovaná frekvence termů, která je definovaná vzorcem:

$$NTF = 0{,}5 + \frac{0{,}5 \cdot f(t,d)}{MaxFreq(d)},$$

kde *MaxFreq* je přes všechny termy v i-tém řádku matice D, tedy maximální frekvence nejfrekventovanějšího termu v rámci dokumentu.

Samotná TF či NTF ještě nezajišťuje kvalitní výsledky. Je nutné vzít do úvahy také frekvenci, s jakou se daný term vyskytuje v celé kolekci dokumentů, neboť nejvhodnějšími termy jsou ty, které se vyskytují zhruba v polovině všech dokumentů. Proto se frekvence termu v dokumentu obvykle násobí opravnými koeficienty. Jedna z používaných metod využívá tzv. frekvenci podle invertovaného dokumentu (IDF). IDF pro term *t* klesá se zvyšujícím se počtem dokumentů, ke kterým je term *t* přiřazen, což je více diskriminativní. IDF je definován předpisem:

$$IDF = 1 + \log\left(\frac{n}{k}\right),$$

kde *n* je celkový počet dokumentů v kolekci a *k* je počet dokumentů, ke kterým je term $t_j$ přiřazen. Zde je šikovně využito logaritmické křivky, term se vyskytuje ve všech dokumentech → log(1) = 0 a term patří mezi nevýznamová slova.

Výslednou váhu termu $t_j$ v dokumentu nakonec vytvoříme součinem TF a IDF, tedy $w_{ij} = TF_{ij} \cdot IDF_{ij}$. Výsledná váha nám velmi dobře zachycuje situaci, kdy je term hodně frekventovaný v rámci dokumentu, pak je vysoká váha. A stejně tak vysoká selektivita mezi ostatními dokumenty, nám zajišťuje vysokou váhu termu.

## 6.2 Podobnostní funkce

Další co potřebujeme pro shlukovací algoritmus je zajistit výpočet podobnosti dvou dokumentů, potažmo vzdálenost. Jak tento výpočet zajistit pro normální data, jsme si uvedli v páté kapitole. Ovšem u dokumentů je to z důvodů mnoha dimenzí, které jsou navíc navzájem řídké, poměrně složitou záležitostí. Podle [10] označujeme podobnost dvou dokumentů $D_i$ $(w_{i1}, w_{i2}, ..., w_{im})$ a $D_j$ $(w_{j1},$

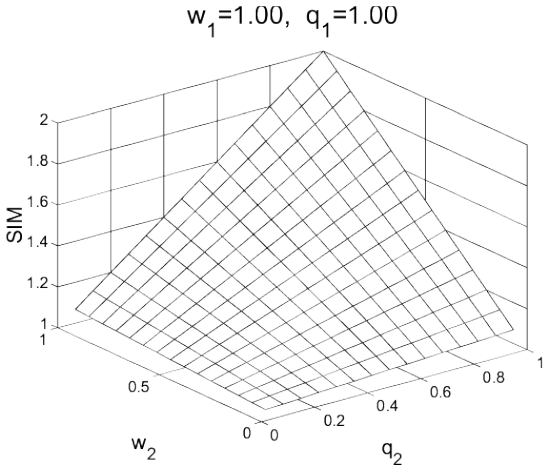
$w_{j2}$, ..., $w_{jm}$) jako *sim* ($D_i$, $D_j$) (z anglického similarity). Pro výpočet takového koeficientu se používá řada vzorců.

### 6.2.1 Skalární součin

V nejjednodušším případě může být definován jako skalární součin, tedy vztahem:

$$Sim(D_i, D_j) = \sum_{t=i}^{N} w_{it} \cdot w_{jt}$$

Je patrné, že kolmé vektory budou mít nulovou podobnost. Nevýhodou je skutečnost, že delší vektory přiřazené obvykle delším dokumentům mohou být zvýhodněny. Tuto nevýhodu odstraňujeme většinou normalizací podobnostní funkce. Nešikovná je i práce s touto podobnostní funkcí, při určování míry nepodobnosti respektive vzdálenosti, což dokládá i obrázek 6.2.




**Obrázek 6.2** Rozložení míry podobnosti skalárním součinem (převzato z [11])

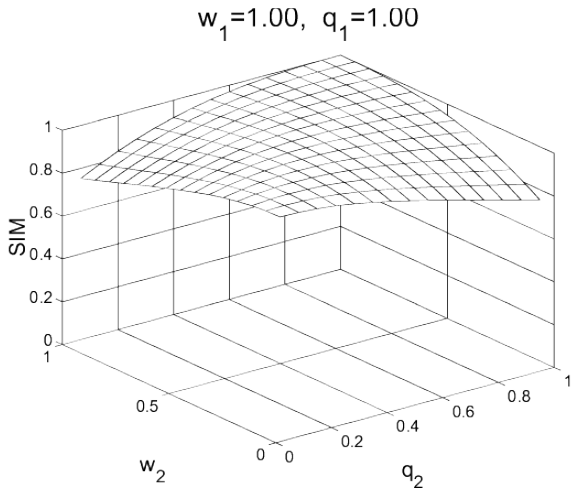
### 6.2.2 Kosinova míra

O něco užitečnější je tzv. Kosinova míra. Ta je podle [10] definována jako normalizovaný skalární součin:

$$Sim(D_i, D_j) = \frac{\sum_{t=i}^{N} w_{it} \cdot w_{jt}}{\sqrt{\sum_{t=1}^{N} (w_{it})^2 \cdot \sum_{t=1}^{N} (w_{jt})^2}}$$

S touto mírou určení podobnosti se setkáme v praxi mnohem častěji, protože dává daleko lepší výsledky. Používá se prakticky pro všechna možná kvantitativní data, včetně dokumentů, zároveň je

lehce převoditelná na míru nepodobnosti. Rozložení Kosinovou mírou nám ukazuje obrázek 6.3, kde je vidět, že koeficient podobnosti se pohybuje v intervalu 0 až 1.



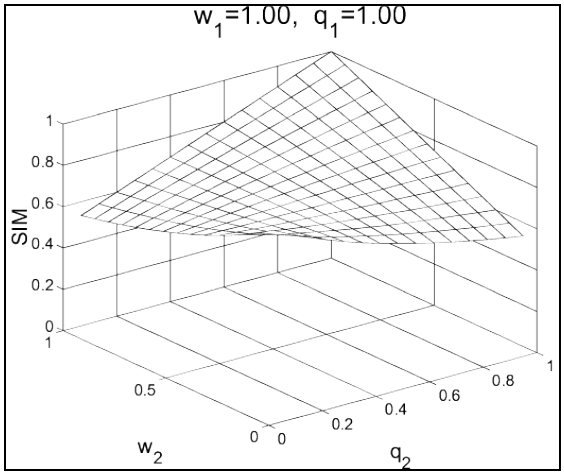

**Obrázek 6.3** Rozložení míry podobnosti Kosinovou mírou (převzato z [11])

## 6.2.3 Jaccardova míra

Trošku jiný přístup je použití heuristických mír. Takovou je podle [11] Jaccardova míra a je definována výpočtem:

$$Sim(D_i, D_j) = \frac{\sum_{t=i}^{N} w_{it} \cdot w_{jt}}{\sum_{t=1}^{N} (w_{it}) + \sum_{t=1}^{N} (w_{jt}) - \sum_{t=1}^{N} (w_{it} \cdot w_{jt})}$$

Tato míra bere v úvahu normalizaci vektorů, vše je opět patrné z obrázku rozložení této míry, jde o obrázek 6.4.



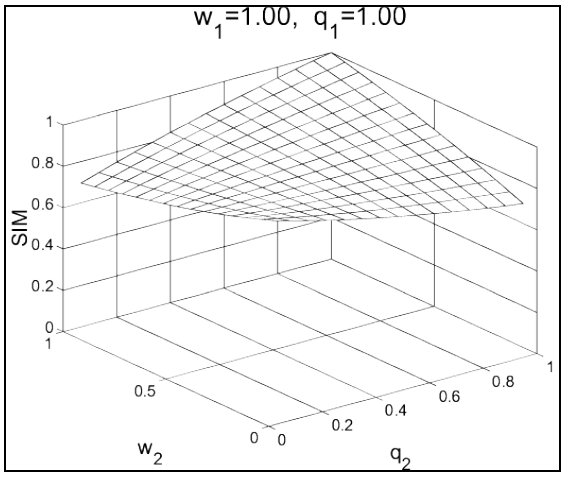

**Obrázek 6.4** Rozložení míry podobnosti Jaccardovou mírou (převzato z [11])

### 6.2.4 Diceova míra

Další míra opět vychází z výpočtů míry pro asymetrické binární proměnné, stejně jako předchozí. Někdy je Diceova míra označována za Czekanowského a má podle [11] vzorec na výpočet následovaný:

$$Sim(D_i, D_j) = \frac{2 \cdot \sum_{t=i}^{N} w_{it} \cdot w_{jt}}{\sum_{t=1}^{N} (w_{it}) + \sum_{t=1}^{N} (w_{jt})}$$

Míra bere opět v úvahu různé délky dokumentů, což lze dobře pozorovat na obrázku 6.5.




**Obrázek 6.5** Rozložení míry podobnosti Diceovou mírou (převzato z [11])

### 6.2.5 Převod měr podobnosti na míry nepodobnosti

Pro shlukovací algoritmus je někdy výhodnější, případně to může i algoritmus vyžadovat, použití vzdálenostní funkce. Vzdálenostní funkce je vlastně opačná hodnota pro míru podobnosti, tedy míra nepodobnosti. Můžeme to chápat tak, že čím větší vzdálenost, tím jsou si objekty (dokumenty) méně podobné. Pro kvantitativní data, jako jsou třeba intervalové proměnné, je situace jednodušší. Vzdálenost nejčastěji určujeme euklidovskou vzdáleností:

$$d(i, j) = \sqrt{\left|x_{i1} - x_{j1}\right|^2 + \left|x_{i2} - x_{j2}\right|^2 + \ldots + \left|x_{ip} - x_{jp}\right|^2}$$

Další v praxi používané vzdálenostní funkce jsou Manhattonská a Minkowského.

Pro náš případ a dokumenty použijeme jiný přístup, a to je převod dříve uvedených mír. Nejprve si shrňme požadavky na takovou vzdálenostní funkci. Vzdálenost je vždy nezáporné číslo, tedy $d(i,j) \geq 0$, vzdálenost od sebe sama je rovna nule $d(i,i) = 0$, musí být symetrická $d(i,j) = d(j,i)$ a vzdálenost mezi objekty *i* a *j* v prostoru nesmí být větší než součet vzdáleností objektů *i* a *j* od objektu *h* (podmínka triangularity). Potom můžeme pro normalizované podobnostní míry definovat

vzdálenost podle [10, 11] následovně: Jestliže podobnostní míra je v intervalu 0 až 1, kde hodnota 1 reprezentuje maximální nepodobnost, platí vztah: *d(i,j) =1 – sim*, pokud jsou hodnoty v intervalu -1 až 1, pak se použije vztah: *d(i,j) =1 – 2·sim*. Podle první varianty převádíme tedy Kosinovu, případně jiné míry na vzdálenost.

### 6.2.6 Míra podobnosti použitá na databázová data

V předchozích kapitolách již bylo naznačeno, jak by mohly být zpracovány objekty s různými atributy. Na databázová data byla tedy použita míra podobnosti, v tomto případě vzdálenost definovaná podle [2] následovně:

$$d(i,j) = \frac{\sum_{l1}^{p} w_{ijl} \cdot d_{ijl}}{\sum_{l1}^{p} w_{ijl}}$$

Kde váha $w_{ijl}$ nabývá hodnoty 0, jestliže hodnota $x_{ijl}$ v databázi není, nebo a $x_{il} = x_{jl} = 0$ a proměnná je asymetrická binární. V ostatních případech je $w_{ijl}$ je rovna 1. Příspěvek proměnné $d_{ijl}$ k celkové vzdálenosti objektů i a j se vypočítá podle typu proměnné takto:

Jestliže jde o binární nebo nominální proměnnou, pak $d_{ijl} = 0$, jestliže $x_{il} = x_{jl}$,, v ostatních případech $d_{ijl} = 0$. V případě, že jde o intervalovou proměnnou pak:

$$d(i,j) = \frac{\left| x_{il} - x_{jl} \right|}{\max_h x_{hl} - \min_h x_{hl}}$$

Kde $h$ jde přes všechny hodnoty proměnné pro jednotlivé objekty, v našem případě řádky tabulky. Lze tedy říci, že jde o absolutní hodnotu z rozdílu hodnot dělenou variačním rozpětím. Jestliže jde o ordinální nebo poměrovou proměnnou, vypočítáme hodnoty $r_{il}$ a $z_{il}$ a hodnotu $z_{il}$ zpracujeme jako intervalovou proměnnou.

Pro databázová data v aplikaci využijeme jen intervalové, binární a nominální proměnné, definované typem sloupce, tedy atributem. Porovnávané objekty pak pro nás budou jednotlivé řádky tabulky.

# 7 Algoritmus k-means

Algoritmus k-means byl poprvé uveden v roce 1967 J. B. MacQueenem v publikaci Some Methods for classification and Analysis of Multivariate Observations. Algoritmus k-means (česky k-průměrů) je založen na vzdálenosti bodů v mnohorozměrném prostoru. Podle [2] vypadá algoritmus následovně:

**Vstup:**

$D = \{ t_1, t_2, ..., t_n \}$ množina objektů

$k$ = počet požadovaných shluků

**Výstup:**

$K$ = množina shluků

**Postup:**

Přiřaď počáteční hodnoty k středům $m_1, m_2, ...m_k$ (libovolně);

**repeat**

přiřaď každý objekt ke shluku s nejbližším středem;

vypočti nové středy shluku;

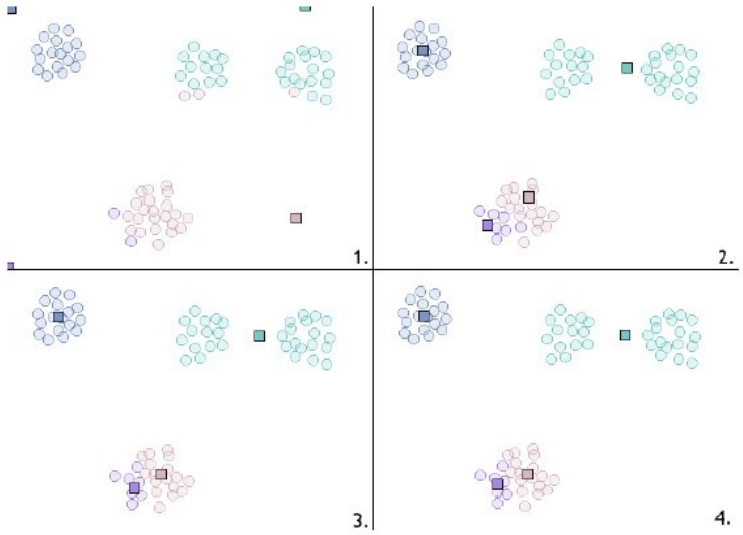
**until** konvergenční kriterium je splněno;

Algoritmus je iterační. V inicializační části nastavíme počet shluků $k$, které na výstupu požadujeme (odtud k-means). Je vytvořeno $k$ bodů s náhodnými souřadnicemi tzv. centroidy. V iterační části je v každém kroku nejdříve každý bod přiřazen nejbližšímu centroidu (shluk je reprezentován centroidem). Poté jsou všechny centroidy přepočítány a aktualizovány. Iterace pokračuje, dokud mění příslušnost některých bodů centoridům. Je dokázáno, že algoritmus je konečný.

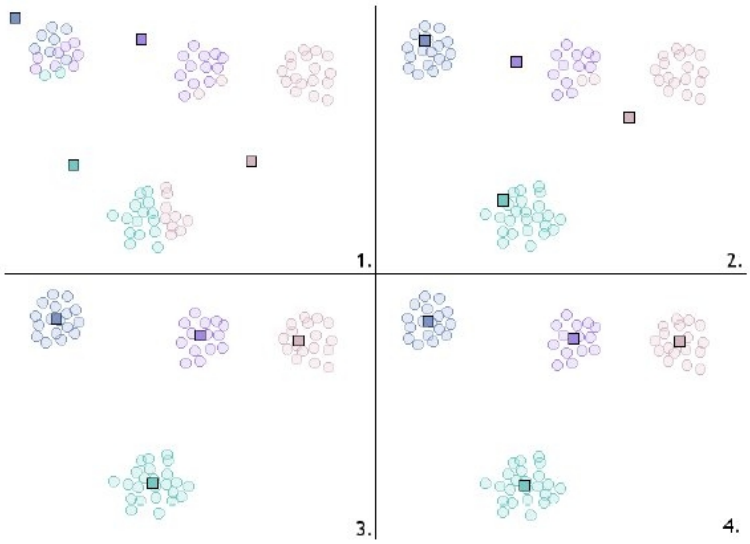
## 7.1 Vlastnosti

Uvedený algoritmus má řadu výhod i nevýhod. Při vytváření malého počtu shluků z velkého počtu objektů se jeví nejužitečnější shlukovací metoda. Funguje dobře, především, pokud data tvoří kompaktní shluky, které jsou poměrně dobře oddělené. Složitost algoritmu je $O(nkt)$, kde $n$ je počet objektů, $k$ je počet shluků a $t$ je počet iterací. Metoda nemusí najít optimální výsledek, nýbrž může uvíznout v lokálním extrému. Nevýhodou této metody je to, že musíme předem znát počet shluků, které chceme nalézt. Zadání tohoto parametru může být velmi obtížné, pokud o struktuře dat nic předem nevíme. Metoda navíc neumožňuje dobře nalézt shluky nekonvexního tvaru a shluky různé

velikosti. Metoda je také citlivá na šum a odlehlé hodnoty v datech, které mohou značně zdeformovat výsledné rozložení shluků.




**Obrázek 7.1** Špatný úvodní „nástřel“ = „špatný“ výsledek. Vpravo dole v konečné fázi, je vidět, že vypočtené středy a shluky neodpovídají intuitivní představě.




**Obrázek 7.2** Dobrý úvodní „nástřel“ = „dobrý“ výsledek. Je vidět, že algoritmus má optimální průběh a výsledek odpovídá intuitivní představě.

Na obrázcích 7.1 a 7.2 vytvořených pomocí appletu dostupného na URL <http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html> jsou demonstrovány dva různé výsledky pro stejná vstupní data, zároveň nám odkrývají velký problém tohoto algoritmu, a tím je úvodní volba centrálních bodů. Na obrazcích lze identifikovat čtyři shluky, přičemž data jsou zobrazena jako kolečka a středy jako čtverečky. Je patrné, že v prvním případě nemůžeme výsledek

považovat za optimální (neodpovídá intuitivní představě). V druhém případě, pro jiné počáteční nastavení centrálních bodů, už výsledek mnohem lépe koresponduje s naší představou dobrého řešení.

## 7.2 Varianty

Existuje několik variant této metody. Většinou se liší inicializací (první výběr reprezentantů shluků), způsobem výpočtu vzdálenosti objektů a způsobem výpočtu centrálních bodů. Dobré výsledky jsou vždy závislé na dobrém úvodním výběru reprezentantů. Čím je „úvodní nástřel“ rozprostřenější a od sebe navzájem vzdálenější, tím lépe.

Jedním z mnoha přístupů, který literatura uvádí, je spustit nejdříve shlukování zdola-nahoru. To znamená, že se zvolí náhodně množina dokumentů, kde každý objekt je na začátku shluk. Shluky se postupně sjednocují do počtu zadaného uživatelem. Poté se nad danými shluky spočítají centrální body a použije se tradiční dříve popsaný k-means. Celková časová složitost se dramaticky nezvýší, zvýší se jen logaritmicky, což může být při některých datech výhodně investovaná složitost navíc.

Jiný přístup je opakování algoritmu, který vždy startuje z jiného počátečního uspořádání. Nakonec vybere optimální řešení ze všech možných dosažených uspořádání shluků. Pro výběr nejlepšího dosaženého řešení lze použít vhodně zvolenou metriku tzv. míru kvality rozložení. Může jí být třeba součet všech vzdáleností od jednotlivých bodů k jednotlivým centrálním bodům. Řešení s nejmenší takto vypočítanou hodnotou je nejlepší námi dosažené (ale opět může jít pouze o lokální optimum).

Důležité pro všechny varianty je tedy především počáteční rozprostření centrálních bodů. Čím rozličnější se podaří, tím se zvýší pravděpodobnost nalezení správného řešení.

## 7.3 Centrální bod

Centrální bod, kterému se také říká centroid, hraje při použití algoritmu k-means důležitou roli, protože je na něm celý algoritmus založen. Centrální bod můžeme chápat jako střed shluku, tedy bod ve více dimenzionálním prostoru, který má nejnižší průměrnou vzdálenost ke všem objektům shluku. Například pro numerická data je výpočet takového bodu vcelku jednoduchý. Vypočteme aritmetický průměr pro každou dimenzi přes všechny objekty shluku. Tyto průměry pak tvoří jednotlivé dimenze vektoru, jímž je definován centrální bod tohoto shluku.

Situace se, ale dosti komplikuje pro data jakými jsou dokumenty. Podle [20, 23] je výhodné, a to nejen pro výpočet centrálních bodů, použití některého z principů snižování dimenzionaility. Dokonce uvádí, že shlukovací algoritmy založené na vzdálenostech fungují efektivně do šestnácti proměnných. Je potřeba si uvědomit, že počet dimenzí, s nimiž pracujeme, se reálně pohybují ve stovkách tisíc. Nejjednodušším způsobem je ignorování přebytečných dimenzí a použitích pouze prvních několika, ale většinou se stává, že tato cesta nevede k dobrým výsledkům. O něco

výhodnějším je výběr vhodných slov, která svým výskytem ve všech dokumentech významným způsobem korelují. Tento proces je výhodné provádět již při předzpracování dat.

V aplikaci použijeme první přístup, vypočítáme průměrnou hodnotu vah termů a vytvoříme z nich vektor reprezentující centrální bod shluku.

V případě shlukování dat z databáze je situace o něco jednodušší. Je použita docela intuitivní reprezentace centrálního bodu. Podle typu atributu vypočteme buď aritmetický průměr, to v případě intervalových proměnných,nebo v případě nominálních proměnných určíme nejčastěji se vyskytující hodnotu a tu použijeme.

# 8 Realizace

Zvolený algoritmus k-means jsem se rozhodl implementovat v jazyce Java, konkrétně podle nového znační Java Standard Edition 6. Pro shlukování se používá buď kolekce dat Reuters, uložená ve formátu XML nebo data z databáze MySQL. Jedná se o konzolovou aplikaci spouštěnou z příkazového řádků, kde se nastavení načítá ze souboru XML. Nástrojem použitým na vývoj aplikace bylo populární prostředí Elipse 3.3.

## 8.1 Použité technologie

### 8.1.1 Java

Dle [21] je Java objektově orientovaný programovací jazyk vyvinutý společností Sun Microsystems uvedený v roce 1995. V roce 2007 Sun uvolnil zdrojové kódy Javy, ta bude nyní vyvíjena jako open-source. Syntaxe Javy je podobná jazyku C. Správa paměti je jednodušší než v jazyce C/C++, v Javě je realizována pomocí Garbage collectoru, který automaticky uvolňuje již nepoužívanou paměť. Java je interpretovaný jazyk, při překladu se nevytváří přímo strojový kód, ale tzv. bajtkód (mezikód). Tento kód je možné spustit na jakémkoli zařízení, které obsahuje interpret Javy, tzv. Java Virtual Machine (JVM). Nevýhodou oproti programovacím jazykům, které provádějí tzv. statickou kompilaci (např. C ++) je, že start programů psaných v Javě bývá pomalejší, protože JVM musí bajtkód nejdříve přeložit a až poté spustit. I jednoduchý program zabírá po spuštění hodně paměti, protože se musí spustit JVM.

### 8.1.2 Databáze MySQL

MySQL je podle [27] databázový systém, vytvořený švédskou firmou MySQL AB. Jeho hlavními autory jsou Michael „Monty“ Widenius a David Axmark a je k dispozici jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí. MySQL je multiplatformní databáze. Komunikace s ní probíhá – jak už název napovídá – pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními. Pro svou snadnou implementovatelnost (lze jej instalovat na Linux, MS Windows, ale i další operační systémy), výkon a především díky tomu, že se jedná o volně šiřitelný software, má vysoký podíl na v současné době používaných databázích.

### 8.1.3 XML

Značkovací jazyk XML (eXtensible Markup Language - rozšiřitelný značkovací jazyk) je obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C, jak uvádí [22]. Umožňuje

snadné vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých typů dat. Jazyk je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. Jeho výhodou je především otevřenost, striktní syntaxe, textová podoba a velká spousta volně dostupných XML parserů. Nevýhodou je pak zbytečná velikost a robustnost při ukládání informací.

## 8.2 Návrh aplikace

Činnost programu se skládá ze z několika částí odlišných od vstupních dat. V obou případech je první částí načtení konfigurace z konfiguračního souboru. O to se stará třída *ConfigurationDB* nebo *ConfigurationDoc.* Třídy se od sebe liší rozdílným nastavením, a tudíž i zpracováním vstupního konfiguračního souboru, obě však předávají načtené proměnné ostatním třídám. Snaží se odchytit i chybná nastavení již před zahájením dalších částí a upozornit na ně uživatele, aby mohl pohodlně zjednat nápravu.. V konfiguračním souboru se očekává buď nastavení přístupu k databázi nebo v případě shlukování na dokumenty nastavení pro předzpracování. V druhé části je pak nastavení pro algoritmus k-means.

Při shlukování textových dokumentů je první fáze (předzpracování) provedena v třídě *Preprocessor*, která načte XML soubory shlukovaných dokumentů a dokumenty převede na lineární vektory (objekty třídy *Termset*) následně použité ve třídě *Docvetor*. Slova dokumentů jsou porovnána oproti stoplistu a případně vynechána. Pro odsekání koncovek slov je tu třída *Stemmer*, (třída vytvořená Martinem Porterem). Pomocí třídy *ClusteringDoc* se pro dokumenty postupně vypočítá celková váha termů a objekty třídy *Docvector* se předají konstruktoru třídy *K-mean*. Kde se realizuje shlukovací úloha pomocí algoritmu k-means. K reprezentaci shluků je pak využívána třída *Cluster.*

Jestliže shlukování probíhá nad daty z databáze, je nejdříve vytvořeno připojení pomocí třídy *MySQL*. V další fázi, která je řízena třídou *ClusteringDB*, jsou načteny informace důležité pro výpočet vzdálenosti dvou řádků tabulky. Řádky (objekty třídy *RowObject*) jsou použity v konstruktoru třídy *K-meanDB*. Zde jsou opět prováděny jednotlivé iterace algoritmu k-means, rozdělující data do shluků (třída *ClusterDB*). Centrální body, vypočítávané v každé nové iteraci algoritmu, jsou díky podobnosti opět objekty třídy *RowObject*.

### 8.2.1 Konfigurace a spuštění aplikace

Při spuštění aplikace převezme z příkazové řádky přepínač rozlišující druh vstupu a jméno správného konfiguračního souboru, ve kterém jsou nastaveny parametry pro běh aplikace. Pro data kolekce Reuters je to přepínač *–reuters,* pro databázová data je to přepínač *-database*.

#### 8.2.1.1 Struktura konfiguračního souboru

Konfigurační soubor je rozdělen do dvou sekcí. V první jsou uvedeny údaje týkající se vstupních dat, v druhé části je to nastavení pro shlukování. Konfigurace se liší od vstupních dat. Soubor má pevnou strukturu a je možné nastavit tyto parametry (uvedeny vždy s příkladem nastavení):

```
<?xml version ="1.0"?>
<configuration>
 <preprocess>
      <dopreprocess>1</dopreprocess>
      <stopwordsfile>stop_words.dat</stopwordsfile>
      <outputfile>output.txt</outputfile>
      <descriptionfile>output-title.txt</descriptionfile>
      <inputpreprocess>
           <file>reut2-000.sgm</file>
      </inputpreprocess>
 </preprocess>
 <clustering>
      <doclustering>1</doclustering>
      <numberofclusters>4</numberofclusters>
      <maxiterations>55</maxiterations>
      <measure>1</measure>
      <detaildisplay>0</detaildisplay>
      <titlefile>output-title.txt</titlefile>
      <inputclusteringfile>output.txt</inputclusteringfile>
 </clustering>
</configuration>
```

Na začátku konfiguračního souboru je uvedený řádek, který definuje, že se jedná o soubor XML. Úvodní značkou je pak <*configuration*> ukončená pak tradičním způsobem XML značkou </*configuration*>. Mezi těmito značkami se pak nachází všechny ostatní značky, které mohou být uvedené v libovolném pořadí, ale musí udržovat výše uvedené schéma zanoření. Jedná se o značky:

<*preprocess*> a </*preprocess*> - značky obalující celé nastavení týkající se předzpracování . Dále již nebudeme uvádět obě části značek, pouze značky uvozující. Z definice XML dokumentů vyplývá, že každá takováto značka musí obsahovat i značku ukončovací. (například pro značku <*numberofclusters*> musí existovat značka </*numberofclusters*>, kterou již v tomto textu uvádět nebudeme). Důležitou značkou <*doprerocess*> s povolenými hodnotami 0 nebo 1, které udávají, zda se má vykonávat proces předzpracování (hodnota 1) nebo nemá (hodnota 0). Řetězec za

značkou <*stopwords*> udává název souboru, kde najdeme nepotřebná slova na odstranění. Dalším nastavením je název výstupního souboru předzpracování značkou <*outputfile*>. Výstupní soubor, který obsahuje označení dokumentu a předzpracovaná slova a je většinou hned nastaven jako vstupní soubor pro shlukování. Značka <*descriptionfile*> uvádí název souboru obsahující nadpisy dokumentů, který má podobnou strukturu jako soubor předcházející. Mezi značkami <*inputpreprocess*> a <*/inputpreprocess*> je uveden libovolný počet dvojic značek <*file*> a <*/file*>, obsahující vstupní soubor, které mají být předzpracované.Výsledkem předzpracování těchto souborů je pak jediný soubor výše uvedené struktury a s názvem uvedeným mezi značkami <*outputfile*> a <*/outputfile*>.

<*clustering*> a <*/clustering*> jsou značky, označující část, ve které se nastavují parametry shlukování respektive algoritmu k-means. Tato část je společná jak pro data z databáze, tak pro data z kolekce Reuters, avšak liší se v některých detailech. Důležitou značkou je <*doclustering*>, která je podobná značce <*doprerocess*> a uvádí, zda se má provést shlukování (hodnota 1) nebo ne (hodnota 0). Toto nastavení, může být užitečné v případě, že budeme chtít pouze předzpracovat vstupní texty. Při použití databázových dat je toto nastavení vcelku zbytečné, přesto musí být uvedeno. Mezi značky <*numberofclusters*> a <*/numberofclusters*> uvádíme množství shluků, které chceme vytvořit. Smysluplné nastavení tohoto parametru je od dvou shluků a více, proto na jiné nastavení uživatele upozorníme. Dalším nastavením je <*maxiterations*>, kde uvádím maximální počet iterací algoritmu k-means. Hodnota není příliš důležitá, protože ve většině případů algoritmus konverguje rychle a je ukončen v několika málo iteracích. Podrobnost výpisu nastavujeme pomocí značky <*detaildisplay*>. Jednoduchý výpis (hodnota 0) nám vypíše pouze základní informace a podrobnější výpis (hodnota 1) vypisuje na standardní výstup informace o vektorech dokumentů, váhy termů, přesuny ke shlukům apod. Značkou <*measure*> nastavujeme způsob určování podobností textů. V případě hodnoty 0 je použita Kosinova míra a pro hodnotu 1 je vypočtena pomocí Jaccardovy míry. Pro data z databáze nemá značka opět smysl, protože je vždy použita jedna míra napříč všemi atributy řádku tabulky. Pro shlukování dokumentů nám ještě chybí definovat jméno vstupního souboru, k tomu slouží značka <*inputclusteringfile*> a značku <*titlefile*> pro nastavení názvu souboru s titulky jednotlivých dokumentů. Nekonzistentnost těchto dvou souborů způsobí nesprávné zpracování shlukování a výpočet je předčasně ukončen. U databázového vstupu opět nejsou značky použity.

Ještě nám zbývá popsat rozdíly konfiguračního souboru pro databázová data oproti předchozímu použití pro dokumenty. Struktura je podobná, avšak přece jen se trochu liší, především v první části, ta definuje připojení na databázi pomocí značek:

<*database*> a <*/database*> mezi ně je uzavřena celá definice pro nastavení vstupu databázových dat. Následují značky v libovolném pořadí. Nastavení serveru a databáze je provedeno pomocí značky <*url*>. Řetězec uvedený mezi značkami <*table*> a <*/table*> definuje tabulku, nad kterou budeme provádět shlukování. Dalším nezbytným nastavením pro přístup k datům v databázi je uživatelské

jméno a heslo. Toto nastavení získáme z části označené <*user*>, respektive <*password*>. Posledním nastavením jsou názvy sloupců, nad kterými se má provádět shlukování. Tato konfigurace se nastavuje mezi značky <*columns*> a <*/columns*>. Jednotlivé názvy sloupců jsou pak nadefinovány označením <*/column*>. Často by však bylo pracné a zbytečné složitě vypisovat všechny sloupce, které chceme shlukovat. Proto, chceme-li spustit algoritmus nad všemi sloupci, vypíšeme mezi značky <*column*> a <*/column*> znak * (hvězdička). Tedy stejný znak jaký je znám z definice jazyka SQL pro výběr všech sloupců.

Příklad použití pro databázový vstup může vypadat následovně:

```
<configuration>
  <database>
        <url>jdbc:mysql://localhost/shluk1</url>
        <table>ostra2</table>
        <user>root</user>
        <password></password>
        <columns>
              <column>class</column>
              <column>weight</column>
              <column>aabove</column>
              <column>fontsize</column>
        </columns>
  </database>

  <clustering>
        <doclustering>1</doclustering>
        <numberofclusters>5</numberofclusters>
        <maxiterations>5</maxiterations>
        <measure>1</measure>
        <detaildisplay>0</detaildisplay>
  </clustering>
</configuration>
```

#### 8.2.1.2 Příkazová řádka

Aplikaci lze přeložit několika způsoby. Jednou z možností je použití nástroje *ant* a vygenerovaného souboru *build.xml*, případně použití celého vývojového prostředí, kde aplikace vznikla, tedy Elipse 3.3. Nejrychlejším způsobem spuštění je přímé, a to pomocí archívu JAR. Nejtradičnějším způsobem kompilace bude na příkazové řádce:

```
javac –classpath src src/Main
```

Po úspěšném zkompilování je možné aplikaci spustit příkazem

```
java –classpath src src/Main –typ nazev_konfiguračního_souboru.
```

Příklad:

```
java –classpath src src/Main –reuters configuratio-reuters.xml.
```

Pokud není nastaven konfigurační soubor, aplikace se pokusí otevřít defaultní konfigurační soubor. Pokud se toto nepodaří, činnost aplikace končí. Připomínám, že pro shlukování dat z databáze pokaždé musíme pamatovat na správné definování JDBC, neboli Java Databáze Connectivity. Tudíž správně nastavit cestu k souboru (mysql-connector-java-5.1.6-bin.jar) s ovladačem nebo jej mít uložen přímo mezi knihovnami prostředí Javy (např. jdk1.6.0_02\jre\lib\ext\).

Aplikaci lze spustit i z archivu JAR:

```
java –jar Clustering.jar –typ config.xml
```

Konkrétně tedy například:

```
java –jar Clustering.jar –reuters config-reuters.xml
```

Při práci s velkým množstvím dat je třeba spustit program s požadavkem na přidělení více operační paměti, tj.

```
java -jar Clustering.jar –Xms512M –Xmx512M –typ config.xml
```

### 8.2.2 Výstup aplikace

Aplikaci bude možno spustit ve dvou módech, jak už jsme si uvedli výše. První mód bude vypisovat podrobné informace o aktuální činnosti, např. váhy termů, průběh přesunů mezi shluky apod. Tato možnost slouží především pro kontrolu správné činnosti. Druhou možností je pouze jednoduchý výpis informací a výsledných shluků. Příklad výstupu aplikace pro shlukování databázových dat:

```
**************************************************************************
*    Program na shlukovani textovych a webovych dat algoritmem K-means   *
*                                 Rychnovsky, xrychn02 VUT FIT 2008      *
**************************************************************************
Probíhá shlukování databázových dat...
Podarilo se vytvořit tyto shluky...
<Shluk 1>
[class none, weight bold, aabove 0.0, fontsize 100.0, tlower 4215.0]
[class none, weight bold, aabove 0.0, fontsize 100.0, tlower 4215.0]
<Shluk 1/>
```

```
<Shluk 2>
[class none, weight bold, aabove 0.0, fontsize 108.0, tlower 30.0]
<Shluk 2/>
<Shluk 3>
[class none, weight normal, aabove 0.0, fontsize 87.0, tlower 0.0]
[class date, weight normal, aabove 1.0, fontsize 94.0, tlower 5.0]
[class none, weight normal, aabove 0.0, fontsize 106.0, tlower 5.0]
[class none, weight normal, aabove 0.0, fontsize 94.0, tlower 0.0]
[class none, weight normal, aabove 0.0, fontsize 94.0, tlower 0.0]
<Shluk 3/>
<Shluk 4>
[class none, weight normal, aabove 0.0, fontsize 75.0, tlower 16.0]
<Shluk 4/>
<Shluk 5>
[class none, weight bold, aabove 0.0, fontsize 55.0, tlower 9.0]
<Shluk 5/>
```

Výhodou výstupu na standardní výstup je možnost přesměrovat výstup do souboru (např. příkazem `java –jar Clustering.jar –reuters config-reuters.xml > clusters.txt`) Tento soubor je pak možné modifikovat a zpracovávat podle potřeby jinými nástroji.

## 8.2.3 Popis důležitých tříd

V této kapitole podrobněji představím implementaci tříd, které mají podstatný vliv na správnou činnost aplikace.

### 8.2.3.1 Třída Main

Hlavní třída, která zpracuje nastavení příkazové řádky. Spouští podle nastavení buď zpracování dokumentů a shlukování dokumentů nebo shlukování databázových dat. Na závěr zobrazuje výsledky.

### 8.2.3.2 Třída ConfigurationDoc a ConfigurationDB

Třídy sloužící na zpracování XML dokumentů s konfigurací, která byl zadaná jako parametr vstupu. Nejdůležitější metodou je metoda ParseXML(), která přečte a interpretuje nastavení do jednotlivých proměnných, jimiž se řídí následně celá aplikace.

### 8.2.3.3 Třída Preprocesor

Tato třída slouží na předzpracování kolekce dat Reuters-21578. Nejdůležitější metodou je preprocessSGML(), která provádí funkci parteru SGML dokumentů. Tělo dokumentu je rozděleno na jednotlivá slova. Každé slovo projde úpravou, kde je převedeno na malá písmena a jsou z něj odstraněny všechny znaky, které nejsou písmeny (např. čárky tečky, pomlčky, …). Následuje odstranění případné koncovky třídou `Stemmer`. Na závěr je slovo porovnáno proti seznamu nevýznamových slov a v případě shody vynecháno. Výstupem jsou dva soubory, v jednom jsou termy jednotlivých dokumentů a v druhém nadpisy jednotlivých článků.

#### 8.2.3.4 Třída Stemmer

Jedná se o upravenou verzi implementace originálního Porterova stemovacího algoritmu. Aplikace pravidel je upravená tak, aby algoritmus nebyl při odstraňování koncovek tak agresivní jako v původní implementaci. Snahou je pokrytí co největší škály možný typů slov.

#### 8.2.3.5 Třída ClusteringDoc

Třída zahrnuje metody pro přípravu objektů *Docvector* a celkově řídí shlukování nad dokumenty. Nejdříve využije metody pro výpočet TF, NTF a IDF vah termů. Následně určí TF-IDF váhy termů a pomocí těchto vektorů vytvoří instance objektu *Docvector*, které pak předává algoritmu k-means.

#### 8.2.3.6 Třída Docvector

Obsahuje nadpis článku a vektor termů. Důležité jsou různé statické metody na určování míry podobnosti, které pak obaluje metoda *distance(Docvector dv1, Dovector dv2, int typ),* ta tyto míry převádí na vzdálenost dvou dokumentů.

#### 8.2.3.7 Třída Kmen a KmeanDB

Realizují samotný algoritmus k-means pomocí metody *runKMeans()*. Další důležitou metodou je *update*(), která se stará o výpočet nových centroidů. Důležitá je i metoda *assignToCluster(Docvector dv)*, které vyhledá nejbližší shluk a k němu pak následně dokument přiřadí.

#### 8.2.3.8 Třída ClusteringDB

Třída zahrnuje metody pro přípravu objektů *RowObject* a dále se stará o řízení shlukování nad databázovými daty. Nejdříve pomocí třídy *MySQL* vytvoří spojení na databázi. Poté pomocí několika metod sestaví SQL dotazy, na základě konfigurace vytvořené třídou *ConfigurationDB*. Těmito dotazy jednak získá nejen samotná data, ale i některé informace, jejichž výpočet by mohl zbytečně zdržovat samotné shlukování. Jde o celkové maximum, minimum, typ apod. To vše zajišťuje metoda readDB(), z které se volají ostatní. Poté vytvoří instance objektu *RowObject*, které pak předává algoritmu k-means

#### 8.2.3.9 Třída RowObject

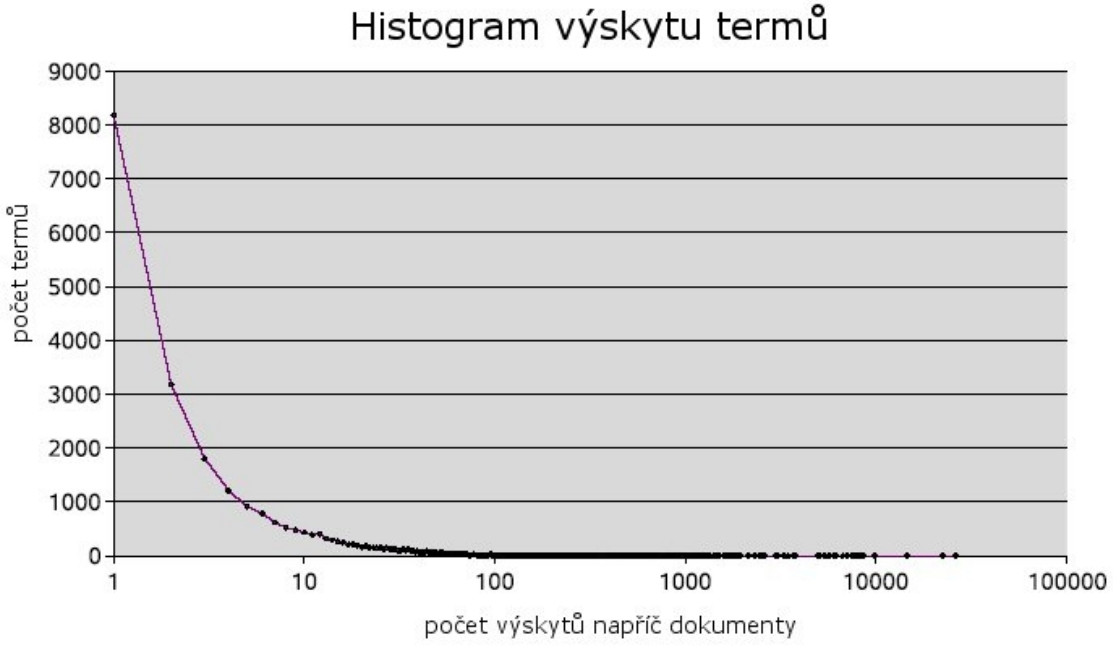
Ukládá informaci o čísle řádku a vektor atributů daného řádku. Důležitá je statická metoda na určování vzdálenosti *distanceRow(RowObject row1, RowObject row2),* která vrací vzdálenost dvou řádků.

# 9 Testování a experimenty

## 9.1 Textová data

### 9.1.1 Redukce slov a stoplisty

Potvrdilo se, že většina termů se vyskytuje pouze v deseti a méně dokumentech, jak ukazuje obrázek 9.1. Tyto termy pak při shlukování ztrácí význam a je možné je vynechat. Pro jejich odstranění bylo použito několik stoplistů. V příloze Příloha 1 je uveden seznam použitých stoplistů pro testování. Jde o seznamy slov použité v různých databázích nebo v projektech zabývajících se zpracováním textu. Nejkratší z nich má 36 slov, nejdelší jich obsahuje 535. Seznamy by měli sloužit pro zlepšení kvality shlukování a urychlení výpočtu odstraněním zbytečných slov. Provedené testy však prokázaly, že odstraněním jakýchkoli slov se snižuje kvalita shluků. Nejlepším řešením by bylo asi vytvoření stoplistu přímo pro konkrétní data, avšak tato tvorba není jednoduchou záležitostí.




**Obrázek 9.1** Histogram výskytu termů. Je vidět, že většina termů se vyskytuje v 10 a méně dokumentech

### 9.1.2 Váhování termů

V kapitole 6.1.4 byly uvedeny možnosti přiřazení vah slovům, která se v dokumentu vyskytují více než jedenkrát. Pro zvýšení kvality shlukování bylo toto výše uvedené váhování použito. Experimenty potvrdily, že normalizované váhování jednotlivých termů je vhodnější. Nezvýhodňuje delší dokumenty, tak jako nenormalizované.

### 9.1.3 Podobnostní míry

Při zkoumání rozdílu použití na dokumentech míry Kosinové, Jaccardové a obyčejného vektorového součinu nebyly zjištěny velké rozdíly. Odlišné výsledky dával vektorový součin. Rozdíl mezi použitím Kosinové či Jaccardové míry byl téměř minimální a na vytváření shluků neměl zásadní vliv.

### 9.1.4 Špatné úvodní centrální body

O důležitosti úvodního vyčíslení centrálních bodů, jsme si již pověděli. Pokusíme se tyto teze ukázat na konkrétním případu konkrétních dat. Mějme tyto dokumenty:

```
D0 - Nadpis 1 [slovo1 - 1.78, slovo2 - 1.48]
D1 - Nadpis 2 [kostka - 1.0, láhev - 1.48, míč - 1.48]
D2 - Nadpis 3 [slovo1 - 1.78, slovo2 - 1.48]
D3 - Nadpis 4 [kostka - 1.0, láhev - 1.48, míč - 1.48]
D4 - Nadpis 5 [kostka - 1.0, láhev - 1.48, míč - 1.48]
D5 - Pokusný nadpis je fajn ho tu mít, že? [košile - 1.48]
```

Potom jeden z výstupů může být:

```
Úvodní nástřel:
Shluk 1
D2 - Nadpis 3 [slovo1 - 1.78, slovo2 - 1.48]
Shluk 2
D5 - Pokusný nadpis je fajn ho tu mít, že? [košile - 1.48]
Shluk 3
D4 - Nadpis 5 [láhev - 1.48, kostka - 1.0, míč - 1.48]
Výstup:
Shluk 1
[D0 - Nadpis 1 [slovo1 - 1.78, slovo2 - 1.48], D2 - Nadpis 3
[slovo1 - 1.78, slovo2 - 1.48]]
Shluk 2
[D5 - Pokusný nadpis je fajn ho tu mít, že? [košile - 1.48]]
Shluk 3
```

```
[D1 - Nadpis 2 [kostka - 1.0, láhev - 1.48, míč - 1.48], D3 -
Nadpis 4 [kostka - 1.0, láhev - 1.48, míč - 1.48], D4 - Nadpis
5 [kostka - 1.0, láhev - 1.48, míč - 1.48]]
```

A nebo také:

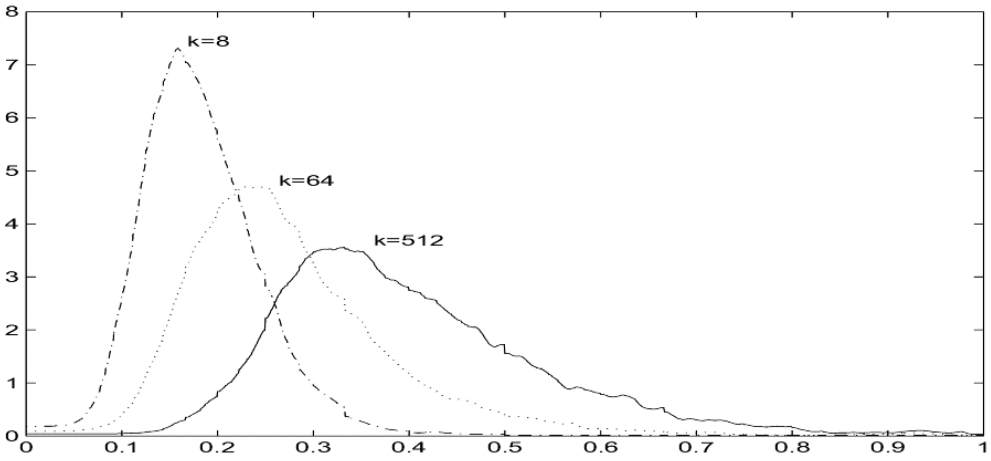
```
Úvodní nástřel:
Shluk 1
D5 - Pokusný nadpis je fajn ho tu mít, že? [košile - 1.48]
Shluk 2
D1 - Nadpis 2 [láhev - 1.48, kostka - 1.0, míč - 1.48]
Shluk 3
D3 - Nadpis 4 [láhev - 1.48, kostka - 1.0, míč - 1.48]

Výstup:
Shluk 1
[D0 - Nadpis 1 [slovo1 - 1.78, slovo2 - 1.48], D2 - Nadpis 3
[slovo1 - 1.78, slovo2 - 1.48], D5 - Pokusný nadpis je fajn ho
tu mít, že? [košile - 1.48]]
Shluk 2
[D1 - Nadpis 2 [kostka - 1.0, láhev - 1.48, míč - 1.48], D3 -
Nadpis 4 [kostka - 1.0, láhev - 1.48, míč - 1.48], D4 - Nadpis
5 [kostka - 1.0, láhev - 1.48, míč - 1.48]]
Shluk 3
[]
```

V prvním případě se úvodní „nástřel“ centrálních bodů povedl a výsledek shlukování je správný. V druhém případě jsou bohužel zvoleny dva stejné dokumenty a výsledek je potom špatný. Vektory termů při porovnání byly i díky své řídkosti již navzájem kolmé, a tudíž neproběhly správné přesuny ke shlukům a jeden shluk zůstal nesprávně prázdný. Ideálními centrálními body jsou dokumenty od sebe co nejvíce vzdálené, což má ale být výsledkem samotného shlukování, tudíž musíme hledat takové způsoby a heuristiky, které jsou nejvhodnější pro konkrétní data.

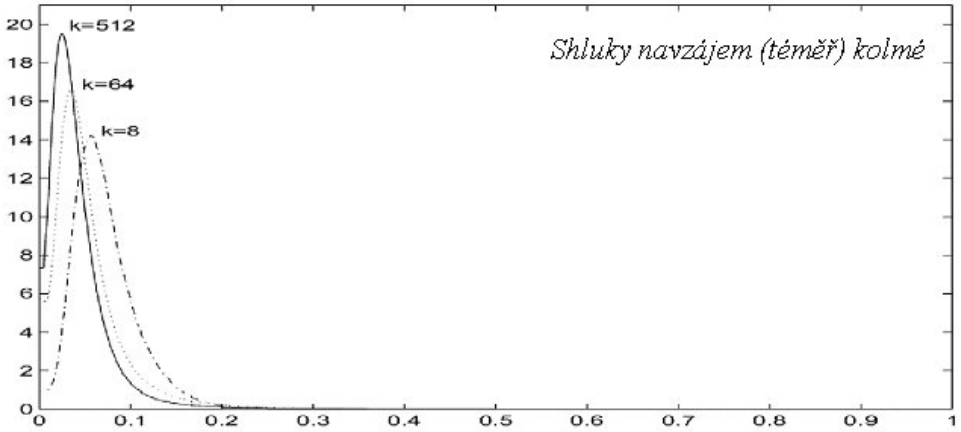
### 9.1.5 Vzájemná kolmost shluků

V předcházející kapitolce jsme si ukázaly problémy s kolmostí vektorů při shlukování dokumentů. Řídkost a vysoká dimensionalita pak může způsobovat tyto problémy s určováním podobností dvou dokumentů. Je nemožné správně přiřazovat podle vektorů dokumenty ke shlukům reprezentovaných jiným vektorem. Pokud totiž nejsou některé termy obsaženy v obou porovnávaných vektorech, je složité interpretovat podobnost těchto dokumentů. Z toho plyne i vzájemná kolmost shluků získaných

k-mean algoritmem. To znamená, že termy důležité pro identifikaci jednoho centra jsou (téměř) nezajímavé pro identifikaci ostatních center. Interpretace kolmosti je patrná z obrázků 9.1 a 9.2.




**Obrázek 9.1** Rozložení podobností dokumentů stejných shluků Převzato[11]




**Obrázek 9.2** Rozložení podobností dokumentů stejných shluků Převzato[11]

# 9.2 Databázová data

Databázová data byly o něco vhodnější pro různé experimenty, protože jejich nastavení a následné pozorování je přece jen podstatně jednodušší a rychlejší. Nad těmito daty je poměrně velký problém určit počet shluků. S příliš podobnými daty si algoritmus neporadil dobře. Měl tendenci konvergovat až příliš rychle a tím seskupit data do opravdu mála počtu shluků, které pak nebyly příliš kvalitní. Kvalitu shluku ovlivnily i příliš odlehlé hodnoty, více jak cca 34% atributů. Tyto řádky pak měly v závislosti na počtu shluků vytvořit samostatné shluky, což je z jednoho úhlu pohledu správně, avšak z druhého nám výrazně snižují kvalitu ostatních shluků. Ukažme si některé fakta na příkladu:

<Shluk 1>
[class h2, weight normal, aabove 0.0, fontsize 162.0, tlower 13.0, contrast 21.0]
[class h2, weight normal, aabove 0.0, fontsize 162.0, tlower 17.0, contrast 21.0]
[class h2, weight normal, aabove 0.0, fontsize 162.0, tlower 25.0, contrast 21.0]
<Shluk 1/>
<Shluk 2>
[class none, weight normal, aabove 0.0, fontsize 106.0, tlower 5.0, contrast 5.73118]
[class pokus, weight normal, aabove 0.0, fontsize 99.0, tlower 0.0, contrast 21.0]
[class none, weight normal, aabove 0.0, fontsize 94.0, tlower 0.0, contrast 1.73031]
[class aktualita, weight normal, aabove 2.0, fontsize 94.0, tlower 10.0, contrast 8.20708]
[class aktualita, weight normal, aabove 1.0, fontsize 94.0, tlower 12.0, contrast 8.20708]
[class aktualita, weight normal, aabove 0.0, fontsize 94.0, tlower 12.0, contrast 8.20708]
<Shluk 2/>
<Shluk 3>
[class none, weight bold, aabove 0.0, fontsize 100.0, tlower 4215.0, contrast 7.47491]
[class none, weight bold, aabove 0.0, fontsize 100.0, tlower 4215.0, contrast 18.0]
<Shluk 3/>

Prvně vytvořená shluková analýza se zdá být co do pohledu na kvalitu shluku intuitivně dobrá. Změnou atributů řádku s hodnotou atributu třída *pokus* se však sníží (řádek „pokus“ se osamostatní):

<Shluk 1>
[class none, weight bold, aabove 0.0, fontsize 100.0, tlower 4215.0, contrast 7.47491]
[class none, weight bold, aabove 0.0, fontsize 100.0, tlower 4215.0, contrast 18.0]
<Shluk 1/>
<Shluk 2>
[class none, weight normal, aabove 0.0, fontsize 106.0, tlower 5.0, contrast 5.73118]
[class none, weight normal, aabove 0.0, fontsize 94.0, tlower 0.0, contrast 1.73031]
[class h2, weight normal, aabove 0.0, fontsize 162.0, tlower 13.0, contrast 21.0]
[class h2, weight normal, aabove 0.0, fontsize 162.0, tlower 17.0, contrast 21.0]
[class h2, weight normal, aabove 0.0, fontsize 162.0, tlower 25.0, contrast 21.0]
[class aktualita, weight normal, aabove 2.0, fontsize 94.0, tlower 10.0, contrast 8.20708]
[class aktualita, weight normal, aabove 1.0, fontsize 94.0, tlower 12.0, contrast 8.20708]
[class aktualita, weight normal, aabove 0.0, fontsize 94.0, tlower 12.0, contrast 8.20708]
<Shluk 2/>
<Shluk 3>
[class pokus, weight normal, aabove 800.0, fontsize 800.0, tlower 800.0, contrast 21.0]
<Shluk 3/>

Nově osamocený řádek s třídou *pokus* nám způsobí přidání všech řádků s hodnotou atributu třída *h2* do nesamostatného shluku, což pro nás zřejmě není intuitivně dobré řešení. Zároveň to ukazuje důležitost volby správného počtu shluků vzhledem k datům. I u databázových dat se objevilo nekvalitní shlukování po nedobrém úvodním „nástřelu“, což můžeme s největší pravděpodobností označit za největší slabinu algoritmu k-means.

# 10 Závěr

V této práci jsem se snažil popsat způsoby a postupy při shlukování dokumentů a dat, která jsou získána z webových stránek. V první části se čtenář seznámil v obecné rovině se získáváním znalostí a především se základními myšlenkami různých typů dolovacích úloh. V další části práce jsem se zabýval dolováním dat, které je zaměřeno na web. Ten jako obrovská heterogenní databáze skýtá velký potenciál právě pro dolování, a to z pohledu jeho užívání, struktury či obsahu.

Pak už následují kapitoly věnované předzpracování textových dat pro následné shlukování. Texty jsou nejdříve pomocí jednoduchých pravidel tokenizovány (rozsekány na slova). Každé slovo je pak term a tvoří jeden rozměr váhového vektoru. Vysoký počet takto vzniklých dimenzí je snížen pomocí stoplistu, který odstraní slova s konstantním rozložením ve všech dokumentech.

Dále jsou popsány různé shlukovací přístupy a metody. Pro implementaci byl vybrán algoritmus k-means. A ten pak realizoval shlukování dvou různých typů dat. Dat textových a dat, která jsou extrahovaná z webového prostředí a uložena v databázi.

Výsledky implementovaného algoritmu byly prověřeny nad různými daty a potvrdily výhody i nevýhody tohoto algoritmu. Algoritmus k-means v nejjednodušší podobě se neukázal jako příliš vhodný pro shlukování databázových dat. Zde se ukazuje největší problém, a tím je úvodní „nástřel“ (náhodný výběr centrálního bodu ze shlukovaných). Nepříjemnosti mu způsobovala i data, která byla málo rozdílná (lišící se pouze několika atributy z mnoha). V takovém případě měl snahu shlukovat vše do velmi malého počtu shluků. S textovými daty se vypořádal o poznání lépe, a pokud byl vhodně zvolen počet shluků a dokumenty nebyly příliš řídké, prokazoval dobré výsledky. To nám potvrzuje, že k-means je vhodné použít, pokud nám stačí malý počet shluků, a je-li předpoklad, že takové shluky v datech opravdu existují .

Tato základní verze programu by mohla být vylepšena o implementaci snížení dimensionality textových dat, čímž by se celé zpracování výrazně zrychlilo. Podle experimentů funguje totiž k-means algoritmus efektivně zhruba do šestnácti proměnných a v případě dokumentů pracujeme s počtem dimenzí, které se pohybují v desítkách a stovkách tisíc. Dobrým rozšířením by byla i implementace grafického uživatelského prostředí a volba typu proměnných pro shlukování databázových dat, čímž by byla zajištěna správná interpretace různých typů dat. K dosažení lepších shluků by bylo vhodné upravit algoritmus k-means, tak aby se s několika různými úvodními nástřely opakoval. Poté by byly vypočteny kvality všech shluků pro jednotlivá opakování a následně vybráno nejlepší řešení.

# Literatura


[1] Berka, P.: *Dobývání znalostí z databází*, Academica, Praha, 2003

[2] Zendulka, J.: *Získávání znalostí z databází*, Brno: VUT Brno, 31.10. 2006, s. 160.

[3] Pyle, D.: *Data preparation for data mining*, Morgan Kaufmann publishers, 1999, ISBN 1-55860-529-0, s. 540.

[4] Pospíšil, J., Nemarava, M.: *Dolování dat a jeho aplikace*, 2006

[5] Greening, D.: *Data Mining on the web*, [online] Dokument dostupný na URL <http://www.webtechniques.com/archives/2000/01/greening/> (leden 2008)

[6] Uchytil, S.: *Dolování dat na www*, [online] Dokument dostupný na URL <http://www.fit.vutbr.cz/study/courses/ZZD/public/seminar0405/uchytil.pdf /> (leden 2008)

[7] Kay, R.: *Webová sklizeň nepřehledných dat*, Computerworld 36, Praha, 2004

[8] Kelbl, J.: Šilhán, D., *Shluková analýza*, [online] Dokument dostupný na URL <http://gerstner.felk.cvut.cz/biolab/X33BMI/slides/KMeans.pdf > (duben 2008)

[9] Bártík, V.: *Dolování z textu a na webu*, [online] Dokument dostupný na URL <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ZZN-T/lectures/10%20TextWeb.pdf> (duben 2008)

[10] Snášel, V: *Jednotná teorie vyhledávacích problémů a její kvantové aspekty*, VŠB-TU Ostrava, 2005, [online] Dokument dostupný na URL <www.datakon.cz/datakon03/d03_tut_pokorny.pdf> (květen 2008)

[11] Kopecký, M.: *Dokumentografické Informační Systémy*, KSI MFF UK, 2005,[online] Dokument dostupný na URL <www.ms.mff.cuni.cz/~kopecky/vyuka/dis/disslide.ppt> (květen 2008)

[12] *Wikipedia Stemming*, [online] Dokument dostupný na URL <http://en.wikipedia.org/wiki/Stemming> (květen 2008)

[13] Porter M.: *An algorithm for suffix stripping*, 1980, [online] Dokument dostupný na URL <http://maya.cs.depaul.edu/~classes/ds575/papers/porter-algorithm.html> (květen 2008)

[14] *The Porter Stemming Algorithm*, 2006, [online] Dokument dostupný na URL <http://tartarus.org/~martin/PorterStemmer/index.html> (květen 2008)

[15] *What is Porter Stemming?*, [online] Dokument dostupný na URL <http://www.comp.lancs.ac.uk/computing/research/stemming/general/porter.htm> (květen 2008)

[16] Fuller M., Zobel J.:*Conflation-based Comparison of Stemming algorithms, Department of Computer Science*, Dokument dostupný na URL <www.cs.rmit.edu.au/~jz/fulltext/adcs98.ps>

(Květen2008)

[17] *Wikipedia Stop words,*[online] Dokument dostupný na URL <http://en.wikipedia.org/wiki/Stop_words> (květen 2008)

[18] Luz, S., Emms, M.: *Machine Learning for Natural Language Processing*, ,[online] Dokument dostupný na URL <http://ronaldo.cs.tcd.ie/esslli07/>(Květen2008)

[19] Masarykova univerzita, Laboratoř zpracování přirozeného jazyka, *Český stoplist*, [online], Dokument dostupný na URL <http://nlp.fi.muni.cz/nlp/aisa/NlpCz/Cesky_stoplist.html> (Květen2008)

[20] Cambridge University Press, 2008, *K-means*, [online], Dokument dostupný na URL <http://nlp.stanford.edu/IR-book/html/htmledition/k-means-1.html> (Květen2008)

[21] *Wikipedia Java*, [online] Dokument dostupný na URL <http://cs.wikipedia.org/wiki/Java> (květen 2008)

[22] *Wikipedia XML*, [online] Dokument dostupný na URL <http://cs.wikipedia.org/wiki/XML> (květen 2008)

[23] Han, J.: *Effective and effective Clustering Methods for Spatial Data Mining,* University of British Columbia, 1994

[24] Popelínský, L.: *Strojové učení a přirozený* jazyk, Masarykova univerzita, Brno, [online] Dokument dostupný na URL <http://www.fi.muni.cz/~popel/nll/tut_znalosti03.ppt>

[25] Žáčková, E.: *Parciální syntaktická* analýza, Disertační práce, Masarykova univerzita, Brno, 2002

[26] *Wikipedia Tezaurus*, [online] Dokument dostupný na URL <http://cs.wikipedia.org/wiki/Tezaurus> (květen 2008)

[27] *Wikipedia MySQL*, [online] Dokument dostupný na URL <http://cs.wikipedia.org/wiki/MySQL> (květen 2008)

# Seznam příloh

# Příloha 1. Přehled stoplistů

1. **ranks-stopword.txt**

   36 slov zdroj: http://www.ranks.nl/tools/stopwords.html

2. **stopwords.txt**

   319 slov zdroj: http://ronaldo.cs.tcd.ie/esslli07/

3. **ranks-stopword.txt**

   535 slov zdroj**:**http://www.sugarcrm.com/wiki/index.php?title=Overview_of_Full_Text_Stop_Words

4. **stoplist.txt**

   66 slov
   zdroj: http://www.csc.kth.se/~xmartin/java/