Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

Aplikace algoritmů umělé inteligence pro data mining v prostředí realitního trhu

Diplomová práce

Bc. Miloslav Thon

Školitel: Ing. Jan Kelnar

České Budějovice 2014

Tuto práci věnuji své manželce Janě.

Thon, M., 2014: Aplikace algoritmů umělé inteligence pro data mining v prostředí realitního trhu. [Use of artificial intelligence algorithms for data mining in the real estate market. Mgr. Thesis, in Czech.] - 79 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace:

This thesis describes selected algorithms and techniques used in processing of unstructured text and a particular application in the implementation of a computer system offering search on publicly available real estate advertising. The thesis covers whole process of the knowledge discovery: obtaining and pre-processing the data, application of the algorithms and evaluation and presentation of results.

Prohlašuji, že svoji diplomovou práci jsem vypracoval/a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

České Budějovice, 4. prosince 2014.

Vážení čtenáři,

zdárné dokončení této práce by nebylo možné bez pomoci mých příbuzných, známých a kolegů. Rád bych jim tímto poděkoval za pomoc při zpracování této práce a za podporu během celého mého studia. Zvláštní a nejvýznamnější dík patří mé manželce Janě. Během mého studia na ni totiž přešla větší část starostí a povinností spojených zejména s péčí o našeho syna Petra. Zároveň mi byla oporou v náročných chvílích během studia. Doufám, že jsem jí tuto obětavou pomoc alespoň částečně vynahradil podporou v mimořádně těžké životní situaci, kterou jí osud přichystal na podzim roku 2014. Bylo to v závěru tvorby mé práce a myšlenky na její úspěšné dokončení musely chtě nechtě na chvíli stranou. Vzhledem k této situaci, kterou jsme naštěstí společně zvládli, patří výsledky celého mého studijního úsilí právě jí.

Za poskytnutí velmi zajímavé příležitosti řešit reálný problém a za průběžné konzultace při tvorbě prototypového řešení děkuji Ing. Janu Kelnarovi, vedoucímu práce. I přes to, že se ve své práci pohybuji ve zcela akademické rovině a odtržen od komerčního využití systémů pro zpracování realitní inzerce, bylo pro mě určitě přínosné, že jsem se mohl zabývat problematikou, kterou se u nás zabývá několik firem.

Za to, že jsem mohl studovat při zaměstnání, děkuji Ing. Luďku Fürstovi, generálnímu řediteli TranSoft a. s. a Ing. Lubomíru Moravčíkovi, řediteli oddělení programování. Děkuji i dalším kolegům z práce, se kterými jsem mohl o svém studiu a o své diplomové práci hovořit. Každá výměna názorů je v zaměstnání vždy přínosem. Doufám, že byla přínosem i pro ně.

Rád bych také poděkoval svým rodičům za podporu během studia. Konečně i oni určitou měrou napomohli mému rozhodnutí přihlásit se do navazujícího magisterského studia na PřF JU. Za odborné konzultace, za pomoc při výběru některých konkrétních témat, za zapůjčení odborné literatury a za pomoc s korekturou textu děkuji svému bratrovi Láďovi.

Miloslav Thon

Obsah

1 Úvod

Tato diplomová práce vznikla jako závěrečná práce studia navazujícího magisterského oboru Aplikovaná informatika na Přírodovědecké fakultě Jihočeské univerzity v Českých Budějovicích. Motivací ke vzniku práce je potřeba řešení automatizovaného zpracování dat z veřejně dostupných zdrojů, zejména z webových serverů. Práce je zaměřena na specifickou oblast realitní inzerce, nicméně popsané algoritmy a postupy mohou být využitelné i pro zpracování jiného nestrukturovaného obsahu. Automatizovaným zpracováním dat je myšleno takové zpracování, které umožní v datech jednoduchým způsobem vyhledávat, získávat z dat požadované údaje ve strukturované formě, popř. data srozumitelným způsobem prezentovat uživateli. Zvláštním případem zpracování je sjednocení dat získaných z různých zdrojů. Konkrétně v případě realitní inzerce je tím myšlena jakási agregace inzerátů a jejich uložení a zpracování v jednom informačním systému.

Zpracovaná témata jsou volena tak, aby bylo možné prezentovat zajímavá řešení zdánlivě jednoduchých problémů, které jsou ovšem obtížně řešitelné běžnými algoritmy a pro jejichž řešení jsou využitelné algoritmy a techniky strojového učení.

1.1 Cíle práce

Výsledkem práce by měl být přehled algoritmů, postupů a návrhů řešení využitelných pro vytvoření aplikace, která koncovému uživateli poskytne:

- nestrukturované vyhledávání inzerátů dle zadaného textu, který se v inzerátu vyskytuje – tj. fulltextové vyhledávání, které umožňuje vyhledání požadovaného inzerátu dle lokality, popisu nemovitosti, vybavení, apod.,
- filtrování inzerátů dle ceny nemovitosti získané z částečně strukturovaných údajů při předzpracování textu inzerátu,
- filtrování inzerátů dle vybraných kritérií, jako typ inzerce (prodej, pronájem, poptávka, apod.) nebo kategorie nemovitosti (byt, dům, chata, komerční prostor, apod.) získaných metodami strojového učení (jde o aplikaci problému klasifikace),
- spojení totožných inzerátů ve výsledcích vyhledávání (jde např. o duplicitně zveřejněné inzeráty, nebo inzeráty týkající se stejné nemovitosti zveřejněné na různých inzertních portálech); sloučením totožných inzerátů je v některých případech možné získat o podrobnější informace o nemovitosti,[1]
- vyhledání podobných inzerátů k zobrazenému inzerátu; jde o obdobu zobrazení podobného zboží k výrobku zobrazeného v e-shopu.[2]

1 Podrobnosti o možnostech využití spojení totožných inzerátů jsou uvedeny v kapitole 4.

2 Tento způsob použití není možné zaměňovat se zobrazováním doporučeného zboží nebo zboží, které si také zákazníci koupili společně s daným výrobkem. V takovém případě by se jednalo o odlišnou úlohu data miningu, a sice analýzu nákupního košíku realizovanou vyhledáváním tzv. asociačních pravidel.

Práci tvoří dvě části. Teoretickou částí práce je provedení analýzy a popisu vybraných algoritmů, které je možné pro naplnění výše uvedených cílů využít. Praktickou částí práce je návrh a implementace aplikace, která uživateli poskytne konkrétní softwarové nástroje splňující výše uvedené cíle. Jedná se o prototypové řešení jednoduchého *Information Retrieval*[3] systému, který dokáže získat data z několika veřejně dostupných inzertních portálů, tato data zpracovat a umožnit uživatelsky srozumitelné a jednoduché vyhledávání v databázi získaných inzerátů.

1.2 Uspořádání textu

Text práce je rozdělen do devíti kapitoly. První dvě tvoří úvod práce a popis řešeného problému. Druhá kapitola je více zaměřena na popis specifik realitní inzerce a představení procesu data miningu. Třetí kapitola je úzce zaměřena na řešení problému získání dat z veřejně dostupných realitních serverů. Čtvrtá až sedmá kapitola se zabývají vlastním zpracováním realitních inzerátů a postupy pro dolování dat. Součástí těchto kapitol je vysvětlení principu vybraných algoritmů a odkaz na jejich implementaci v prototypovém řešení. Osmá kapitola popisuje obecnou architekturu prototypového řešení vytvořeného v rámci práce. V této kapitole je představeno i uživatelské rozhraní umožňující ověření výsledků práce. V závěrečné deváté kapitole jsou uvedeny některé náměty na další možné využití výsledků práce.

1.3 Přiložené DVD

Na přiloženém DVD jsou uloženy kompletní zdrojové kódy zahrnující implementaci všech popsaných algoritmů a prototypového řešení. K dispozici je i spustitelná verze s již připravenou databází inzerátů získaných v době tvorby práce.

Celou práci v elektronické formě včetně obsahu DVD je rovněž možné stáhnout z internetové adresy: `https://sites.google.com/site/thondiplomka/`.

1.4 Použité technologie

Prototypové řešení je vyvinuto v prostředí Microsoft .NET Framework 4.5 s použitím vývojového prostředí Microsoft Visual Studio 2012. V jednotlivých projektech jsou využity následující knihovny:

- HTML Agility Pack 1.4.6,
- System.Data.SQLite 1.0.88 (ADO.NET ovladač obsahující SQLite verze 3.7.17),
- Lucene.NET 3.0.3 a
- Log4net 1.2.11.

3 Pojem Information Retrieval je blíže vysvětlen v kapitole 2.2.

2 Získávání informací z realitní inzerce

Existuje velké množství internetových portálů zaměřených na poskytování určitého typu obsahu a na určitou oblast uživatelů, kteří jejich služeb využívají. Jedním typem z nich jsou portály zaměřené na realitní inzerci, jejichž zpracováním se zabývá tato práce. Záměrně zde neuvažujeme jiné zdroje realitní inzerce, jako jsou např. noviny. U takových zdrojů téměř nepřichází v úvahu jakékoliv jejich automatizované zpracování pomocí výpočetní techniky. Navíc vzhledem k výrazným odlišnostem ve formě zveřejňování inzerce by pravděpodobně byly mnohé postupy rozebrané v této práci nepoužitelné.

Obsah zveřejňovaný na realitních portálech má svá specifika. Některá specifika je samozřejmě možné pozorovat i u jiných typů obsahu. Velmi častým typem obsahu, pokud pochopitelně pomineme obecný webový obsah zpracovávaný internetovými vyhledávači, jsou např. zpravodajské weby nebo internetové obchody. Existuje spousta systémů, které určitým způsobem data z těchto portálů získávají, dále zpracovávají a agregují a poskytují uživatelům množství nástrojů pro vyhledávání či jiné zpracování takového obsahu. Pokud se podrobněji zaměříme na obsah realitní inzerce zveřejňované na internetu, můžeme pozorovat několik charakteristických rysů.

Množství údajů v inzerátu je relativně malé. Dá se říci, že je do určité míry podobné např. s množstvím údajů zveřejňovaným u zboží v internetovém obchodu. Text je často tvořen spíše jednoduchými větami nebo dokonce pouze slovními spojeními. Naproti tomu obsah zveřejňovaný na zpravodajských serverech je často výrazně obsáhlejší a obsahuje souvislý text tvořený převážně souvětími. Množství textu pro zpracování je mnohdy zásadní pro aplikaci některých algoritmů strojového učení. Například problémem klasifikace krátkých textů či problémem vyhledávání duplicitního obsahu v případě krátkého textu se zabývají články [1], [2] a [3]. Jak bude patrné z výsledků práce, pro potřeby zpracování realitní inzerce je možné i přes uvedená specifika považovat obsah inzerátů za standardně dlouhý text.

Ve většině případů je část údajů v obsahu inzerátu dostupná ve strukturované podobě. Podobně jako v předchozím případě můžeme i zde pozorovat podobnost s internetovými obchody, na rozdíl od zpravodajských portálů. Některé vlastnosti popisující inzerovanou nemovitost jsou na portálu zobrazovány v oddělené tabulce společně s názvy jednotlivých vlastností. Často bývá odděleně od vlastního textu inzerátu uvedena cena, lokalita, typ nemovitosti, kontaktní údaje inzerenta a další údaje. Některé z těchto údajů je možné přímo získat parsováním webové stránky, zpracováním textu s použitím regulárních výrazů, apod. Jde o techniky souhrnně nazývané *Web Scraping* nebo jen *scraping*.[1] Takovou extrakci údajů je možné velmi snadno automatizovat pro jeden konkrétní realitní portál, naopak poměrně obtížné je automatizovat ji pro obecný internetový obsah.

Předchozí dva rysy byly pro inzerci podobné s internetovými obchody. U následujících dvou je tomu naopak. Z textu zveřejněného inzerátu je prakticky nemožné získat jakoukoliv jeho jednoznačnou identifikaci pro účely ztotožnění shodných inzerátů zveřejněných na různých portálech. Zboží v internetovém obchodě je často označeno

1 Viz http://en.wikipedia.org/wiki/Web_scraping.

příznačným názvem či modelovým označením. U inzerátů je podobný problém jako v případě zpravodajského článku. Ačkoliv inzerát je často dobře strukturovaný, téměř nikdy neobsahuje údaj, který by ho mohl jednoznačně identifikovat. Některé portály uvádějí v inzerátů číslo zakázky. To je však jednoznačné pouze v rámci jednoho portálu. Některé portály dokonce záměrně zařazují do vyhledávání jeden inzerát vícekrát z odlišným číslem zakázky. Můžeme se pouze domnívat, zda je to obrana proti systémům, které inzeráty agregují, nebo zda se portály snaží zvýšit šanci vyhledání inzerátu v internetových vyhledávačích. V kapitole 7 uvidíme, jakým způsobem je možné zjistit, zda se jedná o totožné inzeráty. A to i přes jejich odlišné označení, které je zdánlivě jednoznačné.

Posledním zmíněným rysem realitní inzerce je velmi krátká trvanlivost obsahu. Tím je myšleno, že zveřejňované inzeráty se velmi často mění, přičemž neplatné inzeráty jsou průběžně odstraňovány z vyhledávání. Není výjimkou, kdy platnost inzerátu je pouze jeden až několik dní. Pro systém, který se snaží inzeráty automatizovaně zpracovávat, tedy vzniká problém s aktualizací databáze inzerátů. Doba zveřejnění nabídky konkrétního zboží v internetovém obchodu bude většinou delší. Málokteré zboží je v nabídce pouze několik dní. V případě zpravodajských serverů můžeme dokonce tvrdit, že trvanlivost obsahu je téměř neomezená. Řada portálů poskytuje dokonce i archivy velmi starých článků.

2.1 Možnosti využití informací

Jednotlivci z řad veřejnosti nebo firmy mají různé potřeby využití údajů získaných z realitní inzerce. Původem těchto potřeb mohou být buď osobní zájmy nalezení vlastního bydlení či naopak prodej vlastní nemovitosti. Mnoho lidí zcela určitě využívá realitní inzerci pouze za účelem vytvoření představy o cenách nemovitostí. Poněkud odlišné potřeby mohou mít firmy zabývající se zprostředkováním prodeje nemovitosti. Takové firmy budou mít potřebu využít údaje z realitní inzerce primárně za účelem nalezení vhodného poptávajícího či nabízejícího a následné nalezení vhodné protistrany ve vlastní databázi klientů.

Z uvedených potřeb je možné vyvodit i základní způsoby využití systému pro automatizované zpracování inzerce. Pravděpodobně nejsnazším způsobem využití je ad hoc vyhledávání v databázi inzerátů. Pod tímto označením si můžeme představit počítačový software, který uživateli poskytne možnosti vyhledávání v obsahu inzerátu, popř. bude inzeráty určitým způsobem kategorizovat a vytvářet tak jakýsi katalog. S tímto způsobem využití souvisí i problém spojení stejného či podobného obsahu z různých zdrojů do jedné kolekce. Obdobu můžeme opět nalézt u internetových portálů, které sdružují nabídky zboží z internetových obchodů nebo sdružují články publikované zpravodajskými servery.[2] Činnost vedoucí k nalezení a získání informací z takto připravené kolekce dokumentů je označována jako *Information Retrieval*. Právě tento způsob využití realitní inzerce tvoří hlavní náplň této práce.

Dalším způsobem využití je monitoring realit, kterým se u nás komerčně zabývá několik firem. Jde o poměrně specifickou oblast zájmu, která úzce souvisí s předchozím způsobem využití. Hlavním požadavkem na počítačový systém je v tomto případě automatické upozorňování uživatele o výskytu nového inzerátu, který odpovídá zadaným

2 Známé jako srovnávače či agregátory.

parametrům. Vzniká zde tedy určitá potřeba strukturovaného vyhledávání a velmi časté kontroly existence inzerátů na realitních portálech.

Oba předchozí způsoby využití mohou být, a většinou i jsou, cíleny na širokou veřejnost. Odlišným způsobem využití realitní inzerce může být nejrůznější analytická činnost. Jejím předmětem může být provádění cenových odhadů, vytváření cenových map či provádění marketingových průzkumů. Zájmovou skupinou mohou v tomto případě být realitní kanceláře, odhadci nemovitostí či nezávislí analytici. Tato problematika již překračuje rámec této práce.

2.2 Information retrieval

V [4] je nastíněn příklad Information Retrieval problému,[3] jako potřeba vyhledávání informací v nějaké kolekci dokumentů. Kolekcí dokumentů může být např. sbírka publikací v knihovně, veřejně dostupný internetový obsah nebo v našem případě databáze realitních inzerátů získaných z různých veřejných realitních portálů. Lidé, kteří potřebují v takové kolekci vyhledávat, neznají její celý obsah. Ten může být velmi rozsáhlý a čítat řádově sta tisíce či milióny dokumentů. V případě obecného internetového obsahu jde dokonce o miliardy dokumentů. Zcela přirozený je však požadavek v takto obsáhlé kolekci dokumentů vyhledávat určité požadované informace a ty pak dále využít, popř. i strojově zpracovat. Činnost vedoucí k získání požadovaných informací se nazývá *Information Retrieval*. Počítačový systém, který poskytuje nástroje pro získání požadovaných informací, označme jako *Information Retrieval System* nebo zkráceně *IR System*.

Jedním ze způsobů vyhledávání údajů z databází je tzv. strukturované vyhledávání, které je velmi často realizováno relačním databázovým systémem. V relační databázi jsou uložena strukturovaná data, ve kterých je možné vyhledávat zadáním SQL dotazů. Ty umožňují velmi snadno kombinovat a poměrně přesně definovat podmínky vyhledávání. Již z textu dotazu je zřejmé, jaký bude obsah vyhledaných záznamů. Například tento jednoduchý dotaz:

```
SELECT Title, Text, Price FROM Adverts WHERE Price <= 12000 AND
Locality = 'České Budějovice'
```

zajistí vyhledání všech inzerátů týkajících se nemovitostí v Českých Budějovicích s cenou nižší nebo rovnou 12 000 Kč. Nespornou výhodou tohoto přístupu je přesnost vyhledávání. Můžeme též předpokládat, že stejný dotaz bude možné bez změny využitelný i v případě změny obsahu databáze. Nevýhodou je však nutnost velmi přesného uložení dat v databázi. Při získávání údajů z veřejně dostupných webových portálů většinou není možné jednoduchým způsobem získat dobře strukturovaná data. Kupříkladu údaje cena a lokalita, uvedené v příkladu, nemusejí být vždy v inzerátu uvedeny. Cena může chybět, lokalita může být uvedena pouze přibližně, apod. Některé další problémy při získávání údajů z textu jsou uvedeny v kapitole 2.4.1.

Odlišným přístupem je vyhledávání pomocí tzv. *free text queries*,[4] neboli pomocí dotazů zadaných ve formě volného textu. Jedná se tedy o formu nestrukturovaného vyhledávání používanou při získávání informací z nestrukturovaného textu. Při tomto způsobu vyhledávání je možné zadat dotaz:

3 Kapitola 1.1 – An example information retrieval problem.
4 Viz [4], kapitola 1.4.

```
byt pronájem české budějovice jižní čechy
```

Je zřejmě, že od strukturovaného dotazu se tento dotaz výrazně liší. Např. není možné jednoduchým způsobem zadat podmínku omezující maximální cenu. Tato informace může být částečně nahrazena výrazem „`byt pronájem`“. Je totiž zřejmé, že tato kategorie inzerátů bude nejlépe odpovídat cenové relaci do 12 000 Kč. Kromě toho je možné mnohem volněji formulovat požadavek na lokalitu inzerátu. Výraz „`jižní čechy`“ zajistí zařazení inzerátů do výsledků vyhledávání, ve kterých není přímo uvedena lokalita České Budějovice, ale je zde uvedena oblast Jižní Čechy. Jakým způsobem je ale možné celý dotaz chápat? Při vyhodnocení takového dotazu předpokládáme, že výsledkem vyhledávání budou takové inzeráty, které nejlépe odpovídají zadanému dotazu, neboli jsou mu nejpodobnější. Výhodou tohoto způsobu vyhledávání je tedy možnost volného formulování dotazu v databázi nestrukturovaných dokumentů. Naproti tomu je potřeba počítat s tím, že výsledek dotazu nemusí zcela přesně odpovídat tomu, co uživatel požaduje. V takovém případě je nutné dotaz vhodným způsobem upravit a vyhledávání opakovat. Většina IR systémů nabízí kombinaci strukturovaného a nestrukturovaného vyhledávání. Takovou kombinaci vyhledávání poskytuje i aplikace vytvořena v rámci této práce.

2.3 Princip fungování IR systému

Jak již bylo uvedeno v předchozí kapitole, základem pro vyhledávání ve strukturovaných datech je nejčastěji relační databáze. Naproti tomu základním nástrojem pro vyhledávání v kolekci nestrukturovaných dokumentů je index.[5] Jedná se o datovou strukturu obsahující seřazený seznam všech výrazů (tzv. *terms*) vyskytujících se v databázi dokumentů.[6] Výrazem může být slovo, popř. základní tvar slova, nebo nějaká specifická hodnota vyjadřující např. zařazení dokumentu do určité kategorie nebo popisující nějakou konkrétní vlastnost. Ke každému výrazu je přiřazen seznam dokumentů, ve kterých se výraz vyskytuje (tzv. *posting list*). V ukázce 2.1 je znázorněna struktura jednoduchého indexu se seznamem příslušných dokumentů.

Výrazy v indexu nemusejí být pouze jednoduché, jak je uvedeno na příkladu, ale mohou být rozděleny do tzv. *zón*. Pomocí zón je možné oddělit samostatné informace týkající se jednoho dokumentu, jako je např. nadpis inzerátu, text, cena, kategorie, apod. Dokument může tedy být částečně strukturovaný, tj. tvořený množinou více položek (tzv. *field*). Stejné položky všech dokumentů v databázi pak představují jednu zónu. Zóny je následně možné využít pro částečné strukturování vyhledávacích dotazů. V této práci je tento mechanizmus využit pro vyhledávání inzerátů dle kategorie nemovitosti, typu obchodu a ceny. V indexu je tedy v samostatné položce uložen název kategorie, označení typu obchodu a číselná hodnota ceny nemovitosti. Podrobnosti o konkrétním způsobu řešení jsou uvedeny v kapitole 8.

Jakým způsobem zajistit vyhledávání v indexu? Nejjednodušším způsobem je vyhodnocování tzv. *boolean queries*. Jde způsob vyhodnocení dotazu na základě přítomnosti či nepřítomnosti zadaného slova v dokumentu. V dotazu je navíc možné slova kombinovat pomocí logických operátorů AND, OR a NOT. Tento způsob vyhledávání byl

5 Často též označovaný jako invertovaný index.

6 V našem případě je dokumentem jeden realitní inzerát, nicméně pro zachování terminologie s použitou literaturou je zde ponecháno označení dokument.

Dokument 1:
Prodám byt v Českých Budějovicích.

Dokument 2:
Pronájem bytu v Českých Budějovicích.

Dokument 3:
Koupím dům se zahradou.

budějovicích: 1, 2
byt: 1, 2
českých: 1, 2
dům: 3
koupím: 3
prodám: 1
pronájem: 2
zahradou: 3

Ukázka 2.1: Ukázka třech krátkých inzerátů a z nich sestaveného indexu. Obsah indexu je znázorněn v rámečku, kde u každého výrazu je za dvojtečkou oddělený posting list.

využit i u prvních internetových vyhledávačů. V moderních IR systémech se využívá způsob vyhodnocování založený na porovnání podobnosti zadaného dotazu s dokumenty uloženými v indexu. Za tím účelem je nutné vhodným způsobem reprezentovat dokumenty ve vícerozměrném prostoru, ohodnotit význam jednotlivých slov, resp. výrazů, kterými je dokument tvořen a následně provést porovnání dokumentů se zadaným dotazem a vyhodnocení vyhledávacího dotazu. Proces vyhodnocení dotazu se nazývá *scoring*.

Vhodným modelem pro reprezentování dokumentů je *Vector Space Model*, který je dnes standardně využíván fulltextovými vyhledávači. V tomto modelu je každý dokument reprezentován vektorem, který má tolik složek, kolik je unikátních výrazů v celé databázi dokumentů. Přirozeně každý dokument obsahuje pouze několik výrazů, ostatní složky vektoru mají tedy hodnotu 0. Pro výrazy obsažené v dokumentu je hodnotou příslušné složky vektoru hodnota odpovídající důležitosti daného výrazu, tj. jeho *váha*. Jednou z metrik pro vyjádření váhy výrazu je *Tf-idf*.[7] Hodnotou je reálné číslo definované vztahem

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \cdot \text{idf}_t \quad ,$$

kde $tf_{t,d}$ je počet výskytů výrazu t v dokumentu d (term frequency), idf_t je hodnota *Inverted Document Frequency* výrazu t, která je definována vztahem

$$\text{idf}_t = \log \frac{N}{\text{df}_t} \quad ,$$

kde N je počet všech dokumentů v databázi a df_t je počet dokumentů obsahujících výraz t. Významem členu *tf* je zvýšení váhy výrazů, které se v dokumentu vyskytují vícekrát. Člen *idf* naopak zajišťuje snížení váhy výrazů, které se vyskytují ve velkém množství dokumentů. Výsledná váha je tedy nejvyšší pro výrazy, které se vyskytují vícekrát v daném dokumentu a nevyskytují se v ostatních dokumentech, tedy jsou pro dokument charakteristické. Naproti tomu malou váhu mají výrazy, které se ve stejné míře vyskytují ve všech dokumentech v databázi, což jsou často obecná slova, zájmena, spojky, apod. Při vyhodnocení dotazu je pak výsledné skóre pro každý dokument určeno jako

$$\text{Score}(q,d) = \sum_{t \in q} \text{tf-idf}_{t,d} \quad ,$$

kde d je dokument, pro který je skóre počítáno a q je zadaný dotaz tvořený jedním či více výrazy. Skóre je zároveň údajem, podle kterého jsou řazeny výsledky vyhledávání. Na

7 Vzorce uvedené v této kapitole jsou převzaty z [4], kapitola 6.2 Term frequency and weighting.

prvním místě je výsledek s nejvyšším skóre, což odpovídá dokumentu, který je zadanému dotazu nejpodobnější. Zjednodušeně můžeme říci, že dokument je dotazu podobný v případě, že obsahuje maximum výrazů, které jsou pro požadovaný dokument typické a mají tedy vysokou váhu.

Pro implementaci indexu v rámci této práce je využita knihovna *Apache Lucene.NET*. Tato knihovna poskytuje nízkoúrovňové API pro tvorbu indexu a pro vyhodnocování všech výše uvedených typů dotazů. Zároveň implementuje uvedený způsob vyhodnocování dotazů a poskytuje přístup k údajům použitým pro výpočet váhy jednotlivých výrazů. Např. je možné z indexu získat počet dokumentů obsahujících zadaný výraz nebo počet výskytů zadaného výrazu v konkrétním dokumentu. Tyto údaje jsou přímo využívány pro řešení dalších problémů, jako je klasifikace inzerátů či vyhledávání podobných inzerátů. Kromě běžného vyhodnocování dotazů umožňuje knihovna filtrování záznamů dle zadaných kritérií. Filtrování omezí množinu výsledných záznamů tak, že záznamy neodpovídající zadanému filtru neovlivní skóre ostatních dokumentů. Filtrování je využito např. pro omezení inzerátů patřících pouze do vybrané kategorie nemovitosti (byt, dům, komerční objekt, apod.) Implementace *Tf-idf* je v prototypovém řešení obsažena ve třídě `Thon.WebMining.Utils` v projektu `Thon.WebMining`. Tato implementace je využita při klasifikaci a shlukové analýze.[8]

2.4 Sběr a předzpracování dat

Abychom mohli v IR systému vytvořit databázi dokumentů a zejména index pro zajištění vyhledávání, je potřeba nejprve získat požadovaná data a určitým způsobem je předzpracovat. Předzpracování dat je nutné i pro následné vytvoření a aplikování data miningových modelů. Ty nám umožní získat další zajímavé údaje, které nejsou přímo uloženy v obsahu inzerátu. V případě této práce se jedná o modely zajišťující automatickou klasifikaci inzerátů a vyhledávání podobných inzerátů.

Sběr dat je v případě realitních serverů relativně jednoduchá záležitost. V zásadě jde o stažení webové stránky pomocí HTTP protokolu. Implementace HTTP protokolu je dostupná pro většinu vývojových platforem[9] a nebudeme se jí blíže zabývat. Následně je nutné zpracovat obsah získané HTML stránky. Musíme samozřejmě rozlišit stránky obsahující celý obsah jednoho inzerátu a stránky obsahující pouze jejich seznam. Většina realitních portálů zobrazuje nejprve seznam inzerátů, ze kterého je možné se pomocí odkazů navigovat na konkrétní inzeráty. Část systému, která provádí takové procházení webových stránek se nazývá *crawler*. Podrobnosti o zpracování a procházení webových stránek a o návrhu konkrétního řešení pro potřeby zpracování realitní inzerce jsou uvedeny v kapitole 3.

Složitější než samotný sběr dat je v případě realitní inzerce jejich předzpracování. Ne všechny realitní portály totiž poskytují údaje strukturované. Např. na některých realitních portálech je cena inzerátu uvedena v samostatné části HTML stránky a je možné při jejím parsování jednoznačně identifikovat HTML element, ve kterém je cena inzerátu uvedena. Na některých portálech je cena inzerátu uvedena přímo v textu inzerátu. Je tedy zřejmě, že určitá část předzpracování dat musí být provedena pro každý realitní portál samostatně. V

8 Pojem shluková analýza je blíže vysvětlen v kapitole 2.5.

9 V prostředí .NET Frameworku můžeme např. využít API `System.Net.WebClient` nebo `System.Net.HttpWebRequest`.

Název	**Hodnota**
Nadpis	S - Prodej udržovaného družstevního bytu 1+kk, 38 m², Praha 17 Řepy, ul. Bazovského
Cena	1 430 000 Kč
Text	Prodej udržovaného družstevního bytu 1+kk, 38 m2, Praha 17 Řepy, ul. Bazovského. Nabízíme k prodeji pěkný byt 1+kk v družstevním vlastnictví, nacházející se ve 4. patře zrekonstruovaného panelového domu s výtahem v Praze Řepích. ...
Lokalita	Praha 6, Řepy, Bazovského
Číslo zakázky	RI-AN7631T6
Počet podlaží budovy	8
Poschodí	5
Užitná plocha	38 m
Konstrukce budovy	panelová
Stav budovy	po celkové rekonstrukci
Stav bytu	po rekonstrukci
Lokalita objektu	centrum obce / města
Topení	ústřední - dálkové

Zdroj: http://reality.idnes.cz/detail/prodej/byt/1+kk/praha-repy-bazovskeho/6968506 ze dne 15. 9. 2013

Příklad 2.1: Příklad datové struktury pro uchování inzerátu v pomocné databázi, která je výstupem crawleru. V tabulce je uveden příklad dobře strukturovaných údajů zveřejněných realitním portálem.

případě této práce je tato část předzpracování součástí crawleru, jehož výstupem je datová struktura obsahující textové údaje ve formě dvojic: [název, hodnota]. V naprosté většině případů je možné z inzerátu získat minimálně dvě takové dvojice, z nichž jedna obsahuje nadpis inzerátu, druhá text. Velmi často je možné získat z inzerátu další údaje, jako je cena, lokalita, dispozice bytu a další údaje o nemovitosti. V příkladu 2.1 je vidět obsah takové datové struktury. V některých případech je vhodné inzerát strukturovat do více úrovní, kdy hodnotou jedné dvojice v datové struktuře není prostý text, ale další seznam dvojic [název, hodnota]. Další zpracování inzerátů probíhá již pouze s využitím takto získané datové struktury.

V této fázi předzpracování je možné z připravené datové struktury získat některé údaje pomocí technik *scrapingu*. Příkladem je již zmíněná cena inzerátu, jejíž získání je ve většině případu řešitelné pomocí regulárních výrazů či jednoduchých pravidel. U portálů, které uvádějí cenu inzerátu odděleně od ostatního textu, stačí z připravené datové struktury použít tu, která obsahuje cenu. V ostatních případech je nutné cenu vyhledávat v textu inzerátu. Některé složitější problémy získání údajů z textu inzerátu jsou uvedeny v kapitole 2.4.1. V této práci je jako součást prototypového řešení implementováno získání ceny inzerátu. Řešení tohoto problému je podrobněji popsáno v kapitole 8.4.

Další fází předzpracování je zpracování textu inzerátu pro účely vytvoření indexu a některých data miningových modelů. Tento proces začíná tzv. *tokenizací textu,* tedy jeho rozdělením na slova či výrazy. Z jednotlivých slov jsou vypuštěna tzv. *stop slova*, která nemají žádný význam pro další zpracování. Jedná se o slova, která nenesou žádnou informaci týkající se nemovitosti, jako jsou například předložky, spojky, ukazovací

Původní text	Výsledek předzpracování
Prodám byt v Českých Budějovicích.	prod byt česk budějovik
Pronájem bytu v Českých Budějovicích.	pronáj byt česk budějovik
Koupím dům se zahradou.	koup dom zahrad

Tabulka 2.1: Výsledek předzpracování textu pomocí knihovny Apache Lucene.NET a implementace stemmeru pro český jazyk

zájmena, některé tvary sloves být a mít, apod. Zbývající významová slova jsou převedeny na určitý normalizovaný tvar. Existují v zásadě dva přístupy k normalizaci slov. Jedním z nich je převedení na základní tvar, kterým je u podstatných jmen tvar v 1. pádu jednotného čísla, u sloves pak 1. osoba jednotného čísla v přítomném čase. Tento způsob normalizace je nazýván *lemmatizace*. V případě, kdy nepotřebujeme koncovému uživateli prezentovat základní normalizovaný tvar slova, postačí jednodušší způsob zpracování, při kterém ze slova pouze odstraníme koncovky a získáme tak kořen slova. Důležité je pouze to, aby různá slova mající stejný kořen byla převedena pokud možno na stejný základní tvar. Tento proces je označován jako *stemming* a je výrazně snáze implementovatelný za pomoci relativně jednoduchých algoritmů. V případě použití jednoduchého algoritmu je však potřeba počítat s určitou mírou nepřesnosti, kdy výsledkem zpracování určitého slova není jeho správný kořen. V tabulce 2.1 je vidět výsledek předzpracování textů krátkých inzerátů z obrázku 2.1 (strana 7).

Jako implementace nástrojů pro předzpracování textu v této práci je použita již zmíněná knihovna Apache Lucene.NET. V rámci práce byla převzata implementace stemmeru pro český jazyk,[10] která v této knihovně chybí, nicméně je obsažena v původní knihovně Apache Lucene pro prostředí Java, na které je Apache Lucene.NET založena.

2.4.1 Problémy při získávání údajů z textu

Jak již bylo uvedeno v předchozí kapitole, součástí práce je i jednoduché řešení získání ceny z textu inzerátu. Pro zajištění určité spolehlivosti je nutné pamatovat na různé způsoby formátování ceny. Jde o různé použití oddělovačů řádů (mezera, popř. jiný oddělovač, nahrazení bezvýznamových nul zkratkou (např. 10 tis. namísto 10 000), různé označení jednotky (např. Kč, Kč/měsíc nebo symbol ,--). Dalším problémem může být uvedení několika cen v textu inzerátu, z nichž pouze jedna má požadovaný význam a ostatní jsou pouze doplňující (např. kromě ceny nájmu může být v inzerátu uvedena výše kauce a samostatně uvedená výše poplatků).

Podobným problémem jako v případě zjištění ceny nemovitosti může být problém vyhledání kontaktních údajů. Ve většině případů je tento problém opět řešitelný technikami scrapingu. Pomocí regulárních výrazů je možné popsat obecný zápis e-mailové adresy, internetové adresy nebo telefonního čísla. Takové údaje mohou být dále využity např. pro ztotožnění inzerenta, který zveřejňuje několik různých inzerátů, nebo jen pro přehledné zobrazení ve výsledcích vyhledávání.

Složitějším problémem je zjištění lokality z textu inzerátu, která navíc nemusí být přesně určena. Pro vyhledání názvů ulic, obcí, měst či krajů bychom mohli využít nějaký předem připravený seznam. Vhodným zdrojem by mohla být např. veřejně dostupná

10 Konkrétně jde o třídu `Thon.WebMining.Utils.Analyzers.CzechStemmer` v projektu `Thon.WebMining`.

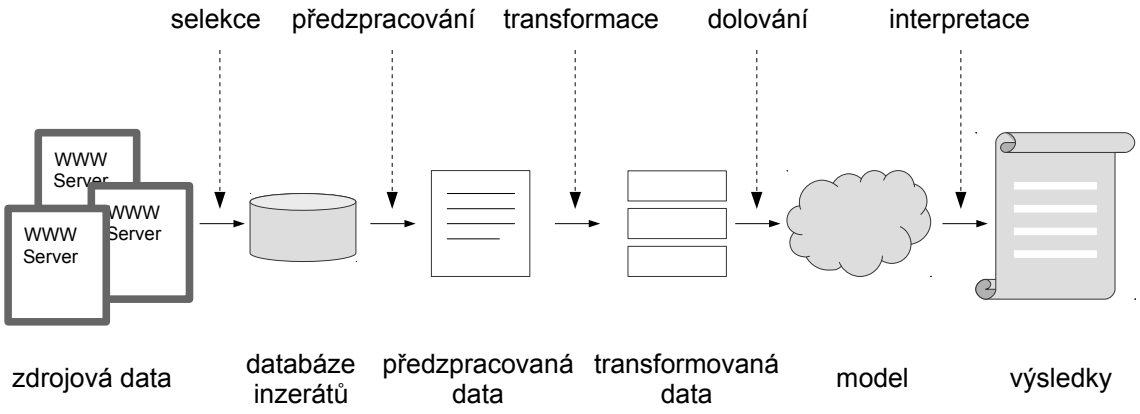
databáze RUIAN (Registr územní identifikace, adres a nemovitostí), kterou spravuje Český úřad zeměměřičský a katastrální. Výhodou databáze RUIAN je to, že obsahuje kompletní seznam adresních míst v České republice. Byla by tedy využitelná pro vyhledávání správnosti ulic v určité obci nebo pro zjištění do jakého okresu či kraje daná obec patří. Tato databáze dokonce obsahuje názvy obcí ve všech pádech kromě pátého, což umožňuje přesnější vyhledání názvu obce v textu inzerátu. I v případě využití takovéto referenční databáze je potřeba pamatovat na možné nepřesnosti v textu způsobené např. překlepy v názvu měst a obcí, popř. použitím zkratek u víceslovných názvů. Je zřejmé, že přesné zjištění lokality z nestrukturovaného textu inzerátu není triviální záležitostí.

2.5 Data Mining

Dolování dat neboli *Data mining* představuje proces získávání užitečných informací z dat. Jde přitom o informace, která nejsou v datech přímo obsažena. Takovými informacemi jsou často skryté závislosti v datech. Primárním zdrojem dat v tomto procesu jsou data uložená ve strukturovaných databázích. Specifickou disciplínou je dolování informací z textu neboli *Text mining*, která se zabývá zpracováním nestrukturovaných dat.

Data mining i text mining se mimo jiné zabývají dvěma základními úlohami. Jednou z nich je třídění dokumentů do předem určených skupin neboli tříd, tj. problém *klasifikace*. Druhou úlohou je seskupování textů podle podobnosti bez ohledu na pojmenování vzniklých skupin. Způsobem řešení tohoto problému je tzv. *shluková analýza*. V obou úlohách jde o nalezení jistých podobností mezi realitními inzeráty. Existuje celá řada dalších úloh, kterými se data mining zabývá, pro naplnění cílů této práce se budeme dále podrobněji věnovat dvěma výše uvedeným.

Proces data miningu zahrnuje kromě dolování informací několik dalších fází, kterými je nutné projít, abychom mohli ze zdrojových dat získat požadované informace a ty vhodnou formou prezentovat uživateli. Celý proces je znázorněn na obrázku 2.1. **Selekcí** dat je v našem případě získání inzerátů z realitních portálů. Selekce tedy přímo vyplývá ze zadání řešeného problému. **Předzpracování** dat zahrnuje vytvoření datové struktury obsahující jednotlivé části inzerátu popsané v kapitole 2.4 a znázorněné v příkladu 2.1 (strana 9), případně získání některých konkrétních údajů z inzerátu (v našem případě zejména získání ceny nemovitosti). V rámci **transformace** dat je proveden výběr



Obrázek 2.1: Proces dolování dat (s úpravami převzato z [5], kapitola 1, obr. 1)

specifických částí inzerátu a jejich převedení do formy, kterou vyžaduje konkrétní data miningový model. Tato fáze je tedy na rozdíl od předchozích odlišná pro řešení dvou řešených úloh. **Dolování dat** je realizováno aplikováním vytvořených modelu. Tato fáze celého procesu je předmětem samostatných kapitol 4 a 5.

Poslední, dosud nezmíněnou, fází procesu dolování dat je **interpretace** výsledků. Z pohledu celé práce má tato fáze významnou spojitost s problematikou Information Retrieval. Právě IR systém vytvořený v rámci této práce interpretaci výsledků zajišťuje. Jedná se o možnost filtrování inzerátů dle typu obchodu a kategorie nemovitosti. V systému jsou tyto údaje uloženy v indexu, nicméně jde o údaje, které nejsou přímo získány ze zdrojových dat, ale jsou na základě vytvořených modelů dolovány. Výsledky shlukové analýzy jsou interpretovány ve formě nabídky několika nejpodobnějších inzerátů k inzerátu zobrazeném z výsledků vyhledávání. To umožňuje uživateli další způsob navigace a prohledávání celé databáze inzerátů bez nutnosti vytváření jiného vyhledávacího dotazu.

2.6 Problém aktualizace inzerátů

Jedním zajímavým problémem, který byl již zmíněn v úvodu kapitoly, je problém aktualizace inzerátů. Pokud odhlédneme od požadavku na vysokou aktuálnost inzerátu, kterou některé komerční služby na internetu nabízejí i v řádu několika málo minut, stále nám zůstane základní problém. Tedy zjištění, zda dříve stažený inzerát je stále platný a zda nejsou na realitním portálu zveřejněny inzeráty nové, které ještě nejsou uloženy v databázi našeho IR systému. Naprosto samozřejmým je totiž požadavek, aby systém zpracovávající realitní inzerci nenabízel uživateli inzeráty, které již byly z původního portálu staženy. Naopak je žádoucí, aby systém umožňoval průběžné ukládání nových inzerátů do své databáze, včetně jejich následného zpracování.

Základním způsobem zjištění nově zveřejněných inzerátů není nic jiného než opakované procházení zdrojových portálů. V případě, že crawler nalezne odkaz na inzerát, jehož URL adresu ještě nemá ve své databázi, uloží jej a připraví jej pro další zpracování. U každého inzerátu navíc zaznamená čas uložení. V případě, kdy je nalezena adresa inzerátu, která je již v databázi uložena, může crawler vynechat její zpracování, ale provede aktualizaci času uložení inzerátu. Pro ověření platnosti dříve navštívené URL adresy existují 2 základní možnosti. Jednou z nich je prosté nastavení doby platnosti inzerátu v databázi, přičemž inzeráty, které jsou v databázi uloženy déle než je nastavený limit, jsou automaticky odstraněny. Toto řešení spoléhá na funkčnost crawleru, který u opakovaně navštívených inzerátů průběžně aktualizuje čas jejich uložení do databáze. Druhým způsobem je přímá kontrola dříve zjištěné URL adresy. Provedením nového HTTP požadavku na tuto adresu můžeme podle návratového kódu zjistit, zda je adresa stále platná. Zásadní nevýhodou tohoto řešení je skutečnost, že není možné spoléhat pouze na návratový HTTP kód získaný v odpovědi webového serveru. Realitní portály mohou záměrně z bezpečnostních důvodů pro všechny URL adresy vracet návratový kód `200 OK` i v případě, že již na adrese není zveřejněn žádný inzerát. V těle HTTP odpovědi je pouze textová poznámka informující uživatele portálu, že inzerát již neexistuje. Při použití této varianty řešení by tedy bylo nutné znát určitou strukturu chybové stránky příslušného portálu a rozlišit ji od stránek obsahující inzerát.

Konkrétní řešení problému aktualizace inzerátů překračuje rámec této práce, nicméně je s ním počítáno při návrhu algoritmů a data miningových modelů. V kapitole 5 je blíže popsán algoritmus K-Means použitý pro vyhledávání podobných inzerátů. V této kapitole je uvedeno, jakým způsobem je možné řešit provádění K-Means algoritmu přírůstkovým způsobem, tedy v případě, kdy dojde k uložení nových inzerátů do databáze, popř. k jejich odstranění. Dále je v kapitole 7 popsán způsob detekce duplicitních inzerátů. I v tomto případě návrh řešení pamatuje na časté změny obsahu databáze.

3 Získání dat

Každý systém, který získává data z veřejně dostupných webových stránek, musí určitým systematickým způsobem tyto webové stránky procházet a vyhledávat v nich požadované údaje. Princip procházení je ve všech případech stejný a je realizován pomocí internetových odkazů. Jedná se o naprosto stejný princip, který využívá každý uživatel internetu při procházení stránek. Problémem k řešení v počítačovém softwaru je automatizace tohoto procházení.

Při automatizaci procházení webových stránek je nutné každou stránku rozdělit minimálně na dvě části: odkazy směřující na další stránky a vlastní obsah stránky, který má být určen pro další zpracování. Pokud se při procházení webových stránek omezíme pouze na procházení obsahu realitních portálů, je navíc nutné provést určitou selekci odkazů a obsahu stránky. Je tedy nutné vhodným způsobem definovat strukturu webových stránek. U realitních portálů je typicky nutné rozlišit stránky obsahující seznam inzerátů a stránky obsahující kompletní obsah jednoho inzerátu. Ve stránce se seznamem inzerátů vyhledáváme odkazy na stránky s obsahy inzerátů. Na stránce s obsahem inzerátu vyhledáváme vlastní textový obsah, který může být částečně strukturovaný.

Výhodou tohoto přístupu automatizace, kdy máme předem definovanou strukturu webového portálu, je relativně snadné získání požadovaného obsahu v částečně strukturované formě. Provádíme tedy zároveň i určitou část předzpracování dat. Nevýhodou ovšem je odlišnost struktury každého realitního portálu. Musíme udržovat popis struktura pro každý portál, ze kterého chceme získávat data. Tento popis však není možné provést pouze jednorázově, ale je nutné jej udržovat s ohledem na to, jak se v průběhu času mění forma prezentace inzerátů na daném realitním portálu. Proces procházení, stažení a zpracování webových stránek je označován jako *crawling*.[1] V prototypovém řešení je realizován samostatným programem `crawler.exe`.

3.1 Zpracování HTML

Samotné zpracování webových stránek je řešeno v zásadě pouze parsováním HTML. Ačkoli je HTML standardizováno, zásadním problémem při jeho programovém zpracování je špatné formátování. Typickými chybami ve formátování HTML jsou neukončené elementy, překřížené elementy nebo chybějící uvozovky pro zápis hodnot atributů. Ve standardu HTML5[2] je popsán algoritmus parsování chybně strukturovaného HTML dokumentu. Příkladem implementace parseru, který tento algoritmus implementuje, je knihovna *jsoup*.[3] V prototypovém řešení je pro parsování použita knihovna *Html Agility Pack*,[4] která přímo neimplementuje uvedený algoritmus, nicméně umožňuje formou konfigurace určité přizpůsobení možným druhům chyb ve struktuře HTML stránky. Tato knihovna poskytuje jednoduché API pro zpracování HTML, které je velmi podobné XML

1 Viz [4], kapitola 20.1.
2 http://www.w3.org/TR/html5/, kapitola 8.2
3 http://jsoup.org/
4 http://htmlagilitypack.codeplex.com/

DOM. Použití je tedy velmi jednoduché, zejména pro vývojáře, kteří mají zkušenost se zpracováním XML v prostředí Microsoft .NET Frameworku.

Výsledkem úspěšného parsování webové stránky je datová struktura pro každý získaný inzerát, která již byla popsána v kapitole 2.4 a ukázána na příkladu 2.1 (strana 9). V prototypovém řešení jsou tyto datové struktury ukládány do SQL databáze.[5] V prototypovém řešení je vytvořeno jednoduché API, které umožňuje programově popsat strukturu webového portálu. Pomocí něj je možné specifikovat hledaný inzerát ve webové stránce a popsat jeho strukturu, která je výstupem crawleru. API je tvořeno několika základními třídami v projektu `Thon.WebMining`, v namespace `Thon.WebMining.Crawler.BasicHtmlSite`.

- Třída `Site` představuje konkrétní realitní portál a definuje sadu výchozích stránek pro procházení, tzv. *seed set*, ve formě seznamů instancí třídy `Page`.
- Třída `Page` uchovává URL adresu stránky a její typ definovaný třídou `PageType`.
- Třída `PageType` definuje obecnou strukturu webových stránek stejného typu. Struktura stránek je popsána dvěma typy elementů: odkaz na další stránku (instance třídy `FollowLinkElement`) s uvedením jejího typu (opět instance třídy `PageType`) a požadovaný obsah, tj. inzerát (instance třídy `DocumentElement`).
- Třída `DocumentElement` popisuje strukturu konkrétního realitního inzerátu ve formě položek instancí tříd odvozených od abstraktní třídy `Field`. Položka může být buď jednoduchá (třída `PlainField`) obsahující pouze název a textový obsah nebo může jít o skupinu položek (třída `FieldGroup`), která představuje seznam dalších jednoduchých položek či skupin položek.

Konkrétní částí webové stránky jsou v příslušných instancích výše uvedených tříd popsány formou XPath výrazů, které jsou využity pro zpracování konkrétního HTML dokumentu ve fázi parsování. Výsledkem parsování je pak datová struktura inzerátu uvedená v příkladu 2.1 (strana 9), která vznikne extrahováním textových údajů popsaných instancemi tříd `PlainField` a `FieldGroup`.

Součástí prototypového řešení je implementace čtyřech známých realitních portálů:

- Avízo.cz (http://reality.avizo.cz/),
- Bazoš.cz (http://reality.bazos.cz/),
- Inzerce pro Brno a okolí – ibyty.com (http://www.ibyty.com/) a
- Reality iDNES.cz (http://reality.idnes.cz/).

Třídy popisující uvedené portály jsou v namespace `Thon.WebMining.Portal`. Na příkladu portálu Reality iDNES.cz uvedeme, jakým relativně snadným způsobem je možné popsat strukturu tohoto portálu. Pro tento portál jsou definovány dva typy stránek: seznam inzerátů, detail inzerátu. Dále je určena jedna výchozí stránka pro procházení portálu typu seznam inzerátů:

```
http://reality.idnes.cz/s/?st=1&qs=&qs_loc=
```

5 V prototypovém řešení je konkrétně použita databáze SQLite, nicméně v projektu je vytvořena abstraktní datová vrstva (namespace `Thon.WebMining.DataLayer`) umožňující poměrně snadnou implementaci jiné SQL databáze.

Definice stránky se seznamem inzerátů obsahuje dva elementy typu odkaz z nichž jeden vždy odkazuje na detail inzerátu a je určen XPath výrazem

```
//table[@class='list-summary2']/tbody/tr/td[1]/a[1]/@href
```

druhý odkazuje na další stránku seznamu inzerátů a je určen XPath výrazem

```
//div[@class='nav2 ']/a[@class='pagination']/@href
```

Definice stránky s detailem inzerátu obsahuje element, jehož obsahem je inzerát a je určen XPath výrazem

```
/html/body/div[@id='main']//div[@class='full']
```

Podobným způsobem jsou dále definovány jednotlivé části inzerátu, jako je nadpis, cena, text a dvě skupiny dalších popisných údajů strukturovaných v HTML tabulce. Je vidět, že z tohoto portálu je možné získat relativně dobře strukturované údaje.

3.2 Architektura crawleru

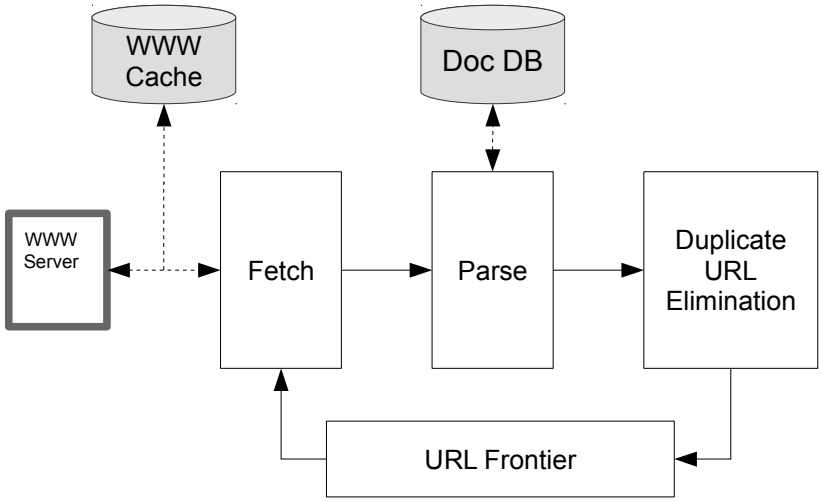
Při návrhu architektury crawleru je vhodné oddělit určité fáze zpracování webových stránek. Jak již bylo zmíněno v úvodu kapitoly, některé části zpracování jsou specifické pro konkrétní realitní portál, jiné jsou naopak společné. Zároveň je vhodné pamatovat při návrhu na možnost jednoduché úpravy v případě změny struktury portálu nebo v případě požadavku na získávání dat z nového portálu. Základní architektura crawleru určeného pro procházení obecného webového obsahu je popsána v [4], kapitola 20.2. Z této architektury vychází i prototypové řešení, které je součástí práce. Pro potřeby zpracování konkrétního typu webového obsahu, v našem případě realitní inzerce, je tato architektura patřičně upravena.

Návrh architektury je znázorněn na obrázku 3.1. Činnost crawleru je vlastně nekonečný proces, který je řízen komponentou zvanou **URL Frontier**. Jde v podstatě o frontu, ve které jsou uloženy jednotlivé webové stránky, resp. příslušné URL adresy stránek. Při zahájení procesu jsou do URL frontieru zařazeny výchozí stránky pro procházení webového portálu, tedy již zmíněný *seed set*.

Proces crawleru zajistí vyzvednutí jedné stránky z fronty a předá ji komponentě **fetch** zajišťující získání obsahu požadované stránky z webového serveru pomocí HTTP protokolu a uložení celého obsahu do pomocné databáze označené jako **WWW Cache**. V případě, že je již požadovaná stránka v této databázi uložena, je pouze načtena bez nutnosti nového stažení z webového serveru. Toho je možné využít např. pro opakované zpracování stejné stránky v případě potřeby opravy chyby při zpracování. Dále je tato pomocná databáze použita pro poskytnutí původního obsahu zdrojové stránky uživateli.[6]

Načtená webová stránka je předána komponentě označené jako **parse**, která zajišťuje zpracování HTML způsobem popsaným v kapitole 3.1. Výsledkem parseru je datová struktura dokumentu, která je v prototypovém řešení implementována v namespace `Thon.WebMining.Documents`. Tato komponenta zajistí uložení získaných inzerátů do

6 Tato funkce je dostupná v uživatelském rozhraní, které je součástí prototypového řešení popsaného v kapitole 8.



Obrázek 3.1: Architektura crawleru – základní komponenty a postavení dvou úložišť používaných crawlerem v procesu zpracování webových stránek

provozní databáze označené jako **Doc DB**, ve které je u každého inzerátu uložen částečně strukturovaný textový obsah, URL adresa, ze které byl získán a datum stažení. Datum stažení by byl využitelný pro případnou aktualizaci inzerátů.

Při parsování stránek zároveň dochází k vyhledání odkazů na další stránky portálu, které mají být zpracovány. Jejich URL adresy parser předá komponentě **Duplicate URL Elimination**, která provede kontrolu, zda již nebyla příslušná URL adresa jednou zpracována. Typicky stránky se seznamy inzerátů obsahují odkazy na několik dalších stránek. První stránka může obsahovat odkazy na stránky 2 až 5. Druhá stránka však bude obsahovat odkaz zpět na první stránku a dále na stránky 3 až 6. Při opětovném zpracování každé stránky by došlo jednoduše k zacyklení crawleru, který by se snažil procházet stále stejné stránky. Tato komponenta tedy zajistí zpracování každé stránky pouze jednou. Existuje několik možností, jakým způsobem tuto kontrolu implementovat. V prototypovém řešení je dostupná jednoduchá implementace, která zajišťuje uchovávání unikátních URL adres v paměti. Kontrola opakovaného navštívení webové stránky je provedena pouhou kontrolou existence dané URL adresy v množině již navštívených adres. Implementace v reálném systému by byla možná např. pomocí kontroly každé URL adresy v databázi uložených inzerátů.

Nově získané URL adresy, které nebyly dosud navštíveny, jsou zařazeny do URL frontieru a celý proces zpracování může pokračovat. Proces crawleru může pracovat teoreticky neomezeně s tím, že pokud URL frontier neobsahuje žádnou URL adresu, je proces blokován. V prototypovém řešení je crawler realizován jednoduchou konzolovou aplikací `crawler.exe`, která v okamžiku zpracování všech adres v URL frontieru ukončena. Kromě výchozího seznamu URL adres a adres získaných při zpracování webových stránek je možné do URL frontieru zařazovat stránky odpovídající dříve získaným inzerátům, u kterých požadujeme jejich aktualizaci. Problém aktualizace byl již nastíněn v kapitole 2.6 a jeho řešení přesahuje rozsah této práce.

V prototypovém řešení jsou jednotlivé komponenty crawleru definovány pomocí několika rozhraní v namespace `Thon.WebMining.Crawler`:

- `IUrlFrontier` představuje frontu stránek pro zpracování, přičemž každá stránka je definována rozhraním `IPage`,
- `IHttpClient` představuje komponentu zajišťující stažení webových stránek z webových serverů,
- `IParser` představuje komponentu zajišťující zpracování HTML dokumentu,
- `IDuplicateUrlElimination` zajišťuje kontrolu již navštívených stránek.

Celý proces je implementován ve třídě `CrawlerProcess`, která pro svou inicializaci vyžaduje přístup k databázi inzerátů a definici jednoho realitního portálu, který je definován rozhraním `ISite`. To poskytuje crawleru výchozí množinu webových stránek daného portálu pro spuštění procházení a zpracování obsahu. Konkrétní příklad spuštění crawleru je uveden v kapitole 8.

3.3 Zajištění spolehlivosti

V prototypovém řešení je implementovaná pouze jednoduchá varianta crawleru, který je spuštěn v jednom procesu konzolové aplikace a dokáže v jednu chvíli zpracovat jeden určený realitní portál. Navržená architektura ovšem poskytuje možnosti provozu v prostředí s požadavky na zajištění vysokého výkonu a spolehlivosti.

Základním požadavkem na zajištění spolehlivosti je možnost obnovy procesu crawleru v případě jeho pádu. Jak již bylo popsáno, jedinou komponentou, která celý proces řídí a zároveň udržuje stav crawleru je *URL Frontier*. Pro zajištění obnovy procesu tedy stačí obsah URL frontieru průběžně ukládat do bezpečného úložiště. Pokud opětovně spustíme proces crawleru, je možné načíst obsah URL frontieru z úložiště, namísto načtení výchozí sady webových stránek. Proces procházení portálů pak pokračuje stejnou stránkou, kterou při pádu skončil.

S minimálními úpravami je možné spustit paralelně více procesů crawleru, přičemž každý proces zajistí zpracování jednoho konkrétního realitního portálu. V tomto případě je každý proces řízen samostatným URL frontierem. Toto řešení je relativně snadno implementovatelné, je však nutné zajistit a sledovat provoz více nezávislých procesů z nichž každý používá vlastní úložiště pro uložení stavu URL frontieru.

Jiným řešením by bylo paralelní zpracování webových stránek, které jsou získávány z jednoho společného URL frontieru. Takové řešení by naopak zajistilo zachování provozu v případě pádu některého z procesů. Webové stránky by jednoduše byly zpracovány jiným procesem, jelikož si je všechny procesy vyzvedávají ze společného URL frontieru. Slabým místem tohoto řešení je riziko nedostupnosti samotného URL frontieru. V tom případě by ostatní procesy nemohly vykonávat svou činnost.

Pro zajištění vysokého výkonu může být každá komponenta crawleru realizována samostatným nezávislým procesem poskytujícím jednu konkrétní službu. Všechny komponenty crawleru s výjimkou URL frontieru jsou totiž v principu bezstavové a pouze transformují zadaný vstup na požadovaný výstup. Celý proces je pak popsán ve formě orchestrace. V takovém případě je crawler realizován jedním distribuovaným systémem s použitím vhodného middleware. Z uvedených případů použití je vidět, jak zdánlivě jednoduchá komponenta URL frontier představuje životně důležitou součást crawleru.

4 Klasifikace inzerátů

Problém klasifikace je jednou ze základních úloh data miningu. Jejím předmětem je třídění vzorů, resp. dokumentů,[1] do předem daných skupin neboli *tříd*. V případě realitní inzerce je dokumentem jeden konkrétní realitní inzerát a třídami mohou být již zmíněné: typ obchodu a kategorie nemovitosti.[2] Jedním z cílů práce je vytvoření klasifikačního modelu, který zajistí třídění inzerátů dle typu a kategorie, a to i přes to, že pocházejí z různých zdrojů a jejich forma se může značně lišit.

Poněkud odlišný pohled na problém klasifikace z pohledu IR systému je uveden v [4].[3] Dle uvedeného zdroje je problémem klasifikace zobecnění určitého typu uživatelských požadavků při vyhledávání, resp. získávání informací. Uživatel systému intuitivně předpokládá, že hledaná informace může být obsažena v dokumentech, které mají určité společné vlastnosti, tj. patří do stejné třídy. Samozřejmě nástroje fulltextového vyhledávání umožňují kombinovat vyhledávací dotaz složením několika výrazů, které nejlépe charakterizují požadovanou třídu dokumentů, pomocí logických operátorů. Pokud by v případě realitní inzerce uživatel požadoval vyhledávat pouze inzeráty, ve kterých inzerent poptává chatu, mohl by formulovat následující dotaz:

```
(koupím OR hledám OR sháním) AND (chata OR chalupa)
```

V takovém dotazu se pokoušíme prvními třemi výrazy charakterizovat skupinu inzerátů spadajících do třídy inzerátů s nabídkami, zbylé dva výrazy pak určují skupinu inzerátů týkající se rekreačních chat. Pokud k takovému dotazu připojíme další výrazy pro upřesnění hledaných inzerátů, výsledný dotaz bude pro většinu uživatelů obtížně srozumitelný. Nehledě na problémy způsobené neúplným vyjmenováním výrazů popisujících danou třídu dokumentů nebo naopak uvedením velkého množství výrazů, které množinu relevantních výsledků příliš rozšíří. Cílem je tedy vytvoření automatizovaného způsobu třídění inzerátů tak, aby měl uživatel jednoduchou možnost zvolit, v jaké třídě inzerátů se má vyhledávat. Třídu však uživatel neurčuje zadáním několika výrazů, ale přímo vybírá jednu z několika tříd, které mu systém nabídne.

Jednoduchým přístupem k řešení problému by mohlo být triviální roztřídění inzerátu podle výskytu nějakého klíčového slova s tím, že inzeráty neobsahující žádné předem definované klíčové slovo budou patřit do třídy „nezařazeno“. Další možností by bylo získání údaje pro třídění inzerátů přímo z realitního portálu, na kterém je inzerát zveřejněn. Většina realitních portálů nabízí možnost vyhledávání inzerátů dle zadané kategorie. Ne všechny portály ale tento údaj poskytují v obsahu inzerátu nebo jako součást URL adresy. Kromě toho, množina používaných kategorií se může na různých portálech lišit.

---

1 Vzorem je myšlena jedna datová n-tice. V prostředí nestrukturovaných textových dokumentů se namísto vzoru obvykle používá označení dokument.

2 Viz kapitola 1.1.

3 Obecný problém klasifikace textu je nastíněn v [4], v úvodu kapitoly 13 - Text classification and Naive Bayes. V této práci je problém zaměřen na klasifikaci textu realitních inzerátu.

Cílem je nalézt pokud možnost univerzální způsob klasifikace inzerátů získaných z různých zdrojů. Je zřejmé, že klasifikaci je možné provádět pouze na základě informací obsažených přímo v textu inzerátu, přičemž základní jednotkou informace je obvykle slovo. Klasifikaci je tedy možné provádět pouze pro dolování dat, která jsou nějakým způsobem v textu vyjádřena, ale není možné je jednoduchým způsobem číst. Za tím účelem je v této práci popsáno a implementováno několik klasifikačních metod. Všechny metody použité v této práci jsou využité pro klasifikaci inzerátů na základě jejich textového obsahu. Základním předpokladem pro úspěšnou klasifikaci je předpoklad, že informace o typu obchodu a kategorii nemovitosti jsou v textu inzerátu nějakým způsobem vyjádřeny.

Pravděpodobně nejznámější metodou je *Naive Bayes*, neboli *naivní bayesovská klasifikace*. Jde o statistickou metodu klasifikace a v práci jsou popsány dvě její varianty. Dalšími metodami jsou metody založené na porovnání vzdálenosti dokumentů ve *Vector Space Modelu*.[4] Ve všech případech se jedná o metody strojového učení s učitelem. Pro vytvoření spolehlivého klasifikačního modelu tedy potřebujeme mít k dispozici sadu dokumentů, u kterých máme předem určené požadované třídy. Jedna část dat je použita ve fázi učení modelu, jde tedy o tzv. *trénovací množinu*. Druhá část dat, tzv. *testovací množina*, je pak použita pro ověření správnosti modelu a následné vyhodnocení.

Existují i další metody strojového učení pro řešení problému klasifikace, které nejsou v rámci této práce představeny. Jako příklad je možné uvést tzv. *Support Vector Machines*. Jde o skupinu klasifikačních metod, které primárně řeší problém rozdělení dokumentů do dvou tříd. Existují způsoby, jak kombinovat více takových klasifikátorů pro řešení problému třídění dokumentů do více tříd. Dalším příkladem jsou tzv. *rozhodovací stromy*. Rozhodovací strom je klasifikační model tvořený pravidly organizovanými ve formě stromu. Uzly stromu představují pravidla pro rozhodování, listy stromu pak třídy.

4.1 Příprava dat

Před tím, než je možné vytvořit klasifikační model, je nutné mít dostatečné množství předzpracovaných dat, která budou použita pro učení a následné ověření modelu. Jak bylo uvedeno v úvodu kapitoly, u každého dokumentu v této množině dat potřebujeme znát třídu. Bylo by možné takovou množinu dat připravit ručně, tedy přiřadit každému dokumentu požadovanou třídu a tu uložit do pracovní databáze. Naštěstí v případě realitní inzerce můžeme využít data získaná z portálů *Reality iDNES.cz*[5] (dále jen iDNES) a *Inzerce pro Brno a okolí – ibyty.com*[6] (dále jen ibyty.com), které poskytují potřebné údaje jako součást URL adresy každého inzerátu. Jednoduchým oddělením určité části URL adresy tedy získáme označení třídy – **typ** nebo **kategorii**[7] inzerátu. Předpokládáme, že inzeráty jsou na portálu tříděny správně a případnou chybu ve zdrojových datech zanedbáváme.

Pro potřeby této práce bylo získáno 10 688 inzerátů z portálu iDNES a 90 inzerátů z portálu ibyty.com. Tyto dva portály byly záměrně zvoleny z toho důvodu, že poskytují inzeráty ve velmi odlišné formě. Na portálu iDNES jsou k dispozici dobře strukturované

4 Pojem Vector Space Model je podrobněji vysvětlen v kapitole 2.3.
5 http://reality.idnes.cz/
6 http://www.ibyty.com/
7 Označení kategorie je v URL adrese dostupné pouze u portálu iDNES.

Zdroj: http://reality.idnes.cz/detail/**prodej**/**dum**/radovy/most-slovanska/6968494
Ze dne: 15. 9. 2013
Nadpis: Prodej, rodinný dům, 3+1, Most, ul. Slovanská
Text: Předmětem nabídky je prodej rodinného domu 3+1 v klidné části města Mostu …
Typ obchodu: **prodej** *Kategorie nemovitosti*: **dum**

Zdroj: http://reality.idnes.cz/detail/**pronajem**/**byt**/3+kk/praha-5/6968491
Ze dne: 15. 9. 2013
Nadpis: Pronájem, byt 3+kk, 70 m², Praha 5 - Smíchov
Text: Pronájem, světlého, nově zrekonstruovaného bytu 3+kk ve 2. NP o CP 70 m2 …
Typ obchodu: **pronajem** *Kategorie nemovitosti*: **byt**

Zdroj: http://www.ibyty.com/main.php?choice=11&ID_INZERATU=4549&NABPOP=**poptavka**
Ze dne: 20. 7. 2014
Nadpis: Koupím byt
Text: Koupím byt v Brně v blízkosti Kampusu do 1000000 Kč.
Typ obchodu: **poptavka** *Kategorie nemovitosti*: neuvedeno (klasifikováno jako **byt**)

Příklad 4.1: Vybrané inzeráty z trénovací množiny. Tučně jsou zvýrazněna označení tříd.

inzeráty s množstvím údajů uvedených v textu inzerátu nebo samostatně v částečně strukturované formě. Naproti tomu portál ibyty.com poskytuje pouze inzeráty tvořené krátkým nestrukturovaným textovým obsahem. Tento rozdíl je pochopitelně dán i odlišným zaměřením obou realitních portálů. Inzeráty jsou uloženy v provozní databázi tak, jak bylo popsáno v kapitole 3 a bylo provedeno indexování jejich textového obsahu tak, jak bylo uvedeno v kapitole 2.4. Dle typu klasifikačního modelu je použit buď text inzerátu uložený v databázi nebo tzv. *Term Vector* nebo také *Term Frequency Vector*, který je uložen v indexu. Každá složka tohoto vektoru odpovídá základnímu tvaru jednoho slova obsaženého v inzerátu, hodnotou je počet výskytů daného slova.

V příkladu 4.1 je uvedeno několik inzerátů využitých při tvorbě klasifikačních modelů. U každého inzerátu je uvedena URL adresa zdroje, ve které je tučně zvýrazněno označení typu obchodu nebo kategorie nemovitosti, nadpis inzerátu a část textu. Označení v URL adrese je pro potřeby klasifikace použito jako označení třídy. Na základě výsledků klasifikace byla vypuštěna kategorie nemovitosti *dražba*. Takto označených inzerátů je v databázi prototypového řešení relativně málo a některé portály takový typ vůbec nerozlišují. Vytvořené klasifikační modely většinu takových inzerátů klasifikují jako *prodej*. Uvažujeme-li, že dražba je ve skutečnosti specifický typ prodeje nemovitosti můžeme pro zjednodušení tyto dva typy obchodu sloučit a považovat je za prodej. Výsledný výčet typů a kategorií použitých v prototypovém řešení je uveden v příkladu 4.2. Výsledkem provedené klasifikace je doplnění těchto typů a kategorií do indexů za účelem zajištění uživatelského filtrování inzerátů.

Typy inzerátů	**Kategorie inzerátů**	
poptavka	byt	komercni-nemovitost
prodej	chata-chalupa	maly-objekt-garaz
pronajem	dům	pozemek

Příklad 4.2: Typy a kategorie inzerátů, které jsou výsledkem klasifikace provedené v prototypovém řešení.

4.2 Vyhodnocení klasifikačního modelu

Získání a předzpracování dat je nutnou podmínkou pro vytvoření klasifikačního modelu. Existuje mnoho metod klasifikace, přičemž každá je vhodná pro řešení jiné úlohy. Výběr metody, která poskytne uspokojivé výsledky využitelné v reálném systému, je tedy nutné provést na základě určitého vyhodnocení úspěšnosti klasifikačního modelu. Vyhodnocení modelu je možné provést jednoduše tak, že aplikujeme vytvořený model na dokumenty v testovací množině, které nebyly použity při učení. Tím získáme sadu dokumentů, u kterých máme informaci o tom do jaké třídy patří a jaká třída byla určena klasifikačním modelem. Na základě vhodného porovnání takto získaných údajů můžeme usuzovat, jak spolehlivá je zvolená metoda klasifikace.

Pro účely této práce byla zvolena základní metoda pro vyhodnocení klasifikačního modelu – tzv. *Split Validation*. Trénovací a testovací množina jsou vytvořeny rozdělením předzpracovaných dat v poměru 70% pro trénovací množinu a 30% pro testovací, přičemž zůstává zachováno poměrné zastoupení všech tříd v trénovací i testovací množině. Po vytvoření modelu a jeho aplikování na data v testovací množině je sestavena tzv. *matice záměn*. Jde o matici typu $n \times n$, kde n je počet tříd. Hodnotou prvku a_{ij} je počet dokumentů, které patří do třídy i a modelem jim byla přiřazena třída j. Je tedy zřejmé, že na diagonále matice jsou počty správně klasifikovaných dokumentů, mimo diagonálu jsou počty chybně klasifikovaných dokumentů. Z matice záměn je možné získat ukazatele, pomocí kterých je možné vyhodnotit úspěšnost klasifikace. Mezi základní ukazatele patří:

- **Precision** (*přesnost*[8]) Jde o poměr správně klasifikovaných dokumentů a všech dokumentů. Může být vyjádřen buď celkově nebo pro danou třídu i jako:

$$\text{precision} = \frac{\sum_{i=1}^{n} a_{ii}}{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}} \qquad \text{precision}_i = \frac{a_{ii}}{\sum_{j=1}^{n} a_{ij}} \quad .$$

- **Recall** (*úplnost*) Jde o poměr správně klasifikovaných dokumentů a všech dokumentů v dané třídě j vyjádřený jako:

$$\text{recall}_j = \frac{a_{jj}}{\sum_{i=1}^{n} a_{ij}} \quad .$$

- **F-measure** (*F-míra*) Jde o ukazatel, který kombinuje přesnost a úplnost dané třídy c. Jeho hodnota je vysoká v případě, že je vysoká zároveň přesnost i úplnost. V případě, že hodnota jednoho z dílčích ukazatelů klesne, klesá i hodnota F-measure. Hodnota tohoto ukazatele je dána vztahem:

$$\text{F-measure}_c = \frac{2 \cdot \text{precision}_c \cdot \text{recall}_c}{\text{precision}_c + \text{recall}_c} \quad .$$

Pro potřeby této práce se jako vhodný ukazatel pro celkové vyhodnocení úspěšnosti klasifikace jeví F-measure. Na příkladu porovnání klasifikačních metod Naive Bayes a Rocchio s použitím kosinové vzdáleností bude vidět, že ačkoli je celková přesnost jednoho z uvedených modelů mírně vyšší, je u něj výrazně nižší průměrná hodnota F-measure. To je způsobeno malým zastoupením dokumentů v jedné ze tříd, přičemž s vzhledem k

8 Dle [5], kapitola 6.1.1 také správnost, resp. celková správnost.

relativně malému množství chybně klasifikovaných dokumentů výrazně klesne hodnota ukazatele úplnosti, a tedy i hodnota F-measure, pro danou třídu. V takovém případě by se mohlo stát, že ačkoli by měl jeden klasifikační model vyšší přesnost, při klasifikaci by téměř nebyla využita jedna konkrétní třída. V závislosti na tom, zda požadujeme pouze maximální míru přesnosti, nebo je důležitější určitá přesnost i pro méně zastoupené třídy, je možné jako úspěšnější klasifikační model hodnotit model s vyšší průměrnou hodnotou ukazatele F-measure, i když má nižší celkovou přesnost.

V prototypovém řešení je celý proces klasifikace včetně vyhodnocení realizován samostatnou konzolovou aplikací `classifier.exe`. Aplikace zajistí předzpracování dat, rozdělení na trénovací a testovací množinu, vytvoření zvoleného klasifikačního modelu a jeho vyhodnocení. Výsledek vyhodnocení je vypsán v podobě matice záměn doplněné o hodnoty výše uvedených ukazatelů úspěšnosti klasifikace. Implementace metody *Split Validation* a uvedených ukazatelů úspěšnosti klasifikace je dostupná ve třídách `Validation` a `ValidationResult`.[9] Výsledný klasifikátor může být následně využit pro vytvoření finální podoby indexu využitelného v prototypu IR systému. Podrobnosti o konkrétním způsobu využití výsledků klasifikace jsou popsány v kapitole 8.

4.3 Naive Bayes

Jednou z typických metod strojového učení pro řešení úlohy klasifikace textu je *Naive Bayes*. Jde o statistickou metodu, která využívá pravděpodobnostní model pro vyjádření příslušnosti dokumentu *d* ke třídě *c* jako podmíněnou pravděpodobnost *P(c|d)*. V této práci jsou představeny dvě varianty této metody. Varianta *multinomial* zohledňuje při výpočtu pravděpodobností výskyt všech slov obsažených v dokumentech ve trénovací množině. Podmíněná pravděpodobnost příslušnosti dokumentu ke třídě je dána vztahem:

$$P(c|d) = P(c)\prod_{k=1}^{n_d} P(t_k|c) \quad ,$$

kde *P(c)* je apriorní pravděpodobnost výskytu dokumentu ve třídě *c*, n_d je počet výrazů v dokumentu *d* a $P(t_k|c)$ je podmíněná pravděpodobnost, která udává, jakou měrou ovlivní výraz *t* příslušnost dokumentu právě ke třídě *c*. Cílem klasifikace je nalézt nejvhodnější třídu pro daný dokument. Nejlepší třídou při použití tohoto modu je třída, pro kterou je podmíněná pravděpodobnost *P(c|d)* nejvyšší. Hledáme tedy tzv. *maximální aposteriorní pravděpodobnost* (MAP) c_{MAP} třídy *c* v množině tříd *C*, tj.:

$$c_{MAP} = \arg\max_{c \in C} P(c|d) \quad .$$

Z důvodů výpočetního výkonu a možné ztráty přesnosti při výpočtu v plovoucí řádové čárce se používá odlišné vyjádření hodnoty c_{MAP}, ve kterém je součin dílčích pravděpodobností nahrazen součtem jejich logaritmů, tj,:

$$c_{MAP} = \arg\max_{c \in C} \left(\log P(c) + \sum_{k=1}^{n_d} \log P(t_k|c)\right) \quad .$$

Výsledná hodnota c_{MAP} bude odlišná, ale stále bude platit, že vyšší hodnota odpovídá vyšší aposteriorní pravděpodobnosti. Je zřejmé, že tento model je velmi citlivý na struktuře trénovacích dat, zejména pak na správném poměru zastoupení všech tříd, který výrazně

9 Namespace `Thon.WebMining.Algorithms.Classification` v projektu `Thon.WebMining`.

ovlivní hodnotu apriorní pravděpodobnosti $P(c)$ a opakovaný výskyt stejného slova v jednom dokumentu ovlivní podmíněnou pravděpodobnost $P(t_k|c)$.

Pro naučení klasifikačního modelu je nutné vyjádřit základní pravděpodobnosti. Pravděpodobnost $P(c)$ je dána poměrem počtu dokumentů ve třídě c a počtu všech dokumentů v trénovací množině, tj.:

$$P(c)=\frac{N_c}{N} \quad ,$$

kde N_c je počet dokumentů ve třídě c a N je celkový počet dokumentů. Pravděpodobnost $P(t_k|c)$ je dána poměrem počtu výskytů výrazu t ve třídě c, včetně opakovaných výskytů stejného výrazu, a celkového počtu výrazů ve třídě c. Při vytváření modelu je nutné vyjádřit i podmíněné pravděpodobnosti pro takové kombinace výrazu a třídy, jejichž hodnota bude 0. Výskyt jedné takové hodnoty by znamenal, že i výsledná hodnota $P(c|d)$ bude mít vzhledem k součinu dílčích pravděpodobností hodnotu 0. Proto se ke každé hodnotě T_{ct} vyjadřující počet výrazů t ve třídě c připočítává hodnota 1.[10] Dílčí podmíněná pravděpodobnost je tedy dána vztahem:

$$P(t|c)=\frac{T_{ct}+1}{\sum_{t'\in V}(T_{ct'}+1)}=\frac{T_{ct}+1}{(\sum_{t'\in V}T_{ct'})+B} \quad ,$$

kde V je slovník všech výrazů v trénovací množině a $B = |V|$. Postup vytvoření a aplikování klasifikačního modelu je popsán ve formě pseudokódu algoritmem 4.1. Pro zápis pseudokódu existuje řada přístupů. Pseudokód těchto algoritmů je uveden i v [4] v kapitole 13.2, který je orientován spíše na matematické vyjádření. Pseudokód uvedený v této práci více odráží způsob skutečné implementace a využívá tedy základní řídící struktury. Implementace algoritmů je v prototypovém řešení dostupná ve třídě `NaiveBayes`.[11]

Druhou variantou modelu Naive Bayes je tzv. *Bernoulli model*, která je rovněž implementována v prototypovém řešení ve stejné třídě, jako předchozí varianta. Základním rozdílem této varianty je odlišný způsob výpočtu dílčích podmíněných pravděpodobností $P(t_k|c)$. V této variantě se nezohledňuje opakovaný výskyt stejného výrazu v jednom dokumentu. Počet výskytů jednoho výrazu v jednom dokumentu je tedy nahrazen hodnotou 1 v případě, že dokument daný výraz obsahuje, nebo hodnotou 0 v opačném případě. Postup učení a aplikování modelu se zásadním způsobem neliší od multinomial varianty. Podrobnosti k výpočtu pravděpodobností a k implementaci algoritmu je možné snadno odvodit přímo z implementace v prototypovém řešení.

Varianta Bernoulli model má za cíl řešit problém vyšší chybovosti klasifikace dlouhých dokumentů, ve kterých obvykle existuje malá množina často se opakujících slov. Dalo by se předpokládat, že v případě realitních inzerátů půjde o krátké texty, ve kterých se jednotlivá slova příliš často neopakují a tato varianta bude tedy výhodnější. Několik provedených pokusů však ukázalo, že varianta multinomial vykazuje lepší výsledky klasifikace. V tabulkách 4.1 a 4.2 jsou vidět výsledky vyhodnocení dvou pokusů. V každém pokusu je použita jiná varianta Naive Bayes modelu a v obou případech jde o klasifikaci dle kategorie nemovitosti. Z uvedených výsledků je možné usuzovat, že textový obsah realitních inzerátů není vždy možné považovat za krátký text. Z tohoto úsudku budeme vycházet i při řešení problému detekce duplicitních inzerátů v kapitole 7. Při

10 Jde o tzv. Laplace smoothing, viz [4], kapitola 13.2.

11 Namespace `Thon.WebMining.Algorithms.Classification`, projekt `Thon.WebMining`.

srovnání výsledků obou pokusů je možné si všimnout, že ačkoli celková přesnost Bernoulli modelu je jen nepatrně nižší, došlo k absolutní chybě při klasifikaci třídy maly-objekt-garaz a hodnota F-measure je pro všechny třídy nižší.

```
TrainMultinomial(trainingSet)
 FOREACH sample IN trainingSet
  classSize[sample.class] += 1
  IF (!classes.contains(sample.class))
   classes.add(sample.class)
  FOREACH t IN sample.splitIntoTerms()
   termVector[sample.class][t] += 1
   IF (!vocabulary.contains(t))
    vocabulary.add(t)
 FOREACH c IN classes
  prior[c] = classSize[c] / trainingSet.size
  sum = vocabulary.size
  FOREACH t IN termVector[c]
   sum += termVector[c][t]
  FOREACH t IN vocabulary
   condProb[c][t] = (termVector[c][t] + 1)/sum
RETURN classes, vocabulary, prior, condProb

ApplyMultinomial(sample, classes, vocabulary, prior, condProb)
 maxScore = 0
 FOREACH c IN classes
  score[c] = log(prior[c])
  FOREACH t IN sample.splitIntoTerms()
   IF (vocabulary.contains(t))
    score[c] += log(condProb[c][t])
  IF (score[c] > maxScore)
   maxScore = score[c]
   class = c
RETURN class
```

Algoritmus 4.1: Pseudokód algoritmů pro trénování a aplikování multinomial Naive Bayes modelu.

class	*byt*	*komercni-nemovitost*	*pozemek*	*dum*	*chata-chalupa*	*maly-objekt-garaz*	**precision**
byt	1381	3	0	19	0	0	**98,43%**
komercni-nemovitost	8	328	4	22	1	0	**90,36%**
pozemek	0	4	457	3	6	0	**97,23%**
dum	11	9	9	778		0	**96,29%**
chata-chalupa	1	0	5	50	90	0	**61,64%**
maly-objekt-garaz	5	4	0	9	0	1	**5,26%**
recall	**98,22%**	**94,25%**	**96,21%**	**88,31%**	**91,84%**	**100,00%**	
F-measure	**0,9833**	**0,9226**	**0,9672**	**0,9213**	**0,7377**	**0,1000**	

Správně klasifikovaných: 3035 **Chybně klasifikovaných**: 174

Celková přesnost: 94,58% **Průměrná F-measure**: 0,7720

Tabulka 4.1: Vyhodnocení klasifikace dle kategorie nemovitosti pomocí multinomial Naive Bayes modelu získané z výstupu aplikace classifier.exe.

class	byt	komercni-nemovitost	pozemek	dum	chata-chalupa	maly-objekt-garaz	precision
byt	1374	2	0	27	0	0	**97,93%**
komercni-nemovitost	0	320	7	24	1	0	**88,15%**
pozemek	19	2	467	1	0	0	**99,36%**
dum	0	3	31	755	0	0	**93,44%**
chata-chalupa	0	0	31	96	19	0	**13,01%**
maly-objekt-garaz	4	4	5	6	0	0	**0,00%**
recall	**97,65%**	**96,68%**	**86,16%**	**83,06%**	**95,00%**	**0,00%**	
F-measure	**0,9779**	**0,9222**	**0,9229**	**0,8794**	**0,2289**	**0,0000**	

Správně klasifikovaných: 2935 **Chybně klasifikovaných**: 274

Celková přesnost: 91,46% **Průměrná F-measure**: 0,6552

Tabulka 4.2: Vyhodnocení klasifikace dle kategorie nemovitosti pomocí Naive Bayes - Bernoulli modelu získané z výstupu aplikace classifier.exe.

4.4 Rocchio model

Odlišný přístup oproti statistické metodě klasifikace představuje *Rocchio model*[12] založený na výpočtu vzdáleností dokumentů ve *Vector Space Modelu*.[13] Jde o tzv. *centroidní* metodu klasifikace, při které je každá třída reprezentována vektorem obdobně jako dokumenty. Vektor reprezentující třídu dokumentů – *centroid* – je vyjádřen jako průměr normalizovaných vektorů odpovídajících dokumentům v dané třídě *c*:

$$\vec{u}(c)=\frac{1}{|D_c|}\sum_{d\in D_c}\vec{v}(d) \quad ,$$

kde D_c je množina dokumentů ve třídě *c* a vektor *v* je normalizovaný vektor dokumentu *d* definovaný jako:

$$\vec{v}(d)=\frac{\vec{V}(d)}{|\vec{V}(d)|}=\frac{\vec{V}(d)}{\sqrt{\sum_{i=1}^{M}\vec{V}_i^2(d)}} \quad ,$$

kde *M* je počet složek vektoru *V* a V_i je hodnota *i*-té složky vektoru *V*. Tato metoda je založena na předpokladu, že dokumenty patřící do stejné třídy jsou v prostoru blízko u sebe, tzn. hranici každých dvou tříd tvoří nadrovina.[14] Jde tedy o řešení tzv. lineárně separabilního problému.

Vytvoření klasifikačního modelu je jednoduše provedeno určením centroidů, resp. vektorů *u* pro všechna *c*. Určení odpovídající třídy k požadovanému dokumentu je provedeno nalezením nejbližšího centroidu. Existují dvě základní varianty určení vzdálenosti požadovaného dokumentu a centroidu. Jednou metodou je výpočet tzv. **kosinové vzdálenosti** či kosinové podobnosti (*cosine similarity*). Tato metoda se využívá

12 Podrobněji popsán v [4], kapitola 14.2.

13 Jak již bylo zmíněno v kapitole 2.3, jde o prostor, ve kterém je každý dokument reprezentován vektorem, přičemž hodnoty vektoru odpovídají hodnotám metriky Tf-idf jednotlivých výrazů ze slovníku.

14 Tou je ve dvojrozměrném prostoru přímka a v trojrozměrném prostoru rovina.

pro vyjádření podobnosti dvou dokumentů ve Vector Space Modelu, přičemž hodnota podobnosti dvou dokumentů d_1 a d_2 je dána vztahem:

$$\mathrm{sim}(d_1, d_2) = \vec{v}(d_1) \cdot \vec{v}(d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{\left|\vec{V}(d_1)\right| \cdot \left|\vec{V}(d_2)\right|} \quad .$$

Označení „kosinová vzdálenost“ vyplývá ze skutečnosti, že jeho hodnota je hodnotou kosinu úhlu daných vektorů. Hodnota podobnosti blízká hodnotě 1 tedy značí vysokou podobnost dokumentů, naopak hodnota blízká 0 značí velmi malou podobnost. Tento vztah podobnosti je založen na předpokladu, že podobné dokumenty budou obsahovat stejné výrazy s podobnou váhou a odpovídající vektory takových dokumentů budou svírat malý úhel jehož kosinus se blíží jedné. Naopak vektory odpovídající dokumentům bez jediného společného výrazu budou na sebe kolmé, tedy hodnota kosinu bude rovna 0.[15] Stejnou metodu vyjádření podobnosti jednoduše použijeme pro vyjádření podobnosti dokumentu a centroidu, přičemž nejpodobnější centroid určí třídu dokumentu. Druhou variantou pro vyjádření vzdálenosti je prosté určení **eukleidovské vzdálenosti** dokumentu a centroidu. Výsledná třída určená klasifikačním modelem dokumentu d je dána vztahem:

$$\arg\max_{c \in C} \vec{u}(c) \cdot \vec{v}(d)$$

v případě kosinové vzdálenosti, resp.:

$$\arg\min_{c \in C} \left|\vec{u}(c) - \vec{v}(d)\right|$$

v případe eukleidovské vzdálenosti. Postup vytvoření a aplikování klasifikačního modelu je popsán ve formě pseudokódu algoritmem 4.2. Při použití eukleidovské vzdálenosti není nutné provádět normalizaci vektorů. Hodnoty vzdálenosti se budou lišit, ale zůstane zachován poměr vzdáleností mezi více podobnými dokumenty a méně podobnými dokumenty. Tato varianta je výpočetně méně náročnější, než v případě použití kosinové vzdálenosti.

Implementace obou uvedených variant Rocchio modelu je v prototypovém řešení dostupná ve třídě `Rocchio`.[16] V implementaci výpočtu kosinové vzdálenosti je provedena určitá optimalizace,[17] která není zohledněna v algoritmu 4.2. Pokud požadujeme nalézt centroid s nejvyšší hodnotou kosinové vzdálenosti od daného dokumentu, je možné vzdálenost vyjádřit vztahem:

$$\mathrm{sim}'(u, d) = \frac{\vec{u}(c) \cdot \vec{V}(d)}{\left|\vec{u}(c)\right|} \quad ,$$

ve kterém je ve jmenovateli vypuštěna velikost vektoru dokumentu. Při výpočtu podobnosti jednoho dokumentu pro všechny třídy budou výsledné hodnoty podobnosti odlišné, ale bude platit:

$$\mathrm{sim}(u_1, d) < \mathrm{sim}(u_2, d) \Leftrightarrow \mathrm{sim}'(u_1, d) < \mathrm{sim}'(u_2, d) \quad .$$

Tato úprava nemá tedy žádný vliv na výsledek klasifikace, ale významným způsobem ovlivní výpočetní náročnost algoritmu.

V tabulkách 4.3 a 4.4 jsou uvedeny výsledky vyhodnocení klasifikace provedené oběma variantami Rocchio modelu. Z výsledků je patrné, že použití kosinové vzdálenosti

15 Toto je zřejmé již z výpočtu skalárního součinu vektorů v čitateli. Pokud dva dokumenty obsahují různá slova, příslušné složky vektorů budou mají vždy pro jeden dokument hodnotu 0, tedy i hodnota skalárního součinu bude rovna 0 a výsledná hodnota $\mathrm{sim}(d_1,d_2)$ taktéž.

16 Namespace `Thon.WebMining.Algorithms.Classification`, projekt `Thon.WebMining`.

17 Vychází ze vztahů podrobně popsaných v [4], kapitola 7.1.

```
TrainRocchio(trainingSet)
 FOREACH sample IN trainingSet
  classSize[sample.class] += 1
  IF (!centroids.containsKey(sample.class))
   centroids[sample.class] = EMPTY VECTOR
  FOREACH v IN sample.getDocumentVector()
   centroid[sample.class][v.term] += v.value
 FOREACH c IN classes
  FOREACH v IN centroid[c]
   v.value = v.value / classSize[c]
  length = centroid[c].computeEuclideanLength()
  FOREACH v IN centroid[c]
   v.value = v.value / length
RETURN centroids

ApplyRocchioCosine(sample, centroids)
 max = 0
 FOREACH c IN centroids.getKeys()
  sim = cosineSimilarity(centroids[c], sample)
  IF sim > max
   max = sim
   class = c
RETURN c
```

Algoritmus 4.2: Pseudokód algoritmů pro trénování a aplikování Rocchio modelu. Varianta aplikování s využitím eukleidovské vzdálenosti je obdobná s tím rozdílem, že nehledáme maximální hodnotu podobnosti, ale minimální vzdálenost.

vede k lepším výsledkům než použití eukleidovské vzdálenosti. Klasifikace s použitím eukleidovské vzdálenosti je co se týká celkové přesnosti dokonce horší než obě použité varianty Naive Bayes modelu. Zajímavé je ale porovnání průměrné hodnoty F-measure Rocchio modelu s kosinovou vzdáleností a multinomial Naive Bayes modelu. Ačkoli přesnost prvního uvedeného modelu je nepatrně nižší, průměrná hodnota F-measure je naopak výrazně vyšší. Porovnáním jednotlivých ukazatelů je možné pozorovat, že v případě Naive Bayes klasifikace dochází k velkým rozdílům v přesnosti a úplnosti pro jednotlivé třídy, zatímco u Rocchio modelu se ukazatele pro jednotlivé třídy příliš neliší. Je

class	*byt*	*komercni-nemovitost*	*pozemek*	*dum*	*chata-chalupa*	*maly-objekt-garaz*	**precision**
byt	1335	12	4	50	2	0	**95,15%**
komercni-nemovitost	6	336	4	15	1	1	**92,56%**
pozemek	0	4	448	3	15	0	**95,32%**
dum	15	25	39	707	22	0	**87,50%**
chata-chalupa	1	0	1	14	129	1	**88,36%**
maly-objekt-garaz	0	4	0	1	0	14	**73,68%**
recall	**98,38%**	**88,19%**	**90,32%**	**89,49%**	**76,33%**	**87,50%**	
F-measure	**0,9674**	**0,9032**	**0,9275**	**0,8849**	**0,8190**	**0,8000**	

Správně klasifikovaných: 2969 **Chybně klasifikovaných**: 240

Celková přesnost: 92,52% **Průměrná F-measure**: 0,8837

Tabulka 4.3: Vyhodnocení klasifikace dle kategorie nemovitosti pomocí Rocchio modelu s využitím kosinové vzdálenosti získané z výstupu aplikace classifier.exe.

class	*byt*	*komercni-nemovitost*	*pozemek*	*dum*	*chata-chalupa*	*maly-objekt-garaz*	**precision**
byt	1359	7	5	30	2	0	**96,86%**
komercni-nemovitost	15	325	7	15	0	1	**89,53%**
pozemek	1	3	457	0	9	0	**97,23%**
dum	50	23	72	651	12	0	**80,57%**
chata-chalupa	2	1	14	23	106	0	**72,60%**
maly-objekt-garaz	3	1	0	3	0	12	**63,16%**
recall	**95,03%**	**90,28%**	**82,34%**	**90,17%**	**82,17%**	**92,31%**	
F-measure	**0,9594**	**0,8990**	**0,8917**	**0,8510**	**0,7709**	**0,7500**	

Správně klasifikovaných: 2910 **Chybně klasifikovaných**: 299

Celková přesnost: 90,68% **Průměrná F-measure**: 0,8537

Tabulka 4.4: Vyhodnocení klasifikace dle kategorie nemovitosti pomocí Rocchio modelu s využitím eukleidovské vzdálenosti získané z výstupu aplikace classifier.exe.

možné tedy usuzovat, že výsledky Rocchio modelu nejsou v takové míře ovlivněny značně rozdílným velikostmi jednotlivých tříd v trénovací množině.

4.5 K-Nearest Neighbor

Poslední metodou klasifikace použitou v této práci je metoda K-Nearest Neighbor. Stejně jako v případě Rocchio modelu využívá i tato metoda reprezentaci dokumentů ve Vector Space Modelu. Na rozdíl od Rocchio modelu řeší nelineárně separovatelný problém. Prostor dokumentů není v tomto případě dělen přímkami, které by představovali hranice tříd, ale je dělen na buňky, přičemž každá buňka je určena jedním vektorem a je tvořena všemi body, které jsou k tomuto vektoru blíže, než k jakémukoliv jinému vektoru. Množina několika buněk tedy může vymezit třídu dokumentů, přičemž takové buňky nemusí tvořit souvislý prostor.

Učení klasifikačního modelu představuje pouze jakési předzpracování trénovacích dat, kdy ke každému dokumentu z trénovací množiny určíme jeho normalizovaný vektor $\vec{v}(d)$ a přiřadíme k němu příslušnou třídu. Výsledkem je množina

$$D' = \{\vec{v}(d) : d \in D\} \quad ,$$

kde D je trénovací množina a C je množina všech tříd, a zobrazení

$$f : D' \rightarrow C \quad ,$$

které prvku z množiny D' přiřazuje odpovídající třídu. Aplikování klasifikačního modelu je provedeno nalezením množiny S_k, která je tvořena K nejbližšími dokumenty z množiny D' pomocí kosinové vzdálenosti. Pro každou třídu c vyjádříme míru jejího zastoupení v množině S_k jako:

$$p_c = |S_c| \quad \text{, kde} \quad S_c = \{d : d \in S_k \wedge f(d) = c\} \quad .$$

Výslednou třídou je třída, která je nejvíce zastoupena v množině S_k, tj.

$$\arg\max_{c \in C} p_c \quad .$$

```
TrainKNearestNeighbor(trainingSet)
  FOREACH sample IN trainingSet
    classes[sample] = sample.class
    vector = sample.getDocumentVector()
    length = vector.computeEuclideanLength()
    FOREACH v IN vector
      v = v / length
    docs.add(vector)
RETURN docs, classes

ApplyKNearestNeighbor(sample, docs, classes, k)
  queue = CREATE PriorityQueue
  v = sample.getDocumentVector()
  FOREACH d IN docs
    sim = cosineSimilarity(sample, d)
    IF (sim > queue.last.sim)
      IF (queue.size > k)
        queue.removeLast()
      entry = CREATE entry
      entry.sim = sim
      entry.class = classes[d]
      queue.add(entry)
  FOR i FROM 1 TO k
    e = queue.pop()
    p[e.class] += 1
  max = 0
  FOREACH c IN classes
    IF p[c] > max
      max = p[c]
      class = c
RETURN class
```

Algoritmus 4.3: Pseudokód algoritmů pro trénování a aplikování modelu K-Nearest Neighbor.

Pro dosažení uspokojivých výsledků je nutné zvolit hodnotu parametru K. Neexistuje přesný způsob, jak tuto hodnotu zvolit. Obvykle se volí liché číslo, přičemž se přihlíží k velikosti trénovací množiny. Vyšší hodnota může zajistit vyšší přesnost modelu, protože dojde k jemnějšímu rozdělení prostoru. Zároveň ale mírně roste náročnost výpočtu hodnot p_c, protože je nutné zpracovat větší množinu S_k, a mírně roste množství porovnání nutných k nalezení většího množství nejbližších dokumentů. Vyhledání nejbližších K dokumentů je však možné velmi efektivně implementovat pomocí prioritní fronty.

Postup vytvoření a aplikování klasifikačního modelu je popsán ve formě pseudokódu algoritmem 4.3. Při implementaci algoritmu je možné provést stejnou optimalizaci výpočtu kosinové vzdálenosti, která byla uvedena v kapitole 4.4. V prototypovém řešení je algoritmus implementován ve třídě `KNearestNeighbor`.[18]

Při několika provedených pokusech se stejnou trénovací množinou s tím, že hodnota K byla postupně volena 51, 71 a 101 bylo postupně dosaženo vyšší přesnosti klasifikace. Při volbě hodnoty 151 byla naopak přesnost klasifikace nižší, než v předchozích případech. V tabulce 4.5 je uveden nejlepší dosažený výsledek, kdy hodnota K byla 101.

Z výsledků je patrné, že při malém zastoupení některé ze tříd ve trénovací množině dochází k podobnému efektu, jako v případě Naive Bayes klasifikace, a sice, že daná třída není při aplikování modelu vůbec použita. Dále je patrné, klasifikace touto metodou poskytuje horší výsledky, než v případě použití Rocchio modelu. Z toho je možné usuzovat, že problém klasifikace realitních inzerátů představuje lineárně separabilní

18 Namespace `Thon.WebMining.Algorithms.Classification`, projekt `Thon.WebMining`.

class	*byt*	*komercni-nemovitost*	*pozemek*	*dum*	*chata-chalupa*	*maly-objekt-garaz*	**precision**
byt	1387	4	0	11	1	0	**98,86%**
komercni-nemovitost	29	307	2	25	0	0	**84,57%**
pozemek	34	4	412	12	8	0	**87,66%**
dum	126	9	21	644		0	**79,70%**
chata-chalupa	4	2	4	32	104	0	**71,23%**
maly-objekt-garaz	16	1	0	2	0	0	**0,00%**
recall	**86,90%**	**93,88%**	**93,85%**	**88,71%**	**85,95%**	**0,00%**	
F-measure	**0,9250**	**0,8899**	**0,9065**	**0,8396**	**0,7790**	**0,0000**	

Správně klasifikovaných: 2854 **Chybně klasifikovaných**: 355

Celková přesnost: 88,94% **Průměrná F-measure**: 0,7233

Tabulka 4.5: Vyhodnocení klasifikace dle kategorie nemovitosti pomocí metody K-Nearest Neighbor při volbě K=101 získané z výstupu aplikace classifier.exe.

problém. Při porovnání celkové přesnosti a průměrné hodnoty F-measure je zřejmé, že pro klasifikaci realitních inzerátů není tato metoda vhodná. Z uvedených modelů vykazuje nejnižší hodnotu celkové přesnosti a průměrná hodnota F-measure je druhá nejnižší po Bernoulli modelu.

4.6 Srovnání modelů

Předmětem práce není provedení a přesné zdokumentování experimentů. Použití vybraných algoritmů má za cíl spíše vytvoření představy o vhodnosti jejich použití v oblasti realitní inzerce. Proto byly v průběhu testování prototypového řešení provedeny jednoduché pokusy s cílem poskytnutí určitého srovnání vhodnosti jednotlivých metod klasifikace. Provedené pokusy potvrzují, že z pohledu výkonnosti je nejvhodnější metodou Naive Bayes, a to jak ve fázi učení, tak ve fázi aplikování modelu. Naopak nejméně vhodnou metodou je K-Nearest Neighbor, u které je výpočetně náročná fáze aplikování modelu. Oba uvedené modely jsou však velmi citlivé na předzpracování dat a na strukturu trénovací množiny. Uvedenými problémy naopak tolik netrpí Rocchio model, u kterého nebyla přesnost klasifikace méně zastoupených tříd v trénovací množině výrazně nižší, než přesnost ostatních tříd. Rocchio model v případě použití kosinové vzdálenosti je ale výpočetně mírně náročnější ve fázi učení oproti Naive Bayes modelu. Použití kosinové vzdálenosti u Rocchio modelu však poskytuje vyšší přesnost klasifikace.

Vzhledem k výsledkům provedených pokusů je pro účely této práce nejvhodnější metodou klasifikace Rocchio model s využitím kosinové vzdálenosti. Menší hodnoty přesnosti některých tříd jsou pravděpodobně způsobeny výraznými rozdíly v zastoupení jednotlivých tříd. Pokud by bylo provedeno preciznější předzpracování dokumentů v trénovací množině a bylo by možné co nejpřesněji dodržet poměr zastoupení jednotlivých tříd, byla by rovněž vhodná metoda multinomial Naive Bayes. Článek [6] uvádí způsob předzpracování dat tak, aby bylo možné použít Naive Bayes model pro kolekce dat s nevyváženým zastoupením tříd.

5 Shluková analýza

Další základní úlohou data miningu je shluková analýza (*clustering*), jejíž cílem je rozdělit množinu vzorů, resp. dokumentů, na podmnožiny – tzv. *shluky* (*clusters*). Vlastností každého shluku je, že dokumenty v něm obsažené jsou si navzájem podobné a zároveň se liší od dokumentů v ostatních shlucích. Oproti problému klasifikace není u této metody zpracování dat tak zřejmá interpretace výsledků. Při klasifikaci předem víme, čím jsou si dokumenty v jedné třídě podobné. Každá třída má tedy předem určené charakteristické vlastnosti dokumentů v ní obsažených. Při shlukové analýze předem nevíme, čím si budou dokumenty v jednotlivých shlucích podobné. Navíc volba použitých metod pro porovnání podobnosti dokumentů může vést k velmi odlišným výsledkům.

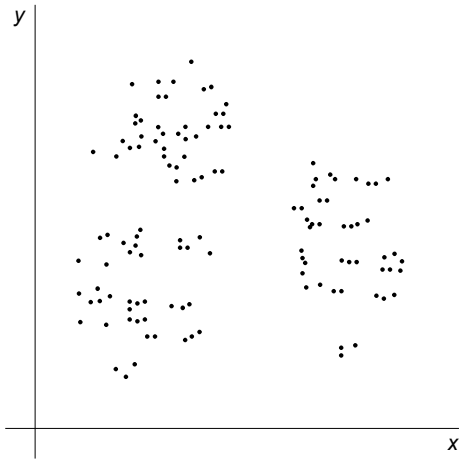
Shluková analýza je metodou strojového učení bez učitele, při které nemáme k dispozici trénovací množinu dat opatřených správnými výsledky. Učení typicky probíhá na již předzpracované databázi, kterou je v našem případě databáze více než 30 000 realitních inzerátů.[1] Před samotným provedením shlukové analýzy je nutné zvolit způsob určení vzdálenosti či podobnosti dokumentů. Při zpracování strukturovaných dat, kdy je možné jednotlivé vzory vyjádřit jako body ve vícerozměrném eukleidovském prostoru, je nejčastěji jako míra podobnosti vzorů použita eukleidovská vzdálenost. Existují ale i další způsoby, např. *Čebyševova vzdálenost* nebo *Manhattanská vzdálenost*. Pro porovnání binárních hodnot je možné využít *Hammingovu vzdálenost*, pro porovnání slov *Levenshteinovu vzdálenost*. Pro porovnání nestrukturovaných textových dokumentů, v našem případě realitních inzerátů, můžeme s výhodou využít *kosinovou vzdálenost* popsanou v kapitole 4.4.

Existují dva základní přístupy k tvorbě shluků. Jedním z nich je vyhledávání shluků v celé množině dat najednou. Tento způsob je označován jako *flat clustering*, jelikož shluky tvoří pouze jednu úroveň. Typickým představitelem algoritmu pro flat clustering je K-Means algoritmus, který je blíže popsán v kapitole 5.2. Při tomto přístupu k tvorbě shluků je ale nutné předem určit či odhadnout požadovaný počet shluků. Příklad množiny vzorů, u které jsou patrné 3 shluky dat, je uveden na obrázku 5.1. Druhým přístupem je tzv. *hierarchické shlukování*, kdy shluky samotné tvoří určitou víceúrovňovou strukturu. Na nejvyšší úrovni je jeden shluk tvořený všemi vzory, tento shluk je rozdělen na několik menších shluků. V každé další úrovni jsou opět shluky rozděleny na menší a v poslední úrovni hierarchie jsou shluky tvořené jednotlivými vzory. V tomto případě není nutné na začátku procesu určit požadovaný počet shluků. Ze vzniklé struktury je možné získat množinu vzájemně disjunktních shluků pomocí tzv. řezu.[2] Hierarchická struktura shluků se obvykle znázorňuje pomocí tzv. *dendrogramu*. Jeho příklad včetně možného řezu je znázorněn na obrázku 5.2.

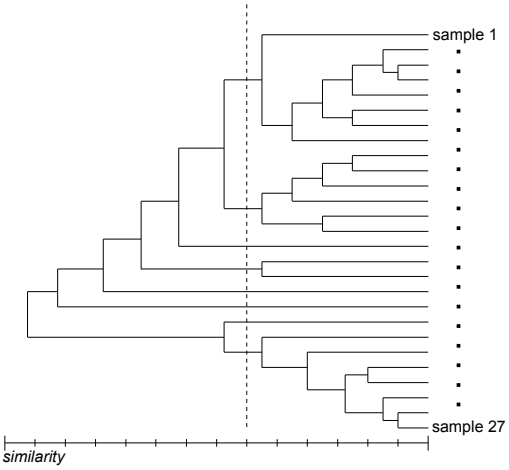
V případě hierarchického shlukování navíc rozlišujeme dva postupy vyhledávání shluků: *aglomerativní* shlukování a *top-down* shlukování. Při aglomerativním shlukování postupujeme od jednotlivých vzorů postupným slučováním dvojic vzorů či již vytvořených shluků. Při top-down shlukování naopak začínáme s celou množinou dat rozdělením na

1 Databáze je dostupná na přiloženém DVD, viz kapitola 1.3.

2 Některé způsoby nalezení vhodného řezu jsou popsány v [4], kapitola 17.1.



Obrázek 5.1: Příklad množiny dat, ve které jsou patrné 3 shluky.



Obrázek 5.2: Dendrogram znázorňující množinu 27 vzorů. Čárkovaně je znázorněn řez, který dělí strukturu na 8 shluků.

dva shluky, které dále dělíme. Postup je opakován tak dlouho, dokud není každý vzor v samostatném shluku. Metoda top-down shlukování má výhodu v tom, že není nutné proces tvorby shluků dokončit, ale je možné je zastavit v okamžiku, kdy velikost nebo počet shluků splňuje stanovené požadavky. Nevýhodou je naopak nutnost využití algoritmu pro provedení flat clusteringu v každé úrovni dělení shluků.

Jak již bylo naznačeno, výsledky shlukové analýzy není snadné interpretovat, přesto poskytuje široké možnosti využití. Jako statistická metoda se využívá ke zpracování dat v oblasti marketingu, přírodních věd či zdravotnictví. Příkladem aplikace shlukové analýzy v oblasti Information Retrieval je zpracování celé kolekce dokumentů, ve kterých může uživatel vyhledávat. Vytvořené shluky jsou následně označeny klíčovými slovy, které daný shluk nejlépe charakterizují. Proces označování shluků se nazývá *Cluster labeling* a je podrobně popsán v [4], kapitola 17.7. Tato klíčová slova mohou být následně nabídnuta uživateli při vyhledávání. Uživatel v takovém případě nemusí sám klíčová slova zadávat, ale pouze vybírá z nabídky dostupných výrazů. Odlišným způsobem využití je shlukování výsledků vyhledávání v IR systému. Uživatel tak získá nejen seznam výsledků odpovídajících zadanému dotazu, ale má navíc k dispozici informaci o tom, které výsledky jsou si navzájem podobné.

V rámci této práce je shluková analýza využita pro k efektivnímu vyhledávání podobných inzerátů. Kromě shlukové analýzy existuje několik dalších způsobů vyhledávání podobných inzerátů, které budou blíže popsány v kapitole 6. Shluková analýza může však toto vyhledávání výrazným způsobem zefektivnit. Řešeným problémem je také způsob efektivního provedení shlukové analýzy nad velkou množinou dat. Za tím účelem budou v této kapitole popsány dva základní algoritmy pro provedení shlukové analýzy, přičemž využití výsledků bude podrobněji popsáno v kapitole 6.

Proces shlukové analýzy je v prototypovém řešení implementován v samostatné aplikaci `clustering.exe`. Aplikace využívá kombinaci metod shlukové analýzy popsaných v této kapitole. Výstupem aplikace je soubor obsahující údaje o vytvořených

shlucích, přičemž každý shluk je tvořen jeho středem – *centroidem*[3] – a seznamem identifikátorů inzerátů, které patří do daného shluku. Centroid je vyjádřen ve formě normalizovaného tvaru vektoru ve Vector Space Modelu. Definici všech shluků reprezentuje instance třídy `ClusteringResult`,[4] která je do výstupního souboru aplikace binárně serializována.

5.1 Příprava dat

Podobně jako v případě klasifikace inzerátů je i před provedením shlukové analýzy nutné provést určité předzpracování dat. V tomto případě je ale metoda aplikována na celou množinu dat, kterou je v našem případě databáze všech získaných realitních inzerátů. Jedná se celkem o 30 469 inzerátů získaných z realitních portálů Avízo.cz, Bazoš.cz, ibyty.com a Reality iDNES.cz.[5]

Vzhledem k tomu, že se struktura inzerátů mezi uvedenými portály značně liší, byla pro provedení shlukové analýzy zvolena pouze ta část obsahu inzerátů, která představuje textový obsah. Ten je totiž dostupný ve všech inzerátech ze všech realitních portálů. Velmi často textový obsah inzerátu obsahuje i důležité informace popisující nemovitost. Ve strukturovaných údajích inzerátů bývají často uvedeny spíše číselné hodnoty, popř. přesnější specifikace lokality. Podobnost inzerátů je tedy posuzována na základě podobnosti jejich textových obsahů. Jak bude patrné z prototypového řešení IR systému, tato jednoduchá úvaha pro porovnání podobnosti inzerátů poskytuje překvapivě dobré výsledky.

Předpokladem pro provedení shlukové analýzy je získání inzerátů z portálů pomocí crawleru a jejich uložení v provozní databázi, jak bylo popsáno v kapitole 3. Následně je provedeno indexování jejich textového obsahu, jak bylo popsáno v kapitole 2.4. Z indexu je pro každý inzerát využit tzv. *Term Frequency Vector*, který umožňuje velmi snadné vyjádření dokumentu pomoci vektoru ve Vector Space Modelu, jak bylo popsáno v kapitole. V prototypovém řešení je vstupní dokument reprezentován instancí třídy `Sample`,[6] která zapouzdřuje jednoznačný identifikátor dokumentu v provozní databázi, vektor dokumentu a normalizovaný tvar vektoru. Předzpracování inzerátů je tedy velmi podobné jako v případě klasifikace popsané v kapitole 4.1.

5.2 K-Means

K-Means je základním algoritmem shlukové analýzy, jehož výsledkem je rozdělení množiny vzorů do předem určeného počtu shluků. Cílem algoritmu je nalezení takových středů shluků, aby průměrná kvadratická vzdálenost všech vzorů od středů jim přiřazených shluků byla minimální. Střed shluku *c* je definován jako:

$$\vec{u}(c)=\frac{1}{|c|}\sum_{\vec{x}\in c}\vec{x} \quad ,$$

3 Význam centroidu v stejný, jako v případě klasifikační metody Rocchio popsané v kapitole 4.4.
4 Namespace `Thon.WebMining.Algorithms.Clustering`, projekt `Thon.WebMining`.
5 Adresy portálů jsou uvedeny v kapitole 3.1.
6 Namespace `Thon.WebMining.Algorithms.Clustering`, projekt `Thon.WebMining`.

kde *c* je zároveň množinou dokumentů zařazených do shluku a vektor *x* představuje vektor dokumentu. Modifikací tohoto algoritmu je algoritmus K-Medoids, při kterém se namísto výpočtu středů shluků vybírá jeden ze vzorů, jehož největší vzdálenost od ostatních vzorů je v rámci shluku minimální. Nalezení takového středu je ale výrazně výpočetně náročnější.

Algoritmus je možné popsat pěti základními kroky:

1. Určení počtu shluků.
2. Určení počátečních středů shluků. Je možné určit náhodné vektory ze stejného prostoru, jejich složky nepřesahují rozsah minimálních a maximálních hodnot u jednotlivých vzorů. Častěji se používá metoda výběru náhodných vzorů z dané množiny, přičemž počet vzorů odpovídá počtu shluků. Vybrané vzory jsou považovány za počáteční středy shluků.
3. Přiřazení všech vzorů ke shlukům dle použitého způsobu určení vzdálenosti. V prototypovém řešení je pro určení vzdálenosti mezi vzorem a středem shluku použita kosinová vzdálenost, bylo by ale možné použít např. eukleidovskou vzdálenost.
4. Výpočet nových středů shluků jako průměr všech vzorů přiřazených do příslušného shluku.
5. Opakování kroků 3 a 4 až do doby splnění podmínky pro ukončení algoritmu.

Existuje několik způsobů stanovení podmínky pro ukončení algoritmu. Nejjednodušší variantou je dosažení maximálního počtu iterací. Určení příliš malého počtu iterací může ale velmi negativně ovlivnit kvalitu výsledku. Často používaným způsobem je provedení kontroly, zda došlo ke změně hodnot centroidů mezi dvěma posledními iteracemi, popř. zda se tato hodnota změnila pouze nepatrně. Tento způsob je poměrně snadno implementovatelný a poskytuje spolehlivý způsob rozpoznání stavu, kdy jsou středy shluků ustálené. Oba dva uvedené způsoby jsou implementované v prototypovém řešení. Dalšími používanými metodami může být kontrola přeřazení některého ze vzorů k jinému shluku oproti předchozí iteraci, popř. může jít o metody založené na porovnání celkové kvadratické odchylky všech dokumentů od přiřazených středů.[7] Postup provedení flat clusteringu je popsán ve formě pseudokódu algoritmem 5.1. Implementace algoritmu v prototypovém řešení je dostupná ve třídě `KMeans`, výsledné shluky jsou reprezentovány instancemi třídy `Cluster`.[8]

Je patrné, že algoritmus je poměrně snadno implementovatelný, zejména pokud je možné pracovat se všemi vzory načtenými v paměti. Nicméně algoritmus je možné modifikovat i pro potřeby paralelního a distribuovaného zpracování velmi velkého objemu dat. Příkladem může být článek [7] představující řešení K-Means algoritmu pomocí *MapReduce*. V prototypovém řešení je použito paralelní zpracování při výpočtu nových středů shluků a při přiřazování vzorů ke shlukům. Paralelní zpracování je řešeno pomocí *paralelní foreach smyčky* implementované v Task Parallel Library.[9] Výhodou algoritmu je možnost přírůstkového zpracování při změně množiny dat, která byla již zmíněna v kapitole 2.6. Středy dříve vytvořených shluků je možné použít jako počáteční shluky. Je možné předpokládat, že pokud dojde ke změně malého množství vzorů, bude nutné při spuštění procesu provést pouze jednu nebo několik iterací. Je ale nutné zachovat původní počet shluků. Nevýhodou algoritmu je poměrně vysoká výpočetní náročnost při velkém

7 Podrobnosti ke způsobu výpočtu odchylky jsou uvedeny v [4], kapitola 16.4.

8 Namespace `Thon.WebMining.Algorithms.Clustering`, projekt `Thon.WebMining`.

9 Konkrétně jde o statickou metodu `ForEach` ve třídě `Parallel`.

```
KMeans(samples, K, maxIterations, deltaTreshold)
 FOR i FROM 1 TO K
  clusters[i].centroid = getRandomSample(samples)
 iteration = 0
 maxDelta = 0
 DO
  divideSamplesIntoClusters(samples, clusters)
  FOREACH c IN clusters
   delta = c.calculateNewCentroid()
   c.samples.clear()
   IF delta > maxDelta
    maxDelta = delta
  iteration += 1
 WHILE iteration < maxIterations AND maxDelta > deltaTreshold
 divideSamplesIntoClusters(dataSet, clusters)
RETURN clusters

DivideSamplesIntoClusters(samples, clusters)
 FOREACH s IN samples
  minDist = ∞
  FOREACH c IN cluster
   dist = cosineSimilarity(s, c.centroid)
   IF dist < minDist
    minDist = dist
    nearestCluster = c
  nearestCluster.samples.add(s)
END
```

Algoritmus 5.1: K-Means algoritmus s možností určení maximálního počtu iterací a minimální změny středů shluků pro jeho ukončení.

počtu shluků. Jak je patrné z části pseudokódu `DivideSamplesIntoClusters`, počet provedení výpočtu vzdálenosti všech vzorů je násoben počtem shluků.

5.3 Top-Down clustering

V prototypovém řešení bude potřeba provedení shlukové analýzy pro poměrně velký počet shluků. V takovém případě je, jak již bylo zmíněno, použití algoritmu K-Means neefektivní. Pro optimalizaci procesu shlukové analýzy nad celou databází inzerátů je tedy vhodné použít metodu hierarchického shlukování. Metoda Top-Down clustering je zvolena právě proto, že nepotřebujeme sestavit celou hierarchii shluků, pouze potřebujeme množinu inzerátů rozdělit do předem známého počtu shluků.

Jde o rekurzivní metodu dělení shluků na menší shluky, přičemž dělení pokračuje tak dlouho, dokud není dosaženo stanovené podmínky. Podmínkou pro ukončení algoritmu může být dosažení dané minimální míry podobnosti vzorů v každém shluku nebo prosté vytvoření požadovaného počtu shluků. Právě druhý z uvedených způsobů je použit v prototypovém řešení.

Proces začíná vytvořením jednoho shluku obsahujícího všechny vzory. Tento shluk rozdělíme na menší počet shluků pomocí algoritmu K-Means, přičemž hodnota K je nějaká malá konstanta. Její konkrétní hodnota může ovlivnit výsledek shlukové analýzy a neexistuje jednoznačný způsob pro její určení. Během několika pokusů provedených při tvorbě prototypového řešení byla jako nejvhodnější zvolena hodnota 5. Bylo by ale možné použít i hodnotu 2. Na každý z vytvořených shluků aplikujeme rekurzivně stejný proces

```
TopDown(samples, maxNumOfClusters, K, maxIterations, deltaTreshold)
  baseCluster = CREATE Cluster
  FOREACH s IN samples
    baseCluster.samples.add(s)
  clusters.add(baseCluster)
  WHILE clusters.length + K - 1 < maxNumOfClusters
    cluster = clusters.removeLast()
    clusters.addAll(KMeans(cluster.samples, K,  maxIterations, deltaTreshold))
    sortClustersBySize(clusters)
  FOREACH c IN clusters
    c.calculateNewCentroid()
RETURN clusters
```

Algoritmus 5.2: Top-Down clustering s použitím K-Means algoritmu pro dělení shluku na menší shluky.

dělení. V okamžiku dosažení stanovené podmínky algoritmus ukončíme a jako výsledek použijeme shluky vytvořené v poslední úrovni dělení.

Postup provedení Top-Down clusteringu je ve formě pseudokódu popsán algoritmem 5.2, ve kterém jsou zohledněny i některé implementační detaily použité v prototypovém řešení. Specifickým způsobem je upraveno dělení shluků, které není prováděno v každé iteraci rekurzivně pro všechny shluky, ale vždy je dělen shluk s nejvyšším počtem vzorů. Na konci každé iterace je tedy množina již vytvořených shluků seřazena vzestupně dle velikosti a K-Means algoritmus je vždy aplikován na největší shluk. Cílem tohoto přístupu je vytvořit pokud možno rovnoměrné rozdělení vzorů do shluků. Implementace algoritmu v prototypovém řešení je dostupná ve třídě `TopDown`.[10]

V ukázce 5.1 je vidět část výsledku provedené shlukové analýzy. Bylo vybráno několik inzerátů ze dvou různých shluků tak, aby byla patrná podobnost inzerátů ve stejném shluku a zároveň odlišnost inzerátů v různých shlucích. Je zřejmé, že všechny uvedené dokumenty patřící do shluku číslo 3 se týkají pronájmu rodinného domu nebo bytu. Dalo by se říci, že podobnost inzerátu týkající se rodinného domu je nižší než podobnost dvou inzerátů týkajících se pronájmu bytu. V každém případě jsou si inzeráty výrazně podobnější než inzeráty ze shluku číslo 160. V tomto shluku je podobnost všech inzerátů zjevná z výrazu „pozemek“, ačkoli v posledním inzerátu jde o prodej domu s pozemkem, v předchozích dvou jde o prodej pozemku samotného. Právě tohoto efektu podobnosti inzerátu je využito při vyhledávání podobných inzerátů, které je podrobněji řešeno v kapitole 6.

10 Namespace `Thon.WebMining.Algorithms.Clustering`, projekt `Thon.WebMining`.

Shluk č. 3 (inzeráty č. 8883, 14698, 20203, ...)

Zdroj: http://reality.avizo.cz/pronajem-rodinny-dum-500-m2-9547896.html *Ze dne*: 8. 12. 2013 *Nadpis*: Pronájem, Rodinný dům, 500 m2 *Text*: Kód zakázky: NM-44, Nabízíme pronájem v dobře dostupném místě s občanskou …
Zdroj: http://reality.idnes.cz/detail/pronajem/byt/4+1/praha-mala-strana-zborovska/7064707 *Ze dne*: 14. 12. 2013 *Nadpis*: Pronájem bytu 4+1, 103 m, Zborovská ul., Praha 5 - Malá Strana *Text*: Dumrealit. cz Vám zprostředkuje pronájem bytu 4+1, 103m , který je ve 3. patře …
Zdroj: http://www.ibyty.com/main.php?choice=11&ID_INZERATU=4560&NABPOP=nabidka *Ze dne*: 20. 7. 2014 *Nadpis*: pronajmeme dvoupokojový byt v blízkosti Výstaviště *Text*: Pronajmeme částečně zařízený byt v třetím poschodí s výtahem.

Shluk č. 160 (inzeráty č. 9480, 17541, 28383, ...)

Zdroj: http://reality.avizo.cz/prodej-stavebni-pozemek-frydek-mistek-8752529.html *Ze dne*: 8. 12. 2013 *Nadpis*: Prodej, stavební pozemek, Frýdek - Místek *Text*: Kód zakázky: 293639. Nabízíme k prodeji stavební parcelu, která se nachází …
Zdroj: http://reality.idnes.cz/detail/prodej/pozemek/stavebni-parcela/frenstat-pod-radhostem/6693975 *Ze dne*: 14. 12. 2013 *Nadpis*: Frenštát p. R - 850 m², stavební poz. s výhledem na Beskydy *Text*: Prodáme krásný stavební pozemek 850 m2 ve Frenštátě pod Radhoštěm v lokalitě …
Zdroj: http://reality.bazos.cz/inzerat/37010209/Prodej-mobilniho-domu-s-pozemkem-300-m2-v-Liberci.php *Ze dne*: 21. 7. 2014 *Nadpis*: Prodej mobilního domu s pozemkem 300 m2 v Liberci. *Text*: RK nevolat.Prodáme mobilní dům 3+1 s pozemkem v Liberci, Doubí. Celoročně …

Ukázka 5.1: Výsledek shlukové analýzy – ukázka vybraných dokumentů ze dvou shluků. Označení shluků a inzerátů je získáno z výstupu aplikace clustering.exe, obsah inzerátů je získán z provozní databáze, která je dostupná na přiloženém DVD.

6 Vyhledávání podobných inzerátů

Předchozí dvě kapitoly byly primárně zaměřeny na tvorbu modelů pro dolování dat, přičemž byly pouze naznačeny možnosti využití případných výsledků. Pokrývaly tedy převážně dvě fáze procesu dolování dat: **transformaci** a **dolování**. Tato kapitola je zaměřena na fázi **interpretace** a na získání výsledků, které jsou přímo využitelné pro uživatele. Jde o splnění jednoho z cílů práce, a sice vyhledání podobných inzerátů k zadanému inzerátu. Jak bylo uvedeno v úvodu, jde o obdobu zobrazení podobného zboží k výrobku nalezeném v e-shopu.

Možnost vyhledávání podobných inzerátů je přímo využitelná pro uživatele IR systému, neboť nabízí jednoduchý a efektivní způsob procházení databáze inzerátů. Uživatel může provést první vyhledání inzerátů dle velmi jednoduchého fulltextového dotazu, např. zadáním části názvu obce. Ve výsledcích vyhledávání zvolí inzerát, který odpovídá požadavkům na typ nemovitosti a zobrazí se mu podrobnosti inzerátu. Pokud následně uživatel zjistí, že inzerát přesně neodpovídá jeho původní představě, může jednoduše zvolit z nabídky několika podobných inzerátů, které mu IR systém automaticky nabídne. Uživatel vybere jeden z nich a opět se mu zobrazí podrobnosti zvoleného inzerátu včetně nabídky dalších podobných inzerátů. Takovým způsobem je možné procházet poměrně rozsáhlou databázi bez nutnosti zadávání přesných vyhledávacích dotazů a kritérií. Takové procházení není náhodné, ale je svým způsobem systematické.

Popsaný způsob procházení podobných inzerátů má svá omezení, která vycházejí zejména z významu podobnosti dokumentů. Jeden uživatel může považovat dva vybrané inzeráty za podobné, zatímco pro jiného uživatele to tak být nemusí. Pro vyjádření míry podobnosti v prototypovém řešení je využito kosinové vzdálenosti, která se ukázala být vhodnou již při řešení problému klasifikace a pro provedení shlukové analýzy.[1] Připomeňme, že pro výpočet míry podobnosti je nutné vyjádřit inzerát ve formě vektoru ve *Vector Space Modelu*.[2] Jak již bylo zmíněno, jako vhodnou datovou strukturou pro uložení údajů potřebných ke snadnému získání vektoru dokumentu je možné využít fulltextový index. Podobnost dvou inzerátů je vyjádřena hodnotou skalárního součinu normalizovaných vektorů odpovídajících jednotlivým inzerátům. Je zřejmé, že čím více mají inzeráty společných výrazů s vysokou váhou, tím více jsou si podobné.[3]

V následujících podkapitolách je představeno několik metod využitelných pro nalezení zadaného počtu nejpodobnějších inzerátu k jednomu zadanému. Nejjednodušší metodou z hlediska implementace je porovnání podobnosti zadaného inzerátu ke všem ostatním z celé databáze. Pro implementaci vyhledávání je samozřejmě možné využít připravený index, metoda je tedy pro účely této práce označena jako *Index traversal*. [4] popisuje v kapitole 7 metody využitelné pro efektivní vyhodnocování dotazů a řazení výsledků ve vyhledávacím systému. V této práci jsou dvě z nich využity pro efektivní vyhledávání podobných inzerátů. Jednou z nich je metoda *Index elimination*, jejímž cílem je omezení množiny inzerátů v indexu tak, aby byl výpočet míry podobnosti prováděn pouze ty inzeráty, u

1 Viz kapitola 4.4 a úvod kapitoly 5.
2 Vysvětleno v kapitole 2.3.
3 Váhou výrazu je hodnota Tf-idf popsána v kapitole 2.3.

kterých je možné předpokládat, že jejich podobnost bude vyšší. Poslední použitou metodou je metoda *Cluster pruning*, jejímž cílem je, jako u předchozí metody, omezení množiny dokumentů, u kterých vyhodnocujeme míru podobnosti. Pro omezení množiny je u této metody využito výsledků shlukové analýzy popsané v předchozí kapitole.

Implementace všech popsaných metod pro vyhledání podobných inzerátů je v prototypovém řešení dostupná v samostatné aplikaci `similarity.exe`. Aplikace provede na základě identifikátoru zadaného dokumentu a zvolené metody vyhledání zadaného počtu inzerátů s nejvyšší mírou podobnosti. Výstupem aplikace je seznam identifikátorů nalezených inzerátů s uvedením míry podobnosti. Na základě vypsaných identifikátorů je možné vyhledat inzeráty v databázi pomocí aplikace `docdb.exe`.

6.1 Příprava dat

Předzpracování dat pro potřeby vyhledávání podobných inzerátů je v podstatě stejné, jako v případě klasifikace či shlukové analýzy. Předpokládáme, že všechny inzeráty získané z realitních portálů jsou indexovány pomocí aplikace `search.exe`. Tato aplikace umožňuje provést zpracování textového obsahu inzerátů získaného z provozní databáze a vytvoření fulltextového indexu. Pro samotné vyjádření podobnosti inzerátů je opět použit pouze textový obsah inzerátů, stejně jako při provádění shlukové analýzy.

Aplikaci `search.exe` je možné rovněž využít pro prvotní vyhledání inzerátů. Tato konzolová aplikace poskytuje velmi jednoduché rozhraní pro provedení fulltextového dotazu. Výstupem aplikace je výpis 10 inzerátů odpovídajících zadanému dotazu. U každého inzerátu je vypsán jeho identifikátor, nadpis, cena a část textového obsahu. Identifikátor je možné následně využít pro vyhledání podobných inzerátů.

V případě použití metody Cluster pruning je nutné jako jeden ze vstupů aplikace `similarity.exe` určit soubor s výsledky shlukové analýzy získaný pomocí aplikace `clustering.exe`. Připomeňme, že výsledkem shlukové analýzy je množina středů shluků a seznam identifikátorů inzerátů náležejících do jednotlivých shluků. Střed každého shluku je reprezentován vektorem ve Vector Space Modelu a je možné jej využít k vyjádření podobnosti libovolného inzerátu a shluku.

6.2 Index traversal

Triviální metodou vyhledání podobných inzerátů je porovnání podobnosti zadaného inzerátu se všemi ostatními inzeráty v databázi. Jelikož pro vyjádření inzerátů v podobě vektorů je výhodné použít index, je možné index využít i pro průchod celou databází. V prototypovém řešení je pro tvorbu indexu využita knihovna Apache Lucene.NET, která poskytuje nízkoúrovňové API pro práci s indexem. Základním nástrojem pro čtení obsahu indexu je abstraktní třída `IndexReader`. Pomocí metody `TermDocs` je možné získat iterátor umožňující průchod všemi indexovanými inzeráty, který ke každému inzerátu poskytne jeho identifikátor.[4] Na základě identifikátoru je možné pomocí metody `GetTermFreqVector` získat informaci o všech výrazech obsažených v příslušném inzerátu a o počtu výskytů každého výrazu. Pomocí metody `DocFreq` je možné získat počet

4 Nejde o identifikátor inzerátu v provozní databázi, ale o identifikaci v indexu, která může být při každé změně indexu jiná.

Inzerát č. 1395

Zdroj: http://reality.idnes.cz/detail/prodej/byt/3+kk/praha-vinohrady-korunni/6005137
Ze dne: 15. 9. 2013
Nadpis: Novostavba Rezidence Korunní, 3+kk/S/G, 120 m², Vinohrady
Text: Novostavba v Praze 10 – Vinohradech. Nový byt 3+kk o ploše 120 m2, s terasou …

Nalezené podobné inzeráty:

1440 (0,9718)
Zdroj: http://reality.idnes.cz/detail/prodej/byt/2+kk/praha-vinohrady-korunni/6005147
Ze dne: 15. 9. 2013
Nadpis: Novostavba Rezidence Korunní, 2+kk/S/G, 66 m², Vinohrady
Text: Novostavba v Praze 10 – Vinohradech. Nový byt 2+kk o ploše 66 m2, s balkonem 5.3 …

1441 (0,9674)
Zdroj: http://reality.idnes.cz/detail/prodej/byt/3+kk/praha-vinohrady-korunni/6005146
Ze dne: 15. 9. 2013
Nadpis: Novostavba Rezidence Korunní, 3+kk/S/G, 120 m², Vinohrady
Text: Novostavba v Praze 10 – Vinohradech. Nový byt 3+kk o ploše 120 m2, s balkonem …

1397 (0,9655)
Zdroj: http://reality.idnes.cz/detail/prodej/byt/4+kk/praha-vinohrady-korunni/6005114
Ze dne: 15. 9. 2013
Nadpis: Novostavba Rezidence Korunní, 4+kk/S/G, 144.3 m², Vinohrady
Text: Novostavba v Praze 10 – Vinohradech. Nový byt 4+kk o ploše 144.3 m2, s balkonem …

Ukázka 6.1: Výsledek vyhledání podobných inzerátů k inzerátu č. 1395 metodou Index traversal. Označení inzerátů a hodnoty míry podobnosti uvedené v závorkách jsou získány z výstupu aplikace similarity.exe.

dokumentů v indexu obsahujících zadaný výraz. Uvedené metody postačují k průchodu všech inzerátů uložených v indexu, k vytvoření vektorů reprezentujících jednotlivé inzeráty a k výpočtu míry podobnosti se zadaným inzerátem. Z vytvořených vektorů je možné vypočítat hodnotu kosinové vzdálenosti k zadanému inzerátu a vybrat z databáze požadovaný počet nejpodobnějších inzerátů. V ukázce 6.1 je uveden výsledek takového vyhledávání. Je vidět, že nalezené inzeráty mají vysokou míru podobnosti vzhledem k zadanému inzerátu. Podobnost je zřejmá i z krátkých ukázek textů inzerátů.

Výhodou této metody je velmi snadná implementace, přičemž není potřeba žádné další zpracování inzerátů, kromě jejich zpracování při indexování. Zároveň tato metoda umožňuje spolehlivé nalezení zadaného počtu nejpodobnějších inzerátů k jednomu zadanému, jelikož při ní dojde k porovnání míry podobnosti se všemi ostatními. Tento přístup je však velmi neefektivní při velkém množství dokumentů. V rámci prováděných pokusů při tvorbě prototypového řešení se doba vyhledání 10 nejpodobnějších inzerátů v indexu obsahujícím přes 30 tis. inzerátů pohybovala v řádu jednotek vteřin. Cílem práce nebylo provedení přesných experimentů a měření, nicméně je zřejmé, že vyhledávání v řádu jednotek vteřin není prakticky využitelné např. pro online zpracování v IR systému dostupného většímu počtu uživatelů současně.

Implementace této metody vyhledávání podobných inzerátů je v prototypovém řešení dostupná ve třídě `IndexTraversal`.[5] Implementace tvorby vektoru reprezentujícího jednotlivý inzerát na základě údajů uložených v indexu je dostupná ve třídě `Tfidf`.[6]

5 Namespace `Thon.WebMining.SimilaritySearch`, projekt `Thon.WebMining`.

6 Namespace `Thon.WebMining.Utils`, projekt `Thon.WebMining`.

6.3 Index elimination

Metoda *Index elimination* je založena na předpokladu, že míra podobnosti dvou inzerátů je nenulová v případě, že mají alespoň jeden společný výraz. Pro vyhledání podobných inzerátů je tedy možné namísto procházení celého indexu porovnat pouze výsledky vyhledávacího dotazu tvořeného všemi výrazy ze zadaného inzerátu spojenými operátorem OR. Je zřejmé, že výsledkem takového dotazu nebudou inzeráty, které nemají se zadaným inzerátem společný žádný výraz.

Výhodou tohoto způsobu vyhledávání je získání shodného výsledku jako v případě procházení celé databáze. Při výpočtu míry podobnosti jsou totiž vynechány pouze inzeráty, u kterých by byla výsledná hodnota nulová. Rovněž implementace je relativně jednoduchá a je k ní potřeba využít stejných nástrojů jako při použití metody Index traversal. Nevýhodou tohoto způsobu vyhledávání je možnost vzniku velmi dlouhého vyhledávacího dotazu. V případě inzerátu uvedeného v ukázce 6.1 (strana 45) by došlo k vytvoření dotazu obsahujícího 134 výrazů, tolik výrazů totiž obsahuje celý text inzerátu. Vyhodnocení takového vyhledávacího dotazu může být časově náročné. V případě realitní inzerce navíc dochází k situaci, kdy výsledkem takového dotazu je pouze o něco málo méně inzerátů, než je jejich celkový počet v databázi. To je zřejmě dáno omezeným množstvím výrazů používaných v inzerátech.

V kapitole 7.1.2 v [4] jsou uvedeny dvě možné heuristiky, jejichž cílem je ještě výraznější omezení množiny inzerátů tak, aby výpočet míry podobnosti byl proveden pouze pro inzeráty, u kterých je možné předpokládat vysokou míru podobnosti se zadaným inzerátem. Jedním způsobem je vytvoření vyhledávacího dotazu pouze s využitím výrazů, které mají vyšší hodnotu *idf*.[7] Je totiž možné předpokládat, že takové výrazy zajistí vyšší míru podobnosti. Může ale dojít k eliminaci výrazů, které mají sice nízkou hodnotu idf, ale v porovnávaných inzerátech mají vysokou hodnotu tf. Takové výrazy by mohli zajistit vysokou míru podobnosti, ale nebudou při vyhledávání uplatněny. Naproti tomu dojde k velmi výraznému urychlení vyhodnocení vyhledávacího dotazu, neboť pro výrazy s vysokou hodnotou idf jsou v indexu výrazně kratší posting listy. Neexistuje ale jednoznačný způsob určení minimální hranice idf pro sestavení vyhledávacího dotazu. Je možné ji určit zkusmo, popř. je možné použít předem daný počet výrazů ze zadaného inzerátu s nejvyššími hodnotami idf.

Druhou možností je vytvoření takového vyhledávacího dotazu, který zajistí vyhledání inzerátů obsahujících více než jeden (nebo dokonce všechny) z výrazů ze zadaného inzerátu. Při použití této metody však může dojít k vynechání inzerátů, u kterých by byla výsledná podobnost vyšší, a to jen proto, že obsahují pouze jeden společný výraz. Obě uvedené heuristiky mohou ovlivnit výsledek vyhledávání podobných inzerátů takovým způsobem, že dojde k vynechání inzerátů s vyšší mírou podobnosti a naopak zůstanou ve výsledku ponechány inzeráty, jejichž podobnost je nižší.

V prototypovém řešení je implementováno vyhledávání metodou Index elimination s využitím první popsané heuristiky, tedy možnosti určení minimální hodnoty idf. Implementace je dostupná ve třídě `IndexElimination`.[8] Při použití tohoto způsobu omezení množiny porovnávaných inzerátů roste velmi výrazně rychlost vyhodnocení výsledku s vyšší hodnotou minimální úrovně idf. Nepřímo úměrně k tomu však klesá přesnost výsledku. V tabulce 6.1 je uvedeno srovnání výsledku vyhledání deseti

7 Viz kapitola 2.3.

8 Namespace `Thon.WebMining.SimilaritySearch`, projekt `Thon.WebMining`.

Podobné inzeráty k inzerátu č. 2591

idf	0	1,5	3
1	2592 (0,3552)	2592 (0,3552)	2592 (0,3552)
2	7364 (0,2725)	7364 (0,2725)	7364 (0,2725)
3	1440 (0,2372)	1440 (0,2372)	17329 (0,1749)
4	29659 (0,2367)	29659 (0,2367)	26218 (0,1602)
5	1395 (0,2367)	1395 (0,2367)	11148 (0,1422)
9	1396 (0,2346)	1396 (0,2346)	19076 (0,1284)
7	1397 (0,2343)	1397 (0,2343)	11095 (0,1206)
8	1441 (0,2342)	1441 (0,2342)	7676 (0,1141)
9	5407 (0,2301)	5407 (0,2301)	13048 (0,1103)
10	12293 (0,2156)	12293 (0,2156)	6548 (0,1091)
Doba	4,9608s	0,8424s	0,1092s

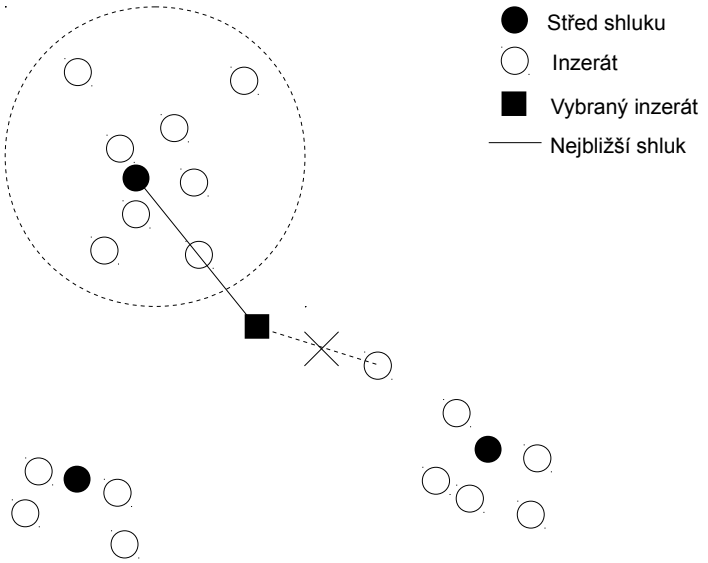
Tabulka 6.1: Srovnání výsledků vyhledání podobných inzerátů metodou Index elimination s různou minimální hodnotou idf. Označení inzerátů a hodnoty míry podobnosti uvedené v závorkách jsou získány z výstupu aplikace similarity.exe.

nejpodobnějších inzerátů ke stejnému inzerátu s nastavením různé minimální hodnoty idf. Pokud je minimální hodnota idf nastavena hodnotu 0, výsledek je shodný s výsledkem vyhledání pomocí metody Index traversal. Pro srovnání časové náročnosti je u každého způsobu uvedena doba potřebná pro získání výsledku. Mezi jednotlivými výsledky je zřejmý i rozdíl v přesnosti v závislosti na zvolené hodnotě idf. Zatímco při hodnotě 1,5 je výsledek shodný, při hodnotě 3 se výsledek výrazně liší a došlo k vynechání inzerátů s vyšší mírou podobnosti. Nižší přesnost výsledku je tedy vyvážena dobou potřebnou pro jeho zpracování.

6.4 Cluster pruning

Cílem metody *Cluster pruning* je, stejně jako u metody Index elimination, omezení množiny inzerátů, u kterých probíhá vyhodnocení podobnosti. Omezení množiny je provedeno na základě výsledků shlukové analýzy. Výsledkem shlukové analýzy jsou shluky tvořené vzájemně si podobnými inzeráty. Předpokládejme, že pokud omezíme množinu všech inzerátů na množinu tvořenou několika vhodně zvolenými shluky, nalezneme v této množině požadované množství nejpodobnějších inzerátů. Volba vhodných shluků je provedena porovnáním míry podobnosti zadaného inzerátu se středy shluků.

Počet shluků je vhodné zvolit jako $\sqrt{N}$, kde N je počet všech inzerátů v databázi. Při rovnoměrném rozdělení inzerátů do shluků bude každý shluk obsahovat $N/\sqrt{N}=\sqrt{N}$ inzerátů. V praxi však dochází k poměrně značným rozdílům ve velikosti jednotlivých shluků. Efektivní metodou pro vytvoření shluků je použití Top-down clusteringu. Jak bylo uvedeno v předchozí kapitole, proces dělení shluků je možné ukončit po dosažení požadovaného počtu shluků. Tato metoda je rovněž použita v prototypovém řešení, přičemž pro dělení shluků je použit K-Means algoritmus s počtem shluků 5, maximálním



Obrázek 6.1: Volba shluků pro vyhledávání podobných inzerátů metodou Cluster pruning.

počtem iterací 10 a hodnotou 0,00001 jako minimální hodnotu, o kterou se musí změnit vzdálenost středu, aby algoritmus pokračoval další iterací.

Vyhledání podobných inzerátů je provedeno ve dvou krocích:

1. Vyhledáme určitý počet shluků, jejichž středy mají nejvyšší kosinovou vzdálenost se zadaným inzerátem (v prototypovém řešení je voleno 10 shluků).
2. Provedeme výpočet kosinové vzdálenosti pro všechny inzeráty ve zvolených shlucích a vybereme požadovaný počet inzerátů s nejvyšší kosinovou vzdáleností k zadanému inzerátu.

Podobně jako v případě použití metody Index elimination může i metoda Cluster pruning poskytnout odlišné výsledky vyhledávání, zejména pokud je zvolen malý počet shluků pro porovnání. V databázi může existovat inzerát s vyšší mírou podobnosti, který ale nebyl ve zvoleném shluku a není tedy zahrnut v celkovém výsledku. Tato situace je znázorněna na obrázku 6.1, kde je vidět, že při volbě jednoho nejbližšího shluku by nebyl do výsledku zahrnut inzerát, který je přiřazen ke shluku v pravé dolní části obrázku. To ale nemusí znamenat jednoznačně horší výsledek vyhledávání, záleží totiž na interpretaci výsledků a na významu podobnosti inzerátů. Byly totiž vybrány nejpodobnější inzeráty z několika shluků, které obsahují sobě navzájem podobné inzeráty a všechny shluky byly podobné se zadaným inzerátem. Z určitého úhlu pohledu by tedy mohlo jít naopak o lepší výsledek než při prostém porovnání podobnosti se všemi inzeráty v databázi. V tabulce 6.2 je uvedeno porovnání výsledků vyhledávání podobných inzerátů různými metodami. Je vidět, že v určitých případech může metoda Cluster pruning poskytnout přesnější výsledky v čase, který je výrazně kratší než při použití metody Index traversal.

Metoda Cluster pruning představuje poměrně efektivní metodu vyhledávání podobných inzerátů v rozsáhlé databázi. Zároveň poskytuje poměrně přesné výsledky, úspěšnost nalezení nejbližších inzerátů je téměř srovnatelná s metodou Index traversal. Nevýhodou je nutnost provedení poměrně náročného předzpracování inzerátů pomocí shlukové analýzy, a tedy i složitější implementace. Vyhledání podobných inzerátů je časově náročnější, než v případě použití metody Index elimination s vyšší hranicí minimální hodnoty idf. Doba vyhledávání roste s odmocninou počtu inzerátů v databázi.

Podobné inzeráty k inzerátu č. 1395

Metoda	Index traversal	Index elimination (idf > 3)	Cluster pruning
1	1440 (0,9718)	1440 (0,9718)	1440 (0,9718)
2	1441 (0,9674)	1441 (0,9674)	1441 (0,9674)
3	1397 (0,9655)	1397 (0,9655)	1397 (0,9655)
4	1396 (0,9646)	1396 (0,9646)	1396 (0,9646)
5	1477 (0,7996)	1477 (0,7996)	1477 (0,7996)
9	1438 (0,7705)	1438 (0,7705)	1438 (0,7705)
7	12293 (0,3774)	18758 (0,1385)	12293 (0,3774)
8	5407 (0,3437)	15755 (0,1160)	5407 (0,3437)
9	29659 (0,2872)	5538 (0,07606)	15627 (0,2118)
10	2591 (0,2367)	18130 (0,0735)	17324 (0,1878)
Doba	6.0996s	0.1404s	1.4352s

Tabulka 6.2: Srovnání výsledků vyhledání podobných inzerátů různými metodami. Označení inzerátů a hodnoty míry podobnosti uvedené v závorkách jsou získány z výstupu aplikace similarity.exe.

Vyhodnocení výsledků vyhledávání by bylo možné dále optimalizovat paralelním zpracováním zvolených shluků, kdy z každého shluku je nezávisle na ostatních vybrán požadovaný počet nejpodobnějších inzerátů a následně jsou tyto dílčí výsledky sloučeny.

Implementace této metody je v prototypovém řešení dostupná ve třídě `ClusterPruning`.[9] Konstruktor třídy vyžaduje jako vstupní parametr výsledek provedené shlukové analýzy. Ten je v aplikaci `similarity.exe` získán deserializací souboru získaného výstupem aplikace `clustering.exe`. Dále umožňuje tato třída volbu počtu shluků použitých pro vyhledávání.

6.5 Vyhodnocení výsledků

V předchozích podkapitolách byly představeny metody, které umožňují efektivně prohledat databázi inzerátů za účelem nalezení podobných inzerátů k jednomu zadanému. V naprosté většině případů je potřeba vybrat pouze předem daný počet inzerátů (např. 10, jak bylo uvedeno v ukázkách v této kapitole) s nejvyšší mírou podobnosti. Možným řešením by bylo uložení výsledků vyhodnocení podobnosti všech porovnávaných inzerátů v paměti. Následně by byly tyto výsledky seřazeny podle míry podobnosti a bylo by použito pouze prvních 10 výsledků. Tento způsob by byl samozřejmě velmi neefektivní v případě rozsáhlé databáze inzerátů.

Efektivním způsobem výběru *n* nejpodobnějších inzerátů může být použití prioritní fronty, která umožňuje nastavení maximálního počtu prvků. V prototypovém řešení je prioritní fronta implementovaná s použitím abstraktní implementace v knihovně Apache Lucene.NET (konkrétně jde o třídu `Lucene.Net.Util.PriorityQueue`). Proces vyhodnocení výsledků vyhledávání je v prototypovém řešení implementován ve třídě `SimSearcher`, která poskytuje kolekci *n* nejpodobnějších inzerátů. Jeden výsledek je

9 Namespace `Thon.WebMining.SimilaritySearch`, projekt `Thon.WebMining`.

reprezentován instancí třídy `SimDocument`,[10] která zapouzdřuje obsah inzerátu vyhledaného v indexu a hodnotu kosinové vzdálenosti. Třída `SimSearcher` je závislá na zvolené metodě vyhledávání, která je definovaná rozhraním `IMethod`. Toto rozhraní implementují třídy uvedené v předchozích kapitolách.

Jak již bylo uvedeno v úvodu kapitoly, výsledky vyhledávání podobných inzerátů je možné přímo prezentovat uživateli. Společně s nalezenými podobnými inzeráty je vhodné i nějakým způsobem prezentovat míru podobnosti. V prototypovém řešení je uživateli prezentována přímo hodnota kosinové vzdálenosti, přičemž do výsledku nejsou zahrnuty inzeráty, jejichž podobnost je nižší než 0,2. Na několika příkladech bylo zjištěno, že inzeráty z nižší hodnotou kosinové vzdálenosti jsou již příliš odlišné od zadaného inzerátu. Zobrazení číselné hodnoty však nemusí být pro běžné uživatele srozumitelné. Pro přehlednější zobrazení by bylo možné hodnotu kosinové vzdálenosti využít např. ke zvýraznění inzerátů, u nichž byla hodnota blíže k jedné.

10 Namespace `Thon.WebMining.SimilaritySearch`, projekt `Thon.WebMining`.

7 Detekce duplicitních inzerátů

Problém detekce duplicitních inzerátů je do jisté míry specifickým problémem, který svou povahou patří spíše do oblasti předzpracování dat, než do oblasti dolování dat. Dalo by se říci, že jde o rozpoznání shodných vzorů v datech, která jsou předmětem dalšího zpracování. Rozpoznáním totožných inzerátů by bylo možné získat podrobnější informace o nemovitosti. Lze totiž předpokládat, že různé výskyty téhož inzerátu obsahují odlišné podrobnosti zveřejňované různými realitními portály, přičemž text inzerátů je stejný. V případě této práce je detekce duplicit využita při prezentaci výsledků vyhledávání v prototypovém IR systému, tedy ve fázi **interpretace**. Cílem práce je návrh řešení zajišťující automatické rozpoznání totožných inzerátů, které jsou zveřejněny na různých URL adresách téhož realitního portálu, nebo jsou zveřejněny na různých realitních portálech. Ve výsledcích vyhledání v prototypovém řešení jsou pak totožné inzeráty zobrazeny jako jeden výsledek.

Důvody vícenásobného publikování stejného inzerátu mohou být různé. Může jít o opakované publikování inzerátu z důvodu ukončení doby jeho zveřejnění. V IR systému přitom může být po nějakou přechodnou dobu i původní verze inzerátu uložena. Mohlo by se také jednat o záměrné uvádění jednoho inzerátů vícekrát ve výsledcích vyhledávání na realitním portálu s odlišnými URL adresami. Tím se údajně některé realitní portály snaží zvýšit pravděpodobnost vyhledání v internetových vyhledávačích. Takové zveřejňování inzerátů by mohlo být dokonce záměrem ze strany realitních portálů pro znesnadnění automatizovaného zpracování zveřejněného obsahu. Takové automatizované zpracování je právě předmětem této práce.

Jiným případem je zveřejňování stejného inzerátu jedním inzerentem na různých realitních portálech. Zřejmým důvodem publikování stejného inzerátu je v tomto případě snaha o zvýšení pravděpodobnosti jeho nalezení případnými zájemci. S tím souvisí hlavní problém detekce duplicit. Obsah inzerátu nemusí být při jeho opakovaném zveřejnění vždy zcela stejný. Text inzerátu může např. obsahovat číslo zakázky, které se pochopitelně pokaždé liší. Inzeráty také vznikají kopírováním textu uživatelem při jeho publikování. Při kopírování může uživatel provést drobné úpravy textu, např. opravy překlepů. Na různých portálech tak bude stejný inzerát zveřejněný s drobnými rozdíly v textu. Kromě toho mohou mít různé realitní portály odlišné požadavky na formu inzerátu, velmi často se liší sada údajů o nemovitosti zadávaných ve strukturované formě. Uživatel tedy musí v některých případech obsah inzerátu při zveřejňování na různých portálech mírně přizpůsobit.

Předpokladem detekce totožných inzerátů je skutečnost, že alespoň významnou část inzerátu uživatel ponechá stejnou. Na druhou stranu, pokud by dva inzeráty týkající se stejné nemovitosti neměly významnou část obsahu stejnou, nemůžeme předpokládat, že jde o totožné inzeráty. Dva dokumenty, které jsou téměř totožné a liší se pouze v nepatrné míře, jsou často označovány jako *near duplicates*. Zmíněný předpoklad je také základním rozdílem mezi problémem detekce totožných inzerátů a problémem vyhledávání podobných inzerátů popsanému v předchozí kapitole. Smyslem není nalézt inzeráty, které mají cosi společného a určit míru podobnosti. Požadováno je jasné tvrzení, zda jsou dva

zadané inzeráty totožné či nikoliv. Snažíme se přitom minimalizovat chybu falešně pozitivních výsledků. Chybné tvrzení, že jsou dva inzeráty totožné, by bylo významnější chybou než nenalezení totožných inzerátů z důvodu jejich určité odlišnosti.

Příkladem mohou být inzeráty uvedené v ukázce 6.1 (strana 45). Na první pohled jde o velmi podobné inzeráty. Pomocí aplikace `similarity.exe` je možné zjistit, že míra jejich vzájemné podobnosti se pohybuje mezi hodnotami 0,95 a 0,97. Při pohledu na text inzerátu je zjevné, že text vznikl kopírováním. Nemůže však jít o totožné inzeráty, neboť každý popisuje jinou nemovitost. Konkrétně jde o různé byty ve stejném bytovém domě, nicméně s velmi podobným popisem, ve stejné lokalitě a se stejným popisem některého vybavení. Již z nadpisu je ale zřejmé, že se jedná o různě velké byty se značně odlišnou cenou. Označení takových inzerátů za totožné by bylo velmi závažnou chybou, která by mohla významným způsobem zkreslit informace prezentované uživateli.

Řešení popsaného problému vyžaduje odlišný přístup, než vyhledávání podobných inzerátů. Jak bylo uvedeno v ukázce 6.1 (strana 45), ani velmi vysoká míra podobnosti nezajistí spolehlivé rozpoznání totožných inzerátů. Pro detekci duplicitních inzerátů provedenou na základě významné shody v textu inzerátu je použit a implementován algoritmus *SimHash*. Řešení vychází z publikací [3] a [8]. Kromě samotné implementace algoritmu je v prototypovém řešení zohledněn i problém časté aktualizace databáze inzerátů popsaný v kapitole 2.6. Dále je řešen problém efektivního vyhledávání duplicitních inzerátů v rozsáhlé databázi tak, aby bylo možné toto řešení využít v rámci online zpracování vyhledávacích dotazů v prototypovém IR systému.

7.1 Příprava dat

Předzpracování dat je pro potřeby detekce duplicit snazší, než v případech klasifikace či shlukové analýzy. Jak již bylo naznačeno v úvodu, cílem je nalezení totožných inzerátů, tedy inzerátů se stejným nebo nepatrně odlišným textem. V tomto případě tedy není žádoucí odstranění nevýznamných slov (tzv. stop slov) či převedení ostatních slov do určitého normalizovaného tvaru. Celý text inzerátu je ponechán ve své původní formě, pouze je rozdělen na jednotlivá slova. K tomu postačí využít provozní databázi inzerátu získanou pomocí aplikace `crawler.exe`. Z inzerátu uloženého v provozní databázi v částečně strukturované formě[1] využijeme pole s názvem `Text`. Rozdělení textu na jednotlivá slova zajišťuje v prototypovém řešení třída `StandardTokenSource`,[2] která využívá implementaci `StandardTokenizer` z knihovny Apache Lucene.NET.

7.2 SimHash

Algoritmus SimHash je často využíván k detekci duplicitních obsahů webových stránek internetovými vyhledávači. V některých případech je využíván i k vyhledávání textů s určitou mírou podobnosti. Algoritmus je založen na předpokladu, že je možné každý dokument vyjádřit ve formě otisku, který je výstupem jednosměrné funkce. Jde o analogii s běžně používanými hešovacími funkcemi, pomocí kterých jsou vstupní data reprezentována relativně malou číselnou hodnotou. Základní vlastností běžné hešovací funkce je skutečnost, že nepatrná změna vstupu způsobí výraznou změnu výsledné

1 Částečně strukturovanou formou myslíme datovou strukturu dokumentu popsanou v kapitole 3.2.

2 Namespace `Thon.WebMining.Utils.TokenSource`, projekt `Thon.WebMining`.

hodnoty. Naproti tomu základní vlastností algoritmu SimHash je fakt, že pro různá vstupní data lišící se pouze nepatrně jsou výstupem hodnoty s nulovou nebo velmi malou *hammingovou vzdáleností*. Naopak vyšší odlišnost vstupních dat vede ke vzniku hodnot s vysokou hammingovou vzdáleností. Hammingova vzdálenost udává počet bitů, ve kterých se dvě hodnoty SimHash liší.

Pro vytvoření hodnoty SimHash pro zadaný vstupní text je nutné nejprve text rozdělit na slova a každému slovu přiřadit váhu odpovídající jeho významu v textu. V obecné rovině může jde o vyjádření množiny vlastnosti[3] popisující vstupní entitu. V případě textového vstupu považujeme každé jednotlivé slovo za jednu takovou vlastnost. Určení váhy každého slova je nepovinnou součástí algoritmu. Při tvorbě práce se ukázalo být jako postačující řešení, kdy všem slovům je přiřazena váha 1. Namísto této konstanty by bylo možné použít váhu Tf-idf.[4] Následně pro každé slovo vypočteme hash pomocí běžné hešovací funkce. V tomto případě je nutné použít hešovací funkci, která splňuje základní podmínku výrazné odlišnosti hodnoty hešovací funkce při nepatrné změně vstupního slova. Výsledná hodnota SimHash je následně sestavena kombinací všech dílčích hash hodnot s ohledem na váhu odpovídajících slov následovně:

1. Inicializujeme pomocný vektor jehož velikost odpovídá požadované velikosti hodnoty SimHash. V našem případě volíme 64 bitovou hodnotu. Vyšší hodnota umožňuje přesnější rozpoznání duplicit, ale zvyšuje výpočetní náročnost algoritmu. Velikost 64 bitů je dle použitých zdrojů vhodnou volbou.
2. Pro každou hash hodnotu všech slov upravíme pomocný vektor tak, že i-tou složku vektoru zvýšíme o hodnotu váhy slova v případě, že hodnota *i*-tého bitu hash hodnoty daného slova je rovna jedné. V případě, že je *i*-tý bit hash hodnoty daného slova roven nule, i-tou složku pomocného vektoru snížíme o hodnotu váhy slova.
3. Výslednou hodnotu SimHash sestavíme tak, že její *i*-tý bit nastavíme na hodnotu 1 v případě, že *i*-tá složka pomocného vektoru je větší než 0, v opačném případě nastavíme *i*-tý bit na hodnotu 0.

Celý postup je popsán ve formě pseudokódu algoritmem 7.1. Pro výpočet hodnoty hešovací funkce jednoho slova je nutné použít takovou hešovací funkci, jejíž výstupem je hodnota stejné velikosti jako požadovaná hodnota SimHash. Běžná implementace metody `GetHashCode` v prostředí Microsoft .NET Framework poskytuje 32 bitovou hodnotu. V rámci práce byla vytvořena jednoduchá implementace hešovací funkce, která pro zadaný

```
SimHash(text)
  vector = NEW float[64]
  FOREACH token IN SplitTextIntoTokens(text)
    hash = ComputeHashcode(token)
    FOR i FROM 0 TO 63
      IF hash[i] == 1
        vector[i] += GetWeight(token)
      ELSE
        vector[i] -= GetWeight(token)
  simHash = 0
  FOR i FROM 0 TO 63
    IF vector[i] > 0
      simhash[i] = 1
RETURN simhash
```

Algoritmus 7.1: Výpočet hodnoty SimHash pro zadaný vstupní text.

3 Neboli *features* – viz. [3], kapitola I.
4 Viz kapitola 2.3.

řetězec vytvoří 64 bitovou hodnotu. Implementace algoritmu SimHash a použité hešovací funkce je dostupná ve třídě `SimHash`.[5]

Z popsaného algoritmu je zřejmé, že výsledná hodnota SimHash je ovlivněna dílčími hodnotami hešovacích funkcí pro jednotlivá slova. Nezáleží však na pořadí jednotlivých slov v textu. Pořadí slov ale může mít poměrně významný vliv na vyhodnocení totožnosti dvou textu. Naopak v případě zjištění podobnosti dvou textu bylo žádoucí, aby na pořadí slov nezáleželo. Za tím účelem se často využívá metoda zvaná *shingling*. Při této metodě nedochází k rozdělení vstupního textu na jednotlivá slova, ale na tzv. *shingles*, což jsou *n*-tice slov, které se navzájem překrývají. Např. z textu:

```
Nabízím k prodeji byt v Českých Budějovicích
```

je možné vytvořit trojice:

```
Nabízím k prodeji
k prodeji byt
prodeji byt v
byt v Českých
v Českých Budějovicích
```

Tímto způsobem je možné zajistit zohlednění pořadí slov v textu při tvorbě hodnoty SimHash, neboť dílčí hodnoty hešovací funkce jsou získány z vytvořených trojic slov. Např. prohození dvou slov by neznamenalo žádnou změnu hodnoty SimHash, pokud by byl text rozdělen po slovech. Při použití shingles velikosti 3 způsobí prohození dvou slov změnu třech trojic.

Pokud bychom vytvářeli hodnotu SimHash pro dokument o 10 slovech bez použití shingles, znamenala by změna jednoho slova změnu 1/10 dílčích hodnot hešovací funkce. Pokud zvolíme shingles velikosti 3, je text o 10 slovech rozdělen na 8 trojic. Změna jednoho slova způsobí změnu třech trojic, tedy 3/8 dílčích hodnot hešovací funkce. Je tedy zřejmé, že velikost shingles má významný vliv na tvorbu hodnoty SimHash, ovšem neexistuje přesný způsob jejího určení. Volba je závislá na provedení určitých testů pro konkrétní řešený problém. Vyšší hodnota zajistí snížení chyby falešně pozitivních výsledků, zároveň ale může dojít ke snížení celkového počtu detekovaných duplicit. Při tvorbě prototypového řešení se ukázalo jako vhodné použití shingles velikosti 3. Implementace této metody je v prototypovém řešení dostupná ve třídě `ShingleTokenSource`.[6]

Ke zjištění, zda jsou dva dokumenty totožné nebo téměř totožné stačí vypočíst jejich SimHash hodnoty a zjistit jejich hammingovu vzdálenost. Dokumenty považujeme za duplicitní v případě, že je hammingova vzdálenost nepřesahuje určenou hranici. Neexistuje přesný způsob určení maximální hammingovy vzdálenosti, opět záleží na konkrétním řešeném problému a hodnotu je nutné určit na základě experimentů. V pokusech provedených při tvorbě prototypového řešení se ukázalo jako vhodné použití hodnoty 3. V ukázce 7.1 jsou vidět dva inzeráty, u nichž je hammingova vzdálenost příslušných hodnot SimHash 4. Je přitom zřejmé, že jde o různé inzeráty pravděpodobně vytvořené kopírováním stejného textu použitého pro nabídku různých bytových jednotek ve stejném domě. Jde o poměrně častý jev, kdy jeden inzerent nabízí na jednom realitním portálu více

5 Namespace `Thon.WebMining.Algorithms`, projekt `Thon.WebMining`.

6 Namespace `Thon.WebMining.Utils.TokenSource`, projekt `Thon.WebMining`.

Různé inzeráty:

15549
Zdroj: http://reality.idnes.cz/detail/prodej/byt/2+kk/praha-liben-svetova/7057856
Ze dne: 14. 12. 2013
Nadpis: Byt 2+kk (66 m²) 5 minut chůze od stanice metra
Cena: 3 260 012 Kč
Text: Dovolujeme si představit bytovou jednotku o dispozici 2+kk, jejíž obytná plocha je 66 m2. Nachází se v kompletně zrekonstruovaném novorenesančním domě ve Světově ulici. Ten leží jen několik kroků od tramvajové zastávky …

15553
Zdroj: http://reality.idnes.cz/detail/prodej/byt/2+kk/praha-liben-svetova/6661707
Ze dne: 14. 12. 2013
Nadpis: Byt 2+kk (100 m²) v klidné ulici, jen 250 metrů od metra
Cena: 5 412 573 Kč
Text: Dovolujeme si představit bytovou jednotku o dispozici 2+kk, jejíž obytná plocha je 100 m2. Nachází se v kompletně zrekonstruovaném novorenesančním domě ve Světově ulici. Ten leží jen několik kroků od tramvajové zastávky …

Ukázka 7.1: Velmi podobné inzeráty, které není možné považovat za duplicitní. Hammingova vzdálenost jejich SimHash hodnot je 4.

Totožné inzeráty:

829
Zdroj: http://reality.avizo.cz/pronajem-garaze-ostrava-nova-ves-8424744.html
Ze dne: 15. 9. 2013
Nadpis: Pronájem garáže Ostrava - Nová Ves
Cena: 699 Kč
Text: Pronajmu řadovou zděnou garáž o velikosti 18m2 v lokalitě Ostrava Nová Ves ul. Bartošova. Garáž se nachází v centru bezpečné a klidné lokality, která je v blízkosti zástavby rodinných domů. Velmi dobrá dopravní a strategická dostupnost. …

21416
Zdroj: http://reality.bazos.cz/inzerat/37263553/Pronajem-garaze-Ostrava-Nova-Ves-Bartosova.php
Ze dne: 21. 7. 2014
Nadpis: Pronájem garáže - Ostrava Nová Ves, Bartošova
Cena: 699 Kč
Text: Pronajmu řadovou zděnou garáž o velikosti 18m2 v lokalitě Ostrava Nová Ves ul. Bartošova. Garáž se nachází v centru bezpečné a klidné lokality, která je v blízkosti zástavby rodinných domů. Velmi dobrá dopravní a strategická dostupnost. …

Ukázka 7.2: Totožné inzeráty zveřejněné na různých realitních portálech. Hammingova vzdálenost jejich SimHash hodnot je 0.

nemovitostí. Podobný případ byl uveden i v ukázce 6.1 (strana 45). Odlišný případ je uveden v ukázce 7.2. V tomto případě je zřejmé, že jde o totožný inzerát zveřejněný na různých realitních portálech. Inzeráty mají nepatrně odlišný nadpis, nicméně hodnota SimHash vytvořená z jejich textového obsahu je shodná.

7.3 Efektivní vyhledávání

Algoritmus SimHash umožňuje výpočet otisku každého inzerátu nezávisle na ostatních. Je tedy možné již při ukládání nových inzerátů do databáze společně s nimi ukládat i odpovídající otisk. Pokud bychom v takové databázi potřebovali vyhledat všechny duplicitní inzeráty k jednomu zadanému inzerátu, bylo by nutné projít všechny otisky ostatních inzerátů, pro každý z nich zjistit hammingovu vzdálenost mezi ním a otiskem zadaného dokumentu a do výsledku vyhledávání zařadit pouze ty záznamy, u

kterých byla zjištěná hammingova vzdálenost nižší než určená hranice. Takový postup je ovšem velmi neefektivní v případě vyhledávání duplicit v rozsáhlé databázi.

Množinu porovnávaných otisků při vyhledávání duplicit v databázi je možné výrazně omezit. Předpokládejme, že budeme vyhledávat pouze otisky s hammingovou vzdáleností menší nebo rovnou *n*, neboli takové otisky, které se liší v *n* bitech. Pokud každý otisk vyjádřený v binární formě rozdělíme na *n+1* částí, nutně musí být alespoň jedna z *n+1* částí pro dva porovnávané otisky shodná. Poté stačí vyhledat v databázi takové záznamy, které mají alespoň jednu část otisku stejnou a pouze pro tyto záznamy zjistit výslednou hodnotu hammingovy vzdálenosti.

V prototypovém řešení je uvedený postup implementován pro vyhledání hodnot SimHash s hammingovou vzdáleností nejvýše 3. V SQL databázi jsou za tím účelem uloženy 4 části hodnoty SimHash, přičemž každá část představuje 16 bitů hodnoty SimHash příslušného inzerátu. Pro zadaný inzerát a jemu odpovídající hodnotu SimHash je možné sestrojit SQL dotaz, jehož výsledkem jsou všechny hodnoty SimHash, které mají alespoň jednu čtvrtinu shodnou. Zjednodušený zápis takového dotazu by mohl vypadat následovně:

```
SELECT Id, Simhash FROM Documents
WHERE Q1 = @q1 OR Q2 = @q2 OR Q3 = @q3 OR Q4 = @q4
```

kde ve sloupci `Simhash` je uložena celá hodnota SimHash, ve sloupcích `Qn` jsou uloženy jednotlivé čtvrtiny a parametry `@qn` určují jednotlivé čtvrtiny hodnoty SimHash zadaného inzerátu. V prototypovém řešení je uvedený způsob vyhledávání duplicitních inzerátů implementován v aplikaci `simhash.exe`, přičemž metoda vyhledávání hodnot SimHash v databázi dostupná ve třídě `DocumentRepository`.[7] Pomocí této aplikace bylo např. zjištěno, že v celé databázi obsahující více než 30 tisíc inzerátů existuje 766 inzerátů, ke kterým byl nalezen duplicitní inzerát, pro 68 z nich dokonce existuje více než jeden duplicitní inzerát. Konkrétní příklad nalezených duplicit je uveden v tabulce 7.1.

Inzerát	Nalezené duplicity
63	1080(2)
86	9852(0)
269	1547(2) 357(3)
1118	1144(0) 1164(0) 1173(0)
10054	10057(0) 10063(0) 10067(0) 10074(0)

Tabulka 7.1: Duplicitní inzeráty nalezené pomocí aplikace simhash.exe. Každý inzerát je určen svým identifikátorem v provozní databázi, v závorce je uvedena hodnota hammingovy vzdálenosti příslušných hodnot SimHash.

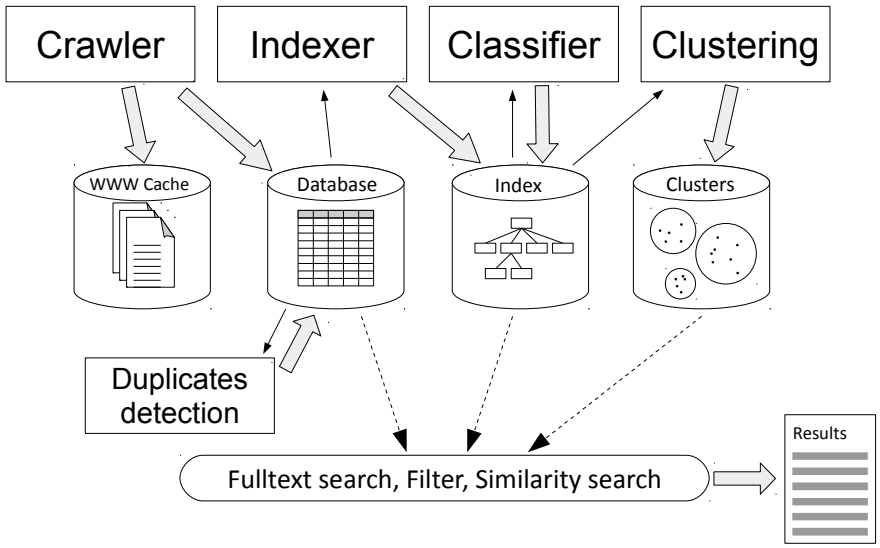
7 Namespace `Thon.WebMining.Documents.Impl`, projekt `Thon.WebMining`.

8 Implementace jednoduchého IR systému

V kapitolách 3 až 7 byly popsány metody použité pro zpracování realitní inzerce nutné pro vytvoření prototypového řešení IR systému. V těchto kapitolách jsou rovněž uvedeny odkazy na příslušné části zdrojových kódů, ve kterých jsou popsané algoritmy a metody implementovány. Předmětem této kapitoly je celkový pohled na architekturu prototypového řešení a doplnění některých dalších implementačních detailů.

Prototypové řešení zajišťuje zejména názornou prezentaci výsledků metod použitých pro automatizovaného zpracování dat. Funkčnost prototypového řešení vychází z cílů práce uvedených v kapitole 1.1. Jeho součástí je i jednoduché grafické uživatelské rozhraní poskytující základní funkce IR systému, jako je fulltextové vyhledávání v obsahu inzerátů, filtrování inzerátů dle vybraných kritérií, zobrazení nalezených duplicit mezi inzeráty a vyhledávání podobných inzerátů. Aplikace je řešena jako desktopová aplikace využívající lokálně uloženou databázi, index a další potřebná data. Uživatelské rozhraní je realizováno aplikací `gui.exe`.[1]

Celý IR systém je tvořen několika komponentami, které spolu určitým způsobem spolupracují. Všechny komponenty jsou znázorněny na obrázku 8.1. Šipky znázorňují proces zpracování a ukládání dat. **Crawler** zajišťuje získání požadovaných HTML stránek z realitních portálů, jejich uložení do WWW Cache, základní předzpracování inzerátů a naplnění provozní databáze. **Indexer** využívá data uložená v provozní databázi k vytvoření fulltextového indexu. **Klasifikátor** zajišťuje třídění či kategorizaci inzerátů. Výsledky klasifikace jsou opět uloženy do indexu. Proces **shlukování** zajišťuje vyhledání skupin podobných inzerátů a vytvoření datového souboru s údaji o vytvořených shlucích. Proces



Obrázek 8.1: Kompletní IR systém. Na obrázku jsou znázorněny základní komponenty IR systému, použitá úložiště dat a tok dat v procesu jejich zpracování.

1 Projekt `gui` ve zdrojových kódech prototypového řešení.

detekce duplicit zajišťuje vytvoření hodnot SimHash a jejich uložení do samostatné tabulky v provozní databázi. Vybraná úložiště jsou následně využita při vyhledávání a zobrazení výsledků uživateli. V následujících podkapitolách budou uvedeny podrobnější údaje týkající se návrhu řešení jednotlivých komponent IR systému.

8.1 WWW Cache

Základním předpokladem funkčnosti celého IR systému je získání požadovaných zdrojových dat. Tuto činnost zajišťuje crawler, jehož architektura je popsána v kapitole 3.2. Jedním z požadavků na IR systém je poskytnutí zdrojových HTML stránek ke všem inzerátům uloženým v databázi a vyhledaným pomocí uživatelského rozhraní. Za tím účelem jsou všechny stažené HTML stránky ukládány do souborového systému. Stránky získaného z jednotlivých realitních portálů jsou ukládány do samostatných podadresářů (v našem případě jde o podadresáře `Avizo-Reality`, `Bazos-Reality`, `iByty-Reality` a `iDnes-Reality`). Základní adresář celého úložiště je nastaven v konfiguračním souboru aplikace volbou `CrawlerStorageDirectory`. Obsahem každého podadresáře je SQLite databáze uložená v souboru `files.db`. Databáze obsahuje jednu tabulku, která mapuje zdrojové URL adresy na názvy souborů obsahující příslušnou HTML stránku. Soubory jsou označeny jednoznačným identifikátorem typu GUID.[2] Aby nedocházelo k ukládání velkého množství souborů do jednoho adresáře, je každý soubor uložen v podadresáři, jehož název je tvořen prvními dvěma znaky z názvu souboru.

Implementace souborového úložiště je realizována třídami v namespace `Thon.WebMining.Crawler.LocalStorage` (projekt `Thon.WebMining`). Třídy poskytují jednoduché API pro uložení souboru do úložiště a načtení obsahu souboru z úložiště na základě zadané URL adresy. Implementace úložiště zajišťuje i automatické vytvoření pomocné databáze při uložení prvního souboru. Součástí zdrojových kódů je i SQL skript použitý pro vytvoření databáze. Ostatní části aplikace využívající toto API nejsou závislé na vnitřní struktuře úložiště, ale pracují pouze s URL adresami, které jednoznačně identifikuje realitní inzeráty.

8.2 Provozní databáze

Data v provozní databázi jsou produkována procesem crawleru. Ten zajistí zpracování každé stažené HTML stránky, přičemž výstupem je částečně strukturovaný inzerát. Proces zpracování HTML stránky je popsán v kapitole 3.1. Výstupem zpracování je datová struktura inzerátu, která je v aplikaci reprezentována polem objektů typu `Item`.[3] Tato datová struktura je ukládána do provozní databáze s použitím XML serializace.

Všechny takto získané inzeráty jsou uloženy v jedné SQLite databázi (uložené v souboru `documents.db`). Tabulka `SiteInfo` obsahuje označení a název realitního portálu. Tabulka `Document` obsahuje jednotlivé inzeráty. U každého inzerátu jsou uloženy následující údaje:

- označení realitního portálu, ze kterého byl získán (formou cizího klíče do tabulky `SiteInfo`),

2 Viz http://en.wikipedia.org/wiki/Globally_unique_identifier.

3 Namespace `Thon.WebMining.Documents`, projekt `Thon.WebMining`.

- původní URL adresu, na které byl inzerát zveřejněn,
- XML strukturu popisující vlastnosti inzerátu a
- datum získání inzerátu.

Manipulace s databází inzerátů je implementována ve třídě `DocumentRepository`.[4] Tato třída není závislá na konkrétní použité databázi, ale využívá jednoduchou abstraktní vrstvu popsanou rozhraními v namespace `Thon.WebMining.DataLayer` závislou pouze na rozhraní ADO.NET. Dostupná je implementace právě pro použitou SQLite databázi, nicméně tato vrstva umožňuje relativně snadnou implementaci podpory jiného relačního databázového systému. Proces crawleru zajišťuje automatické vytvoření databáze při prvním spuštění. Konfigurace databáze je řešena pomocí standardních konfiguračních souborů a je stejná ve všech aplikacích prototypového řešení. Součástí zdrojových kódů je skript pro vytvoření provozní databáze dostupný v souboru `CreateDatabase.sql`.[5]

Crawler implementovaný konzolovou aplikací `crawler.exe` umožňuje spuštění procesu zpracování pro jeden určený realitní portál. Realitní portál je určen vstupním parametrem aplikace, přičemž jeho hodnota odpovídá plně kvalifikovanému názvu třídy implementující podporu daného portálu. V prototypovém řešení jsou dostupné tyto třídy:

- `Thon.WebMining.Portal.AvizoSiteFactory`,
- `Thon.WebMining.Portal.BazosSiteFactory`,
- `Thon.WebMining.Portal.IBytySiteFactory` a
- `Thon.WebMining.Portal.IDNESSiteFactory`.

8.3 Fulltextový index

Fulltextový index je vytvářen samostatnou aplikací `search.exe`. Tato aplikace poskytuje i jednoduché možnosti vyhledávání pro potřeby ověření funkčnosti. Z důvodu jednoduchosti implementace je v prototypovém řešení realizováno pouze jednorázové vytvoření indexu, není řešeno průběžné indexování nově získaných inzerátů. Proces indexování je implementován ve třídě `Indexer`[6] a zahrnuje zpracování všech inzerátů uložených v provozní databázi, extrahování textových a strukturovaných údajů pro potřeby vyhledávání a vytvoření indexu. Textovými informacemi jsou zejména nadpis inzerátu, obsah inzerátu a textové části všech získaných údajů popisujících podrobnosti inzerátů. Každý realitní portál poskytuje jinou množinu údajů popisujících inzerovanou nemovitost. Pro potřeby fulltextového vyhledávání jsou všechny takové údaje spojeny do jedné položky označené `podrobnosti`. Strukturovanými informacemi uloženými v indexu jsou typ obchodu, kategorie nemovitosti a cena inzerátu. Získání těchto specifických údajů je podrobněji popsáno v dalších podkapitolách.

Pro implementaci fulltextového indexu je použita knihovna Apache Lucene.NET. Součástí prototypového řešení je implementace filtru zajišťujícího zpracování českých slov – tzv. *stemming*. Jde o třídu `CzechAnalyzer`,[7] která je vytvořena převodem stejnojmenné

4 Namespace `Thon.WebMining.Documents.Impl`, projekt `Thon.WebMining`.
5 Adresář odpovídající umístění namespace `Thon.WebMining.Documents.Impl`.
6 Namespace `Thon.WebMining.Fulltext`, projekt `Thon.WebMining`.
7 Namespace `Thon.WebMining.Utils.Analyzers`, projekt `Thon.WebMining`.

třídy z projektu Apache Lucene pro prostředí Java. Indexované inzeráty je možné do určité míry strukturovat pomocí tzv. *fields*. V prototypovém řešení je použita tato logická struktura indexu:

- `id` – hodnota jednoznačného identifikátoru inzerátu v provozní databázi,
- `nadpis` – nadpis inzerátu, pokud je realitním portálem poskytován,
- `text` – obsah inzerátu (dostupný u všech inzerátů),
- `podrobnosti` – textové údaje popisující inzerovanou nemovitost (údaj nemusí poskytovat všechny realitní portály),
- `cena` – cena inzerátu v číselné formě pro zajištění filtrování v uživatelském rozhraní,
- `cenaText` – původní textové vyjádření ceny použité pro zobrazení ve výsledcích vyhledávání (většinou obsahuje i označení měny a oddělovače řádů),
- `source` – URL adresa, ze které byl inzerát získán,
- `trade` – označení typu obchodu (poptávka, prodej, pronájem) a
- `category` – označení kategorie nemovitosti (byt, dům, komerční objekt, apod.).

Pro vyhledávání inzerátů pomocí uvedené aplikace je možné využít dotazy ve formátu Apache Lucene Query.[8] Filtrování inzerátů dle ceny, typu obchodu a kategorie dostupné v uživatelském rozhraní je řešeno tzv. *filtry*.[9]

8.4 Zjištění ceny z inzerátu

V kapitole 2.4.1 byly uvedeny některé problémy při získávání konkrétních údajů z textu inzerátu. Pro zajištění jednoduchého strukturovaného vyhledávání by bylo vhodné získat z textu inzerátu co možná nejvíce údajů ve strukturované formě. Součástí práce je řešení jednoho jednoduchého případu, a to vyhledání ceny inzerátu. Získání ceny v číselné formě poskytne uživateli možnost filtrování inzerátů dle zadaného rozmezí ceny. Řešení problému je výrazně zjednodušeno tím, že všechny použité realitní portály poskytují údaj o ceně v částečně strukturované formě, tzn. u všech portálů je možné z HTML stránky získat textový údaj oddělený od ostatních částí inzerátů obsahující právě údaj o ceně.

Získání ceny je řešeno pomocí systému pravidel, která jsou aplikovaná na jednotlivá slova textu obsahujícího cenu. V tomto případě je text dělen na slova tak, že za oddělovače slov je považován jeden nebo více po sobě jdoucích netisknutelných znaků. Následně jsou slova převedena na malá písmena. Obě tyto operace jsou implementovány pomocí tříd `WhitespaceTokenizer` a `LowerCaseFilter` z knihovny Apache Lucene.NET. Postup získání ceny je následující:

1. Rozdělíme text obsahující cenu inzerátu na slova.
2. Pro každé slovo zjistíme, zda odpovídá regulárnímu výrazu `[0-9]+` (tj. obsahuje pouze číselné znaky).
3. Všechny výskyty odpovídající regulárnímu výrazu přidáme na konec pomocného řetězce.

8 Viz http://lucene.apache.org/core/3_0_3/queryparsersyntax.html.
9 Viz http://lucene.apache.org/core/3_0_3/api/core/org/apache/lucene/search/Filter.html.

4. Pokud slovo neodpovídá regulárnímu výrazu, ukončíme zpracování textu.
5. Hodnotu pomocného řetězce převedeme na číselnou hodnotu.

V některých inzerátech je primárně uvedená cena v eurech, přičemž odpovídající cena v korunách je uvedena v závorce za ní. Proto je v uvedeném postupu doplněno pravidlo, které při nalezení výskytu výrazu `EUR` vymaže hodnotu pomocného řetězce a celý postup se opakuje od bodu 1. Uvedený systém pravidel je v případě prototypového řešení dostačující pro spolehlivé zjištění ceny inzerátu. Je zřejmé, že v případě zpracování inzerátů z jiných realitních portálů může být postup výrazně složitější.

Implementace uvedeného postupu je v prototypovém řešení dostupná ve třídě `FeatureExtractorBase`.[10] Pro ověření funkčnosti je možné využít aplikaci `preprocess.exe`, jejíž výstupem je výpis identifikátorů všech inzerátů uložených v provozní databázi, u každého identifikátoru je uvedena zjištěná cena a původní textový údaj obsahující cenu. Uložení ceny do indexu za účelem vyhledávání je součástí procesu indexování.

8.5 Aplikování modelů

V kapitole 2.3 je uvedeno, že vhodným způsobem pro vyjádření textového dokumentu pro účely klasifikace a shlukové analýzy je tzv. *Vector Space Model*. Jak bylo uvedeno v kapitole 4.1 a v úvodu kapitoly 6, vhodnou datovou strukturou pro ukládání údajů důležitých pro vyjádření vektoru dokumentu je index. V prototypovém řešení ale potřebuje, aby byly do indexu uloženy inzeráty s již známou třídou přiřazenou při provedení klasifikace. Za tím účelem by bylo možné vytvořit nejprve pomocný index, ve kterém by byly pouze inzeráty z trénovací množiny. Pomocí takového indexu by bylo možné vytvořit klasifikátor a ten následně použít pro určení třídy všech inzerátů v provozní databázi. Z důvodu jednodušší implementace byl v prototypovém řešení zvolen odlišný způsob.

Proces indexování popsaný v kapitole 8.3 je rozdělen do dvou fází. V první fázi je do indexu uložen pouze textový obsah inzerátu, který je postačující pro vytvoření klasifikátorů. Druhou fází je aplikování vytvořených klasifikátorů na všechny inzeráty uložené v provozní databázi a doplnění ostatních položek popisujících inzeráty do indexu. Součástí těchto položek je i označení tříd přiřazených použitými klasifikátory. Aplikování klasifikačního modelu je prováděno při procesu indexování aplikací `search.exe`. Ta vyžaduje klasifikátory vytvořené pomocí aplikace `classifier.exe`, které jsou binárně serializované v souborech.

Po vytvoření indexu je možné provést shlukovou analýzu pomocí aplikace `clustering.exe`. Tato aplikace využívá pouze index, jejím výstupem je pak soubor obsahující informace o nalezených shlucích (popsáno v úvodu kapitoly 5). Nezávisle na klasifikaci a shlukové analýze je možné provést přípravu údajů potřebných pro detekování duplicitních inzerátů. To je provedeno aplikací `simhash.exe`, která do provozní databáze doplní pomocnou tabulku `Simhash` obsahující hodnoty SimHash všech inzerátů (jak je popsáno v kapitole 7). Po provedení těchto přípravných kroků jsou k dispozici všechny typy úložišť využité uživatelským rozhraním IR systému.

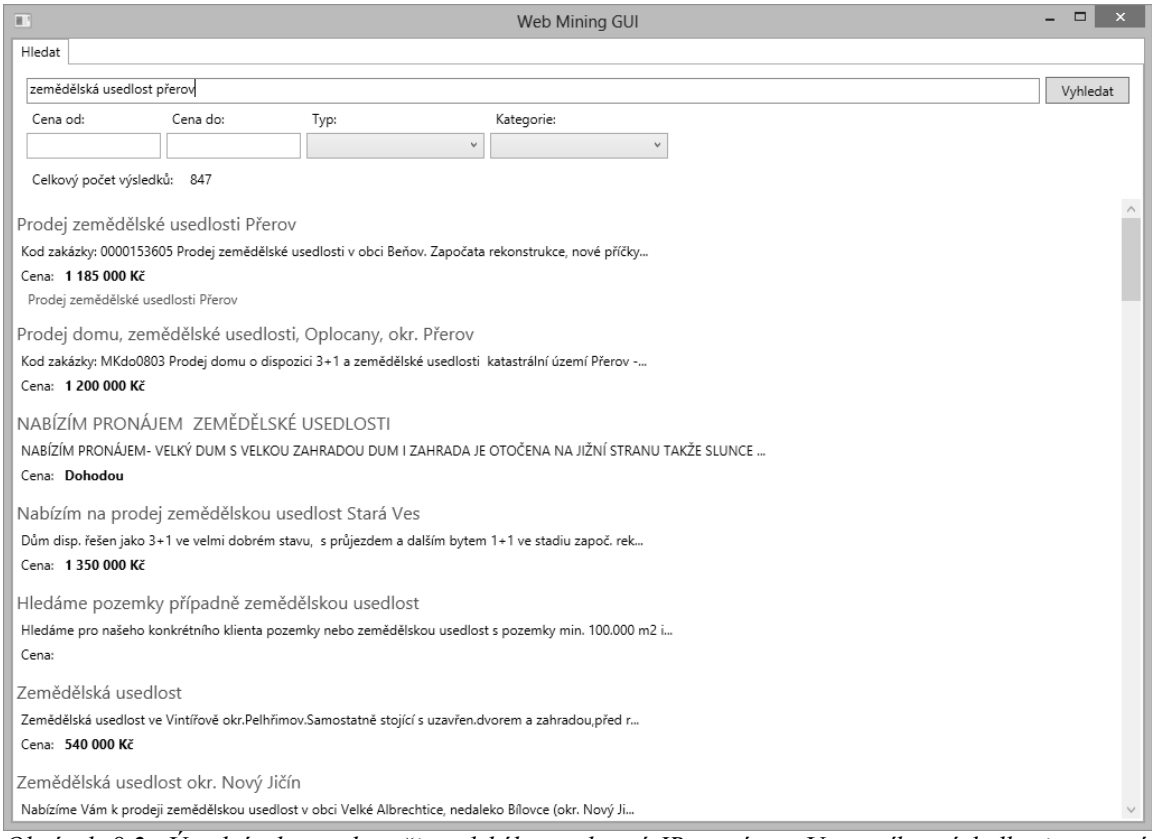
10 Namespace `Thon.WebMining.Fulltext`, projekt `Thon.WebMining`.

8.6 Prezentace výsledků

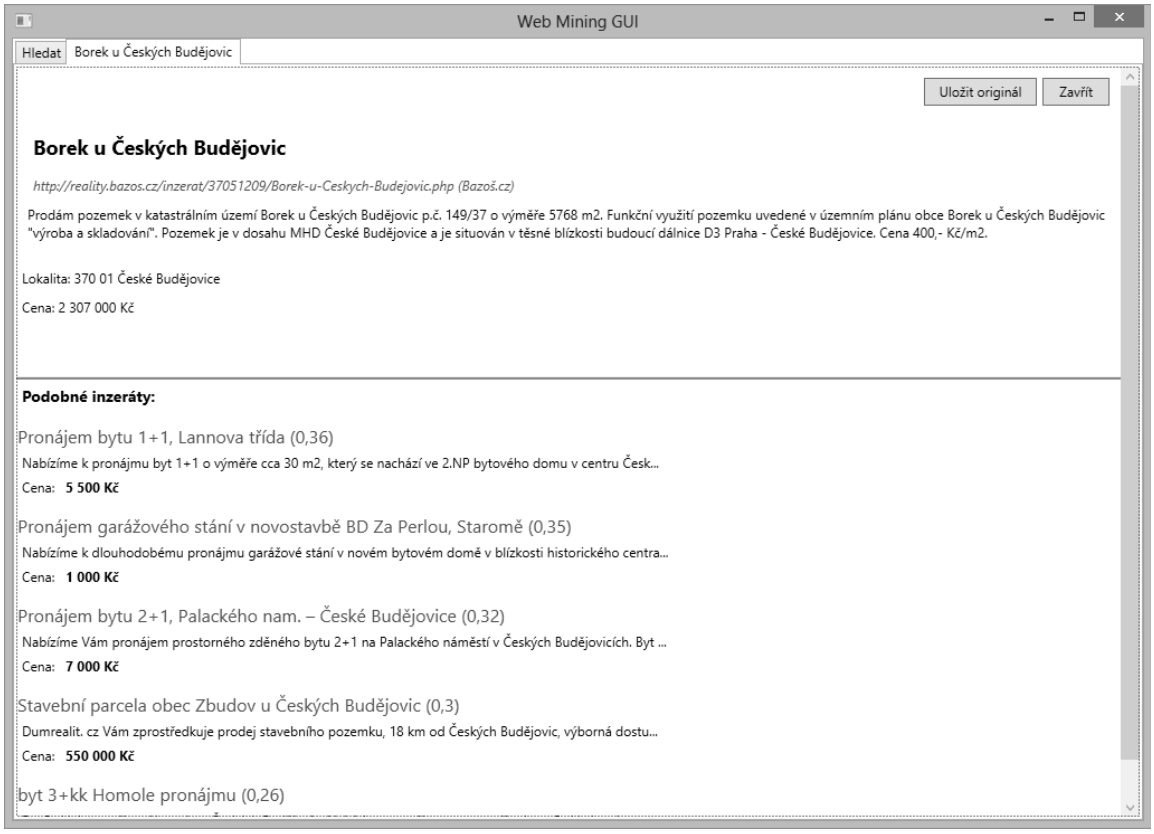
Splnění cílů práce uvedených v kapitole 1.1 je možné ověřit pomocí uživatelského rozhraní IR systému. Jde o desktopovou aplikaci umožňující vyhledávání získaných realitních inzerátů. K tomu využívá připravený index, provozní databázi s předpřipravenými hodnotami SimHash a výsledek shlukové analýzy. Vyhledávání podobných inzerátů je prováděno metodou Cluster pruning. Pro stažení originálního HTML souboru vybraného inzerátu je využita WWW Cache.

Uživatelské rozhraní je vytvořené s použitím Windows Presentation Foundation. K ovládání slouží dva typy obrazovek. Úvodní obrazovka nabízí vyhledávání inzerátů pomocí fulltextových dotazů s možností filtrování dle ceny, typu obchodu a kategorie nemovitosti. Po vyhledání je zobrazeno prvních 50 výsledků a celkový počet výsledků odpovídajících zadaným kritériím. Pokud je k některému ze zobrazených výsledků nalezen duplicitní inzerát, je zobrazen jako součást jednoho výsledku vyhledávání. Příklad takového výsledku vyhledávání je vidět na obrázku 8.2.

Po kliknutí na nadpis vyhledaného inzerátu je na nové záložce zobrazen jeho kompletní obsah v částečně strukturované formě tak, jak je uložen v provozní databázi. K inzerátu je zobrazeno 5 nejpodobnějších inzerátů s uvedením míry podobnosti (hodnotou kosinové vzdálenosti). V seznamu podobných inzerátů jsou zobrazeny pouze inzeráty s mírou podobnosti nejméně 0,2. Na záložce s obsahem inzerátu je možné získat zdrojový HTML soubor inzerátu získaný z uvedené URL adresy. Příklad zobrazení celého inzerátu s nalezenými podobnými inzeráty je vidět na obrázku 8.3.



Obrázek 8.2: Úvodní obrazovka uživatelského rozhraní IR systému. U prvního výsledku je patrné spojení výsledku s nalezeným duplicitním inzerátem.



Obrázek 8.3: Obsah inzerátu. V horní části obrazovky je uveden nadpis, URL adresa zdroje, text inzerátu a podrobnosti inzerátu. V dolní části obrazovky je seznam podobných inzerátů s mírou podobnosti uvedenou v závorce za nadpisem.

9 Závěr

Cílem práce byla aplikace algoritmů a postupů dolování dat pro oblast realitní inzerce. Význam byl kladen především na problémy vyžadující netriviální řešení. Jde např. o problémy klasifikace textových dokumentů a vyhledávání podobných dokumentů. Za tím účelem byly popsány a implementovány vybrané techniky a algoritmy strojového učení. Kromě algoritmů samotných byl v práci řešen celý proces dolování dat. Ten zahrnuje získání dat, jejich předzpracování, vytvoření a aplikování data miningových modelů a interpretaci výsledků.

Výsledkem práce je prototypové řešení jednoduchého Information Retrieval systému, který umožňuje získání realitních inzerátů z veřejně dostupných zdrojů, jejich zpracování a poskytuje vyhledávání a prezentaci výsledků uživateli. Jeho součástí je databáze obsahující přes 30 tis. realitních inzerátů. Dílčí výsledky prezentované v jednotlivých kapitolách byly získány právě na základě vytvořené databáze realitních inzerátů.

Pro řešení všech problémů byly zvoleny metody určené pro zpracování nestrukturovaných dat. Jde o přístup, kdy jsou nestrukturovaná textová data ponechána pokud možno ve své původní formě. Tomu je přizpůsoben i způsob vyhledávání v takových datech, kdy namísto vytváření strukturovaných dotazů, na základě kterých jsou data filtrována, je možné vyhledávat pomocí tzv. free text queries.

Odlišným přístupem by mohlo být získání co největšího množství údajů z textu inzerátů, jejich uložení ve strukturované formě a následné zpracování takto strukturovaných dat. Uvedený přístup byl zvolen proto, že dolování dat z nestrukturovaného textu a ze strukturované databáze představuje ve své podstatě dvě samostatné disciplíny. Obdobně by bylo nevhodné při zpracování strukturovaných dat tato nejprve transformovat do nestrukturované formy.

Existuje nepochybně velké množství požadavků na získání dalších údajů týkajících se realitních inzerátů. Příkladem může být problém detekce odlehlých hodnot, přičemž odlehlou hodnotou může být určitým způsobem zajímavý inzerát. Další zajímavou úlohou by mohlo být provádění odhadů cen nemovitostí v závislosti na lokalitě s využitím metody lineární regrese. V tomto případě by bylo nutné provést takové předzpracování dat, které zajistí spolehlivé získání některých údajů ve strukturované formě. Konkrétně jde o určení lokality a ceny nemovitosti. Stejné údaje by bylo možné využít i pro provádění statistických výstupů, jako jsou např. cenové mapy. Tyto problémy tedy mohou být námětem pro další využití výsledků práce.

Seznam použité literatury

[1] BOBICEV, Victoria a SOKOLOVA, Marina. *An Effective and Robust Method for Short Text Classification* [online]. 2008 [cit. 2014-07-22]. Dostupné z: http://www.aaai.org/Papers/AAAI/2008/AAAI08-230.pdf

[2] YUAN, Quan, CONG, Gao a THALMANN, Nadia M. *Enhancing Naive Bayes with Various Smoothing Methods for Short Text Classification* [online]. 2012 [cit. 2014-07-26]. Dostupné z: http://www3.ntu.edu.sg/home/gaocong/papers/wpp095-yuan.pdf

[3] BINGFENG, Pi, SHUNKAI, Fu, WEILEI, Wang a SONG, Han. *SimHash-based Effective and Efficient Detecting of Near-Duplicate Short Messages* [online]. 2009 [cit. 2014-07-22]. Dostupné z: http://academypublisher.com/proc/iscsct09/papers/iscsct09p20.pdf

[4] MANNING, Christopher D., RAGHAVAN, Prabhakar, SCHÜTZE, Hinrich. *Introduction to Information Retrieval.* New York: Cambridge University Press, 2008. ISBN 978-0-521-86571-5.

[5] BERKA, Petr. *Dobývání znalostí z databází*. Praha: Academia, 2003. ISBN 80-200-1062-9

[6] EIBE, Frank a REMCO, R. Bouckaert. *Naive Bayes for Text Classification with Unbalanced Classes* [online]. 2006 [cit. 2014-08-06]. Dostupné z: http://www.cs.waikato.ac.nz/~eibe/pubs/FrankAndBouckaertPKDD06new.pdf

[7] WEIZHONG, Zhao, HUIFANG Ma a QING, He. *Parallel K-Means Clustering Based on MapReduce* [online]. 2009 [cit. 2014-11-01]. Dostupné z: http://www.cs.ucsb.edu/~veronika/MAE/parallelkmeansmapreduce_zhao.pdf

[8] SADOWSKI, Caitlin a LEVIN, Greg. *SimiHash: Hash-based Similarity Detection* [online]. 2011 [cit. 2014-10-13]. Dostupné z: http://users.soe.ucsc.edu/~supertri/docs/simihash.pdf

Rejstřík