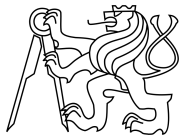
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Nástroj pro analýzu článků z českých zpravodajských serverů
Student:	Bc. Markéta Filipová
Vedoucí:	Ing. Jaroslav Kuchař
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Cílem práce je analýza článků a souvisejících dat ze současných českých zpravodajských serverů. Práce bude obsahovat návrh, implementaci a testování modulů pro získání obsahu z webu, automatickou detekci především textových informací v HTML a následnou analýzu.
1) Prozkoumejte dostupné crawlery pro sběr dat na webu a zvolte z nich řešení vhodné pro případ zpravodajských serverů.
2) Analyzujte možnosti separace relevantních dat novinových článků z HTML stránek generovaných zpravodajskými servery.
3) Navrhněte a implementujte vlastní algoritmus pro automatické získání potřebných dat z českých serverů ve formě Node.js modulů.
4) Prozkoumejte základní metody pro analýzy textů a zvolené metody implementujte jako Node.js moduly. Zaměřte se na český jazyk.
5) Proveďte experiment na předních českých zpravodajských serverech. Zaměřte se na správnost extrakce informací z HTML stránek a porovnejte výsledky analýz na jednotlivých serverech za zvolené období.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 10. ledna 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Nástroj pro analýzu článků z českých zpravodajských serverů

Bc. Markéta Filipová

Vedoucí práce: Ing. Jaroslav Kuchař

10. května 2016

Poděkování

Velice děkuji vedoucímu své diplomové práce Ing. Jaroslavu Kuchařovi za výborné vedení, cenné připomínky, ochotu a trpělivost.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 10. května 2016 .....

Odkaz na tuto práci

Filipová, Markéta. *Nástroj pro analýzu článků z českých zpravodajských serverů.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

V dnešní době, kdy množství informací na internetu stále narůstá, se automatické zpracování a třídění dat stalo velmi oblíbeným oborem iformačních technologií. Jednou z oblastí, kde se nachází množství různých, ale i podobných informací, je internetové zpravodajství. Cílem této práce bylo vytvořit nástroj pro analýzu článků z českých zpravodajských serverů, který by zjednodušil orientaci v obsahu, jenž je těmito servery každý den generován. Pro splnění tohoto cíle bylo vytvořeno několik Node.js modulů, kde se každý z nich zabývá určitou částí v procesu získání zajímavých dat. První z nich je crawler, díky kterému je možné stáhnout články k analýze ze zpravodajských webů. V druhé části je ze stažených HTML stránek extrahován relevantní obsah článků a jejich další atributy. Třetí částí je pak textová analýza, kde byly vytvořeny moduly pro extrakci pojmenovaných entit a pro analýzu sentimentu českého textu. V závěru práce pak byly vytvořeny dva CLI programy, kterými je možné pohodlně stáhnout a analyzovat články, jejichž výstupem jsou atributy článku, jeho obsah, sentiment a seznam entit a jejich výskytů v textu. Tyto články pak byly nahrány do databáze Apache Solr, pro kterou bylo vytvořeno několik dotazů a provedeny experimenty.

Klíčová slova zpravodajské servery, čeština, web mining, crawler, extrakce obsahu, text mining

Abstract

Nowadays, when the amount of information on the internet continues to grow, automatic processing and analysis of data has become a very popular specialisation in the field of the information technologies. Online news service is one of the domains in which a significant amount of diverse as well as similar information exists. The goal of this thesis was to create a tool for analysis of Czech news articles which would simplify orientation in the data that is generated by these servers every day. To accomplish this goal a several Node.js modules have been created, each of which deals with a specific part of the process of obtaining interesting data. The first one is a crawler which allows downloading articles for analysis from news servers. In the second part, relevant content of articles and their other attributes are extracted from downloaded HTML pages. The third part is a text analysis for which modules for extraction of named entities and for sentiment analysis of Czech texts have been created. Finally, two CLI programs have been created, which allow easy download and analysis of articles, and the output of which are attributes of the article, its content, sentiment and a list of entities and its occurrences in the text. These articles have then been imported to the database Apache Solr for which several queries have been created and a number of experiments have been performed.

Keywords news servers, Czech, web mining, crawler, content extraction, text mining

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Dnešní internet je velmi rychle se rozrůstající struktura, ve které je pro jednotlivce stále těžší se orientovat. To platí i pro zpravodajské servery. Ačkoli stále vycházejí tištěné noviny a časopisy informující o každodenních událostech, mnoho čtenářů se přesunulo k jejich online verzím a nově vzniklým zpravodajským webům. Vyznat se v množství informací, které tyto zpravodajské servery produkují, může být značně náročné. Tato práce si klade za cíl vytvořit nástroj, který by byl nápomocný při analýze novinových článků produkovaných různými českými zpravodajskými weby. Problematika se skládá z několika oblastí, které na sebe vzájemně navazují.

Dle jednotlivých oblastí, kterými se tato práce zabývá, je vytvořena i struktura kapitol. První kapitola je věnována teoretickému úvodu. Bude zde vysvětleno, co je to data mining, web mining a text mining, některé jejich části pak budou popsány detailněji. Další části se již věnují analýze zpravodajských webů a dále procesu získání, extrakce a analýzy obsahu článků. V závěrečné kapitole jsou spojením těchto částí získána data, na nichž jsou provedeny experimenty.

V procesu získání relevantních dat se práce nejprve zabývá analýzou zpravodajských serverů a jejich obsahu. Jejím cílem je zjistit, jaké existují české zpravodajské servery, co za obsah a v jaké formě se na nich nachází a jaké části článků by bylo vhodné extrahovat. Dále jsou v této části rozebrány možnosti textové analýzy obsahu zpravodajských článků.

Dále se práce zaměřuje na proces získání dat ze zpravodajských serverů. Na každém webu je třeba detekovat stránky, jejichž obsahem jsou články vhodné k analýze. Ty jsou pak automatizovaně stahovány do zadaného období.

Stažené HTML stránky ale obsahují množství nežádoucích prvků, jako je šablona webu nebo reklamy, které s obsahem článků nijak nesouvisí. Ty je nutné odfiltrovat a tím získat pouze relevantní obsah a další atributy článku.

Jakmile jsou k dispozici čisté informace o článku, lze na nich provést textovou analýzu. V této části práce jsou prozkoumány možnosti analýzy pro zpravodajské články. Ze zjištěných metod byly vybrány pro implementaci ex-

trakce entit a analýza sentimentu.

V poslední části jsou vytvořeny dva CLI programy, z nichž jeden získává a parsuje články ze zpravodajských serverů a druhý stažená data doplňuje o textovou analýzu. Programy je možné spustit paralelně. Jejich výsledkem je pro každý článek jeho obsah s dalšími atributy, sentiment a seznam entit a jejich výskytů v textu. Tato data jsou pak uložena do databáze Apache Solr, pro kterou jsou vytvořeny složitější dotazy. Na závěr jsou pomocí uvedených modulů a programů staženy a analyzovány články ze čtyř českých zpravodajských serverů za období leden až duben 2016. Pro tato data jsou spuštěny připravené dotazy a výsledky jsou analyzovány.

Kapitola 1

Dolování dat z webu

Dnešní web je rozsáhlá dynamická struktura, která se každým dnem mění a její objem roste. Je možné zde najít téměř cokoli, od různých informačních stránek, zpravodajských webů či blogů, přes sociální sítě a zájmové weby až po komerční webové aplikace jako jsou například e-shopy. Díky tomu je web velký zdroj dat, ze kterého je možné při použití správných algoritmů získat množství užitečných informací. A právě tím se zabývá obor dolování dat z webu, který vychází z oboru tradičního dolování z dat.

V této kapitole bude stručně vysvětleno, co je to dolování z dat, z jakých oblastí se tento obor skládá a jak s nimi tato práce souvisí. Vybraná témata budou popsána detailněji.

1.1 Data mining a jeho oblasti

Data mining (česky překládáno jako *dolování z dat*) je podle Liu [1] definováno jako proces objevování užitečných vzorů nebo znalostí z datových zdrojů jako jsou například databáze, texty, obrázky, web atd. Vzor musí být platný, potenciálně užitečný a srozumitelný. Data mining se skládá z více oborů, jako jsou například strojové učení, statistika, databázové systémy, umělá inteligence a vizualizace. Typickými úlohami data miningu jsou učení s učitelem (neboli klasifikace), učení bez učitele (neboli clustering), dolování asociačních pravidel nebo dolování sekvenčních vzorů.

Samotný proces dolování dat začíná pochopením domény, ze které se data analyzují. Díky tomu je možné identifikovat zdroje a konkrétní data, na která se pak bude aplikovat dolování, a stanovit cíl celého procesu. Jakmile jsou potřebná data získána, proces probíhá ve třech krocích, které se mohou iterativně opakovat [1].

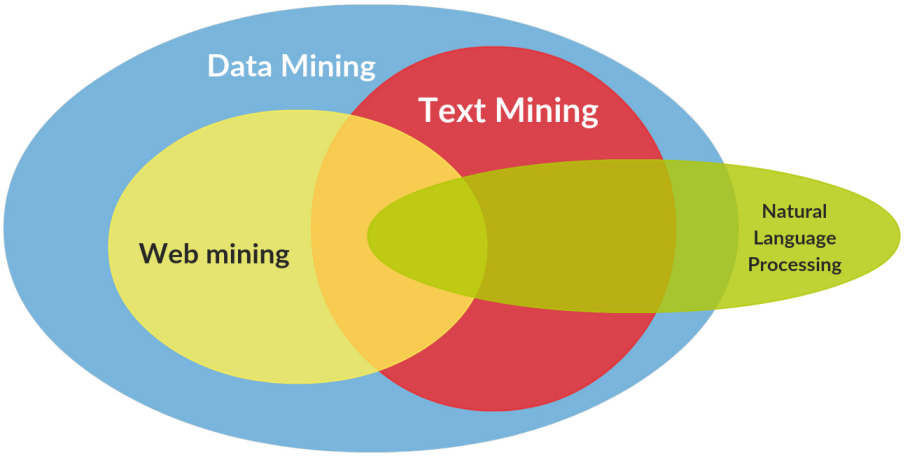
1. *Preprocessing*, neboli *předzpracování*, je proces, kdy se data připraví pro použití v samotném algoritmu data miningu. Z dat mohou být vynechány nerelevantní informace a mohou být pročištěna od různého šumu

či abnormalit. Pokud jsou zbytečně rozsáhlá, je možné na ně aplikovat algoritmy, které je rozumně zredukují.

2. *Data mining* je část, kdy se na získaná data aplikují algoritmy dolování dat.

3. *Postproccesing* je poslední fáze, kdy je o vydolovaných informacích rozhodnuto, zda jsou užitečné pro zamýšlený účel. Pro toto rozhodnutí jsou využívány různé techniky vyhodnocování a vizualizace.

Součástí data miningu jsou mimo jiné *web mining* či *text mining*, což jsou hlavní oblasti, do kterých spadá téma této práce. Do nich částečně zasahuje i disciplína *zpracování přirozeného jazyka*, neboli NLP (zkratka anglického pojmu *Natural Language Processing*), kterou využiji při analýze dat získaných ze zpravodajských serverů. Vztah zmíněných oblastí zobrazuje obrázek 1.1.



Obrázek 1.1: Vztah oblastí týkajících se data miningu [2]

1.2 Web mining

Web mining (česky překládáno jako *dolování dat z webu*) je součást data miningu, která se specializuje na získávání dat či znalostí z webu. Informace je možné získávat ze struktury webových odkazů (tedy jak jsou mezi sebou stránky prolinkovány), z obsahu webových stránek a nebo z dat o jejich využívání. Ve web miningu se využívá mnoho tradičních technik data miningu. Navíc kvůli různorodosti a různé strukturovanosti webových dat bylo pro tuto oblast v posledním desetiletí vytvořeno množství nových úloh a algoritmů [1].

Kroky v procesu dolování dat z webu jsou stejné jako v procesu dolování dat obecně (viz sekce 1.1), liší se pouze v použitých algoritmech. Větší důraz

je kladen na části získávání dat a jejich předzpracování, kdy v tradičním data miningu jsou většinou data pro analýzu již posbírána a uložena v datovém skladu, zatímco ve web miningu je potřeba nejprve kolekci čistých dat získat [1]. To zahrnuje stažení dat z datového zdroje a následnou identifikaci hlavního obsahu a jeho vyčištění od nepotřebných informací. Sběrem dat z webu se budu zabývat v kapitole 1.4 a vyfiltrováním relevantního obsahu v kapitole 1.5.

V závislosti na typu dat, která jsou v procesu web miningu využita, se tato oblast dělí na tři kategorie: *Web structure mining*, *Web content mining* a *Web usage mining* [1].

1.2.1 Web structure mining

Web structure mining využívá faktu, že web je strukturován jako graf, jehož jednotlivé uzly, tedy webové stránky, jsou propojeny hypertextovými odkazy, které reprezentují hrany grafu. Z této struktury jsou pak dolovány užitečné informace. Tuto část web miningu využívají především webové vyhledávače a crawlery. Typickou úlohou web structure miningu je PageRank, což je algoritmus, který využívá struktury odkazů webových stránek pro určení skóre kvality či reputace každé z nich [1].

V této práci je web structure miningu využito pro procházení struktury zpravodajských webů a nalezení stránek, které obsahují samostatné články a tudíž je žádoucí je stáhnout pro analýzu.

1.2.2 Web content mining

Web content mining se soustředí na získávání dat a znalostí z obsahu webových stránek. Na webu se nachází data v různém stupni strukturovanosti — jednak data úplně strukturovaná, jako jsou třeba excelové tabulky, jednak částečně strukturovaná, např. stránky obsahující microdata, ale i nestrukturovaná v podobě čistého neanotovaného textu. K dolování informací z těchto dat mohou být využity jednak tradiční techniky data miningu, např. klasifikace či clustering webových stránek dle jejich tématu, ale i úlohy specifické pro web mining. Příkladem takových úloh může být objevování vzorů na webových stránkách, čímž se extrahují užitečná data jako je popis produktů, příspěvky v diskuzních fórech a podobně.

V případě této práce je web content mining využit pro získání relevantních textových dat ze stránek, které obsahují zpravodajské články.

1.2.3 Web usage mining

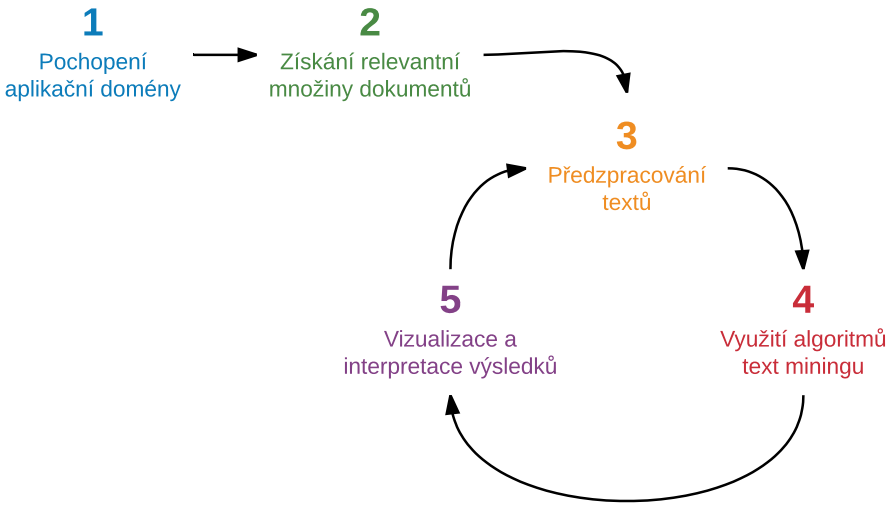
Web usage mining analyzuje informace o tom, jak uživatelé používají webové stránky a jak se na nich chovají. Data pro analýzu se získávají z dat uložených na webovém či aplikačním serveru, např. uživatelské logy, nebo přímo z dat uložených webovou aplikací, např. na co uživatel kdy kliknul apod. Všechny

typy těchto dat většinou prochází některým druhem předzpracování, čímž se generují data vhodná pro dolování.

1.3 Text mining

Text mining (česky překládáno jako *dolování znalostí z textů*) je specifická úloha data miningu, kdy místo strukturovaných dat jsou k dispozici pouze nestrukturovaná data, tedy čistý neanotovaný text. Vzhledem k důležitosti přirozeného jazyka v oblasti textové analýzy jsou v text miningu využívány výhody jiných počítačových disciplín týkajících se jeho zpracovávání. Jako příklad je možné uvést techniky a metody z oblastí information retrieval, extrakce informací či počítačové lingvistiky založené na korpusech [3].

Kroky procesu dolování dat z textů jsou zobrazeny na obrázku 1.2 a podobně jako u web miningu jsou totožné s procesem tradičního dolování z dat (viz sekce 1.1). Liší se pouze v konkrétních úlohách jednotlivých kroků, hlavně v algoritmech předzpracování dat a samotného dolování. Právě o těchto dvou krocích v procesu text miningu bude zbytek této sekce.



Obrázek 1.2: Proces text miningu

1.3.1 Předzpracování textu

Fáze předzpracování textů má v text miningu za úkol strukturovat získaná neupravená data a tím je připravit pro vstup do algoritmů pro analýzu. Konkrétní použité techniky předzpracování se liší v závislosti na dané úloze text miningu a potřebném formátu dat pro použití zamýšleného algoritmu [3].

V první řadě je vždy nutné převést dokumenty v datové kolekci do jednotného formátu, nejlépe obyčejného textu (neboli *plain textu*). Jednotlivé dokumenty totiž mohou být uloženy v různých formátech jako jsou PDF, MS Word, HTML, plain text apod. Při převodu je také třeba ošetřit možnou neúmyslnou ztrátu metadat dokumentu. Zásadní vliv má při transformaci analyzovaných dokumentů do jednotného formátu také jejich kódování [4]. Dokumenty se buď musí všechny převést do jednoho typu kódování, nebo je nutné v text miningových algoritmech s různým kódováním počítat. Problém kódování budu detailněji řešit v sekci 1.5.2.3.

Některé prameny [4] do fáze předzpracování řadí i modely reprezentace textových dokumentů a s ním související úpravy jako například váhování a normování termů či odstranění slov s nízkou četností v dokumentu. Jelikož tyto techniky nejsou v práci využity, nebudu se jimi v této kapitole zabývat.

Další částí této fáze jsou úlohy lingvistického předzpracování, které jsou závislé na jazyce, ve kterém jsou psané zkoumané texty. Z důvodu zaměření této práce se budu dále věnovat pouze úlohám předzpracování pro češtinu.

V této práci je text miningu využito pro analýzu obsahu článků stažených ze zpravodajských serverů. Vstupem této analýzy budou dokumenty uložené v jednotném formátu a kódování, proto se tato kapitola bude dále zabývat pouze metodami lingvistického přezpracování textů.

1.3.1.1 Segmentace a tokenizace

Segmentace, členění na slova a následná tokenizace jsou počáteční textové operace, při kterých se ve vstupním textu identifikují základní lexikální textové jednotky — slova, slovní spojení a fráze, věty, odstavce apod. [4]

Ve fázi segmentace se text dělí na nejmenší přípustné sekvence znaků, tzv. *elementární textové jednotky*, kterými jsou jednak souvislé řetězce alfanumerických znaků oddělené mezerami nebo interpunkcí, a jednak jednotlivé znaky interpunkce [4].

Například věta „*Fakulta informačních technologií (FIT) je nejmladší fakultou ČVUT, vznikla 1. července 2009.*“ se segmentací rozdělí způsobem, jaký je vidět ve výstupu 1.1.

Výstup 1.1: Výstup tokenizace

```
1 [Fakulta][informačních][technologií][(][FIT][)][je][
    nejmladší][fakultou][ČVUT][,][vznikla][1][.][čer-
    vence][2009][.]
```

Ve fázi tokenizace se tyto elementární textové jednotky spojují do tzv. *lexikálních jednotek*, neboli *tokenů*, které většinou odpovídají konkrétnímu slovníkovému záznamu. Například textové jednotky `[1]` a `[.]` se mohou spojit do tokenu `[1.]`, což odpovídá slovníkovému záznamu řadové číslovky „první“. Tokeny mohou být ale i složitější číselné výrazy (jako například „3,14“, „1 984“,

„25 °C“), ustálená složená pojmenování a názvy („Ústí nad Labem“, „Masarykovo nádraží“), zkratky („apod.“), e-maily, adresy webových stránek a další speciální formáty [4].

Výsledek tokenizovaného textu může být v několika formátech (např. XML), záleží na použitém softwaru. Někdy se přidává i informace o začátku a konci vět v textu.

1.3.1.2 Lematizace a morfologická analýza

Čeština je vysoce flektivní jazyk, což znamená, že gramatické funkce vyjadřuje koncovkami připojovanými ke kořenu nebo kmenu slova, přičemž koncovka může vyjadřovat (a běžně vyjadřuje) několik gramatických významů najednou. [5] Například 2. pád slova *strom*, stejně jako jeho 6. pád, má tvar *stromu*. I přes toto zjednodušení je čeština, narozdíl například od angličtiny, bohatá na různé tvary slov vlivem různých pádů, osob, čísel atd.

V procesu *lematizace* se pro každé slovo ve větě zjišťuje jeho základní tvar, který se nazývá *lema*. Například pro podstatná a přídavná jména je to první pád jednotného čísla (*stromům* → *strom*, *zeleným* → *zelený*) a pro slovesa infinitiv (*prezentovala* → *prezentovat*) [4].

Speciální formou lematizace je tzv. *stemming*, neboli izolace kořenu slova, kdy se zkoumané slovo nahradí pouze jeho kořenem (*spisovatel* → *pis*, *lesní* → *les*) [4]. V češtině to ale znamená značnou ztrátu významných informací o slovu.

Pro český jazyk se lematizace řeší morfologickou analýzou [4]. „*Morfologie (neboli tvarosloví) zkoumá tvary slov jednotlivých slovních druhů, jejich formu a funkci.*“ [5] Cílem morfologické analýzy slova je určit morfologické kategorie daného slova v textu. V češtině se používá celkem 13 slovních kategorií: slovní druh, slovní „poddruh“, rod, číslo, pád, přivlastňovací rod, přivlastňovací číslo, osoba, čas, slovesný rod, negace, stupeň a varianta [6].

1.3.2 Metody analýzy textu

Samotná analýza textu je v procesu text miningu nejdůležitější část. V této sekci budou stručně popsány nejznámější metody, které se pro analýzu používají.

1.3.2.1 Kategorizace

Úkolem kategorizace dokumentů je klasifikovat daný textový dokument do předem stanovené množiny kategorií [3]. To znamená, že zkoumaný dokument je možné zařadit do žádné, jedné nebo i více kategorií. Kategorizace se používá například pro filtrování spamu v e-mailových schránkách, automatické generování metadat, detekci témat (např. webových stránek či novinových článků) atd.

Kategorizaci je možné provádět dvěma způsoby. První způsob je přístup pomocí znalostního inženýrství, kdy znalostní inženýr za pomoci doménového experta vytvoří systém (tzv. *klasifikátor*) založený např. na rozhodovacích pravidlech. Ta jsou manuálně vytvořena pro jednotlivé kategorie na základě doménových znalostí o klasifikovaných dokumentech [4]. Druhý způsob vytváření klasifikátorů je založen na metodách strojového učení. Klasifikátor je tak vytvořen automaticky pomocí induktivního procesu na základě množiny manuálně klasifikovaných dokumentů [3].

1.3.2.2 Shlukování

Problém shlukování se dá definovat jako nalezení skupin podobných objektů v kolekci dat. Podobnost mezi objekty se měří pomocí podobnostních funkcí [7]. Výsledkem shlukování jsou skupiny dokumentů, kde dokumenty uvnitř jedné skupiny si jsou co nejpodobnější a zároveň dokumenty v různých skupinách jsou si podobné co nejméně [4].

Úlohy shlukování a kategorizace jsou si svým způsobem podobné. V kategorizaci (viz sekce 1.3.2.1) jsou známé kategorie dokumentů a úkolem je správně určit, do jaké kategorie (či do jakých kategorií) bude patřit nový dokument. Při shlukování je naopak k dispozici kolekce dokumentů, ze kterých je třeba vytvořit shluky navzájem podobných objektů bez jakékoli předem známé informace o příslušnosti k nějakým třídám [4].

Algoritmů pro shlukování existuje více, jsou to například hierarchické metody, k-means clustering, pravděpodobnostní shlukování, biologicky inspirované metody a další [4].

1.3.2.3 Shrnutí dokumentu

Metoda shrnutí dokumentu (neboli *text summarization*) vytvoří stručný a plynulý souhrn obsahující klíčové informace zadaného textu. Jako vstup je zadán buď samotný dokument nebo skupina vzájemně souvisejících dokumentů. Algoritmus v textu identifikuje nejdůležitější věty a zřetězí je tak, aby vytvořily konečný souhrn. Rozhodnutí o tom, jaká část dokumentu je důležitá, se řídí primárně vstupem od uživatele [7]. Tato metoda se tedy hodí v případě nutnosti zpracování velkého množství textu v krátkém čase, kdy není v lidských silách přečíst a pochopit celý nezkrácený text [8].

1.3.2.4 Analýza sentimentu

Analýza sentimentu je klasifikační problém, kdy se daný text na základě názorů a emocí v něm obsažených zařadí do jedné ze tří kategorií — pozitivní, negativní nebo neutrální. Sentiment je možné analyzovat z hlediska celého zkoumaného textu (článek, recenze, komentář…), ale také z hlediska jednotlivých vět, což je přesnější varianta. Kromě toho je také výhodnější v případě, že se zkoumá sentiment věty vůči entitám v ní obsažených.

Dnes je analýza sentimentu velmi praktická a oblíbená metoda. Najde využití především v komerci na webu, kde uživatelé hodnotí nejrůznější produkty a služby, ať již v e-shopech, recenzích na blozích, na sociálních sítích či na webech přímo určených k hodnocení produktů. Poskytovatelé a výrobci často chtějí znát názor odběratelů na jejich služby či produkty, stejně tak ostatní uživatelé chtějí znát názory jiných na služby či výrobky, které plánují využít či zakoupit.

Spolehlivá automatická analýza sentimentu má rovněž jistou výhodu nad určováním sentimentu lidmi anotátory. Sentiment se dá totiž považovat za poměrně subjektivní a více různých anotátorů se nemusí vždy shodnout na výsledku. To platí zvlášť pokud texty nejsou lehce rozhodnutelné. Kromě toho je známý fakt, že interpretace textů lidmi závisí na jejich názorech. Například lidé věnují větší pozornost názorům, které se shodují s jejich vlastními, což může vést k určitému zkreslení výsledně určeného sentimentu. Rovněž pokud je třeba anotovat vyšší objem dokumentů, kvalita odvedené práce anotátorů klesá [7].

Pro určení sentimentu textu existují dvě hlavní metody. První z nich je tzv. *slovníková metoda*, kdy se ve zkoumaném textu hledají emočně zabarvená slova. Podle toho, zda se v textu nacházejí slova spíše negativně nebo spíše pozitivně zabarvená se určí výsledný sentiment celého dokumentu. Je tedy třeba mít k dispozici seznam emočně zabarvených slov. Druhá možnost je metoda strojového učení a využití některého z *klasifikátorů*. Pro použití této metody je třeba mít k dispozici sadu trénovacích dat skládající se z množiny předem klasifikovaných dokumentů.

1.3.2.5 Rozpoznání jazyka

Jak už název napovídá, smyslem této metody je rozpoznání jazyka, ve kterém je napsaný vstupní text. „*Toho je možné dosáhnout vybudováním tabulek s frekvencemi charakteristických dvojic a trojic písmen.*“ [9] V tomto řešení mohou být problémové krátké texty, ve kterých nemusí být obsažen dostatek podkladů pro správné rozpoznání jazyka [9].

1.3.2.6 Extrakce informací

„*Extrakce informací je identifikace a následná klasifikace specifické informace nalezené v nestrukturovaných zdrojích (jako například v textových zdrojích v přirozeném jazyce) do sémantických kategorií tak, aby se daná informace stala vhodnější pro další zpracování.*“ [4]

Systémy extrakce informací jsou použitelné, pokud splňují následující podmínky:

- extrahovaná informace je specifikována explicitně a není nutné žádné další odvozování,

- k sumarizaci relevantních částí dokumentu stačí malé množství šablon,
- požadovaná informace je v textu vyjádřena relativně lokálně [3].

Cílem procesu je nalézt v obsahu dokumentu konkrétní objekty (neboli *entity*), jejich vlastnosti a smysluplné vztahy mezi nimi [4].

Existují 4 základní typy prvků, které se dají extrahovat [3].

- *Entity* jsou základními stavebními bloky, které lze najít v dokumentu. Mohou to být osoby (Tomáš Garrigue Masaryk, Bedřich Smetana...), místa (Česká republika, Praha, Krkonoše...), předměty (HP Probook 6460b...), organizace (Google, OSN...), geny, léky a další.
- *Atributy* jsou vlastnosti extrahovaných entit. Mohou to být například titul či věk osoby, typ organizace, souřadnice geografické polohy apod.
- *Fakty* jsou vztahy, které existují mezi entitami. Pod tím je možné si představit například pracovní vztah firmy a zaměstnance.
- *Události* jsou činnosti nebo výskyty zájmu, kterého se entita účastní. Příkladem mohou být data narození, sloučení firem či vývoj nějakého produktu.

Extrahovat informace z textu je možné jednak metodami znalostního inženýrství a jednak metodami strojového učení. Pro použití metod znalostního inženýrství je zapotřebí znalostního inženýra (doménového experta) a programátora. Způsob je založen na manuálním vytváření gramatik (většinou pravidel) použitých jako komponenty systému pro extrakci informací. Vytvoření systému je často náročné a jeho změna složitá, metoda je ale spolehlivější oproti druhému způsobu. Při použití metod strojového učení je systém pro extrakci informací automaticky vygenerovaný na základě předem anotovaných trénovacích dat. V tomto případě není potřeba doménový expert, je ale třeba velké množství trénovacích dat, které se v případě změny systému musí přeznačkovat [4].

Rozpoznávání pojmenovaných entit Rozpoznávání pojmenovaných entit (angl. *Named Entity Recognition*, zkráceně NER) je základní fáze v procesu extrakce informací. „*Za pojmenované entity jsou považována slova a slovní spojení, která v textu vystupují jako jména osob, geografické názvy, jména produktů, názvy organizací, ale také jako časové údaje apod.*“ [10] Systém musí nejdříve identifikovat entity v nestrukturovaném textu a následně je přiřadit do správné kategorie, jako například osoba, organizace, lokace apod.

Rozpoznávání entit je v extrakci informací nejdůležitější krok. Závisí na něm totiž i další kroky, jako je extrakce relací nebo událostí, které se entit týkají. Slouží tedy i jako předzpracování textu pro náročnější úlohy extrakce informací. [7]

Extrakce relací Extrakce relací (angl. *relation extraction*) v textu s již rozpoznanými entitami identifikuje relace (vztahy) mezi dvojicemi těchto entit. Například z věty „*Larry Page je spoluzakladatel Googlu.*“ je možné extrahovat vztah `zakladatelSpolečnosti(Larry Page, Google)` [7].

Stejně jako u rozpoznávání pojmenovaných entit jsou definovány typy entit, i zde jsou definovány typy vztahů mezi entitami. Program ACE (*Automatic Content Extraction*) definuje množinu hlavních relací a jejich podtypů. Jsou to například relace *fyzické* (entita je fyzicky v blízkosti jiné entity), *osobní/sociální* (například osoba je rodinným příslušníkem jiné entity) či *zaměstnání/příslušnosti* (například osoba je zaměstnaná organizací).

1.4 Crawler

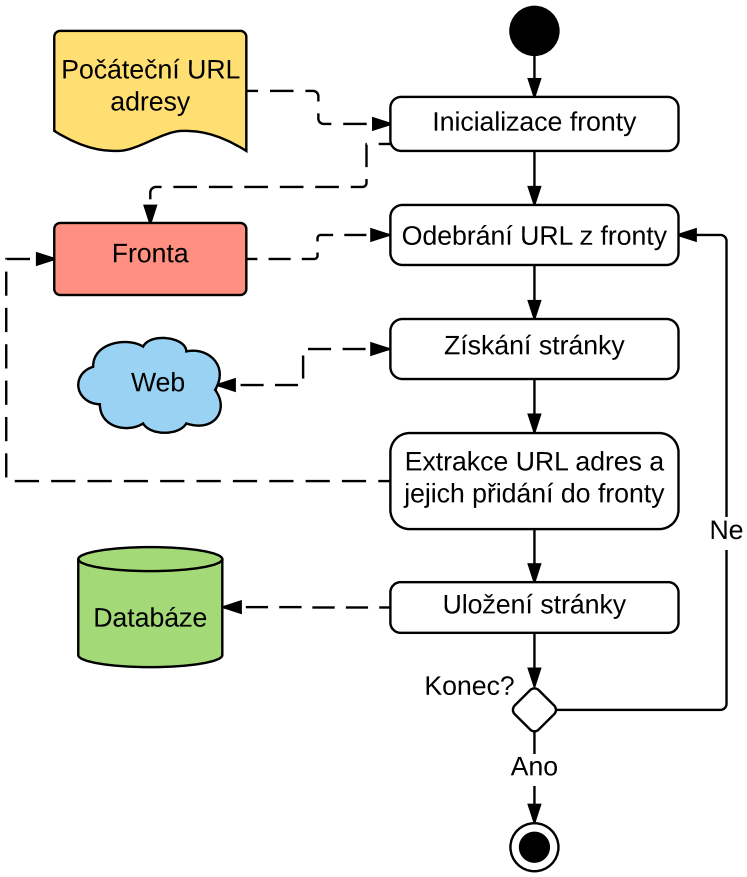
Crawler je program, který prochází kolekci webových dokumentů propojenou pomocí hyperlinků. Na proces je možné nahlížet jako na algoritmus prohledávání grafu, kdy uzly jsou jednotlivé webové stránky a hrany mezi nimi jsou hypertextové odkazy. Proces sekvenčního crawleru je zobrazen na obrázku 1.3.

Vstupem do procesu crawlování je jedna nebo více URL adres (anglicky se jim říká *seed pages*). Crawler inicializuje frontu nenavštívených URL adres, kam vloží tyto vstupní adresy. Poté odebere vždy jednu adresu z fronty a stránku na této adrese stáhne z webu pomocí HTTP. Její obsah projde a identifikuje všechny hyperlinky, které opět uloží na konec fronty. Staženou stránku pak uloží do uživatelem zvoleného úložiště. Proces končí, pokud již byl stažen předem definovaný počet stránek, jinak pokračuje stažením další stránky z fronty. Ve výjimečných případech může crawler skončit i v případě, že je fronta prázdná. Vzhledem k množství odkazů na webových stránkách se to stává ale velmi zřídka [1].

V závislosti na implementaci crawleru je možné definovat další nastavení. Jednou ze základních možností je hloubka zanoření, která udává maximální délku v grafu stránek od originální stránky, kterou je možno projít pro získání nové adresy. Dalším možným nastavením je filtrování URL adres pomocí regulárních výrazů před tím, než se vloží do fronty. Pokud adresa regulárnímu výrazu nevyhovuje, do fronty se nevkládá a tedy nebude ani stažena. Mezi další nastavení crawlerů patří například možnost nastavit intervaly, po kterých se posílají dotazy na server (aby servery nebyly zahlceny požadavky crawleru), filtrování stránek podle domény atd.

1.5 Detekce relevantního obsahu na webu

Podstatnou částí v procesu content web miningu je získání relevantního obsahu ze stažených webových stránek. V dnešní době je s tímto úkolem spojeno několik problémů, které jeho řešení značně znesnadňuje. V této sekci bude



Obrázek 1.3: Proces sekvenčního crawleru [1]

vysvětleno, jaké nesnáze detekce relevantního obsahu přináší a jak je možné je řešit.

1.5.1 Struktura webových stránek

Webové stránky jsou prohlížeči většinou posílány jako HTML (*HyperText Markup Language*) nebo XHTML (*Extensible HTML*) dokumenty. Ty mohou být v (X)HTML buď přímo napsány, nebo mohou být vygenerovány webovou aplikací na serveru. Stránky se dělí na *statické*, které mají neměnný obsah, a *dynamické*, které se s časem mění. Kromě (X)HTML může webová stránka obsahovat i další prvky, jako jsou kaskádové styly (CSS), skripty (většinou JavaScript) či multimediální soubory (obrázky, videa...).

1.5.1.1 (X)HTML

HTML je značkovací jazyk používaný k tvorbě webových stránek, aktuální verze je HTML5. Jazyk je definován množinou značek (tzv. *tagů*) a jejich atributů, obojí je specifické pro danou verzi. Názvy značek se uzavírají do ostrých závorek, tedy `<` a `>`. Značky existují buď párové, kdy koncová značka má stejný název jako počáteční, jen se před její název přidá lomítko (např. `<em>Text kurzívou.</em>`), a nepárové (např. `<br>`). Značky se do sebe mohou vnořovat, např. `<p>Odstavec a <strong>text tučně</strong>.</p>`, a díky tomu pak tvoří stromovou strukturu. Ukázku je možné si prohlédnout na ukázce kódu 1.2.

Ukázka kódu 1.2: Ukázka HTML [11]

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Page Title</title>
5    </head>
6    <body>
7
8      <h1>This is a Heading</h1>
9      <p class="red">This is a paragraph.</p>
10
11   </body>
12 </html>
```

Původní verze jazyka XHTML pouze přenesla již existující HTML značky z verze 4.01 do standardů XML tak, aby vyhovovaly normě XML. To znamená například nutnost uzavírání tagů či psaní tagů malými písmeny [12]. Jelikož verze HTML 4.01 měla být poslední a HTML se již dále vyvíjet nemělo, předpokládalo se, že XHTML nahradí HTML. V roce 2014 byla ale vydána nová verze HTML, a to HTML5.

1.5.1.2 DOM

Dokumentový objektový model (anglicky *Document Object Model*, zkráceně DOM) je standard W3C (*World Wide Web Consortium*) pro přistupování k dokumentům. DOM je platformně a jazykově nezávislé rozhraní, které umožňuje programům a skriptům dynamický přístup a aktualizaci obsahu, struktury a stylu dokumentu [13].

DOM je rozdělen do tří kategorií [13]:

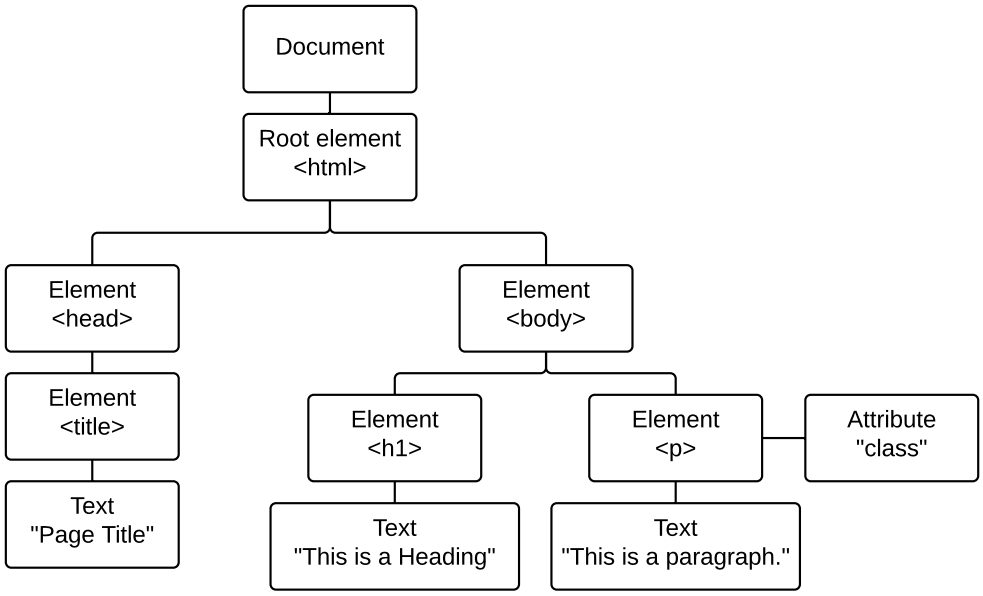
- Core DOM — standardní model pro dokumenty všech typů,
- XML DOM — standardní model pro XML dokumenty,
- HTML DOM — standardní model pro HTML dokumenty.

HTML DOM je standardní objektový model a programové rozhraní pro HTML. Definuje[13]:

- HTML elementy jako objekty,
- vlastnosti všech HTML elementů,
- metody pro přístup ke všem HTML elementům,
- události pro všechny HTML elementy.

Jinými slovy HTML DOM je standard, který definuje, jak získávat, měnit, přidávat a mazat HTML elementy [13].

Jakmile se načte webová stránka, prohlížeč vytvoří dokumentový objektový model stránky. HTML DOM je zkonstruován jako strom objektů. Na obrázku 1.4 je zobrazen dokumentový objektový model HTML kódu zobrazeného na ukázce kódu 1.2. Pomocí DOM je pak možné například změnit text elementu `h1`, přidat za něj jiný element, smazat element `p`, zjistit jeho textovou hodnotu a mnoho dalších operací.



Obrázek 1.4: Dokumentový objektový model HTML stránky z kódu 1.2

1.5.1.3 Microdata

Postupem času vzniklo několik možností, jak přidat anotace do HTML kódu pro lepší strojovou čitelnost webových stránek. Microdata jsou jedním z nich.

Jsou součástí stadardu HTML5, který rovněž obsahuje slovníky microdat a globální atributy [14].

HTML se anotuje pomocí atributů, které lze přidat k jakémukoli HTML elementu. Atributy mohou být následující [14]:

- *Itemscope* značí element, jehož potomci obsahují microdata vlastnosti.
- *Itemtype* definuje odkaz na slovník, ve kterém je popsána položka (angl. *item*) a její vlastnosti (angl. *properties*).
- *Itemid* je globální identifikátor položky (URI).
- *Itemprop* je termín ze slovníku, jehož hodnota se nachází v obsahu elementu.
- *Itemref* je reference na jinou položku nacházející se ve stejném dokumentu.

V ukázce kódu 1.3 je zobrazena anotace krátkého článku pomocí microdat.

Ukázka kódu 1.3: Ukázka anotace článku pomocí microdata

```
1 <article itemscope itemtype='https://schema.org/
    NewsArticle'>
2   <h1 itemprop='headline'>Nadpis článku</h1>
3   <date itemprop='datePublished'>20. dubna 2016</date>
4   <div itemprop='articleBody'>
5     <p>Lorem ipsum dolor sit amet, consectetuer
        adipiscing elit. Curabitur dictum gravida mauris
        .</p>
6   </div>
7 </article>
```

1.5.2 Problémy při získávání obsahu z webových stránek

Jakmile je v procesu web miningu stažena webová stránka, zdaleka to neznamená, že je získávání dat ukončeno. Tento dokument se totiž vůbec nemusí nacházet v potřebném formátu a v drtivé většině případů ho po stažení čeká ještě proces předzpracování dat, který stažený HTML dokument transformuje do použitelné podoby.

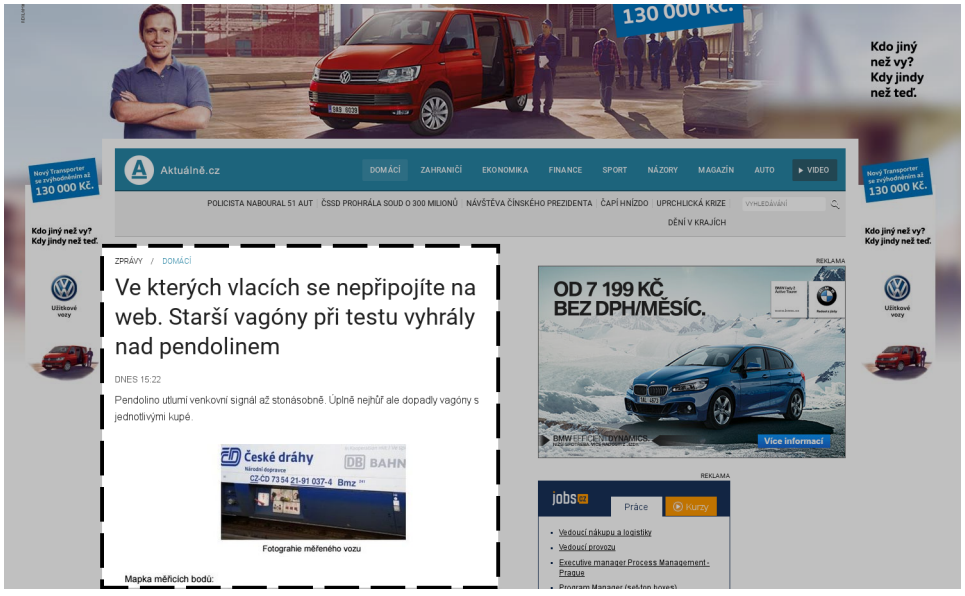
1.5.2.1 Nepotřebné prvky webové stránky

Běžná webová stránka většinou obsahuje následující části, ačkoli to není žádným pravidlem.

- *Hlavička*, která obsahuje většinou logo webu nebo firmy, o které stránka pojednává, hlavní menu webu a nějakou grafiku. Dále v ní může být umístěno i vyhledávání, informace o přihlášeném uživateli (nebo formulář k přihlášení, pokud žádný uživatel není přihlášen), výběr jazykové mutace atd.
- *Postranní panel* často slouží pro umístění vedlejšího menu či jiných odkazů na další stránky webu, například nejčtenější nebo nejoblíbenější příspěvky.
- *Hlavní obsah* je část, kde se nachází relevantní obsah stránky.
- *Patička* obsahuje většinou informace o webu, jako například kontakt na provozovatele webu či copyright. Může se tam nacházet i seznam odkazů na nejdůležitější stránky na webu apod.

Nedílnou součástí mnoha stránek je i reklama, jelikož je to často hlavní způsob, kterým si provozovatelé webů vydělávají. Může se nacházet prakticky v jakékoli části webové stránky, často i na více místech.

Kromě hlavního obsahu se tedy na webové stránce nachází mnoho dalších informací, které pro účely web miningu nejen, že nejsou třeba, ale většinou i škodí. Relevantní obsah je tedy třeba na stránce identifikovat a vyčistit ho od ostatního obsahu, což nemusí být vždy zrovna jednoduchá úloha.



Obrázek 1.5: Ukázka článku[1] z webu Aktualne.cz

[1] http://zpravy.aktualne.cz/domaci/foto-v-kterych-vlacich-se-nepripojite-na-web-stare-vagony-pu/r a122052806e911e6b5c3002590604f2e/

Na obrázku 1.5 je zobrazen úryvek článku ze serveru Aktualne.cz, kde je možné si všimnout, že relevantní obsah z tohoto úseku je pouze rubrika, nadpis, datum a pár vět a obrázků pod nadpisem, zbytek jsou části šablony webu a hlavně reklamy.

1.5.2.2 Změny struktury webové stránky

Pokud je již proces vyčištění relevantního obsahu od toho nepotřebného hotový, je nutné si dávat pozor na změnu ve struktuře (HTML kódování) sledované webové stránky. Pokud je použitý postup na nakódování webu příliš závislý, mohl by se po změně layoutu stránky celý algoritmus sesypat nebo najít úplně jiný obsah, než ten požadovaný. Je tedy třeba buď najít takový algoritmus, který nebude na struktuře webu úplně závislý, nebo sledovaný web kontrolovat, zda se nezměnil.

1.5.2.3 Kódování stránky

Dalším problémem při získávání obsahu z webu může být různé kódování webových stránek. Je tedy potřeba buď stránky získat v preferovaném kódování (většinou UTF-8) a nebo do něj stažené stránky převést.

V českém prostředí se dnes vyskytují hlavně kódování UTF-8, Windows-1250 a ISO 8859-2 [15]. Kódování webové stránky se dá zjistit dvěma způsoby, a to z HTTP hlavičky v odpovědi serveru, a z kódu HTML stránky. Některé weby informují o svém kódování oběma způsoby, jiné pouze jedním — s tím je třeba počítat a nespoléhat se pouze na jeden z nich.

Pro zjištění kódování dokumentu pomocí HTTP odpovědi je možné použít hlavičku *Content-Type*, kde v případě, že se jedná o textový obsah jako je např. `text/html` či `text/plain`, může být za středníkem uveden parametr *charset* sdělující, o jaké se jedná kódování, například `text/html; charset=UTF-8` [16].

Pro zjištění kódování přímo z obsahu webové stránky by v hlavičce HTML (`<head>...</head>`) měl být tag *meta* s parametrem *charset*, například `<meta charset="utf-8"/>` [17]. Alternativně se někdy používá tag *meta* s parametrem *content*, jehož hodnota je stejná, jako hodnota hlavičky *Content-Type*, například tedy `<meta content="text/html; charset=utf-8" http-equiv ="Content-Type">`.

Existuje také HTTP hlavička *Accept-Charset*, která umožňuje poslat serveru v požadavku seznam kódování podporovaných klientem uživatele. V praxi se ale může stát, že server tuto žádost ignoruje a pošle obsah v jiném kódování. Na tuto možnost tedy není možné se spolehnout a je třeba kódování ověřovat.

1.5.2.4 JavaScript

Při získávání obsahu z webu je také nutné dát si pozor na to, že některé části stránky se mohou dynamicky načítat pomocí JavaScriptu až po načtení

hlavního obsahu stránky. Například při požadavku na stránku, která obsahuje článek, se může načíst veškerý obsah kromě komentářů ke článku. Ty se načtou pomocí JavaScriptu až po tom, co se uživatel dostane na místo, kde se komentáře mají nacházet. Když ale robot stahuje stránky, tak stáhne jen to, co mu vrátí server na původní požadavek, což je stránka, kde budou úplně chybět komentáře.

1.5.3 Způsoby identifikace relevantního obsahu

Cílem identifikace relevantního obsahu je získat z webové stránky čistý text či text v jiném formátu vhodném pro nástroje text miningu, které ho budou zpracovávat. K identifikaci je možné využít i toho, že webové stránky jsou napsány ve strukturovaném HTML. Na konci procesu je pak ale nutné HTML značky odstranit.

Způsobů pro extrakci relevantních informací z webové stránky je mnoho, zde popíši pouze vybrané možnosti.

1.5.3.1 Manuální extrakce

Nejjednodušší možností, jak získat hlavní obsah webové stránky, je manuální extrakce. V HTML kódu stránky se manuálně identifikují prvky, které v sobě nesou hlavní obsah nebo jeho části. Poté je možné napsat program, který s využitím DOM extrahuje text z identifikovaných elementů. Metoda je velice spolehlivá a jednoduchá na implementaci. Její nevýhodou ale je úplná závislost na způsobu nakódování webové stránky. Pokud se totiž její struktura změní (například kvůli změně designu), je nutné program upravit nebo i napsat úplně znovu.

Na ukázce kódu 1.4 je zobrazen kód v Node.js, který pomocí manuální extrakce zjistí a vypíše hlavní nadpis článku ze serveru Novinky.cz.

Ukázka kódu 1.4: Ukázka manuální extrakce nadpisu článku na serveru Novinky.cz

```
1 var jsdom = require('jsdom');
2
3 jsdom.env(
4   'http://www.novinky.cz/krimi/401049-hasici-na-
      ceskolipsku-likvidovali-pozar-special nim-vodnim-
      paprskem.html',
5   ['http://code.jquery.com/jquery.js'],
6   function (err, window) {
7     console.log(window.$('#articleHeaderBig h1').text()
        );
8   }
9 );
```

1.5.3.2 Microdata

Pokud má extrahovaná stránka obsah anotovaný pomocí vyhovujících microdat, je možné tyto anotace použít pro extrahování potřebných částí webu. Způsob je podobný manuální extrakci, není ale tolik závislý na struktuře dokumentu, jelikož stejné značky může používat více webových stránek a pokud se stránka jakkoli změní, je poměrně pravděpodobné, že i v nové verzi budou data stále anotována pomocí microdat.

1.5.3.3 Rozdělení podle vizuálních stop

Metoda k nalezení bloků s relevantním obsahem využívá vizuální informace o webové stránce, které se o každém HTML elementu dají zjistit pomocí webového prohlížeče. Například Internet Explorer poskytuje API, přes které se dají zjistit souřadnice každého prvku. Pomocí strojového učení pak lze natrénovat model podle umístění a vzhledu prvků použitých pro uchovávání hlavního obsahu. K tomu je samozřejmě potřeba množina ručně anotovaných trénovacích dat [1].

1.5.3.4 Tree matching

Tato metoda je založená na pozorování, že komerční webové stránky používají nějaký druh neměnných šablon. Metoda se proto snaží objevit takto skryté šablony. Jelikož značky v HTML je možné vnořovat do sebe, je pak snadné vytvořit ze stránky stromovou strukturu. Pro nalezení skrytých šablon se pak porovnávají takto vytvořené stromové struktury více stránek ze stejného webu. Jakmile je nalezena šablona, je možné identifikovat bloky s hlavním obsahem. Ty se totiž budou na každé stránce značně lišit, zatímco ostatní prvky šablony (například menu, logo, reklamy…) webu si budou velice podobné [1].

1.5.3.5 Wrappery

V případě, že sledované webové stránky mají některé společné poznávací znaky, dle kterých by bylo možné identifikovat a extrahovat hlavní obsah ze všech stránek stejným postupem, je možné použít wrapper. Na základě analýzy struktury všech webových stránek, ze kterých se budou extrahovat data, lze vytvořit soubor pravidel, pomocí kterých bude rozpoznán relevantní obsah na každé z cílových webových stránek [18].

Tento způsob se hodí pro skupiny stránek zabývající se stejným tématem, jelikož takové stránky mají často velmi podobnou strukturu a liší se jen v detailech. Typickým příkladem může být skupina zpravodajských webů, dále i blogy, hodnotící servery (pro filmy, knihy…), fóra apod.

Kapitola 2

Analýza zpravodajských serverů

V této kapitole seznámím čtenáře s nejznámějšími českými zpravodajskými servery a jejich návštěvností. Dále se budu zabývat analýzou struktury zpravodajských webů a informací, které se z nich dají získat. Poslední část kapitoly bude informovat o možnostech analýzy vydolovaných textových informací.

2.1 České zpravodajské servery

Zpravodajských serverů dnes existuje na českém internetu velké množství, ať už těch, které se zabývají každodenními událostmi u nás i v zahraničí, nebo serverů zaměřených na publikování informací z určitého okruhu zájmů, jako je sport, technika apod. Hodně serverů tyto zájmové oblasti také spojuje a svému publiku nabízí zpravodajství z více kategorií. Tato práce je zaměřena na zpravodajství každodenních událostí, do čehož nebude počítáno bulvární zpravodajství.

Velký vliv na české zpravodajství má Česká tisková kancelář (ČTK), což „*je národní tisková a informační agentura, jejímž posláním je poskytovat objektivní a všestranné informace pro svobodné vytváření názorů*“ [19]. Články z ČTK přebírá mnoho českých zpravodajských serverů, mimo jiné i ty nejčtenější. ČTK sama provozuje vlastní zpravodajský server *CeskeNoviny.cz*.

Dále se na českém webu nacházejí

- zpravodajské servery tištěných deníků, jako jsou *iDnes.cz*, *Lidovky.cz*, *Denik.cz* či *IHNED.cz*,
- zpravodajské servery internetových portálů, tedy například *Novinky.cz*, *Aktualne.cz* a *tiscali.cz*,
- zpravodajské servery rozhlasových a televizních stanic, např. *rozhlas.cz* či *CeskaTelevize.cz* [20], a
- ostatní zpravodajské servery, například *ParlamentniListy.cz*, *E15.cz* ad.

Z tabulky 2.1, kde je zobrazen seznam zmíněných a některých dalších českých zpravodajských serverů seřazený podle návštěvnosti za březen 2016, je vidět, že zdaleka nejnavštěvovanější český zpravodajský server jsou *Novinky.cz*. Na druhém místě se pak nachází server *iDnes.cz* a s velkým propadem je na třetím místě server *Aktualne.cz*.

Tabulka 2.1: České zpravodajské servery a jejich návštěvnost za březen 2016 podle údajů ze serveru NetMonitor.cz [21]

Novinky.cz	122 923 076
iDnes.cz	87 905 521
Aktualne.cz	19 952 937
CeskaTelevize.cz	12 526 335
Denik.cz	10 792 786
Lidovky.cz	9 234 547
ParlamentniListy.cz	7 471 708
tiscali.cz	5 929 990
iHned.cz	5 424 392
rozhlas.cz	3 392 446
E15.cz	3 256 206
Tyden.cz	2 801 554
EuroZpravy.cz	2 525 321
Echo24.cz	1 762 257
CeskeNoviny.cz	909 761

2.2 Analýza obsahu českých zpravodajských webů

Jelikož české zpravodajské servery zastupují jednu zájmovou skupinu, druh informací, které se z nich dají získat, se napříč jednotlivými servery mnoho neliší. V této sekci popíši, jaké informace se na jednotlivých stránkách zpravodajských serverů nachází, a jaké z těchto informací by mohlo být užitečné získat pro analýzu.

2.2.1 Druhy stránek

Při vstupu prakticky na jakýkoli zpravodajský server se návštěvník dostane na hlavní stranu. Ta většinou obsahuje kromě základních částí webové stránky (logo webu, menu, patička, reklamy...) výběr nejpodstatnějších článků za poslední dobu, seznam nejnovějších článků ze všech rubrik či pro každou rubriku zvlášť, seznam rubrik a další informace. Některým z odkazů na hlavní stránce je možné přejít na stránku s výpisem článků z dané rubriky nebo na konkrétní článek.

Zpravodajské servery zařazují své články do rubrik, jako jsou *domácí*, *zahraniční*, *ekonomika*, *sport* atd. Díky menu je pak možné vypsat si pouze články spadající do určitě rubriky. Některé servery dále článkům přiřazují tzv. *štítky* neboli *klíčová slova*. Těmi by měla být popsána témata, kterými se článek zabývá, například názvy zemí, osob, organizací apod. Pokud server klíčová slova podporuje, nabízí pak často možnost zobrazit si pouze články, které mají přiřazený daný štítek a nebo při kliku na daný štítek zobrazí informace o něm.

Některé zpravodajské servery nabízejí návštěvníkům také archiv článků, kde je možné si zobrazit články za určité období. Jiné zase spoléhají pouze na stránkování článků v daných rubrikách.

Důležitým typem stránky zpravodajských serverů je stránka s vlastním článkem, která obsahuje informace, kvůli kterým lidé zpravodajské servery navštěvují. Jejím obsahem se budu zabývat v další sekci.

2.2.2 Obsah stránky s článkem

Na stránkách s článkem existuje pro zpravodajské servery mnoho společných prvků, ale i různých. Dalo by se ale říci, že obecná struktura zpravodajských článků je stejná. Jako názorný příklad v této sekci může posloužit článek ze serveru *iDnes.cz* ukázaný na obrázku 2.1 nebo článek ze serveru *Novinky.cz* zobrazený na obrázku 2.2.

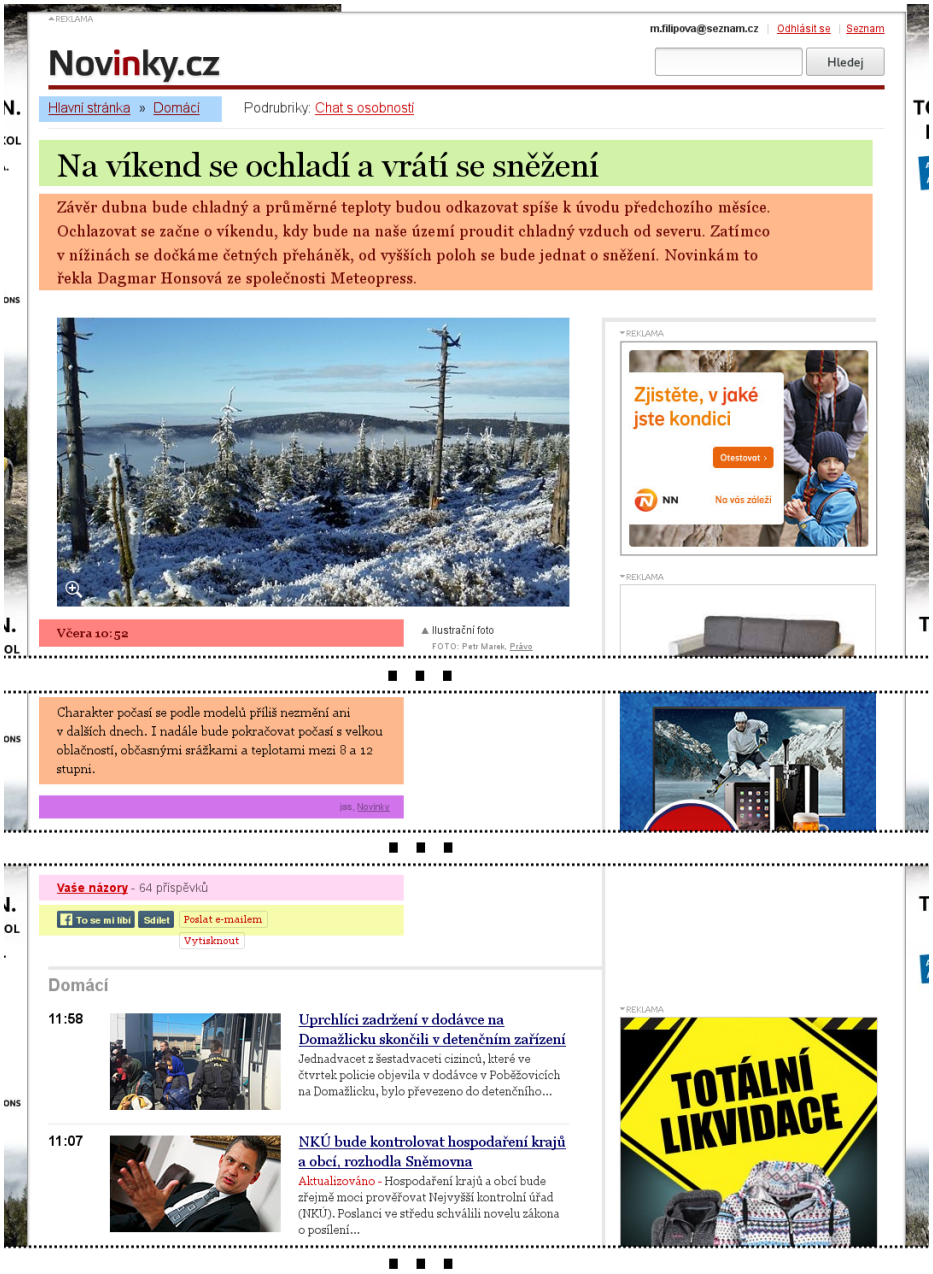
Každá stránka obsahuje nejdříve hlavičku na daném serveru společnou i pro další typy stránek. Informace uvnitř ní se mohou mírně lišit, často je v ní uváděno menu či vyhledávání, některé servery zde uvádějí aktuální datum, kdo má ten den svátek apod. Z menu může být také zřejmá rubrika, ve které je článek zařazen. Na obrázku 2.1 je vidět, že server *iDnes.cz* má hlavičku poměrně bohatou a nachází se na ní všechny výše zmíněné prvky, naopak z obrázku 2.2 je zřejmé, že server *Novinky.cz* mají hlavičku minimalistickou.

Další informace, která se většinou na stránce s článkem nachází, je již samotný článek. Někdy mezi hlavičkou a prostorem s článkem mohou být upoutávky na další články (jako je vidět i na obrázku 2.1), nebývá to ale běžné. Struktura článku se již na různých serverech mírně liší. Každý článek má ale svůj nadpis, datum publikace, obsah a autora.

Obrázek 2.1: Ukázka článku[2] z webu iDnes.cz

Jako hlavička článku by se daly označit informace zahrnující nadpis, datum, úvodní obrázek a úvodní odstavec obsahu. Nadpis bývá většinou umístěn jako první (tak, jako je vidět na obou obrázcích, zeleně zvýrazněno), nemusí to ale platit úplně vždy — např. server *Lidovky.cz* má jako první informace uvedené datum a rubriku článku, nadpis je umístěn až pod nimi. Vždy až pod nadpisem je umístěn první odstavec z obsahu článku (na obrázcích vy-

[3]http://zpravy.idnes.cz/predpoved-pocasi-0eg-/domaci.aspx?c=A160421_064236_domaci_san



Obrázek 2.2: Ukázka článku[3] z webu Novinky.cz

značen oranžovým zvýrazněním) a úvodní obrázek (je vidět pouze na obrázku 2.2). Úvodní odstavec může být jak nad obrázkem (*Novinky.cz*, *iDnes.cz*, *Denik.cz*...), tak pod ním (*Aktualne.cz*, *Lidovky.cz*...). V hlavičce se nachází i

[3] http://www.novinky.cz/domaci/401110-na-vikend-se-ochladi-a-vrati-se-snezeni.html

datum publikace článku (v obrázcích zvýrazněno červeně). To je často umístěno přímo pod nadpisem (*iDnes.cz*, *Aktualne.cz*, *ParlamentniListy.cz*...), ale může být i pod úvodním obrázkem (*Novinky.cz*, *Denik.cz*...) nebo dokonce nad nadpisem (*Lidovky.cz*).

Po úvodních informacích o článku již následuje jeho obsah (na obrázcích 2.1 a 2.2 zvýrazněno oranžově), kde se kromě běžných odstavců mohou nacházet i podnadpisy, obrázky a videa, méně často pak tabulky či graficky odlišené bloky se souvisejícím obsahem. Některé servery umisťují do obsahu článku i odkazy na jiné články, většinou zabývající se stejným tématem.

Na konci článku se pak vždy nachází autor článku (na obrázcích 2.1 a 2.2 zvýrazněn fialově). Pokud server podporuje štítky, může se pod článkem nacházet seznam štítků přiřazených k článku (na obrázku 2.1 zvýrazněno modře), existují ale i výjimky, kdy je seznam štítků umístěn v hlavičce článku. Na konci hlavního obsahu bývají také umístěny komentáře, nebo alespoň odkaz na stránku s nimi (na obrázcích zvýrazněno růžově).

Někde v blízkosti článku bývají také umístěna tlačítka pro sdílení článku na sociálních sítích či e-mailem (na obrázcích zvýrazněno žlutě). Mohou být buď v hlavičce článku, na jeho konci, na více místech, nebo i po straně, kde se mohou pohybovat společně s tím, jak uživatel postupuje ve čtení článku. Celá stránka je pak proložena reklamami.

Pod článkem pak bývá mnoho odkazů na různé další články, ať už ze stejné rubriky či odjinud. Stejný obsah pak bývá v postranním menu, pokud jím stránka disponuje. Na konci stránky je pak patička společná pro celý web.

2.2.3 Výběr informací pro analýzu

Pro extrakci informací ze stránek obsahujících články je možné vybírat z několika prvků popsaných v předchozí sekci. Mimo ně je nutné si ke každému článku uložit URL, na které se nachází, a název serveru, ze kterého byl článek vyextrahován.

Pro analýzu článku jsou jistě stěžejními informacemi nadpis, datum zveřejnění a obsah, což jsou informace nacházející se na všech serverech. Další zajímavou informací na stránce jsou klíčová slova, která vhodně shrnují témata, o kterých článek pojednává. Nevýhodou je, že ne všechny servery tuto informaci uvádějí (zcela chybí např. na serveru *Lidovky.cz*), podporuje je ale většina. Klíčová slova by tak neměla chybět v extrahovaných informacích.

Další informace již pro analýzu zpravodajských článků nejsou natolik podstatné. Důležitou informací by byl jistě autor článku, v praxi se ale na tomto místě málokdy objevuje reálné jméno osoby. Velmi často je uvedena jako autor článku pouze ČTK, jindy je to název serveru, kde byl článek publikován, nebo pro návštěvníky celkem nic neříkající přezdívka autora. Další možnou informací pro extrakci je rubrika, pod kterou je článek zařazen. Tu je ale vzhledem k možnosti procházení článků podle rubrik možné ukládat při crawlování článků a není tak třeba ji extrahovat přímo z článku. Zajímavou položkou pro

extrakci by mohly být také komentáře ke článku, to je ale nad rámec této práce.

Z informací o článku na webové stránce bude tedy extrahován *název*, *datum*, *obsah* a *klíčová slova*. V procesu crawlování článků bude k těmto informacím doplněna *rubrika*, *název serveru*, odkud byl článek stažen, a *URL*, kde je článek umístěn.

2.3 Možnosti textové analýzy článků

V této části pro každou z metod text miningu uvedených v sekci 1.3.2 rozeberu možnosti využití pro novinové články v rámci této práce. Na závěr z nich vyberu metody vhodné pro implementaci v této práci.

Kategorizace Kategorizace by v případě novinových článků byla vhodná pro řazení článků do předem definovaného seznamu rubrik. Jelikož je ale rubrika každého článku určena již při jeho stažení, není v této práci kategorizace potřeba.

Shlukování Novinové články je možné shlukovat podle jejich obsahu, názvu a data pro více účelů. Jelikož všechny uvedené české zpravodajské servery referují o každodenních událostech, je velice pravděpodobné, že ke každé významné události bude vydán článek na každém serveru, a tyto články si budou velice podobné. Navíc většina serverů přebírá zprávy od ČTK, což podobnost článků ještě zvyšuje. Pomocí shlukování by bylo možné identifikovat články z různých serverů, které píší o stejné události. Další možností je doporučování článků na základě shlukování. Ve stejných shlucích pak skončí články píšící o stejných či podobných tématech. Oběma těmito problémy se již zabýval David Ešner ve své diplomové práci [4].

Shrnutí dokumentu Metoda shrnutí dokumentu se používá pro vytvoření souhrnu dlouhých dokumentů. Novinové články většinou mívají délku pouze pár odstavců, není tedy nutné je nijak zkracovat.

Analýza sentimentu Pomocí analýzy sentimentu je možné zjistit, zda je článek pozitivní, negativní či neutrální, což může být spolu s informací o klíčových slovech (a tedy tématech článku) užitečná informace. Důležitý je ale i způsob interpretace toho, co sentiment článku vlastně znamená. Nemusí být totiž vůbec jasné, kdo je původce sentimentu a na co se zjištěný sentiment vztahuje. Například pokud je analyzován článek, který obsahuje informaci o tom, že jeden politik se negativně vyjádřil o druhém, bude mít článek stejný

[4] Ešner, David. Chytré noviny: Shlukování novinových článků podle sémantických vztahů mezi nimi. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2013.

sentiment, jako kdyby v něm byla informace, že druhý politik se negativně vyjádřil o prvním. Kromě toho vzhledem k délce novinových článků je velmi pravděpodobné, že se v něm budou nacházet jak věty s negativním sentimentem, tak věty s pozitivním sentimentem. Bylo by tedy výhodnější zkoumat sentiment nejen celého článku, ale rovněž po větách.

Rozpoznání jazyka Jelikož je tato práce zaměřena výhradně na český jazyk a české zpravodajské servery, jsou všechny extrahované články v této práci pouze v češtině. Rozpoznávání jazyka tedy není třeba.

Extrakce informací Pomocí extrakce informací je možné v obsahu novinových článků rozpoznat entity, tedy např. osoby, státy či organizace, případně i jaké jsou mezi nimi vztahy. Pokud se entity podaří přiřadit k entitám sémantického webu (např. pomocí odkazu na web DBpedie.org), je možné odvodit i další informace o entitě.

Kapitola 3

Crawler zpravodajských článků

V této kapitole se budu věnovat analýze existujících crawlerů v Node.js a výběru nejvhodnějšího z nich pro crawlování článků ze zpravodajských serverů. Dále popíši návrh, implementaci a testování vlastního Node.js modulu stavícího na vybraném již existujícím nástroji.

3.1 Existující nástroje pro crawlování webu

Pro Node.js existuje crawlerů hned několik. Níže stručně popíši několik z nich.

js-crawler Modul vytvořený Antonem Ivanovem umožňuje crawlování stránek pomocí HTTP i HTTPS. Kromě počáteční URL dovoluje nastavit také maximální hloubku zanoření, maximální počet odeslaných požadavků za vteřinu, maximální počet souběžných požadavků, hlavičku *User-Agent* posílanou v požadavku, ignorování příbuzných adres či filtrování získaných URL adres ke crawlování. Poskytuje API pro předání callbacků, které se zavolají po dokončeném stažení jedné stránky, po dokončení crawlování a při chybě [22].

node-crawler Modul, který vytvořil Sylvain Zimmer, umožňuje definovat callback, který se zavolá pro každou staženou stránku, a další, který se zavolá, jakmile crawler skončil. Je možné nastavit timeout požadavku, maximální počet spojení, prioritu požadavku, počet pokusů pro opětovné spojení a dobu, jakou se má mezi pokusy čekat. V callbacku vrátí i modulem Cheerio (defaultně) či modulem JSDOM vytvořený dokumentový objektový model stažené stránky. Dále poskytuje cache a převedení kódování stažené stránky vždy na UTF-8. Modul již není dále udržován [23].

node-simplecrawler Ačkoli se z názvu může zdát, že modul, který vytvořil Christopher Giffard, bude obsahovat jen nejzákladnější možnosti crawlování, opak je pravdou. Poskytuje totiž velké množství různých nastavení od těch

základních, jako je nastavení hloubky zanoření, maximální počet souběžných požadavků či interval mezi požadavky, až po složitější jako je nastavení autentifikace nebo přidávání funkcí pro filtrování URL adres. Modul poskytuje API řízené událostmi, které pokrývají většinu z událostí crawlovacího procesu. Příkladem může být start crawleru, přidání položky do fronty, pokus o přidání duplikátu do fronty, dokončené stažení stránky, chyba při stažení stránky, skončení crawleru a mnoho dalších. Modul rovněž respektuje informace ze souborů `robots.txt` [24].

Spider Spider, vytvořený Mikealem Rogersem, je nejjednodušší ze zmíněných modulů. Umožňuje nastavit maximální počet spojení, hlavičku *User-Agent* při posílání požadavků a cache [25].

3.2 Návrh crawleru zpravodajských článků

V této části popíši návrh vlastního modulu pro získávání článků ze zpravodajských serverů. Modul bude založen na existujícím Node.js modulu pro crawlování webu.

3.2.1 Výběr existujícího crawleru

Z existujících nástrojů pro crawlování webů popsaných v sekci 3.1 jsem si pro vlastní modul pro crawlování českých zpravodajských serverů vybrala modul *node-simplecrawler*. Disponuje množstvím nastavení a je tak velice flexibilní a tedy vhodný jako základ vlastního modulu.

3.2.2 Výběr zpravodajských serverů

Pro sběr dat v této práci jsem vybrala ze serverů uvedených v sekci 2.1 weby *Novinky.cz*, *iDnes.cz*, *Aktualne.cz* a *ParlamentniListy.cz*. První tři byly vybrány z důvodu nejvyšší návštěvnosti, zároveň všechny tři v oblasti zpráv přebírají zprávy z ČTK a pouze k nim doplňují vlastní obsah. *ParlamentniListy.cz* pak byly vybrány jelikož kromě zpráv přebraných z ČTK v rubrice *Zprávy* publikují i články napsané politiky a rozhovory s nimi.

V dalším textu budu server *Novinky.cz* nazývat pouze Novinky, *iDnes.cz* pouze iDnes, *Aktualne.cz* pouze Aktuálně a *ParlamentniListy.cz* pouze jako PL či Parlamentní listy.

3.2.3 Vstup a výstup modulu

Crawler bude stahovat články z dané rubriky na vybraném zpravodajském serveru. Články se budou stahovat od nejnovějších po články publikované v určitém datu. Vstupem tedy bude *název zpravodajského serveru*, *rubrika* a *datum*. Staženy budou články s datem pozdějším nebo stejným jako zadané datum.

Vstupní parametr *server* bude moci nabývat hodnot *Novinky*, *iDnes*, *Aktualne* a *PL*. Pro parametr *rubrika* pak půjde vybírat jednu ze dvou možností, a to *domácí* a *zahraniční*, což jsou rubriky, které mají servery Novinky, iDnes a Aktuálně společné. Parlamentní listy nemají rubriku pro zprávy dále rozdělenou. Další rubriky by bylo nutné doimplementovat v závislosti na konkrétních rubrikách vybraného serveru. Vstupní parametr *datum* je pak objekt typu `Date`.

Vzhledem k vybranému crawleru bude na vstupu také zadán callback s parametry *queueItem*, *responseBuffer* a *response*, který se zavolá vždy po úspěšném stažení jedné stránky s článkem. Parametr *queueItem* je objekt nesoucí informace o položce ve frontě crawleru, má například vlastnosti *URL*, *depth* (hloubka zanoření), *host* (název domény) atd. Parametr *responseBuffer* obsahuje objekt typu `Buffer` s obsahem stažené stránky a parametr *response* je node objekt `http.ServerResponse` nesoucí informace o odpovědi serveru [24]. Jako výstup tedy bude fungovat volání callbacku při každém úspěšném stažení článku.

3.2.4 Procházení struktury

Pro stažení určitého množství článků ze zpravodajských webů pomocí crawleru je třeba nejprve zjistit, jaké počáteční adresy crawleru zadat a zda se z nich crawler bude moci dostat ke všem požadovaným článkům. Na vybraných webových serverech je vždy implementován jeden ze dvou způsobů, jak procházet starší články, a to archiv nebo stránkování.

Server Novinky umožňuje procházení starších článků pomocí archivu. Archiv je možné zobrazit vždy pro danou rubriku na určitý měsíc, například tedy články z rubriky *domácí* pro měsíc *duben 2016*. Měsíce v archivu není možné procházet pomocí odkazů, ale pouze díky formuláři umístěnému na stránce s archivem.

Servery iDnes, Aktuálně a Parlamentní listy řeší procházení starších článků pomocí stránkování v daných rubrikách. Je tedy možné se pomocí odkazů dostat ze stránky s nejnovějšími články postupně dál až po ty nejstarší. Každý ze serverů má ale stránkování řešeno mírně odlišně. iDnes zobrazuje vždy odkaz na předchozí a následující stránku a dále odkazy na tři stránky před a tři stránky po aktuálně zobrazené stránce. Aktuálně zobrazuje pouze odkazy na další a předchozí stránku. PL zobrazují odkazů nejvíce, a to odkaz na předchozí a následující stranu, odkazy na dvě stránky vpřed a dvě stránky vzad, a odkazy na první dvě stránky a na poslední dvě stránky článků v rubrice. Stránkování serveru Parlamentní listy je zobrazeno na obrázku 3.1.

Na každém serveru lze pomocí URL adresy stránky rozpoznat, zda se jedná o stránku s článkem, stránku archivu či stránku se seznamem článků a nebo o úplně jinou stránku. Díky tomu je možné v crawleru nastavit, aby se callback volal výhradně pro stránky obsahující článek, a ze stránek s archivem nebo se seznamem článků se pouze extrahovaly další URL adresy do fronty.

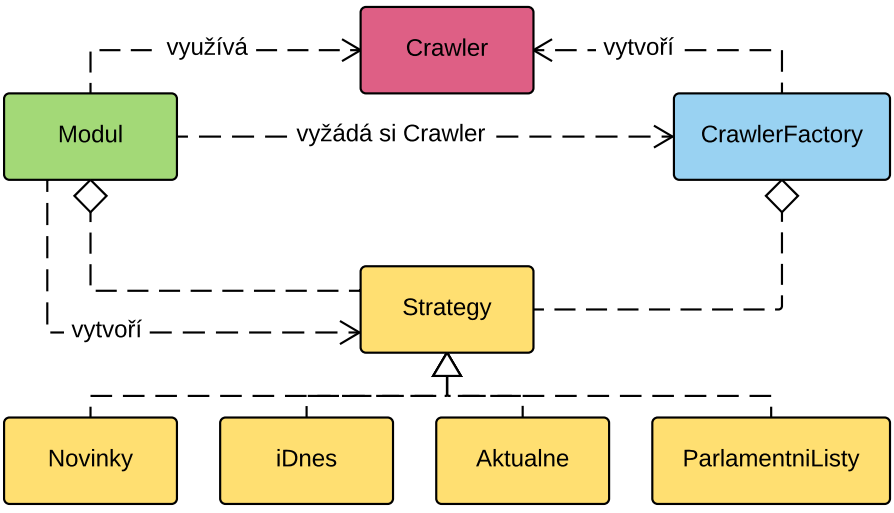
Obrázek 3.1: Stránkování na serveru Parlamentní listy

Ostatní adresy se pak do fronty nevkládají. Podstatné zrychlení představuje také omezení hloubky zanoření.

3.2.5 Návrh architektury

Pro použití modulu *node-simplecrawler* je třeba znát URL stránky, kde má crawler začít, jaká má být maximální hloubka zanoření a podmínky pro filtrování URL adres. To jsou parametry, které budou pro každý server jiné. Jakmile ale bude crawler vytvořen, proces stahování článků již bude pro všechny servery stejný.

Architektura modulu je zobrazena na obrázku 3.2. Pro vytvoření modulu jsem použila návrhové vzory *Factory* (na obrázku *CrawlerFactory*) a *Strategy*. CrawlerFactory dostane v konstruktoru jako parametr strategii vytvořenou modulem podle vstupních parametrů. Na jejím základě vytvoří instanci třídy Crawler, kterou vrátí modulu a ten pak může spustit stahování článků. Takto je zajištěna snadná rozšiřitelnost modulu o nové zpravodajské servery, jelikož bude stačit pro každý server pouze přidat novou strategii.



Obrázek 3.2: Návrh architektury modulu pro stahování článků

3.3 Implementace crawleru zpravodajských článků

Kód modulu byl napsán na základě architektury navržené v sekci 3.2. Pro každý zpravodajský server je tedy vytvořena instance node-simplecrawleru s jinými parametry.

Nejprve je dle vstupních parametrů *server* a *rubrika* zjištěna URL adresa stránky se seznamem nejnovějších článků, která se zadá jako vstup crawleru. Ten pak najde všechny odkazy na stránce a pokud projdou filtrováním URL adres, přidá je do fronty stránek ke stažení. Pro každý server existují dva tvary URL adres, které jsou nastaveny, aby prošly filtrem. První typ je URL adresa článku a druhý je URL adresa seznamu článků či archivu. Oba typy adres je možné filtrovat pomocí regulárních výrazů. Pokud je nalezená adresa typu článek, je stránka stažena a zavolán callback. V případě, že je nalezena stránka archivu či stránka se seznamem článků, jsou z ní pouze extrahovány další URL adresy.

Pomocí rozlišování typu adres je také možné značně omezit stahování nepotřebných stránek. Každá HTML stránka totiž obsahuje množství odkazů na nepotřebné stránky či soubory, které crawler detekuje, a pokud by nebyly vyfiltrovány, tak by se i zbytečně stáhly. Jedná se např. o obrázky, kaskádové styly a podobně, ale také o odkazy v menu, patičce atd. Také by se do crawlování mohly plést upoutávkové odkazy na další články ze stránky se staženým článkem. Takto je možné crawleru říci, že má stahovat pouze stránky, které mají URL adresu typu článek nebo archiv či seznam článků a přišlo se na ní ze stránky typu archiv či seznam článků.

Na ukázce kódu 3.1 jsou zobrazeny regulární výrazy a podmínka crawleru pro filtrování stránek na serveru iDnes. Výraz `page` se používá pro filtrování URL adres článků, výraz `archiv` pro filtrování URL adres stránek se seznamem článků. Proměnná `myCrawler` je objekt node-simplecrawleru, kterému se pomocí funkce `addFetchCondition()` přidá možnost filtrovat URL adresy před jejich přidáním do fronty pro stažení stránek. Objekt `queueItem` představuje stránku, ze které jsou extrahovány nové URL adresy, a objekt `parsedURL` představuje získanou URL adresu, na kterou se aplikuje filtrování.

Ukázka kódu 3.1: Filtrování URL adres před jejich přidáním do fronty crawleru

```
1 var page = /[a-z\-0-9\_]+.[a-z\-0-9\_]+\.aspx\?c=[A-Za-
    z0-9\_]+$/;
2 var archiv = /zahranicni\.aspx\?strana=\d+$/;
3
4 myCrawler.addFetchCondition(function(parsedURL,
    queueItem) {
5   return (parsedURL.path.match(archiv) && queueItem.
      path.match(archiv)) || (parsedURL.path.match(page)
       && queueItem.path.match(archiv));
6 });
```

Crawleru je při vytváření zadána URL adresa, na které má začít s crawlováním. Proces crawlování pak běží dokud není prázdná fronta se stránkami ke stažení nebo dokud jsou stažené články správného data (nejsou starší než zadané datum).

Pro servery iDnes, Aktuálně a PL není nastavena maximální hloubka zanoření crawleru kvůli ponechání možnosti stránkování. Do fronty se ukládají pouze URL adresy stránek s články a stránek se seznamem článků.

Pro server Novinky je situace složitější, jelikož ze stránky archivu není možné pomocí odkazů přejít na archiv předchozího měsíce. Pokud je tedy potřeba stáhnout články přes více měsíců, musí být vytvořeno i více crawlerů. Je pak ale možné nastavit hloubku zanoření pouze na hodnotu 2 (stránka, na které crawler začíná, a jedna další), protože na všechny stránky s článkem se přistoupí z prvotní stránky.

Na Parlamentních listech bylo ještě potřeba vyřešit, aby se články stahovaly postupně od nejnovějších po starší. Protože ve stránkování jsou zobrazeny i odkazy na stránky s nejstaršími články, crawler by se dostal i tam a začal by stahovat články třeba několik let staré. To by mohlo vést k předčasnému ukončení crawleru, protože by se tím začaly stahovat články, které by mohly být staršího data, než do jakého se články mají stahovat. Tento problém je v modulu vyřešen zjištěním čísla stránky z URL adresy a v případě, že je rozdíl čísla aktuální stránky a čísla nové stránky vyšší než 2 (což je pro PL počet zobrazených odkazů na další a předchozí stránky, viz obrázek 3.1), nová stránka se do fronty nepřidává.

Na ukázce kódu 3.2 je zobrazen kód pro spuštění crawlování. V komentáři na řádku 2 jsou vypsány možnosti pro volbu zpravodajského serveru, na řádku 3 pak možnosti pro výběr rubriky.

Ukázka kódu 3.2: Použití modulu pro stahování článků

```
1  const crawler = require("../src/crawler.js");
2
3  var server = 'Novinky'; // Novinky | iDnes | Aktualne |
       PL
4  var section = 'zahraniční'; // domácí | zahraniční
5  var dateTo = new Date(2016, 2, 31, 0, 0); // year,
       month (indexed from 0!), day, hour, minute
6
7  function processData(queueItem, responseBuffer,
       response) {
8    console.log("Processing " + queueItem.url);
9  }
10
11 crawler.crawl(server, section, dateTo, processData);
```

3.4 Testování crawleru zpravodajských článků

Vlastní modul využívá existující modul *node-simplecrawler*, který již byl otestován jeho tvůrci. Modul pro stahování článků pouze nastavuje existující modul pro možnost crawlování z více zpravodajských serverů a poskytuje jednoduché rozhraní pro spuštění crawlování s požadovanými parametry. Byly tedy otestovány skutečnosti, zda se stránky stahují pouze ze správného serveru a rubriky a pouze do zadaného data. Modul byl otestován především ostrým provozem popsaným v kapitole 6.

Kapitola 4

Parser zpravodajských článků

Tato kapitola se po seznámení s existujícími nástroji pro parsování HTML obsahu bude věnovat návrhu, implementaci a testování vlastního algoritmu pro získání relevantních informací z článků publikovaných na českých zpravodajských serverech. Algoritmus bude zaměřen na získání nadpisu, obsahu, klíčových slov a data publikace jednotlivých článků, jak bylo navrženo v kapitole 2.

4.1 Existující nástroje pro získávání obsahu z HTML

Nejvýraznějším nástrojem pro získání obsahu z HTML stránek je v Node.js modul *Readability*. Známá je rovněž java knihovna *Boilerpipe*, pro kterou existuje wrapper *node-boilerpipe*, jenž umožňuje její využití v Node.js. Dále existují komerční online nástroje, jako například *AlchemyAPI* či *Diffbot*, kterými se zde ale nebudu zabývat.

4.1.1 Readability

Readability je modul, který se snaží jakoukoli webovou stránku vyčistit od zbytečných informací a nechává pouze čistý relevantní obsah. Je založen na algoritmu *Arc90 Readability* [26].

Celý algoritmus je založen na tom, že v HTML filtruje značky podle obsahu jejich atributů `id` a `class`. Pokud by název některého z atributů mohl znamenat, že daný element obsahuje relevantní text (např. *post*, *article*, *content*...), je zařazen do seznamu pozitivních názvů. Naopak pokud obsah atributu značí, že by mohlo jít například o součást webové šablony (např. *header*, *footer*, *menu*...), jméno je zařazeno do seznamu negativních názvů. Jakmile jsou sestaveny tyto dva seznamy, algoritmus v HTML zadané stránky najde všechny odstavce (HTML značka `p`) a přidá jejich rodičovské elementy do seznamu elementů. Pro každý odstavec pak v závislosti na hodnotách jeho

atributů `id` a `class` upraví skóre rodičovského prvku. Po projití všech odstavců je v seznamu elementů nalezen prvek s nejvyšším skóre, jeho obsah je pak považován za relevantní obsah stránky [27].

Samotný modul pak umožňuje definovat vlastní funkci pro předzpracování HTML (úprava HTML stránky před vstupem do samotného algoritmu) a vlastní funkci pro doplnění pravidel filtrování HTML elementů.

Na výstup je vrácen objekt `article`, který má atributy

- `content`, který v sobě nese extrahovaný obsah stránky,
- `title` obsahující titulek stránky,
- `textBody`, což je řetězec obsahující veškerý text nalezený na stránce,
- `html` obsahující HTML originální stránky,
- `document`, který obsahuje DOM vygenerovaný modulem *JSDOM*, a
- `meta object`, což je odpověď knihovny *request* [26].

Readability tedy dokáže z článku extrahovat nadpis a obsah, neumožňuje ale získání data publikace a klíčových slov. Kromě toho při testování na českých zpravodajských serverech se ukázalo, že modul poměrně často do relevantního obsahu nezahrne úvodní odstavec článku. Stává se to například na serverech *Novinky.cz*, *iDnes.cz*, *CeskaTelevize.cz* či *Lidovky.cz*. Ve vyextrahovaném obsahu také nebývají zahrnuty graficky odlišené bloky s doplňujícím obsahem, který se občas u článků nachází. To někdy nemusí vadit, například pokud se jedná o blok s odkazy na podobné články, jindy je tím ale ztracena část informace, třeba pokud text bloku nějakým způsobem článek doplňuje.

4.1.2 Boilerpipe

Boilerpipe je javovská knihovna, pro kterou existuje Node.js wrapper *node-boilerpipe* [28]. Z HTML stránek se snaží odstranit veškerý nerelevantní obsah.

Algoritmus funguje na principu rozdělení webové stránky do atomických textových bloků, které jsou následně anotovány. Podle anotací je pak určeno, zda daný textový blok patří do relevantního obsahu, či nikoli. Atomický textový blok je definován jako sekvence znaků, které jsou odděleny jednou nebo více HTML značkami s výjimkou značky `a` (odkaz). Anotace jsou informace o bloku textu, mohou být

- *strukturální*, kde se zkoumá přítomnost určitých HTML tagů, jako nadpisy (`h1`–`h6`), odstavce (`p`), odkazy (`a`) a oddělující tagy jako `div`,

- *textové*, kde se určuje např. průměrná délka slova, průměrná délka věty, počet slov, nebo také absolutní a relativní pozice textu ve stránce (pokud má stránka vysokou granularitu, je pravděpodobnější, že za obsahovým blokem bude následovat další obsahový blok, a za blokem šalony další blok šablony, bloky hlavního obsahu pak bývají obaleny bloky šablony),

- *denzitometrické*, kde se měří hustota odkazů v textu [29].

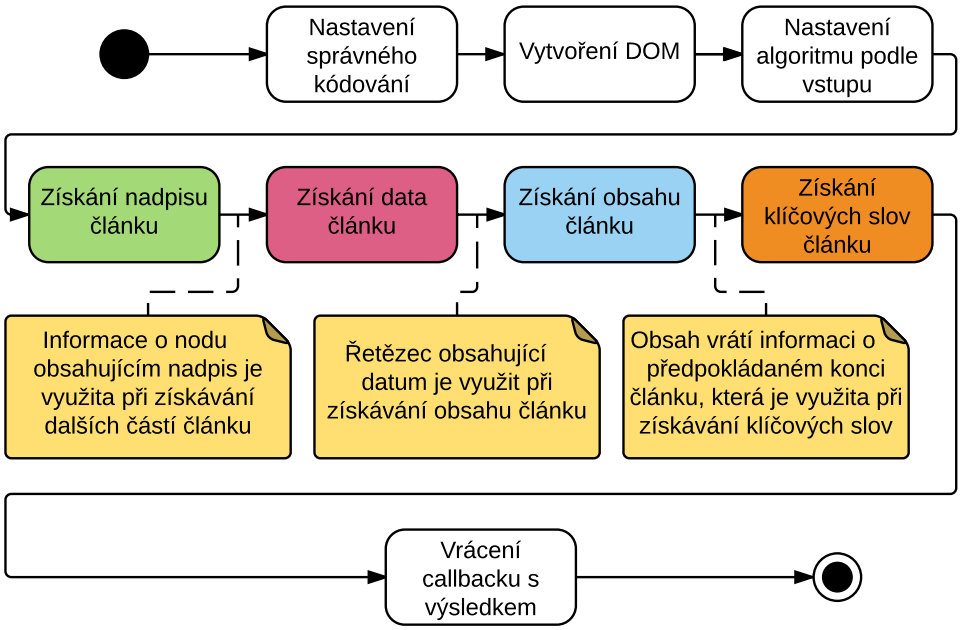
Boilerpipe extrahuje ze stránky jednotlitý text, nedokáže tedy získat zvlášť nadpis článku, datum publikace a klíčová slova. Pokud se extrakce zdaří, extrahovaný text obsahuje i nadpis, ale datum či klíčová slova už nikoli. Pro některé české zpravodajské stránky extrakce obsahu funguje dobře (např. servery *iDnes.cz* nebo *Aktualne.cz*), u jiných má ale stejný problém jako Readability, a to vynechání úvodního odstavce článku (např. *Novinky.cz* nebo *CeskaTelevize.cz*).

4.2 Návrh modulu pro parsování článků

Cílem algoritmu je z čistého HTML vyextrahovat nadpis, datum publikace, obsah a případná klíčová slova zpravodajského článku. Algoritmus je založen na analýze společných prvků článků z českých zpravodajských serverů a je rozdělen na čtyři hlavní části, které spolu vzájemně spolupracují. Proces parsování článku je zobrazen na obrázku 4.1.

Před samotnou extrakcí je potřeba převést vstupní článek do kódování UTF-8. Z takto upraveného HTML kódu stránky bude vytvořen dokumentový objektový model (DOM), který bude fungovat jako vstup do částí pro získání nadpisu, data, obsahu a klíčových slov. Části na sebe pak budou navazovat, ale každá z nich může fungovat i samostatně. Nejjednodušší bude získání nadpisu a data publikace článku. Výsledky obou těchto částí vstupují do procesu extrakce obsahu článku, ačkoli pro samotné získání obsahu nebudou zcela nutné. Pokud by při vstupu nebyl zadán nadpis, algoritmus by si ho zjistil sám, čímž by došlo pouze k mírnému zpomalení. Zjištěné datum pak bude použito k vylepšení výsledku extrakce tak, že pokud datum zůstane ve vyextrahovaném obsahu, bude z něho odstraněno. Součástí algoritmu pro získání obsahu bude i odhad konce relevantního obsahu. Ten pak bude použit v procesu získání klíčových slov. Některé zpravodajské servery totiž mají v patičce nebo v postranním menu seznam buď všech klíčových slov nebo nějakého jejich výběru, což by si algoritmus mohl splést s klíčovými slovy. Pokud mu bude předán odhadovaný konec článku, tak tyto odkazy bude ignorovat.

V částech hledání data, klíčových slov a obsahu budou využita také microdata. Na `https://schema.org/` existují slovníky `Article` a `NewsArticle`, které některé české zpravodajské servery používají pro anotaci svých článků, čímž velmi zjednodušují a zpřesňují extrakci jejich článků.



Obrázek 4.1: Proces parsování článků

4.2.1 Extrakce nadpisu

Extrakce nadpisu článku je nejjednodušší část z celého modulu. Algoritmus bude založen na vypozorované informaci, že nadpis článku je vždy ve značce nadpisu první úrovně, tedy v HTML značce `h1`. Většinou je tento typ nadpisu na stránce pouze jeden, některé servery jich ale mají více. Kromě toho zpravodajské servery dávají nadpis článku také do titulku okna prohlížeče, neboli do HTML tagu `title`. Tam ale mohou být také další informace, jako třeba název serveru, a nelze tedy vzít obsah tagu `title` jako doslovný nadpis článku. V případě, že je tagů `h1` na stránce více, je dobré z nich vybrat ten, jehož obsah je nejpodobnější titulku stránky.

4.2.2 Extrakce data

Datum publikace článku je možné z webové stránky extrahovat pomocí regulárních výrazů. Získaný textový řetězec je pak nutné transformovat do jednotného formátu.

České zpravodajské servery mají datum na stránce umístěno různě. Ve velké většině se nachází až pod nadpisem článku, existují ale výjimky, kdy je datum těsně nad nadpisem. Problémem při parsování data ze stránky pomocí regulárních výrazů je, že na stránce se často nenachází pouze datum publikování článku, ale i jiná data. Může to být například aktuální datum v hlavičce

stránky (*iDnes.cz*, *Lidovky.cz*, *ParlamentniListy.cz*...), datum u doporučovaných článků pod hlavním článkem na stránce (např. *CeskaTelevize.cz*, dále servery Novinky a Parlamentní listy zobrazují pouze čas) nebo data u komentářů k článku (*EuroZpravy.cz*). Nemusí být tedy jasné, které z nalezených dat je datum publikace článku.

Různé servery zobrazují datum v různých formátech, regulárním výrazem je tedy nutno jich pokrýt co nejvíce. Kromě toho i jeden server může mít formátů pro zobrazení data několik. Například server Novinky má různý formát pro články vydané v aktuální den (*Dnes 16:22*), předchozí den (*Včera 16:22*) a jiné dny (*pondělí 25. dubna 2016, 6:10*), což je časté rozdělení i pro další servery. Jiné pak odlišují články vydané v nedávné době od starších článků, např. *CeskaTelevize.cz* (*před 45 minutami*, *před 6 hodinami*). Dále je potřeba ve staženém HTML počítat s různými mezerami a jinými bílými znaky. Formáty různých serverů se mohou i částečně překrývat a lišit se pouze v drobnostech jako zda je oddělovacím znakem částí data čárka či mezera, jestli je před datem zobrazen i den v týdnu, či nikoli, zda se s datem zobrazuje i čas vložení apod. Například data starších článků serveru iDnes (*28. dubna 2016 16:01*) se od data serveru Novinky liší pouze v zobrazení dne v týdnu a oddělením času mezerou místo čárkou, pak server *ParlamentniListy.cz* (*28. 4. 2016 15:01*) od serveru *CeskaTelevize.cz* (*4. 12. 2015*) pouze zobrazením času a od serveru *Tyden.cz* pouze chybějícími mezerami v datu (*28.04.2016 12:16*). Dále existují i netradičnější datové formáty, například jako má server *aeronet.cz* (*Dub 28, 2016*).

Jakmile je ze stránky extrahován řetezec s datem, je nutné ho pomocí dalších regulárních výrazů převést do jednotného formátu, např. uložit do třídy `Date`, ze které je pak možné vypsat datum v libovolném formátu.

4.2.3 Extrakce obsahu

Obsah článků z českých zpravodajských serverů je vždy obklopen šablonou webu. Jako první v kódu se nachází hlavička, poté je článek, pak další informace jako komentáře a upoutávky na další články, dále případné postranní menu a na konci patička webu. Obsah článku vždy začíná až za jeho nadpisem. Je tedy možné veškerý obsah nad nadpisem článku ignorovat. Stejně tak prvky pod obsahem článku již nejsou nijak potřebné. Text pod obsahem se vyznačuje vysokým množstvím textu v odkazech, jelikož se většinou jedná o reklamy, odkazy na další články, štítky k článku, odkaz do diskuze a tlačítka na sociální sítě. Všechny zpravodajské servery mají vytvořené zobrazení pro upoutávky na další články, které se nacházejí většinou hlavně pod článkem, mohou být ale i před ním nebo v jeho průběhu. Upoutávka se většinou skládá z obrázku, nadpisu článku a krátkého textu k němu. Na obrázku a nadpisu je pak umístěn odkaz na daný článek. Upoutávku ke článku je možné si prohlédnout i na obrázcích 2.1 a 2.2 z kapitoly 2.

Algoritmus se tedy bude snažit vybrat z DOM uzly obsahující text a pomocí hustoty odkazů v těchto textech a jejich pozici oproti ostatním uzlům detekovat, které uzly nejsou součástí obsahu článku, a podle toho i určit konec obsahu článku a zahodit zbytek stránky. Zvlášť pak budou z HTML odstraněny zjevně nepotřebné uzly (např. tagy `script` či `video`) a uzly, které disponují názvy tříd (`class`) nebo identifikátorů (`id`), jenž naznačují, že je v nich umístěna reklama nebo odkazy na sociální sítě. Na konci algoritmu tedy v HTML zbyde pouze obsah článku.

Pokud je článek anotován pomocí microdat, pak budou z DOM odstraněny pouze uzly podle jejich hustoty odkazů, nikoli podle pozice, a pomocí microdat pak bude vybrán obsah. Uzly podle hustoty se odstraňují z důvodu, že některé servery (např. *Echo24.cz*) mají uprostřed článku odkazy na další články, což není relevantní obsah článku a měly by tedy být odstraněny.

Algoritmus je částečně založen na principu *boilerpipe*, místo dělení článku na textové bloky ale využívá dokumentový objektový model stránky, ze kterého vybírá pouze uzly s textem. Je také více specializovaný, jelikož *boilerpipe* se snaží detekovat obsah na webech různých typů, zatímco algoritmus v této práci je soustředěn pouze na extrakci obsahu zpravodajských článků. Určité kroky v algoritmu mohou připomínat i princip *Readability* díky odstraňování uzlů podle názvu HTML značky či obsahu jeho atributů.

4.2.4 Extrakce klíčových slov

Pokud článek není anotován microdaty, bude algoritmus extrakce klíčových slov založen na podobném způsobu, jako extrakce obsahu v modulu *Readability*, tedy na hodnotách HTML atributů `id`, `class`, `rel` a v horším případě `href`. Klíčová slova na českých zpravodajských serverech se nacházejí buď v odkazu s atributem `rel='tag'`, nebo v odkazu (`a`), seznamu (`ul`), odstavci (`p`) či tagu `div` s atributem `id` nebo `class`, který má hodnotu obsahující slovo `tag`. Může se ale stát, že klíčová slova jsou uložena pouze v odkazech bez jakýchkoli dalších atributů než je `href`. Odkazy na klíčová slova ale většinou obsahují v adrese rovněž slova naznačující, že se jedná o klíčové slovo, je tedy možné podle něj poznat odkaz na klíčové slovo. Nevýhoda tohoto přístupu je, že mimo klíčových slov u článku může být na stránce umístěna i jiná skupina klíčových slov, např. seznam oblíbených témat a podobně. Aby algoritmus extrahoval pouze klíčová slova článku, je možné využít detekci konce obsahu z algoritmu pro detekci obsahu.

4.3 Implementace modulu pro parsování článků

V této části popíši způsob implementace modulu pro parsování článků, který se dle návrhu algoritmu zobrazeného na obrázku 4.1 skládá z několika částí, které popíši v jednotlivých sekcích.

4.3.1 Vstup a výstup modulu

Vstupem do algoritmu bude buď URL adresa webové stránky s článkem, cesta k souboru s článkem a nebo buffer (objekt typu `Buffer`) s obsahem stránky s článkem. Dále bude na vstupu zadán objekt s nastaveními pro algoritmus. Nepovinně je možné zadat na vstupu i řetězec obsahující informaci *Content-Type* o předaném obsahu (použije se v případě, že byl zadán buffer nebo cesta k souboru).

Po skončení parsování modul zavolá callback definovaný na vstupu. Callback bude mít parametry *error*, vyplněný v případě chyby, a objekt *article*. Ten bude mít vlastnosti

- *header*, obsahující řetězec s extrahovaným nadpisem článku,
- *date*, obsahující objekt `Date` s extrahovaným datem článku,
- *keywords*, obsahující pole extrahovaných klíčových slov,
- *content*, obsahující řetězec s extrahovaným obsahem,
- *result*, což je objekt s vlastnostmi header, date, keywords a content, a
- *contentHTML*, což je DOM obsahující HTML vyextrahovaného obsahu (včetně hlavičky obsahující definici kódování).

4.3.2 Předzpracování

Před samotným parsováním stránky jsou provedeny následující tři kroky.

Převedení kódování Kódování dokumentu je možné určit dvěma způsoby. První z nich je pomocí hlavičky *Content-Type* v případě, že na vstupu byla zadána URL adresa, nebo pomocí stejnojmenného vstupního parametru v případě, že byl zadán soubor nebo buffer. Pokud není možné zjistit kódování touto možností, tak je zjištěno přímo z obsahu souboru, konkrétně z HTML hlavičky. Jakmile je zjištěno kódování, tak pokud dokument není v UTF-8, převede se obsah dokumentu právě do kódování UTF-8 pomocí Node.js modulu *encoding*[5].

Vytvoření DOM Dokumentový objektový model je vytvořen z dokumentu v kódování UTF-8 pomocí Node.js modulu *JSDOM*[6]. Na vstupu je mu předán buffer obsahující HTML stránku s článkem.

[5] https://www.npmjs.com/package/encoding
[6] https://github.com/tmpvar/jsdom

Nastavení algoritmu podle vstupu Před začátkem parsování je algoritmus nastaven podle vstupního objektu s nastaveními. Konkrétní možnosti nastavení budou popsány v sekci 4.3.7.

4.3.3 Extrakce nadpisu

Kód algoritmu pro získání nadpisu je zobrazen na ukázce kódu 4.1. Algoritmus nejprve najde všechny nadpisy první úrovně. Pokud na stránce existuje pouze jeden, znamená to, že jde o nadpis článku. Pokud jich je více, tak pomocí Levenshteinovy vzdálenosti porovná nalezené nadpisy s titulkem stránky a ten nejpodobnější považuje za nadpis článku. Algoritmus vrací DOM uzel s nadpisem článku. Pro získání textu je třeba použít funkci `text()`.

Ukázka kódu 4.1: Extrakce nadpisu článku

```
1  var header;
2  var headerLdist = Infinity;
3  var ldist_temp;
4
5  if($("h1").length > 1) {
6    $("h1").each(function(key, value) {
7      ldist_temp = ldist.getEditDistance($("title").text
           (), $(this).text());
8      if (ldist_temp < headerLdist) {
9        headerLdist = ldist_temp;
10       header = $(this);
11     }
12   });
13 }
14 else header = $("h1");
```

4.3.4 Extrakce data

Na ukázce kódu 4.2 je zobrazen pseudokód algoritmu pro získání data publikace článku.

Nejprve se zjistí, zda je článek anotován pomocí microdat. Pokud ano, tag, kde je zobrazeno datum článku, by měl mít atribut `itemprop` s hodnotou `datePublished` nebo `dateCreated`. Tento tag může mít také vlastnost `content` nebo `datetime`, kde je uloženo datum ve formátu *2015-12-04T16:43CET* nebo *2015-12-03T20:32:57+00:00*. Pokud jím disponuje a obsah vyhovuje jednomu z těchto formátů, je jeho hodnota brána jako datum publikace. Pokud ne, vezme se jako datum publikace text daného tagu.

Pokud článek není anotovaný pomocí microdat, je potřeba správné datum nalézt pomocí regulárního výrazu v textu článku. K tomu je potřeba rozlišit datum článku od ostatních dat na stránce. Algoritmus toto řeší pomocí detekce prvního nadpisu první úrovně. Pokud byl nalezen řetězec odpovídající

regulárnímu výrazu pro datum před detekcí nadpisu, je toto datum považováno za datum publikace článku pouze pokud se nachází v blízkosti nadpisu, tedy pokud je tag `h1` jeho sourozencem či sourozencem jeho rodiče. Pokud bylo nalezeno datum až po detekci nadpisu, je považováno za datum článku a algoritmus končí. Další data na stránce se již nehledají, protože to s nejvyšší pravděpodobností budou data jiných článků nebo komentářů.

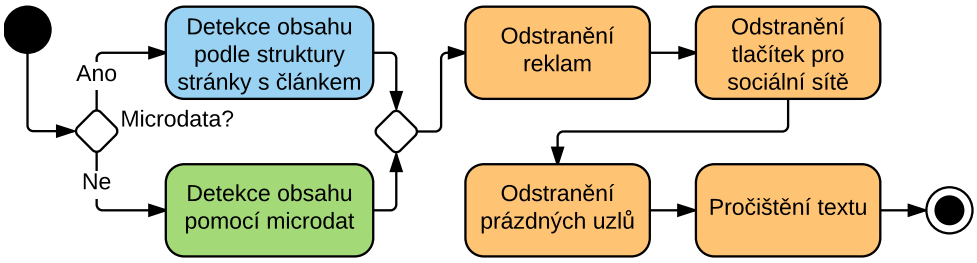
Po nalezení řetězce s datem článku je třeba pomocí dalších regulárních výrazů v řetězci detekovat rok, měsíc a datum, případně i čas, a pomocí těchto hodnot vytvořit objekt `Date`, který je pak vrácen jako extrahované datum článku.

Ukázka kódu 4.2: Pseudokód algoritmu pro extrakci data

```
1  var date = '';
2  var dateRegExp = regulární výraz pro extrakci data;
3
4  if (článek je anotován pomocí microdat) {
5    var itprop = $("*[itemprop='datePublished'], *[
       itemprop='dateCreated']");
6    if (itprop.attr("datetime") != null && itprop.attr("
       datetime").match(dateRegExp))
7      date = itprop.attr("datetime");
8    else if (itprop.attr("content") != null && itprop.
       attr("content").match(dateRegExp))
9      date = itprop.attr("content");
10   else
11     date = itprop.text().match(dateRegExp)[0];
12 } else {
13   var nodes = rodičovské uzly všech textových uzlů;
14   var h1 = false;
15   for (var i = 0; i < nodes.length; i++) {
16     if (!h1 && node[i].prop("tagName") == "H1")
17       h1 = true;
18     if (node[i].match(dateRegExp)) {
19       if (h1 == true)
20         date = node[i].text().match(dateRegExp)[0];
21         break;
22       if (node[i] nebo node[i].parent má sourozence "H1
           ")
23         date = node[i].text().match(dateRegExp)[0];
24         break;
25     }
26   }
27 }
28
29 return getDateFromString(date);
```

4.3.5 Extrakce obsahu

Na obrázku 4.2 je zobrazen zjednodušený graf procesu pro získání obsahu. Nejprve algoritmus zjistí, zda je článek anotován microdaty. Pokud tomu tak je, využijí se microdata pro získání obsahu. Pokud ne, obsah je získán pomocí detekce textových uzlů, vypočítání jejich statistik a smazání uzlů, které jsou rozpoznány jako nenesoucí relevantní obsah článku. Z obou těchto částí je výstupem promazaný DOM, ve kterém by měl z větší části zůstat pouze relevantní obsah. K jeho dočištění se následně algoritmus pokusí odstranit případné prvky obsahující reklamy a odkazy/tlačítka na sociální sítě. Po tomto kroku jsou pak z DOM smazány veškeré prázdné uzly, které v něm mohly zůstat po promazání. Na konci je pak z vyčištěného DOM získán pouze text, který je pročištěn od nadbytečných bílých znaků. Níže popíši detailněji zvlášť každou zmíněnou část.



Obrázek 4.2: Proces získání obsahu

4.3.5.1 Detekce obsahu podle struktury stránky s článkem

Níže budou popsány jednotlivé kroky získání obsahu podle struktury stránky s článkem.

Odstranění netextových HTML tagů Na začátku algoritmu jsou z DOM odstraněny netextové HTML tagy jako `script`, `style`, `form`, `video` apod.

Odstranění obsahu nad nadpisem článku Jelikož obsah článku se vždy vyskytuje pouze pod jeho nadpisem, části stránky nad ním je možné odstranit. V této části algoritmus odstraní všechny předcházející sourozence uzlu s nadpisem a všechny předcházející sourozence jeho předků. Poté je odstraněn i uzel s nadpisem, který v extrahovaném obsahu rovněž není žádoucí.

Získání množiny uzlů s texty a jejich statistiky V této fázi je vybrána množina DOM uzlů, pro které budou vypočteny statistiky a v dalších fázích algoritmu se bude rozhodovat, zda jejich obsah patří do relevantního obsahu

článku, či nikoli. Do množiny jsou vybrány jednak rodičovské uzly textových uzlů, které nemají žádného z předků typu odkaz (`a`), a jednak rodičovské prvky odkazů (uzlů typu `a`), které obsahují nějaký text. Výběr uzlů je zobrazen na ukázce kódu 4.3.

K takto vybraným uzlům se pak vypočítají statistiky, na základě kterých je posléze rozhodnuto o tom, zda daný uzel patří do obsahu článku. Pro každý uzel se počítá počet potomků, počet potomků typu odkaz, počet potomků typu text, počet slov textu mimo odkazy, počet slov textu v odkazech a délka znaků textového obsahu uzlu.

Ukázka kódu 4.3: Výběr uzlů pro procházení a výpočet statistik

```
1 var withNoAParents = $(":not(:has(*))").contents().
    filter(function() {
2   return (this.nodeType == 3 && $(this).parents("a").
      length == 0);
3 }).parent();
4
5 var aParents = $(":not(:has(*))").contents().filter(
    function() {
6   return (this.nodeType == 3 && $(this).parents("a").
      length > 0);
7 }).closest("a").parent();
8
9 var allNodes = withNoAParents.add(aParents);
```

Označení uzlů ke smazání podle hustoty odkazů v jejich textovém obsahu V tomto kroku jsou na základě vypočítaných statistik každého uzlu v předchozím kroku některé uzly označeny jako pro obsah nepotřebné, neboli ke smazání. Podmínkou pro toto označení je, že počet slov mimo odkazy nebo počet potomků typu text je nulový, nebo poměr počtu slov v odkazech a počtu slov v uzlu celkem je vyšší než stanovená konstanta, defaultně nastavena na hodnotu 0,7. Pokud je tedy v uzlu více než 70 % textu v odkazech, uzel je označen k odstranění. Konstantu pro poměr slov v odkazu vůči všem slovům je možné nastavit ve vstupních parametrech algoritmu. Tímto krokem by se měly označit ke smazání všechny uzly obsahující reklamy, odkazy na jiné články, případné seznamy odkazů na témata, menu a podobně, tedy většina uzlů pod obsahem článku.

Označení uzlů ke smazání podle jejich obsahu a pozice vůči ostatním uzlům V této části algoritmu se v cyklu projde celá množina uzlů vybraných v předchozí části. Algoritmus si v každém kroku cyklu zjistí nebo pamatuje, kolik předchozích uzlů a kolik následujících uzlů zkoumaného uzlu bylo označeno ke smazání. Na základě takto zjištěné pozice aktuálního uzlu v dokumentu se rozhoduje, zda je pravděpodobnější, že uzel do obsahu článku

patří, nebo nepatří. Kromě toho detekuje i uzly s určitým obsahem, které rovněž z výsledku vylučuje.

Nejdůležitější podmínkou pro ponechání uzlu je jeho pozice mezi ostatními uzly. Algoritmus si udržuje informaci o odstranění či ponechání určitého počtu uzlů nacházejících se před a po právě zkoumaném uzlu. Defaultně je tento počet nastaven na číslo 3, je možné ho ale upravit v nastavení algoritmu, kde se nachází pod názvem *NEARBY*. Zkoumaný uzel se odstraní pokud

- všechny ze stanoveného počtu uzlů před ním, resp. po něm, jsou označeny ke smazání a zároveň pro uzly po něm, resp. před ním, platí, že je jejich počet ke smazání děleno jejich stanoveným počtem vyšší než předem stanovená konstanta *REMOVE_RATIO* (defaultně nastavená na hodnotu 0,6 — tedy pokud dva ze tří následujících, resp. předcházejících, uzlů jsou označeny ke smazání, tedy jejich poměr je $0,\overline{66}$, a všechny předcházející, resp. následující, uzly jsou rovněž označeny ke smazání, pak je zkoumaný uzel také označen ke smazání; příkladem může být množina uzlů {S, S, S, A, S, O, S}, kde *S* znamená odstranění, *O* ponechání a *A* aktuální uzel),

- zkoumaný uzel se nenachází na začátku ani na konci článku (existuje alespoň NEARBY předchozích a NEARBY následujících uzlů) a počet předchozích a následujících uzlů ke smazání děleno počtem všech předchozích a následujících uzlů je vyšší než *REMOVE_RATIO* — např. pokud jsou uzly za sebou označeny {S, S, O, A, S, S, O}, tak jejich poměr uzlů pro smazání a uzlů pro ponechání je $0,\overline{66}$, což pro případ, že *REMOVE_RATIO* = 0,6 vede k označení aktuálního uzlu ke smazání),

- všechny předcházející uzly byly označeny jako relevantní obsah a všechny následující uzly byly označeny ke smazání, již byl extrahován nějaký text o minimální délce a text ve zkoumaném uzlu má nejvýše stanovenou maximální délku (jedná se tedy o nápadně krátky text na konci článku).

Jedním z typů uzlů, které projdou přes síto podle hustoty odkazů ale obsah článku nenesou, jsou uzly obsahující seznam klíčových slov oddělených čárkou nebo jiným oddělovačem (např. „|“). Počet slov v odkazu je pak o jedna vyšší než počet slov mimo odkaz, ale všechna slova mimo odkaz (tedy oddělovače) mají stejnou hodnotu. Algoritmus tedy uzly s takovými seznamy označuje ke smazání. Dále vylučuje také uzly s datem publikace, které do relevantního obsahu nepatří.

V průběhu cyklu si pak udržuje informaci o tom, kolik uzlů za sebou bylo označeno ke smazání. Jakmile zjistí, že byla naplněna předem definovaná konstanta za sebou označených uzlů ke smazání, určí, že poslední uzel neoznačený ke smazání byl poslední uzel s obsahem článku a značí tedy konec obsahu. Všechny uzly nacházející se až za daným uzlem pak označí ke smazání. Počet za sebou smazaných uzlů pro rozhodnutí o konci článku je defaultně určen na

hodnotu 8, je ale možné ji změnit v nastavení algoritmu. V tomto směru bylo třeba vyřešit problém, kdy některé servery (např. *ParlamentniListy.cz* nebo *Neovlivni.cz*) umisťují do obsahu článku seznam odkazů na články s podobným tématem, což by mohlo vést k mylné detekci konce článku a předčasnému ukončení algoritmu. Pro vyřešení tohoto problému byla do algoritmu zapracována podmínka kontrolující, zda po seznamu uzlů ke smazání nenásleduje několik uzlů neoznačených ke smazání a zda délka textu těchto uzlů je vyšší než nějaká nastavená minimální délka textu (to zabraňuje algoritmu považovat například uzly s jedním slovem za relevantní obsah). Pokud byly za seznamem uzlů ke smazání nalezeny i uzly s relevantním obsahem, je tato skupina považována za odkazy ve článku a čítač za sebou odstraněných uzlů je vynulován. Rovněž některé servery umisťují před článek odkazy na další články, což by mohlo vést k ukončení algoritmu ještě před extrakcí jakéhokoli relevantního obsahu. Konec článku tedy může být označen až po tom, co byl nějaký relevantní obsah nalezen.

Jelikož označení aktuálně zkoumaného uzlu ke smazání se promítne pouze ve vyhodnocování jeho následujících uzlů, je možné v algoritmu nastavit, aby se množina uzlů prošla vícekrát. Na většině serverů to ale nevede k výraznému zpřesnění extrahovaného obsahu.

Na ukázce kódu 4.4 je zobrazen zjednodušený pseudokód funkce pro průchod uzlů a jejich označení ke smazání dle pozice a typu obsahu.

Ukázka kódu 4.4: Pseudokód označení uzlů ke smazání podle jejich pozice a typu

```
function markToRemoveAccordingToTypeAndOrder(nodes) {
  var positionStats = initPositionStats();
  var textLen = 0;
  var toRemoveConsecutive = 0;

  for (var k~= 0; k~< ITERATE; k++) {
    for (var i = 0; i < nodes.length; i++) {
      positionStats = updatePositionStats();
      if (shouldBeRemoved(node[i], positionStats)) {
        nodes[i].remove = true;
        toRemoveConsecutive++;
        if (removeConsecutive(nodes, i,
            toRemoveConsecutive, textLen)) {
          markNextNodesForRemoval(nodes, i + 1);
          break;
        }
      } else {
        textLen += nodes[i].textsLen;
        toRemoveConsecutive = 0;
      }
    }
  }
}
```

Detekce upoutávek na další články a jejich odstranění U odstranění upoutávek na další články je využito faktu, že v jednom uzlu existuje obrázek a text, kde na každém z nich je odkaz se stejným obsahem atributu `href`. Je třeba ale kontrolovat, zda stránka nemá upoutávky ve stejné úrovni jako je hlavní obsah, pak by totiž hrozilo, že se odstraní i uzel s relevantním obsahem. Kód pro odstranění uzlů obsahujících upoutávky je zobrazen na ukázce kódu 4.5.

Ukázka kódu 4.5: Odstranění upoutávek na další články

```
var removeAncestorOfImgAndA = function($, textLen,
    minPageLen) {
  $("a > img").each(function(key, val) {
    var href = $(this).parent().attr("href");
    if (href.length > 1 && !href.match(/\'|\"/) && $(
        this).parent().parent().find("a[href='" + href +
         "']").length > 0) {
      if (textLen == undefined || minPageLen ==
          undefined) var minTextOk = true;
      else var minTextOk = textLen - $(this).parent().
          parent().text().length > minPageLen;
      if (minTextOk) $(this).parent().parent().remove()
          ;
    }
  });
};
```

Odstranění uzlů označených ke smazání Na závěr se pomocí seznamu uzlů odstraní uzly dříve označené ke smazání z dokumentového objektového modelu.

4.3.5.2 Detekce obsahu pomocí microdat

Níže budou popsány jednotlivé kroky získání obsahu pomocí microdat.

Detekce a odstranění uzlů s vysokým poměrem odkazů Tato část je shodná s kroky *Získání množiny uzlů s texty a jejich statistiky*, *Označení uzlů ke smazání podle hustoty odkazů v jejich textovém obsahu* a *Odstranění uzlů označených ke smazání* v detekci obsahu podle struktury stránky v sekci 4.3.5.1. V detekci obsahu pomocí microdat je tento krok zařazen kvůli serverům, které v obsahu svých článků uvádějí odkazy na další články, například související s daným tématem. Tyto odkazy v případě získání obsahu pomocí microdat zůstanou ve vyextrahovaném textu, ačkoli k obsahu článku nepatří. V tomto kroku jsou tedy uzly s těmito odkazy odstraněny.

Extrakce uzlů označených microdaty jako obsah Obsah je ve slovnících `Article` a `NewsArticle` označen atributem `itemprop`, který má hodnotu `articleBody`. Na některých serverech pak bývá zvlášť označen první (úvodní) odstavec článku atributem `itemprop` s obsahem `description`. V této fázi jsou tedy uzly s těmito vlastnostmi v HTML zachovány, zatímco všechny ostatní uzly (kromě přímých předků) jsou odstraněny.

Odstranění netextových HTML tagů Tento krok je stejný jako stejnojmenný krok v detekci obsahu podle struktury v sekci 4.3.5.1. V detekci obsahu pomocí microdat je zařazen zejména kvůli vyčištění HTML hlavičky, není tedy nezbytně nutný.

4.3.5.3 Odstranění reklam

Pokud má některý uzel v DOM atribut `id` nebo `class`, jehož obsah vyhovuje regulárnímu výrazu `ADVERTISING_REGEXP` na ukázce kódu 4.6, uzel je odstraněn. Právě tak v případě, že text uzlu vyhovuje regulárnímu výrazu `ADVERTISING_IN_TEXT_REGEXP`, uzel je také odstraněn.

Ukázka kódu 4.6: Regulární výrazy pro odstranění uzlů obsahujících reklamu

```
1 ADVERTISING_REGEXP = /.*(reklama([^a-z\u00C0-\u017F]|$)
    |komercni|komerce|commercial|reklamni|sponzorovane|
    sponzor|sponzori).*/i;
2 ADVERTISING_IN_TEXT_REGEXP = /reklama([^a-z\u00C0-\
    u017F]|$)/i;
```

4.3.5.4 Odstranění tlačítek pro sociální sítě

Pokud má některý uzel v DOM atribut `href` nebo `src`, jehož obsah vyhovuje regulárnímu výrazu na ukázce kódu 4.7 (`SOCIAL_NETWORKS_REGEXP`), uzel je odstraněn.

Ukázka kódu 4.7: Regulární výraz pro odstranění uzlů obsahujících tlačítka pro sociální sítě

```
1 SOCIAL_NETWORKS_REGEXP = /(facebook\.com|twitter\.com|
    plus\.google\.com)/i;
```

4.3.5.5 Odstranění prázdných uzlů

Po promazávání uzlů v DOM jistě zbyly uzly, které v sobě již nenesou žádný obsah. V tomto kroku algoritmus takové uzly z DOM odstraní.

4.3.5.6 Pročištění textu

Ze zbylého HTML je pak získán text, který je pročištěn od nadbytečných bílých znaků a poslán na výstup.

4.3.6 Extrakce klíčových slov

Na ukázce kódu 4.8 je zobrazen pseudokód algoritmu pro extrakci klíčových slov článku. Jeho myšlenka je velmi jednoduchá. V první řadě se zjišťuje, zda je článek anotován microdaty. Pokud ano, jsou klíčová slova získána z uzlů s atributem `itemprop` s hodnotou `keywords`, což je způsob, jakým jsou klíčová slova popsána ve slovnících `Article` a `NewsArticle`.

Ukázka kódu 4.8: Pseudokód algoritmu pro získání klíčových slov

```
function getParsedKeywords($, h1, articleEnd) {}
  var tags = new Array();
  if ($("*[itemprop='keywords']").length == 1) {
    tags = getKeywordsAccordingToMicrodata();
  } else if ($("a[rel='tag']").length > 0) {
    tags = getKeywordsAccordingToArel();
  } else if ($("a[class*='tag'], a[id*='tag']").length
     > 0) {
    tags = getKeywordsAccordingToAProps();
  } else if (existuje uzel ul, div nebo p s~atributem
     id nebo class s~hodnotou 'tag') {
    tags = getKeywordsAccordingToTagProps();
  } else if ($("ul[class*='tag'], ul[id*='tag']").
     length > 0) {
    tags = getKeywordsAccordingToUlProps();
  } else if ($("div[class*='tag'], div[id*='tag']").
     length > 0) {
    tags = getKeywordsAccordingToDivProps();
  } else {
    removeUpFrom(h1);
    removeEndOfPage($, articleEnd);
    var i = 0;
    $('a[href]').each(function(key, value) {
      var href = $(this).attr("href");
      if ($(this).attr("href").match(/.*(tag|tagy|tags|
         tema|temata|klicove(-|_)slovo|klicova(-|_)
         slova|keyword|keywords|klic|klice).*/i))
        tags[i++] = $(this).text().trim();
    });
  }
  return deleteDuplicatesAndEmptyStrings(tags);
}
```

V případě, že článek není anotován microdaty, klíčová slova servery většinou umisťují buď do odkazů (značka `a`) s atributem `rel='tag'`, nebo do odkazů, seznamu (značka `ul`), odstavce (značka `p`) nebo značky `div` s atributem `class` či `id`, které mají hodnotu, kde se vyskytuje slovo *tag*. Přednost mají, pokud je jejich hodnota pouze *tag* bez dalších znaků. Některé servery ale mají uzly s klíčovými slovy v odkazu bez jakýchkoli dalších atributů než `href`, detekce takovýchto klíčových slov je tedy obtížnější. Naštěstí zde platí, že URL adresy umístěné na klíčových slovech, obsahují některá slova, která napovídají, že jde o adresu klíčového slova, například *tag*, *keywords* nebo *témata*, a porovnáním těchto adres s regulárním výrazem je možné určit, zda jde o klíčové slovo.

Problémem určování klíčových slov podle URL odkazu je, že takovéto odkazy se mohou nacházet třeba i v menu (např. seznam nejaktuálnějších témat), ať už v hlavičce či postranním panelu, nebo v patičce ve formě např. nejoblíbenějších témat. To pak může vést k extrakci klíčových slov, která k článku vůbec nemusí patřit. Tato situace se dá vyřešit pomocí vstupních parametrů obsahujících uzel s nadpisem a uzel následující několik uzlů po odhadnutém konci článku (tj. uzel, na kterém algoritmus pro získání obsahu rozhodl, že všechny další uzly mohou být smazány). Z DOM jsou pak odstraněny uzly před nadpisem a několik uzlů po odhadnutém konci článku. Tím je v DOM ponechán obsah článku a jeho okolí, kde by se měla nacházet i klíčová slova. Díky tomu není možné extrahovat různé odkazy na témata v menu či patičce. Před vrácením pole s klíčovými slovy je pole ještě pročištěno od případných prázdných prvků a možných duplikátů (může se stát např. na serveru Novinky, který má klíčová slova vyznačena odkazy v textu, nikoli jejich výčtem v blízkosti článku).

4.3.7 Použití modulu

Na ukázce kódu 4.9 je vidět příklad použití modulu pro extrakci informací ze stránky s článkem. V tomto případě je jako vstup definována URL adresa článku, může to být ale i cesta k souboru a nebo objekt typu `Buffer` s načteným obsahem stránky. Objekt `options` definuje nastavení pro algoritmus, která budou popsána níže. Algoritmus se spustí zavoláním funkce `parse()` s parametry `url`, `options` a funkcí, kterou má algoritmus zavolat po dokončení parsování.

Na ukázce kódu 4.10 je zobrazen výstup programu zobrazeného na ukázce kódu 4.3.7. Výstup je zobrazen ve formátu JSON a obsahuje vlastnosti `header` s extrahovaným nadpisem článku, `date` s datem publikace článku ve formátu výstupu funkce `toISOString()` objektu `Date`, `keywords` s polem zjištěných klíčových slov a `content` s obsahem článku.

Ukázka kódu 4.9: Příklad použití modulu pro získání informací o článku

```
var parser = require("./src/parser.js");
var url = "http://www.novinky.cz/zahranicni/svet
    /396487-u-indonesie-uderilo-mohutne-zemetreseni-jsou
    -obeti-na-zivotech.html";
var options = { "linksRatio": 0.5, "removeRatio": 0.5,
    "consecutiveCountRemove": 5 };

parser.parse(url, options, function(err, article) {
  if (err) console.log(err);
  else {
    console.log(article.result);
  }
});
```

Ukázka kódu 4.10: Ukázka výstupu modulu

```
{
  "header": "U Indonésie udeřilo mohutné zemětřesení,
     jsou oběti na životech",
  "date": "2016-03-02T13:27:00.982Z",
  "keywords": [
    "Zemětřesení",
    "Tsunami"
  ],
  "content": "Západním pobřežím Indonésie otřáslo
     mohutné zemětřesení o~síle 7,8 stupně Richterovy š
     kály. Informovala o~tom americká geologická služba
      USGS. Australské a indonéské úřady následně
     vydaly varování před přívalovou vlnou tsunami, to
     však bylo později zrušeno. Indonéské úřady podle
     agentury Reuters hlásí oběti na životech.
     Informace o~tom, kolik lidí zemřelo, kolik jich
     utrpělo zranění ani rozsah škod zatím není znám.
     Obyvatelé města Padang v~regionu Západní Sumatra
     se kvůli strachu z~hrozící tsunami uchýlili do vyš
     ších poloh."
}
```

Modul má hlavně kvůli algoritmu pro extrakci obsahu několik možností nastavení. Ta se předávají jako vlastnosti objektu v parametru. Možností jsou následující.

- *nearby*, neboli okolí uzlu, určuje, u kolika předchozích a u kolika následujících uzlů se zjišťuje, zda byly tyto uzly označeny ke smazání, a z výsledku se určuje, zda zkoumaný uzel má být také vyloučen. Možné

hodnoty jsou $\langle 1, 20\rangle$, defaultně je nastaveno 3. To znamená, že když algoritmus zkoumá, zda uzel vyřadit na základě jeho pozice, kouká se na tři předchozí a tři následující uzly, zda byly označeny ke smazání.

- *linksRatio* nastavuje, jaká část slov v uzlu musí být umístěna v odkazech, aby byl ze stránky odebrán. Možné hodnoty jsou $(0, 1\rangle$, defaultně je nastaveno $0,7$, což znamená, že se odstrání uzly se 70 a více procenty slov v odkazech. V případě, že při extrakci zůstaly ve vyextrahovaném obsahu i prvky, které neměly, a obsahují větší množství odkazů, je vhodné tuto hodnotu snížit. Naopak, pokud algoritmus maže, co nemá, je nutné hodnotu zvýšit. Při testování se ale nutnost zvyšování hodnoty nikdy neprojevila.

- *removeRatio* určuje, jak velké musí být množství okolních uzlů označených ke smazání, aby byl i zkoumaný uzel označen ke smazání. Možné hodnoty jsou $(0, 1\rangle$, defaultně je nastaveno $0,6$, což znamená, že aspoň 60 % ze zkoumaných předchozích a následujících uzlů musí být označeno ke smazání, aby se i právě zkoumaný uzel vyřadil. V případě, že algoritmus neodstraní všechny nežádoucí prvky ze stránky, je možné tuto hodnotu snížit. Naopak pokud odstraní i něco navíc, je potřeba ji zvýšit (nebyl zjištěn případ, kdy by to bylo nutné).

- *consecutiveCountRemove* nastavuje počet za sebou označených uzlů ke smazání (s výjimkou odkazů uprostřed textu článku a odkazů před článkem), kdy algoritmus rozhodne, že text článku již skončil a označí všechny zbylé uzly ke smazání. Možné hodnoty jsou celá čísla větší nebo rovná dvěma, defaultně je nastaveno 8. Pokud by algoritmus obsah článku nečekaně zkrátil, je třeba nastavit vyšší hodnotu. Naopak pokud algoritmus neodstraňuje vše, co má, číslo je možné nastavit na nižší hodnotu.

- *linksInsideArticle* určuje, zda server, ze kterého jsou články extrahovány, zobrazuje odkazy na jiné články uprostřed obsahu článku. Možné hodnoty jsou $\{true, false\}$, defaultně je nastaveno na $false$.

- *linksInsideArticleAfterOkCount* určuje, kolik uzlů neoznačených ke smazání musí následovat za sérií uzlů označených ke smazání, aby tato série byla považována za odkazy uprostřed článku. Hodnota může být celé číslo větší nebo rovno dvěma, defaultně je nastaveno 3.

- *minPageLen* určuje minimální délku extrahovaného textu. Dokud nebyl extrahován text aspoň o této délce, algoritmus neskončí na základě splněného počtu po sobě jdoucích uzlů označených ke smazání. Hodnotu je možné nastavit jako celé číslo od jedné. Defaultní hodnota je 500.

- *minTextLen* značí minimální délku textu v uzlech označených jako nesoucí obsah článku. Může nabývat celého čísla většího než 0. Defaultně nastaveno na hodnotu 200.

- *maxTextLen* značí maximální délku textu v uzlech označených ke smazání na základě jejich pozice. Může nabývat celého čísla většího než 0. Defaultně nastaveno na hodnotu 50.

- *socialNetworksRegexp* je regulární výraz, který se používá pro porovnání adresy odkazů při určování, zda se jedná o odkaz na sociální sítě. Výraz na ukázce kódu 4.11 je nastaven defaultně.

  Ukázka kódu 4.11: Defaultní hodnota regulárního výrazu pro odstranění odkazů na sociální sítě

```
1 /(facebook\.com|twitter\.com|plus\.google\.com)/i
```

- *advertisingRegexp* je regulární výraz, který se používá pro porovnání obsahu atributů `class` a `id` kvůli detekci uzlů s reklamou. Defaultně je nastaven výraz zobrazený na ukázce kódu 4.12.

  Ukázka kódu 4.12: Defaultní hodnota regulárního výrazu pro odstranění reklamy dle atributů

```
1 /.*(reklama([^a-z\u00C0-\u017F]|$)|komercni|
    komerce|commercial|reklamni|sponzorovane|
    sponzor|sponzori).*/i
```

- *advertisingInTextRegexp* je regulární výraz, který se používá pro porovnání obsahu uzlů kvůli detekci těch s reklamou. Defaultně je nastaven výraz zobrazený na ukázce kódu 4.13. Ačkoli to tak původně nebylo zamýšleno, je možné tímto způsobem nastavit odstranění uzlů se specifickým obsahem.

  Ukázka kódu 4.13: Defaultní hodnota regulárního výrazu pro odstranění reklamy dle obsahu

```
1 /reklama([^a-z\u00C0-\u017F]|$)/i
```

- *iterate* definuje, kolikrát se projdou uzly pro posouzení, zda budou odstraněny na základě jejich pozice. Je možné nastavit celé číslo v intervalu $\langle 1, 20 \rangle$, defaultně je nastavena hodnota 1. Více jak dvě iterace ale nemívají na extrahovaný text vliv.

- *debugMode* může nabývat hodnot $\{true, false\}$, defaultně je nastaveno na $false$. Pokud se změní na *true*, jsou aktivovány výpisy do konzole o průběhu algoritmu.

- *keywordsRegexp* je jediné nastavení, které se netýká extrakce obsahu, ale extrakce klíčových slov. Definuje se jím regulární výraz, pomocí něhož se porovnávají adresy odkazů na potenciální klíčová slova. Defaultní hodnota je zobrazena na ukázce kódu 4.14.

  Ukázka kódu 4.14: Defaultní hodnota regulárního výrazu pro porovnání adresy odkazu potenciálních klíčových slov

```
1 /.*(tag|tagy|tags|tema|temata|klicove(-|_)slovo|
    klicova(-|_)slova|keyword|keywords|klic|klice)
    .*/i
```

Nejužitečnějšími z těchto nastavení jsou *linksRatio*, *removeRatio*, *consecutiveCountRemove* a *linksInsideArticle*. Extrakce obsahu se pomocí nastavení dá pro různé servery pomocí různých nastavení zpřesnit. Na ukázce kódu 4.15 jsou zobrazena možná nastavení pro server Novinky, PL a *Lidovky.cz*. Pro Novinky je bez problémů možné snížit poměr odkazů pro odstranění uzlu a počet smazaných uzlů, po kterém se odstraní i zbytek stránky. Pro PL je pak nutné nastavit informaci, že server s oblibou v textu článku zobrazuje množství odkazů na další články. Server *Lidovky.cz* pak v HTML kódu má dvakrát uzel s textem „Zkrácená adresa“ schovaný pomocí CSS a tedy návštěvníkům neviditelný. Schovaný uzel se ale nachází přímo před a za článkem a neobsahuje žádný odkaz, je tedy v pořádku, že ho algoritmus do extrahovaného textu zahrnul, ačkoli z logického hlediska tam text nepatří. Pomocí doplnění regulárního výrazu je pak možné tyto dva uzly odstranit.

Ukázka kódu 4.15: Defaultní hodnota regulárního výrazu pro porovnání adresy odkazu potenciálních klíčových slov

```
1 var novinky = { "linksRatio": 0.5, "removeRatio": 0.5,
    "linksInsideArticle": true };
2 var pl = { "linksInsideArticle": true };
3 var lidovky = { "advertisingInTextRegexp": /reklama([^a
    -ž]|$)|Zkrácená adresa/ig };
```

4.4 Testování modulu pro parsování článků

V této části bude otestován modul pro parsování článků. V první části budou otestovány části pro získání nadpisu, data a klíčových slov, v druhé části pak zvlášť algoritmus pro získání obsahu. Testování algoritmů je poměrně obtížné,

jelikož existuje velmi málo možností jejich automatizace, pokud vůbec. K zjištění úspěšnosti algoritmů je totiž třeba vědět, jaký měl být výsledek extrakce (tedy jaký je nadpis, datum, všechna klíčová slova a přesný text obsahu), aby bylo možné s ním porovnat extrahované hodnoty a zjistit tak úspěšnost algoritmů.

Testovány byly články ze serverů *Aeronet.cz*, *Aktualne.cz*, *CeskeNoviny.cz*, *CeskaTelevize.cz*, *Denik.cz*, *Echo24.cz*, *E15*, *EuroZpravy.cz*, *iDnes.cz*, *iHned.cz*, *Lidovky.cz*, *Novinky.cz*, *ParlamentniListy.cz*, *rozhlas.cz*, *tiscali.cz* a *tyden.cz*, celkově tedy články ze šestnácti zpravodajských serverů.

4.4.1 Testování extrakce nadpisu, data a klíčových slov

Extrakce nadpisu, data a klíčových slov článku byla na výše uvedených šestnácti českých zpravodajských serverech otestována na náhodně vybraných 44 článcích napříč zmíněnými servery. Výběr byl ale prováděn tak, aby vždy byly otestovány všechny formáty data na daném serveru. Články byly manuálně anotovány správnými hodnotami a ty poté byly porovnány s těmi extrahovanými. Nadpis a datum článku se podařilo správně extrahovat ve všech případech, klíčová slova se pak podařila vždy, když na serverech byla k dispozici.

4.4.2 Testování extrakce obsahu

Pro vyjádření úspěšnosti algoritmu pro extrakci obsahu využiji metriky *precision* (neboli míru přesnosti) a *recall*, neboli míru úplnosti. Měřit jimi budu, zda byla extrahována slova, která měla být. Slovem se zde myslí skupina znaků oddělených mezerou.

Precision určuje, kolik z extrahovaných slov skutečně je relevantním obsahem článku. Recall měří, kolik ze slov relevantního obsahu bylo extrahováno. Vzorec výpočtu precision a recall jsou zobrazeny na rovnicích 4.1 a 4.2.

$$precision = \frac{|\{všechna\ extrahovaná\ slova\} \cap \{slova\ relevantního\ obsahu\}|}{|\{všechna\ extrahovaná\ slova\}|} \tag{4.1}$$

$$recall = \frac{|\{všechna\ extrahovaná\ slova\} \cap \{slova\ relevantního\ obsahu\}|}{|\{slova\ relevantního\ obsahu\}|} \tag{4.2}$$

Stejně jako pro testování extrakce nadpisu, data a klíčových slov, i zde byly testy provedeny na 44 manuálně anotovaných dokumentech ze šestnácti zpravodajských serverů. Pro každý z nich byl vypočítán precision a recall a tyto hodnoty byly pro články z každého serveru zprůměrovány. Výsledky jsou zobrazeny v tabulce 4.1. Průměr precision všech článků je 0,993, průměrný recall 1,000.

Tabulka 4.1: Výsledky manuálního testování parseru

server	precision	recall	server	precision	recall
Aeronet.cz	0,996	1,000	iDnes.cz	1,000	1,000
Aktualne.cz	1,000	1,000	iHned.cz	0,998	1,000
CeskeNoviny.cz	0,968	1,000	Lidovky.cz	1,000	1,000
CeskaTelevize.cz	0,992	1,000	Novinky.cz	0,996	1,000
Denik.cz	0,999	1,000	ParlamentniListy.cz	0,995	1,000
Echo24.cz	1,000	1,000	Rozhlas.cz	0,996	1,000
E15.cz	0,997	1,000	Tiscali.cz	0,988	1,000
EuroZpravy.cz	0,997	1,000	Tyden.cz	0,973	1,000

Z tabulky je vidět, že ne vždy se podaří odmazat ze stránky všechny prvky, které do relevantního textu nepatří, ale vždy se povede extrahovat všechen relevantní obsah. Těch několik neodstraněných slov pak bývají různé nadpisy pod článkem, které na sobě nemají odkazy a tedy se špatně detekují. Mohou to být nadpisy upoutávek na další články (např. „Kam dál? “ či „Související články“), komentáře, název sekce, pod kterou článek patří, a podobně.

4.4.3 Testování změn parametrů

Algoritmus s defaultním nastavením funguje dobře na většině serverů. Pro některé z nich ale funguje lépe s upraveným nastavením. V této části bych ráda ukázala, jak některá nastavení ovlivní extrakci obsahu.

Pro porovnání výsledků bylo potřeba najít způsob, jakým by se dal automaticky porovnat extrahovaný obsah s opravdovým relevantním obsahem. Jako řešení jsem zvolila extrakci relevantního obsahu pomocí zjištění, v jakých HTML značkách se nachází relevantní obsah každého testovaného serveru, a následné extrakce textu pomocí DOM. Tím se podaří získat obsah článku bez prvků okolo. Nemusí to ale znamenat, že extrahovaný text je úplně přesný, jelikož se i uvnitř článku mohou nacházet reklamy, odkazy na jiné články a podobně. Jelikož ale v této části je podstatné spíše, jak se hodnoty mění, než jak přesné jsou, je tento způsob dostačující. Pro zpřesnění porovnávání jsem ze staženého HTML před extrakcí z tagů odstranila typicky netextové uzly, které by mohly produktovat nějaký text mimo relevantní obsah, tedy např. tagy `script`, `noscript`, `video` apod.

Pro automatické testování jsem zvolila jinou metriku než pro manuální, a to procentuální podobnost vyextrahovaných řetězců zjištěnou pomocí Levenshteinovy (editační) vzdálenosti. Levenshteinova vzdálenost je definována

jako minimální počet operací nahrazení, přidání nebo smazání pro to, aby se z jednoho řetězce stal druhý. Pro procentuální podobnost řetězců jsem použila vztah definovaný rovnicí 4.3, kde *levenshtein* je funkce pro výpočet Levenshteinovy vzdálenosti, *relevant* je text extrahovaný pomocí HTML značek, tedy téměř relevantní obsah, a *extracted* je obsah extrahovaný modulem. Z obou řetězců jsou nejprve odstraněny všechny bílé znaky. Hodnota *similarity* pak říká, kolik procent znaků z řetězců je shodných.

$$similarity = 1 - \frac{levenshtein(relevant, extracted)}{max(|relevant|, |extracted|)} \tag{4.3}$$

Testování proběhlo pro tři nejčtenější servery, tedy Novinky (109 článků), iDnes (79 článků) a Aktuálně (120 článků). Byly testovány dvě nejpoužívanější nastavení *linksRatio* a *removeRatio*. Výsledky úspěšnosti pro různé parametry jsou zobrazeny v tabulce 4.2, kde v prvním a druhém sloupci jsou uvedeny hodnoty nastavení a ve zbylých třech sloupcích je uvedeno, kolik procent z řetězce extrahovaného přes HTML značky a řetězce extrahovaného algoritmem je stejných. Kurzívou jsou v posledních třech sloupcích tabulky vyznačené výsledky úspěšnosti v buňkách, které pro daný server odpovídají nastavením použitým v manuálním testování.

Tabulka 4.2: Výsledky automatického testování parseru

linksRatio	**removeRatio**	**Novinky** [%]	**iDnes** [%]	**Aktuálně** [%]
0,7	**0,6**	92,3	*97,7*	*91,5*
1	**0,6**	92,3	98,4	91,5
0,5	**0,6**	92,7	97,7	92,3
0,05	**0,6**	85,6	88,0	83,3
0,7	**1**	91,1	97,7	89,6
0,7	**0,5**	93,3	97,7	91,9
0,7	**0,05**	93,0	97,7	90,5
1	**1**	91,1	98,4	89,6
0,5	**0,5**	*93,8*	97,7	92,4
0,05	**0,05**	83,5	88,0	82,6

Zjištěné hodnoty bohužel z výše zmíněných důvodů nemusí přesně odpovídat realitě. Pro Novinky platí, že se v textu občas nachází odkazy na jiné články, které v „referenčním“ textu jsou, ale modul je záměrně odstraňuje. Stejná je situace i na iDnes, ale tam se projevuje v menší míře, což je vidět i z vyšší úspěšnosti výsledků. Nejhorší je situace na serveru Aktuálně, jelikož

web není nakódovaný tak, že by měl v jednom uzlu pouze obsah článku, ale má tam umístěn i nadpis, datum, upoutávky na další články atd. Bylo tedy nutné z těchto prvků vybrat pouze odstavce a nadpisy, což ale zdaleka nemusí odpovídat všem relevantním prvkům článku.

I přes nepřesnost měření je možné si všimnout změn úspěšností při různých parametrech, ačkoli většinou ne dramatických. Výrazný pokles úspěšnosti je patrný pro nastavení minimálního poměru odkazů pro smazání (*linksRatio*) na 0.05 (5 % textu). To má totiž za následek, že jsou smazány i texty s velmi malým množstvím odkazů a pokud se odkazy nacházejí i v textu článku, tak je smazán i relevantní text. Při nastavení na 1 (odstraní se uzly, které obsahují výhradně odkazy) poklesne úspěšnost jen pro Novinky a Aktuálně, pro iDnes překvapivě úspěšnost vzroste. Dle výsledků by také bylo vhodné zvážit, zda by pro server Aktuálně nebylo vhodnější použít stejné nastavení jako pro Novinky, pro které je úspěšnost vyšší než u defaultního nastavení, a pro server iDnes zvýšit parametr *linksRatio* na 1, kde se úspěšnost rovněž jeví vyšší než pro defaultní nastavení.

Kapitola 5

Analýza zpravodajských článků

V této kapitole budou prozkoumány dostupné nástroje pro analýzu textu v českém jazyce. Na základě zjištěných možností budou vybrány metody pro analýzu stažených novinových článků a ty budou implementovány ve formě Node.js modulů. Dále bude popsán návrh, implementace a testování těchto modulů.

5.1 Existující nástroje pro text mining v češtině

Pro analýzu textů existuje relativně velké množství nástrojů, ať už dostupných online či jako Node.js moduly. Po jejich omezení na český jazyk jich ale nezbyde mnoho. Jednoznačně nejvíce nástrojů v tomto oboru poskytuje Ústav formální a aplikované lingvistiky Univerzity Karlovy v Praze (ÚFAL). Dalším zdrojem nástrojů je množství závěrečných prací z různých univerzit, ke kterým ale není jednoduché se dostat. V následujícím textu popíši zejména nástroje, které poskytuje ÚFAL, zmíním se ale i o několika dalších.

5.1.1 MorphoDiTa

MorphoDiTa je open-source nástroj pro morfologickou analýzu přirozeného textu vyvíjený ÚFALem [30]. Jelikož je to nástroj využívající strojového učení, je nutné mu poskytnout trénovaný lingvistický model. Tím může být pro češtinu například MorfFlex, rovněž poskytnutý ÚFALem [31].

MorphoDiTa umožňuje provádět morfologickou analýzu, morfologické generování, tagování a tokenizaci. Poskytuje online demo, online REST API, binární soubory pro Linux, Windows a OSX a zdrojové kódy, ze kterých je možné si zkompilovat REST API server.

Na výstupu 5.1 je zobrazen výstup morfologické analýzy pro větu „*Tomáš Garrigue Masaryk byl první prezident Československé republiky.*“. Výstup má tři sloupečky, kde první je tvar slova ve větě na vstupu, druhý je jeho morfologické lema a třetí morfologická značka (tag) [10].

Výstup 5.1: Výstup morfologické analýzy programu MorphoDiTa

```
1 Tomáš Tomáš_;Y NNMS1-----A----
2 Garrigue Garrigue_;Y_,t NNMS1-----A----
3 Masaryk Masaryk_;S NNMS1-----A----
4 byl být VpYS---XR-AA---
5 první první CrMS1----------
6 prezident prezident NNMS1-----A----
7 Československé československý AAFS2----1A----
8 republiky republika NNFS2-----A----
9 . . Z:-------------
```

Morfologické lema obsahuje kromě vlastního lema sadu technických přípon, které jsou nepovinné (pro každou platí, že se nemusí vyskytovat vůbec, nebo se může vyskytovat jednou nebo i vícekrát). Celé lema má tedy tvar `vlastnílema_:P1_;P2_,P3_^(P4)`. Vlastní lema funguje jako jednoznačný identifikátor analyzovaného tvaru, což někdy vyžaduje přidání číslice, která odlišuje různé významy stejného lematu, např. *vazba-1* jako *uvěznění* a *vazba-2* jako *spojení* [10]. Pro technické přípony pak platí, že každá z nich začíná specifickou kombinací znaků. Jednotlivé přípony jsou popsány v tabulce 5.1. Přípona `P1` obsahuje informaci o vidu slovesa nebo zda se jedná o zkratku, přípona `P2` určuje typ vlastního jména či podobných výrazů, přípona `P3` určuje stylový příznak (např. zda daný výraz patří do hovorové češtiny, slangu, cizích slov...) a `P4` obsahuje komentáře různého typu [10]. Konkrétní hodnoty, jakých může nabývat každá přípona, zde nebudu uvádět, je možné si je dohledat v *Příručce pro morfologické anotace*, v kapitole 2.1 [32].

Tabulka 5.1: Popis přípon morfologického lema [10]

Uvozovací znaky	Význam	Příklad
_:	vid / zkratka	_:B (zkratka)
_;	typ vlastního jména	_;K (společnost, organizace, instituce), _;p: (politika, vláda, armáda)
_,	stylový příznak	_,t (cizí slovo)
_^	komentář	_^(Hnutie za demokratické Slovensko) (vysvětlení zkratky)

„*V morfologickém tagu, který má podobu řetězce o 15 pozicích, je udána informace o slovním druhu a gramatických kategoriích analyzovaného slovního tvaru.*“ [10] Co znamená konkrétní pozice je uvedeno v tabulce 5.2. Stejně jako pro morfologická lemata zde nebudu uvádět, jakých hodnot může který tag nabývat. Tato informace je případně dostupná v *Příručce pro morfologické anotace*, v kapitole 2.2 [32].

Tabulka 5.2: Popis pozic morfologického tagu [32]

Pozice	Jméno	Popis
1	POS	slovní druh
2	SUBPOS	slovní poddruh
3	GENDER	rod
4	NUMBER	číslo
5	CASE	pád
6	POSSGENDER	přivlastňovací rod
7	POSSNUMBER	přivlastňovací číslo
8	PERSON	osoba
9	TENSE	čas
10	GRADE	slovesný rod
11	NEGATION	negace
12	VOICE	stupeň
13	RESERVE1	rezerva
14	RESERVE2	rezerva
15	VAR	varianta

5.1.2 NameTag

NameTag je open-source nástroj pro rozpoznávání pojmenovaných entit v textu vyvíjený ÚFALem. Je to první volně dostupný software tohoto druhu pro český text. Detekuje entity v textu a rozřadí je do předem definovaných kategorií, jako jsou jména osob, organizace, lokace atd. Nametag je k dispozici spolu s natrénovaným lingvistickým modelem pro češtinu (*Czech Models (CNEC) for NameTag* [33]) [30].

NameTag je možné využít buď jako binárky dostupné pro Linux a Windows, nebo jako online API. Pro OSX je možné si binární soubor vytvořit kompilací zdrojového kódu, který je rovněž dostupný a k němuž je připraven makefile.

Na výstupu 5.2 je zobrazen výstup z programu NameTag pro větu „*Tomáš Garrigue Masaryk byl první prezident Československé republiky.*“. Výstupem je seznam detekovaných entit seřazených podle jejich výskytu ve vstupním textu. V prvním sloupci se nachází pořadí slova, které bylo detekováno jako entita, druhý sloupec obsahuje typ entity a třetí dané slovo.

Výstup 5.2: Výstup rozpoznávání entit programu NameTag

```
1,2,3 P Tomáš Garrigue Masaryk
1 pf Tomáš
2 ps Garrigue
3 ps Masaryk
7,8 gc Československé republiky
```

Typů entit je poměrně mnoho, v tabulce 5.3 jsou vypsány nadtypy entit a pouze několik příkladů jejích podtypů. Celou tabulku je možné si prohlédnout ve *Zpracování pojmenovaných entit v českých textech* v tabulce 4.2 [10].

Tabulka 5.3: Typy entit [10]

Značení	Popis	Příklady podtypů
a	čísla jako usoučásti adres	*ah* (číslo popisné), *az* (PSČ)
c	součásti bibliografických údajů	*cs* (název článku), *cp* (číslo strany)
g	geografické názvy	*gc* (státní útvary), *gt* (kontinenty), *gu* (obce, hrady a zámky)
i	názvy institucí	*if* (firmy, koncerny, hotely), *io* (státní a mezinárodní instituce, politické strany a hnutí, náboženské skupiny)
m	názvy médií	*mi* (internetové odkazy), *mn* (periodika, redakce, tiskové agentury)
n	čísla se specifickým významem	*na* (věk), *np* (číslo jako součást jména osoby)
o	názvy věcí	*op* (výrobky), *oe* (měrné jednotky zapsané zkratkou)
p	jména osob	*pf* (křestní jméno), *ps* (příjmení), *pc* (obyvatelská jména)
q	čísla s významem počtu a pořadí	*qc* (číslo s významem počtu)
t	časové údaje	*tc* (století), *td* (den), *tm* (měsíc)

5.1.3 SubLex

SubLex (neboli *Czech Subjectivity Lexicon*) je seznam citově zabarvených českých slov (jejich lemat) zveřejněný ÚFALem. Seznam obsahuje 4626 položek, z toho 1672 pozitivních a 2954 negativních. Slovník byl vytvořen automatickým překladem z anglického lexikonu *English Subjectivity Lexicon* [34].

5.1.4 Node.js moduly

Node.js modulů pro text mining existuje více, žádný z těch jazykově závislých ale nedisponuje podporou pro český jazyk. Příkladem mohou být moduly *natural*[7], který disponuje množstvím nástrojů pro analýzu textů (tokenizace, stemování, klasifikace a další), *node-ner*[8], který využívá pro rozpoznávání entit *Stanford Named Entity Recognizer*, nebo na internetu oblíbený modul *sentiment*[9], který provádí analýzu sentimentu na základě seznamu citově zabarvených slov v angličtině. Teoreticky by bylo možné modul použít pouze s výměnou slovníku, ale kvůli skloňování slov v českém jazyce by nebylo možné porovnávat slova přímo z textu s těmi ve slovníku a analýza by tedy nefungovala.

Pro metody shlukování a klasifikace existují jazykově nezávislé moduly jako např. *kMeans*[10] pro shlukování nebo *limdu*[11], *brain*[12] či *bayes*[13] pro klasifikaci. Pro ty je ale třeba mít natrénované modely pro češtinu, které ale zatím nejsou k dispozici.

5.1.5 Závěrečné práce

Závěrečné práce z různých univerzit jsou kromě ÚFALu nejbohatším zdrojem nástrojů pro analýzu textů v češtině. Nevýhodou je, že nástroje se nedají jednoduše stáhnout a začít využívat a rovněž nemají své online verze. Závěrečné práce se zabývají rozpoznáváním entit a někdy i vztahů mezi nimi, analýzou sentimentu či emocí, nebo také klasifikací či shlukováním českých textů. Žádný z nástrojů ale není dostupný jako Node.js modul.

5.1.6 Ostatní nástroje

Mimo výše uvedené existuje množství dalších nástrojů. Příkladem mohou být *Stanford Named Entity Recognizer*[14] pro rozpoznávání entit, *Open Calais*[15] poskytující komplexní textovou analýzu (rozpoznání entit, témat, událostí,

[7] https://github.com/NaturalNode/natural
[8] https://github.com/26medias/node-ner
[9] https://github.com/thisandagain/sentiment
[10] https://github.com/emilbayes/kMeans.js
[11] https://github.com/erelsgl/limdu
[12] https://github.com/harthur/brain
[13] https://github.com/ttezel/bayes
[14] http://nlp.stanford.edu/software/CRF-NER.shtml
[15] http://www.opencalais.com/

vztahů atd.), *DBpedia Spotlight*[16], který se snaží detekovat entity v textu a přiřadit jim globální DBpedia identifikátor, či *AlchemyAPI*[17], který rovněž poskytuje komplexní textovou analýzu. Ani jeden z těchto nástrojů ale nepodporuje češtinu.

5.2 Výběr metod analýzy textu pro implementaci

Z metod uvedených v sekci 2.3 jsem se rozhodla pro analýzu novinových článků vybrat *analýzu sentimentu* a *extrakci informací*, konkrétně *extrakci entit*. Ostatní metody pro analýzu novinových článků v této práci buď nejsou potřeba (*kategorizace*, *shrnutí dokumentu*, *rozpoznání jazyka*), nebo se spíše než pro analýzu obsahu hodí pro doporučování jiných článků čtenářům (*shlukování*). Rovněž pro tyto dvě metody existují nástroje, které se dají využít při implementaci modulů, a to MorphoDiTa, NameTag a SubLex.

5.3 Modul pro extrakci entit

V této sekci bude popsán návrh, implementace a testování modulu pro extrakci entit z obsahu stažených zpravodajských článků.

5.3.1 Návrh modulu pro extrakci entit

Modul staví na open-source programech *NameTag* a *MorphoDiTa* od Ústavu formální a aplikované lingvistiky. Za cíl si klade analyzovat entity v textu, k čemuž používá nástroj NameTag, a zjištěné entity s pomocí nástroje MorphoDiTa převést do jejich základního tvaru (např. *České republice* → *Česká republika*) a pokud možno nalézt k nim identifikátor na webu české DBpedie (`cs.dbpedia.org`). Různé výskyty entit v textu pak budou shluknuty do jednoho objektu. Pokud se totiž entita vyskytuje v textu vícekrát, což se často stává, NameTag ji vrátí detekovanou na několika řádcích, často v různých slovních tvarech. Takto není možné např. zjistit, kolikrát se entita v textu vyskytla. Cílem tedy je pod jeden název entity uložit všechny její výskyty v textu. Výhodné by rovněž bylo, aby entity typu osoba byly pod jedním objektem, ať už by se na ně v textu odkazovalo pomocí jména a příjmení, nebo pouze pomocí příjmení.

Jelikož NameTag a MorphoDiTa jsou nástroje napsané v jazyce C++, je potřeba je v Node.js buď spouštět pomocí spustitelných programů nebo využívat jejich online API. Kvůli snaze nevytěžovat online API se modul snaží využít spíše spustitelné programy nástrojů. Ty lze v Node.js spustit pomocí funkcí `spawn` nebo `exec` z modulu *child_process*.

[16] https://github.com/dbpedia-spotlight/dbpedia-spotlight
[17] http://www.alchemyapi.com/

Modul si rovněž klade za cíl, aby fungoval na Linuxu, Windows i OSX, ačkoli NameTag i MorphoDiTa jsou pro každý systém dostupné různě.

Vstupem do modulu bude text pro extrakci entit. Výstupem pak bude objekt obsahující jednotlivé entity podle jejich identifikátoru a každá entita bude mít definovaný její základní tvar, seznam výskytů v textu a pokud možno odkaz na svůj ekvivalent v české DBpedii.

5.3.1.1 Převedení entity na základní tvar

Převedením entity na její základní tvar je zde myšlen převod entity z tvaru, v jakém se nachází ve vstupním textu, na její základní, slovníkový, tvar. Např. *Praze* → *Praha*, *Českému vysokému učení technickému* → *České vysoké učení technické* apod.

Pro převedení entity na základní tvar je možné využít program MorphoDiTa a pomocí něho provést morfologickou analýzu vstupního textu. Jeho výstup je možné převést na pole slov s jejich analýzou. Potom bude možné pomocí indexů rozpoznaných entit z výstupu NameTag přistupovat do tohoto pole a mít tak pro každou entitu i její morfologickou analýzu. Např. na výstupu 5.2 je vidět, že extrahovaná entita *Masaryk* má index 3, což je i číslo řádku ve výstupu 5.1, kde se nachází morfologická analýza této entity.

Jednoduchá je situace pro jednoslovné entity, kdy stačí podle indexu výskytu entity v textu najít její morfologickou analýzu a v ní její lema, což je pak základní tvar entity. Pokud se jedná o zkratku, je možné z lematu využít případný komentář, kde je rozepsáno, co zkratka znamená. Např. lema pro slovo *EU* je `EU-1_:B_;K_;p_^(Evropská_Unie)` a je možné z něho určit díky značce `_:B`, že se jedná o zkratku, a díky značce `;p_^(Evropská_Unie)`, jaký je plný název této entity.

Pro víceslovné entity tento způsob pro určení základního tvaru ve většině případů fungovat nebude. Například entita *Česká republika* by po převedení na základní tvar pomocí lemat vypadala jako *Český republika*. Ještě horší je situace, kdy slova v jedné entitě mají různé pády, např. entita *Řád Bílého lva*. V těchto případech je výhodné zjišťovat, zda je entita v textu vícekrát a pokud ano, zda je v textu také v základním tvaru, což se dá poznat podle toho, zda je první slovo entity v prvním pádě. Pokud se podaří entitu v textu v základním tvaru najít, není třeba ji již do základního tvaru znovu převádět. Pokud se pro danou entitu základní tvar v textu najít nepodařil, algoritmus v případě, že entita má všechna slova ve stejném pádu (např. *Česká republika*), vygeneruje pomocí MorphoDiTa generátoru správný tvar slova podle lematu a morfologického tagu, kde je změněna pouze pozice pádu (tedy pátý znak tagu) na první pád, a všechny ostatní pozice tagu jsou ponechány nezměněny. U entit, kde jednotlivá slova mají různý pád (*Řád Bílého lva*), většinou platí, že s větou se skloňují první slova do změny pádu, tedy pro entitu *Řád Bílého lva* se bude skloňovat s větou pouze slovo *řád* a další slova *Bílého lva* budou již stejná ve všech pádech entity. Algoritmus tedy převede do prvního pádu pouze slova,

která mají stejný pád jako první slovo entity, a to stejným způsobem, jako se převádí víceslovné entity se slovy se stejným pádem. Zbylá slova ponechá ve stejném tvaru.

5.3.1.2 Nalezení identifikátoru entity na české DBpedii

Jakmile algoritmus získal základní tvar entity, není těžké k entitě přiřadit odkaz na ní na české DBpedii. Pro to bude stačit poslat na DBpedii dotaz, zda existuje záznam o názvu entity a podle výsledku pak entitě přiřadit odkaz. Prioritu by měly mít výsledky, které označují entitu a nikoli kategorii. Také je potřeba dát si pozor na rozepsané zkratky, které se mohou na DBpedii nacházet pouze v nerozepsaném tvaru.

5.3.1.3 Sjednocení výskytů entit

Pokud se entita nachází v článku vícekrát, může být užitečné mít přehled o všech jejích výskytech. Algoritmus tedy každou entitu převede na její základní tvar, ze kterého extrahuje identifikátor, pod kterým ukládá výskyty každé entity. Na konci se pak snaží určit, jestli neexistují dva objekty, které ukládají výskyty stejné entity. To se může stát u jmen osob, kdy je daná osoba v článku zmíněna pod jménem i příjmením, ale i pouze pod příjmením. Takovéto entity se pak spojí do jednoho objektu.

5.3.2 Implementace modulu pro extrakci entit

Na ukázce kódu 5.3 je zobrazen zjednodušený pseudokód modulu pro extrakci entit. Níže popíši detailněji jednotlivé kroky procesu.

Vstup a výstup modulu Modul přijímá na vstupu řetězec obsahující text, ve kterém mají být rozpoznány entity, dále číslo portu, na kterém má být spuštěn MorphoDiTa server (nebo hodnotu 0, pokud server nemá být spuštěn), a callback, který se má zavolat po dokončení extrakce. Callback má čtyři parametry, a to

- `error`, vyplněný v případě, že nastala chyba,
- `sentences`, což je pole obsahující indexy slov, kterými začínají věty,
- `words`, což je pole obsahující pro každé slovo ve větě další pole o třech položkách, a to tvar slova ve vstupním textu, jeho lema a morfologický tag,
- `entities`, což je objekt obsahující získané entity. Každá entita je zde uložena pod vlastním identifikátorem a dále má vlastnosti `name`, která obsahuje základní tvar entity, `type` obsahující typ entity, `dbpedia` obsahující odkaz na entitu na české DBpedii, a pole `occurrences`, které

obsahuje výskyty entity v textu, kde každý výskyt je ve tvaru pole obsahujícího indexy slov entity, typ entity a tvar entity ve větě.

Spuštění MorphoDiTa serveru Program MorphoDiTa nabízí možnost kompilace a následného použití REST API serveru. Modul využívá v průběhu extrakce entit program MorphoDiTa několikrát, nejprve pro morfologickou analýzu celého vstupního textu, a dále pro generování slov v určitém tvaru. Pokud se MorphoDiTa spustí pomocí spustitelného souboru, při každém dotazu načítá lingvistický model, což trvá kolem dvou až tří vteřin. Celá extrakce entit je pak velmi pomalá a vzrůstá s počtem detekovaných entit. Z tohoto důvodu jsem se rozhodla využít raději MorphoDiTa server, který trénovací model načítá pouze jedenkrát při jeho spuštění.

Bohužel na počítačích s operačním systémem Windows je jeho kompilace značně komplikovaná a nehodí se pro běžné uživatele. Pokud je tedy modul spuštěn na Windows, modul místo vlatního MorphoDiTa serveru využívá online API poskytované ÚFALem[18].

Modul přijímá na vstupu parametr `port`, kde uživatel definuje, na kterém portu chce, aby MorphoDiTa server běžel. Zadáním hodnoty `0` lze nastavit, aby se MorphoDiTa server nespouštěl. To je výhodné v případě hromadného zpracování, kdy je možné server spustit před začátkem procesu a při zpracovávání jednotlivých souborů se využívá již spuštěný server a není nutné si spouštět vlastní instanci. Modul pak nabízí metody `startMorphoditaServer(port)` a `endMorphoditaServer()` pro samostatné spuštění a ukončení serveru.

Pokud se uživatel nachází na Linuxu nebo OSX, modul zkontroluje, zda má zkompilovaný MorphoDiTa server, a pokud ne, tak ho zkompiluje. Jakmile je spustitelný soubor připravený, modul ho spustí pomocí funkce `spawnSync` z Node.js modulu *child_process*.

Ukázka kódu 5.3: Pseudokód modulu pro získání entit

```
1  var start = function(inputText, port) {
2    if (process.platform != 'win32') {
3      if (!isMorphoditaServerCompiled())
         compileMorphoditaServer();
4      if (port > 0 || !isMorphoditaServerRunning())
         startMorphoditaServer();
5    }
6
7    var taggedText = getTaggedText(inputText);
8    var ner = getNerArray(taggedText);
9    preprocessEntitiesList(ner);
10
11   var entities = {};
12   for (var i = 0; i < ner.length; i++) {
13     var name = getEntityName(taggedText, ner[i]);
```

[18] http://lindat.mff.cuni.cz/services/morphodita/api-reference.php

```
14      var id = getEntityIdentifier(taggedText, ner[i],
            name);
15      if (!entities[id]) {
16        setNewEntity(entities, ner[i], name, id);
17        if (isEntityInBasicForm(taggedText, ner[i]))
              entities[id].nameFinal = true;
18        else entities[id].nameFinal = false;
19      } else if (!entities[id].nameFinal &&
            isEntityInBasicForm(taggedText, ner[i])) {
20          entities[id].name = name;
21          entities[id].nameFinal = true;
22      }
23    }
24    joinSurnameToName(taggedText, entities);
25    setDBpediaLinks(entities, taggedText);
26
27    return entities;
28  }
29
30  process.on('exit', (code) => {
31    if (morphoditaServerStarted()) endMorphoditaServer();
32  });
```

Získání morfologické analýzy vstupního textu Jakmile je MorphoDiTa server spuštěn, je možné na něj poslat požadavek o morfologickou analýzu vstupního textu. V případě uživatele Windows modul posílá požadavek místo na lokální server na online API od ÚFALu. Z výsledku se pak vytvoří dvoudimenzionální pole, kde pro každý token ze vstupního textu existuje pole s jeho tvarem v textu, lematem a morfologickým tagem.

Rozpoznání pojmenovaných entit Pojmenované entity se získají pomocí programu NameTag od ÚFALu. Platí zde stejné zpomalení, jako pro spuštění MorphoDiTa, tedy při každém spuštění programu se nejdříve načítá lingvistický model, což trvá asi dvě až tři vteřiny. NameTag ale možnost spuštění lokálního serveru nenabízí, není tedy možné se tomuto zpomalení vyhnout. Naštěstí NameTag je v průběhu procesu získání entit spuštěn pouze jednou, zpomalení se tedy nenačítá, jako by se načítalo u MorphoDiTy.

Problémem je použití NameTagu na OSX, pro který program nenabízí spustitelný soubor. Místo něho je tedy pro rozpoznání entit využito online API NameTagu poskytované ÚFALem[19].

Jako vstup do NameTagu se používá výstup z morfologické analýzy. Bylo by možné použít i přímo vstupní text, problémem ale při tomto řešení je, že NameTag používá jinou verzi tokenizeru než MorphoDiTa a při některých

[19] http://lindat.mff.cuni.cz/services/nametag/api-reference.php

vstupních textech je pak rozdělení na tokeny jiné, což by vedlo k tomu, že indexy slov rozpoznaných entit by neseděly s polem slov z morfologické analýzy. Takto je možné předat NameTagu již tokenizovaný text a popsaný problém tak nenastává.

Přezpracování seznamu entit NameTag vrátí seznam rozpoznaných entit ve vstupním textu. Entity jsou seřazeny za sebou tak, jak se nachází i v textu. Víceslovné entity jsou v seznamu nejprve uvedeny jako celá entita se všemi slovy a na dalších řádcích je pak každé slovo uvedeno zlvášť, jako je vidět na výstupu 5.2 pro entitu *Tomáš Garrigue Masaryk*. V předzpracování seznamu entit jsou ponechány pouze řádky s celou entitou.

Vytvoření seznamu entit Pro každý řadek z předzpracovaného seznamu entit se zavolá funkce pro převedení entity do základního tvaru. Z tohoto tvaru je pak vytvořen identifikátor entity, který slouží jako klíč v objektu entit, jelikož každý výskyt dané entity by se měl převést na stejný identifikátor. Pokud je tedy entita v textu nalezena poprvé, je vytvořena vlastnost v objektu entit a k té jsou přiřazeny další vlastnosti jako základní tvar či pole výskytů dané entity v textu. Pokud již dříve byla daná entita v textu rozpoznána, je její výskyt přidán do pole výskytů ke stejnému identifikátoru. Pokud právě zpracovávaná entita ještě nebyla v textu nalezena v základním tvaru a aktuální výskyt se v něm nachází, je přepsán i zjištěný základní tvar entity.

Základní tvar entity je zjišťován způsobem popsaným v návrhu modulu, tedy podle toho, zda se jedná o jednoslovnou nebo víceslovnou entitu, jsou některá slova entity vygenerována pomocí MorphoDiTa generátoru v potřebném tvaru.

Identifikátor je pak vytvořen jako spojená lemata všech slov v entitě. Např. pro entitu *Řád Bílého lva* má identifikátor tvar *řád_bílý_lev*. Díky tomu je možné spojit i výskyty rozepsaných entit a zkratek. Častým příkladem ve zpravodajských článcích je entita *Evropská unie*, která se může v textu vyskytovat rovněž jako zkratka *EU*. Díky morfologické analýze modul ví, že *EU* je zkratka entity *Evropská unie* a díky vytvořenému identifikátoru *evropský_unie* spojí jak výskyty rozepsaného názvu, tak zkratky.

Připojení seznamu entit s příjmením k seznamu entit s celým jménem Velmi často se ve zpravodajských serverech stává, že se entita typu osoba vyskytuje v textu jak celým jménem, např. *Edvard Beneš*, tak pouze příjmením, tedy *Beneš*. Tím se pro jednu entitu vytvoří dva identifikátory a tedy dvě skupiny entit místo jedné. V této části dojde je spojení takovýchto seznamů do jedné skupiny entit, kde budou výskyty jak jména a příjmení, tak pouze příjmení.

Nalezení odkazů do české DBpedie Díky zjištěnému základnímu tvaru je možné poslat dotaz na endpoint české DBpedie, který zjistí, zda existuje záznam pod jménem zjištěného základního tvaru entity. Příklad dotazu je zobrazen na ukázce kódu 5.4. Někdy se samozřejmě může stát, že entita na české DBpedii neexistuje. Pro známé entity ale tento způsob funguje v pořádku. Je zde pouze nutné ošetřit možnost, že na DBpedii pro některé entity existuje záznam pouze pro jejich zkratku a nikoli pro rozepsaný tvar. Pokud se tedy entita v textu nachází jako zkratka a záznam pro její rozepsaný tvar nebyl nalezen, je potřeba vyzkoušet, zda existuje záznam pro tuto zkratku.

Bohužel každý dotaz do DBpedie trvá kolem jedné až dvou vteřin, proces to tedy značně zpomaluje. Čím více entit bylo v textu rozpoznáno, tím pomalejší proces je.

Ukázka kódu 5.4: Příklad dotazu na endpoint DBpedie

```
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?ent ?title WHERE {
  ?ent rdf:type dbo:Person ;
  rdfs:label ?title .
  FILTER (str(?title)='Tomáš Garrigue Masaryk')
}
ORDER BY ASC(?ent)
```

5.3.3 Použití modulu pro extrakci entit

Na ukázce kódu 5.5 je zobrazeno použití modulu pro extrakci entit se spuštěním vlastní instance MorphoDiTa serveru. Pro výpis entit je zde použita funkce modulu `printEntities(entities)`. Pro zpracování většího množství textů je pak dobré spustit server funkcí `startMorphoditaServer(port)` a ve funkci `start()` nastavit port na hodnotu `0`. Server je pak potřeba i vypnout, a to funkcí `endMorphoditaServer()`, kterou je možné zavolat před koncem procesu.

Ukázka kódu 5.5: Použití modulu pro extrakci entit

```
const ner = require("../src/ner.js");
var fileContent = 'Edvard Beneš byl druhý prezident Č
    eskoslovenska. Výrazným výsledkem Benešovy
    diplomacie bylo uznání Československé národní rady n
    ěkolika státy.';

ner.start(fileContent, 1123, function(error, sentences,
     words, entities) {
  if (error) return console.error(error);
  ner.printEntities(entities);
});
```

Na výstupu 5.6 je zobrazen výstup programu na ukázce kódu 5.5, tedy extrahované entity textu „*Edvard Beneš byl druhý prezident Československa. Výrazným výsledkem Benešovy diplomacie bylo uznání Československé národní rady několika státy.*“.

Výstup 5.6: Výstup programu na ukázce 5.5

```
1  Edvard Beneš
2     key: edvard_beneš
3     type: P
4     length: 2
5     dbpedia: http://cs.dbpedia.org/resource/Edvard_Beneš
6     occurences:
7        1,2, P, Edvard Beneš, 1
8        11, ps, Benešovy, 2
9
10 Československo
11    key: československo
12    type: gc
13    length: 1
14    dbpedia: http://cs.dbpedia.org/resource/Č
         eskoslovensko
15    occurences:
16       6, gc, Československa, 1
17
18 Československá národní rada
19    key: československý_národní_rada
20    type: io
21    length: 3
22    dbpedia: http://cs.dbpedia.org/resource/Č
         eskoslovenská_národní_rada
23    occurences:
24       15,16,17, io, Československé národní rady, 2
```

5.3.4 Testování modulu pro extrakci entit

Nejpodstatnější část modulu tvoří externí program NameTag, který rozpoznává entity v textu a na výstup posílá seznam rozpoznaných entit. Modul pak tento seznam pouze upraví a doplní o další informace. Samotný NameTag byl již otestován jeho autory. Momentálně je to nejpřesnější NER nástroj pro český jazyk [30], dosahuje F-míry 82,82 [35].

Seskupení a převedení entity do základního tvaru bylo otestováno na dvaceti článcích. Jelikož je zde stejný problém jako u parseru, tedy špatná ověřitelnost výsledku, byly články opět manuálně zkontrolovány. Ve všech článcích se nachází celkem 405 entit se 772 výskyty. Seskupování funguje ve většině případů. Výjimek je pár, například příjmení „Sobotka“ bývá NameTagem označováno jako obec, a pak algoritmus příjmení k celému jménu nepřiřazuje.

Někdy se také nepovede vygenerovat cizí jména v základním tvaru, pak je rovněž seskupení znemožněno. Převedení do základního tvaru funguje vesměs dobře, problém se vyskytne ojediněle. Očekávaně se špatně převede víceslovná entita, pokud její další slova jsou ve stejném pádu, jako je entita ve větě (například tvar *Rady Evropy* se převede na *Rada Evropa*). Takový případ se ale vyskytl pouze jednou. Další potíží je, že se v textech článků občas vyskytují překlepy, což znemožňuje převedení na základní tvar. Stejně tak převodu brání přivlastňovací jména (např. *Nečasův*, *Cameronův*), pro která nelze vygenerovat základní tvar jména.

Celkově z 405 entit

- byly 4 špatně převedeny kvůli překlepům,
- 5 zůstalo ve tvaru přivlastňovacího jména (ale vlastní jméno bylo extrahováno z článků také, pouze z jiného výskytu osoby),
- dalších 5 bylo špatně určeno NameTagem (např. *pražské letiště Václava Havla* bylo rozděleno do dvou entit, stejně tak jméno *Veronika Sušová-Salminen* bylo rozděleno na jméno před pomlčkou a po pomlčce),
- 5 bylo špatně převedeno do základního tvaru (např. výše zmíněná *Rada Evropy*, ale také cizí názvy jako *Husíové* → *Husí* nebo jména, např. *Jelčenkem* → *Jelčenkem*),
- u 3 entit nebylo správně přiřazeno příjmení k celému jménu (z toho dvakrát zmíněné příjmení/obec *Sobotka* a jedenkrát jméno i s prostředním jménem nebylo přiřazeno k jménu bez prostředního jména),
- 1 entita byla špatně rozepsána kvůli špatnému určení zkratky NameTagem (*AP* → *adoptivní péče*).

Ze 772 výskytů bylo tedy špatně rozpoznáno či přiřazeno 23 výskytů, to jsou necelá 3 % z rozpoznaných výskytů entit.

Nalezení odkazu do DBpedie se daří podle toho, zda je entita správně převedena na základní tvar, a zda na české DBpedii existuje. To je pak velmi individuální. Velmi často se najdou odkazy pro státy či geografické útvary, hůře již pro ne příliš známá jména osob apod.

Celkově převádění entit do základního tvaru a nalezení odkazu do DBpedie funguje pro většinu entit, jelikož ve zpravodajských článcích se často vyskytují entity několikrát a mnohdy i v jejich základním tvaru. Většinou se jedná o známé entity typu států a důležitých osob či organizací, pro které existuje záznam v DBpedii. Do budoucna by každopádně bylo vhodné rozsáhlejší testování na větší množině článků.

5.4 Modul pro analýzu sentimentu

V této části bude popsán návrh, implementace a testování Node.js modulu pro jednoduchou analýzu sentimentu českého textu.

5.4.1 Návrh modulu pro analýzu sentimentu

Analýzu sentimentu je možné provést dvěma metodami, jak bylo vysvětleno v sekci 1.3.2.4. První z nich je strojové učení, pro které je třeba mít k dispozici sadu trénovacích dat, druhou je slovníková metoda, kdy je třeba mít k dispozici slovník pozitivních a negativních slov. Jelikož pro češtinu se mi nepodařilo najít žádnou volně dostupnou sadu trénovacích dat a naopak existuje slovník citově zabarvených slov SubLex (viz sekce 5.1.3), zvolila jsem si slovníkovou metodu.

Díky povaze analyzovaných textů není třeba řešit předzpracování textu ve formě oprav překlepů (které se vyskytují ojediněle) či nespisovných slov, ani případné doplnění diakritiky, což jsou běžné problémy analýzy sentimentu textů stažených z internetu, např. komentářů k produktům nebo textů ze sociálních sítí.

5.4.1.1 Vstup a výstup modulu

Modul bude na vstupu přijímat buď čistý text a nebo výstup textu z programu MorphoDiTa rozdělený do pole slov se svým tvarem ve větě, lematem a morfologickým tagem. To je kvůli předpokladu, že modul bude v analýze novinových článků spouštěn po použití modulu pro extrakci entit, který již morfologickou analýzu textu provedl a vrací ji na výstup. Zde je tedy možné ji znovu použít.

Jako výstup bude modul vracet celkový sentiment textu (tedy součet hodnot nalezených citově zabarvených slov) a pak sentiment zvlášť pro každou větu vstupního textu. Hodnoty vrátí zavoláním callbacku, který bude přijímat na vstupu s parametry

- `error`, který bude vyplněn v případě chyby,
- `sentiment`, který bude obsahovat výslednou hodnotu sentimentu, a
- `sentences`, což bude pole, kde pro každou větu budou dvě buňky, první sdělující skóre pozitivního sentimentu a druhá skóre negativního sentimentu.

5.4.2 Implementace modulu pro analýzu sentimentu

SubLex obsahuje seznam lemat, kde každé má přiřazenou informaci, zda je pozitivní (značeno *POS*) nebo negativní (značeno *NEG*), jaký je jeho slovní druh, zda se jedná o negaci, a tvar slova z angličtiny, ze kterého bylo české

slovo přeloženo. Z tohoto seznamu jsem si vytvořila objekt JSON, kde každé lema má svůj klíč a jeho jednota je 1, pokud je slovo pozitivní, a -1, pokud je negativní. Občas se po převedení stalo, že z důvodu překladu slov se ve slovníku nacházela dvě stejná slova, ale každé s jiným sentimentem. Taková slova jsem z objektu vyřadila.

Hlavní část algoritmu pracuje na jednoduchém principu. Pro každé slovo v textu se zjistí jeho lema a to je pak porovnáno se slovy ve slovníku. Pokud slovo bylo nalezeno, přičte se větě sentiment podle jeho hodnoty, tedy +1 pro pozitivní slova a -1 pro negativní slova. Na ukázce kódu 5.7 je zobrazena funkce pro výpočet sentimentu textu po větách.

Ukázka kódu 5.7: Zjištění sentimentu textu po větách

```
function getSentimentOfSentences(sublex, words) {
  var sentiment = [];
  var sentenceIdx = 0;
  sentiment[sentenceIdx] = [0, 0];
  for (var i = 0; i < words.length; i++) {
    if (words[i][0] == '' && i != words.length - 1) {
      sentenceIdx++;
      sentiment[sentenceIdx] = [0, 0];
      continue;
    }
    if (sublex.hasOwnProperty(words[i][1])) {
      if (sublex[words[i][1]] == 1) sentiment[
          sentenceIdx][0]++;
      else sentiment[sentenceIdx][1]--;
    }
  }
  return sentiment;
}
```

Lemata slov v textu jsou buď zadána již na vstupu, nebo se vytvoří pomocí programu MorphoDiTa. V tomto modulu jsou použity dostupné spustitelné soubory, nikoli server jako v modulu pro rozpoznání entit, jelikož se MorphoDiTa spouští pouze jednou.

5.4.3 Použití modulu pro analýzu sentimentu

Jedno z možných použití modulu je zobrazeno na ukázce kódu 5.8. Výsledek je pak zobrazen na výstupu 5.9.

Ukázka kódu 5.8: Použití modulu pro analýzu sentimentu

```
const sentiment = require("../src/sentiment.js");
const input = 'Dnes byl báječný den. Každému se dařilo.
    Ale počasí bylo špatné.';

```

```
4 sentiment.sentiment(input, function(error, sentiment,
      sentences) {
5   if (error) return console.error(error);
6   console.log("Sentiment textu je " + sentiment);
7   console.log(sentences);
8 });
```

Výstup 5.9: Výstup analýzy sentimentu spuštěné kódem 5.8

```
1 Sentiment textu je 1
2 [ [ 1, 0 ], [ 1, 0 ], [ 0, -1 ] ]
```

5.4.4 Testování modulu pro analýzu sentimentu

Modul byl otestován na 32 ručně anotovaných článcích ze čtyř serverů. Článkům byla přiřazena anotace *pozitivní*, *neutrální* nebo *negativní*. Jelikož modul vrací sentiment číselně, byly stanoveny intervaly pro porovnání se slovním hodnocením. Negativní jsou body sentimentu v intervalu $(-\infty, -4\rangle$, neutrální v intervalu $\langle -3, 3\rangle$ a pozitivní v intervalu $\langle 4, \infty)$. Interval pro neutrální sentiment byl lehce rozšířen od nuly na obě strany, protože vzhledem k průměrné délce článku je velmi nepravděpodobné, aby se počet pozitivních a negativních slov přesně shodoval a sentiment pak byl přímo nulový. Mírně záporný nebo mírně kladný sentiment tedy může znamenat i neutrálně napsaný článek.

Z 32 testovaných článků se výsledek algoritmu neshodoval s manuálně určeným zařazením v 7 případech, což je 21,9 %. Je nutné ale dodat, že určení sentimentu článku je obtížné i manuálně, zvlášť u zpravodajských článků, které mohou informovat v jednom textu jak o pozitivních, tak o negativních událostech. Není také jasné, co pak takový sentiment vypovídá a velmi záleží na jeho interpretaci.

Analýzu sentimentu by bylo vhodné v budoucnu otestovat na větším vzorku dat. Jelikož analýza sentimentu v textu je poměrně rozsáhlý problém, jehož řešení se stále vyvíjí, bylo by také vhodné uvažovat o rozšíření algoritmu a tedy zpřesnění analýzy tohoto modulu.

Kapitola 6

Experiment na českých zpravodajských serverech

V této kapitole budou využity implementované moduly k získání množiny zpravodajských článků spolu s jejich textovou analýzou (tedy seznamem extrahovaných entit a zjištěným sentimentem). Získaná data pak budou vložena do databáze, nad kterou budou připraveny komplexní dotazy. V závěru kapitoly budou popsány výsledky těchto dotazů a ty pak budou porovnány pro jednotlivé zpravodajské servery.

6.1 Stažení, extrakce a analýza článků

Proces získání vyparsovaného obsahu článků a jejich textová analýza byl rozdělen do dvou částí. To je z toho důvodu, že analýza entit poměrně dlouho trvá, což zbytečně zdržuje crawler a parser. Také je výhodné mít možnost analýzu článků přerušit a pokračovat později. Pro každou část byl vytvořen jednoduchý Node.js program s ovládáním přes příkazovou řádku.

V první části procesu získání dat se pomocí crawleru stáhnou články z vybraného zpravodajského serveru a vybrané sekce do zadaného data. Pro každý článek je pak pomocí parseru získán relevantní obsah, ke kterému je přidán název serveru a název sekce, ze kterých byl článek získán, a URL adresa článku. Všechny tyto informace se pak ve formátu JSON uloží do souboru s názvem vygenerovaným z vyparsovaných informací. Každý server má svojí složku, kam se články ukládají.

Použití programu je zobrazeno na ukázce vstupu 6.1. Na tomto konkrétním příkladě budou staženy a uloženy parsované články ze serveru *Novinky.cz*, ze sekce *domácí* a od těch nejnovějších článků do článků s datem *1. května 2016*.

Ukázka vstupu 6.1: Ukázka vstupu programu pro získání parsovaných článků

```
1 node get-data.js Novinky domácí 2016-05-01
```

V druhé části procesu jsou čteny soubory uložené první částí a pro obsah článků je spuštěna analýza entit a analýza sentimentu. Jejich výstupy jsou pak přidány do objektu získaného v první části. Takto obohacená data jsou pak uložena do složky, která bude později fungovat jako vstup do databáze. Jakmile je nový soubor uložen, starý je ze složky s parsovanými soubory odstraněn. V příloze D je pro ukázku zobrazen výstupní soubor krátkého článku.

Na ukázce vstupu 6.2 je zobrazena ukázka použití programu pro zpracování článků stažených ze serveru *Novinky.cz*. Proces je možné kdykoli přerušit a pro dokončení spustit znovu.

Ukázka vstupu 6.2: Ukázka vstupu programu pro textovou analýzu článků a jejich uložení do složky se zpracovanými články

```
node process-data.js Novinky
```

První i druhá část mohou být spuštěny paralelně, jelikož jakmile druhá část zpracuje všechny články, které se nacházely v adresáři při jejím spuštění, začně rovnou zpracovávat nové. To se opakuje, dokud není složka prázdná.

Pro experiment byly staženy články ze serverů *Novinky.cz*, *iDnes.cz*, *Aktualne.cz* a *ParlamentniListy.cz*, a to z období od 1. ledna do 30. dubna 2016.

6.2 Databáze pro kolekci získaných článků

Pro možnost snadného dotazování nad získanou kolekcí zpravodajských článků jsem použila databázi *Apache Solr*[20] ve verzi 5.5.0. Databáze umožňuje zaindexovat články uložené v souborech ve formátu JSON a poskytuje webové rozhraní a REST API pro dotazování nad zaindexovanými daty. Data je možné do databáze nahrát jednoduchým příkazem, kde se pouze definuje složka s uloženými soubory.

Solr poskytuje dotazovací jazyk podobný SQL. V databázi je možné filtrovat články podle zvolených polí, k dispozici je množství operátorů a modifikátorů. Je možné definovat, která pole se mají zobrazit na výstupu, kolik zobrazit záznamů, podle čeho výstupy řadit atd. Například dotaz zobrazený na vstupu 6.3 zobrazí články, které mají v nadpisu slovo *počasí*, obsahují klíčové slovo *déšť* nebo *sníh*, a byly vydány mezi 12. lednem a 16. dubnem 2016.

Ukázka vstupu 6.3: Příklad dotazu v Solr

```
header:*počasí* && keywords:(déšť || sníh) && date:
    [2016-01-12T00:00:00.000Z TO 2016-04-16
    T23:59:59.999Z]
```

Pro složitější dotazy je možné využít REST API jakýmkoli programovacím jazykem, kde je možné posílat HTTP požadavky.

[20] http://lucene.apache.org/solr/

6.3 Dotazy nad kolekcí dat

Jako již bylo popsáno výše, nad daty je možné se dotazovat pomocí webového rozhraní. Tak je možné zjišťovat například kolik za dané období bylo zveřejněno článků obsahujících jednu nebo více entit, jaké byly v dané období vydány negativní články atd. Také je možné přímo vyhledávat v textu článku a výsledky omezit dalšími parametry. Kromě tohoto běžného dotazování je možné využít databázové API a napsat složitější dotazy, např. pro generování grafů či jiných souborů s daty za určité časové období.

Pro experimenty jsem napsala několik dotazů v Node.js, jejichž nastavení lze ovládat pomocí příkazové řádky. Níže každý z nich popíši. Jejich výsledky bude možné si prohléhnout v následující sekci.

6.3.1 Analýza sentimentu

Dotaz pro analýzu sentimentu vytvoří graf pozitivního a negativního sentimentu po dnech za určité období. Na vstupu přijímá jako parametry počáteční a koncové datum období, za kdy se graf bude vytvářet, a server, jehož články bude analyzovat. Volitelně je možné přidat jednu či více entit oddělených mezerou (pro víceslovné entity je třeba je uzavřít do uvozovek) — v tom případě bude zobrazen graf pro články, kde byly extrahovány všechny zadané entity. Ukázka použití programu se nachází na vstupu 6.4, na řádku 1 je použití pro analýzu sentimentu všech článků, na řádku dva pak použití pro články, kde se vyskytují dvě zadané entity.

Ukázka vstupu 6.4: Použití dotazu pro získání grafu sentimentu

```
1 node sentiment.js 2016-04-01 2016-04-30 iDnes
2 node sentiment.js 2016-04-01 2016-04-30 iDnes Entita_1
    "Víceslovná entita 2"
```

Dotaz si zjistí pro každý den všechny články, které vyšly na zadaném serveru. Spočítá zvlášť body negativního a zvlášť pozitivního sentimentu. Výsledky pak vydělí počtem článků, které ten den byly publikovány. Výsledný graf tedy na ose y zobrazuje průměrný počet bodů sentimentu daného typu na jeden článek.

6.3.2 Analýza entit

Dotaz pro analýzu entit prochází opět články po dnech. Za každý den zjistí 8 nejvíce vyskytovaných entit v článcích pro zadaný server. Do počtu výskytů se počítá nejen v kolika článcích se entita vyskytla, ale i počet jejích výskytů v každém článku. Na vstupu je opět možné zvolit si časové období a server, navíc je přidán parametr pro určení, jaký sentiment mají mít analyzované články. Je možné vybrat si z pozitivního (možnost `p`), negativního (možnost

`n`) nebo obojího (možnost `a`). Ukázka použití programu je zobrazena na vstupu 6.5. Výstupem dotazu je csv soubor, kde je na každém řádku datum a seznam entit oddělených čárkou, a graf zobrazující výskyty třech nejčastějších entit po dnech.

Ukázka vstupu 6.5: Použití dotazu pro získání nejvyskytovanějších entit

```
1 node entities.js 2016-04-01 2016-04-30 iDnes n
```

6.3.3 Analýza klíčových slov

Dotaz pro analýzu klíčových slov je skoro stejný jako dotaz pro analýzu entit. Jediný rozdíl je, že místo nejčastějších výskytů entit zjišťuje nejčastější výskyt klíčových slov extrahovaných z článku algoritmem parsování. Ukázka vstupu je na vstupu 6.6.

Ukázka vstupu 6.6: Použití dotazu pro získání nejvyskytovanějších klíčových slov

```
1 node kw.js 2016-03-01 2016-03-31 Novinky a
```

6.3.4 Analýza nadpisů

Dotaz pro analýzu nadpisů vrátí pro každý den ze zadaného časového období a pro daný server články, které mají nejvyšší zadaný sentiment. Pro negativní, resp. pozitivní, sentiment tedy vrátí určitý počet nejvíce negativních, resp. pozitivních, článků za daný den. Na vstupu se stejně jako v předchozím dotazu zadává časové období, server a sentiment, ale s tím rozdílem, že sentiment může být zadán pouze negativní nebo pozitivní. Na ukázce vstupu 6.7 je zobrazen jeden z možných vstupů programu.

Ukázka vstupu 6.7: Použití dotazu pro získání nadpisů článků s nejvyšším vybraným sentimentem

```
1 node headers.js 2016-03-01 2016-03-31 Aktualne p
```

6.4 Výsledky dotazů pro české zpravodajské servery

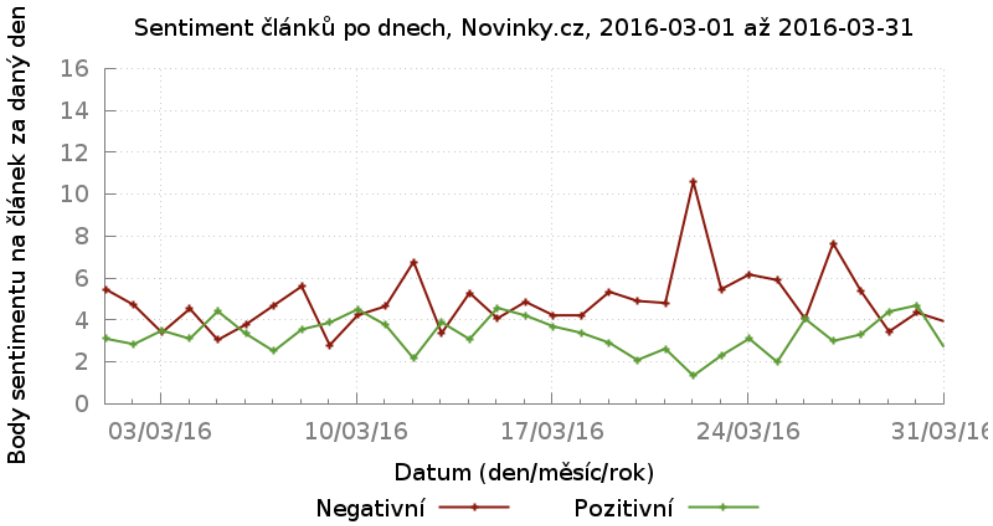
V tabulce 6.1 jsou uvedena procenta negativních, pozitivních a neutrálních článků pro každý zpravodajský server za období leden až duben 2016. Za negativní jsou považovány články se sentimentem v intervalu $(-\infty, -4\rangle$, neutrální v intervalu $\langle -3, 3\rangle$ a pozitivní v intervalu $\langle 4, \infty)$. Z tabulky je vidět, že

kromě Parlamentních listů mají všechny články více negativních článků než pozitivních, u PL je to naopak.

Tabulka 6.1: Procenta pozitivních, neutrálních a negativních článků pro vybrané servery

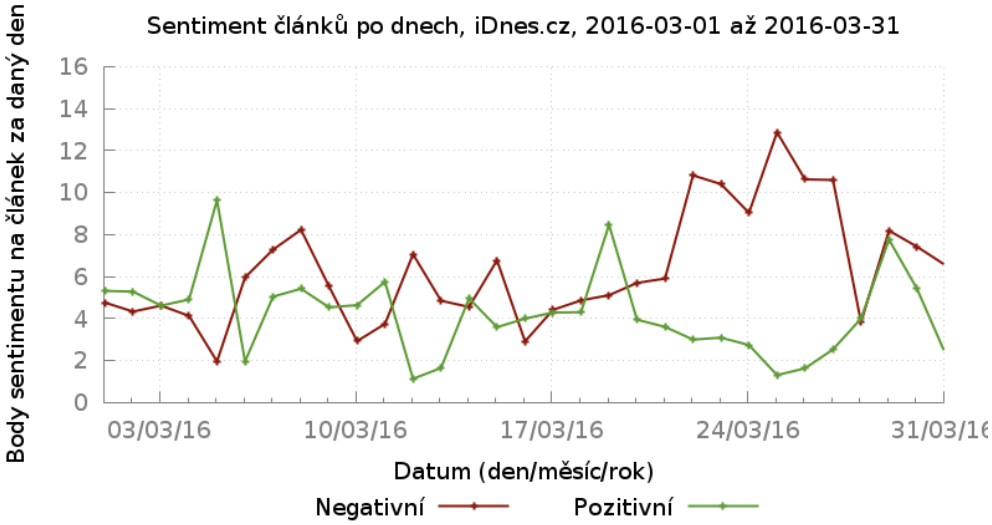
server	**negativní** [%]	**neutrální** [%]	**pozitivní** [%]
Novinky.cz	41,52	29,69	28,79
iDnes.cz	45,8	23,63	30,57
Aktualne.cz	43,47	29,17	27,35
ParlamentniListy.cz	31,05	30,49	38,46

Na obrázcích 6.1, 6.2, 6.3 a 6.4 jsou zobrazeny grafy pro každý z testovaných serverů znázorňující průměrný negativní a pozitivní sentiment na článek v březnu 2016.

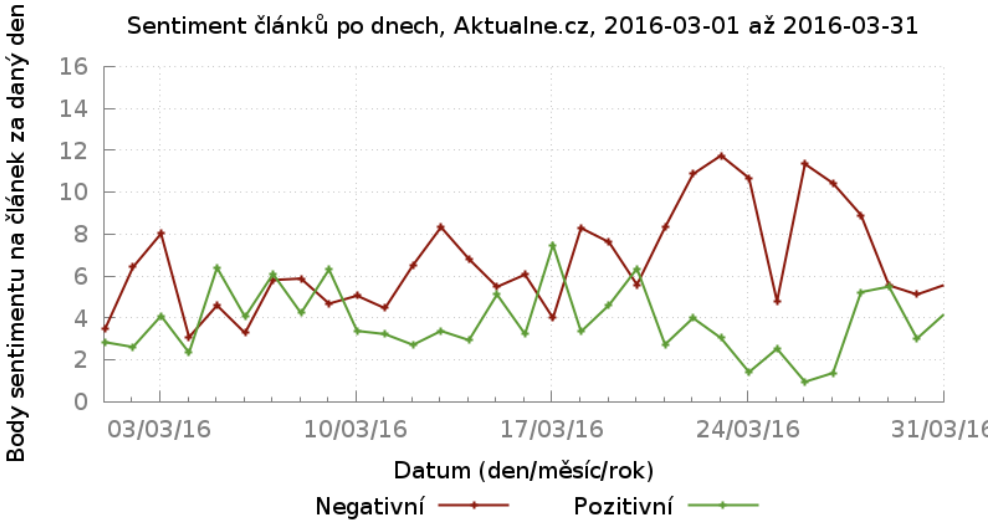


Obrázek 6.1: Sentiment článků ze serveru Novinky.cz za březen 2016

Na grafech je na první pohled poznat, že jsou velmi „zubaté“, což znamená, že zjištěný sentiment se jednotlivé dny hodně liší. Rovněž by se dalo očekávat, že grafy všech serverů si budou podobné, jelikož stažené články jsou zaměřené na stejnou tématiku (domácí a zahraniční zpravodajství), příliš tomu tak ale není. Dále se zaměřím pouze na negativní sentiment. Z 31 dnů v měsíci březnu 2016 mají negativní sentiment nad 6 bodů na článek všechny servery pouze ve čtyřech dnech, a to 12., 22., 24. a 27. Ve 23. dnu se pak shodnou všechny kromě



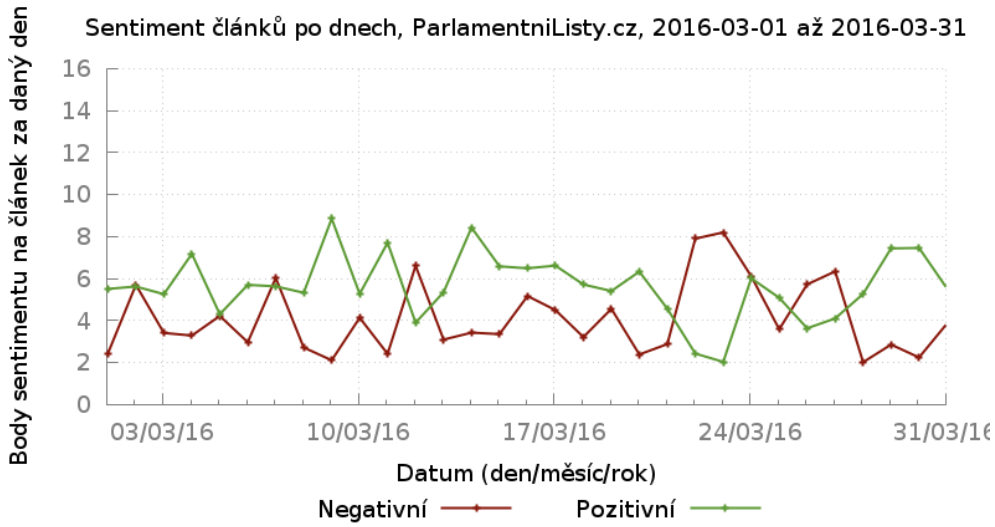
Obrázek 6.2: Sentiment článků ze serveru iDnes.cz za březen 2016



Obrázek 6.3: Sentiment článků ze serveru Aktualne.cz za březen 2016

Novinek. To může být částečně ovlivněno i tím, že některé servery mají celkově články psané s vyšším sentimentem než jiné. Konkrétně v březnu 2016 má dní nad 6 negativních bodů sentimentu server Novinky 5, iDnes 11, Aktuálně 13 a Parlamentní listy rovněž 5, což je celkem výrazný nepoměr.

Na obrázcích 6.5 a 6.6 je možné si prohlédnout grafy, které znázorňují 3 nejvyskytovanější entity a 3 nejvyskytovanější klíčová slova v daném dni pro

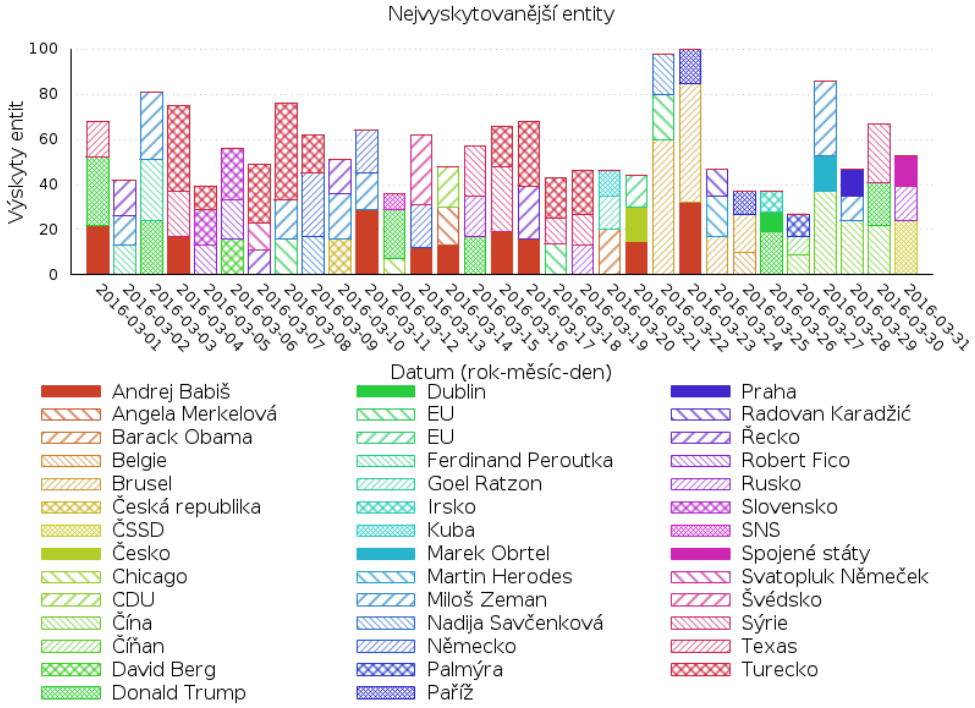


Obrázek 6.4: Sentiment článků ze serveru ParlamentniListy.cz za březen 2016

měsíc březen 2016 a server iDnes. Grafy zobrazují entity a klíčová slova pro články jakéhokoli sentimentu. V závislosti na sentimentu je možné se podívat na graf entit a klíčových slov pro výrazné negativní dny.

Pro 12. březen je z grafu entit možné zjistit, že nejvyskytovanější entity byly *Donald Trump*, *Chicago* a *SNS (Slovenská národní strana)*, pro klíčová slova pak *Demonstrace*, *Twitter* a *Islámský stát*. První dvě entity souvisí s nadcházejícími volbami amerického prezidenta (článek *Rvačka příznivců s odpůrci donutila Trumpa zrušit mítink v Chicagu*), třetí pak s již proběhlými parlamentními volbami na Slovensku (článek *Nechceme hybridy, řekl šéf SNS. Odmítl vstup do slovenské pravicové koalice*). Klíčová slova toho pak vypovídají poměrně méně. Pro klíčové slovo *Demonstrace* vypsáním negativních nadpisů pro ten den a server je možné najít tři články, ze kterých je zřejmé, že proběhla demonstrace proti České televizi, proti vládě v Polsku a pak proti uprchlíkům ve Francii v Calais. Z celého grafu klíčových slov je zřejmé, že sociální síť Twitter se stala oblíbeným nástrojem pro veřejnou komunikaci. Co se týče posledního klíčového slova, tak výpisem negativních nadpisů je možné najít článek *Radikálové z IS prý při útoku v Iráku použili chemické zbraně*.

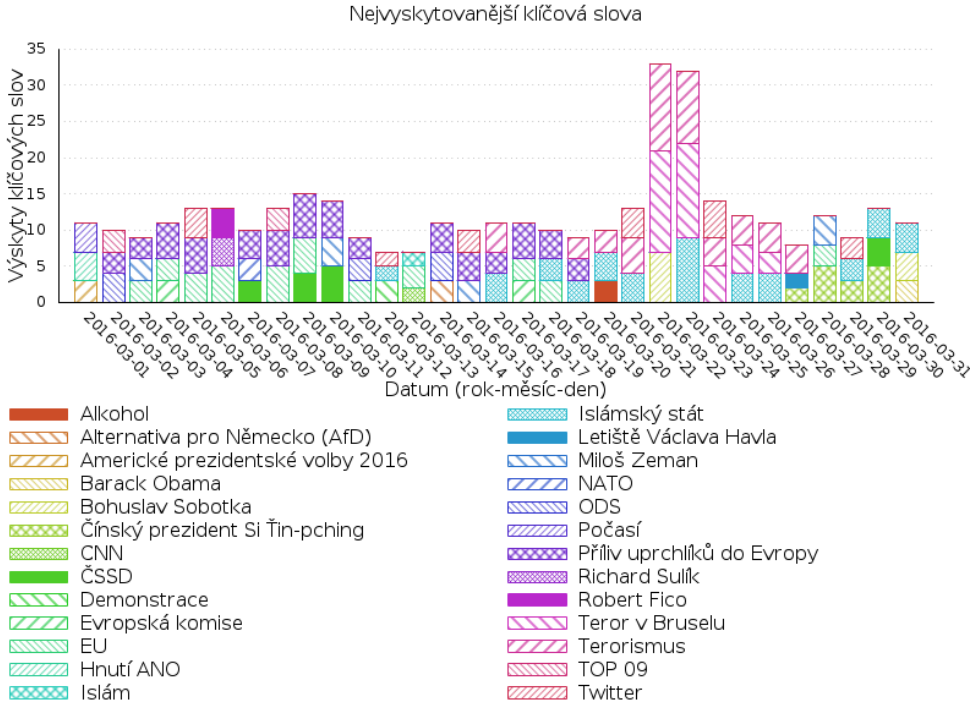
Zcela nejvýraznější je na všech grafech sentimentu období od 22. 3. 2016, kdy se staly teroristické útoky v Bruselu. Velmi dobře o této skutečnosti vypovídá i graf entit, kde v období 22.–25. 3. je nejvyskytovanější entitou *Brusel* (s výjimkou 24. 3., kde je až na druhém místě). Dále se pak mezi entitami často nachází *Paříž*, což vychází z toho, že některé články týkající se atentátů v Bruselu připomínaly i nedávné atentáty v Paříži. Další entity mezi třemi nejvíce vyskytovanými v tomto období jsou očekávaně *Belgie* či *Ev-*



Obrázek 6.5: Nejvyskytovanější entity ze serveru iDnes.cz za březen 2016

ropská unie. Dále se v první trojici vyskytují entity *Andrej Babiš* a *Martin Herodes*, což jsou jména související s v té době aktuální kauzou Čapí hnízdo. Entita *Radovan Karadžić* je pak 24. 3. na třetím místě v souvislosti s velmi negativním článkem *Karadžić dostal 40 let, podle Haagu nese vinu za Srebrenici i Sarajevo*. Entita *Nadija Savčenková* pak souvisí s článkem *Savčenková stráví za mřížemi 22 let, Ukrajina nabídla výměnu vězňů*. O útocích v Bruselu pak ještě lépe vypovídá graf klíčových slov, kde v období 22.–25. 3. se na prvních místech v pořadí výskytů nachází klíčová slova *Teror v Bruselu*, *Terorismus* a *Islámský stát*. Jediné dny, kdy tato klíčová slova neobsadila první tři místa, jsou 24. 3., kdy je na druhém místě *Twitter*, a 22. 3., kdy je na třetím místě klíčové slovo *Bohuslav Sobotka*, což souvisí jednak s vyjádřeními české vlády k útokům v Bruselu a vyhlášením opatření v České republice, ale také s jinými tématy, jako vyjádření premiéra (Sobotky) k soudu s Nadijou Savčenkovou nebo k tomu, že čeští poslanci odmítli dohodu o zrušení víz a přerozdělování uprchlíků s Tureckem.

Další výrazněji negativní den je pak 27. březen. Nejvyskytovanější entity pro ten den jsou *Palmýra* (město v Sýrii), *Čína* a *Číňan*, nejvyskytovanější klíčová slova pak *Terorismus*, *Čínský prezident Si Ťin-pching* a *Letiště Václava Havla*. Klíčové slovo *Palmýra* souvisí s článkem *Dobyli jsme Palmýru zpět, džihádisté se stáhli, tvrdí Asadova armáda*, kde se objevuje i entita *Terorismus*,



Obrázek 6.6: Nejvyskytovanější klíčová slova ze serveru iDnes.cz za březen 2016

která ale souvisí i s mnohými jinými událostmi, např. s teroristickým útokem v Pákistánu, s raziemi a protesty v Belgii či s objevením dílny nelegálních pasů pro teroristy. Ostatní entity a klíčová slova pak mají souvislost s nadcházejícím příjezdem čínského prezidenta do ČR.

Kromě výše popsaných událostí je možné vyčíst zajímavé informace zvlášť z grafů entit a klíčových slov. Spousta důležitých událostí totiž nemusí mít výrazné hodnoty sentimentu a na jeho grafu pak nejsou vidět. Na rozdíl od toho graf entit či klíčových slov může tyto události naznačovat. Kromě toho jsou z grafů zřejmá aktuální nejprobíranější témata a o to více nápadné jsou pak mezi nimi zajímavé události týkající se jiných témat.

Prvním příkladem může být nápadnost entity *Ferdinand Peroutka* (spolu s obecně častěji se vyskytující entitou *Miloš Zeman*), jejíž výskyt značí zajímavější událost, v tomto případě informaci, že vnučka Ferdinanda Peroutky uspěla u soudu s žalobou proti kanceláři prezidenta ohledně jeho vyjádření o Peroutkovi.

Dále 5. a 6. března jsou na prvním a druhém místě v grafu entit entity *Slovensko* a *Robert Fico* (slovenský premiér), což odpovídá skutečnosti, že 5. března 2016 na Slovensku proběhly parlamentní volby. Zajímavé je, že klíčová slova tuto událost příliš nenapovídají, jelikož ohledně tohoto tématu se

na prvních třech pozicích nachází pouze entita *Robert Fico* a *Richard Sulík* (slovenský politik).

14. 3. jsou pak opět nápadné entity na prvních dvou místech, a to *CDU* (německá politická strana) a *Angela Merkelová* (německá kancléřka). To je z toho důvodu, že 13. 3. se v Německu konaly zemské volby a druhý den se články věnovaly komentářům k tomuto tématu (13. se objevuje pouze entita *Německo* na druhém místě).

V období 17.–19. 3. je pak na prvním místě entita *Turecko*, což může mít spojitost se summitem Evropské unie a Turecka (17. 3.) a jejich následnou dohodou o vracení uprchlíků z Řecka do Turecka (18. 3.). Každopádně na grafu entit je možné si všimnout, že Turecko je častým tématem celého měsíce.

Dne 22. 3. proběhly teroristické útoky v Bruselu, o čemž jsem se již zmínila výše v souvislosti s grafem sentimentu.

Poslední zajímavou událostí, která je z grafu entit nápadná, je návštěva čínského prezidenta v České republice ve dnech 28.–30. 3., kdy se v těchto dnech entita *Čína* objevuje na prvním nebo druhém místě.

Z grafu klíčových slov můžeme vidět, že častými tématy jsou *Příliv uprchlíků do Evropy*, *Islámský stát, Evropská unie* a *Terorismus* (ten hlavně v druhé polovině měsíce, kdy byl zatčen atentátník z Paříže a proběhly teroristické útoky v Bruselu). Nápadná jsou z grafu klíčová slova *Teror v Bruselu* a *Čínský prezident Si Ťin-pching*, kterými jsou označeny události útoku v Bruselu a návštěvy čínského prezidenta v ČR.

Závěr

Cílem práce bylo vytvořit nástroj pro analýzu článků z českých zpravodajských serverů. Pro dosažení tohoto cíle byly po analýze zpravodajských serverů vytvořeny čtyři Node.js moduly a několik doplňujících Node.js skriptů pro jejich snadné ovládání z příkazové řádky. Moduly pak byly využity k získání množiny článků z několika zpravodajských webů, nad kterými byl posléze proveden experiment.

První vytvořený modul umožňuje crawlování čtyř českých zpravodajských serverů a stažení jimi publikovaných článků z rubriky domácí či zahraniční do určitého data. Jeho návrh umožňuje snadné rozšíření o další zpravodajské servery. Ke svému běhu využívá již existující Node.js modul *node-simplecrawler*, který byl vybrán na základě analýzy dostupných Node.js crawlerů.

Pro druhý modul bylo nutné analyzovat strukturu a obsah článků z českých zpravodajských serverů. Na základě zjištěných společných prvků byl navrhnut, implementován a otestován algoritmus, který z HTML stránek generovaných zpravodajskými servery extrahuje nadpis, datum, klíčová slova a obsah článku. Z těchto prvků bylo nejtěžší extrahovat obsah článku. Algoritmus je postaven zejména na zjišťování hustoty odkazů v textu a na pozici textových bloků vůči svému okolí, vypomáhá si ale i odstraňováním uzlů podle názvu jejich HTML značky či jejích atributů. Detekce klíčových slov je pak založena na hodnotách HTML atributů, ve kterých jsou hledána slova napovídající přítomnost klíčových slov, jako např. *tag* či *keyword*. Je v něm rovněž využit odhad konce článku, který je výstupem detekce obsahu článku. Jednodušší pak byla extrakce data, která je založena na detekci řetězce s datem na stránce pomocí regulárních výrazů. Nejjednodušší částí byla extrakce nadpisu, jelikož ten bývá vždy v nadpisu první úrovně a jeho podstatná část se shoduje s titulkem webové stránky.

Pro analýzu vyparsovaného obsahu pak byly prozkoumány možnosti textové analýzy novinových článků. Na jejím základě byly vybrány dvě nejvhodnější metody, a to extrakce pojmenovaných entit z textu a analýza sentimentu. Pro každou byl vytvořen Node.js modul na základě open-source projektů Mor-

phoDiTa a NameTag pro extrakci entit a MorphoDita a SubLex pro analýzu sentimentu. Všechny tři nástroje byly vytvořeny Ústavem formální a aplikované lingvistiky Univerzity Karlovy. První modul z textu na vstupu extrahuje seznam entit v jejich základním tvaru spolu se všemi jeho výskyty v textu a pokud možno přiřazeným odkazem na entitu na webu `cs.dbpedia.org`. Druhý modul pak analyzuje sentiment vstupního textu a vrátí jeho celkový sentiment a sentiment jednotlivých vět.

V další části práce pak byly vytvořeny skripty, které s využitím všech zmíněných modulů umožňují přes příkazovou řádku spustit stažení a analýzu článků. Jejich výstupem je množina souborů ve formátu JSON, kterou je pak možné nahrát do databáze Apache Solr. Ta poskytuje webové rozhraní, kde je možné psát jednoduché i složitější dotazy, a REST API, které bylo využito k vytvoření dotazů, jenž zobrazují vývoj v čase. Jednodušší dotazy mohou být např. vyhledání článků obsahující zadaná klíčová slova nebo entity, omezovat výsledky podle data článků nebo serveru, na kterém byl článek vydán, vyhledávat články dle sentimentu, zjišťovat sentiment článků obsahujících zadanou entitu atd. Složitější dotazy se věnovaly hlavně spuštění jednoduchých dotazů pro zadané časové období a server a vygenerování grafu na základě získaných dat. Pomocí nich je tedy možné si vygenerovat za dané období a server graf sentimentu článků nebo graf nejvyskytovanějších entit či klíčových slov.

V experimentální části bylo pomocí implementovaných nástrojů staženo 17 409 článků ze serverů *Novinky.cz*, *iDnes.cz*, *Aktualne.cz* a *Parlamentni-Listy.cz*. Pro tyto články bylo spuštěno několik dotazů a jejich výsledky byly rozebrány v poslední kapitole.

Díky možnosti uložení kolekce článků do databáze a snadnému rozšíření této kolekce pomocí implementovaných modulů vznikl flexibilní nástroj, který poskytuje rozsáhlé možnosti pro analyzování uložených článků, ať už pomocí webového rozhraní databáze Apache Solr, či pomocí předpřipravených dotazů.

V budoucnu by bylo vhodné otestovat úspěšnost modulů pro parsování, extrakci entit a analýzu sentimentu na větší množině dat, jelikož každý z těchto modulů pro ověření své správnosti potřebuje člověkem anotovanou rozsáhlejší množinu dat, na jejíž vytvoření nebyl v této práci prostor.

V práci se také nachází možnosti pro budoucí rozšíření. Nejjednodušší z nich je rozšíření crawleru o další zpravodajské servery. Přirozeným pokračováním by mohlo být vytvoření dalšího modulu (či rozšíření modulu pro extrakci entit), který by se zabýval detekcí vztahů mezi extrahovanými entitami. Pro taková data by ale možná byl vhodnější jiný typ databáze, např. grafová. Z práce by také jistě šlo vytvořit webovou aplikaci, která by svým návštěvníkům poskytovala příjemnější uživatelské rozhraní a rozsáhlejší možnosti dotazování nad uloženými daty.

Literatura

[1] Liu, B.: *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Data-Centric Systems and Applications, Springer Berlin Heidelberg, 2011, ISBN 9783642194603.

[2] Svátek, V.: Dolování dat z webu (nepublikovaná přednáška), Praha, ČVUT 2016.

[3] Feldman, R.; Sanger, J.: *The Text Mining Handbook*. New York: Cambridge University Press, 2007, ISBN 978-0-511-33507-5.

[4] Paralič, J. e. a.: *Dolovanie znalostí z textov*. Košice: Technická univerzita v Košiciach, prvé vydání, 2010, ISBN 978-80-89284-62-7.

[5] Cvrček, V.: *Mluvnice současné češtiny: Jak se píše a jak se mluví*. Mluvnice současné češtiny, Univerzita Karlova V Praze Nakladatelství Karolinum, 2010, ISBN 9788024617435.

[6] Hajič, J.: Statistické modelování a automatická analýza přirozeného jazyka (morfologie, syntax, překlad). In *Slovenčina a čeština v počítačovom spracovaní*, Bratislava: VEDA, vydavate stvo SAV, 2001, ISBN 80-224-0692-9, str. 23.

[7] Aggarwal, C.; Zhai, C.: *Mining Text Data*. Springer, 2012, ISBN 9781461432234.

[8] Chernykh, N.: *Analýza textu (text mining) pomocí vybraného softwaru*. Bakalářská práce, Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky. 2012.

[9] Sedláček, P.: Text mining a jeho možnosti (aplikace) *Fakulta informatiky Masarykovy univerzity* [online]. 2004. Dostupné z: `http://www.fi.muni.cz/usr/jkucera/pv109/2003p/xsedlac5.htm`

[10] Ševčíková, M.; Žabokrtský, Z.; Krůza, O.: *Zpracování pojmenovaných entit v českých textech*. Universitas Carolina Pragensis, 2007.

[11] HTML(5) Tutorial *W3schools.com* [online]. 2016, [cit. 2016-04-17]. Dostupné z: http://www.w3schools.com/html/

[12] Janovský, D.: Verze HTML *Jak psát web* [online]. 2016, [cit. 2016-04-12]. Dostupné z: http://www.jakpsatweb.cz/html/verze-html.html

[13] JavaScript HTML DOM *W3schools.com* [online]. 2016, [cit. 2016-04-17]. Dostupné z: http://www.w3schools.com/js/js_htmldom.asp

[14] Vitvar, T.: Web 2.0 (nepublikovaná přednáška), Praha, ČVUT 2015.

[15] Janovský, D.: Čeština na webu v HTML *Jak psát web* [online]. 2016, [cit. 2016-04-12]. Dostupné z: http://www.jakpsatweb.cz/cestina.html

[16] Setting the HTTP charset parameter *W3.org* [online]. 2016, [cit. 2016-04-18]. Dostupné z: https://www.w3.org/International/articles/http-charset/index

[17] Handling character encodings in HTML and CSS (tutorial) *W3.org* [online]. 2016, [cit. 2016-04-18]. Dostupné z: https://www.w3.org/International/tutorials/tutorial-char-enc/

[18] Singh, M.: *The Practical Handbook of Internet Computing*. Chapman & Hall/CRC Computer and Information Science Series, CRC Press, 2004, ISBN 9780203507223.

[19] Česká tisková kancelář *Základní informace* [online]. 2016, [cit. 2016-04-21]. Dostupné z: http://www.ctk.cz/o_ctk/zakladni_informace/

[20] Ireinová, H.: *Online žurnalistika a zpravodajské servery v ČR [online]*. Diplomová práce, Masarykova univerzita, Filozofická fakulta, Brno, 2009 [cit. 2016-04-21]. Dostupné z: DostupnézWWW<http://is.muni.cz/th/110118/ff_m_c1/>

[21] Veřejné výstupy *NetMonitor.cz* [online]. 2016, [cit. 2016-04-21]. Dostupné z: http://www.netmonitor.cz/verejne-vystupy

[22] Ivanov, A.: js-crawler *GitHub.com* [online]. 2016, [cit. 2016-04-23]. Dostupné z: https://github.com/antivanov/js-crawler

[23] Zimmer, S.: node-crawler *GitHub.com* [online]. 2015, [cit. 2016-04-23]. Dostupné z: https://github.com/sylvinus/node-crawler

[24] Giffard, C.: simplecrawler *GitHub.com* [online]. 2016, [cit. 2016-04-23]. Dostupné z: https://github.com/cgiffard/node-simplecrawler

[25] Rogers, M.: Spider *GitHub.com* [online]. 2014, [cit. 2016-04-23]. Dostupné z: `https://github.com/mikeal/spider`

[26] Readability *GitHub.com* [online]. 2016, [cit. 2016-04-26]. Dostupné z: `https://github.com/luin/readability`

[27] Uhrig, T.: Extracting meaningful content from raw HTML *tuhrig.de* [online]. 2013, [cit. 2016-04-26]. Dostupné z: `http://tuhrig.de/extracting-meaningful-content-from-raw-html/`

[28] Kim, T.: node-boilerpipe *GitHub.com* [online]. 2014, [cit. 2016-04-27]. Dostupné z: `https://github.com/xissy/node-boilerpipe`

[29] Kohlschütter, C.; Fankhauser, P.; Nejdl, W.: Boilerplate detection using shallow text features. In *Proceedings of the third ACM international conference on Web search and data mining*, ACM, 2010, s. 441–450.

[30] Straková, J.; Straka, M.; Hajič, J.: Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Baltimore, Maryland: Association for Computational Linguistics, June 2014, s. 13–18. Dostupné z: `http://www.aclweb.org/anthology/P/P14/P14-5003.pdf`

[31] Straka, M.; Straková, J.: Czech Models (MorfFlex CZ 160310 + PDT 3.0) for MorphoDiTa 160310. 2016, LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague. Dostupné z: `http://hdl.handle.net/11234/1-1674`

[32] Hana, J.; Zeman, D.; Hajič, J.; aj.: Manual for Morphological Annotation, Revision for the Prague Dependency Treebank 2.0. Technická Zpráva TR-2005-27, Praha, Czechia, 2005.

[33] Straka, M.; Straková, J.: Czech Models (CNEC) for NameTag. 2014, LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague. Dostupné z: `http://hdl.handle.net/11858/00-097C-0000-0023-7D42-8`

[34] Veselovská, K.; Bojar, O.: Czech SubLex 1.0. 2013, LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague. Dostupné z: `http://hdl.handle.net/11858/00-097C-0000-0022-FF60-B`

[35] Straková, J.; Straka, M.; Hajič, J.: A New State-of-The-Art Czech Named Entity Recognizer. In *Lecture Notes in Computer Science, Vol. 8082, Text, Speech and Dialogue*, Berlin / Heidelberg: Springer Verlag, 2013, ISBN 978-3-642-40584-6, ISSN 0302-9743, s. 68–75.

Příloha A

Seznam použitých zkratek

ACE Automatic Content Extraction

CSS Cascading Style Sheets

ČTK Česká tisková kancelář

DOM Document Object Model

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

PDF Portable Document Format

PL Parlamentní listy

NER Named Entity Recognition

NLP Natural Language Processing

ÚFAL Ústav formální a aplikované lingvistiky Univerzity Karlovy v Praze

W3C World Wide Web Consortium

XHTML Extensible HyperText Markup Language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `database`
      - `data` ..... složka obsahující balík stažených článků
      - `queries` .. složka obsahující skripty se složitějšími dotazy do DB
      - `script` ... složka obsahující skripty pro stažení a analýzu článků
      - `solr-5.5.0.zip` .. soubor obsahující databázi Apache Solr 5.5.0
    - `modules` ..... zdrojové kódy implementovaných modulů
      - `czech-news-ner` ..... modul pro analýzu entit
      - `czech-news-server-content-parser` ..... modul pro parsování zpravodajských článků
      - `czech-news-servers-crawler` ... modul pro crawlování českých zpravodajských serverů
      - `czech-sentiment-analysis` .... modul pro analýzu sentimentu
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS

Příloha C

Instalační příručka

Tato příloha obsahuje návod na stažení, extrakci a analýzu zpravodajských článků, přípravu databáze a nahrání získaných dat. Velmi stručně vysvětluje možnosti dotazování v databázi. Použité skripty byly odzkoušeny na OS Linux.

Na začátku je potřeba si stáhnout z přiloženého CD celou složku `src/impl` do svého počítače.

C.1 Stažení, extrakce a analýza zpravodajských článků

Pro použití již stažených a analyzovaných článků je třeba rozbalit do složky `src/impl/database/data/` soubor `processed.zip` umístěný ve stejné složce.

Pro případ stažení nových souborů či rozšíření těchto se nacházejí ve složce `src/impl/database/script/` soubory `get-data.js` a `process-data.js`, kde první z nich stahuje a parsuje články ze zpravodajských serverů, a druhý k nim přidává textovou analýzu. Pro použití spustíme terminál ve složce se soubory a zadáme příkazy, jejichž použití je následující.

```
1 node get-data.js server section dateTo
2 node process-data.js server
3
4 server: PL | Novinky | iDnes | Aktualne
5 section: domácí | zahraniční
6 dateTo: YYYY-mm-dd
```

Pokud se na začátku skriptu `process-data.js` vyskytla chyba, je možné, že se procesu nepodařilo zkompilovat MorphoDiTa server. To je možné provést ručně spuštěním příkazu `make server` ve složce `src/impl/moduls/czech-news-ner/lib/morphodita-master/src/rest_server/`.

C.2 Příprava databáze a nahrání souborů

Na přiloženém CD se ve složce `src/impl/database/` nachází archiv s databází `solr-5.5.0.zip`, ten je třeba v této složce rozbalit. Poté si otevřeme terminál ve složce s databází a spustíme následující příkaz, který databázi spustí.

```
1 bin/solr start -e cloud -noprompt
```

Jakmile tento příkaz doběhne, otevřeme si webové rozhraní databáze pomocí prohlížeče na adrese `http://localhost:8983/solr/`. V záložce *Core Admin* pak tlačítkem *Add Core* otevřeme dialog pro přidání nového jádra databáze. Vyplníme jeho název v poli *name* (např. *news*), a klikneme na modré tlačítko *Add Core*, čímž je jádro vytvořeno.

Vrátíme se do příkazové řádky a příkazem níže spustíme nahrávání souborů do vytvořeného jádra databáze. To potrvá několik minut, záleží na počtu článků. Přiložený balík článků (přes 17 tisíc souborů) se nahrává něco kolem osmi minut.

```
1 bin/post -c news ../data/processed/
```

C.3 Možnosti dotazování

Po nahrání článků je databáze již připravena pro dotazování. Ve webovém rozhraní se k psaní dotazů můžeme dostat tak, že v levém menu v select boxu s názvem *Core Selector* vybereme nově vytvořené jádro (zde pojmenováno *news*) a v nově objeveném menu vybereme položku *Query*. Způsob dotazování je možné najít v manuálu databáze[21].

Další dotazy se nacházejí na přiloženém CD ve složce `src/impl/queries/`. Jejich výsledek a použití je vysvětleno v sekci 6.3.

C.4 Vypnutí databáze

Databázi je možné vypnout příkazem níže (opět ze složky, kde je databáze umístěna).

```
1 bin/solr stop -all
```

[21] https://cwiki.apache.org/confluence/display/solr/Apache+Solr+Reference+Guide

Příloha D

Ukázka výstupu článku získaného použitím všech implementovaných modulů


```
{
  "header": "VIDEO: Fotograf ulovil desítky blesků,
     spojil je do úchvatného klipu",
  "date": "2016-02-15T20:44:00.510Z",
  "keywords": [
    "Zoo"
  ],
  "section": "zahraniční",
  "content": "Jedinečné snímky bouřky v~okolí Sydney se
      podařilo pořídit amatérskému fotografovi. Během
     pozorování divokých zvířat zvěčnil blesky, které
     bouřku doprovázely. Jednotlivé snímky pak spojil v
     ~ohromující podívanou.\nExpert na obnovitelné
     zdroje Ketan Joshi vyrazil minulý čtvrtek do
     zoologické zahrady v~Sydney, kde chtěl využít poč
     así a vyfotografovat co nejvíce snímků bouřky.
     Jednotlivé obrázky, na kterých zachytil blesky nad
      lesem a poli, pak spojil do jednoho časosběrného
     snímku. Podle Joshiho bylo na focení nejtěžší
     zvolit správný expoziční čas, uvedla agentura
     Reuters. Kdyby nechal závěrku otevřenou příliš
     dlouho, nezachytil by pohyb mraků. V~opačném pří
     padě by snímky byly tmavé.",
  "server": "iDnes.cz",
  "url": "http://zpravy.idnes.cz/australan-zvecnil-
     bourku-u-sydney-d5u-/zahranicni.aspx?c=
     A160215_213511_zahranicni_san",
  "entities": [
```

```
    {
      "name": "Sydney",
      "type": "gu",
      "dbpedia": "http://cs.dbpedia.org/resource/Sydney
         ",
      "occurrences": [
        {
          "sentenceIndex": 1,
          "type": "gu",
          "wordForm": "Sydney"
        },
        {
          "sentenceIndex": 4,
          "type": "gu",
          "wordForm": "Sydney"
        }
      ],
      "occurrencesCount": 2
    },
    {
      "name": "Ketan Joshi",
      "type": "P",
      "dbpedia": "",
      "occurrences": [
        {
          "sentenceIndex": 4,
          "type": "P",
          "wordForm": "Ketan Joshi"
        },
        {
          "sentenceIndex": 6,
          "type": "ps",
          "wordForm": "Joshiho"
        }
      ],
      "occurrencesCount": 2
    }
  ],
  "articleSentiment": 0,
  "sentencesSentiment": [
    {
      "sentenceIndex": 1,
      "sentiment": 1,
      "positiveWords": 1,
      "negativeWords": 0
    },
    {
      "sentenceIndex": 2,
      "sentiment": -1,
```

```
      "positiveWords": 0,
      "negativeWords": -1
    },
    {
      "sentenceIndex": 3,
      "sentiment": 1,
      "positiveWords": 1,
      "negativeWords": 0
    },
    {
      "sentenceIndex": 4,
      "sentiment": -1,
      "positiveWords": 0,
      "negativeWords": -1
    },
    {
      "sentenceIndex": 5,
      "sentiment": 0,
      "positiveWords": 0,
      "negativeWords": 0
    },
    {
      "sentenceIndex": 6,
      "sentiment": 0,
      "positiveWords": 1,
      "negativeWords": -1
    },
    {
      "sentenceIndex": 7,
      "sentiment": 0,
      "positiveWords": 1,
      "negativeWords": -1
    },
    {
      "sentenceIndex": 8,
      "sentiment": 0,
      "positiveWords": 0,
      "negativeWords": 0
    }
  ]
}
```