Univerzita Pavla Jozefa Šafárika v Košiciach

Prírodovedecká fakulta

EXTRAKCIA DÁT NA ZÁKLADE KATEGORIZÁCIE WEBOVÝCH STRÁNOK

DIPLOMOVÁ PRÁCA

Študijný odbor:	**Informatika**
Školiace pracovisko:	**Ústav informatiky**
Vedúci záverečnej práce:	**RNDr. Róbert Novotný, PhD.**

Košice 2012

Bc. Pavol Rajzák

Poďakovanie

Vedúcemu práce RNDr. Róbertovi Novotnému, PhD. za postrehy a vedenie práce a Kataríne Kubaskej za významné korektúry a trpezlivosť pri oprave chýb.

Univerzita P. J. Šafárika v Košiciach
Prírodovedecká fakulta

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Pavol Rajzák
Študijný program: Informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: Diplomová práca
Jazyk záverečnej práce: slovenský

Názov: Extrakcia dát na základe kategorizácie webových stránok

Cieľ:
1. Rozšíriť systém pre výber extrakčnej metódy o ďalšie algoritmy.
2. Navrhnúť spôsob pre získavanie zdrojov pre klasifikáciu a extrakciu.
3. Vytvoriť klasifikácie stránok, na základe ktorých bude možné zvoliť vhodnú extrakčnú metódu.
4. Vytvoriť používateľské rozhrania pre hodnotenie výsledkov extrakcie a ich následné zapracovanie do učiaceho mechanizmu systému.
5. Vykonať experimenty na reálnej aplikačnej doméne a vyhodnotiť ich efektivitu v porovnaní s existujúcimi riešeniami.

Literatúra:
[1] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma.: Extracting content structure for web pages based on visual representation. In Proceedings of the 5th Asia-Pacific web conference on Web technologies and applications (APWeb'03), Xiaofang Zhou, Maria E. Orlowska, and Yanchun Zhang (Eds.). Springer-Verlag, Berlin, Heidelberg. 2003. p. 406-417
[2] Qi X.,Davison B. D.: Web Page Classification: Features and Algorithms. In Technical Report LU-CSE-07-010, Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA, 18015.. 2007
[3] Liu B., Grossman R., Zhay Y.: Mining Data Records in Web Pages. In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2003. p. 601-606.

Kľúčové slová: extrakcia informácií z webu, klasifikácia webových stránok

Vedúci: RNDr. Róbert Novotný, PhD.
Ústav: ÚINF - Ústav informatiky

Dátum zadania: 01.08.2011

Dátum schválenia: 27.06.2012

prof. RNDr. Viliam Geffert, DrSc.
riaditeľ ústavu

Univerzita Pavla Jozefa Šafárika v Košiciach
Prírodovedecká fakulta
Ústav informatiky

Abstrakt

Práca sa zaoberá vytvorením systému pre automatickú extrakciu údajov z webových stránok. Systém bude implementovať viacero extrakčných metód, ktoré budú poskytovať rôzne výsledky v závislosti od vstupnej množiny stránok. Prvým krokom extrakcie bude automatická kategorizácia stránok vstupnej množiny, to znamená zaradenie do skupiny podľa štruktúry, obsahu a pod. Na základe tejto kategorizácie sa vyberie vhodná extrakčná metóda, ktorá poskytuje najlepšie výsledky.

Abstract

This thesis aims to develop a system for automatic data extraction from web pages. The system will implement a number of extraction methods, which will provide different results depending on the input set of pages. The first extraction step is an automatic categorization of pages in input set, in other words splitting into groups according to the structure, content and so on. Based on this categorization the system will select appropriate extraction method that provides the best results.

Obsah

Úvod

Táto diplomová práca nadväzuje na bakalársku prácu *Kolaboratívna anotácia webových stránok* [1], ktorá sa venovala problému poloautomatickej extrakcie dát z webových stránok a ich ďalšie spracovanie. Navrhnutý systém poskytoval rozhranie pre pridávanie rôznych metód, ktoré slúžili na spracovanie stránok rôzneho typu. V tejto práci bolo naznačené to, že pri vhodnej kategorizácii webových stránok, by bolo možné zovšeobecniť výber extrakčného algoritmu, ktorý by (ideálne bez zásahu používateľa) vyselektoval relevantné informácie a objekty a uložil ich do jednotnej štruktúry.

Cieľom tejto diplomovej práce je rozšíriť systém navrhnutý v bakalárskej práci o automatickú kategorizáciu vstupnej množiny stránok, ktorá by zodpovedala predpokladom nutným pre použitie konkrétnej extrakčnej metódy. Pre tento proces bude nutné zaviesť klasifikáciu, ktorá dokáže prideliť webovej stránke vektor určujúci percentuálny podiel príslušnosti obsahu alebo štruktúry stránky k niektorej z vybraných kategórií. Výber kategórií by mal odzrkadľovať hlavné typy webových stránok. Táto klasifikácia neskôr napomôže aj pri samotnej anotácii obsahu.

Obsah tejto práce je členený do niekoľkých kapitol podľa tematiky, ktorou sa zaoberá. Prvá kapitola opisuje problém kategorizácie a extrakcie a poskytuje niekoľko motivačných príkladov, ktoré predstavujú výzvu pre tento problém. Druhá kapitola je venovaná definovaniu kategorizácie a popisu kategorizačných metód a algoritmov. V tretej kapitole sa venujeme zhrnutiu získaných poznatkov o extrakcii dát a problémami, ktoré treba riešiť. V štvrtej kapitole sú popísané riešenia hlavných problémov, ktoré je nutné prekonať pre dosiahnutie cieľov práce. Ďalej nasleduje návrh riešenia a implementácia systému. Výsledky kategorizácie a extrakcie na reálnych dátach sú prezentované v siedmej kapitole. Posledná kapitola zhŕňa výsledky tejto práce a načrtáva možnosti ďalšieho využitia tohto výskumu.

Kapitola 1

Definovanie problému a motivácia

V moderne informovanej spoločnosti je existencia nástroja, ktorý by vedel získať užitočné informácie z množiny vstupných dát veľmi potrebná. Množstvo softvérových komponentov je závislých práve na správne štruktúrovanom zdroji vstupných údajov. Medzi ne patria napríklad systémy pre sledovanie reputácie produktu na webe (*ORS – Online reputation systems*) alebo webové portály, ktoré porovnávajú ceny identických produktov v rôznych elektronických obchodoch (*PCE – Price comparison engines).* Súčasné riešenia sú do veľkej miery závisle na vstupných dátach, napríklad systémy pre porovnávanie cien používajú ako vstupné dáta formátovaný výstup katalógu poskytnutý elektronickým obchodom. Toto riešenie však vylučuje z porovnávania cien všetky elektronické obchody, ktoré z rozličných dôvodov nie sú schopné exportovaný katalóg dodať.

Hlavnou myšlienkou nástrojov pre dolovanie informácií z webových stránok (*web-mining tools*) je vybrať a uložiť informácie z ľubovoľného zdrojového textu tak, aby sa dali použiť ako vstupné dáta pre niektorú z vyššie spomenutých aplikácií. Existujúce riešenia sa buď zameriavajú na špecifický typ stránok (katalógy s produktami, spravodajské portály a pod.) alebo sa spoliehajú na zásah používateľa, ktorý koriguje výsledok extrakcie.

Napríklad pri použití statických extrakčných metód, ktoré predpokladajú stabilnú štruktúru webovej stránky, je ťažšie prispôsobiť systém meniacim sa podmienkam. Tento prístup síce väčšinou zabezpečí relatívne vysokú účinnosť, ale len v statických podmienkach. Na druhej strane dynamické metódy, ktoré sa väčšinou sústredia viac na pochopenie významu a vzťahov medzi objektami na stránke, si často vyžadujú zásah

používateľa. Častokrát sa totiž stáva, že vstupná množina stránok nie je homogénna, a vtedy je prispôsobenie extrakcie používateľom náročná činnosť. Napríklad výstup vyhľadávania na internete pomocou kľúčových slov obsahuje rôzne typy stránok, ktoré síce majú spoločnú tému, ale líšia sa v spracovaní obsahu a štruktúre.

Navrhovaný systém by preto mal čo najviac obmedziť vplyv používateľa na proces extrakcie a mal by byť flexibilný a jednoduchý. Ako príklad môžeme uviesť zamestnávateľa, ktorý chce zistiť čo najviac informácií o uchádzačoch na pracovnú pozíciu, ktoré sa voľne nachádzajú na internete. Tu môžu patriť osobné stránky, prehľad doterajšej práce, fotografie zo sociálnych sietí alebo emailové adresy a telefónne čísla. Namiesto toho aby pri každom uchádzačovi vyhľadával informácie manuálne pomocou dostupných vyhľadávacích nástrojov, môže použiť náš navrhovaný systém, ktorý vyhľadá dostupné informácie, získa množinu vstupných stránok, priradí im kategórie a na základe tejto klasifikácie vyberie vhodné extrakčné metódy. Získané informácie sa potom uložia a zobrazia vo forme vhodnej na prezentáciu, napríklad pomocou priradenia do jednotlivých kategórií.

V tejto práci budeme pracovať s príkladom univerzálneho vyhľadávača informácií o produktoch, to znamená pre konkrétny produkt budeme zisťovať jeho atribúty, cenu, obchodný reťazec ktorý ho predáva, ale aj recenzie a hodnotenia produktov. Takýto systém bude mať formu vyhľadávača, ktorý na základe kľúčových slov vráti požadované množiny vlastností v používateľsky prívetivej a jednoduchej forme. Rovnako sa budeme v implementačnej fáze snažiť zostrojiť prototyp takéhoto nástroja a overiť jeho funkčnosť.

Kapitola 2

Kategorizácia webových stránok

Kategorizácia (alebo aj klasifikácia) webových stránok je proces zaradenia webovej stránky do predom definovaných kategórií. Často je označovaná aj ako proces učenia s asistenciou na množine označených dát, pomocou ktorých je možné priradiť označenie pre budúce dáta [2].

2.1 Úvod do problematiky

Vo všeobecnosti vieme problém klasifikácie webových stránok rozdeliť na niekoľko podproblémov: klasifikácia predmetu (subjektu) stránky, klasifikácia funkcie stránky, klasifikácia sentimentu stránky a niekoľko ďalších.

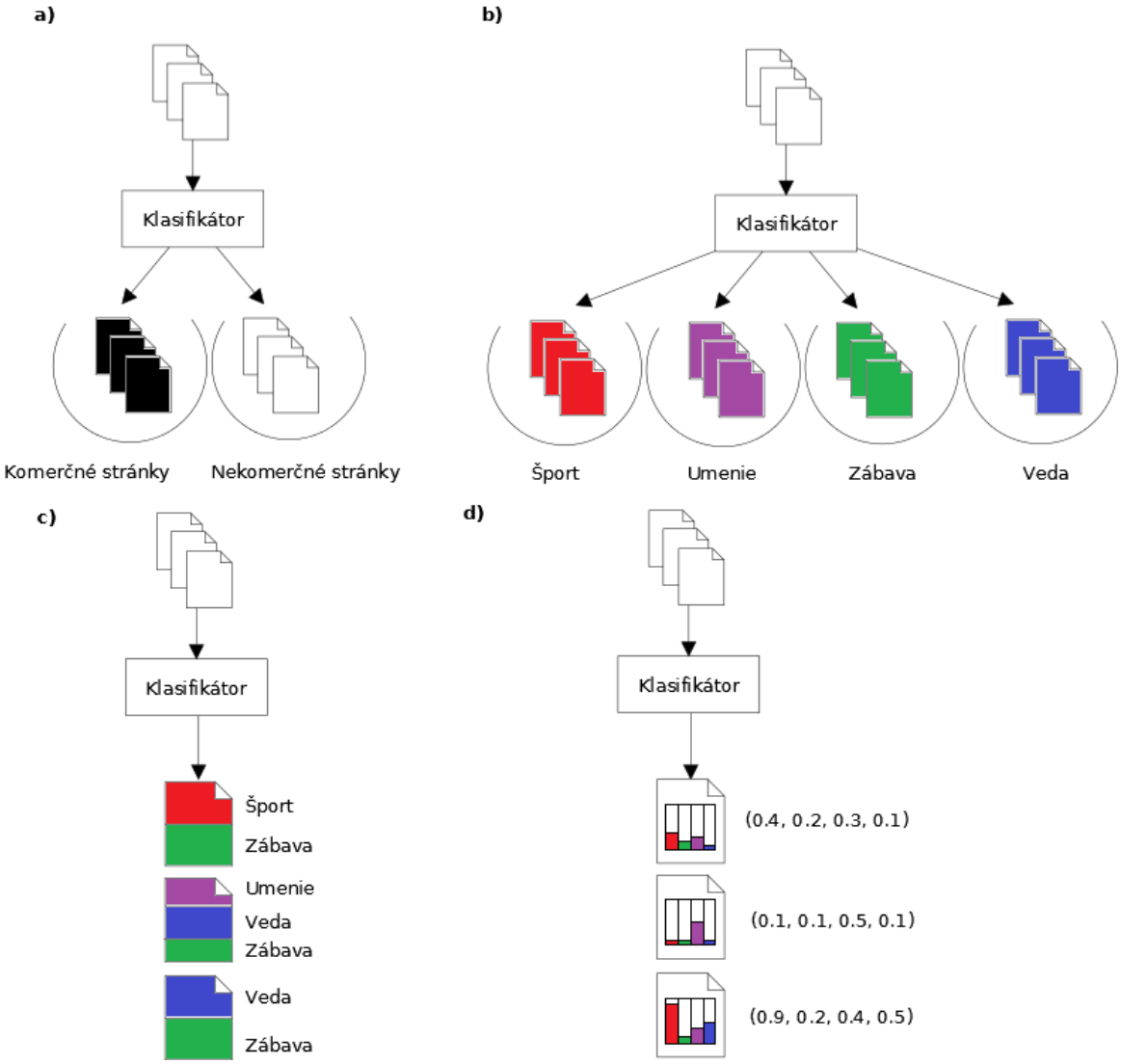
- Klasifikácia predmetu sa vzťahuje na zaradenie stránky do kategórie podľa témy, ktorá najviac reprezentuje obsah stránky. Tieto témy predstavujú niekoľko hlavných významových domén, ako napríklad *šport*, *umenie* alebo *spravodajstvo*. V našom príklade budú významové domény tvoriť oblasti v katalógu produktov, napríklad *elektronika, športové príslušenstvo, hobby a záhrada* a pod.

- Funkčná klasifikácia zaraďuje webovú stránku do kategórie podľa funkcie, ktorú zohráva, napríklad *osobná stránka*, *elektronický obchod*, resp. *popis produktu*, *katalóg s produktami* a pod.

- Význam klasifikácie sentimentu zase spočíva v zaradení webovej stránky do kategórie vyjadrujúcej postoj od pozitívnej, až po negatívnu, odstupňovaných podľa vopred stanovenej stupnice[1].

[1]Napríklad recenzie na produkt môžu mať pozitívny alebo negatívny charakter v závislosti od použitých slov a štylizácie.

V tejto práci sa budeme venovať najmä funkčnej a subjektívnej klasifikácii, keďže problém priradenia sentimentu webovej stránke si vyžaduje hlbšiu analýzu (viac v [7]).

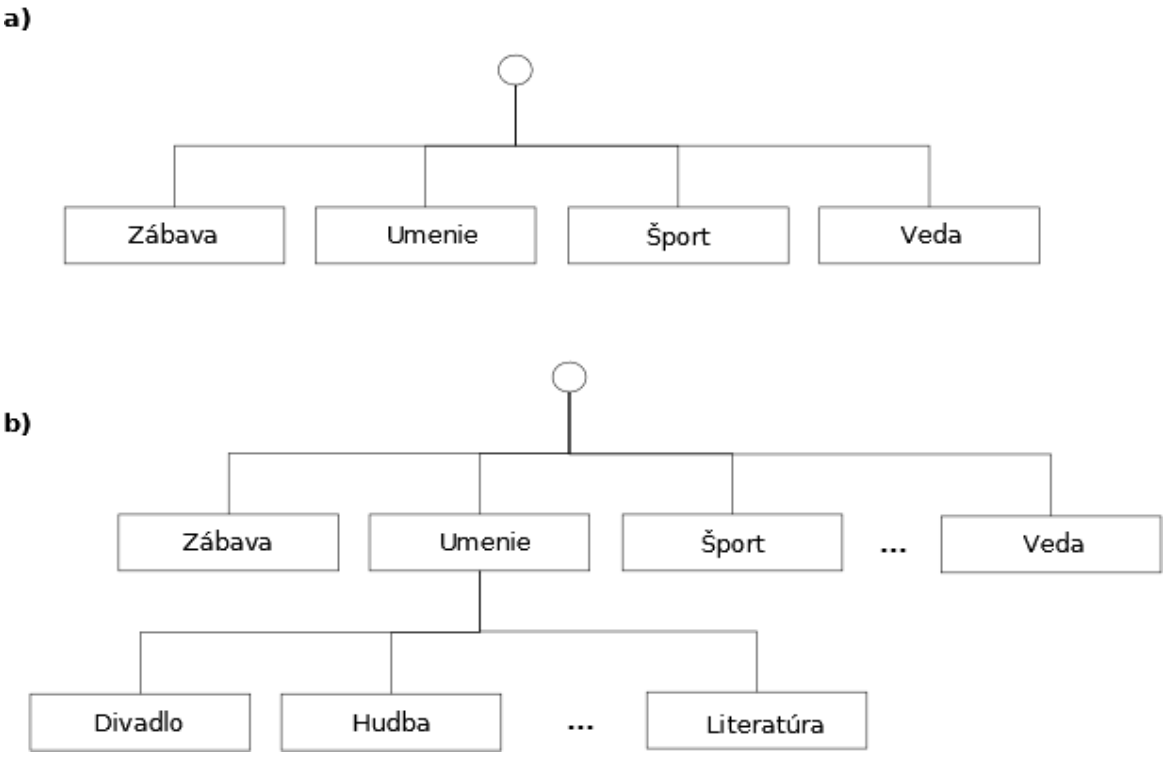


Obr. 2.1: Prehľad klasifikácie webových stránok.

V závislosti od počtu použitých podtried rozlišujeme klasifikáciu binárnu a n-nárnu. Ak chceme zaradiť stránku do jednej z dvoch kategórií, hovoríme o binárnej klasifikácii. Ak vyberáme z väčšieho množstva tried, jedná sa o n-nárnu klasifikáciu, tak ako je to znázornené na obr. 2.1 a) a b).

V praxi sa tiež vyskytuje to, že nie je možné zaradiť stránku do práve jednej kategórie. Vtedy hovoríme o viaczložkovej klasifikácii. Ak je problém klasifikácie n-nárny, kedy chceme napríklad zaradiť stránku do jednej z kategórií *šport*, *umenie*, *veda* alebo *zábava*, môžeme jej priradiť buď práve jednu alebo viac kategórií, ktoré

vyhovujú jej obsahu. Priradenie môže byť buď pevné (stránka patrí do kategórie) alebo voľné (stránka vyhovuje kategórií s nejakou pravdepodobnosťou), ako je to zobrazené na obr. 2.1 c) a d).



Obr. 2.2: Príklad kategorizácie podľa hĺbky väzieb.

Klasifikáciu môžeme rozlišovať podľa väzieb medzi kategóriami na jednoduchú a hierarchickú. Pri jednoduchej klasifikácii patria všetky kategórie pod jednu úroveň, pri hierarchickej sú radené pod sebou podľa príslušnosti (obr. 2.2).

Dôležité je tiež rozlišovať kategorizáciu webovej stránky ako celku a textu, ktorý sa nachádza na webových stránkach. Je niekoľko vlastností, ktoré odlišujú bežný textový dokument od webovej stránky. Webová stránka je štandardne prezentovaná vo forme značkovacích jazykov (XHTML, HTML), ktoré sú zobrazované webovým prehliadačom. Obsahuje rôzne druhy objektov, dynamicky sa meniaci obsah a v neposlednom rade prepojenia na ostatné webové stránky.

2.2 Metódy kategorizácie webových stránok

Existuje niekoľko prístupov klasifikácie webových stránok. Medzi hlavné patria:

a) **Manuálne zaradenie.** Používateľ definuje do ktorej z kategórií patrí daná stránka. Táto metóda je najpresnejšia a v konečnom dôsledku aj najspoľahlivejšia, no zároveň neefektívna pri väčšej množine vstupných stránok. Vhodná je na prvotné určenie niekoľkých kategórií pre neskoršie automatické učiace procesy.

b) **Zaradenie do klastrov.** Algoritmy používajúce klastre sú veľmi rozšírené, keďže môžu byť použité bez akejkoľvek znalosti pozadia - môžu spracovávať ľubovoľné vlastnosti webovej stránky. Väčšina klastrovacích algoritmov ako napríklad *K-Means* a pod. potrebujú mať vopred definovaný nejaký počet klastrov. Tento prístup je tiež náročnejší na výpočet.

c) **Kategorizácia založená na META tagoch.** Tento prístup sa spolieha len na META atribúty HTML, konkrétne `<META name="keywords"...>` a `<META name="description"...>`. Tieto dva tagy charakterizujú kľúčové slová a popis stránky. Problémom je to, že nie vždy sú tieto tagy správne vyplnené alebo chýbajú.

d) **Kategorizácia založená na textovom obsahu.** Táto metóda spracováva textový obsah pomocou vopred pripravenej databázy kľúčových slov pre danú kategóriu. Databáza obsahuje predpripravené kľúčové slová, ktoré boli vytvorené na základe frekvenčnej analýzy tréningovej množiny stránok charakteristických pre danú kategóriu, pričom sú vynechané často opakované a nerelevantné slová (tzv. *koncové slová*). Pre zaradenie webovej stránky do kategórie je nutné očistiť textový dokument o koncové slová. Zvyšné slová a frázy sú reprezentované pomocou *charakteristického vektora*. Takto spracovaný dokument je potom klasifikovaný do príslušnej kategórie na základe algoritmu hľadajúceho k-teho najbližšieho suseda. Tento prístup spolieha na kvalitne spracovanú tréningovú množinu pri vytváraní databázy kľúčových slov. Avšak obsah webových stránok sa líši čo sa týka kvality a aj kvantity. Z pozorovaní [3] vyplýva, že 94.65% webových stránok obsahuje menej ako 500 rozličných slov. Takisto priemerná frekvencia slov každého dokumentu je menej ako 2.0, čo znamená, že veľmi malý počet slov sa vyskytuje v nejakom dokumente viac ako dva krát.

e) **Analýza prepojení a kontextu.** Táto metóda pracuje nad množinou stránok, pričom predpokladá, že prepojenia ukazujúce na spracovávanú webovú stránku obsahujú dostatočné množstvo informácií na to, aby ju bolo možné klasifikovať. Referenčná stránka by mala naviesť používateľa, aby na daný odkaz klikol, preto musí prepojenie dostatočne označiť. Najjednoduchší spôsob zistenia popisu je jeho odvodenie z HTML tagu pre odkazy (`<A></A>`. Tieto však nemusia vždy obsahovať relevantné kľúčové slová, keďže množstvo tvorcov internetového obsahu používa nesémantické popisy, napríklad *Pre pokračovanie kliknite sem.* Komplexnejšie riešenie spočíva v extrakcii informácií o kontexte dokumentov, ktoré odkazujú na danú stránku, pomocou analýzy ich štruktúry [4] a [5]. Tento prístup je obzvlášť výhodný, keď samotná stránka poskytuje veľmi málo informácií o jej obsahu (napríklad obsahuje priveľa alternatívneho obsahu ako je grafika bez alternatívneho textu alebo prvky Flash - obr. 3.7 v kapitole 3.3).

2.3 Algoritmy pre kategorizáciu webových stránok

V predchádzajúcej sekcii boli spomenuté základné metódy kategorizácie webových stránok. V tejto sekcii sa budeme sústrediť na rôzne triedy algoritmov, ktoré dané metódy využívajú.

a) **Obmedzenie priestoru určeného na klasifikáciu.** Tento prístup sa zameriava na redukciu priestoru, s ktorým sa bude pracovať len na jeho podstatnú časť. Napríklad kategorizácia stránky spravodajského portálu sa bude sústrediť len na nadpis, perex článku a jeho obsah. Týmto štýlom sa odstránia nežiadúce elementy ako napríklad reklamy alebo odkazy na nesúvisiace články. Výhodou tohto predspracovania je nižšia zložitosť pri procese samotnej klasifikácie, ktorá je kombináciou viacerých (ľubovoľných) metód. Takýto druh klasifikácie je tiež presnejší. Nevýhodou tohto prístupu je to, že nie všetky webové stránky poskytujú možnosť redukcie, respektíve určovanie charakteristickej oblasti (*pagelet*) nie je vždy triviálny proces. Populárnou implementáciou tohto prístupu je *Latentné sémantické indexovanie (LSI)* [6], ktoré transformuje text dokumentov do menšej, menej intuitívnej formy. Avšak veľká výpočtová zložitosť znemožňuje nasadenie tejto metódy v praxi na veľkej množine webových stránok. Niektoré pokusy o vylepšenie tejto metódy [8] odporúčajú jej ďalší výskum.

b) **Algoritmy využívajúce učenie zo vzťahov.** Keďže webové stránky obsahujú prepojenia na ostatné webové dokumenty, existuje trieda algoritmov, ktorá využíva túto populárnu techniku založenú na automatizovanom učení. Najpopulárnejší z tejto kategórie je algoritmus využívajúci pozvoľné označovanie (*Relaxation labeling).* V kontexte kategorizácie hypertextových dokumentov tento algoritmus najprv použije textový klasifikátor na priradenie pravdepodobnosti príslušnosti k danej triede pre každú stránku. Potom v cykle zohľadní jej pravdepodobnosť na základe pravdepodobností jej susedov získaných v predchádzajúcej iterácii algoritmu. Konkrétne implementácie a modifikácie tohto prístupu sa ukázali ako dostatočne efektívne, aby boli nasadené na väčšie množstvo stránok [9].

c) **Modifikácia a zjednodušenie problému pre použitie na tradičných algoritmoch.** V praxi sa často vyskytuje aj použitie modifikácie klasických algoritmov na rozpoznávanie vzorov ako napríklad hľadanie k-teho najbližšieho suseda *k-Nearest Neighbor* a *Support Vector Machine (SVM)* v kontexte kategorizácie webových stránok. KNN klasifikátory vyžadujú určenie rozdielnosti dokumentov, aby bolo možné kvantifikovať vzdialenosť medzi dokumentom a každým ostatným dokumentom z tréningovej množiny. Z pozorovaní [10] vyplýva, že najvhodnejšou mierou odlišnosti dokumentov je tá, ktorá berie do úvahy opakovaný výskyt pojmov v dokumentoch. Čím viac je spoločných opakujúcich sa pojmov v dvoch dokumentoch, tým silnejšia je väzba medzi danými dokumentami. Taktiež je možné využiť pravdepodobnostný KNN algoritmus, kde pravdepodobnosť, že dokument patrí kategórii je určená na základe vzdialenosti od jej susedov a pravdepodobnosti, že títo susedia patria do požadovanej kategórie. Klasifikácia zvyčajne vyžaduje predpripravenú množinu ohodnotených webových stránok. Niektoré algoritmy založené na prístupe SVM [11] eliminujú potrebu manuálneho zberu ohodnotených príkladov, pri zachovaní rovnako presnej klasifikácie.

d) **Algoritmy pre hierarchickú klasifikáciu.** Väčšina existujúcich prístupov sa sústredí na kategorizáciu webových stránok v rámci jednej úrovne. Ako však už bolo spomenuté, niekedy je potrebné použiť zložitejšiu, viacúrovňovú hierarchiu. Existuje viacero algoritmov, ktoré umožňujú klasifikáciu na základe hierarchie, využívajúc pritom tradičné metódy ako napríklad *Rozdeľuj a panuj.* Tento prístup rozdelí problém klasifikácie na rôzne podproblémy v závislosti

od úrovne v hierarchii. Ďalšie vylepšenia navrhujú priradenie popisu z vyššej úrovne hierarchie, v prípade ak nie je možné určiť presne do ktorej kategórie spadá stránka na nižšej úrovni.

e) **Kombinácia informácií z viacerých zdrojov.** Ako najefektívnejšie sa zatiaľ javia algoritmy založené na kombinácii prístupov. V kategorizácii webových stránok je časté kombinovanie analýzy prepojení a obsahu [14]. Bežný postup ako spojiť rôzne výstupy samostatných algoritmov do jedného, je pristupovať k týmto výstupom ako k disjunktným množinám. Tie sú neskôr vhodne skombinované, napríklad pomocou *hlasovania* a *hromadenia* [15], poprípade *spoločného učenia*.

   Hlavným predpokladom pre úspech kombinácie týchto metód je to, že každá poskytuje rozličný pohľad na problém, a tak zachytáva viacej detailov. Nevýhodou môže byť zložitejšia implementácia a vyššia výpočtová zložitosť.

Kapitola 3

Extrakcia informácií

Extrakcia informácií (*information extraction*) typovo spadá pod oblasť získavania informácií (*information retrieval*). Jej cieľom je automaticky extrahovať štruktúrované informácie z neštruktúrovaných alebo čiastočne štruktúrovaných strojovospracovateľných dokumentov. Vo väčšine prípadov sa jedná o spracovanie ľudsky čitateľného textu, poprípade automatickú extrakciu a anotáciu multimediálneho obsahu. V súčasnosti sa jednotlivé metódy využívané pri extrakcii informácií sústredia na špecifické významové domény alebo štruktúry. To je spôsobené tým, že nie je možné navrhnúť postup, ktorý by vedel vykonať extrakciu informácií univerzálne a efektívne.

3.1 Základné pojmy a úvod do problematiky

Úloha formálne popisujúca proces extrakcie informácií obsahuje vstup a cieľ. Vstupom môže byť neštruktúrovaný dokument písaný v ľudsky čitateľnom jazyku alebo čiastočne štruktúrovaný dokument, ktorý je upravený pre zobrazenie na Internete (napríklad obsahuje tabuľky, číslované zoznamy a pod.). Cieľom extrakcie môže byť usporiadaná k-tica, kde k definuje počet atribútov v zázname alebo môže mať charakter komplexnejšieho objektu s hierarchicky usporiadanými atribútmi. Niektoré úlohy môžu mať žiaden alebo viacero inštancií atribútu v zázname.

Program, ktorý vykonáva úlohu extrakcie informácií sa nazýva *extraktor* alebo aj *wrapper*. Proces generácie wrapperov (*wrapper induction*) využíva množstvo softvérových nástrojov, ktoré sa venujú extrakcii informácií na webe. Wrapper zvyčajne vykonáva procedúru porovnávania a zhodnosti vzorov, ktorá závisí na množine extrakčných pravidiel. Navrhnutie univerzálneho wrappera, ktorý poskytuje čo najväčšiu

efektivitu na čo najrozličnejšej množine dokumentov je stále predmetom ďalších výskumov.

Proces extrakcie informácií z webových zdrojov sa líši v tom, že vstupom sú zvyčajne HTML dokumenty buď generované na serveri alebo pripravené používateľmi. Z pohľadu významu sú tieto stránky považované za čiastočne štruktúrovaný zdroj, keďže na rozdiel od XML neexistuje presná XML schéma alebo iné pravidlá popisujúce dáta.

Vzhľadom na cieľ extrakcie môžeme rozdeliť HTML wrappery na úrovne: záznamu, stránky alebo celého súboru stránok tvoriace webovú prezentáciu (*webový portál - website*). Záznamy zvyčajne tvorí niekoľko opakujúcich sa inštancií extrakčného cieľa na stránke. Wrappery na úrovni stránky zase pracujú s údajmi, ktoré sú obsiahnuté na celej webovej stránke a wrappery extrahujúce údaje z celej webovej prezentácie môžu vytvárať objekty zberom údajov z viacerých stránok webovej prezentácie.

Je niekoľko spôsobov ako popísať ciele extrakcie na stránke. Najbežnejšou štruktúrou je hierarchický strom, kde sú listy jednoduché typy (reťazec, číslo) a vnútorné uzly tvoria zložené typy objektov. Dátový objekt môže mať jednoduchú alebo vnorenú štruktúru. Pri objekte s jednoduchou štruktúrou uvažujeme len koreň stromu a listy a naopak, pri vnorenom objekte vyžadujeme minimálne koreň a aspoň jednu úroveň vnútorných uzlov. Keďže štruktúra webových stránok je ľudsky čitateľná (je tvorená značkovacím jazykom) je možné vizuálne oddeliť elementy zoznamu alebo elementy usporiadanej k-tice. Avšak formát atribútov dátového objektu môže obsahovať viacero variácií [16]:

- Atribút môže mať žiadnu alebo viacero hodnôt v dátovom objekte. Ak nemá žiadnu hodnotu, ide o *chýbajúci* atribút; ak má viac ako jednu hodnotu, hovoríme o *viacnásobnom* atribúte. Napríklad hodnota atribútu *názov knihy* môže predstavovať viacnásobný atribút a *zľava* zase chýbajúci atribút, keďže nie všetky produkty sú v zľave.

- Množina atribútov $(A_1, A_2, ..., A_k)$ môže mať viacero možných poradí, v prípade ak v rôznej inštancii dátového objektu má atribút A_i rôznu pozíciu. V tomto prípade hovoríme o *viacnásobnom poradí* atribútu. Napríklad produkty, ktoré vznikli pred rokom 2010, ale boli vydané po produktoch, ktoré vznikli neskôr.

- Atribút môže mať rôzne formáty v závislosti od rôznych inštancií dátového objektu. Ak nie je formát atribútu fixný, potrebujeme disjunktné pravidlá, ktoré zovšeobecňujú všetky prípady. Napríklad v online elektronickom obchode cena tovaru môže byť zaznačená rôznymi typmi písma v závislosti od toho, či sa jedná o akciovú alebo regulárnu cenu. Na druhej strane v tabuľkách, kde tag `<TD></TD>` reprezentuje skupinu rôznych atribútov, nevieme určiť presné pravidlá. V týchto prípadoch je kľúčovou informáciou na rozlíšenie ich poradie. Ak sa však medzi atribútmi nachádzajú chýbajúce atribúty alebo atribúty s viacnásobným poradím, je nutné prispôsobiť extrakčné pravidlá.

- Viaceré extrakčné systémy pracujú so vstupnými dokumentami ako s reťazcami tokenov, pretože sú jednoduchšie na spracovanie ako reťazce znakov. Nie vždy je však možné atribút rozdeliť na jednotlivé tokeny. Napríklad kód produktu u distribútora môže pozostávať s viacerých reťazcov určujúcich napríklad odbor a kód produktu (napríklad "PC2011").

3.2 Prehľad niekoľkých extrakčných metód a algoritmov

Pre potreby nášho systému budeme potrebovať rôzne metódy, ktoré pracujú s čo najširším spektrom vstupných dát. V tejto kapitole si zhrnieme prehľad niekoľkých typov, ktorými sa budeme zaoberať.

3.2.1 Extrakcia informácií pomocou regulárnych výrazov

Veľa extrakčných úloh môže byť úspešne vyriešených použitím regulárnych výrazov. Príkladom entít, ktoré sa dajú získať takouto extrakciou sú emailové adresy, čísla kreditných kariet, čísla účtov, telefónne a faxové čísla alebo rôzne geografické názvy. Tieto entity majú niekoľko spoločných kľúčových prvkov, ktoré sa dajú vyjadriť ako štandardný regulárny výraz. Ich použitie nie je vždy jednoduché a vytvorenie vhodného výrazu znamená mať dôkladne preštudovanú oblasť, v ktorej chceme dáta získavať. Regulárne výrazy je možné vopred definovať alebo využiť automatické učiace metódy [17]. Vo všeobecnosti ich prispôsobiteľnosť závisí vo veľkej miere od používateľa.

Využitie regulárnych výrazov v extrakcii informácií z webových stránok je v sú - časnosti veľmi rozšírené a existuje mnoho metód a algoritmov, ktoré ich používajú. Za zmienku stojí *ReLIE (Regular expression Learning for Information Extraction)* [18], ktorý využíva učiace mechanizmy pre rozlišovanie medzi pozitívnymi a negatívnymi výstupmi extrakcie a aplikuje niekoľko disjunktných transformácií pre spresnenie regulárneho výrazu. Obsahuje tiež niekoľko vylepšení akými sú: slovníky s negatívnymi výrazmi, *pažravé* (*greedy*) heuristiky pre napĺňanie slovníkov obsahujúce negatívne výrazy, silnejšie obmedzenie pravidiel pre triedy znakov a kvantifikátory. Vyhľadávanie v ReLIE pracuje na princípe horolezeckého algoritmu (*Hill climbing algorithm)*, kedy sa v každej iterácii aplikuje niekoľko transformácií pre získanie kandidátskej množiny regulárnych výrazov. Z tej sa následne vyberie regulárny výraz s maximálnou hodnotou objektívnej funkcie na tréningovej množine. Takto je možné prispôsobovať generované regulárne výrazy, aby vedeli extrahovať významovo rovnaké informácie s rozličným formátovaním.

Z ďalších algoritmov a hotových riešení stojí za zmienku *ONTEA* [19], ktorá bola bližšie rozoberaná v bakalárskej práci. ONTEA (**ON**tology based **TE**xt **A**nnotation) je nástroj na poloautomatickú extrakciu dát z neštruktúrovaných textov pomocou regulárnych výrazov. Tento nástroj najprv analyzuje dokument alebo text pomocou vzorov regulárnych výrazov a deteguje ekvivalentné sémantické elementy podľa vopred definovanej doménovej ontológie. Tieto vzory môžu byť preddefinované a používané na niekoľko druhov vstupných dát, ale pre zabezpečenie vysokej efektivity je potrebné pred každým spustením vytvoriť nové vzory. Na reprezentáciu vzorov slúži trieda *Pattern*, ktorej inštancie sú použité na definovanie a identifikáciu vzťahov medzi textom (dokumentom) a jeho sémantickou verziou vzhľadom na doménovú ontológiu.

3.2.2 Extrakcia informácií na základe porovnávania štruktúry dokumentu

Hlavným predmetom záujmu bakalárskej práce bola implementácia porovnávacej metódy. Táto metóda je vysoko efektívna pri spracovaní stránok s rovnakou šablónou, kedy je menený len obsah, ktorý je často nosičom informácií. To platí pri získavaní detailov o produktoch (tzv. *product pages*) z katalógov a elektronických obchodov. Princíp tejto metódy je porovnávanie DOM (*Document Object Model*) stromu HTML, ktorý je reprezentáciou stránky v prehliadači. Algoritmus rekurzívne (alebo iteratívne)

Obr. 3.3: Zobrazenie rozdielnych atribútov vo výpise podrobností produktu.

porovnáva dva stromy a pamätá si absolútne cesty (napríklad *XPath*) k elementom, v ktorých sa nachádza rôzny obsah.

Nevýhody tejto metódy sú zjavné už hneď - čím je porovnávaný obsah menej rozdielny, tým menej informácií zachytí. Tak je to aj v príklade na obr. 3.3, kedy prídeme o kľúčové vlastnosti produktu. Pre čo najvyššiu efektivitu je nutné mať dostatočne veľkú a najmä rozličnú množinu porovnávaných stránok.

Čo je zdanlivou nevýhodou pri získavaní presných informácií môže byť výhodou pri inom type úlohy - porovnávaní produktu. Pri dopyte, kde sa vyžaduje zobrazenie rozdielov (z hľadiska spotrebiteľa je časté, že sa snaží zistiť, ktorý produkt je lepší) je to dokonca žiadané - výstupom sú rozdielne atribúty produktu, ktoré (ak je to možné) sa dajú porovnať. Ďalšie problémy, ktoré vznikajú pri použití tejto metódy sme už opísali v bakalárskej práci a ich riešenie je načrtnuté v kapitole 4.

3.2.3 Extrakcia záznamov na základe porovnávania ciest k elementom

Ďalším typom stránok sú stránky, ktoré obsahujú viacero záznamov rovnakého typu. Jedná sa zvyčajne o stránky elektronických obchodov, ktoré prezentujú tovar (*product list*) alebo katalógy produktov s viacerými záznamami na stránke.

Obr. 3.4: Zobrazenie dátových záznamov (červená farba) v časti dátového regiónu (zelená farba).

Jedným z algoritmov, ktorý sa zaoberá hľadaním záznamov na stránke je algoritmus *Mining Data Records (MDR)* [20]. Ten hľadá rovnako formátované informácie – *dátové záznamy*, v konkrétnej štruktúre údajov – *dátovej oblasti*. Dátový záznam je zvyčajne odvodený od štruktúry informácie na stránke, ktorá sa môže nachádzať buď vo formulári alebo tabuľke a pod. Hlavnými krokmi tohto algoritmu sú: vytvorenie DOM reprezentácie stránky, dolovanie dátových oblastí na stránke na základe vytvoreného stromu a porovnávaním reťazcov, a nakoniec identifikovanie dátových záznamov v každej dátovej oblasti.

Je jednoduchšie najprv nájsť dátovú oblasť, ktorá pozostáva z *kombinácií uzlov*. Tieto kombinácie sú v podstate susediace uzly s rovnakým rodičom. S ich pomocou totiž vieme vylúčiť presah informácie vo viacerých dátových záznamoch. Potom možno definovať *dátovú oblasť* ako kolekciu dvoch a viacerých susediacich kombinácií uzlov, ktoré majú toho istého rodiča, sú rovnakej dĺžky a ich normalizovaná editačná vzdialenosť je menšia ako vopred zadaná hranica.

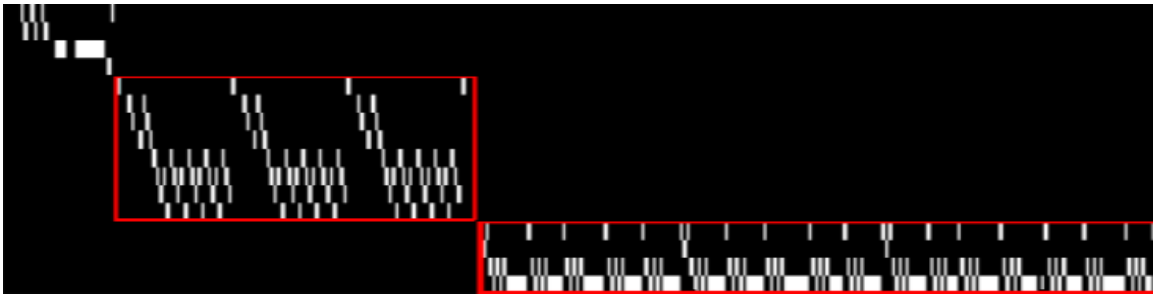
Samotné identifikovanie dátových záznamov vychádza z faktu, že dátová oblasť obsahuje významovo príbuzné objekty. To znamená, že je pomerne jednoduché určiť či kombinácia uzlov obsahuje záznam – stačí sa pozrieť na uzly v rovnakej alebo nižšej úrovni stromu tagov.

Iný prístup využíva analýzu postupnosti ciest k elementom, resp. vyhľadávanie opakovaní v zozname ciest (*Tag path clustering*), ktorý vznikne pri preorder prechode DOM stromu spracovávaného dokumentu. Všeobecne platí, že dátové záznamy, ktoré patria tomu istému regiónu budú mať rovnakú postupnosť ciest k elementom. Znázornenie časti tejto cesty na základe obr. 3.4 môže vyzerať takto:

```
...
87: /html/body/div/div/h4
88: /html/body/div/div/div
89: /html/body/div/div/div/p
90: /html/body/div/div/div/p/strong
91: /html/body/div/div/div/p
92: /html/body/div/div/div/p/strong
93: /html/body/div/div/div/p
94: /html/body/div/div/div/p/strong
95: /html/body/div/div/div/p/i
96: /html/body/div/div/div/p
97: /html/body/div/div/div/p/a
98: /html/body/div/div/h4
99: /html/body/div/div/div
100: /html/body/div/div/div/p
101: /html/body/div/div/div/p/strong
102: /html/body/div/div/div/p
103: /html/body/div/div/div/p/strong
```

```
104: /html/body/div/div/div/p
105: /html/body/div/div/div/p/strong
106: /html/body/div/div/div/p/i
107: /html/body/div/div/div/p
108: /html/body/div/div/div/p/a
109: /html/body/div/div/h4
110: /html/body/div/div/div
...
```

Tieto opakovania je možné nájsť vyhľadávaním klastrov. Jedným z možných riešení [21] je vytvoriť z ciest postupnosť vektorov, ktoré budú dokopy tvoriť vizuálny signál. Formálne *vizuálny signál* s_i je trojica $\langle p_i, S_i, O_i \rangle$, kde p_i je cesta k elementu, S_i je vektor vizuálneho signálu reprezentujúci výskyt pozícií p_i v dokumente a O_i je usporiadaný zoznam, ktorý reprezentuje jednotlivé výskyty. S_i je binárny vektor, kde $S_i(j) = 1$ ak p_i sa vyskytuje v DOM strome na pozícii j, inak $S_i(j) = 0$. Príklad jednej takejto trojice z horeuvedenej postupnosti je $\langle$`/html/body/div/div/div/p`
$, [..., 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, ...], [..., 89, 91, 93, 96, 100, 102, 104, 107, ...]\rangle$.



Obr. 3.5: Vizuálna reprezentácia klastrov.

Autori článku [21] využívajú na nájdenie klastrov algoritmus *normalized cut spectral clustering*. Pri samotnej extrakcii aplikujú heuristiku a rôzne zovšeobecnenia, napríklad dátový záznam by sa mal skladať s minimálne 3 elementov a pod.

3.2.4 Extrakcia záznamov pomocou vizuálneho členenia webovej stránky

Ďalším prístupom ako získať dátové záznamy (alebo iné časti stránky) je použiť vizuálnu reprezentáciu [22]. Tento prístup má viacero výhody – snaží sa napodobniť správanie človeka pri rozoznávaní záznamov na stránke. Ľudský mozog totižto vníma

webovú stránku ako usporiadané bloky, vizuálne oddelené grafickými prvkami (okraje elementov, iné pozadie, horizontálne čiary a pod.).

Aby sme mohli použiť tento prístup, musíme pristupovať k jednotlivým elementom ako k blokom, ktoré majú pevne stanovené vlastnosti, napríklad šírku, výšku, farbu pozadia, farbu okrajov a pod. Získať hodnoty týchto vlastností priamo zo zdrojového kódu nie je možné, keďže bežné webové stránky majú oddelený obsah (formátovaný HTML) a formu (definovanú pomocou kaskádových štýlov). Ak by sme aj získali informáciu o napríklad veľkosti elementu z CSS, tak v prípade, že sa jedná o relatívnu veľkosť nevieme určiť skutočnú veľkosť kým nepreskúmame všetkých rodičov.

Tieto vlastnosti je možné určiť pomocou rozličných nástrojov z DOM stromu renderovanej (t.j. zobrazenej v prehliadači) stránky, ktoré dokážu simulovať správanie sa prehliadača, poprípade pomocou nástrojov, ktoré využívajú priamo jadro prehliadača (a poskytujú tak často presnejšie výsledky).

Samotný algoritmus potom rekurzívne prehľadáva bloky a hľadá elementy, ktoré majú vlastnosti dátového regiónu a dátového záznamu. Určiť hranice dátového regiónu a jeho záznamov môžeme na základe viacerých parametrov, autori článku [22] spomínajú medzi inými:

- *Hranicu na základe názvu elementu.* Niektoré elementy môžeme považovať za separátory, ako napríklad `<HR>` symbolizuje horizontálnu čiaru.
- *Hranicu na základe farby.* Pri kontrastne ladených prvkoch môžeme napríklad pomocou farby pozadia určiť hranice záznamu.
- *Hranicu na základe textu.* Ak element obsahuje viacero textových elementov (napríklad odstavce), tak ho nedelíme na menšie celky.
- *Hranicu na základe veľkosti.* Rovnaká veľkosť elementov môže znamenať, že majú rovnako formátovaný obsah – dátové záznamy.

Jednou z možných implementácií, ktorá využíva detekciu hraníc na základe veľkosti elementov, sa budeme zaoberať v praktickej časti práce.

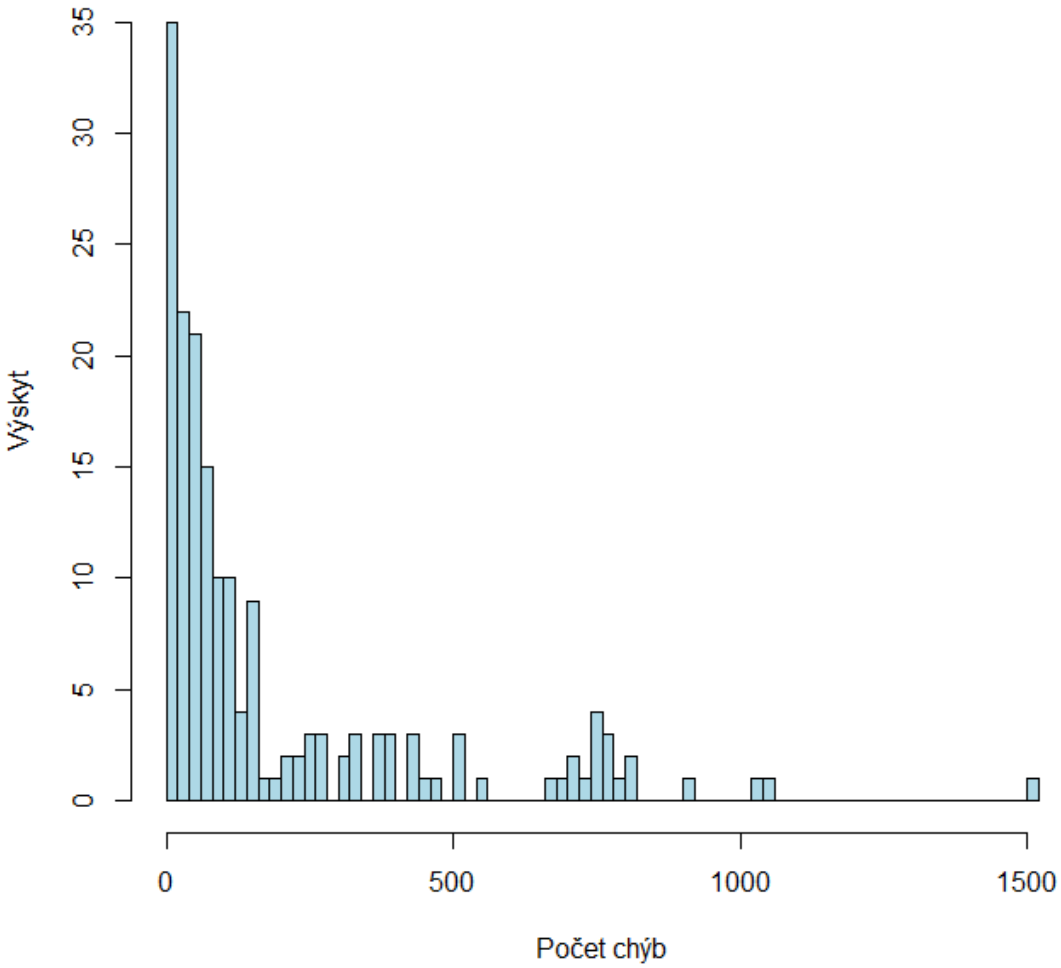
3.3 Ostatné postrehy

Tak ako v bakalárskej práci, aj tu je potrebné spomenúť niektoré z problémov, ktoré sťažujú automatickú kategorizáciu stránok a extrakciu informácií. Jedným z nich je zlá štruktúra HTML dokumentov. Jednak je to nevhodné použitie nesémantických tagov (napríklad nepoužívanie `<em></em>` tagov pri zdôraznení časti textu), ďalej je to nerešpektovanie štandardov pri tvorbe stránok (ktoré neprejdú procesom validácie, napríklad miešanie začiatočných a koncových tagov, používanie zastaraných tagov a pod.) alebo prílišná komplexnosť a veľkosť zdrojového kódu, čo je následkom použitia WYSIWYG editorov. Tieto problémy môžu byť čiastočne odstránené pomocou špeciálnych parserov, ktoré vedia vyriešiť niekoľko základných chýb. Takéto parsery však pracujú tak, že sa snažia dosiahnuť platnosť štandardu aj za cenu straty dát. Ďalším problémom je dostupnosť dynamicky generovaných stránok. Niektoré odkazy majú totižto len dočasnú platnosť, čo môže byť spôsobené nevhodným volaním parametrov alebo zmena parametrov stránky (napríklad ak je URL adresa generovaná nadpisom stránky a ten sa zmení, starý odkaz prestáva platiť).

Z našej štatistiky (obr. 3.6) vyplýva, že výskyt týchto javov je pomerné častý. Z testovaných 207 stránok bolo len 5 dostupných a validných podľa definovaného štandardu, 177 stránok obsahovalo chyby napriek definovanému štandardu a 25 stránok nebolo vôbec dostupných prostredníctvom uloženej URL adresy.

Ďalej je dôležité podotknúť, že extrakčná metóda vo svojej podstate nemusí byť nijak zložitá. Napríklad stránka, ktorá je zaradená do kategórie *Obrázky a fotogalérie* bude spracovaná jednoduchým algoritmom, ktorý na nej nájde všetky obrázky (podľa tagu `<IMG>`) a stiahne ich. Dôležité je zachovať variabilitu extrakčných metód, aby bolo pokryté čo najširšie spektrum prípadov. Tieto extrakčné metódy môžu byť do systému pridané neskôr, podľa potreby, vďaka jednotnému, modulárnemu rozhraniu.

V dnešnej dobe sa predpokladá, že sa tvorcovia stránok snažia zamerať na ich prístupnosť pre strojové spracovanie. Fenoménom doby sa stala *SEO optimalizácia* (search engine optimalisation - optimalizácia pre vyhľadávače), ktorá tlačí na vývojárov, aby sa popri vizuálnom prevedení webovej prezentácie sústredili aj na jej štruktúru. Existuje niekoľko pravidiel, ktoré zvyšujú SEO hodnotu stránky, napríklad vhodné používanie hierarchie nadpisov, dostatok paragrafov a textu, sémantické označovanie hypertextových odkazov a pod.



Obr. 3.6: Histogram výskytu chýb na testovanej množine 207 stránok. Väčšina stránok obsahuje do 100 vážnych chýb, napriek tomu sťažujú ich strojové spracovanie.

Jedným z ďalších problémov prístupnosti pre strojové spracovanie stránok je dynamický obsah generovaný na strane klienta. Tento často zabezpečuje klientský skript vykonávaný v prehliadači, ktorý reaguje na činnosť používateľa webovej stránky. Súčasným trendom je využívanie asynchrónneho prístupu k dátam servera, teda potlačenie koncepcie klient-server.

Takto prezentované dáta nie je možné získať efektívne a je nutné použiť buď API rozhranie služby (ako to je v prípade niekoľkých sociálnych sietí) alebo vytvoriť sofistikovaného prehľadávacieho robota (*crawler*). Takisto ako už bolo spomenuté, k niektorým objektom nie je možné pristupovať vôbec, pretože sú tvorené neprístupnými objektmi ako je to znázornené na obr. 3.7.

V neposlednom rade treba spomenúť aj právne problémy pri extrakcii informácií. Tvorcovia obsahu na Internete často chránia svoj obsah pomocou autorských práv a jeho následné šírenie a používanie vymedzujú v takzvaných *podmienkach používania* (*terms of use*). Niektoré webové portály dokonca chránia svoj obsah proti prehľadávaniu a extrakcii dát aj po technickej stránke, kedy vedia odhadnúť *agenta*, ktorý sa snaží získať obsah dokumentu a nedovolia mu jeho následné zobrazenie. Keďže má táto práca vedecký, nekomerčný charakter, vystačíme si s ukladaním spätných referencií na stránky, ktoré sú pôvodom získaných informácií.

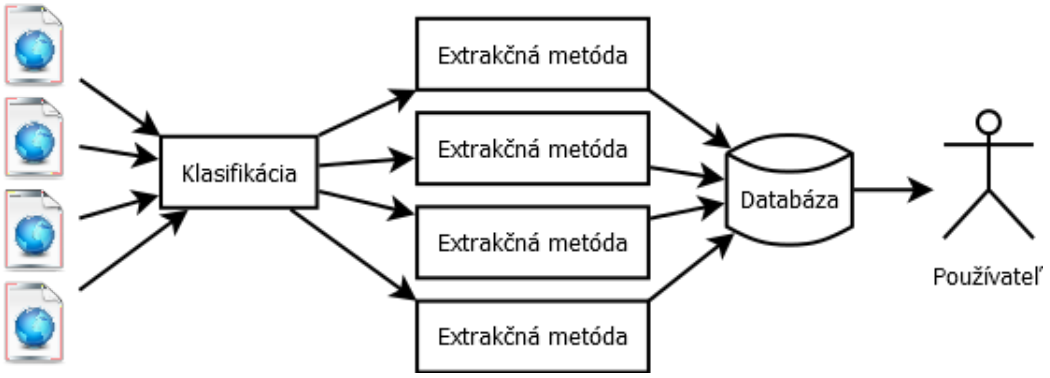
Obr. 3.7: Stránka vytvorená v programe Adobe Flash. Jej obsah tvorí binárny, proprietárny objekt. Prezentáciu obsahu zabezpečuje špeciálny zásuvný modul do prehliadača.

Kapitola 4

Riešenie hlavných problémov

Ako už bolo spomenuté, v práci budeme musieť riešiť niekoľko problémov a úloh potrebných k úspešnému dosiahnutiu cieľa práce. Stručne by sa dali zhrnúť do týchto bodov:

- **1. krok:** Získanie zdrojov
- **2. krok:** Pridelenie kategórií (ručne, učiaci algoritmus)
- **3. krok:** Extrakcia dát na základe pridelenej kategórie
- **4. krok:** Uloženie získaných údajov do databázy
- **5. krok:** Prezentácia a ohodnotenie výsledkov procesu extrakcie



Obr. 4.8: Grafické znázornenie procesu získavania informácií.

Prvé tri kroky sú ukončené a ich prehľad je stručne popísaný v tejto kapitole. Kroky 4 a 5 sú z väčšej časti implementačné, preto budú v tejto kapitole spomenuté okrajovo a ďalej sa im budeme venovať v kapitolách *Návrh riešenia* a *Implementácia systému*.

4.1 Získanie zdrojov

Väčšina úloh z oblasti dataminingu a extrakcie informácií závisí od kvalitného zdroja dát, s ktorým je možné pracovať. V našej práci je zdrojom dát webová stránka, respektíve množina webových stránok. Umiestnenie webovej stránky na internete je definované pomocou *URL adresy*, z čoho vyplýva, že vyhovujúcim vstupom pre systém bude zoznam URL adries. Získať adresy URL je možné niekoľkými postupmi, od manuálneho zberu až cez automatické prehľadávacie roboty (*web crawlers*). Ďalším spôsobom je využiť už existujúce riešenia tretích strán, ktoré sa zaoberajú prehľadávaním a indexovaním webu. V tejto práci budeme na získavanie URL adries používať riešenie spoločnosti Google, ktoré poskytuje API[1] pre službu vyhľadávania vo webových stránkach[2].

Služba pracuje na základe technológie REST[3] (*Representational State Transfer*), ktorá odpovedá na dopyty vo forme JSON objektov alebo XML súborov vo formáte ATOM. Ako spracovávaný výstupný formát sme si zvolili objekty typu JSON (*JavaScript Object Notation*), hlavne pre jeho všestrannosť, vysokú podporu v majorite programovacích jazykov a v neposlednom rade pre jeho jednoduchosť a efektívnosť, čo znižuje komunikačnú zložitosť a veľkosť prenášaných dát. Odpoveď obsahuje - okrem URL odkazu aj nadpis stránky, časť obsahu, približný počet výsledkov hľadania, zoznam odkazov na ďalšie strany vyhľadávania (API volania). Preto je jednoduché získať metainformácie už v prvotnej fáze. Príklad takéhoto API volania môže byť nasledovný:

```
https://www.googleapis.com/customsearch/v1?
v=1.0
&q=KEYWORD
&start=START
&num=COUNT
&key=KEY
```

Hodnota *q* vyjadruje hľadaný reťazec `KEYWORD`, *start* číslo aktuálnej podstránky vyhľadávania, *num* počet objektov v zozname a *key* súkromný kľúč, ktorý slúži ako identifikátor účastníka API volania. Reťazec `KEYWORD` nemusí obsahovať len slová,

[1] *Application programming interface*

[2] `http://code.google.com/apis/customsearch/v1/overview.html`

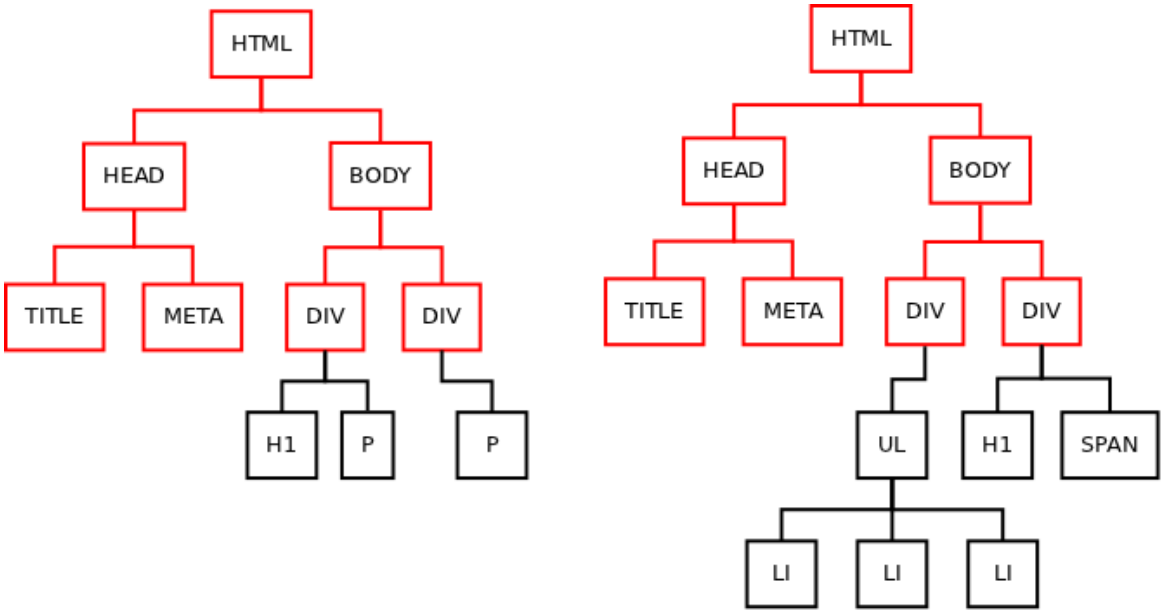
[3] `http://www.infoq.com/articles/rest-introduction`

môže pozostávať aj zo špeciálnych operátorov vyhľadávania, ktoré Google používať, napríklad:

```
"lenovo thinkpad" -t40 site:bestbuy.com
```

Významná je aj možnosť obmedziť vyhľadávanie na určitú doménu (webový portál), v našom príklade je to obchod s elektronikou `bestbuy.com`. Pri vyhľadávaní produktu, respektíve značky v elektronickom obchode, môžeme dostať v závislosti od ponuky niekoľko výsledkov, ktoré majú spoločné prvky. Dôvodom je použitý redakčný systém, ktorý generuje výsledky na základe dopytov používateľa - v našom prípade vyhľadávača. Tieto výsledky majú zvyčajne podobnú štruktúru a líšia sa len v špecifických vlastnostiach (parametre produktu, špeciálne zľavy a ceny, a pod.).

Pri vyhľadávaní sa môžu vyskytnúť aj iné stránky, ktoré nesúvisia s produktom, napríklad informácie o obchode, spôsobe doručenia, akciách a pod. V elektronických obchodoch (a množstvách iných portáloch) je však hlavným obsahom práve katalóg s produktmi, a preto by bolo vhodné vedieť odlíšiť, či stránka získaná vo vyhľadávaní patrí do majoritnej skupiny stránok alebo nie.



Obr. 4.9: Spoločná šablóna pre dva rôzne DOM stromy dokumentu.

Jedným z možných riešení je vytvoriť *šablónu*, ktorá by reprezentovala majoritu generovaných stránok. Šablóna by pozostávala z kostry DOM stromu viacerých stránok. Algoritmus na získanie kostry by bol nasledovný:

1: Získanie stránok s rovnakou štruktúrou
2: Vytvorenie šablóny na základe porovnávania DOM štruktúry
3: Porovnanie novozískaných stránok so šablónou na základe *tree edit distance*[4] (editačnej vzdialenosti v DOM strome)

Prvý krok je možné realizovať buď manuálne alebo pomocou vstupnej množiny stránok na základe porovnávania editačnej vzdialenosti medzi dokumentmi – bude to najpočetnejšia množina dokumentov s editačnou vzdialenosťou menšou ako konštanta. Druhý krok bude pozostávať z prechádzania vstupnej množiny a porovnávania dvoch dokumentov medzi sebou rekurzívne. Zjednodušene môžeme danú procedúru zapísať takto:

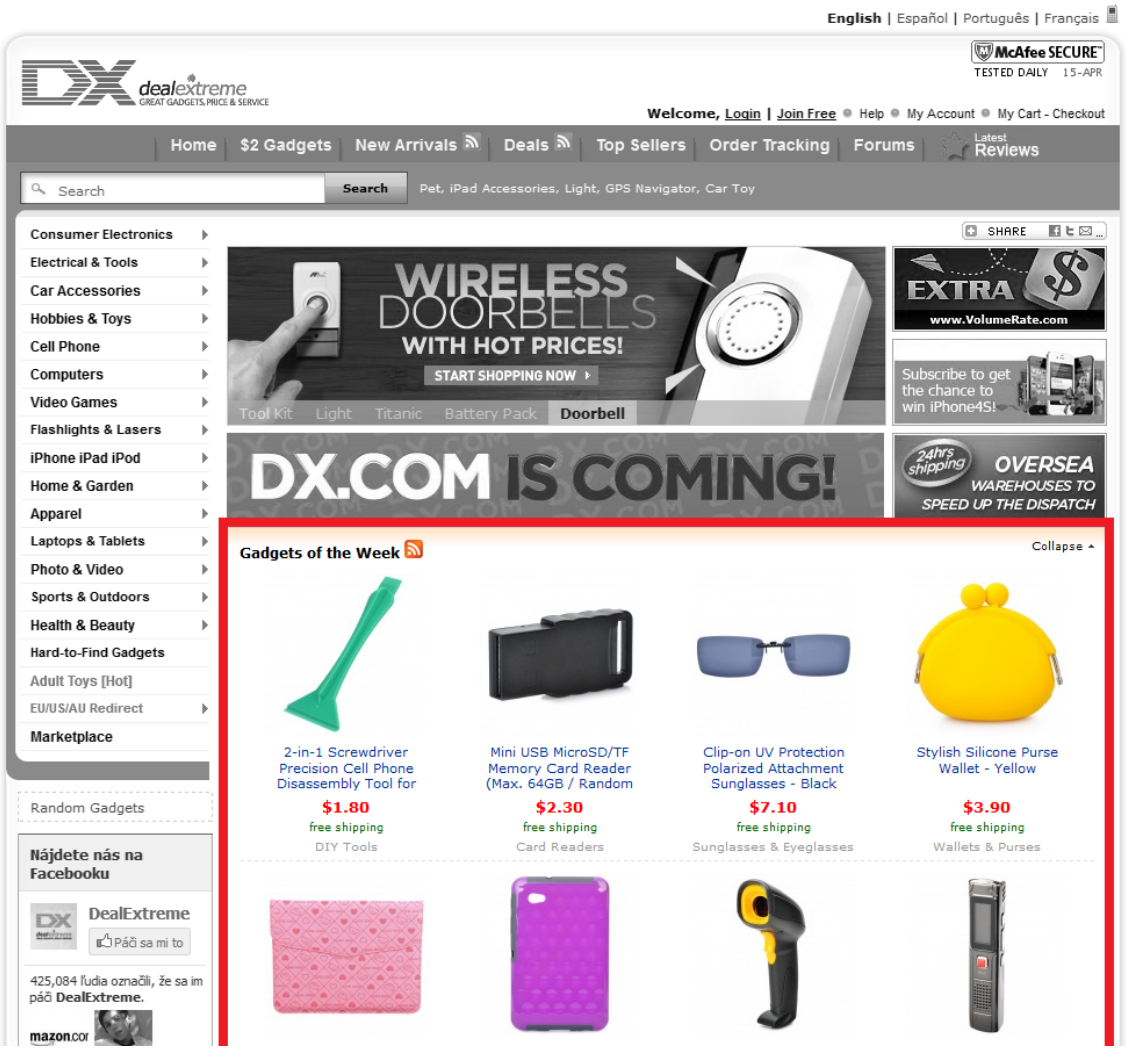
Algorithm 1 createTemplate(d_1,d_2)

1: $Template \leftarrow emptyDocument$
2: **if** $d_1.nodeCount = d_2.nodeCount$ **then**
3: **while** $d_1.hasNextChild$ **or** $d_2.hasNextChild$ **do**
4: $c_1 \leftarrow d_1.nextChild$
5: $c_2 \leftarrow d_2.nextChild$
6: **if** $c_1.name = c_2.name$ **then**
7: $Template.append(createTemplate(d_1.nextChild, d_2.nextChild)$
8: **end if**
9: **end while**
10: **end if**
11: **return** $Template$

Algoritmus sa teda sústredí len na tie vetvy stromu DOM dokumentu, ktoré majú rovnaký počet potomkov a do šablóny pridáva len tie, ktoré sa zhodujú v názve (teda tagu), ako je to zobrazené na obr. 4.9. Výhodou je to, že v neskorších procesoch budeme môcť na základe tejto šablóny získať časti s rozličnou štruktúrou, čo nám uľahčí zistiť hlavnú obsahovú časť stránky. Túto významovo dôležitú časť budeme nazývať *pagelet*. Časti stránky s rozličnou štruktúrou často popisujú hlavný obsah, napríklad články spravodajského portálu majú rozličný počet slov, nadpisov, odstavcov, obrázkov a pod. Šablónu je možné renovovať alebo upravovať v čase ďalšími

[4] `http://grfia.dlsi.ua.es/ml/algorithms/references/editsurvey_bille.pdf`

stránkami, čo zvyšuje jej presnosť. Bohužiaľ v prípade výsledkov vyhľadávaní na internete sú zvyčajne rovnaké výsledky zoskupené a zobrazuje sa len prvý, najrelevantnejší. Takto je zachovaná rôznorodosť výsledkov vyhľadávania, no v našom prípade nebude viesť k úspešnému vytvoreniu šablóny. Preto je horeuvedený postup vhodný len vtedy, keď máme určenú množinu stránok s rovnakou štruktúrou (napríklad manuálne alebo ak je vyhľadávanie špecifikované pre konkrétnu doménu).



Obr. 4.10: Rozpoznanie pageletu na stránke môže byť niekedy netriviálna záležitosť. Na obrázku je príklad s horizontálnym aj vertikálnym menu, časťou s reklamami a banermi. Požadovaná obsahová časť je ohraničená hrubou čiarou.

Ako sme však uviedli, získanie *pageletu*, resp. hlavnej obsahovej časti by spresnilo výsledky extrakcie. Ak chceme získať pagelet pre jednu konkrétnu stránku, môžeme použiť postup spomínaný v 3.2.4, teda vizuálne vyčleniť hlavnú obsahovú časť stránky. Získanie pageletu je veľmi dôležité, pretože ho budeme využívať pri procese klasifikácie

(získanie obsahovej časti, odhadnutie štruktúry obsahu), ale aj pri procese extrakcie (jednoduchšie nájdenie dátových záznamov, extrakcia informácií len z obsahovej časti).

Navrhovaný algoritmus bude hľadať cestu XPath, ktorá vedie k hlavnej obsahovej časti. Bude postupovať od koreňa dokumentu (`HTML`) rekurzívne cez všetkých potomkov a bude si vyberať vždy element s najväčšou veľkosťou (súčin zobrazovanej výšky a šírky), pričom bude stanovených niekoľko podmienok vyhovujúcich rekurzii:

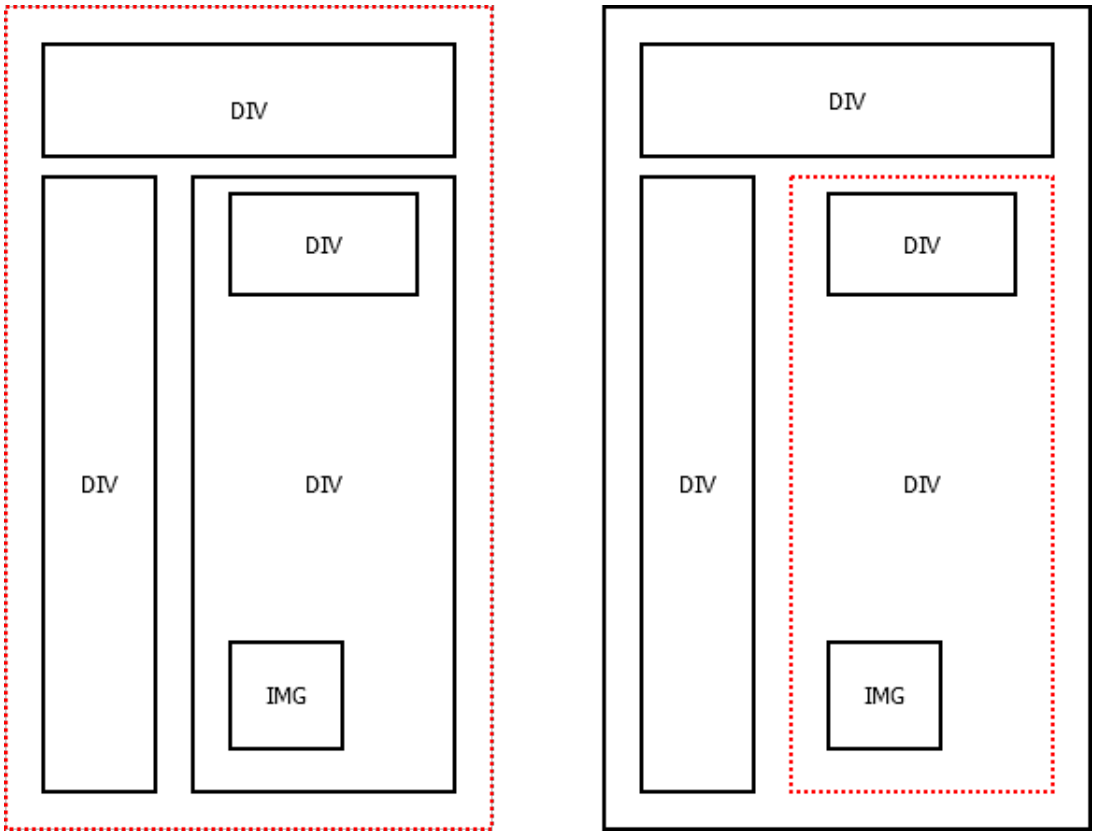
- *veľkosť najväčšieho elementu s rodičom X musí byť väčšia ako medián veľkosti všetkých elementov s rodičom X*,
- *veľkosť najväčšieho elementu musí tvoriť aspoň k-percent plochy rodiča X.*

Algorithm 2 getPageletNode(*subElements*,*depth*),*parentElement*

```
1:  maxElement ← maxSize(subElements)
2:  ratios ← calculateContentToParentRatio(foreach subElements)
3:  median ← median(ratios)
4:  if calculateContentToParentRatio(maxElement) >= median
    and calculateContentToParentRatio(maxElement) > k then
5:    getPageletNode(maxElement.subElements(), depth + 1, maxElement)
6:  else
7:    if calculateContentToParentRatio(maxElement) <= k then
8:      return maxElement.getParent()
9:    else
10:     return maxElement
11:   end if
12: end if
13: return parentElement
```

Ako je možné vidieť, algoritmus pracuje s niekoľkými pomocnými metódami a premennými. Premenná k určuje hodnotu, ktorú má tvoriť pagelet v pomere k svojmu rodičovi. Metóda *maxSize* vráti element s maximálnou veľkosťou (plochou), *calculateContentToParentRatio* vyráta pomer veľkosti elementu k jeho rodičovi ($<= 1$). Ďalšie metódy sú intuitívne, štandardné pre DOM model. Algoritmus nám vráti element, z ktorého však už vieme jednoducho získať cestu XPath.

Princíp tohto prístupu je znázornený aj na obrázku 4.11. V prvom kroku začíname koreňovým elementom stromu – v prípade HTML stránok sa obsah začína v tagu `BODY`. V rámci tohto elementu hľadáme najväčší element a testujeme, či jeho veľkosť v pomere k rodičovi (telu HTML stránky) je väčšia ako k. Ak áno, vnoríme sa do tohto elementu a opäť hľadáme najväčšieho potomka. V tomto (zjednodušenom) prípade majú potomkovia k rodičovi pomer menší ako k, čo znamená, že daný element je hľadaný pagelet.



Obr. 4.11: Nájdenie hlavnej obsahovej časti. Začíname celým telom dokumentu (vľavo) a vnárame sa až kým nenájdeme pagelet (vpravo).

4.2 Pridelenie kategórií

Kategorizácia bude pozostávať z dvoch hlavných krokov:

- pridelenie kategórie na základe štruktúry,
- pridelenie kategórie na základe analýzy obsahovej časti (textová analýza).

Na základe týchto dvoch klasifikácií budeme môcť vytvoriť klasifikačný vektor, ktorý bude charakterizovať dokument a bude určovať, ktorú extrakčnú metódu systém použije.

4.2.1 Pridelenie kategórie na základe štruktúry

Pri tomto type kategorizácie budeme analyzovať štruktúru dokumentu, pričom budeme deliť stránky do dvoch kategórií - **štruktúrované** a **textové**, resp. neštruktúrované. Už z popisu je jasné, že štruktúrované stránky budú obsahovať viac obsahu formátovaného do významovo oddelených častí, napríklad tabuliek, blokov, zoznamov a pod. Naopak, textové stránky budú obsahovať viac súvislého textu, to znamená, že budú mať väčšiu početnosť slov v relevantných tagoch akými sú odstavce, bloky a pod.

Budeme teda využívať pozorovanie: *Štruktúrované dokumenty majú vyššiu početnosť používania tagov definujúcich štruktúru (`TABLE`, `TR`, `TD`, `TH`, `DIV`, `UL`, `OL`, `LI` a pod.) než textové dokumenty, zatiaľ čo textové dokumenty budú mať vyššiu početnosť slov.* Už zo začiatku je jasné, že nie je možné vždy presne určiť do akej kategórie stránka patrí, keďže množstvo portálov sa snaží poskytovať zmiešaný obsah. Sem patria napríklad internetové obchody, ktoré okrem katalógu produktov poskytujú aj ich recenzie.

Tento problém je možné čiastočne riešiť tak, že sa stránka bude členiť na úseky podľa štruktúry a každý sa bude spracovávať samostatne. Ďalším riešením je vytvoriť dvojrozmerný vektor, ktorý bude vyjadrovať príslušnosť k danej kategórii. V implementačnej časti budeme pracovať s dvoma deleniami: v prvom prípade budeme deliť stránky na štruktúrované a textové, v druhom prípade kde sa budeme snažiť docieliť vyššiu funkcionalitu budeme deliť stránky na *detail produktu*, *recenzia produktu* a *zoznam produktov*.

Takúto klasifikáciu je možné realizovať za pomoci neurónových sietí. Ak máme n sledovaných znakov, ktoré chceme klasifikovať do k tried, môžeme zostaviť sieť z troch vrstiev: prvá vrstva bude mať n neurónov, druhá $n+1$ neurónov a tretia, výstupná vrstva bude mať k neurónov. Pre každú z k kategórií potom vieme pripraviť tréningovú

množinu, pomocou ktorej naučíme neurónovú sieť rozpoznávať jednotlivé triedy. Tento prístup má však nevýhodu v tom, že je veľmi citlivý na vybraný typ siete, jej parametre a v neposlednom rade je málo odolný voči chybným prvkom v tréningovej množine, čo komplikuje proces učenia.

V tejto práci použijeme prístup opísaný v [12]. Vieme, že určitý typ kategórie bude obsahovať znaky vo vyššej početnosti ako iné, napríklad textová stránka bude obsahovať viacej odsekov a väčší počet slov. Preto pre klasifikáciu využijeme *apriori* prístup kde predpokladáme, že niektorá vlastnosť je významovo hodnotnejšia pre jednu triedu než pre tie ostatné. Konkrétny pomer príslušnosti k danej klasifikovanej triede je možné získať na základe učiaceho procesu nad klasifikovanou množinou webových stránok. Budeme využívať hodnoty reálnych čísel od -1 po 1, kde -1 znamená, že daná vlastnosť je neprípustná pre danú kategóriu a naopak 1 znamená, že daná vlastnosť je povinná pre danú kategóriu.

Samotný proces klasifikácie bude prebiehať tak, že najprv získame hodnoty jednotlivých vlastností pre danú webovú stránku. Pre tieto účely si definujeme pomocnú procedúru, ktorá na základe zadaného pravidla (cesta XPath, počet slov a pod.) vráti hodnotu danej vlastnosti. Z týchto vlastností nám vznikne vektor $\Phi = (\varphi_1, ..., \varphi_n)$, kde n je počet skúmaných vlastností. $\Omega_{i,j}$ bude reprezentovať hodnoty váh jednotlivých vlastností pre danú kategóriu, kde $i = 1, ..., n$ bude udávať index konkrétnej vlastnosti a $j = 1, ..., k$ bude udávať index kategórie, respektíve klasifikovanej triedy.

$$\Omega = \begin{bmatrix} \omega_{1,1} & \cdots & \omega_{1,k} \\ \vdots & \ddots & \vdots \\ \omega_{n,1} & \cdots & \omega_{n,k} \end{bmatrix} \Phi = \begin{bmatrix} \varphi_1 \\ \vdots \\ \varphi_n \end{bmatrix} \Theta = \Omega \cdot \Phi = \begin{bmatrix} \theta_1 & \cdots & \theta_k \end{bmatrix} \tag{4.1}$$

Výsledné hodnoty príslušnosti k danej kategórii $\Theta = (\theta_1, ..., \theta_k)$ získame ako súčin matíc $\Omega \cdot \Phi$ (4.1), ktoré môžeme reprezentovať buď tak, že za výslednú kategóriu vezmeme maximálnu hodnotu z $(\theta_1, ..., \theta_k)$ alebo budeme pracovať s celým vektorom.

Pre náš konkrétny prípad, kedy pracujeme s dvomi kategóriami, budeme môcť o stránke povedať či patrí do textovej alebo do štruktúrovanej triedy klasifikácie na základe štruktúry.

Na príklade v rovnici (4.2) vidíme konkrétny príklad na reálnych hodnotách. $\Omega_{i,1}$ sú v tomto prípade hodnoty pre prvú kategóriu (štruktúrovaná) a $\Omega_{i,2}$ sú hodnoty pre druhú kategóriu (textová). Tieto hodnoty boli získané učením na vopred (manuálne) kategorizovaných webových stránkach. Φ je v tomto prípade vektor vlastností, ktorý predstavuje početnosti výskytu jednotlivých tagov, ako aj početnosť slov v tagoch, ktoré formátujú text (`P`, `SPAN`, `FONT`, a pod.). Výsledkom je dvojrozmerný vektor, z neho získame výsledok ako *maximum*, čiže v tomto prípade bude stránka zaradená do kategórie štruktúrovaných stránok.

$$\Omega = \begin{bmatrix} 0.6 & -0.5 \\ 0.6 & -0.5 \\ 0.2 & 0.2 \\ -0.2 & 0.0 \\ -0.2 & -0.8 \\ -1.0 & -0.8 \\ 0.0 & -0.8 \\ -0.4 & -0.6 \\ 1.0 & -0.8 \\ 0.0 & -0.6 \\ -0.2 & 0.4 \\ 0.8 & -0.6 \\ 0.6 & 1.0 \end{bmatrix} \quad \Phi = \begin{bmatrix} \texttt{DD} = & 0 \\ \texttt{DIV} = & 121 \\ \texttt{DL} = & 0 \\ \texttt{DT} = & 0 \\ \texttt{FORM} = & 27 \\ \texttt{LI} = & 148 \\ \texttt{OL} = & 0 \\ \texttt{P} = & 0 \\ \texttt{TABLE} = & 83 \\ \texttt{TD} = & 339 \\ \texttt{TR} = & 152 \\ \texttt{UL} = & 21 \\ words = & 124 \end{bmatrix} \quad \Theta = \Omega \cdot \Phi = \begin{bmatrix} 62.90 & -298.09 \end{bmatrix} \tag{4.2}$$

4.2.2 Pridelenie kategórie na základe analýzy obsahovej časti

Kategorizácia textu má zmysel hlavne v tom prípade, keď sú požadované informácie zakomponované v textovej časti stránky. Napríklad ak si chce používateľ kúpiť nový smartfón, kategória *mobilné telefóny* nám pomôže určiť metódu, ktorá ich dokáže vyhľadať, poprípade spresní výsledok takejto metódy. Tu je však dôležité správne určiť hlavnú obsahovú (textovú) časť stránky (*pagelet*).

Keď už máme získanú textovú časť, môžeme ju analyzovať na základe niektorého z používaných algoritmov pre klasifikáciu textu. Existuje niekoľko riešení, no všetky sa približne zhodujú v počiatočnom kroku - predspracovaní textu. Keďže budeme

pracovať s anglickými stránkami[5], použijeme niekoľko štandardných postupov:

- odstránenie tagov, poprípade získanie alternatívneho textu pri netextových prvkoch,
- vyhľadanie a odstránenie koncových slov (*stop words*), teda často sa opakujúcich slov, ktoré nedefinujú význam dokumentu (medzi takéto slová patria spojky, predložky, častice a pod., napríklad *the, a, for, with,* a pod.),
- tokenizovanie textu na slová
- frekvenčná analýza slov, a pod.

Samotnú klasifikáciu textu bude vykonávať *naivný Bayesov klasifikátor*, ktorý využíva pravdepodobnostný Bayesov model:

$$\text{posteriórna pravdepodobnosť} = \frac{\text{apriórna pravdepodobnosť} \times \text{vierohodnosť}}{\text{dôkaz}}.$$

Presnejšie

$$p(K_j|S) = \frac{p(S|K_j) \cdot p(K_j)}{\sum_{i=1}^{n} p(S|K_i) \cdot p(K_i)},$$

kde S je slovo (resp. množina slov), ktoré chceme klasifikovať K_j je kategória, do ktorej chceme dané slovo zaradiť $j \in 1, ..., n, n$ je počet kategórií. Sumu

$$\sum_{i=1}^{n} p(S|K_i) \cdot p(K_i),$$

je možné rozpísať aj ako

$$p(S|K_j) \cdot p(K_j) + \sum_{i=1, i \neq j}^{n} p(S|K_i) \cdot p(K_i),$$

čo znamená, že sa snažíme otestovať množinu slov získaných zo stránky na kategóriách, pričom postupne prechádzame všetky kategórie ($1 \leq j \leq n$) a pre každú získame hodnotu pravdepodobnosti, kde množina slov zodpovedá danej kategórii.

[5]Angličtina ako taká má jednoduchšie gramatické pravidlá ako slovenčina, napríklad nepoužíva skloňovanie, prípony sú rovnaké pre každý pád, to znamená, že sa nemusíme zaoberať hľadaním základu slova.

Ako príklad ukážeme zaradenie do jednej z kategórií v rámci katalógu produktov elektronického obchodu. Členenie katalógu je nasledovné[6]:

- Appliances (Domáce spotrebiče)
- Audio & MP3
- Camera & Camcorder Accessories (Príslušenstvo k fotoaparátom)
- Cameras & Camcorders (Fotoaparáty a kamery)
- Car, Marine & GPS (Auto a lodné navigácie)
- Computers & Tablets (Počítače a tablety)
- Mobile Phones (Mobilné telefóny)
- Movies & Music (Film a hudba)
- TV & Video
- Video Games & Gadgets (Videohry a elektronické hračky)
- Washers & Dryers (Práčky a sušičky)

Konkrétne

$$p(\texttt{Appliances}|'display') = \\ = \frac{p('display'|\texttt{Appliances}) \cdot p(\texttt{Appliances})}{p('display'|\texttt{Appliances}) \cdot p(\texttt{Appliances}) + \sum_{i=2}^{n} p('display'|K_i) \cdot p(K_i)}$$

nám vráti pravdepodobnosť pre prvú kategóriu a na základe porovnania s ostatnými potom vieme určiť výsledok kategorizácie obsahu.

4.3 Extrakcia informácií

Ako už bolo spomenuté, systém by mal umožňovať výber medzi niekoľkými extrakčnými algoritmami. V bakalárskej práci sme sa venovali najmä *diferenčnej metóde*, kde sme v závere navrhli niekoľko rozšírení, ktoré by umožnili lepšie extrahovať a anotovať získané informácie. Tiež sa sústredíme na získanie dátových záznamov zo štruktúrovanej stránky a extrakciu textových informácií pomocou regulárnych výrazov.

[6]Názvy kategórií ako aj vstupná tréningová množina pre každú kategóriu boli získané z elektronického obchodu `http://www.bestbuy.com`, resp. API rozhrania pre vývojárov *BBYOpen*.

4.3.1 Diferenčná metóda

Jedným z návrhov pre vylepšenie porovnávacieho algoritmu bolo obmedzenie prehľadávacieho priestoru, ktorý sa medzi sebou porovnáva. Podstatná časť informácie sa nachádza v hlavnej časti (*pagelet*), ktorú vieme získať na základe šablóny. Keďže diferenčná metóda predpokladá na vstupe množinu stránok rovnakej štruktúry (teda získanú z rovnakého webového portálu),šablóna sa tak vytvorí automaticky.

Nedostatok základného algoritmu porovnávacej metódy je ten, že neuvažuje extrakciu dát, ktoré sa zhodujú v obsahu. Tieto však často obsahujú popisy dát, ktoré by mohli byť použité pri anotácii. Najčastejším príkladom je tabuľka, ktorá obsahuje v jednom stĺpci popis dát a v druhom samotné hodnoty. Pri tomto type štruktúry je možné získať popis tak, že pri rozdielnej hodnote v jednom stĺpci získame jeho popis z druhého stĺpca (v prípade ak je jeho hodnota rovnaká pre každý vstup, t.z. cestu XPath).

Námetom na ďalší výskum bolo aj upraviť porovnávanie tak, aby pružnejšie reagovala na nerovnakú štruktúru stránok. Niekedy sú totiž nosičom informácie listové elementy DOM stromu, inokedy sú však zaobalené v (*n-1*)-vej úrovni, ako je to napríklad v štruktúrach členených pomocou elementov `DIV`. Jednou z možností je pridanie hranice (*threshold*) vzdialenosti, do ktorej budú porovnávania započítavané. Hranica bude definovaná ako vzdialenosť od listového elementu smerom ku koreňu stromu, vrámci ktorej budú brané do úvahy cesty XPath elementov na porovnanie.

Metóda pre porovnávanie bude síce implementovaná v našom systéme, ale vzhľadom na metodológiu získavania zdrojov (výsledky vyhľadávania Google) bude jej použitie zvážené po intenzívnom testovaní.

4.3.2 Extrakcia dátových záznamov

Cieľom v tomto prípade je získať informácie z opakujúcich sa štruktúr na jednej stránke. V popise problému sme si predstavili niekoľko algoritmov. Algoritmy založené na princípe použitom v [20] nie sú v súčasnosti práve najvhodnejšie - hlavným dôvodom je, že v súčasnosti je členenie stránok zložitejšie a využívajú sa neštandardné prvky (napríklad obrázok, ktorý reprezentuje oddeľovač dát umiestnený pomocou kaskádových štýlov).

Na druhej strane [21] využíva všeobecné data-miningové postupy, ktoré sa snaží aplikovať na problém extrakcie informácií z webových stránok (*web mining*). Napriek tomu je implementácia tohoto algoritmu závislá od presne stanovených pravidiel a heuristík, ktoré nemusia platiť vo všeobecnosti.

Konkrétne, ak sú dátové záznamy v jednom regióne oddelené nepravidelným obsahom (*šumom*), tak sa opakovania nemusia nájsť nikdy, pretože vzdialenosť medzi príslušnými elementmi nie je rovnaká. Tento jav je pozorovateľný najmä pri zoznamoch s produktami, kde sú jednotlivé produkty vizuálne rozlíšené na základe ceny (výpredaj, akcia, novinka a pod.) alebo iných parametrov.

Vhodnejšia voľba je využiť postup spomínaný v kapitole 3.2.4 a navrhnúť algoritmus, ktorý by vedel vhodne určiť hranice dátového regiónu a rozpoznať v ňom dátové záznamy. Podobne ako v prípade získavania pageletu budeme postupovať tak, že jednotlivé oblasti budeme deliť podľa veľkosti.

Obr. 4.12: Vyznačenie dátového regiónu a dátových záznamov. Najprv nájdeme riadkové záznamy, potom (ak existujú) vyčleníme stĺpce v jednotlivých riadkoch.

V prvej fáze nájdeme dátový región, ktorý budeme hľadať na základe pravidla: *Dátový región obsahuje dátové záznamy rovnakej veľkosti.* Samozrejme toto tvrdenie počíta s tým, že v rámci dátového regiónu sa nenachádzajú žiadne iné dáta (šum), preto budeme musieť pridať hranicu, ktorá bude určovať koľko blokov s rovnakou veľkosťou sa musí v oblasti nachádzať, aby bola považovaná za dátový región.

Pri identifikácii jednotlivých záznamov budeme brať ohľad na fakt, že majú (približne) rovnakú štruktúru, resp. zjednodušený fakt, že koreňový element DOM stromu dátového záznamu je rovnaký pre všetky dátové záznamy. Za prvky dátového regiónu budeme považovať tie elementy, ktoré majú rovnakú šírku. Takto získame riadkové dátové záznamy a analogicky môžeme získať z týchto riadkov aj stĺpcové hodnoty.

4.3.3 Extrakcia textových informácií

Ako už bolo spomenuté, použitie regulárnych výrazov je často používaná metóda pri získavaní textových informácií z webových stránok. Cieľ, ktorý sa splní pri použití regulárneho výrazu musí byť vopred jasne definovaný, jeho zovšeobecnenie je často veľmi ťažké a vyžaduje si presnú znalosť problematiky, napríklad *slovenské telefónne čísla, ktoré patria mobilnému operátorovi XYZ s prefixom 0999*, budú reprezentované nasledujúcim regulárnym výrazom:

```
(\+)(421)(\s*)(999)(\s*)([0-9]{3})(\s*)([0-9]{3})
```

Tento výraz zachytí číslo v požadovanom tvare, no v ojedinelom prípade môže zachytiť aj čísla, ktoré nie sú telefónne. Napríklad pri ekonomickom vyjadrení *firma skončila v minulom roku v zisku +421999123456 dolárov* by sme síce dostali číslo v požadovanom tvare, ale zrejme by sme sa naň nedovolali.

V tomto prípade je tiež možné využiť výsledky fázy kategorizácie a pre každú triedu navrhnúť vhodné regulárne výrazy, ktoré budú použité pri extrakcii. Napríklad pre kategóriu *počítače a tablety* budú ako základné vzory použité regulárne výrazy, ktoré získajú informáciu napríklad o frekvencii procesora, veľkosti operačnej pamäte alebo veľkosti dátového úložiska a pod. Tieto regulárne výrazy bude možné dopĺňať a budú ukladané v databáze pre príslušnú kategóriu.

Manuálne písanie regulárnych výrazov je niekedy veľmi zdĺhavé a náročné, preto na ich uľahčenie vznikli internetové knižnice[7] a nástroje, ktoré ponúkajú ich asistované generovanie[8].

4.4 Ostatné

Uloženie získaných údajov do databázy závisí na tom, ako ich chceme ďalej používať. Informácie, ktoré chceme neskôr strojovo spracovávať musia byť vhodne členené a anotované. V našom prípade nás zaujíma skôr extrakčná časť, preto budeme uchovávať získané údaje ako reťazce, ktorým budú pridelené základné metaúdaje ako napríklad:

- URL adresa stránky, z ktorej boli údaje získané,
- kategórie, ktoré boli pridelené stránke,
- extrakčná metóda, pomocou ktorej boli údaje získané,
- typ extrahovaných dát (dátový záznam, text, číselný údaj a pod.),
- generický alebo priradený popis (napr. pri extrahovaní pomocou regulárnych výrazov)
- čas získania informácií, a pod.

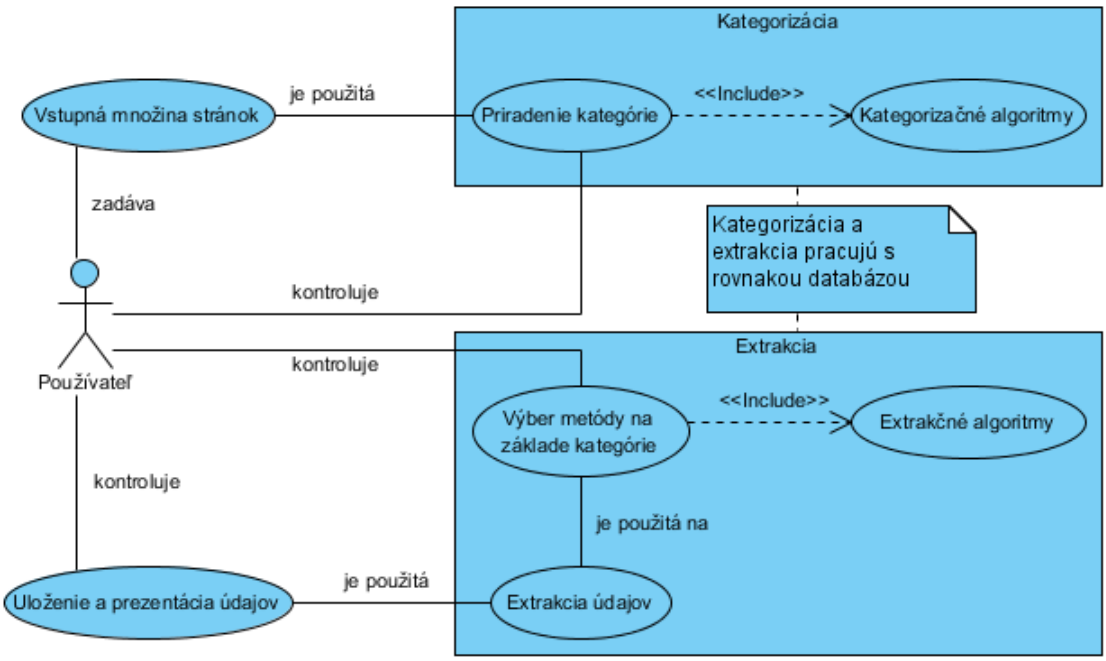
Prezentácia údajov bude realizovaná pomocou webového prehliadača a viac sa ňou budeme zaoberať v kapitole 5.

[7] `http://regexlib.com/`
[8] `http://txt2re.com/`

Kapitola 5

Návrh riešenia

Systém bude rozšírením prototypu navrhnutého v bakalárskej práci, to znamená, že pôjde o webovú aplikáciu, ktorá bude mať samostatné webové rozhranie a bude poskytovať aj webové služby pomocou jednoduchého API rozhrania. Jadro systému sa bude deliť na časť venovanú kategorizácii stránok a časť venovanú extrakcii informácií.



Obr. 5.13: Use case diagram systému.

Prvým krokom v procese dolovania údajov z webu bude získanie vstupnej množiny stránok. Túto množinu systém získa na základe kľúčových slov, ktoré zadal používateľ pri vyhľadávaní vo webovom rozhraní alebo pri API volaní. Tieto kľúčové slová budú použité vo vyhľadávaní prostredníctvom služby Google a jej výsledná množina URL

odkazov bude spracovaná tak, aby bol každý dokument validný podľa štandardov W3C (čo je možné dosiahnuť prostredníctvom HTML parserov a validátorov) a aby stránka vracala HTML dokument (a nie chybové hlásenie spôsobené neprístupnosťou dokumentu).

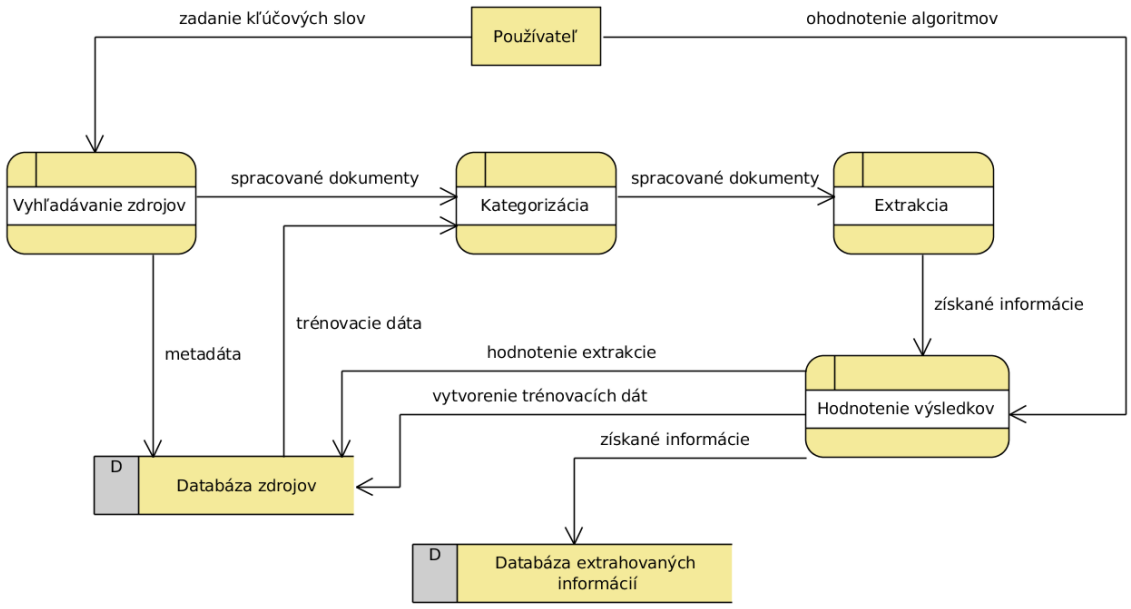
Ďalším krokom bude kategorizácia, ktorá bude na základe implementovaných algoritmov vytvárať vektor kategórií, do ktorých stránka spadá. Tento proces by mal byť v pokročilých fázach projektu plne automatizovaný, no zo začiatku bude potrebné pripraviť tréningovú množinu, ktorá bude vhodne reprezentovať požadované triedy. Pre učiaci algoritmus klasifikácie štruktúry bude pripravená vstupná množina o veľkosti najmenej 200 rôznych stránok. Za rôzne považujeme stránky, ktoré nepochádzajú z toho istého portálu, respektíve majú na prvý pohľad rôznu štruktúru.

Klasifikácia textu bude pozostávať z pripravenej množiny kľúčových slov, ktoré budú zbierané z webových stránok použitím predspracovania spomenutého v 4.2. Zoznam kategórií by mal byť modifikovateľný s možnosťou pridávania nových kategórií s následnou úpravou kategorizačného algoritmu.

Nasledujúcim krokom bude extrakcia dát, ktorá bude prebiehať automaticky na základe priradenej kategórie. Na začiatku nebude možné priradiť extrakčný algoritmus žiadnej z kategórií, preto je potrebné najprv naučiť systém manuálnym výberom tak, že sa vyberú najvhodnejší kandidáti. Tento proces bude prebiehať v jednoduchom používateľskom rozhraní, v ktorom bude náhodne zvolená stránka a formulár pre výber vhodnej metódy.

Extrahované údaje budú potom spracované a uložené do databázy. Potom sa používateľovi zobrazí stránka s náhľadom aktuálne spracovávaného dokumentu a informácia o tom, ktorá metóda bola použitá, respektíve jej výstup. Používateľ bude môcť vybrať, ktorá metóda vrátila najlepší výsledok na základe porovnania výsledkov a obsahu webovej stránky.

Ako už bolo spomenuté, jedným z cieľov je navrhnúť učiaci proces pre výber vhodných extrakčných algoritmov. Práve vďaka týmto hodnoteniam používateľa bude možné získať percentuálnu úspešnosť extrakcie pre zvolenú kategóriu, čo bude zohľadnené v ďalšom výbere. Systém bude pracovať s dvomi databázami, ktoré budú navzájom prepojené.



Obr. 5.14: Distribúcia dát pri učiacom procese systému.

Prvá databáza bude obsahovať popisné informácie pre vstupné dáta (*metadáta*), kde budú uchované informácie o získaných vstupných stránkach, portáloch, kategóriách a extrakčných algoritmoch k nim priradeným. Tiež bude uchovávať tréningové množiny získané hodnotením používateľov - výstup, ktorý bude jednoznačne ohodnotený ako najlepší sa vloží do tréningovej množiny. Druhá databáza bude slúžiť pre uchovávanie získaných dát podľa vopred predpísanej štruktúry tak, aby ich bolo možné ďalej strojovo spracovať.

Prezentácia získaných údajov v pokročilých fázach projektu bude spočívať v zobrazení výstupov jednotlivých algoritmov, pričom výstup, ktorý je považovaný za relevantnejší (použitý algoritmus má štatisticky vyššiu úspešnosť) bude zobrazený ako prvý a ostatné výstupy budú zobrazené dodatočne alebo vôbec.

Kapitola 6

Implementácia systému

6.1 Použité technológie

Systém sme sa rozhodli naprogramovať podobne ako v bakalárskej práci pomocou jazyka Java. Je to najmä kvôli nadviazaniu na predchádzajúci systém, ale aj preto, že tento jazyk poskytuje rýchle a efektívne nástroje na tvorbu webovej aplikácie. Aplikáciu budú tvoriť JSP stránky v kombinácii s obslužným servletom, ktorý bude spracovávať a vykonávať požiadavky. Rozhranie bude tvorené v HTML a CSS, pričom používateľské interakcie bude dotvárať knižnica jQuery[1], rozšírenie jazyka JavaScript.

Podobne ako v bakalárskej práci, na spracovanie webových stránok budeme využívať niekoľko knižníc. DOM4J[2] bude použitá ako štandardný nástroj pre prácu s DOM stromom. Jej výhody sme popísali v bakalárskej práci, za zmienku tentokrát stojí aj dobre spracovaný návrhový vzor *Visitor*, ktorý nám vo viacerých prípadoch uľahčí rekurzívne prechádzanie stromu. Parsovanie (spracovanie a štandardizácia) stránok bude prebiehať pomocou nástroja JTidy, teda verzie HTMLTidy[3] pre jazyk Java. Jedná sa o štandardný parser používaný spoločnosťou W3C a v súčasnosti je považovaný za jeden z najlepších. Spracovanie požiadaviek z Google Search API bude realizované pomocou JSON knižnice pre Javu[4].

Pre realizáciu samotných algoritmov použijeme tiež niekoľko nástrojov. Konkrétne pre nájdenie pageletu (kap. 4.1) a dátových záznamov (kap. 4.3.2) budeme využívať

[1] `www.jquery.com`
[2] `http://dom4j.sourceforge.net/`
[3] `http://tidy.sourceforge.net/`
[4] `http://json.org/java/`

knižnicu CSSBox[5], ktorá ako jediná spĺňala požiadavky: podpora DOM modelu, renderovanie nezávislé od jadra prehliadača, rýchlosť a efektivita. Vďaka nej je možné realizovať rýchle nájdenie jednotlivých blokov a získanie ciest XPath (keďže pracuje s inou verziou DOM knižnice, tak nepodporuje priamo XPath, ale tie je možné získať na základe jednoduchých rekurzívnych algoritmov). Jej nevýhodou je to, že ešte stále obsahuje niekoľko chýb, ktoré sú spôsobené nízkou robustnosťou a odolnosťou voči chybným stránkam.

Klasifikáciu pomocou naivného Bayesovho modelu budeme realizovať pomocou knižnice classifier4J[6], ktorá ho má štandardne implementovaný. Jej výhodou je najmä možnosť pracovať s textovými súbormi ako vstupmi ako aj pripojenie na databázové zdroje. Tento postup je veľmi výhodný, pretože zo začiatku budeme pracovať s pripravenou množinou kľúčových slov, no v neskoršej fáze budeme získavať kľúčové slová zo spracovaných webových stránok.

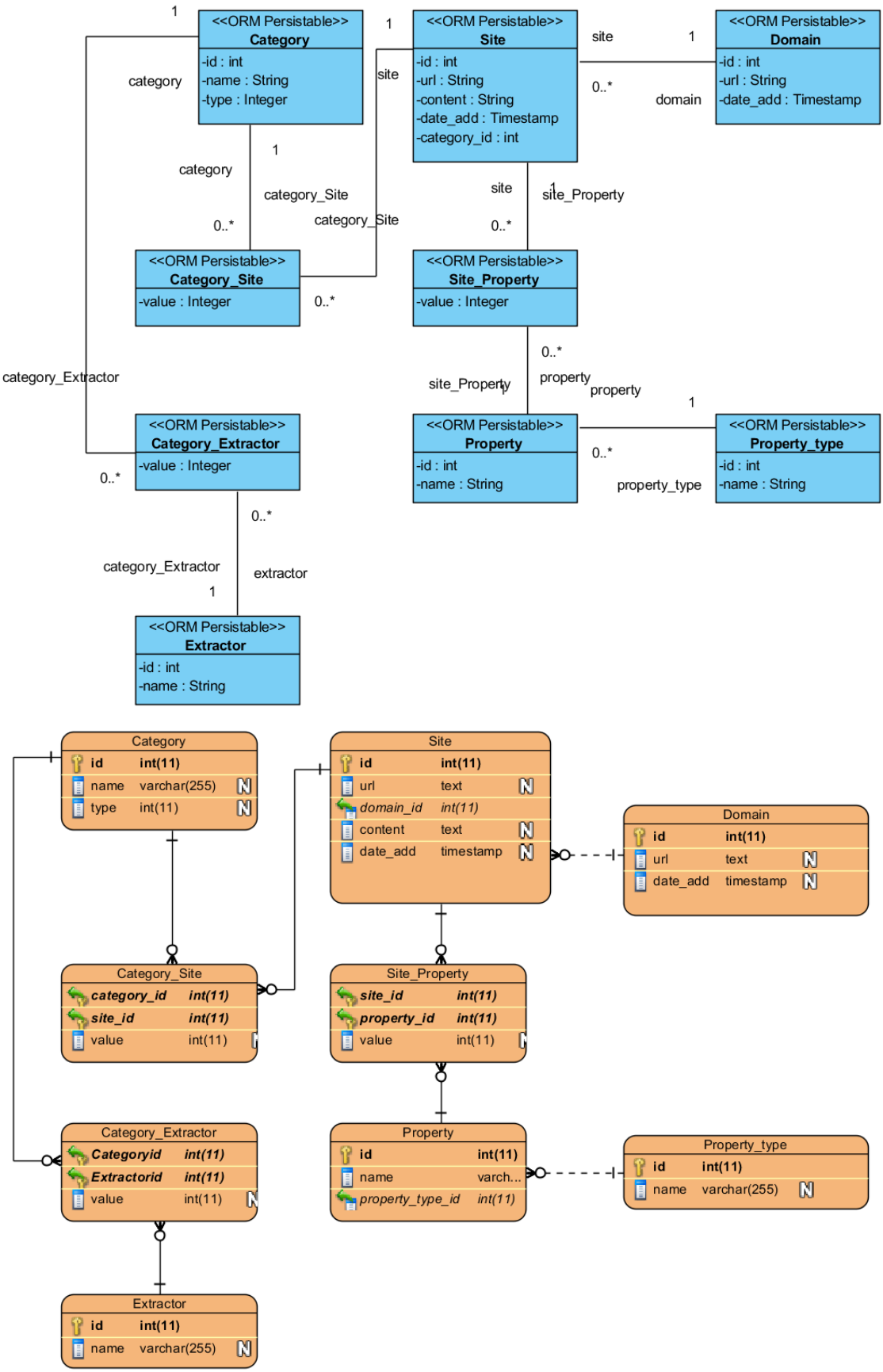
V neposlednom rade treba spomenúť technológiu Hibernate[7], ktorá bude zabezpečovať perzistenciu objektov. Oproti bakalárskej práci sa kladie väčší dôraz na celistvosť objektového modelu a prehľadnosť kódu. Preto bol pre vývoj entitno-relačného modelu databázy a objektový model zvolený program Visual Paradigm for UML[8], v ktorom vie používateľ navrhnúť tieto modely a tiež vygenerovať príslušný kód (Java, Hibernate mapovanie).

6.2 Objektový model

Na obrázku 6.15 je popísaný objektový a príslušný entitno-relačný model pre ukladané objekty primárnej databázy. Budeme vychádzať zo všeobecných predpokladov:

- *Domain* (doména) bude reprezentovať najvyššiu úroveň portálu, teda jej doménové meno, napríklad `www.upjs.sk`
- *Site* (stránka, miesto) bude jedna konkrétna webová stránka (lokalita) portálu, teda domény, napríklad `http://www.upjs.sk/studenti/`

[5] `http://cssbox.sourceforge.net/`
[6] `http://classifier4j.sourceforge.net/`
[7] `http://www.hibernate.org/`
[8] `http://www.visual-paradigm.com/product/vpuml/`



Obr. 6.15: Objektový a entitno-relačný model systému.

- *Property* (vlastnosť) bude jedna konkrétna analyzovaná vlastnosť stránky (na základe ktorej bude analyzovaná štruktúra), napríklad *počet slov*. Vlastnosť má svoj typ, teda napríklad či sa jedná o číselnú, textovú a pod. vlastnosť.
- *Category* (kategória) a *Extractor* (extraktor, extrakčná metóda) budú vopred definované hodnoty kategórií a dostupných extrakčných metód.
- Jednotlivé prepojenia sú reprezentované na obr. 6.15, teda:
  - Jedna doména má viacero stránok
  - Stránky môžu mať viacero vlastností
  - Stránky môžu mať viacero kategórií
  - Kategórie môžu mať viacero extrakčných metód

Objektový model sekundárneho úložiska nie je potrebné zvlášť opisovať, pretože fyzicky budú obidve databázy spojené. Ukladané dáta, ktoré boli získané v procese extrakcie budú ukladané pre konkrétnu stránku (dodatočná tabuľka), takže bude kedykoľvek možné získať ich metainformácie.

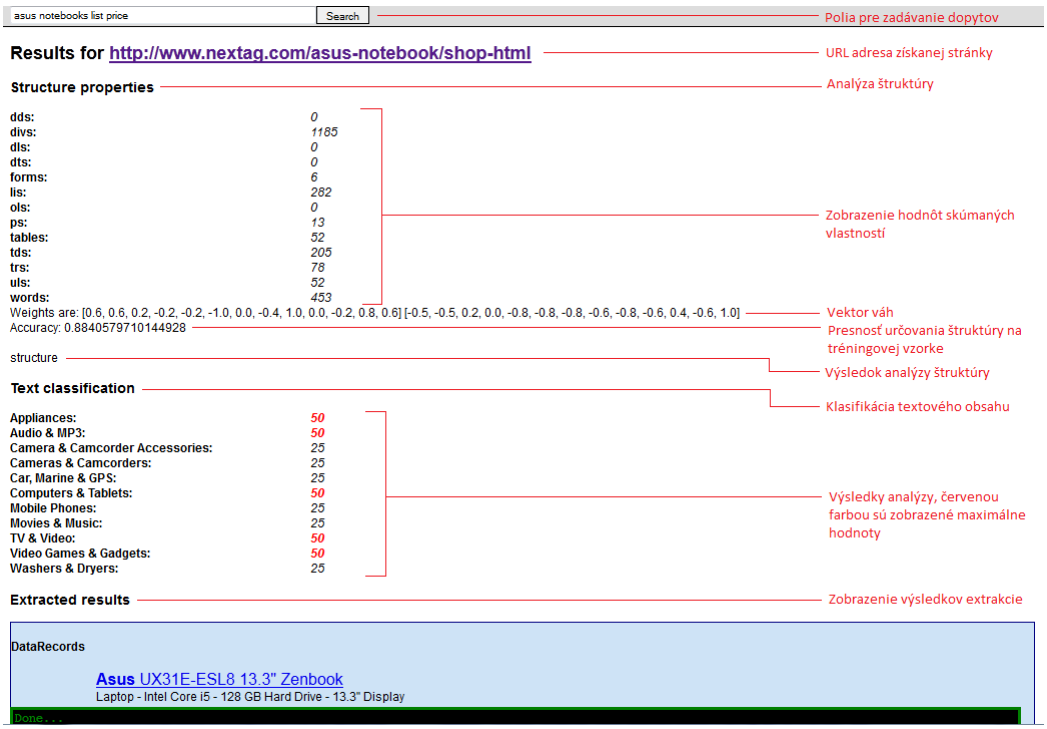
6.3 Webové rozhranie aplikácie

Úvodná stránka aplikácie bude poskytovať jednoduchý formulár pre vyhľadávanie stránok prostredníctvom vyhľadávača Google. V rámci zachovania maximálneho priestoru budú tieto polia vložené do hlavičky stránky. Obsahová časť bude obsahovať priestor pre zobrazenie výsledkov, ako aj pomocnú konzolu, ktorá bude opisovať stav aktuálne vykonávanej činnosti.

Používateľ si bude môcť zvoliť režim, v ktorom chce pracovať so systémom. V režime *train* bude môcť trénovať systém pre kategorizáciu stránok. Znamená to vyhľadanie stránok, pokusnú kategorizáciu a manuálne priradenie kategórií. Ak si zvolí režim *test*, bude môcť otestovať kategorizáciu už so zobrazením výsledkov extrakcie. Tieto výsledky bude môcť ohodnotiť pomocou zaškrtávacích polí. Nakoniec, v režime *run* bude môcť používať výsledný systém bez pomocných výpisov. Výsledkom bude priame získanie informácií.

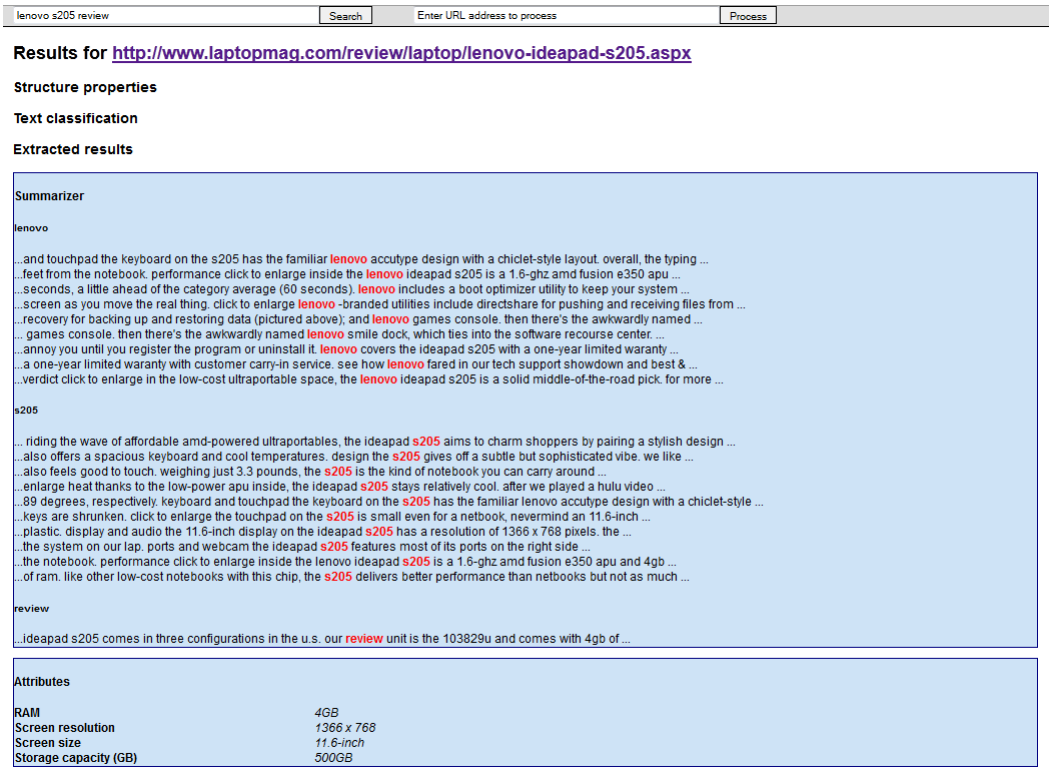
Po odoslaní požiadavky systém upozorní používateľa na to, že musí počkať kým sa spracujú výsledky. Výsledky sú potom prezentované v nasledujúcej podobe[9] (viď. obr. 6.16):

- *nadpis* - vo všeobecnosti to bude URL adresa stránky,
- *analýza štruktúry stránky* - spolu s ňou bude zobrazená aj analýza početnosti jednotlivých vlastností, váhy vektora použitého pri výbere kategórie, presnosť klasifikácie na aktuálnej tréningovej vzorke a v neposlednom rade výsledok klasifikácie,
- *analýza textovej časti* - výsledok klasifikácie na základe obsahu, percentuálna príslušnosť pre danú kategóriu,
- *extrahované údaje* - rozdelené podľa typu extrakčnej metódy,
- *konzola* - ktorá sa bude dať podľa potreby minimalizovať, a ktorá bude zobrazovať základné hlásenia systému (spustenie, zahájenie extrakcie, ukončenie extrakcie a pod.).



Obr. 6.16: Ukážka analytickej časti systému.

[9]Zobrazenie výsledkov sa môže vo všeobecnosti líšiť od zvoleného režimu



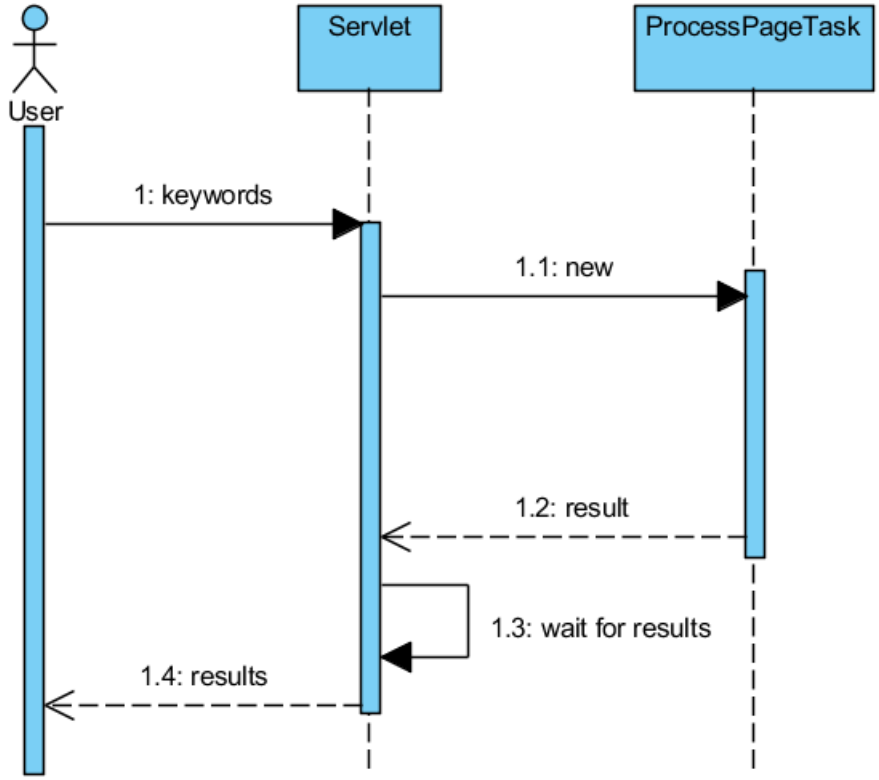
Obr. 6.17: Ukážka výsledkov extrakčného procesu.

Výsledky extrakcie budú zobrazené podľa typu výstupu, ktorý produkujú, napríklad metóda pre extrakciu dátových záznamov bude zobrazovať oddelené dátové záznamy, metóda pre extrakciu pomocou regulárnych výrazov bude zobrazovať atribúty a ich hodnoty (obr. 6.17).

Obrázky 6.16 a 6.17 popisujú stav, ktorý nastane ak je systém natrénovaný a vie samostatne rozhodovať pri výbere extrakčných algoritmov. V prvých fázach však používateľ môže korigovať výsledky dvoma spôsobmi:

- *manuálne priraďuje stránkam kategórie* - to znamená vo výsledku na obr. 6.16 je pridaný formulár pre manuálny výber kategórie, po štrukturálnej aj obsahovej stránke,
- *manuálne rozhoduje o tom, ktorá extrakčná metóda dáva najlepšie výsledky* - opäť na základe formulára. Pri každom výsledku ohraničenom modrým oknom (obr. 6.17) je pridané zaškrtávacie políčko, ktorým používateľ označí spokojnosť s extrahovanými dátami.

Ďalej je nutné dodať to, že počet výsledkov sa môže líšiť, keďže jedno volanie Google API štandardne vráti až 8 výsledkov. Dôležitá je teda aj rýchlosť spracovávania stránok týmto systémom. Vzhľadom na to, že pracujeme s renderovanou stránkou a jej DOM stromom je rýchlosť jej spracovania ekvivalentná načítaniu jej *zjednodušenej*[10] verzie v prehliadači. Takto by spracovanie viacerých stránok naraz trvalo neprimerane dlho. Vzhľadom na požiadavku spracovávania dát v reálnom čase sme upravili systém tak, aby dokázal paralelne spracovávať niekoľko stránok naraz. Pre tento účel bola vytvorená trieda `ProcessPageTask`, ktorú volá priamo servlet. Ten potom čaká na výsledok jednotlivých vlákien a posiela ich na zobrazenie do prehliadača.



Obr. 6.18: Proces extrakcie informácií za použitia vlákien.

[10]Nenačítavame externé skripty ani obrázky, objekty a pod.

Kapitola 7

Testovanie s reálnymi dátami

7.1 Testovanie kategorizácie

7.1.1 Podľa štruktúry

Pre otestovanie kategorizácie sme pripravili niekoľko testovacích množín, ktoré sa skladali z webových stránok a ich hodnotení pre príslušnú kategóriu. V prvom prípade sme pracovali s klasifikáciou, ktorá brala do úvahy len dve kategórie - štruktúrované a textové stránky. V tomto prípade bolo použitých 205 stránok. V druhom prípade sme klasifikovali stránky do kategórií: detail produktu, recenzia produktu a zoznam produktov. Testovacia množina pozostávala z 220 manuálne kategorizovaných stránok.

Pri klasifikácií stránky sa bralo do úvahy 16 parametrov:

- Počet paragrafov, resp. tagov `p`
- Počet neusporiadaných zoznamov, resp. tagov `ul`
- Počet usporiadaných zoznamov, resp. tagov `ol`
- Počet prvkov zoznamu, resp. tagov `li`
- Počet tabuliek, resp. tagov `table`
- Počet riadkov tabuliek, resp. tagov `tr`
- Počet buniek tabuliek, resp. tagov `td`
- Počet zoznamov s definíciami, resp. tagov `dl`

- Počet definícií, resp. tagov `dt`
- Počet vysvetlení definícií, resp. tagov `dd`
- Počet formulárov, resp. tagov `form`
- Počet deliacich oblastí, resp. tagov `div`
- Počet slov, ktoré sa nachádzajú buď v paragrafoch alebo deliacich oblastiach
- Počet zvýraznení, resp. tagov `b`,`i`,`strong`,`em`
- Počet nadpisov, resp. tagov `h1` – `h6`

Úspešnosť klasifikácie je možné vidieť v tabuľke 7.1. Pri použití klasifikácie pomocou neurónových sietí[1] bola úspešnosť klasifikácie nižšia - hlavným problémom bol odhad parametrov siete a čas potrebný na naučenie. Ďalším problémom bol aj fakt, že skúmané parametre na jednotlivých stránkach boli málo špecifické pre danú kategóriu.

Použitím FVC algoritmu [12] sa úspešnosť zvýšila pri prvej aj druhej množine. Na zistenie váh bol použitý randomizovaný algoritmus hľadajúci maximum, ktorý mal stanovený minimálny limit pre úspešnosť. V prvom prípade bola najlepšia úspešnosť až 88%, čo bolo spôsobené tým, že sme klasifikovali len do dvoch kategórií. V druhom prípade už presnosť klasifikácie klesla, skúmané znaky zrejme dosť vhodne nerozdeľovali tréningovú množinu do troch stanovených kategórií. Zvýšenie úspešnosti klasifikácie pomocou FVC algoritmu by bolo možné dosiahnuť použitím vhodnejších príznakov, ktoré lepšie charakterizujú danú skupinu a v ostatných sa nevyskytujú.

	Neurónové siete	FVC
1. množina	64%	88%
2. množina	61%	71%

Tabuľka 7.1: Úspešnosť kategorizácie na testovacích množinách. Uvedená úspešnosť predstavuje najlepšie hodnoty zo série niekoľkých pokusov.

[1]Konkrétne viacvrstvový perceptrón so spätným šírením chyby (backpropagation) a využitím parametra *momentum*.

7.1.2 Podľa obsahu

Testovanie kategorizácie obsahu prebiehalo podobne, teda na pripravenú množinu vopred kategorizovaných stránok bol aplikovaný implementovaný naivný Bayesov klasifikátor. Kategorizácia obsahu v tomto prípade nebola jednoznačná, čo malo za následok pridelenie viacerých tried jednej stránke. Tréningová množina bola zostavená podľa členenia spomenutého v 4.2.2, avšak samotné trénovanie bolo zamerané na dve konkrétne kategórie a to *Mobile phones* a *Computer and tablets*. Testovanie sme teda zamerali na správnu kategorizáciu týchto dvoch tried. Za úspešnú sme považovali takú klasifikáciu, ktorá obsahovala triedu definovanú používateľom.

Všetky stránky	Mobile phones	Computers and tablets	Ostatné
215 / 13 (94%)	96 / 5 (95%)	83 / 1 (98%)	36 / 7 (85%)

Tabuľka 7.2: Úspešnosť kategorizácie textového obsahu (dobré / zlé a percentuálna úspešnosť).

Ako ukazuje tabuľka 7.2, Bayesov klasifikátor nájde správnu kategóriu vo veľkej väčšine prípadov. Podstatné je však spomenúť, že klasifikátor často priradil jednej stránke viacero tried. To je spôsobené jednak malou množinou kľúčových slov, ale aj nízkej špecifickosti jednotlivých kategórií. Napríklad kľúčové slovo *display* je charakteristické pre viacero tried, napríklad obrazovku (*display*) má mobilný telefón, fotoaparát, ale aj tablet a pod.

7.2 Testovanie extrakcie

Pri tomto testovaní sme sa zamerali na správne rozhodnutie systému vybrať extrakčný algoritmus na základe pridelenej kategórie. Výber metód sa sústreďoval na kategorizáciu podľa štruktúry, ale metódy využívali aj klasifikáciu podľa obsahu (regulárne výrazy). Extrakčné algoritmy boli vyberané tak, aby pokryli požadovanú kategóriu. V tomto prípade sme sa snažili, aby:

- metóda pre extrakciu dátových záznamov pokrývala kategóriu *zoznam produktov*,
- sumarizátor pokrýval kategóriu *recenzia produktu*,

- metóda pre extrakciu atribútov pomocou regulárnych výrazov pokrývala kategóriu *detail produktu*.

Ako už bolo spomenuté, používateľ má v režime *test* možnosť vybrať (jednu ale aj viac) najvhodnejšie metódy, ktoré pre konkrétnu stránku a kategóriu extrahujú najrelevantnejšie informácie. Sériu týchto pokusov je možné vidieť v tabuľke 7.3. Výsledok zodpovedá predpokladom, za povšimnutie stojí aj fakt, že algoritmus `RegExp` bol v niektorých prípadoch úspešný aj v kategórii *Recenzia produktu*, čo je dôsledok toho, že v recenziách sa často vypisujú atribúty produktu.

	Detaily produktu	Recenzia produktu	Zoznam produktov
DRE	6	2	**45**
Sumarizér	11	**77**	13
RegExp	**57**	21	3

Tabuľka 7.3: Preferencie používateľa pri výbere extrakčnej metódy na základe priradenej kategórie. DRE je skratka algoritmu pre extrakciu dátových záznamov, RegExp je skratka algoritmu pre extrakciu atribútov na základe regulárnych výrazov.

Záver

V tejto práci sme sa zaoberali navrhnutím systému pre extrakciu dát, ktorý by využíval kategorizáciu webových stránok ako predprípravu pred samotnou extrakciou. Navrhli sme prototyp webovej aplikácie, ktorá dokáže v reálnom čase z výsledkov vyhľadávania pomocou kľúčových slov získať webové stránky a im následne priradiť kategórie, pomocou ktorých bude možné vybrať najvhodnejšie extrakčné metódy. Systém v prvotných fázach využíva učenie za asistencie učiteľa pre vytvorenie tréningovej množiny, no neskôr pracuje bez zásahu používateľa.

Tento systém rozvíja myšlienku načrtnutú v bakalárskej práci [1] a zachováva jeho pôvodnú filozofiu výberu extrakčných metód. Pre porovnanie bolo implementovaných niekoľko extrakčných metód, ktoré poskytujú rôzne výsledky v závislosti od zadanej vstupnej množiny dát. Dôležitým krokom v tejto práci bolo aj navrhnutie získavania zdrojov na základe vyhľadávania pomocou kľúčových slov. Takýmto spôsobom bolo zabezpečené, že zdroje budú obsahovať relevantné a aktuálne údaje. Ďalším cieľom bolo navrhnúť proces klasifikácie webových stránok – vybraný hierarchický model kategorizuje stránky na základe obsahu a štruktúry. Vybraná klasifikácia zohľadňovala viacero znakov, na základe ktorých bolo možné vybrať vhodnú extrakčnú metódu. Systém obsahuje aj niekoľko učiacich postupov, ktoré napomáhajú efektívnejšej klasifikácii a pomáhajú pri výbere extrakčných metód. Používateľské rozhranie obsahuje ovládacie prvky, ktorými je možné manuálne priraďovať kategórie a preferovanú extrakčnú metódu. V neposlednom rade bolo vykonaných aj niekoľko testov, ktoré potvrdzujú úspešnosť a zdôrazňujú úlohu kategorizácie pri výbere správneho extrakčného algoritmu.

Vytvoriť plne automatizovaný systém, ktorý by poskytoval bezchybné výsledky je takmer nemožné. V súčasnosti je veľká väčšina webových stránok chybná, t.j. v rozpore s definovaným štandardom, čo do veľkej miery sťažuje možnosti strojového spra-

covania. Ako efektívne riešenie sa ukazuje poloautomatický systém, kde používateľ vie korigovať výsledky extrakcie, resp. statické (*hard-coded*) wrappery, ktoré sú naprogramované pre konkrétny portál. Klasifikácia webových stránok však môže byť nápomocná aj v týchto riešeniach, kedy môže slúžiť ako nultý krok extrakčného procesu, teda predspracovanie neznámej množiny za účelom zúženia vstupnej množiny, skvalitnenia výsledkov a v neposlednom rade menšej miere zásahu používateľa.

V testoch sa ukázalo, že kategorizácia podľa štruktúry pomocou FVC algoritmu je veľmi citlivá na sledované znaky. Kvôli zvýšeniu úspešnosti, by bolo vhodnejšie podrobné skúmanie stránok tréningovej množiny v každej kategórii. Napríklad pre niektorú kategóriu je charakteristický nielen počet slov, ale aj rozmer a umiestnenie použitých obrázkov, či farebnosť stránky. Kategorizácia na základe obsahu je na druhej strane citlivejšia na vhodný výber kategórií. Ak sú kategórie príliš významovo podobné, tak množiny kľúčových slov, ktoré ich reprezentujú majú veľký prienik, čo spôsobuje nejednoznačné pridelenie kategórie.

V budúcnosti by sme sa preto najviac zamerali na navrhnutie presnejšej klasifikácie, čo znamená detailnejšie skúmanie požadovaných tried a ich vzájomné rozlíšenie. Samotnému procesu extrakcie by pomohlo implementovanie ďalších metód, ktoré sa zaoberajú rôznym typom stránok. Takisto aj webové rozhranie systému je možné ďalej upravovať, pre lepšiu funkcionalitu by bolo vhodné doplniť prácu s výslednými dátami a ich anotáciu.

Zoznam použitej literatúry

[1] Rajzák P., Novotný R.: Kolaboratívna anotácia webových stránok. *Bakalárska práca, ÚINF PF UPJŠ*. 2010.

[2] Mitchell, T. M.: Machine Learning. 1997. New York: McGraw-Hill.

[3] Wai-chiu Wong, Ada Wai-chee Fu: Incremental Document Clustering for Web Page Classification. 2000.

[4] Glover, E. J., K. Tsioutsiouliklis, S. Lawrence, D. M. Pennock, and G. W. Flake: Using web structure for classifying and describing web pages. In *Proceedings of the 11th International Conference on World Wide Web, New York, NY*. 2002. p. 562—569. ACM Press.

[5] Cohen, W. W.: Improving a page classifier with anchor extraction and link analysis. In *S. Becker, S. Thrun, and K. Obermayer (Eds.), Advances in Neural Information Processing Systems, Volume 15*. 2002. p. 1481—1488. Cambridge, MA: MIT Press.

[6] Deerwester, S. C., S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman: Indexing by latent semantic analysis. In *Journal of the American Society of Information Science 41(6)*. 1990. p. 391-–407.

[7] Alias-i: LingPipe 4.0.1. 2008. `http://alias-i.com/lingpipe` (posledné zmeny 21.10.2010, 19:27:48)

[8] Fisher, M. J. and R. M. Everson: When are links useful? Experiments in text classification. In *Advances in Information Retrieval. 25th European Conference on IR Research*. 2003. p. 41—56. Springer.

[9] Jensen, D., J. Neville, and B. Gallagher: Why collective inference improves relational classification. In *KDD '04: Proceedings of the 10th ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY. 2004. p. 593-–598. ACM Press.

[10] Kwon, O.-W. and J.-H. Lee: Text categorization based on k-nearest neighbor approach for web site classification. In *Information Processing and Management 29(1)*. 2003. p. 25-–44.

[11] Yu, H., J. Han, and K. C.-C. Chang: PEBL: Web page classification without negative examples. In *IEEE Transactions on Knowledge and Data Engineering 16 (1)*. 2004. p. 70-–81.

[12] Arul Prakash Asirvatham, Kranthi Kumar. Ravi: Web Page Classification based on Document Structure. *Technical Report*. Center for Information Technology, India, 2001.

[13] Qi X.,Davison B. D.: Web Page Classification: Features and Algorithms. In *Technical Report LU-CSE-07-010, Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA, 18015.*. 2007.

[14] Qi, X. and B. D. Davison: Knowing a web page by the company it keeps. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM), New York, NY*. 2006. p. 228-–237. ACM Press.

[15] Wolpert, D.: Stacked generalization. In *Neural Networks 5*. 1992. p. 241—259.

[16] Chang Chia-Hui, Kayed M., Girgis M. R., Shaalan K.: A Survey of Web Information Extraction Systems. In *IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 10*. 2006. p. 1411–1428.

[17] Cohen W. W., McCallum A.: Information Extraction from the World Wide Web. 2002.

[18] Li Y., Krishnamurthy R., Raghavan S., Vaithyanathan S.: Regular Expression Learning for Information Extraction. 2008.

[19] Laclavík M., Šeleng M., Babik M.: OnTeA: Semi-automatic Ontology based Text Annotation Method. In *Tools For Acquisition, Organisation and*

Presenting of Information and Knowledge. 2006. ISBN 80-227-2468-8. p. 49–63.

[20] Liu B., Grossman R., Zhay Y.: Mining Data Records in Web Pages. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2003. p. 601–606.

[21] Gengxin Miao, Junichi Tatemura, Wang-Pin Hsiung, Arsany Sawires, Louise E. Moser1: Extracting Data Records from the Web Using Tag Path Clustering. In *Proceedings of the 18th international conference on World wide web*. 2009.

[22] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma.: Extracting content structure for web pages based on visual representation. *In Proceedings of the 5th Asia-Pacific web conference on Web technologies and applications (APWeb'03), Xiaofang Zhou, Maria E. Orlowska, and Yanchun Zhang (Eds.). Springer-Verlag, Berlin, Heidelberg.* 2003. p. 406–417

Prílohy

Príloha A: Systém CaES (*Categorization and Extraction System*) a inštalačná príručka v elektronickej forme.