Mendelova univerzita v Brně
Provozně ekonomická fakulta

Aplikace pro získávání názorů z uživatelských recenzí

Diplomová práce

Vedoucí práce:

Ing. Jan Přichystal, Ph.D. **Bc. Jiří Švec**

Brno 2017

Rád bych poděkoval Ing. Janu Přichystalovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích diplomové práce. Mé poděkování patří též rodině, přítelkyni i přátelům za trpělivost a podporu ve vzdělávání.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Aplikace pro získávání názorů z uživatelských recenzí**
vypracoval/a samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

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

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

V Brně dne 4. ledna 2017 _____

Abstract

Švec, J. Application for customers' reviews opinion mining. Diploma thesis. Brno: Mendel University, 2017.
This paper describes some current methods for customer's opinion mining and text processing. Application architecture is created along with database structure. Based on this architecture, new application is implemented. Emphasis is placed on automated downloading and processing of customers' reviews with opinion extraction and user-friendly presentation. Part of this paper is dedicated to optimizing the application for better efficiency and speed boost.

Keywords

Opinion mining, review, text processing, web scraping, word cloud, configuration.

Abstrakt

Švec, J. Aplikace pro získávání názorů z uživatelských recenzí. Diplomová práce. Brno: Mendelova univerzita v Brně, 2017.
V práci jsou probírány některé současné metody v oblasti získávání názorů a zpracování textu. Je realizován návrh architektury aplikace společně se strukturou databáze. Posléze je provedena implementace softwarové aplikace. Důraz se klade na automatizované stahování recenzí od běžných spotřebitelů, zpracování recenzí, extrakci významných názorů a prezentaci uživateli. Část práce se věnuje optimalizaci aplikace pro zvýšení její efektivity a rychlosti.

Klíčová slova

Získávání názorů, recenze, zpracování textu, web scraping, word cloud, konfigurace.

Obsah

1 Úvod a cíl práce

1.1 Úvod

Od svého vzniku prošel World Wide Web podstatnou změnou. Zvrat nastal zejména v momentě, kdy se běžní uživatelé začali připojovat k síti. Vznik nových technologií umožnil posílit počítače i sítě. Neustálý pokles provozních nákladů webových stránek i samotných cen hardwaru vedl k masové polarizaci internetu.

Následoval rozvoj vývojových nástrojů, aplikací a různorodých frameworků. Na jejich základě bylo umožněno vytvářet ve velmi krátké době další webové stránky, které umožňovaly aktivní účast nejen jednotlivým uživatelům, ale i celým komunitám. Neustálý růst komunit byl ovlivněn zejména prostřednictvím možných příspěvků každého člena. Vytvořenými příspěvky došlo k obohacení celkové informace v dané oblasti. Zmíněná doba bývá označována jako Web 2.0 (Carey, 2006).

V současné době informačních a komunikačních technologií se stalo běžnou skutečností, že si zákazníci nevytváří názor na produkt pouze na základě komunikace s přáteli či zdlouhavým čtením recenzí z odborných časopisů. Čím dál více začínají využívat názorů od jiných zákazníků, kteří mají s daným produktem určité zkušenosti (Přichystal, 2016). Je podstatné zdůraznit, že přínos ve využití jiných názorů či recenzí pramení z jejich neutrální povahy. Obvykle nebývají produkovány organizací ani společností. Představují hlas běžných spotřebitelů, čímž se značně liší od reklam. Reklamy bývají ve velké většině zkreslené. Mají sklon upřednostnit určitý produkt, zdůraznit pozitivní aspekty a skrýt negativní, čímž klamou zákazníka.

Převážná většina recenzí je v podobě textových dokumentů. Jsou k dispozici napříč mnoha oblastmi obchodu. Pokrývají elektroniku, zdravotnictví, stavebnictví, stravování, módu a mnoho dalších. Zdroje potenciálně významných informací jsou k dispozici na webových stránkách některých prodejců, na sociálních sítích, v blozích či v diskuzních fórech. Na základě zjištěných informací si zákazník může udělat reálnější pohled na daný produkt, popřípadě mu vypomoci učinit správné rozhodnutí (Přichystal, 2016). Recenze mohou být nápomocné v situacích, kdy si samotný zákazník vybírá z řady několika produktů a neví, pro který se rozhodnout.

Pro pořízení informací o požadovaném produktu mohou zákazníci využít jejich mobilní zařízení a vyhledat hodnocení na internetu. Problémem se však stává sběr a rozbor nalezených recenzí, které mohu být i z více zdrojů. Výsledná datová množina se může stát velmi rozsáhlou. Není v silách člověka zpracovat množinu nalezených recenzí ručně v přijatelném čase (Přichystal, 2016). Oproti tomu je výpočetní technika přímo určena pro zpracování velkého množství dat. Nevýhodou počítačů je, že nedokáží porozumět významu ani sentimentu textových recenzí psaných v přirozeném jazyce. Navzdory těmto problémům, je možné využít výpočetní zdroje k analýze a odhalení skrytých informací. Nejlépe se zmíněná skutečnost demonstruje v situacích, kde je k dispozici větší množství recenzí pro konkrétní produkt. Následně lze zákazníkovi poskytnou užitečné informace, na základě kterých může učinit snazší rozhodnutí.

1.2 Cíl práce

Vytvořit graficky přehlednou, konfigurovatelnou, maximálně rozšiřitelnou aplikaci, s dostatečně rychlou odezvou a reprezentativním zobrazením výsledků pro uživatele. Prostřednictvím této aplikace uživatel získá přehled o nejčastěji probíraných tématech v recenzích zvoleného produktu. Recenze budou pořizovány a zpracovávány automatizovaným způsobem. Součástí práce je návrh i realizace databáze pro uložení zpracovaných dat.

2 Metodika

Nejvíce vyhovující bude agilní metodika extrémního programování. Hlavním důvodem bude neustálá změna dílčích požadavků a jejich následné testování. Druhotným důvodem může být fakt, že se ve zvolené metodice zbytečně nevytváří již vytvořené a je prioritou dodávat co v nejkratší době funkční řešení.

Autor práce bude spolu s vedoucím zároveň i zákazníkem. Dle jejich připomínek budou provedeny úpravy v aplikaci. Zvolený přístup urychlí dobu vývoje oproti zapojení třetích stran.

Práce bude obsahovat následující kroky, které budou využity v praktické části. Nejprve bude zapotřebí nastudovat problematiku zaměřenou převážně na zpracování textu. Posléze budou vytvořeny vhodné podmínky pro vývoj a testování aplikace. Následně bude vykonána analýza a návrh současné problematiky. Dále proběhne samotný vývoj střídaný s testováním. Výsledný software bude zoptimalizován a zdokumentován pro případné budoucí navazující práce.

2.1 Vytvoření podmínek

Bude zapotřebí vytvořit vhodné prostředí pro vývoj a testování aplikace. V rámci vývoje a prvotního testování bude instalován virtuální operační systém z důvodu lepší přenositelnosti. V prostředí bude instalován textový editor, databáze a příslušné knihovny.

V pokročilejší fázi vývoje bude aplikace nasazena a testována na vyhrazeném serveru o vyšším výkonu.

2.2 Návrh

Důraz bude kladen na kvalitativní návrh s širšími možnostmi rozšiřitelnosti, jako je snadné integrování dalšího datového zdroje nebo změna databáze. Budou vytvořeny názorné diagramy popisující zmíněné oblasti. Aplikace bude částečně i konfigurovatelná.

2.3 Implementace

Bude probíhat v jazyce Perl 5 s využitím objektově orientovaného přístupu a externích knihoven.

Prvním krokem bude zajistit automatizovaný sběr dat z veřejně dostupného zdroje. Výchozím zdrojem bude server Heureka.cz. Data budou obsahovat jednotlivé pozitivní i negativní recenze. Budou zpracovány pomocí metod zaměřených na zpracování textu.

Jelikož je požadavkem rychlá odezva, bude nezbytné navrhnout a zrealizovat databázi pro zpracované uživatelské recenze. Uživatel aplikace bude následně operovat pouze s již zpracovanými daty. Samozřejmostí bude aktualizační program, který bude obstarávat nové recenze.

Prostřednictvím kamery mobilního telefonu bude identifikován konkrétní produkt na základě čárového kódu. Posledním krokem bude grafické zobrazení zpracovaných recenzí.

2.4 Testování

Testování bude probíhat v krátkých časových intervalech vždy po dokončení zadané úlohy. Střední doba mezi jednotlivými testy bude tři dny. V reakci na zákazníkovu spokojenost s danou úpravou bude zadaná funkcionalita reimplementována, popřípadě bude pozměněno zadání konkrétního požadavku. Testy budou vykonány na malých datových vzorcích pro snazší vyhodnocení požadovaných výsledků.

2.5 Optimalizace

Části kódu, které budou dosahovat vysoké časové složitosti, budou optimalizovány prostřednictvím technik paralelních algoritmů, pokud to bude možné. Paměťová náročnost bude redukována pomocí minimalizace uchovávání dat v operační paměti počítače.

2.6 Dokumentace

Každá netriviální část aplikace bude obsahovat příslušný komentář. Obsahem komentáře bude definice vstupu, výstupu a účel dané funkcionality. Dle komentáře bude pro následného čtenáře mnohem jednodušší pochopit autorovu myšlenku.

3 Současný stav

Kapitola představí stěžejní pojmy používané nejen v současné práci. Dále budou uvedeny základní technologie a nástroje využívané v oblasti zpracování textu. Cílem kapitoly je, aby čtenář získal obecný přehled nad současnými možnostmi v rámci zpracování uživatelských recenzí a dokázal se v dokumentu dále orientovat.

3.1 Důležité pojmy

Velmi podstatná kapitola, která představí klíčové pojmy, které jsou dále využívány v práci. Autor se pokusil o vzestupné seřazení pojmů dle jejich náročnosti od jednoduchých po složitější. Od čtenáře se očekáváná alespoň základní znalost vývoje webových stránek.

3.1.1 **Recenze**

Pojednává o subjektivním kritickém posudku jednoho či několika lidí vzhledem k určitému předmětu, např. k filmu, knize, jídle atp. Z pravidla se vyskytují v časopisech, v diskuzních fórech, na sociálních sítích, popřípadě přímo na webové stránce prodejce (Cambridge dictionary, 2016a). Autor recenze je nazýván recenzent.

3.1.2 **Mobilní zařízení**

Výpočetní zařízení, zpravidla dostatečně malé, aby mohlo být přenositelné. To znamená rozměrové a hmotnostní omezení (Cambridge dictionary, 2016b). Každé mobilní zařízení obsahuje displej pro zobrazení informací a klávesnici pro manipulaci. V současné době jsou mechanické klávesnice téměř vytlačeny virtuálními klávesnicemi, které jsou promítány přímo na displej.

3.1.3 **Čárový kód**

Je zakódovaná sekvence znaků do grafické reprezentace. Jedná se o symbol, který obsahuje určité množství černých pruhů a mezilehlých mezer. Před i za samotným kódem musí být tzv. *klidová zóna*. Jinými slovy jde o prázdné místo určité šířky bez potisku. Celý kód začíná start znakem. Následují vlastní data někdy doplněná o kontrolní součet. Výsledná sekvence je ukončena stop znakem (Gaben, 2016).

V jednotlivých kódech se nachází různý počet čar a mezer s odlišnou šířkou. Termín zastřešující konfiguraci zmíněnou v předchozí větě se nazývá *symbolika*. Dle této symboliky lze následně identifikovat různorodé typy čárových kódů. Mezi nejčastěji používaná symbolika patří např. UPC-A, EAN 13, EAN 8 (Gaben, 2016). Zmíněné varianty budou nastíněny níže.

Hlavní výhodou a tedy i důvodem použití čárových kódů je jejich přesnost a rychlost zadávání v poměru s klávesnicovým vstupem.

3.1.3.1 **UPC-A**

Universal Product Code byl poprvé používán v obchodních řetězcích. Jeho vzniku předcházel požadavek jednoznačné identifikace výrobku a jeho výrobce. Má pevnou délku. UPC verze A se používá k zakódování 12ti místného čísla. První číslice je znak systému číslování. Dalších 5 je identifikační číslo výrobce. Následujících 5 je číslo výrobku a poslední číslice je kontrolní znak (Gaben, 2016). Na obrázku 1 je uveden ukázkový čárový kód UPC verze A.

Obr. 1 Ukázka kódu UPC-A, (Gaben, 2016)

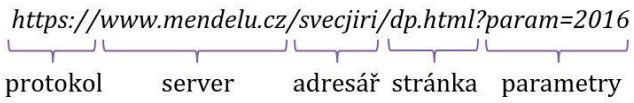
3.1.3.2 **EAN 13, EAN 8**

European Article Numbering je nadstavbou symboliky UPC. EAN Obsahuje dvě verze EAN 13 a EAN 8. Obě jsou pevné délky. EAN 8 vznikl z důvodu označování malých produktů. Z názvu může být patrné, že EAN 13 kóduje 13 číslic a EAN 8 kóduje 8 číslic. Následující charakteristika je určena pro EAN 13, jelikož je nejčastěji využíván. První dvě nebo tři číslice vždy určují stát původu, např. ČR má číslo 859. Několik dalších číslic, obvykle čtyři až šest určují výrobce. Zbývající číslice kromě poslední určují konkrétní zboží. Poslední číslice je kontrolní. Je vyhrazena pro ověření správnosti dekódování (Kodys, 2009). Na obrázku 2 jsou prezentovány čárové kódy. Vlevo EAN 13, vpravo EAN 8.

Obr. 2 Vlevo EAN 13, vpravo EAN 8, (Gaben, 2016)

3.1.4 **Webová stránka**

Je textový dokument, který lze prostřednictvím webového prohlížeče zobrazit. Jazyk HTML byl ustanoven pro tvorbu webových stránek. Každá stránka má jedinečný odkaz či název, označený jako URL. Součástí URL je protokol, který definuje způsob přístupu k webové stránce. Dále název serveru, za kterým je obvykle i adresář, eventuálně podadresáře. Celá URL je ukončena samotným názvem stránky s případnými parametry (Crowder a Crowder, 2001) viz obrázek 3 Struktura URL.



Obr. 3 Struktura URL

Existuje několik pohledů, jak webové stránky kategorizovat. Základní dvě rozdělení mohou být vzhledem k datům a k designu.

3.1.4.1 **Data**

Lze říci, že každá webová stránka může být statická nebo dynamická.

Statická stránka je snazší a méně nákladnější na počáteční tvorbu. Nepotřebuje webový server pro zpracování obsahu. Pouze zmíněný prohlížeč na straně klienta (Kavourgias, 2014). Z názvu statická plyne i fakt, že je obsah dané stránky téměř neměnný. Stránky tohoto typu jsou většinou pouze informačního charakteru nebo menších společností, které neinvestovaly do dynamického webu. S uživatelem nenavazují žádnou interakci (Carey, 2006).

Oproti tomu dynamická stránka je z počátku nákladnější na vývoj, ale zahrnuje mnoho výhod. O dynamickou webovou stránku se jedná v případě, že lze se webem interagovat či zjednodušeně určitým způsobem pracovat (Kavourgias, 2014). Často jsou napojeny na příslušnou databázi, které slouží nejen pro načtení dat k zobrazení, ale i k zápisu některých akcí uživatele (Carey, 2006). Stránky zmíněného druhu vyžadují serverovou stranu neboli back-end. Ten zajišťuje překlad zdrojových kódu programovacích jazyků, jako je např. PHP, Perl, Java, C# aj. Příkladem dynamických stránek jsou sociální sítě, e-komerce, blogy, informační systémy, kalendáře atd.

3.1.4.2 **Design**

Dle uvedeného pohledu lze každou webovou stránku klasifikovat jako responzivní a neresponzivní.

Neresponzivní design se stává v současnosti téměř přežitkem. Pojednává o fixním designu, který se zobrazuje vždy stejně na zařízeních o různém rozměru. Při změně rozlišení obrazovky se komponenty neupraví do „ideálního“ rozložení (Frain, 2012). Jedinou možností je přidávat nadbytečné části kódu. Tzn. vytvářet několik například domovských stránek pro jednotlivá zařízení. Údržba se následně stává pracnou a nákladnou.

Na popud předchozího problému vznikl tzv. responzivní či adaptabilní design. Hranice mezi desktopovým a mobilním zařízením je již natolik zanedbatelná, že ji lze ignorovat a ke všem zařízením se chovat jako k zařízením stejného typu. Responzivní design umožňuje tvůrcům zaměřit se pouze jedinou stránku, která se bude měnit dle rozměrů použitého zařízení. Tímto zaniká potřeba vytvářet další design pro každé nové zařízení (Frain, 2012).

3.1.5 Parser

Parser je zjednodušeně algoritmus, který rozdělí vstupní strukturovaný souboru a zajistí co rychlejší přístup k rozděleným částem. Na současném poli parserů existují dva hlavní zastupitelé DOM a SAX.

DOM je zkratka pro *Document Object Model*. Každý vstupní dokument je převeden do stromové struktury. Strom se skládá z jednotlivých elementů, uzlů, které spolu tvoří větve stromu. Při použití DOM parseru je celá struktura nejprve načtena do paměti počítače. Následně je rozparserována, což vede k vyšší paměťové náročnosti. Výhodou DOM parseru je jeho rychlost přístupu ke konkrétnímu elementu (Hunter, 2007). Tato výhoda se ztrácí v případě objemnějšího vstupního souboru. Pro zpracování větších souborů více vyhovuje SAX parser, jelikož nemusí ukládat celou strukturu do paměti.

SAX znamená *Simple API for XML Parsing*. Parsování probíhá řádek po řádku přímo v rámci načítání vstupního souboru. Není tedy zapotřebí načítat celou strukturu do paměti, ale pouze nepatrnou část. Element je vrácen v okamžiku, kdy parser narazí na požadovanou část (Hunter, 2007). Proto se více používá při práci s většími soubory. Nevýhodou je složitější implementace a nižší výkonnost oproti DOM parseru.

3.1.6 Hashování

Pojednává o transformaci vstupního řetězce do výstupního za pomoci hashovací funkce, viz obrázek 4. Výstupní řetězec je nazýván jako *hashová hodnota*, *hash kód*, po případě pouze *hash*. Klíčovým atributem je, že transformace je pouze jednocestná (James, 2016). Hashování se využívá především u šifrovacích algoritmů. Typickým použitím je prokázání totožné vstupní sekvence. Nejpopulárnější hashovací algoritmy jsou v současné době SHA1, SHA2 a MD5.

Obr. 4 Transformace textu za pomoci hashovací funkce

3.1.7 Information retrieval

Information retrieval je obor zaměřený na získání dokumentů z relevantních i irelevantních kolekcí dokumentů, ve většině případů na základně klíčových slov (Feldman a Sanger, 2007). O dokumentech a kolekcích dokumentů více pojednává kapitola Text mining.

Rozmach internetu vedl ke zvýšení pozornosti k information retrieval. Search engine, česky vyhledávač, se stal klíčovým prvkem pro nalezení informací na webu. V současnosti je search engine nejčastěji použitým výrazem v oblasti information retrieval.

Jedním z hlavních důvodů, proč jsou některé vyhledávače lepší než jiné, tkví v kvalitě jejich služeb. Stěžejním atributem se stala rychlost zpracování dotazu uživatele. V současném internetu jsou miliardy dostupných webových stránek. Je zapotřebí poskytnout mechanizmus, který zajistí odpověď ve velmi krátkém časovém úseku. Nejznámější zástupci v oblasti vyhledávačů jsou například *Google*, *Yahoo!*, *MSN* a jiní. Zmínění představitelé dosahují rychlé odezvy převážně kvůli důkladné indexaci spolu s high-end infrastrukturou. Infrastruktura se skládá z několika stovek clusterů optimalizovaných na velký výpočetní výkon. Jejich systémy jsou škálovatelné a jsou schopny poskytnout kvalitní služby i v případě milionů současných připojení (Mattosinho, 2010).

Důležitou roli v information retrieval mají *ranking algoritmy*. Jsou schopny setřídit důležité dokumenty vzhledem k dotazu uživatele. Bez zmíněných algoritmů by uživatel neměl ponětí, se kterým dokumentem začít a pravděpodobně by musel procházet velké množství dokumentů. Výše zmíněné algoritmy poskytnou hierarchii dokumentů. Hierarchicky nejvýše jsou dokumenty, u kterých je více pravděpodobné, že obsahují požadovanou informaci (Mattosinho, 2010).

Pro lepší představu následuje workflow pro vyhledávání:

1. Uživatel zadá vyhledávací dotaz
2. Proběhne kontrola dotazu, zda je vhodný k vyhledávání. Kontrola vyústí v odstranění stopwords z dotazu. Provedení redukce slov na kořen, čili stemming. Případně se vykoná kontrola pravopisu. O stemming a stopwords více v kapitole Preprocessing.
3. Dotaz je porovnán oproti dostupným indexům. Jsou získány dokumenty, které obsahují některé z vyhledávaných výrazů. Na získanou množinu dokumentů jsou posléze aplikovány ranking algoritmy.
4. Výsledná seřazená množina je prezentována uživateli. Jinými slovy uživatel získává přístup k odpovídajícím dokumentům.

Information retrieval i information extraction, viz dále, jsou důležitou součástí pro nalezení a extrakci užitečných informací z nestrukturovaných nebo semi-strukturovaných dat. Jedná se o techniky, které mají významnou roli zvláště v současné době informační nadbytečnosti.

3.1.8 Information extraction

Information extraction je součástí umělé inteligence. Soustředí se na extrakci užitečných informací z nestrukturovaných nebo semi-strukturovaných dat. Extrakce informací se obvykle zaměřuje na objekty, jako jsou lidé, místa, vlastnosti a jiné. Pravidla extrakce mohou být závislá i na doméně či oblasti, ze které jsou data extrahována (Feldman a Sanger, 2007).

Cílem information extraction je identifikovat užitečné části textu z nestrukturovaných nebo semi-strukturovaných dat. Výsledná množina může být vhodná pro další operace jako je data mining.

Obecně je rozdíl mezi cílem information extraction a information retrieval. Na zmíněné procesy lze pohlížet jako na své doplňky s cílem zdokonalit jejich správnost a přesnost. Vzájemnou kooperaci ukázkově nastínila Moens (2006). Tvrdí, že klasické paradigma pro získávání informací již není nejvhodnější. Současné paradigma našlo inspiraci v příkladu běžné knihovny. Information retrieval vypomáhá nalézt potenciálně vhodné knihy a dokumenty. Pokud však bude knihovna oplývat značným množstvím tiskovin, bude zmíněná potenciální množina dokumentů a knih stále obrovská. Lidé vyžadují pokročilejší technologii, která jim vypomůže v hledání informací. Information extraction se stalo tíženým prostředkem.

Pojem extrakce znamená, že je požadovaná informace explicitně dostupná. Jinými slovy fyzicky existuje v textu. Na druhou stranu data mining techniky informace odvozují z dostupných dat (Mattoshinho, 2010). Důvodem objasnění byla potřeba rozlišovat mezi zmíněnými technikami. Je pravdou, že obě dokážou získávat znalosti, což neznamená, že mohou být zaměnitelné. Ve skutečnosti je information extraction nezbytnou části v preprocessingu pro zmenšení datové množiny. Teprve následně lze využít data mining a získat prozatím skryté znalosti.

3.1.9 **Natural language processing**

Česky zpracování přirozeného jazyka. Pojem je znám někdy jako počítačová lingvistika. Jedná se o oblast vědy, která se zabývá interakcí mezi počítačem a člověkem. Hlavním cílem je umožnit kvalitní komunikaci mezi zmíněnými dvěma prvky psanou nebo mluvenou formou (Nugues, 2006). Zde bude uvedena pouze psaná forma.

Pro mnohé aplikace je žádoucí automatizovaným způsobem zpracovávat text napsaný přirozeným jazykem. Počítače zvládají extrahovat i vytvářet texty v přirozeném jazyce, určit význam i identifikovat klíčová slova (Kumar, 2011).

3.1.10 **Preprocessing**

Dříve než proběhne samotné zpracování textu, je žádoucí data upravit. Zajistit datům určitou strukturu, zbavit se bezvýznamných slov, popřípadě slova převést do vhodnějšího tvaru. Prostřednictvím následujících metod lze zvýšit výslednou přidanou hodnotu.

3.1.10.1 **Tokenizace**

Tokenizace je proces segmentace řetězce znaků (Moreno a Torres, 2014). Zmíněnou operaci lze vykonat na několika úrovních. Dokumenty mohou být rozděleny po kapitolách, odstavcích, větách, slovech nebo dokonce po slabikách. V text miningových systémech se nejčastěji využívá přístupu rozdělení do vět a slov. Jedna věta či slovo, v závislosti na úrovni, jsou posléze nazývána jako *token*. Seznam tokenů se stane vstupem pro další zpracování textu (Feldman a Sanger, 2007).

Největší výzvou je identifikace hranic jednotlivých vět. Je rozdíl mezi tečkou, která udává konec věty a tečkou, která je součástí přechozího tokenu. Například za titulem se uvádí tečka a následně jméno či příjmení. Zde tečka neoznačuje konec věty.

Vzhledem k vývoji tokenizérů byla vypozorována určitá pravidla, která by měl každý token respektovat (Moreno a Torres, 2014):

- Výsledný seznamu tokenů může obsahovat interpunkční znaménka a netisknutelné znaky. Na úrovni slov je doporučeno zmíněné znaky vynechat.
- Všechny souvislé znaky řetězce jsou součástí jednoho tokenu. Obdobné pravidlo platí i pro čísla.
- Tokeny jsou vytvářeny rozdělením dle netisknutelných znaků, jako je například mezera nebo znak konce řádku, popřípadě na základě interpunkčních znamének.

3.1.10.2 **Normalizace**

Pojednává o procesu transformace vstupního textu do podoby, která je lépe uzpůsobena dalšímu zpracování. Existuje několik zásadních metod, kterých využívá velká většina text miningových systémů (Moreno a Torres, 2014).

- Jednou z metod normalizace je nahrazení několika mezer pouze jedinou. Pokud bude věta obsahovat více mezer a bude požadována separace věty dle mezer, nastává problém. Některé slovo bude uvozeno mezerou, což může způsobit další komplikace.
- Další metodou je nahrazení velkých písmen malými. Metoda zjednodušuje porovnávání. Není potřeba ověřovat všechny varianty slova, tzn. s velkým i malým písmenem.
- V případě, že daný jazyk obsahuje diakritiku, provést její odstranění. Výsledná množina by se měla pohybovat pouze v oblasti základních znaků ASCII.
- V důslednosti jsou i lemmatizace a stemming speciálními případy normalizace. Z kapitoly jsou vynechány, poněvadž se nejedná o triviální normalizační metody. Jsou jim věnovány separátní kapitoly.

3.1.10.3 **Lemmatizace**

V lingvistice pojem lemmatizace znamená proces získání základní větné formy daného slova. Jednotlivá slova se mohou vyskytovat v různých větných formách. Například v odlišných pádech či časech. Občas se zabývá i seskupením odvozených slov. Hlavním úkolem lemmatizace je snížit počet větných forem a odvozených slov (Katerattanakul, 2010). V následujícím textu je na množinu různých větných forem a odvozených slov odkazováno jako na množinu podobných slov. Výsledek lemmatizace je, že se celou množinou podobných slov zachází jako s jediným slovem, které je zástupné pro celou uvedenou množinu.

V procesu lemmatizace se zpravidla využívá slovníku a morfologické analýzy. Dle pana Andersona (2016) se morfologie v oblasti lingvistiky zabývá výzkumem slov. Jakým způsobem jsou tvořena a jaký mají vzájemný vztah vzhledem k dalším slovům v daném jazyce. Samotná lemmatizace odstraňuje konce slov a vrací pouze základní větnou formu slova. Získaná část je pojmenována jako *lemma* (Feldman a Sanger, 2007).

Jako příklad by mohla posloužit slova *výkonnostní*, *výkonný*. Slova jsou pouze orientační. Ve skutečnosti by do procesu lemmatizace měly vstupovat již normalizovaná slova, tedy *vykonnostni* a *vykonny*. Z uvedeného příkladu je výsledné lemma *vykon*. Pokud by byla množina vstupních slov rozšířena o slovo *vykonat*, pak by lemmatizér měl rozpoznat, že se jedná o slovo jiného významu, které nebude asociováno se zbylými slovy. Nevýhodami jsou náročnější implementace a závislost na kvalitě použitého slovníku.

3.1.10.4 **Stemming**

Stemming je úzce spjat s lemmatizací. Rozdíl je, že stemmer pracuje vždy s jediným slovem bez znalosti kontextu věty. Nemůže ve výsledku rozlišit mezi slovy, která mají stejný základ slova, ale jiný význam (Katerattanakul, 2010). Stejně jako lemmatizace má stemming za úkol redukovat množinu podobných slov. Na stemming se pohlíží jako na surový proces, který odstraňuje konce slov, nejčastěji přípony v naději, že se dosáhne požadovaného výsledku (Moreno a Torres, 2014). V mnoha případech stemming a lemmatizace vrátí stejné výsledky. Výhodou stemmingu je snazší implementace a rychlejší zpracování. Nevýhoda tkví v neschopnosti rozpoznání významu slova (Katerattanakul, 2010). Pokud by byl použit příklad z kapitoly Lemmatizace, pak by výsledná množina mohla obsahovat i slovo *vykonat*. Záleží na použitém stemmeru.

3.1.10.5 **Stopwords**

Stopwords, česky stopslova, mají široká využití v případě zpracování přirozeného jazyka. Jedná se o slova, která se v dokumentu objevují velmi často, ale jejich informační hodnota je nulová (Moreno a Torres, 2014). Svojí absencí význam neovlivní, proto mohou být odstraněna či ignorována. Stopwords se často používají i u fulltextových vyhledávačů. Jelikož by byl vyhledávací dotaz příliš velký, je vhodné nevýznamná slova ignorovat (Mattosinho, 2010). V českém jazyce se z pravidla jedná o zájmena, předložky, spojky nebo významově slabá slovesa jako jsou *být* či *mít.* Běžná česká stopslova mohou být například *a, aby, ale, když, nebo, protože* a mnoho dalších.

Slovník vytvořený ze stopwords je někdy nazýván jako negativní slovník. Pokud je součástí i lemmatizér, pak není potřeba pokrýt různé větné formy jednoho stopslova. V opačném případě je požadován úplný výčet (Mattosinho, 2010).

3.1.10.6 **Part-of-speech**

Je metoda pro zpracování přirozeného jazyka, která každé slovo ve větě označí příslušnou značkou či tagem. Metoda je nazývána jako part-of-speech, dále už jen POS.

POS je používán v lingvistice pro určení slovních druhů slova či sousloví (Kumar, 2011). Nejčastěji se využívají *podstatná jména*, *přídavná jména, příslovce* a *slovesa*. Sporadicky *zájmena*, *předložky*, *spojky* či *citoslovce*. Někdy se však používá i více kategorií současně pro sousloví. Důvodem vzniku POS byla potřeba zjistit, jakou roli mají jednotlivá slova ve větě. Podstatná jména dávají objektům jejich název. Přídavná jména popisují podstatná jména. Někdy mohou být významná příslovce nebo zájmena, pokud určitým způsobem mění význam následujícího slova (Mattosinho, 2010), například *nikdy nekupovat*.

Hned několik algoritmů může využít označená slova. Při využití tokenizace věty s označenými slovy lze těžit z pomyslných ukazatelů u jednotlivých slov. Prostřednictvím ukazatelů je umožněno pracovat pouze s určitými druhy slov, popřípadě lze určité druhy slov ignorovat a zvýšit efektivitu algoritmu, který na POS navazuje. Níže je uveden názorný příklad, jak vypadají označená slova dle POS. Je využit lingvistický nástroj od společnosti Xerox. Slovní druhy jsou uvedeny kapitálkami za znakem +.

Vstupní sekvence slov
Televize má vynikající obraz

Sekvence slov s aplikovaným POS
Televize+NOUN má+VERB vynikající+ADJ obraz+NOUN

3.1.11 **Data mining**

V kapitole Information extraction bylo uvedena problematika extrakce explicitně dostupných informací z textových zdrojů. Na druhou stranu data mining může vytvářet korelace mezi daty, které nejsou na první pohled patrné. Zmíněným způsobem lze prostřednictvím data mining získávat znalosti. Proces je znám pod názvem *Knowledge discovery*, česky získávání znalostí. Je definován jako netriviální extrakce implicitních, předem neznámých a potenciálně užitečných informací z datových zdrojů (Cheung a kolektiv, 2005). Data mining techniky jsou užitečným nástrojem v mnoha aplikacích. Převážně v systémech, které obsahují nepředstavitelné množství dat k analýze. Pro člověka by bylo nereálné provést úlohu manuálně.

3.1.12 **Text mining**

Text mining může být ze široka definován jako proces náročný na získávání znalostí z textových dokumentů, kde uživatel pracuje s kolekcí dokumentů pomocí sady nástrojů pro analýzu. Stejně jako data mining se text mining zaměřuje na extrakci užitečných informací z datových zdrojů. Na rozdíl od data mining jsou datové zdroje textové soubory známé jako kolekce dokumentů (Feldman a Sanger, 2007).

Samozřejmě se text mining velmi inspiruje z výzkumu data mining. Proto není náročné zjistit, že mají oba zmíněné systémy značnou podobnost v architektuře. Například oba využívají preprocessing.

V případě data mining se předpokládá, že jsou data již uložena ve strukturované podobě. Jeho preprocessing metody jsou změřeny na čistění, normalizování dat a vytváření rozsáhlého množství joinů mezi tabulkami (Cheung a kolektiv, 2005). V kontrastu se preprocessing u text miningových systémů soustředí na identifikaci a extrakci reprezentativních příznaků z dokumentů psaných přirozeným jazykem. Výsledkem textového preprocessingu je převod nestrukturovaných dat, do přechodně strukturované podoby (Feldman a Sanger, 2007).

3.1.12.1 **Dokument**

Základním elementem v text mining je dokument. Může být popsán jako diskrétní jednotka textových dat v kolekci. Obvykle, ale ne nezbytně, souvisí s dokumentem z reálného světa. Jako například *obchodní zpráva*, *email*, *článek*, *reportáž* aj.

Feldman a Sanger (2007) tvrdí, že i přes poněkud zavádějící popisek nestrukturovaných dat může být textový dokument viděn z několika hledisek i jako strukturovaný objekt.

Typografické prvky jako jsou interpunkční znaménka, velká písmena, čísla, mezery, znak nového řádku a jiné určitým způsobem značkují soubor. Dle zmíněných elementů lze identifikovat odstavce, nadpisy, data vydání, jméno autora, data z hlavičky či patičky.

Existují dva základní typy dokumentů (Berry a Kogan, 2010):

- **Slabě strukturované / Nestrukturované** dokumenty vytvořené prostřednictvím WYSIWYG editorů, pojednávají o textových zdrojích, které nemají žádnou strukturu. Jedná se o tzv. *plaintext*.
- **Semi-strukturované** dokumenty jako jsou například HTML stránky, PDF soubory a jiné. Podstatné je, že jsou určité části textu přímo ohraničeny konkrétními elementy či tagy. Například u HTML lze odstavec ohraničit tagy * *, nadpis první úrovně pomocí *<h1></h1>* atd.

3.1.12.2 **Kolekce dokumentů**

Několik text miningových systémů se zaměřuje na kolekci dokumentů. Kolekce dokumentů je jakékoliv seskupení textových dokumentů (Feldman a Sanger, 2007).

Každý dokument je reprezentován množinou příznaků, zjednodušeně lze říci množinou slov. Příkladem může být následující seskupení (kolekce).

$$C \in \cup \forall D_i \mid i \in N \wedge W \in D_i$$

- Množina C představuje výslednou kolekci
- D_i je i-tý dokument
- W je konkrétní slovo, například *definice*

Pak lze zmíněný výraz interpretovat následovně. Kolekce dokumentů C náleží sjednocení všech dokumentů D_i, které obsahují slovo W.

3.1.12.3 **Dokument features**

Preprocessing se pokouší využít mnoho různých prvků obsažených v dokumentu, který je psán v přirozeném jazyce. Je vhodné dokument převést z nevyhovující, výchozí podoby do explicitní strukturované podoby (Berry a Kogan, 2010). Nicméně i obsahově malý dokument může navrátit potenciálně enormní množství slov, frází, vět i typografických elementů. Je samozřejmé, že se slova mohou vyskytovat v různých tvarech, pádech, popřípadě i časech, čímž daný kontext ještě více naroste.

Základní úlohou většiny text miningových systémů je identifikovat zjednodušenou podmnožinu *features* v dokumentu. V češtině se *features* obvykle říká příznaky, které lze následně využít k reprezentaci dílčího dokumentu. Feldman a Sanger (2007) se na zmíněnou množinu odkazují jako na *reprezentační model dokumentu*. Říkají, že je každý dokument reprezentován *množinou příznaků*, kterou daný reprezentační model obsahuje. Základní rozdělení příznaků dokumentu je rozebráno v následujících podkapitolách.

3.1.12.4 **Znaky**

Pojednávají o jednotlivých *písmenech*, *číslech*, *speciálních znacích* či *mezerách*. Zmíněné prvky jsou stavebními kameny pro vyšší úrovně příznaků, jako jsou slova nebo termy. Znaková úroveň může obsahovat úplnou množinu všech znaků nebo jen určitou podmnožinu. Znaková reprezentace bez určení pozice konkrétního znaku má velmi omezenou použitelnost. S určenou pozicí je poněkud použitelnější. Obecně bývá znaková úroveň nepraktická a téměř nepoužitelná. Na druhou stránku je však nejkompletnější vzhledem k obsahu dokumentu (Feldman a Sanger, 2007).

3.1.12.5 **Slova**

Jednotlivá slova vybraná přímo z dokumentu. Jsou základní prvky, které dávají význam větě či souvětí. Obecně platí, že jeden příznak na úrovni slov by neměl mít hodnotu větší než jeden token. Čili fráze nebo víceslovná spojení nejsou vyhodnocena jako příznak na slovní úrovni. Je možné zahrnout všechny příznaky, jinak řečeno každé slovo, které se vyskytuje v určitém dokumentu. Dokument, který je reprezentován úplnou množinou jeho příznaků se nazývá *full text* (Feldman a Sanger, 2007). Dotyčný přístup však může vést i ke stovkám tisíc jedinečných slov. Nicméně většina systémů na úrovni slov využívá alespoň minimální optimalizaci. Odfiltrování stopwords, speciálních znaků a číslic podstatně sníží velikost výsledné množiny, přičemž infomační hodnota zůstává téměř neměnná.

3.1.12.6 **Termy**

Termy jsou konkrétní *klíčová slova* a zároveň i víceslovná spojení vybraná přímo z dokumentu, prostřednictvím metod nazvaných jako *term-extraction* (Berry a Kogan, 2010). Dle uvedené skupiny metod jsou zvolena slova, které by svým významem měla být reprezentativní.

Výsledné termy je vhodné skládat i za pomoci sjednocení podmnožin termů. Například, pokud by v dokumentu byla následující věta (Feldman a Sanger, 2007):

„Prezident Abraham Lincoln zažil kariéru, která ho poslala ze srubu

do Bílého domu.“

Seznam termů reprezentujících dokument by mohla zahrnovat jednotlivá slova *Lincoln*, *zažil*, *kariéru*, *srubu*, *domu*. Součástí výsledné množiny termů mohou být i víceslovná spojení jako jsou *Prezident Abraham Lincoln* a *Bílého domu*.

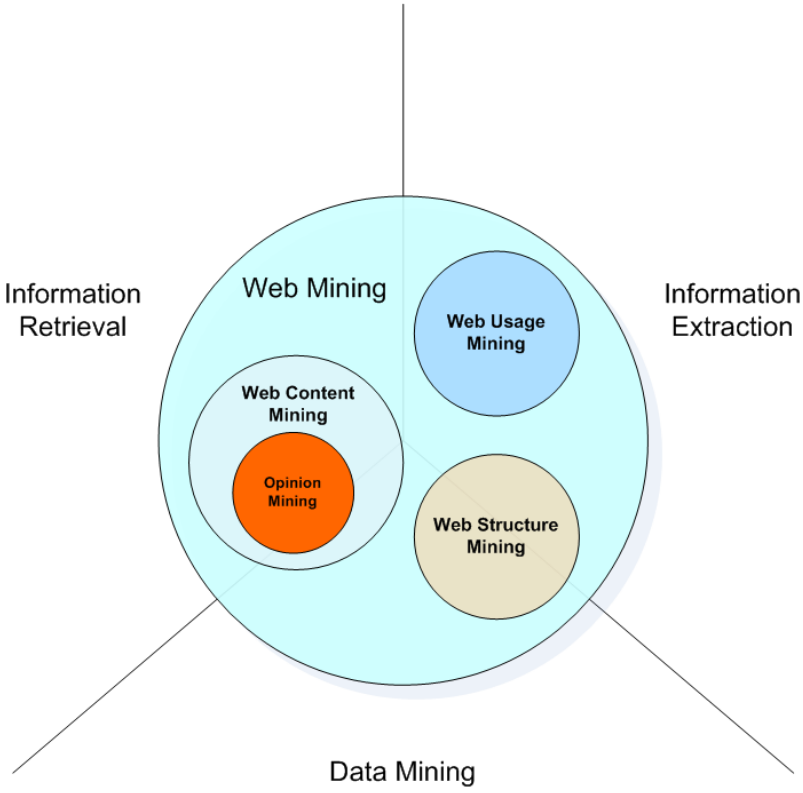
Proces by měl vést k relativně malé, ale sémanticky bohatší množině. Měla by lépe reprezentovat dokument, než výsledná množina na úrovni slov. Nevýhodou, je že se nemusí „povést“ extrakce významných slov, čili jsou přidána slova nevýznamná. Na slovní úrovni, v případě full textu, nemůže ke zmíněnému problému dojít, jelikož obsahuje množinu významných i nevýznamných slov (Feldman a Sanger, 2007).

3.1.13 **Web mining**

Typické data mining systémy mají počátek dlouho před internetem. Zmíněné systémy byly využívány převážně na rozpoznání vzorů nebo na získávání znalostí z obrovských databází. Databáze uchovávaly data ve strukturované podobě. Jednotlivé informace byly uloženy v malých smysluplných celcích, které reprezentovaly znalosti o určité doméně (Cheung a kolektiv, 2005).

Historicky byl data mining zaměřován převážně na databáze. Avšak s vývojem internetu došlo i k expanzi oblasti data mining. Vznikl web data mining nebo zkráceně pouze web mining. Zjednodušeně se jedná o techniky data mining aplikované na oblast internetu. Web mining má významné odlišnosti od klasického data mining. U data mining se předpokládá, že jsou data již shromážděna a připravena v databázi. V případě web mining je zapotřebí využít metod pro information retrieval a information extraction, které pořídí data a vhodným způsobem je připraví pro samotný mining. S neustálým rozvojem internetu došlo i k vývoji web mining. Rozpadl se na následující podkategorie (Mattosinho, 2010). Graficky jsou znázorněny na obrázku 5.

- **Web structure mining** cílí na získání znalostí z hypertextových odkazů. Jedná se o způsob, jak zjistit vazby či vztahy mezi webovými stránkami.
- **Web usage mining** se zaměřuje na nalezení vzorů ve využití webové stránky. Každý klik uživatele má potenciální hodnotu a vypovídá o jeho zájmech.
- **Web content mining** pracuje přímo s informacemi na webové stránce. Cílem web content mining je získat znalosti ze zmíněných stránek



Obr. 5 Web mining, (Mattosinho, 2010)

3.1.14 **Opinion mining**

Na obrázku 5 je vyobrazeno, že opinion mining je podoblastí web content mining. Opinion mining se koncentruje na získávání užitečných informací z uživatelských recenzí či názorů. Jedná se o poměrně novou oblast. Její význam rapidně vzrostl převážně z důvodu rychlého rozvoje e-komerce, blogů, sociálních sítí, fór a jiných zdrojů uživatelských názorů (Mattosinho, 2010).

Mnoho lidí již změnilo návyk z fyzického nakupování v obchodech na virtuální nakupování z pohodlí jejich domova. Pro mnoho zákazníků se stalo běžnou praxí, že si nejprve zjišťují užitečné informace a názory ostatních zákazníků před samotnou koupí určitého produktu. Závažným problémem se však stává nalezení požadovaných informací a jejich vyhodnocení. Není obtížné vyhledat webové stránky se stovkami recenzí o nějakém produktu, ale najít v recenzích užitečné informace (Dařena a kolektiv, 2014).

Například nový zákazník má zájem o recenze, které jsou zaměřeny na škálu funkcí určitého mobilního telefonu konkrétní značky. Avšak starší zákazník, který si kdysi zakoupil mobilní telefon značky A, má zájem pouze o recenze, které se zabývají výhradně kvalitou displeje aktuálně prodávaného mobilního telefonu opět od společnosti značky A. Z obchodního pohledu může být získávání informací z názorů kvalitním zdrojem pro tvorbu reklam. Podstatná je i zpětná vazba pro výrobce, který má možnost zkvalitňovat jejich produkty či služby. Například webová stránka ori-

entovaná na prodej elektroniky může na své stránky umístit reklamy, které jsou založeny na názorech spotřebitelů. Dalším příkladem může být i reakce na produkty s negativními ohlasy. Zřizovatel webové stránky, pak může umístit reklamu, která bude zobrazovat lépe hodnocený, alternativní produkt (Bing, 2012).

Názor je osobní přesvědčení nebo rozhodnutí o předmětu. Existují základní dvě kategorie názorů. Pojem *odborná recenze* či *znalecký posudek* má zaručit, že osoba či skupina, která produkuje zmíněný typ recenze, má vyšší vzdělání o daném produktu či službě. Zatímco *uživatelské recenze* jsou vytvářeny běžnými uživateli či spotřebiteli (Bing, 2012). Je samozřejmé, že odborný posudek má mnohem vyšší kvalitu a přesnost v technických podrobnostech. Na druhou stranu mohou být řízeny společnostmi, aby privilegovaly určité produkty. Uživatelské recenze, zmíněných kvalit sice nedosahují, ale bývají z převážné většiny od skutečných spotřebitelů. Často jsou ve formě seznamu kladů a záporů.

V současné práci jsou pojmy recenze, recenze produktu, uživatelská recenze, názor uživatele, názor na produkt zaměnitelné a podstatně se liší od odborných recenzí, jak bylo objasněno výše. Práce nebude pracovat s odbornými recenzemi, ale výhradně s uživatelskými.

3.1.15 **Sentiment analysis**

Česky analýza sentimentu, někdy označována i jako klasifikace sentimentu. Je oblast studia, která se zabývá stanovením sentimentu. Slovo sentiment je synonymem pro polaritu či cítění. Nejčastěji používaný výraz je polarita, která charakterizuje orientaci textových dat (Mattosinho, 2010).

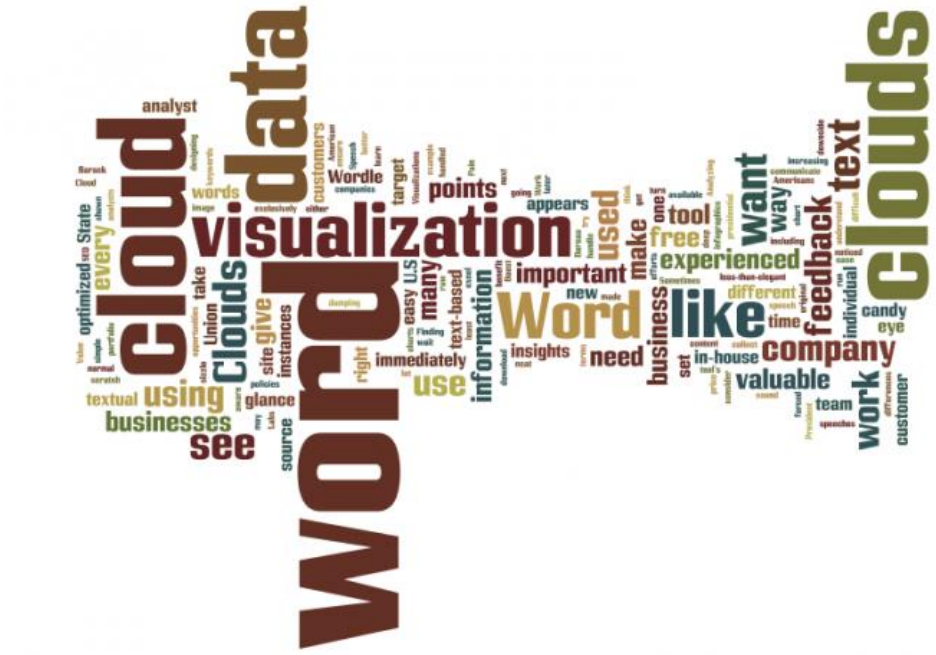
Analýza sentimentu může být provedena na několika úrovních. Lze klasifikovat dokumenty, odstavce, jednotlivé věty, fráze či dílčí slova. Pro některé aplikace postačí aplikovat klasifikaci sentimentu na celé dokumenty. Avšak jsou i jiné, které vyžadují jemnější stupeň (Bing, 2012).

Na klasifikaci sentimentu dílčích slov je pohlíženo jako na klasifikaci jednotlivých features ve větě, nikoliv každého slova (Mattosinho, 2010). Hlavním důvodem, proč je metodika na slovní nebo frázové úrovni více v oblibě než ostatní, bude objasněno na následujícím příkladu. Předpokladem je specializovaný internetový obchod s televizemi. V rámci webových stránek zmíněného obchodu mohou spotřebitelé psát pozitivní i negativní recenze k jednotlivým produktům. Příklad konkrétní recenze: *„Televize má vynikající obraz, ale ovládání trochu zaostává“.* Z příkladu je patrné, že recenze je spíše pozitivní. Pokud by byla klasifikace sentimentu vykonána na úrovni věty, došlo by k určité ztrátě negativní informace (Mattosinho, 2010).

Identifikace sentimentu se provádí pomocí tzv. *názorových slov*. Názorová slova kódují pocitový stav, který může být žádoucí nebo nežádoucí. Názorová slova, která ukrývají žádoucí stav, například slova *krásné*, *pěkné*, *vhodné*, *úžasné*, mají pozitivní orientaci. Zatímco slova nežádoucího stavu, například *špatné*, *otřesné*, *neuspokojivé*, mají negativní orientaci. Názorová slova mohou pramenit z mnoha kategorií slovních druhů, avšak z pravidla se jedná o přídavná jména a příslovce (Bing, 2012).

3.1.16 Word cloud

Vizualizace dat jako jsou sloupcové, koláčové grafy a jiné dávají pozorovateli hodnotný způsob vizuální prezentace dat. Problém nastává v případě textových dat. *Word cloud* je jedním z řešení. Někdy je označován za *text cloud* či *tag cloud*. Princip je jednoduchý. Čím větší frekventovanost dané slovo, či skupina slov má, tím je font větší a někdy i tučnější vzhledem k méně významným slovům (Boost Labs, 2016). Na obrázku 6 lze na první pohled identifikovat, která témata jsou probírána nejvíce.



Obr. 6 Vizualizace textu prostřednictvím Word cloud, (Boost Labs, 2016)

3.2 Metody a trendy

V kapitole budou nastíněny některé současné techniky využívané v oblastech párování názvů, information retrieval, information extraction, text či opinion mining.

3.2.1 Párování názvů

Existují situace, kdy je žádoucí porovnávat řetězce nejen absolutní shodou, ale počítat i s určitou podobností.

3.2.1.1 LIKE operátor

Je operátor pro porovnání řetězců v databázi. Využívá se s tzv. *wild cards*. Jedná se o znaky například % či _. Procento udává posloupnost několika jakýchkoliv znaků, zatímco podtržítko právě jeden. Výhodou je snadné použití a rychlost. Nevýhodou je však stálá nutnost určité sekvence znaků, například *Microsoft Xbox One%*.

3.2.1.2 Levenshteinova vzdálenost

Je vzdálenost dvou řetězců, definovaná jako nezbytný počet operací (odstranit, přidat nebo nahradit), aby byly porovnávané řetězce totožné (Algoritmy.net, 2015). Vzdálenost je realizována prostřednictvím tvorby jednoduché tabulky a určitého pohybu v ní. Jednotlivé pohyby jsou uvedeny na stránce Algoritmy.net (2015). Tabulka má počet řádků stejný jako je délka zadaného řetězce (viz níže *abdc*) a počet sloupců rovný délce řetězce, se kterým porovnáme (viz níže *abcde*). Výhodou je, že není potřeba splnit určitou sekvencí znaků, jako v případě LIKE operátoru. Nevýhodou může být složitější implementace a rychlost zpracování při rozsáhlém objemu dat. Jako příklad poslouží řetězce *abdc* a *abcde*. Pomocí tří operací lze zajistit identický řetězec. Proto je Levenshteinova vzdálenost rovna 3.

3.2.2 Information retrieval

3.2.2.1 Internetový vyhledávač

Vyhledávač je služba, která umožňuje dotazovanému vrátit sekvenci HTML dokumentů. Nejprve je zapotřebí zadat klíčová slova pro vyhledávání, která popisují požadovanou informaci. Uživatel následně získá seznam dokumentů, které vyhověly zadanému dotazu. Posledním krokem je volba konkrétní webové stránky z výsledného seznamu. V současnosti je nejběžnějším prostředkem pro získávání dokumentů.

3.2.2.2 Webová služba

Nejprve je zapotřebí definovat API. API je softwarový prostředník, který umožňuje aplikacím vzájemnou komunikaci a výměnu dat. Jinými slovy pojednává o specifikaci rozhraní aplikace.

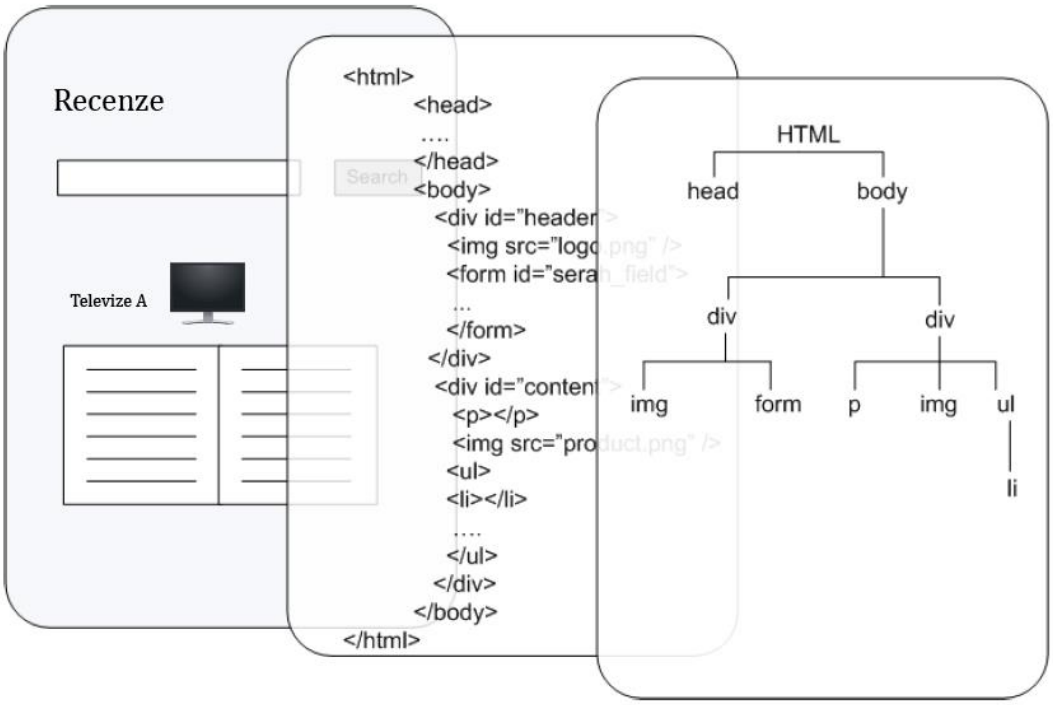
Často se využívá RESTful API, které umožňuje vystavit určitou funkcionalitu prostřednictvím webových služeb i aplikacím třetích stran. Jednotlivé služby konzumují nebo produkují požadovaná data. RESTful API využívá nejčastěji soubory s formátem XML nebo JSON pro datovou výměnu. Zmíněné API obsahuje dotazy typu GET, POST, PUT a DELETE. Jednotlivé dotazy jsou volány na specifikovanou URI. Na základě uvedených dotazů lze operovat s daty dané aplikace (Richardson a Amundsen, 2013). Výhodou je definované rozhraní vytvořené tvůrcem aplikace. Dále rychlost přístupu k datům vzhledem k web scraping, viz kapitola níže. Nevýhodou může být omezení počtu dotazů za jednotku času, jako má například Amazon. Webové služby nebývají poskytnuty zdarma, a pokud ano, tak omezeně.

3.2.3 Information extraction

Kapitola popisuje zástupce v oblasti extrakce dat. Je podstatné zmínit, že je soustředěna na metody pro získání dat z internetových zdrojů. Hlavním zástupcem je web scraping v případě extrakce dat přímo z webové stránky. Při využití webové služby je stěžejní parser.

3.2.3.1 **Web scraping**

Web scraping metoda je využívaná pro sběr informací z webových stránek (Munzert a kolektiv, 2015). V kapitole Webová stránka bylo zmíněno, že jsou vytvářeny prostřednictvím HTML či XHTML jazyka. Dokumenty jsou ve formě stromové struktury, známé jako *Document Object Model*, zkráceně DOM. Cílem HTML je ustanovit strukturu dokumentu, který bude zobrazen webovým prohlížečem, což je názorně vyobrazeno na obrázku 7. Součástí nejsou kaskádové styly.



Obr. 7 HTML dokument ze tří pohledů. Zleva doprava: Prezentovaná stránka, HTML kód, DOM, (Mattosinho, 2010)

Web scraping využívá data z online zdrojů. Zřizovatelé webových stránek jsou obvykle vlastníky prezentovaných dat. Jelikož se jedná o použití cizích dat, je vhodné zjistit, zda nedochází k porušení podmínek provozovatele nebo se dotázat o svolení (Munzert a kolektiv, 2015).

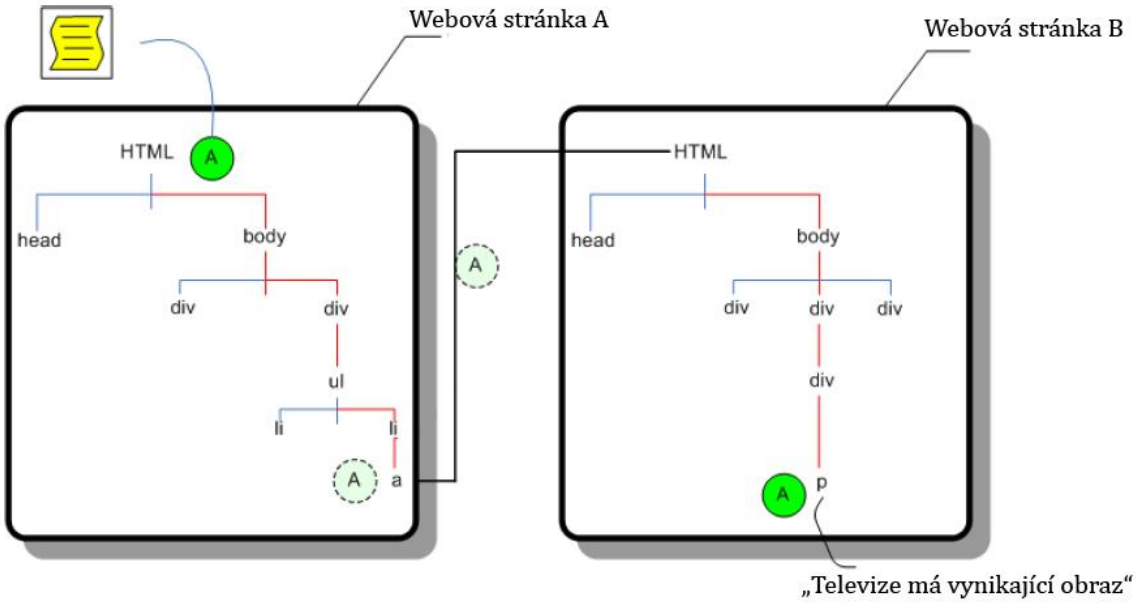
Z funkčního pohledu je web scraping z velké části podobný operacím kopírovat a vložit. S tím rozdílem, že jsou zmíněné operace vykonány automatizovaný a organizovaným způsobem prostřednictvím programu či agenta. Agent zpracovává jednotlivé elementy na webové stránce. Jinými slovy vykonává totožnou činnost, jakou by vykonal uživatel, který by chtěl získat určitá data z webové stránky. Pro zpracování odkazů agent vysílá požadavek typu *HTTP GET* nebo *HTTP POST* (Mattosinho, 2010).

Je samozřejmostí, že program dokáže vykonat nesrovnatelně více operací oproti člověku za stejný čas. Není tedy překvapení, že výhoda tkví v rychlosti vzhledem k uživateli. Nevýhoda může být v politice maximálního počtu požadavků na stránku. Server, na kterém je umístěna požadovaná stránka, se může bránit potenciálnímu útoku prostřednictvím tzv. *Denial-of-Service* či *DoS*. Důvodem je značné množství požadavků z konkrétního zdroje v krátkém časovém úseku. Více o DoS lze zjistit od autora Mirkovic (2015).

Po získání požadované stránky je hlavním aktérem DOM parser, jelikož je webová stránka relativně malý dokument. Parser vyhledává klíčové elementy na základně CSS selektorů nebo XPATH (Mattosinho, 2010). Více se lze o parserech dočíst v kapitole Důležité pojmy, podkapitola Parser. Po nalezení požadovaného prvku se často využívá regulárních výrazů, v případě, že není žádoucí celý text elementu nebo jeho atributů. Proces sběru a průchodu přes jednotlivé webové stránky je názorně zobrazen na obrázku 8. Červenou linkou je znázorněn průchod web scraperu. Písmeno A v zeleném kruhu značí odkaz na další webovou stránku.

Jelikož má zpravidla každá webová stránka odlišné CSS selektory, je téměř nemožné vytvořit obecný web scraper a využívat jej napříč vícero stránek (Munzert a kolektiv, 2015).

Za zmínku stojí i výkonová stránka. Při rozsáhlém sběru informací nemusí být web scraping optimálním řešením. Odpověď na požadavek vrátí celý dokument, přičemž je využita pouze nepatrná část. Výsledný proces je z pohledu účinnosti velmi neefektivní. Nicméně i se svými nedostatky je web scraping velmi účinný nástroj a často i jediná veřejně dostupná metoda pro získání informací z webové stránky (Mattosinho, 2010).



Obr. 8 Web scraping agent, (Mattosinho, 2010)

3.2.3.2 Parser

Při použití webové služby je navrácen soubor určitého formátu. Z pravidla se jedná o soubor formátu XML a JSON (Richardson a Amundsen, 2013). Pokud nebude soubor neúměrně rozsáhlý, lze použít DOM parser. A ve většině případů se i využije. Pro aplikaci není vůbec žádoucí přesouvat rozsáhla XML. V takovém případě je většinou navrácena chybová hláška o příliš obecném dotazu. O parserech bylo řečeno již dříve a proto zde nebudou znovu objasněny jejich metodiky a typy.

3.2.4 Text mining

Kapitola pojednává o způsobu jakým zjistit reprezentativní příznaky v textových datech. Obsahuje i popis tzv. *frekventovaných* a *nefrekventovaných příznaků*. Dále jakým způsobem získat možné vztahy mezi příznaky. Zmíněné metody lze aplikovat i na opinion mining, jelikož se jedná o podmnožinu oblasti text mining.

3.2.4.1 Identifikace příznaků

Proces zaměřený na získávání možných příznaků z označených textových zdrojů. Zdroje byly označeny prostřednictvím POS. Klíčová slova jsou v reálném světě podstatná jména. Dávají jméno produktu i jeho příznakům, například *kvalita*, *displej*, *baterie* a další. V práci jsou definovány dvě kategorie příznaků. Jedná se o tzv. *frekventované* a *nefrekventované* příznaky (Mattosinho, 2010).

Je důležité objasnit klady i zápory automatizovaného a manuálního přístupu. Velké pozitivum frekventovaných i nefrekventovaných příznaků je automatizace celého procesu s pouze minimálním zásahem člověka. U frekventovaných příznaků je však nevýhodou, že závisí na velkém množství textu k analýze. Celkový proces je přímo závislý na POS. Není zajištěno, že se v rámci frekventovaných příznaků naleznou správné příznaky. Správné příznaky jsou takové, které jsou příznaky i ve skutečném světě. V jiných pracích (Chungping, 2009), byla identifikace příznaků vykonána manuálním označováním. Výhodou je samozřejmě přesnost identifikovaných příznaků i těch, které nejsou frekventované. Na druhou stranu je obrovskou nevýhodou velká pracnost při označování rozsáhlejších zdrojů či neustálá potřeba označení inkrementů v textových zdrojích.

- **Frekventované příznaky** – V předešlém textu bylo nastíněno, že se extrahují pouze podstatná jména. U frekventovaných příznaků jsou zmíněná podstatná jména nazývána jako kandidátní příznaky. Pro extrakci je využito algoritmu, který počítá výskyt jednotlivých kandidátních příznaků. Algoritmus je inspirován částí algoritmu z asociačních pravidel. Konkrétně hodnotou support, viz následující kapitola. Myšlenka algoritmu je, že příznak, který se objeví ve více názorech, má potenciálně vyšší šanci být významný. Je vhodné definovat prahovou hodnotu, pod kterou bude příznak vyhodnocen jako nevýznamný.
- **Nefrekventované příznaky** – V případě, že frekventované příznaky nezískají žádné příznaky, je vhodné využít metodu nefrekventovaných příznaků. Předpokladem je, že existují kandidátní příznaky a neexistují frekventované příznaky. Následně je každý kandidátní příznak klasifikován jako významný.

3.2.4.2 **Asociační pravidla**

Jsou jedním z nejrozšířenějších pojmů. Jejich cílem je objevit vztah mezi datovými položkami. Formálně jsou asociační pravidla definována následně. W je množina položek $w_1, w_2, \dots, w_n$. Dále T je množina transakcí $t_1, t_2, \dots, t_m$. Každá transakce má jedinečný identifikátor. Jednotlivé transakce jsou složeny z položek. Platí, že $T \subseteq$ W. Asociační pravidlo implikuje následující: $A \rightarrow B$, kde $A \in W, B \in W$ *a* $A \cap B = \emptyset$ (Adamo, 2012).

Výstupem asociačních pravidel jsou dva významné atributy *support* a *confidence*, které budou vysvětleny níže. Nejznámějším algoritmem implementujícím asociační pravidla je tzv. Apriori algoritmus (Adamo, 2012).

Typických výukovým příkladem je analýza nákupního košíku (Adamo, 2012). Vlastníka obchodu zajímá, které komodity lidé kupují současně.

$$Chleba \rightarrow Máslo\ (support = 10\ \%, confidence = 90\ \%)$$

Zmíněný zápis říká, že 10 % všech zákazníků zakoupilo chleba i máslo současně, což představuje hodnotu support. Confidence značí, že zákazníci, kteří si koupili chleba, si v 90 % zakoupili i máslo. Chleba představuje znak A, máslo B.

Support lze formálně popsat jako procentuální výskyt v množině transakcí T, které obsahují $A \cup B$. Na support lze pohlížet jako na pravděpodobnost výskytu.

$$support = \frac{(A \cup B) \cdot i}{m}$$

- *i* je počet transakcí, které obsahují A i B
- *m* je celkový počet transakcí, čili |T|

Conficence je obdobná s jediným rozdílem, že není počítáno vzhledem k celé množině transakcí T, ale pouze k podmnožině, která obsahuje A.

$$confidence = \frac{(A \cup B) \cdot i}{A \cdot j}$$

- *i* je počet transakcí, které obsahují A i B
- *j* je počet transakcí, které obsahují pouze A

Pro úplné pochopení je uveden následující příklad. Na každém řádku je uveden obsah zákazníkova košíku, čili položky jednotlivé transakce. Majitel obchodu specifikoval, že požaduje minimální support 30 % a zároveň confidence alespoň 60 %.

t_1: Máslo, Chleba, Mléko
t_2: Pomeranč, Cukr, Mléko
t_3: Máslo, Mléko, Cukr
t_4: Chleba, Máslo, Sýr
t_5: Chleba, Sýr, Mléko

$$Chleba \rightarrow Máslo \left(support = \frac{2}{5}, confidence = \frac{2}{3}\right)$$

Chleba a Máslo, splňuje kritéria. Minimální support je roven 40 % a confidence 66,6 %.

$$Máslo \rightarrow Sýr \left(support = \frac{1}{5}, confidence = \frac{1}{3}\right)$$

Oproti tomu Máslo a sýr kritéria nesplňuje. Minimální support je roven 20 % a confidence 33.3 %.

4 Vlastní práce

V předchozí kapitole byly vysvětleny stěžejní pojmy, o které se práce opírá. Pro konkrétní oblasti bylo uvedeno několik metod, které byly zvoleny jako kandidátní. V současné části práce bude z kandidátních metod zvolena právě jedna, která poslouží daným účelům. Hlavní podstata kapitoly tkví ve zdokumentování praktické části diplomové práce.

4.1 Prostředí

Prvotním požadavkem bylo vytvořit vhodné prostředí pro vývoj. Z důvodu lepší přenositelnosti a případné pozdější migrace byla využita virtualizace operačního systému prostřednictvím hypervizoru *VMWare Workstation verze 10.0.1*. V podkapitole je uvedena hardwarová specifikace, která posloužila pro vývoj i testování aplikace.

Vhodný programovací jazyk pro tvorbu aplikace byl zvolen Perl 5, který je znám svou silou pro práci s textem pomocí regulárních výrazů. Pro větší přehlednost i modernost byla aplikace implementována prostřednictvím objektově orientovaného paradigma.

Jako operační systém byl zvolen Linux. Jelikož se jedná o open source systém, není zatížen licenčními poplatky. Dalším atributem pro volbu zmíněného systému byla nejširší podpora knihoven pro jazyk Perl vzhledem k ostatním operačním systémům. Distribuce byla vybrána *Ubuntu, kernel version 4.4.0-42-generic*. Vybrána byla z důvodu autorových dřívějších zkušeností.

Dalším krokem byla volba IDE. Z předchozích zkušeností autora vyplynulo, že běžný textový editor, jako je *gedit*, není vhodný pro tvorbu rozsáhlejších aplikací. Bylo využito prostředí *Eclipse* s rozšiřujícím pluginem *EPIC*, který zajišťuje podporu pro Perl. EPIC podporuje zvýraznění syntaxe, její kontrolu, zobrazení varovných hlášení již při psaní zdrojového kódu či napovídání názvu proměnných a metod.
Pro vyšší zabezpečení a lepší kontrolu nad vytvořenou funkcionalitou, byly veškeré zdrojové kódy i konfigurační soubory verzovány pomocí systému *Git* a nástroje *Bitbucket*. Lze zde procházet celé soubory či pouze přírůstky mezi jednotlivými commity. Adresa úložiště je `https://bitbucket.org/Luffhassa/dp/src`.

Následující potřebou byla instalace databáze pro uložení zpracovaných výsledků. Využita *MariaDB*, jakožto ekvivalent k MySQL, opět z důvodu open source a častého použití v rámci OS Linux. Pro práci s databází byl využit nástroj *MySQL Workbench*. Nástroj obsahuje prostředky pro tvorbu ERD diagramu. Ze zmíněného diagramu lze přímo generovat SQL skript pro vytvoření databáze, tzv. *create skript*.

Posledním krokem v rámci přípravy prostředí byla instalace knihoven pro Perl. Některé existovaly již v operačním systému, jiné byly nutné doinstalovat ze serveru cpan.org. V podkapitole Použité knihovny je uveden výčet použitých knihoven včetně krátkého popisu.

4.1.1 Hardwarová specifikace

Následující tabulka 1 obsahuje popis komponent, které byly využity pro virtuální operační systém Ubuntu Linux. V tabulce je nejprve uvedena popisovaná komponenta, následně její výrobce, dále výrobní řada daného výrobce a nakonec hodnota komponenty. V případě pevného disku je informativně uvedeno, že se jedná o SSD disk.

Tab. 1 Hardwarová specifikace

Komponenta	Výrobce	Řada	Hodnota
Procesor	Intel	Core i7 4700HQ	4 jádra, 8 vláken
Operační paměť	Kingston	DDR3	12 GB, 1600 MHz
Pevný disk	Crucial	MX500	20 GB, SSD

4.1.2 Použité knihovny

Kapitola obsahuje popis použitých knihoven v aplikaci. Některé z nich mají podobnou funkcionalitu jako je CGI a LWP::UserAgent. Avšak bylo v nejlepším zájmu aplikace využít obě. Každá z nich byla použita ve specifických případech.

Knihovna HTML::TagParser obsahovala chybu, níže je uvedena její úprava. Knihovna Data::Mining::Apriori byla též upravena.

4.1.2.1 CGI

Obstarává tvorbu HTTP požadavků a zpracování HTTP odpovědí. Obsahuje funkce pro odeslání formuláře, upload souboru, čtení a zápis cookies, manipulaci s dotazem, tvorbu a zpracování HTTP hlaviček.

4.1.2.2 Config::IniFiles

Je prostředek pro zpracování konfiguračních souborů. Konfigurační soubor se skládá ze sekcí. Každá sekce může mít několik nastavení. Sekce slouží pro seskupení nastavení i jako klíč ke skupině. K nastavením lze přistupovat prostřednictvím hashe. Později bude uveden popis konfiguračního souboru pro vlastní aplikaci.

4.1.2.3 Data::Dumper

Knihovna, která se zabývá výpisem datových struktur, jako jsou například seznamy, hashe, popřípadě jejich kombinace. Je podstatná pro fázi testování.

4.1.2.4 Data::Mining::Apriori

Obstarává implementaci algoritmu Apriori, tudíž vytváří asociační pravidla.

Knihovnu Apriori.pm bylo žádoucí omezit pouze na digramy a trigramy. Na řádku 197 je zapotřebí obohatit podmínku návratu o výraz:

```
|| $self->{largeItemsetLength} >= 3
```

Dále zakomentovat řádek 201, popřípadě 202, obsahující následující přiřazení:

```
$self->{association_rules} = undef;
```

4.1.2.5 DBI

Knihovna pro práci s databází. Připojení a CRUD operace.

4.1.2.6 Digest::MD5

Slouží pro generování hash kódu ze zadaného řetězce.

4.1.2.7 Fcntl

Slouží pro zamykání sdílených souborů mezi vlákny. Jedná se o mutex nad soubory.

4.1.2.8 File::Basename

Nástroj pro rozložení cesty k souboru na části adresář, název souboru a přípona. Podporuje všechny běžné platformy operačních systémů.

4.1.2.9 File::Slurp

Knihovna pro efektivní čtení, zápis a modifikaci celých souborů. Obsahuje metody pro načtení i zápis celého souboru jediným příkazem.

4.1.2.10 HTML::TagParser

Modul, který parsuje HTML/XHTML soubory. Obsahuje obdobné metody, jako mají DOM parsery. Základní metody jsou získání elementu dle identifikátoru, obdržení jeho atributů a další operace. Knihovna neprovádí validaci XHTML souborů.

Pro opravu chyby je zapotřebí vložit do souboru TagParser.pm níže uvedený výraz (vložit za řádek 323).

```
next unless length $attr->{key};
```

4.1.2.11 HTML::TreeBuilder

Slouží nejen parsování HTML stránky, ale i k tvorbě. Nemá obdobné rozhraní jako DOM parser. Pracuje s jednotlivými HTML elementy, jako je * *, *<li>*, *<p>* a další.

4.1.2.12 JSON

Knihovna pro práci s formátem typu JSON.

4.1.2.13 LWP::UserAgent

Implementuje webového agenta. Umožňuje vytvořit HTTP požadavky typu *GET*, *POST*, *PUT* a další. Dále definovat protokol, timeout či proxy. Agent vrací instanci HTTP odpověď.

4.1.2.14 Switch

Programový prostředek pro rozšířený rozhodovací blok.

4.1.2.15 Sys::CpuAffinity

Knihovna pro práci s informacemi o procesoru. Například počet jader.

4.1.2.16 Unicode::Normalize

Knihovna pro normalizaci textu. Například odstranění diakritiky ze zadaného řetězce.

4.1.2.17 URI

Prostředek pro práci s URI. Hlavní přínosem je vytvoření instance URI ze zadaného řetězce. Instanci URI lze následně použít pro webového agenta.

4.1.2.18 WWW::Mechanize

Modul zaměřený na procházení webových stránek. Hlavním přínosem je získat obsah webové stránky tzv. *content*. Následně lze aplikovat HTML::TagParser či HTML::TreeBuilder.

4.2 Požadavky

Kapitola popisující funkční a nefunkční požadavky aplikace.

4.2.1 Funkční požadavky

Funkční požadavky aplikace byly zvoleny:

- Stahovat recenze
- Zpracovávat stažené recenze
- Ukládat výsledky do databáze
- Nahrávat snímky čárových kódu prostřednictvím kamery mobilního telefonu
- Zobrazovat zpracované recenze

4.2.2 Nefunkční požadavky

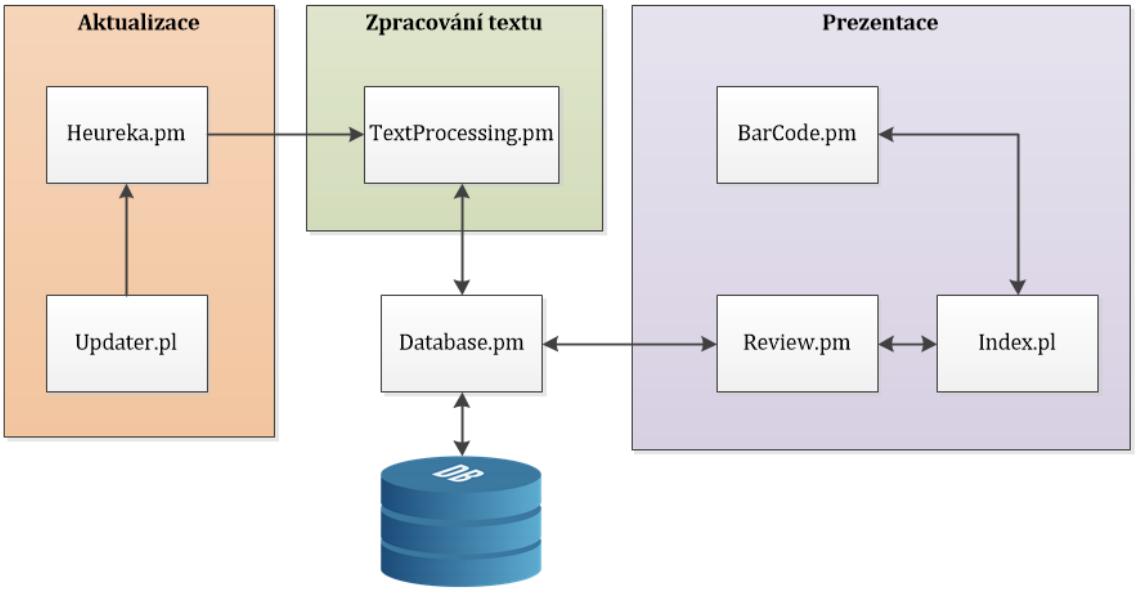
Nefunkční požadavky aplikace byly zvoleny:

- Webová aplikace
- Přívětivý vzhled
- Jednoduchá manipulace s aplikací
- Snadná integrace dalších zdrojů recenzí
- Grafické uživatelské rozhraní v češtině
- Konfigurovatelnost
- Rychlá odezva

4.3 Návrh

4.3.1 Architektura

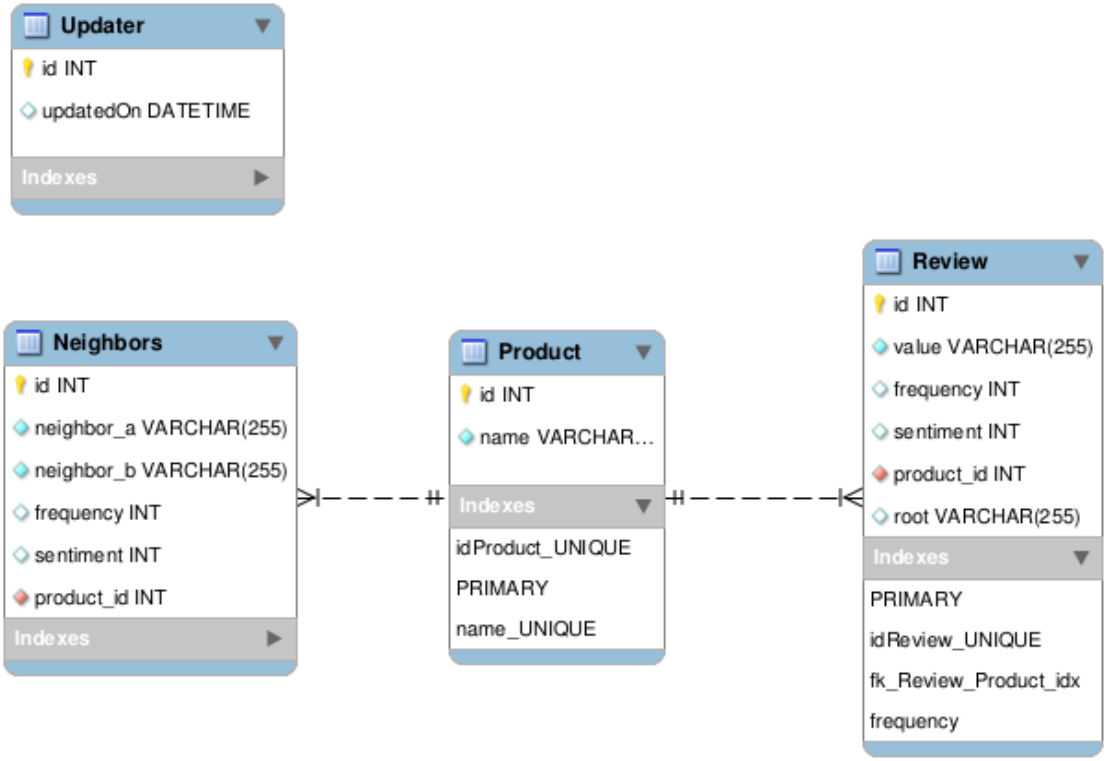
Pro automatizovaný systém sběru a prezentaci uživatelských názorů byla zvolena následující architektura. Na obrázku 9 níže jsou vyobrazeny jednotlivé části aplikace. Důraz byl kladen na rozšiřitelnost. Celá aplikace byla rozdělena do několika částí. První část je *aktualizační*, v obrázku oranžově. Obstarává stahování recenzí z definovaných zdrojů. Druhá část, v obrázku zeleně, je část *zpracování textu*. Zajišťuje extrakci relevantních dat ze stažených recenzí a uložení do databáze. Poslední část je *prezentační*, v obrázku fialově. Slouží pro zobrazení výsledků uživateli. Databáze nebyla kategorizována. Je chápana jako prvek bez kategorie, který slouží potřebám aplikace. Entity, které obsahují příponu .pm jsou Perl moduly, zatímco .pl jsou Perl skripty. V kapitole Implementace budou jednotlivé moduly i skripty objasněny podrobněji.



Obr. 9 Architektura aplikace

4.3.2 Databáze

Kapitola definuje databázi současné aplikace. V podkapitolách jsou popsány jednotlivé tabulky a indexy na konkrétních polích. Celý diagram databáze je znázorněn na obrázku 10. Databáze byla vytvořena nástrojem MySQL Workbench. Vygenerovaný skript pro vytvoření databáze se nachází v adresáři *conf/mysqlCreate.sql*. Adresářová struktura bude popsána v další kapitole.



Obr. 10 Databáze aplikace

4.3.2.1 Tabulky

Kapitola se zaobírá vytvořenými tabulkami a jejich vztahy. V nižší úrovni nečíslovaného seznamu je uveden popis polí dané tabulky.

- *Product* je centrální tabulka aplikace. Slouží k uložení stažených produktů.
    - id – primární klíč
    - name – název produktu
- *Review* je tabulka pro zpracované recenze. Jinými slovy obsahuje slova nebo sousloví, které úspěšně prošly fází preprocessing.
    - id – primární klíč
    - value – hodnota v recenzi, například slovo *výkonný*
    - frequency – počet výskytů určitých slov či sousloví ve všech recenzích u daného produktu
    - sentiment – polarita slova či sousloví
    - product_id – cizí klíč do tabulky Product, vazba 1:N (Product:Review)
    - root – kořen slova nebo zkrácená forma slova, podstatná pro shlukování

- *Neighbors* je pomocná tabulka pro zpracování dvou sousedních slov.
    - id – primární klíč
    - neighbor_a – první slovo dvojice
    - neighbor_b – druhé slovo dvojice
    - frequency – počet výskytů určité dvojice slov ve všech recenzích u daného produktu
    - sentiment – polarita dvojice
    - product_id – cizí klíč do tabulky Product, vazba 1:N (Product:Neighbors)
- *Updater* je informační tabulka. Slouží pro periodické stahování katalogu produktů a jejich recenzí.
    - id – primární klíč
    - updatedOn – datum a čas poslední iterace (stažení a zpracování recenzí)

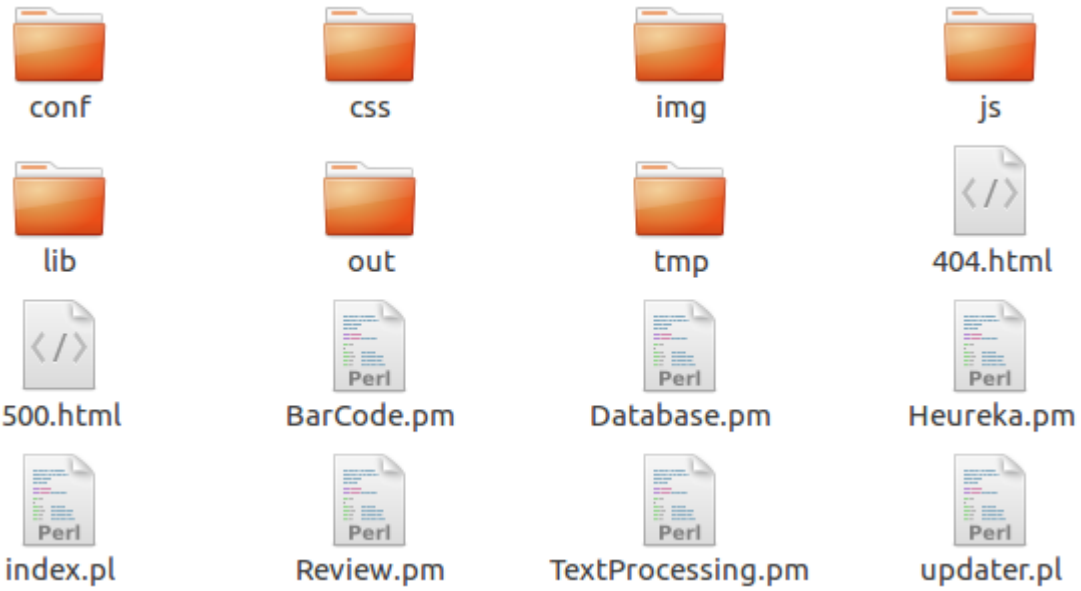
4.3.2.2 **Indexy**

Na klíčová pole byly umístěny indexy pro rychlejší zpracování dotazů. Je samozřejmostí, že jsou umístěny na každém primárním a cizím klíči. Druhotná pole, která byla opatřena indexy, jsou uvedena v následujícím seznamu. Nejprve je uvedena tabulka a následně pole.

- Product – name
- Review – frequency

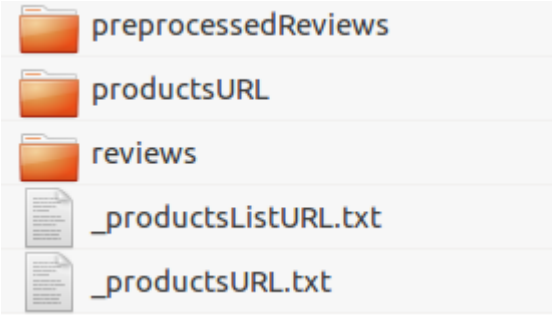
4.3.3 **Adresářová struktura**

Celá aplikace figuruje v adresáři *var/www/html/dp*. Jedná se o adresář webového serveru, který je konfigurován pro práci s Perlem. V dalším textu na něj bude pohlíženo jako na kořenový adresář aplikace. V kořenovém adresáři se nachází několik podadresářů a souborů, viz obrázek 11 níže. Současná kapitola vysvětluje význam pouze podadresářů. O jednotlivých souborech bude řečeno v kapitole Implementace.



Obr. 11 Adresářová struktura

- *conf* obsahuje konfigurační soubory aplikace. Jmenovitě lemmatization_cs.txt je lemmatizační slovník. Stopwords_cz.txt je slovník stopslov. MysqlCreate.sql je skript pro vytvoření databáze a poslední je my.ini, jakožto konfigurační soubor aplikace. O konfiguračním souboru pojednává následující kapitola Konfigurace.
- *css* se skládá z kaskádových stylů pro grafické uživatelské rozhraní.
- *img* adresář slouží pro nahrání vyfocených čárových kódu prostřednictvím kamery mobilního telefonu.
- *js* obsahuje JavaScriptové funkce pro grafické uživatelské rozhraní. Součástí je i nástroj pro word cloud od autora Davies (2015).
- *lib* adresář obsahuje použité Perl knihovny a nástroj ZBar (Brown, 2010) pro práci s čárovými kódy.
- *out* obsahuje výstupní soubory *.tsv, které budou zobrazeny prostřednictvím nástroje pro word cloud.
- *tmp* je nejobsáhlejší adresář celé aplikace. Demonstrován bude na poskytovateli recenzí Heureka. V případě přidání dalšího poskytovatele je zapotřebí ctít danou strukturu. Adresář *tmp* obsahuje nejprve podadresář s názvem poskytovatele, tedy *heureka*. Obsah adresáře heureka je zobrazen na obrázku 12. Adresář *preprocessedReviews* obsahuje již zpracované recenze, čili ty, které úspěšně prošly fází preprocessing. Součástí adresáře *productsURL* je katalog jednotlivých produktů (URL adres) s alespoň jednou recenzí. Poslední adresář *reviews* obsahuje původní stažené recenze bez jakýchkoliv úprav.



Obr. 12 Obsah podadresáře heureka

4.3.4 Konfigurace

Konfigurační soubor aplikace je umístěn v adresáři *conf/my.ini*. Soubor obsahuje sekce *GLOBAL*, *DATABASE* a *HEUREKA*. Vizuální prezentace konfiguračního souboru je umístěna v příloze práce. V následujících odstavcích jsou parametry stručně vysvětleny. Jednotlivé parametry jsou popisovány shora dolů.

Sekce *GLOBAL* konfiguruje obecné parametry aplikace. Zmíněná sekce se skládá z následujících parametrů:

- *DEBUG* slouží pro přepínání vývojového a uživatelského módu. Při hodnotě 1 se vypisuje větší množství kontrolních hlášení do konzole.
- *THREAD_COUNT* definuje kolika vlákny lze současně stahovat a zpracovávat recenze.
- *THREAD_SLEEP_INT* určuje interval pro ověření počtu aktuálně pracujících vláken.
- *PRODUCTS_PER_FILE_LIMIT* je optimalizační parametr. Po stažení katalogu produktů (URL adres) je tento soubor rozdělen na několik menších souborů, které budou obsahovat zadané množství produktů. Slouží jako příprava pro paralelní zpracování. Čím větší hodnota, tím budou vlákna zpracovávat obsáhlejší množinu dat.
- *PRODUCT_NAME_PERCENTAGE_MATCH* udává kolik procent názvu produktu použít pro vyhledání produktu v databázi aplikace. Název produktu byl získán prostřednictvím čárového kódu.
- *WORD_CLOUD_MIN_WORD_FREQUENCY* definuje zobrazení slov či sousloví, která mají alespoň zadanou minimální frekvenci ve zdrojových datech.
- *SHOW_WORD_CLOUD* zajišťuje zobrazení word cloud, který obsahuje jednotlivá slova.
- *SHOW_WORD_CLOUD_NEIGHBORS* obstarává zobrazení word cloud s dvojicemi slov.
- *UPDATE_EVERY* definuje po kolika dnech vykonávat aktualizaci recenzí. V případě hodnoty 0 se aktualizace vykonává ihned po dokončení předchozí iterace.

- *DELETE_ALL_ON_UPDATE* zajišťuje, zda se mají vymazat všechny tabulky vyjma tabulky Updater. Pokud je nastaven na 0, pak se promažou pouze tabulky Review a Neighbors.
- *STOPWORDS_PATH* určuje cestu ke slovníku obsahující stopslova.
- *LEMMATIZATION_PATH* definuje cestu ke slovníku, který obsahuje dvojice slovo a jeho lemma.
- Poslední parametr současné sekce je *ZBARIMG_HOME*. Jedná se o cestu k aplikaci třetí strany, která se využívá pro extrakci čísel z čárového kódu.

Sekce *DATABASE* obsahuje standardní parametry pro připojení k databázi a konkrétnímu schématu:

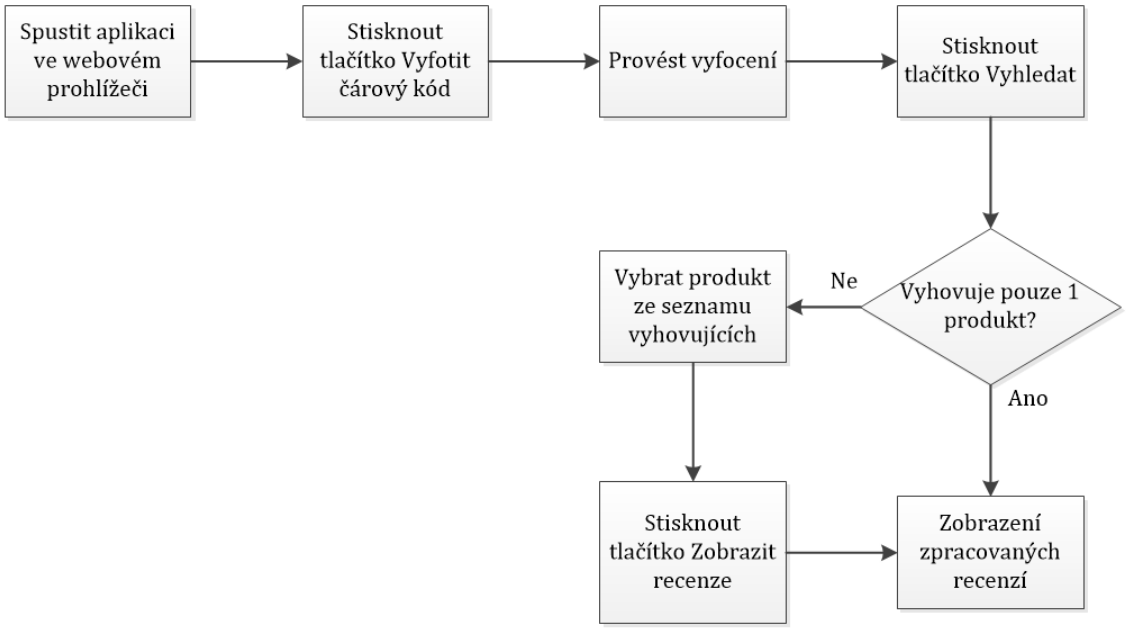
- *HOST* je IP adresa serveru s databází. Pokud je k dispozici DNS server, lze využívat i jména serverů.
- *PORT* stanovuje port, na kterém databáze obsluhuje požadavky.
- *USER* definuje uživatele pro přístup k databázi.
- *PASSWORD* je heslo uživatele pro přístup k databázi.
- *SCHEMA* konkretizuje schéma databáze, se kterým se bude pracovat.

Sekce *HEUREKA* slouží pro nastavení chování aplikace v případě konkrétního poskytovatele recenzí heureka.cz. V případě přidání dalšího zdroje recenzí je zapotřebí přidat sekci s příslušným názvem poskytovatele a vytvořit parametr URL.

- *URL* určuje adresou domovské stránky poskytovatele recenzí.
- *POPULATE_PRODUCTS_LIST* je parametr pro vývojové účely. Pokud je nastaven na hodnotu 1, pak se provede stažení všech seznamů produktů z heureka.cz.
- *POLULATE_PRODUCTS* je parametr pro vývojové účely. Pokud je nastaven na hodnotu 1, pak se provede stažení všech produktů (jejich URL).
- *POPULATE_REVIEWS* je parametr pro vývojové účely. Pokud je nastaven na hodnotu 1, pak se provede stažení všech recenzí.
- *GET_ERRORS_LIMIT* a *GET_ERRORS_SLEEP* slouží pro obsluhu výjimek v případě, že Heureka dočasně přeruší komunikaci z důvodu velkého počtu požadavků na server. *GET_ERRORS_LIMIT* definuje maximální počet opakovaných požadavků v případě přerušené komunikace. *GET_ERRORS_SLEEP* určuje dobu nečinnosti mezi opakovanými požadavky.
- *MAX_CATEGORY_PAGE* je optimalizační parametr. Určuje, do kolikáté stránky se v seznamu produktů mají prohledávat produkty v dané kategorii. Heureka má často prohlížené produkty na začátku seznamu, kde se předpokládá i existence recenzí. Od určité stránky jsou však produkty bez recenzí a zbytečně zdržují celé stahování katalogu.

4.3.5 Workflow uživatele

Pro snazší představivost je na obrázku 13 prezentován proces obdržení zpracovaných recenzí uživateli. Proces je vykonáván zleva doprava. V procesu Provést vyfocení je většinou zapotřebí nejen vyfotit, ale i potvrdit snímek. Záleží na operačním systému daného zařízení. Dále lze pozorovat, že existuje jedna alternativní cesta pro zobrazení recenzí. Pokud bude dotazu vyhovovat více než jeden produkt, je na uživateli, aby z připraveného seznamu produktů zvolil jeden sám.



Obr. 13 Workflow uživatele

4.4 Implementace

V kapitole Architektura byla nastíněna interakce mezi jednotlivými moduly a skripty. Současná kapitola popisuje elementární prvky daného schématu. Nebude popsán veškerý zdrojový kód, ale pouze vybrané oblasti. Kapitola Implementace obsahuje podkapitoly Programové moduly a Programové skripty. Všechny moduly i skripty jsou uloženy v kořenovém adresáři aplikace.

4.4.1 Programové moduly

Kapitola popisuje programové moduly. Jedná se o soubory s příponou .pm.

4.4.1.1 Heureka

Modul, který slouží pro stažení produktů a jejich recenzí z portálu heureka.cz. Kapitola bude popisovat zajímavé části metod souboru *Heureka.pm*.

Na počátku samotného souboru je uveden výčet globálních proměnných. Ve většině případů jde o inicializaci z konfiguračního souboru *my.ini*. Za konfiguračními proměnnými následují názvy interních adresářů, které by neměly být pro přehlednost modifikovány. Z důvodu jednotného tvaru mají adresářové proměnné prefix D_ a souborové F_. Jak již bylo zmíněno dříve, je žádoucí ctít adresářovou hierarchii i v případě dalšího poskytovatele.

V rámci konstruktoru se provede inicializace proměnných třídy. Prostřednictvím metody *clean* se odstraní všechny podadresáře *tmp/heureka*. Pokud v adresáři *tmp/heureka* existují soubory *_productsURL.txt* a *productsListURL.txt*, ponechají se. Následně se vytvoří požadovaná adresářová struktura.

Hlavní metodou současného modulu je *processReviews* viz obrázek 14. Jejím účelem je spustit metodu *downloadReviews*, řádek 88. Po jejím úspěšném dokončení zavolat další modul *TextProcessing*. Parametrem je adresář, ve kterém se nacházejí stažené recenze z portálu heureka.cz. Zmíněná funkcionalita se nachází na řádcích 89 až 91. Metoda *processReviews* je společně s metodou *clean* jedinými povinnými metodami, které je nutno implementovat u případných dalších datových zdrojů.

```
81 # main
82 # @params
83 #   $self - reference to object
84 # @return overall status of heureka vendor
85 sub processReviews {
86     my $self = shift;
87     print "Heureka Started\n";
88     my $downloadSuccess = downloadReviews($self);
89     if ($downloadSuccess) {
90         my $textProcessing = new TextProcessing($D_REVIEWS);
91         $textProcessing->processing();
92     } else {
93         die "Error during downloading Heureka reviews\n";
94     }
95     return 1;
96 }
```

Obr. 14 Modul Heureka – Metoda processReviews

Metoda *downloadReviews* slouží ke stažení produktů a jejich recenzí. Jedná se o velmi obsáhlou metodu. V rámci modulu *Heureka* byla tato metoda rozdělena na tři podmíněné části. Na základě konfiguračního souboru lze zmíněné části aktivovat či deaktivovat. Každá část volá další příslušné metody. Následující odstavce by měly objasnit autorovy myšlenkové pochody při tvorbě metody.

První část zajišťuje stažení všech tzv. *productsList*. Jedná se o seznamy všech produktů v dané kategorii. Například *mobilní telefony*, *televize*, *obleky* a další. Heureka má systém kategorií hierarchicky seřazen. Vhodným nástrojem pro průchod hierarchickou strukturou je rekurze. Sběr veškerých *productLists* zajišťují metody *populateTopCategoriesURL* spolu s *populateSubCategoriesURL*. *PopulateTopCategoriesURL* stáhne nejvyšší kategorie na Heurece, například *Foto*, *Mobily* a *GPS*, *Oblečení a móda* a další. V rámci sběru kategorií na nejvyšší úrovni (Top level kategorie) byly

vynechány položky *Dárky* a *Slevy*, jelikož jde o pouhé seskupení produktů z jiných kategorií. *PopulateSubCategoriesURL* má na starost průchod všemi podkategoriemi až po *productsListy*, které jsou ukládány do paměti, obrázek 15, řádek 231. Průchod je podnícen web scrapingem, čili sběr elementů, které mají určité vlastnosti. Zde konkrétně CSS třídu *cat-list*, viz obrázek 15, řádek 214. V některých podkategoriích může existovat smyčka. Příklad *Herní zařízení -> Příslušenství -> Ovladače -> Herní zařízení*. Jako protiopatření bylo zvoleno trasování již navštívených kategorií. Pokud program narazí na kategorii, kterou již prozkoumal, přeskočí ji. Názorně na obrázku 15, řádek 225 a 226. Po úspěšném sběru všech seznamů produktů se provede uložení do souboru s názvem *_productsListURL.txt*, pro případné pozdější využití.

```
203 # populate subcategories
204 # @params
205 #   $self               - reference to object
206 #   $parentCategeoryURL - url of parent category
207 # @return state of population subcategory
208 sub populateSubCategoriesURL {
209     my ($self, $parentCategoryURL) = @_;
210     my $m = WWW::Mechanize->new();
211     $m->get($parentCategoryURL);
212     my $content = $m->content();
213     my $html = HTML::TagParser->new($content);
214     my @ulList = $html->getElementsByClassName("catlist");
215 
216     if (scalar @ulList == 0) {
217         @ulList = $html->getElementsByClassName("cat-list");
218     }
219 
220     for my $ul (@ulList) {
221         my @liList = @{$ul->childNodes()};
222         for my $li (@liList) {
223             my $href = $li->firstChild()->firstChild()->getAttribute("href");
224             # cycle protection
225             $self->{"productsCategoryURL"}->{$href}++;
226             next if $self->{"productsCategoryURL"}->{$href} > 1;
227             if (isProductList($self, $href)) {
228                 $href =~ /(^http:\/\/.+\.cz\/)/;
229                 $href = $1;
230                 next if !$href;
231                 $self->{"productsListURL"}->{$href} = 1;
232             } else {
233                 populateSubCategoriesURL($self, $href);
234             }
235         }
236     }
237     return 1;
238 }
```

Obr. 15 Modul Heureka – Metoda populateSubCategoriesURL

Druhá část obsluhuje stažení všech produktů s alespoň jednou recenzí. Jinými slovy vytvoření katalogu produktů. Zmíněna funkcionalita byla realizována prostřednictvím stažených seznamů produktů (*productLists*, první část). Není zapotřebí neustále dokola stahovat seznamy produktů. Nejdříve se testuje, zda existují seznamy produktů v paměti. V případě, že neexistují, načtou se ze souboru. Pokud ne-

existuje ani soubor, je vypsána chybová hláška. Pro úplnost je však zapotřebí deaktivovat první část pomocí konfiguračního parametru *POPULATE_PRODUCTS_LIST*, jinak stažení seznamů proběhne. Následuje sekvenční průchod přes všechny produkty v daném seznamu produktů. Bylo by zbytečné stahovat všechny. U každého produktu je evidován počet recenzí. CSS třída elementu, který zmíněnou informaci vlastní, se nazývá *review-count*, obrázek 16, řádek 285. Na Heurece existují dva typy seznamů produktů. První pokrývá všechny kategorie až na *módu*. Móda má odlišnou HTML strukturu. Bylo zapotřebí upravit parsovaní pro oba typy, obrázek 16, řádek 286 až 307. Móda neobsahuje element určující, zda má produkt nějakou recenzi, což znamená, že musí být staženy všechny produkty z této kategorie. V dřívějších kapitolách byl zmíněn DoS útok, čili blokování služby z důvodu velkého množství požadavků na server. Současnou obranou proti timeoutu serveru je opakovaní dotazu společně se sleepem, obrázek 16, řádek 261 až 270. Počet opakování a trvání sleep je určeno konfigurací, konkrétně parametry *GET_ERRORS_LIMIT a GET_ERRORS_SLEEP*. Pro vyšší výkonnost celkového sběru produktů byla druhá část paralelizována na konfigurovatelný počet vláken, parametr *THREAD_COUNT*. Každý zpracovávaný seznam produktů si vytvoří vlastní dočasný soubor pro URL adresy vyhovujících produktů. Krátce předtím než vlákno ukončí svou práci, zapíše obsah svého dočasného souboru do hlavního souboru, který shlukuje všechny produkty. Jelikož se jedná o proměnnou (soubor) mezi vlákny, bylo zapotřebí zajistit zámek na soubor, aby nedošlo ke kolizi. Hlavní soubor je pojmenován jako *_productsURL.txt*.

```
260         # DoS protection
261         my $connected = eval {
262             $m->get($productListURL . "?f=" . $f);
263             1
264         };
265         if (!$connected) {
266             $getERRCounter++;
267             last if $getERRCounter == $GET_ERRORS_LIMIT;
268             sleep $GET_ERRORS_SLEEP;
269             next;
270         };
271
272
273         $content = $m->content();
274         $html = HTML::TagParser->new($content);
275         $currPage = $html->getElementsByClassName("curr");
276
277         last if (!$currPage);
278         if ($currPage->innerText() =~ /([0-9]+)/) {
279             $currPage = $1;
280         }
281
282         last if ($f > $currPage || $f > $MAX_CATEGORY_PAGE);
283
284         # review-count class is only on regular sites
285         my $reviewCountClass = $html->getElementsByClassName("review-count");
286         if ($reviewCountClass) {
287             # regular sites
288             my @reviewCountDivs = $html->getElementsByClassName("review-count");
289             for my $reviewCountDiv (@reviewCountDivs) {
290                 my $href;
291                 my $connected = eval {
292                     $href = $reviewCountDiv->firstChild()->getAttribute("href");
293                     1
294                 };
295                 next unless $connected;
296                 print TEMP $href."\n";
297             }
298
299         } else {
300             # fashion sites
301             my @productsCon = $html->getElementsByClassName("product-image");
302             for my $productCon (@productsCon) {
303                 my $href = $productCon->getAttribute("href");
304                 print TEMP $href."\n";
305             }
306
307         }
```

Obr. 16 Modul Heureka – Část metody populateProductsURL

Poslední třetí část metody *downloadReview* slouží ke stažení samotných recenzí. Z předchozí části byl pořízen soubor obsahující všechny vyhovující produkty. Při opětovném spuštění není zapotřebí opětovně stahovat katalog produktů. Avšak je zapotřebí deaktivovat konfigurační parametr *POPULATE_PRODUCTS*. Stejně jako v případě předchozí části se jedná o proces náročný, proto byl též paralelizován. Nejprve bylo zapotřebí připravit data pro paralelní zpracování. Hlavní soubor *_productsURL.txt* byl rozdělen menší soubory o konfigurovatelném množství produktů, parametr *PRODUCTS_PER_FILE_LIMIT.* Následně se vykoná stažení všech recenzí dle pořízených produktů. Vlastní stažení obstarává metoda *populateReviews*. Jako parametr obdrží cestu k souboru, který vznikl rozdělením hlavního souboru *_productsURL.txt*. Samotné recenze mohou být pozitivní, negativní nebo bez udání polarity čili neutrální. Obrázek 17 prezentuje část metody *populateReviews*. Zobrazený kód

slouží k rozlišení polarit recenzí dle CSS tříd plus a mínus. Neutrální recenze jsou umístěny pouze v HTML tagu odstavce ( ). Spolu s příslušnou polaritou jsou recenze zapsány do souboru. Pro každý produkt je vytvořen separátní soubor s recenzemi. Na začátku každého souboru je vždy uveden název produktu a jeho URL pro případné další zpracování.

```
383             foreach my $review ( @review_list ) {
384                 my @review_plus = $review->find_by_attribute("class", "plus");
385                 foreach my $p ( @review_plus ) {
386                     print TEMP "P:";
387                     my @ul = $p->look_down(_tag => 'li');
388                     foreach my $li ( @ul ) {
389                         print TEMP $li->as_text (), ".";
390                     }
391                 print TEMP "\n";
392                 }
393
394                 my @review_minus = $review->find_by_attribute("class", "minus");
395                 foreach my $m ( @review_minus ) {
396                     print TEMP "N:";
397                     my @ul = $m->look_down(_tag => 'li');
398                     foreach my $li ( @ul ) {
399                         print TEMP $li->as_text (), ".";
400                     }
401                 print TEMP "\n";
402                 }
403
404                 my @review_text = $review->look_down(_tag => 'p');
405                 foreach my $tx ( @review_text ) {
406                     print TEMP "R:";
407                     print TEMP $tx->as_text ();
408                 print TEMP "\n";
409                 }
410             }
```

Obr. 17 Modul Heureka – Část metody populateReviews

4.4.1.2 TextProcessing

Modul *TextProcessing* je druhý nejobsáhlejší, ale předpokládá se, že se bude rozšiřovat nejvíce. Z názvu je patrné, že se jedná o modul, který zajišťuje zpracování textu. V úvodu souboru je provedena inicializace globálních proměnných. Za zmínku stojí hlavně slovníky pro stopslova (Bouge, 2011) a lemmatizaci (Měchura, 2016).

Současný modul je společný pro všechny potenciální poskytovatele recenzí. V kapitole Heureka.pm bylo zmíněno, že po úspěšném stažení recenzí vytvoří instanci modulu *TextProcessing* a parametrem předá adresář, kde se recenze vyskytují. Podstatnými metodami jsou převážně *processing*, *preprocessing* či *preprocessFile*.

Metoda *processing* je hlavní metoda, která spouští vše ostatní. Je zobrazena na následujícím obrázku 18. Pokud by byla zapotřebí přidat další metoda pro zpracování textu, bude umístěna kamkoliv pod řádek 54, kde se vykonávána metoda *preprocessing*. Lze pozorovat, že je řádek 55 zakomentovaný. Funkcionalita asociačních pravidel je úplná, avšak je zapotřebí důkladnější preprocessing. Konkrétně zapojení metody POS, viz kapitola Natural language processing. Pomocí POS by se označila slova slovními druhy. Následná asociační pravidla by šlo aplikovat pouze na určité

slovní druhy. Bez POS se algoritmus uplatňuje na úplnou množinu všech slov (všech slovních druhů). Algoritmus je časově i paměťově velmi náročný.

```
45 # main
46 # @params
47 #   $self - reference to object
48 # @return overall status of text processing
49 sub processing {
50     my $self = shift;
51     my $db;
52     print "TextProcessing Started\n";
53 
54     $self->preprocessing();
55     #$self->associationRules();
56     $self->neighborsTwo();
57 
58     $self->{"endTime"} = localtime();
59     print "START: " . $self->{"startTime"} . "\n";
60     print "END: " . $self->{"endTime"} . "\n";
61     print "TextProcessing Complete\n";
62     return 1;
63 }
```

Obr. 18 Modul TextProcessing – Metoda processing

Klíčovou metodou pro jakoukoliv další práci je *preProcessing*, respektive *preprocessFile*. Vstupem metody je konkrétní soubor s recenzemi určitého produktu. Výstupem je předzpracovaný soubor. V následujícím textu bude popisován obrázek 19. Na začátku každého souboru s recenzemi je název produktu a URL, odkud byly recenze pořízeny. Je zapotřebí zmíněné prvky odfiltrovat. Vše je na řádcích 109 až 120. Při vytváření produktu, řádek 113, je v rámci metody *insertProduct* nejprve kontrolováno zda již produkt neexistuje. Není tedy umožněno vkládat více produktů se stejným názvem. Je možné budovat katalog produktů a stahovat pouze recenze, pokud se neprovede výmaz tabulky *Product*. Vše je závislé na konfiguraci, konkrétně na parametru *DELETE_ALL_ON_UPDATE*. Na řádku 122 je extrahována polarita, která byla asociována k dané recenzi. Na řádcích 125 až 127 se provede nahrazení stopslov, čísel, speciálních znaků a více mezer vedle sebe. Jádro metody *preProcessing* je mezi řádky 134 a 147. Nejprve se provede redukce slov, které nesplňují minimální délku 2. Bylo zapotřebí slova převádět na slova bez diakritiky. Slova s diakritikou jsou uložena přes více znaků. Metoda *length* by selhávala. Následuje provedení lemmatizace, řádek 139. Na následujícím řádku je lemma dále posláno na stemming. Výsledné slovo je uloženo do předzpracovaného souboru. Na řádku 142 se počítá frekvence určitého slova s určitým kořenem. Frekvence udává určitou váhu každému slovu. S frekvencí slov hodně operuje modul *Review*.

```
    # original non-preprocessed data
    open (REVIEWS, "<", $reviewPath) or die "Can't open for read: $!";
    # containing preproccesed data
    open (PREPROCESSED, ">", $self->{"preprocessedDir"} . basename($reviewPath));

    while (<REVIEWS>) {
        chomp;
        $_ =~ s/^\s+//g;                                    # no leading white spaces
        next unless length;

        if ($_ =~ /^#productName=(.*)/) {
            $self->{"productName"} = $1;

             # PRODUCT
            $productId = $db->insertProduct($self->{'productName'});
            print PREPROCESSED "#productId=" . $productId . "\n";
            next;
        }
        if ($_ =~ /^#productURL=(.*)/) {
            $self->{"productURL"} = $1;
            next;
        }

        my $reviewType = uc substr $_, 0, 1;
        my $line = lc substr $_, 2;

        $line =~ s/$stopwordsPattern/ /g;                   # stopwords
        $line =~ s/[\$#@~!&*()\[];\.,:?^`\\\/0-9+-]+/ /g;  # numbers, special chars
        $line =~ s/ +/ /g;                                  # more spaces

        my $unacWord;                                       # word without diacritics
        my $root;                                           # root form of word
        my $hasProperWord = 0;
        my $lemma;                                          # lemma of word

        for my $word (split /\s+/, $line) {
            next if !$word;
            $unacWord = unac_string('UTF-8', $word);
            next if length $unacWord <= 2;

            $lemma = $self->lemma($word);
            $root = $self->stem($lemma);

            $wordsFrequency{$word . "_" . $root}++;

            print PREPROCESSED $lemma . " ";
            $hasProperWord = 1;
        }
        print PREPROCESSED "\n" if $hasProperWord;
    }
    close REVIEWS or die "Cannot close $reviewPath: $!";
    close PREPROCESSED;
```

Obr. 19 Modul TextProcessing – Část metody preProcessing

V předešlém textu byly zmíněny slova stemming a lemmatizace. Pro stemming nebyla využitá knihovna třetích stran. Jediná použitelná a podporující češtinu nevracela adekvátní výsledky. Byl vytvořen vlastní stemmer, který je jednoduchý a svojí funkcionalitou plně dostačující. Pro lemmatizaci byl využit lemmatizační slovník od autora Měchura (2016).

Byla vytvořena metoda, která realizuje asociační pravidla. Knihovna, která řeší daný algoritmus, byla upravena, aby generovala pouze digramy a trigramy. Jelikož nejsou asociační pravidla v práci využita, nebudou zde důkladněji popisována.

4.4.1.3 BarCode

Kapitola objasní práci s modulem *BarCode*. Slouží pro extrakci typu a čísel čárového kódu, který je dodán ve formě fotografie. Obsahuje i funkcionalitu pro zjištění jména

produktu prostřednictvím extrahovaných informací. Modul je využíván skriptem *Index*. Uvedený skript vytváří instanci *BarCode*. Parametrem mu posílá cestu k fotografii s čárovým kódem. Stejně jako u všech předchozím modulů je na začátku souboru BarCode.pm umístěna sekce s globálními, konfiguračními proměnnými.

Pro samotnou extrakci informací je využit nástroj třetích stran s názvem *ZBar* od autora Brown (2010). Jedná se o jednoúčelovou řádkovou aplikaci. Je zajištěna podpora pro Windows, Linux a částečně pro iOS. Pro spuštění ZBar je zapotřebí cesta k domovskému adresáři nástroje. Dále jako parametr cesta k fotografii s čárovým kódem. Cesta k domovskému adresáři nástroje je obsluhována konfiguračním parametrem *ZBARIMG_HOME*. V případě linuxových distribucí se cesta ponechává prázdná. Metoda, která zajišťuje extrakci zmíněných informací, byla nazvána *extract*. Obsahuje zpracování čárového kódu včetně několika stavů. Požadovaný je jeden čárový kód a jeden typ (návratová hodnota 1). Jsou však zpracovány i informace v případě, že je na fotografii více čárových kódů (návratová hodnota -1). Popřípadě neexistující čárový kód (návratová hodnota 0). Všechny zmíněné stavy jsou prezentovány uživateli.

Následuje identifikace produktu dle extrahovaných informací. Existuje několik databází čárových kódu, ale jsou ve velké většině neúplné nebo chybné. Jako nejefektivnější řešení se ukázalo využít vyhledávače společnosti Google. První odkaz ve velké většině identifikuje produkt dle zadaných informací, čili dle typu čárového kódu a samotné číselné sekvence. Celý následující text odstavce bude pracovat s obrázkem 20, proto se na něj autor nebude znovu odkazovat. Konstrukce dotazu pro vyhledávač je uvedena na řádku 70, popřípadě na řádcích 66 a 67. Ukázalo se, že lepší výsledky produkuje dotaz bez typu čárového kódu, proto jsou řádky 66 a 67 zakomentovány. Zpracování prvního odkazu je na řádcích 78 až 85. Následuje čištění názvu, jelikož obsahuje i irelevantní znaky. Vše na řádcích 88 až 90. Celá metoda *identifyProduct* končí vrácením jména produktu.

```perl
# find product with extracted code
# @params
#   $self - reference to object
# @return name of product
sub identifyProduct {
    my $self = shift;
    my $query;

    #with barcodeType
    #$query = 'http://www.google.com/search?q=' . $self->{"barcodeType"}
    #    . '+' . $self->{"barcode"};

    #without barcodeType
    $query = 'http://www.google.com/search?q=' . $self->{"barcode"};

    # send request with code
    $m = WWW::Mechanize->new();
    $m->get($query);
    $content = $m->content();

    # manage results
    my $responseTree = HTML::TreeBuilder->new_from_content($content);
    my @responseList = $responseTree->find_by_attribute("class", "r");
    if (scalar @responseList == 0) {
        print "Product with barcode: " . $self->{"barcodeType"} . ":"
            . $self->{"barcode"} . " NOT IDENTIFIED!!!\n" if $DEBUG;
        return 0;
    }
    my $productName = $responseList[0]->as_text();

    # clear product name
    $productName =~ s/$self->{"barcodeType"}|$self->{"barcode"}|[^A-Za-z0-9 ]+//ig;
    $productName =~ s/^\s+|\s+$//g;
    $productName =~ s/ +/ /g;

    print "Product with barcode: " . $self->{"barcodeType"} . ":"
        . $self->{"barcode"} . " IDENTIFIED as " . $productName . "\n" if $DEBUG;
    $self->{"productName"} = $productName;

    return $self->{"productName"};
}
```

Obr. 20 Modul BarCode – Metoda identifyProduct

4.4.1.4 Database

Je modul, který slouží výhradně jako rozhraní k databázi pro ostatní moduly aplikace. Modul obsahuje nejen základní sadu dotazů pro CRUD operace, ale i několik specifických.

V konstruktoru je provedena inicializace proměnných potřebných pro přístup k databázi. Jmenovitě *host*, *schema*, *user* a *password*. Všechny jsou ovlivněny konfiguračním souborem.

Pro prací s jakoukoliv metodou z modulu *Database* je nejprve zapotřebí vykonat následující kroky:

1. Vytvořit instanci modulu Database.
2. Zavolat metodu *openDBconnection*.
3. Zavolat požadovanou metodu.
4. Zavolat metodu *closeDBconnection*.

Zajímavou metodou je *findReviewsByProductIdAndMinWordFrequency*. Slouží pro získání dat k prezentaci. Je zapotřebí získat všechny recenze (slova) a jejich frekvenci. Frekvenci je však zapotřebí sčítat v případě, že existují různá slova, která mají společný kořen. Podstatná je volba kandidátního slova, které bude reprezentovat celou skupinu. Jako kandidátní slovo bylo vybráno slovo s nejvyšší frekvencí v dané skupině. Dotaz, který požadovanou funkcionalitu zajišťuje, se nachází na obrázku 21, řádky 114 až 121. Hodnota *subQuery* představuje hodnotu kandidátního slova a *sumFreq* sečtenou frekvenci v rámci skupiny.

```
108 sub findReviewsByProductIdAndMinWordFrequency {
109     my ($self, $productId, $minWordFrequency) = @_;
110     my $sth;
111     my $query;
112     my %words;
113
114     $query = "SELECT (SELECT Value FROM Review
115                     WHERE product_id = RP.product_id and Root = RP.Root
116                     ORDER BY Frequency desc
117                     limit 1) as subQuery, sum(Frequency) as sumFreq
118                 FROM Review as RP
119                 WHERE product_id = ?
120                 GROUP BY Root, subQuery
121                 HAVING sumFreq >= ?";
122
123     $sth = $self->{"connection"}->prepare($query);
124     $sth->execute($productId, $minWordFrequency);
125
126     while (my $row = $sth->fetchrow_hashref) {
127        $words{$row->{subQuery}} = $row->{sumFreq};
128     }
129     $sth->finish();
130
131     return \%words;
132 }
```

Obr. 21 Modul Database – Metoda findReviewsByProductIdAndMinWordFrequency

Další pozoruhodnou metodou je *getPossibleProductList*. Existuje problém s párováním názvu produktu. Název produktu byl získán prostřednictvím čárového kódu, viz kapitola BarCode. Není však zaručeno, že navrácený název bude obsahovat stejnou posloupnost znaků jako název produktu u některého poskytovatele recenzí. V současném řešení se použije konfigurovatelná procentuální shoda názvu produktu, který byl získán z čárového kódu. Jako algoritmus byla použita Levenshteinova vzdálenost. LIKE operátor by byl příliš otupělým řešením. Samotná funkce je psána prostřednictvím PL/SQL. Je uložena v databázových funkcích. Parametr udávající procentuální hodnotu je *PRODUCT_NAME_PERCENTAGE_MATCH.*

4.4.1.5 Review

Slouží jako prostředník mezi skriptem *Index* a modulem *Database*. V architektuře MVC (Model-view-controller) by se jednalo o controller. Více o MVC architektuře se

lze dočíst v publikaci od McGovern (2004). Účelem modulu *Review* je připravit zpracovaná data k prezentaci pro uživatele. Na počátku souboru je výčet globálních proměnných. Za zmínku stojí převážně konfigurační parametr pojmenován jako *WORD_CLOUD_MIN_WORD_FREQUENCY*. Význam parametru je uveden v kapitole Konfigurace. Následující odstavce popisují významnější metody.

Jediným explicitním parametrem konstruktoru je název produktu. Samotný název byl získán prostřednictvím modulu *BarCode*. Z důvodu snazší manipulace s produktem je ze zadaného názvu získán jeho identifikátor, čili id produktu. Id produktu je podstatné pro následující metody.

Klíčové metody jsou bez pochyby *getValidReviews* a *getValidReviewsNeighbors*. Obě metody mají společný účel připravit data. Avšak každá volá jinou metodu z modulu *Database*. Čili připravuje data k prezentaci odlišným způsobem. Metoda *getValidReviews* připravuje data dle jednotlivých slov, zatímco *getValidReviewsNeighbors* dle sousedních dvojic. Pro zobrazení připravených dat slouží nástroj word cloud od autora Davies (2015). Aby nástroj fungoval adekvátním způsobem, je zapotřebí připravená data uložit do souboru s určitou příponou a strukturou. Přípona souboru je .tsv. V prvním řádku souboru je uvedena hlavička, která slouží jako klíč. V současné aplikaci hlavička obsahuje hodnoty *VALUE* a *FREQUENCY*. Mezi zmíněnými hodnotami je vložen znak tabulátor (mezera není chápána jako oddělovač). *VALUE* představuje slovo či dvojici slov, záleží na použité metodě, viz klíčové metody výše. *FREQUENCY* udává počet výskytů daného slova či dvojice slov.

4.4.2 Programové skripty

Kapitola popisuje programové skripty. Jedná se o soubory s Updater.pl a Index.pl. Dále je uveden stručný popis grafického uživatelského rozhraní.

4.4.2.1 Updater

Je aktualizační skript, který je časově řízen. Spuštěn je vždy, když uplyne konfigurovatelná doba, konfigurovatelný parametr *UPDATE_EVERY*. Nejprve se provede vymazání konfigurovatelné části databáze, parametr *DELETE_ALL_ON_UPDATE*. Následně se spustí proces pro stažení a zpracování recenzí z definovaných zdrojů. Pokud by došlo k přidání dalšího zdroje, je zapotřebí vytvořit instanci po vzoru modulu *Heureka* a zavolat metodu *processReviews*, která je povinná pro každý zdroj recenzí. Po zpracování všech recenzí se zapíše datum a čas vykonané aktualizace do tabulky *Updater*.

4.4.2.2 Index

Obstarává popis skriptu, který definuje grafické rozhraní. V úvodu souboru je uveden výčet globálních proměnných spolu s jejich inicializací. Skript obsahuje dvě podstatné metody *uploadFile* a *displayWordCloud*.

Metoda *uploadFile* zajišťuje nahrání vyfoceného čárového kódu do definovaného adresáře. V aplikaci se jedná o adresář *img/*, viz kapitola Adresářová struktura.

Metoda *displayWordCloud* obstarává zobrazení dvou typů word cloud. V kapitole Review bylo zmíněno, že existují metody zaměřené na přípravu dat pro word

cloud, obsahující jednotlivá slova a dvojice slov. Na základně parametrů *SHOW_WORD_CLOUD* a *SHOW_WORD_CLOUD_NEIGBORS* lze vynutit, které word cloud budou zobrazeny. Je samozřejmé, že můžou být zobrazeny oba současně. Může nastat i případ, že je nastaveno zobrazení obou typů, ale zobrazen není žádný. Zmíněná situace nastane v případě, že metody, které připravovaly data, nevrátily jediný záznam k zobrazení. Pokud jsou navráceny alespoň nějaká data k zobrazení, potom je do proměnných *$pathToWords* nebo *$pathToWordsNeigbors* (záleží na typu word cloud) uložena cesta k vytvořeným souborům s příponou .tsv. O tvorbě souboru *.tsv bylo řečeno v kapitole Review. Zmíněné proměnné jsou následně využity v části s JavaScriptem. JavaScriptová část je umístěna ve spodní části skriptu *Index*.

Stěžejní část skriptu *Index* je zobrazena na obrázku 22. Podmínkou je nahraná fotografie s čárovým kódem, řádek 149. Následně se vytvoří instance modulu *BarCode* spolu s extrakcí výsledného kódu. O výsledných kódech extrakce bylo psáno v kapitole BarCode, metoda *extract*. Jediný vyhovující stav pro zobrazení word cloud je 1. Následně se provede identifikace jména produktu, řádek 159. S identifikovaným jménem se volá metoda *getPossibleProductList*, která byla popsána v kapitole Database. V případě, že je navrácen seznam s více produkty, je zapotřebí nechat uživatele rozhodnout. Zmíněná funkcionalita je na řádcích 163 až 178. Na základě volby uživatele je posléze zobrazen word cloud. Pokud však metoda *getPossibleProductList* vrátí jediný produkt, dojde k přímému zobrazení word cloud se zpracovanými recenzemi, řádky 179 a 180. Pokud není nalezen vyhovující produkt nebo modul *BarCode* nevrátil požadovaný stav, jsou zobrazeny příslušné informační hlášky.

```
149 if ($barcodeImg) {
150     my $productName;
151 
152     $barcodeImg = $UPLOAD_DIR."/".$barcodeImg;
153 
154     my $barCode = new BarCode($barcodeImg);
155     my $resultCode = $barCode->extract();
156 
157     switch ($resultCode) {
158         case 1 {
159             $productName = $barCode->identifyProduct();
160             if ($productName) {
161                 my $db = Database::new();
162                 $db->openDBConnection();
163                 my @possibleProducts = $db->getPossibleProductList($productName);
164                 $db->closeDBConnection();
165                 if (scalar @possibleProducts > 1) {
166                     $html .= "<h3>Zvolte produkt</h3>";
167                     $html .= "<form class='m-t' action='index.pl' method='post'>";
168                     $html .= "<div class='col-sm-12'>";
169                     my $i;
170                     foreach my $possibleProduct (sort @possibleProducts) {
171                         $i++;
172                         $html .= "<div class='radio' style='text-align:left'>";
173                         $html .= "<label><input type='radio' name='productListValue' value='"
174                             . $possibleProduct  . "' required='true' />" . $possibleProduct . "</label>";
175                         $html .= "</div>";
176                     }
177                     $html .= "</div><input type='submit' class='btn btn-success block
178                         full-width m-b' value='Zobrazit recenze'></form>";
179                 } elsif (scalar @possibleProducts == 1) {
180                     displayWordcloud(shift(@possibleProducts));
181                 } else {
182                     $html .= productWithoutReviews($productName);
183                 }
184 
185             } else {
186                 $html .= productNotFound();
187             }
188         }
189         else {
190             $html .= barcodeNotIdentified();
191         }
192     }
193 }
```

Obr. 22 Skript Index – Stěžejní část skriptu

Samotný HTML kód je inspirován frameworkem (WrapBootStrap, 2016). Použitá šablona je nazývána jako *Inspinia*. Šablona zajišťuje responzivní design aplikace.

4.4.2.3 Grafické uživatelské rozhraní

Kapitola volně navazuje na programový skript *Index*. Současná kapitola popisuje prezentační stránku aplikace. Využívá proces, který byl znázorněn na diagramu Workflow uživatele. Na obrázku 23 níže je uvedena prezentační vrstva celé aplikace. Vlevo je výchozí vzhled formuláře, se kterým bude uživatel pracovat nejdříve. Uprostřed je uveden seznam produktů v případě, že dotazu vyhovělo více produktů, viz kapitola Index metoda *getPossibleProductList*. Uživatel následně zvolí právě jeden. V případě, že vyhoví jediný produkt, nebude prostřední část obrázku uživateli zobrazena. Na pravé straně obrázku jsou prezentovány výsledky. Nejvýše je uveden název produktu. Následuje word cloud s individuálními slovy a word cloud s dvojicemi slov. V patičce je uveden autor práce spolu s univerzitou a rokem vydání.

Obr. 23 Skript Index – prezentace

4.5 Optimalizace

V rámci aplikace bylo zapotřebí velmi vhodně volit algoritmy a promýšlet, jakým způsobem pracovat s daty. Zda je ponechávat v operační paměti či v souborech na pevném disku. V závorkách je uveden soubor, kterého se optimalizace týkala.

Všechny procesy s vysokou časovou náročností byly paralelizovány. Jmenovitě se jedná o stažení katalogu produktů a jejich recenzí (Heureka.pm). Dále byla paralelizována část pro předzpracování textu (TextProcessing.pm).

V aplikaci byla hojně využívána práce se soubory. Všechny paralelní procesy pracují se soubory pro snížení využití operační paměti.

Vznikl problém s pojmenováním souborů. Nejprve byla použita URL s escapovanými znaky. Některé URL byly však příliš dlouhé. Problém byl vyřešen prostřednictvím hashování URL daného produktu. Všechny názvy souborů nyní dosahují stejné délky (Heureka.pm a TextProcessing.pm).

Nebylo zapotřebí znovu mazat a ukládat všechny produkty při každé aktualizaci. Byl vytvořen konfigurovatelný systém, který umožňuje inkrementálním způsobem budovat katalog produktů. V rámci pořízení produktů bylo dodatečně zajištěno, že se získávají pouze produkty s alespoň jednou recenzí (Heureka.pm).

Byla zajištěna určitá obrana proti timeoutu ze strany serveru. Bez ní by bylo mnoho vláken předčasně ukončeno a celkový proces by přišel o značnou část dat (Heureka.pm).

Z důvodu maximálního urychlení výběru dat z databáze byla klíčová pole indexována, viz kapitola Database.

Grafické uživatelské rozhraní obdrželo optimalizaci formou responzivního designu a podpory všech moderních prohlížečů (Index.pl).

Celkovou optimalizaci lze navíc vylepšovat testováním ideálních hodnot parametrů v konfiguračním souboru vzhledem k určitému hardwaru.

4.6 Testování

Pro testování čárových kódů s následnou identifikací jména produktu byly využity běžně dostupné čárové kódy. Nejčastěji *potraviny*, *čaje* a *medikamenty*.

V rámci testování stahování a zpracování recenzí byly použity kontrolní výpisy, které byly porovnány s očekávanými výsledky. Kontrolní výpisy lze aktivovat parametrem *DEBUG*.

Testování word cloud probíhalo na produktech, které obsahují větší množství recenzí, přibližně 100 a více. Nejčastěji testované byly Microsoft Xbox One 500 GB a Apple iPhone 5S 16 GB. Jelikož pro oba zmíněné produkty existuje více variant (například různé paměťové kapacity), byly vhodné i pro testování seznamu vyhovujících produktů, ze kterého uživatel zvolí právě jeden, viz obrázek 23 uprostřed.

Grafické uživatelské rozhraní bylo testováno na prohlížečích *Google Chrome*, *Opera*, *Mozilla Firefox*, *Microsoft Edge* a *Internet Explorer*. Všechny uvedené prohlížeče byly aktualizovány na poslední dostupnou verzi (prosinec 2016).

4.7 Dokumentace

Každá netriviální část kódu byla opatřena komentářem. Před každou složitější nebo významnější metodou byl uveden komentář, který obsahuje nejprve stručný účel dané metody, následně výčet vstupních parametrů a nakonec návratovou hodnotu. Výčet parametrů je umístěn pod klíčové slovo *@params*. O metodu objektu se jedná v případě, že je uveden parametr *$self*, čili reference na objekt. Návratová hodnota je psána za klíčovým slovem *@return*.

4.8 Nasazení

Kapitola popisuje nasazení na dedikovaný server. Uvádí posloupnost kroků, které je potřeba vykonat pro zprovoznění aplikace. Uvedené části byly použity v rámci vývojového a testovacího prostředí. Odlišná distribuce operačního systému nebo databáze nebyla testována.

4.8.1 Operační systém

Operační systém Ubuntu Linux lze získat prostřednictvím hypertextového odkazu pod textem.

```
http://www.ubuntu.cz/stahnout
```

Vytvořit uživatele, například *mendelu* s heslem *mendelu*. Uživateli mendelu přiřadit práva root, viz příkazy níže.

```
visudo
# User privilege specification
root ALL=(ALL:ALL) ALL
mendelu ALL=(ALL:ALL) ALL
```

4.8.2 Webový server

Podrobný popis konfigurace Apache je uveden v příloze.

4.8.3 Zdrojové kódy

Před samotným stažením je nutno vstoupit do adresáře, ve kterém se bude aplikace nacházet, zde */var/www/html*. Následně stáhnout zdrojové kódy z Git repozitáře. Po spuštění příkazu *git clone* je zapotřebí zadat heslo pro uživatele s uživatelským jménem *uzivatelske_jmeno*. Následně nastavit vlastníka, skupinu a restartovat webový server.

```
sudo apt-get install git
cd /var/www/html
sudo git clone
https://uzivatelske_jmeno@bitbucket.org/Luffhassa/dp.git
chown www-data:www-data /var/www/html -R
systemctl restart apache2
```

4.8.4 Databáze

Instalace databáze MariaDB se provede následujícími příkazy. Soubor *mysqlCreate.sql* obsahuje strukturu databáze i funkci pro Levenshteinovu vzdálenost. Port i uživatele se ponechá výchozí.

```
sudo apt-get install software-properties-common

sudo apt-key adv --recv-keys --keyserver
hkp://keyserver.ubuntu.com:80 0xcbcb082a1bb943db

sudo add-apt-repository 'deb
http://mirror.jmu.edu/pub/mariadb/repo/5.5/ubuntu trusty
main'

sudo apt-get update

sudo apt-get install mariadb-server

cd /var/www/html/dp/conf

sudo mysql –uroot –proot < mysqlCreate.sql
```

4.8.5 Perl knihovny

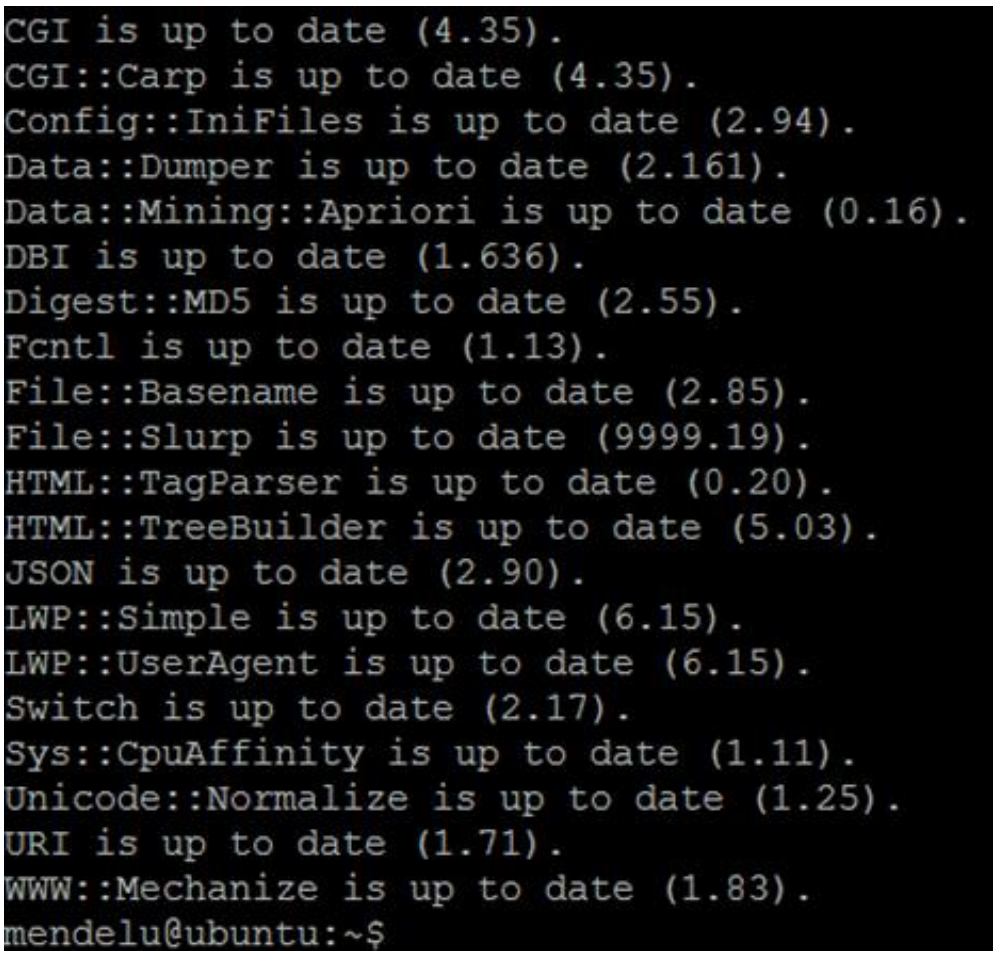
V kapitole Použité knihovny byl uveden výčet knihoven. Pro funkční aplikaci je zásadní nainstalovat všechny. Lze je získat ze serveru cpan.org. Instalaci lze realizovat následujícími příkazy.

```
sudo apt-get install perl-CPAN

sudo apt-get install build-essential

sudo cpan CGI CGI::Carp Config::IniFiles Data::Dumper
Data::Mining::Apriori DBI Digest::MD5 Fcntl File::Basename
File::Slurp HTML::TagParser HTML::TreeBuilder JSON
LWP::Simple LWP::UserAgent Switch Sys::CpuAffinity
Unicode::Normalize URI WWW::Mechanize
```

Po opětovném spuštění příkazu jsou zobrazeny verze jednotlivých knihoven, viz obrázek 24.



Obr. 24 Verze instalovaných knihoven

4.8.6 Nástroj ZBar

Instalace nástroje se provede pomocí příkazu níže.

```
sudo apt-get install zbar-tools
```

4.8.7 Konfigurace

Až na parametr *THREAD_COUNT* se ponechá výchozí konfiguraci. Celá konfigurace byla stažena spolu se zdrojovými kódy aplikace. Do zmíněného parametru se nastaví počet vláken, která budou sloužit ke stahování a zpracování recenzí.

4.8.8 Spuštění aplikace

Vstoupit do kořenového adresáře aplikace. Spustit skript *updater*. Následně již aplikace žije svým životem.

```
cd /var/www/html/dp
./updater.pl
```

5 Diskuze

Velkým přínosem byla paralelizace zmíněných procesů. Proces stahování recenzí byl urychlen na třetinu oproti sekvenčnímu zpracování. Bohužel i tak není proces pořízení nejrychlejší. Web scraping je časově náročná metoda. Střední doba pro stažení katalogu a všech recenzí je 20 hodin. Aplikace však byla navržena jako škálovatelná. Pokud dostane silnější hardwarové komponenty, než byly uvedeny v kapitole Hardwarová specifikace, předpokládá se další zrychlení. Klíčovými atributy jsou počet jader procesoru a dostupná operační paměť.

Bez ukládání do souboru by nebylo téměř možné testovat. I když se sestava pro testování zdá být dostačující, množství paralelních procesů dokázalo operační paměť bez problému vyčerpat. Častá operace nad soubory byla nezbytná. V případě omezené RAM i nutná. Bylo prokázáno, že na konfigurovaný počet vláken je zapotřebí minimálně poloviční počet GB operační paměti (8 vláken, minimálně 4 GB RAM).

Částečným úkolem bylo možné stahování produktů a jejich recenzí i z více zdrojů. Existuje však určitý problém s názvem produktu, respektive jakým způsobem názvy mezi více zdroji párovat. Původně byla zamýšlena metoda pouze s částečnou procentuální shodou. Bylo zjištěno, že produkt například Microsoft Xbox One 500 GB a Microsoft Xbox One S 500 GB mají být chápany jako dva rozdílné produkty. Stejně tak čokoláda 100 g a identická čokoláda pouze 500 g. Následně nepomůže ani procentuální shoda. U příkladu s Xboxem se názvy liší pouze v písmenu „S“ (slim verze). Pokud by byla definována podmínka například 90 % a více společných znaků v názvech, znamenalo by to identický produkt. Potom by byly oba zmíněné produkt vyhodnoceny jako stejné, což nejsou. Jediným jistým přístupem je tedy 100% shoda názvů. Jiným způsobem nelze s naprostou jistotou zajistit párování produktů z více zdrojů. Je na administrátorovi aplikace, aby definoval určitou procentuální shodu názvů.

V rámci zpracování textu bylo v úmyslu využít webová rozhraní společnosti Xerox. Měli implementovaný algoritmus POS s dobrými výsledky i pro češtinu. Bohužel byla jejich dokumentace natolik obecná, až byla nepoužitelná. Při pokusu o navázání komunikace společnost odmítala reagovat. Pokud by se podařilo zapojit algoritmus do procesu, zvýšila by se celková výkonnost zpracování textu (snížil by se počet příznaků). Mohl by být zapojen i implementovaný algoritmus na asociační pravidla. Pro zpracování textu byla použita tokenizace, lemmatizace, stemmizace, normalizace a odstranění stopslov. Výsledky předešlých operací byly uloženy do databáze. Přidanou hodnotu lze zkvalitnit lepšími slovníky, popřípadě dalšími algoritmy, jako je například uvedený POS.

Rychlá odpověď na uživatelské požadavky byla splněna z důvodu uložení zpracovaných výsledků do databáze. Nejpomalejší odpověď je u produktu Apple iPhone 5S 16 GB, jelikož obsahuje nejvíce recenzí (téměř 700). Průměrná doba byla stanovena na 3 vteřiny. U produktů s menším množstvím recenzí byla odezva téměř okamžitá.

Současná extrakce čárových kódu nefunguje ve všech případech ideálně. Bylo vypozorováno, že u produktů mladších 2 let bývá správně identifikován 1 ze 2 čárových kódů. Se staršími produkty hodnota rapidně klesá.

5.1 Konkurence

Mobilní aplikace Heureka provádí identifikaci produktu stejným způsobem jako současná práce, prostřednictvím čárového kódu. Jejich aplikace aktivovala zvláštní režim fotografování, se kterým se dlouhou dobu nedařilo pořídit čárový kód. Po úspěšném pořízení došlo téměř vždy ke správné identifikaci produktu. Pořízení čárového kódu má vlastní aplikace lepší. Následující identifikaci produktu horší. Heureka aplikace umožňuje nákup vyhledaných produktů většinou u několika dodavatelů. Současná práce se však zaměřuje na situace, kdy uživatel stojí přímo v obchodě a chce znát názor na daný produkt. Oproti současné práci lze vytvářet seznamy oblíbených produktů, popřípadě zobrazit historii procházených. Je samozřejmé, že jejich aplikace obsahuje recenze pouze ze serveru heureka.cz. Současná práce je navržena pro přidání dalších zdrojů recenzí. Dalším rozdílem je, že Heureka aplikace recenze nijak nezpracovává. Je zapotřebí pročíst mnohdy dlouhé seznamy.

V případě aplikace dTest bylo zapotřebí nejprve instalovat software třetích stran pro identifikaci produktu pomocí čárového kódu. Identifikace nebyla příliš úspěšná. Byla však testována na malém množství vzorků. Současná práce identifikovala zkoumané produkty lépe. Aplikace dTest obsahuje navíc funkci nákupní průvodce, který radí jakým způsobem vybírat určité produkty. Přesněji na co si dávat zvláštní pozor. Aplikace obsahuje vypracované hodnocení produktu. Zmíněná funkce je však zpoplatněna. Hodnocení je formou článku týmu lidí, kteří mají za úkol zjistit informace o daném produktu. V současné práci jsou informace získávány z příspěvků běžných uživatelů.

Aplikace Customer Reports provádí identifikaci pomocí zadávání znaků do vyhledávače. K hodnocení využívá vypočteného skóre. Na základě skóre lze jednoduše porovnávat produkty. Chybí však slovní názory. Navíc aplikace není určena pro český trh.

V rámci aplikace MouthShut je identifikace produktu stejná jako u aplikace Customer Reports. Aplikace obsahuje hodnocení v podobě počtu hvězdiček a výčtu recenzí. Zajímavou funkcí je zjištění, co si o daném produktu myslí lidé v určitém okolí. MouthShut neobsahuje hromadné zpracování recenzí. Aplikace není určena pro český trh.

Všechny předešlé aplikace měly identifikaci upravenou na vyhledání pomocí čárového kódu nebo na zadání názvu produktu. Obsahovaly i seznam kategorií, na základě kterých bylo umožněno procházet katalog bez identifikace produktu. V současné aplikaci nelze procházet názory produktů bez prvotní identifikace.

5.2 Inovace

Vznik kapitoly inovace byl podnícen potřebou vyjádřit možná vylepšení v definovaných oblastech. Je zde uveden i výčet nápadů, které by mohly práci rozšířit a zkvalitnit.

Prvotně by se autor zaměřil na integrování funkcionality POS. POS je podstatnou funkcionalitou pro další uvedené metody. Prostřednictvím POS by se daly optimalizovat současné algoritmy. Konkrétně přesnější extrakce příznaků (například pouze podstatná a přídavná jména). Dále by bylo umožněno využít asociační pravidla.

Velkým přínosem by mohla být klasifikace sentimentu. Původně měla být součástí práce, ale nakonec nebyla provedena. Databáze však obsahuje příslušné položky (sentiment). I rozhraní je nachystané pro případnou práci. Prezentace sentimentu byla zamýšlena koláčovým grafem, kde by byl na první pohled patrný poměr mezi pozitivními a negativními recenzemi. Mohlo by dojít i k zobrazení koláčového grafu pro dvojice s jednotlivými polaritami. Autora napadlo i možné porovnání vytvořeného sentimentu se sentimentem, které byl explicitně zadán recenzentem při vkládání recenze na webové stránky.

Zajímavé by bylo implementovat vlastní algoritmus pro word cloud a zajistit, chování barev dle jejich frekvence a sentimentu. Například časté pozitivní slovo by bylo označeno sytou zelenou barvou. Méně časté bledě zelenou.

Určité vylepšení či nový přístup by zasloužila část identifikace názvu produktu z čárového kódu. Autora napadlo, že pokud není čárový kód rozpoznán, bude zobrazeno textové pole s našeptávačem, kde by autor přímo vkládal název produktu.

Párování produktů dle názvu je též vhodným kandidátem na inovaci. Bohužel, jak bylo zmíněno dříve, téměř nelze zajistit identičnost produktů.

Zajímavým vylepšením by mohly být uživatelské účty v rámci současné aplikace. Uživatel by si mohl přidávat do oblíbených určité produkty, popřípadě procházet historii prohlížených produktů nebo je vzájemně porovnávat.

V aplikaci byl vytvořen systém komentářů s příslušnými identifikátory (@params, @return). Bylo by vhodné vytvořit skript pro hromadné zpracování komentářů a vytvořit například webovou dokumentaci API. Zmíněné API by mohlo zjednodušit vývoj dalších aktivit.

6 Závěr

Práce se zabývala automatizovaným stahováním a zpracováním uživatelských recenzí a prezentací výsledků uživateli. Důraz byl kladen na rozšiřitelnost o další datové zdroje i o metody na zpracování textu. Dále na konfigurovatelnost, přehlednost a rychlou odezvu grafického uživatelského prostředí.

Nejprve byla představena metodika pro zamýšlenou činnost. Popsala vytvoření vhodných podmínek pro implementaci i fáze životního cyklu vývoje aplikace.

Následující část se věnovala vysvětlení klíčových pojmů, o které se opírá zbylá část práce. Kapitola byla podstatná i pro sjednocení názvosloví. Čtenář získal všeobecný přehled o možných nástrojích a metodách v konkrétních oblastech. Některé z uvedených metod byly posléze využity v praktické části práce.

V rámci vlastní tvorby autor nejprve stanovil prostředí určené pro vývoj a testování aplikace. Byl specifikován operační systém včetně hardwarových prostředků. Dále výčet potřebných vývojových nástrojů spolu se souhrnem použitých knihoven. Následoval seznam funkčních a nefunkčních požadavků, které usměrnily dané řešení. Systémová architektura aplikace prezentovala provázání jednotlivých modulů a skriptů. Prostřednictvím seznamu tabulek, jejich vazeb a jednotlivých polí bylo formalizováno databázové schéma. Následující sekce byla věnována významu jednotlivých adresářů a jejich členění. Byl popsán konfigurační soubor, který podstatným způsobem ovlivňuje chování celé aplikace. Umožňuje modifikovat výkonnost programu pomocí konfigurace vláken a nastavením obsluhy chybných požadavků. Součástí byla i možnost aktivovat či deaktivovat určité funkce aplikace, specifikovat parametry pro připojení k databázi a konkretizovat cesty k použitým externím nástrojům i slovníkům. Následoval diagram workflow, který měl za úkol prezentovat posloupnost kroků při práci s grafickým uživatelským prostředím z pohledu uživatele aplikace. Samotná implementační část přinesla souhrnný popis všech programových modulů a skriptů. Zvláštní pozornost dostaly podstatné či nějakým způsobem zajímavé části zdrojového kódu. Prvotním cílem bylo prezentovat autorovy myšlenky. Podstatné bylo uvést i provázání mezi parametry konfiguračního souboru, adresářovou strukturou a samotným programem. Optimalizace nastínila, jakými prostředky byla realizována vylepšení časově a paměťově náročných funkcí. Dále možnosti pro omezení datové množiny či inkrementální přístup při pořizování katalogu produktů. V rámci testování byly uvedeny využité produkty spolu s podporovanými webovými prohlížeči. V kapitole diskuze byly shrnuty výsledky z celkového vývoje aplikace. Byly stanoveny konkrétní hodnoty pro potřebný čas stahování recenzí. Dále správný poměr mezi počtem vláken a operační pamětí. V neposlední řádě úspěšnost extrakce čárových kódů a identifikace produktu. Součástí byl i výčet možných nápadů, jakým způsobem by šla aplikace nadále zkvalitňovat.

V souhrnu byl cíl práce naplněn. Podařilo se vytvořit graficky přehlednou, konfigurovatelnou aplikaci s rychlou odezvou. V zadaných intervalech se provede automatické pořízení a zpracování recenzí. Aplikace je škálovatelná a rozšiřitelná o další datové zdroje. Uživateli jsou prezentovány zpracované názory. Lze si udělat představu o nejčastěji probíraných tématech a přispět k celkovému rozhodnutí o nákupu.

7 Literatura

Kapitola literatura pojednává o použitých zdrojích v rámci diplomové práce. Použitá literatura obsahuje informace z knižních publikací, diplomových prací, vědeckých článků a z několika webových stránek. V citacích jsou odkazy i na použité nástroje a slovníky.

ADAMO, J. M., 2012: *Data mining for association rules and sequential patterns: sequential and parallel algorithms.* S.l.: Springer. ISBN 9781461265115.

ALGORITMY.NET., 2015: *Levenshteinova vzdálenost.* [online]. Dostupné z: https://www.algoritmy.net/article/1699/Levenshteinova-vzdalenost.

ANDERSON, S. R., 2016: *Morphology*. Yale University [online]. Macmillan publishers. Dostupné z: https://cowgill.ling.yale.edu/sra/morphology_ecs.htm.

BERRY, M. W., KOGAN, J., 2010: *Text mining: applications and theory.* Hoboken, NJ: John Wiley & Sons. ISBN 978-0-470-74982-1.

BING, L., 2012: *Sentiment analysis and opinion mining.* San Rafael: Morgan & Claypool Publishers. Synthesis lectures on human language technologies, 16. ISBN 978-1-60845-884-4.

BOOST LABS, 2016: *Word Clouds & the Value of Simple Visualizations.* [online]. Dostupné z: http://www.boostlabs.com/what-are-word-clouds-value-simple-visualizations/.

BOUGE, K., 2011: *Stop words.* [online]. Dostupné z: https://sites.google.com/site/kevinbouge/stopwords-lists.

BROWN, J., 2010: *Bar bar code reader.* [online]. Dostupné z: http://zbar.sourceforge.net/index.html.

CAMBRIDGE DICTIONARY, 2016: *Mobile device.* [online]. Cambridge University Press. Dostupné z: http://dictionary.cambridge.org/dictionary/english/mobile-device.

CAMBRIDGE DICTIONARY, 2016: *Product review.* [online]. Cambridge University Press. Dostupné z: http://dictionary.cambridge.org/dictionary/english/product-review.

CAREY, P., 2006: *New perspectives on creating Web pages with HTML, XHTML, and XML*. 2nd ed. Boston, Mass.: Thomson/Course Technology. ISBN 9780619268015.

CROWDER, D. A., CROWDER R., 2001: *Creating Web pages bible.* London: Transworld. ISBN 0764547917.

DAŘENA, F., ŽIŽKA, J., PŘICHYSTAL, J., 2014: *Clients' freely written assessment as the source of automatically mined opinions.* In Nerudová, D. 17th International Conference Enterprise And Competitive Environment. 1. vyd. Amsterdam, Netherlands: Elsevier Science Bv, 2014, s. 103-110. ISSN 2212-5671.

DAVIES, J., 2015: *Word Cloud Generator.* Jason Davies [online]. Dostupné z: https://www.jasondavies.com/wordcloud/.

EL-GOHARY, H., EID, R., 2013: *E-marketing in developed and developing countries: emerging practices.* Hershey, PA: Business Science Reference, an Imprint of IGI Global. ISBN 9781466639546.

FELDMAN, R., SANGER J., 2007: *The text mining handbook*. Advanced approaches in analyzing unstructured data. New York: Cambridge University Press. ISBN 978-0-521-83657-9.

FRAIN, B., 2012: *Responsive web design with HTML5 and CSS3*. Learn responsive design using HTML5 and CSS3 to adapt websites to any browser or screen size. Mumbai: Packt Pub. Community experience distilled.

GABEN, 2016: *Čárové kódy*. [online] Dostupné z: http://www.gaben.cz/cz/faq/carove-kody-teorie.

HUNTER, D., 2007: *Beginning XML*. 4th ed. Indianapolis, IN: Wrox/Wiley Pub. Wrox beginning guides.

CHEUNG, D., HO, T. B., LIU, H., 2005: *Advances in Knowledge Discovery and Data Mining* (vol. [3518) 9th Pacific-Asia Conference, PAKDD 2005, Hanoi, Vietnam, May 18-20, 2005, Proceedings. Berlin Heidelberg: Springer-Verlag. ISBN 9783540319351.

CHUNGPING, L., ZHAO, L., 2009: *Ontology Based Opinion Mining for Movie Reviews.* Knowledge Science, Engineering and Management. Lecture Notes in Computer Science, s. 204-214. DOI: 10.1007/978-3-642-10488-6_22.

JAMES, M., 2016: *Hashing - The Greatest Idea In Programming*. I programmer [online]. Dostupné z: http://www.i-programmer.info/babbages-bag/479-hashing.html.

KATERATTANAKUL, N., 2010: *A pilot study in an application of text mining to learning system evaluation*. Diplomová práce. Missouri University of Science and Technology. Vedoucí práce Dr. Wen-Bin Yu.

KAVOURGIAS, C., 2014: *Static and Dynamic Websites: There's a Difference*. PluralSight [online]. Dostupné z: http://blog.digitaltutors.com/static-dynamic-websites-theres-difference/.

KODYS, 2009: *EAN 13 A EAN 8.* [online]. Dostupné z: http://www.kodys.cz/carovy-kod/ean-13-a-ean-8.html.

KUMAR, E., 2011: *Natural language processing.* New Delhi: I.K. International Publishing House. ISBN 9789380578774.

MATTOSINHO, F. J. A. P., 2010: *Mining Product Opinions and Reviews on the Web*. Drážďany. Diplomová práce. Technická univerzita Drážďany. Vedoucí práce Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill. DOI: 10.1.1.461.7357.

MCGOVERN, J., 2004: *A practical guide to enterprise architecture.* Upper Saddle River, NJ: Prentice Hall/Professional Technical Reference. ISBN 0131412752.

MĚCHURA, M. B., 2016: *Lemmatization Lists.* Lexiconista [online]. Dostupné z: http://www.lexiconista.com/datasets/lemmatization/.

MIRKOVIC, J., 2005: *Internet denial of service: attack and defense mechanisms.* Upper Saddle River, N.J.: Prentice Hall Professional Technical Reference. ISBN 0131475738.

MOENS, M., 2006: *Information extraction: algorithms and prospects in a retrieval context.* Dordrecht: Springer. ISBN 1-4020-4987-0.

MORENO, J., TORRES, M., 2014: *Automatic text summarization*. Hoboken, New Jersey: John Wiley & Sons. ISBN 978-1-119-04414-7.

MUNZERT, S., RUBBA, CH., MEISSNER P., NYHUIS D., 2015: *Automated data collection with R: a practical guide to web scraping and text mining.* Chichester: Wiley. ISBN 978-1-118-83481-7.

NUGUES, P. M., 2006: *An introduction to language processing with Perl and Prolog an outline of theories, implementation, and application with special consideration of English, French, and German.* Berlin: Springer. ISBN 9783540343363.

PŘICHYSTAL, J., 2016: *Mobile application for customers' reviews opinion mining*. 19th International Conference Enterprise and Competitive Environment, s. 373-381.

RICHARDSON, L., AMUNDSEN, M., 2013: *RESTful Web APIs.* Beijing: O'Reilly. ISBN 9781449358068.

WRAPBOOTSTRAP, 2016: *Inspinia.* [online]. Dostupné z: https://wrapbootstrap.com/theme/inspinia-responsive-admin-theme-WB0R5L90S.

8 Seznam obrázků

9 Seznam tabulek

Přílohy

A Konfigurační soubor

```
[GLOBAL]
DEBUG=0
THREAD_COUNT=8
THREAD_SLEEP_INT=3
PRODUCTS_PER_FILE_LIMIT=3000
PRODUCT_NAME_PERCENTAGE_MATCH=90
WORD_CLOUD_MIN_WORD_FREQUENCY=2
SHOW_WORD_CLOUD=1
SHOW_WORD_CLOUD_NEIGHBORS=1
UPDATE_EVERY=30
DELETE_ALL_ON_UPDATE=1
STOPWORDS_PATH=conf/stopwords_cz.txt
LEMMATIZATION_PATH=conf/lemmatization_cs.txt
ZBARIMG_HOME=
#-----------------------------------------------
[DATABASE]
HOST=localhost
PORT=3306
USER=root
PASSWORD=root
SCHEMA=dp
#-----------------------------------------------
[HEUREKA]
URL=http://www.heureka.cz
POPULATE_PRODUCTS_LIST=1
POPULATE_PRODUCTS=1
POPULATE_REVIEWS=1
GET_ERRORS_LIMIT=5
GET_ERRORS_SLEEP=5
MAX_CATEGORY_PAGE=15000
```

B Konfigurace Apache

```
su root

apt-get install apache2
apt-get install libapache2-mod-perl2
systemctl enable apache2
systemctl start apache2
a2enmod cgi

vi /etc/apache2/sites-enabled/000-default.conf
Vložit následující řádky mezi tagy VirtualHost
<Directory "/var/www/html/dp">
     AddHandler cgi-script .cgi .pl .pm
     Options +Indexes +ExecCGI
     DirectoryIndex index.cgi
     AllowOverride Limit
</Directory>
```