VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SÉMANTICKÁ ANALÝZA WEBOVÉHO OBSAHU

SEMANTIC ANALYSIS OF WEB CONTENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. LUKÁŠ HUBL**
AUTHOR

VEDOUCÍ PRÁCE **Ing. RADEK BURGET, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Hubl Lukáš, Bc.**
Program: Informační technologie Obor: Informační systémy
Název: **Sémantická analýza webového obsahu**
Semantic Analysis of Web Content
Kategorie: Web

Zadání:

1. Prostudujte technologie sémantického webu se zaměřením na popis dat pomocí ontologií a na projekty DBPedia a DBPedia Spotlight.
2. Seznamte se s existujícími metodami a nástroji pro předzpracování webových stránek za účelem extrakce dat.
3. Navrhněte způsob extrakce témat nebo konkrétních entit z webových stránek nebo jejich částí.
4. Po dohodě s vedoucím zvolte vhodné technologie a implementujte demonstrační aplikaci pro analýzu témat ve webových dokumentech.
5. Proveďte experimentální vyhodnocení navržených metod.
6. Zhodnoťte dosažené výsledky

Literatura:

- Lasila, I., Swick, R. R.: Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222
- Torrero, Ch., Caprini, C., Miorandi, D.: A Wikipedia-based approach to profiling activities on social media, arXiv:1804.02245

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Burget Radek, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 20. května 2020
Datum schválení: 16. října 2019

Abstrakt

Tato práce se zabývá problematikou sémantického webu, segmentace webových stránek a technologiemi, které se v těchto oblastech využívají. Dále se zaobírá modifikací jedné z metod pro segmentaci webových stránek, konkrétně metodou využívající DOM stromu, s využitím technologií z oblasti sémantického webu. Tedy navrhuje způsob segmentace webové stránky na základě sémantické analýzy obsahu jednotlivých prvků webové stránky. V rámci této práce byla také vytvořena aplikace, která demonstruje funkcionalitu navržené metody. S aplikací byly následně prováděny experimenty, jejichž zhodnocení je také součástí této práce.

Abstract

This work deals with the topics of semantic web, web page segmentation and technologies, which are used in this area. It also deals with a modification of one web page segmentation method, specifically DOM-based segmentation, using semantic web technologies. Thus, this work designs the way of web page segmentation based on semantic analysis of individual elements of the web pages content. An application that demonstrates the functionality of the designed segmentation method was also created within this work. With the implemented application, experiments were performed, whose results are also part of this work.

Klíčová slova

Sémantický web, RDF, RDFS, RDF/XML, ontologie, DBpedia, DBpedia-Spotlight, Segmentace webových stránek, Python, lxml, Pyspotlight

Keywords

Semantic web, RDF, RDFS, RDF/XML, ontology, DBpedia, DBpedia-Spotlight, Web page segmentation, Python, lxml, Pyspotlight

Citace

HUBL, Lukáš. *Sémantická analýza webového obsahu.* Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Burget, Ph.D.

Sémantická analýza webového obsahu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Radka Burgeta. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Lukáš Hubl
2. června 2020

Poděkování

Tímto bych velice rád poděkoval vedoucímu práce Radku Burgetovi, za poskytnuté konzultace a vedení při řešení této práce.

Obsah

Kapitola 1

Úvod

Oblast sémantického webu je pro většinu uživatelů internetu, mnohdy i těch, kteří se zabývají informačními technologiemi, zcela neznámá. Přitom už je tu s námi řadu let a projektů z této oblasti již exituje celá řada. V rámci této práce je tato problematika, respektive její klíčové části, stručně představena a nabízí čtenáři letmý pohled do této problematiky. Jsou zde uvedeny i projekty, které převádí teorii o sémantickém webu do praxe, avšak pouze ty, které jsou stěžejní pro další oblast této práce.

Další velice rozsáhlou oblastí, která je zde stručně představena, je segmentace webových stránek za účelem strojového získávání dat. Ve zkratce se jedná o vědní obor, zabývající se analyzováním webových stránek s cílem vyfiltrovat užitečný obsah a rozdělit stránku na oblasti obsahující prvky stránky, které spolu nějakým způsobem souvisí. V dnešní době metody vychází metody, jenž se zabývají segmentací, ze zavedených principů, ale snaží se je jistým způsobem modifikovat, za účelem dosáhnutí lepších výsledků segmentace a to navíc za kratší čas.

Tato práce není výjimkou a také se zabývá návrhem metody segmentace webových stránek, stavějící na segmentaci založené na DOM stromu a modifikuje ji tak, že k určení, zda spolu jednotlivé elementy nějak souvisí, využívá sémantickou analýzu obsahu elementu. Rozděluje tedy stránku na oblasti obsahující elementy, jejichž obsah spolu souvisí z hlediska tématu.

Za účelem ověření funkcionality navržené metody byla v rámci této práce vytvořena demonstrační aplikace, jenž implementuje navrženou metodu a provádí tak segmentaci webové stránky, respektive HTML dokumentu, na základě sémantické analýzy jejího DOM stromu.

Práce se také zabývá popisem implementace demonstrační aplikace a jejím testováním. S aplikací byly prováděny experimenty, pro ověření funkčnosti navržené metody, jejichž zhodnocení a výsledky jsou součástí závěrečné části této práce.

Kapitola 2

Sémantický web

V rámci této kapitoly si přiblížíme pojem *sémantický web*, myšlenku, proč se začalo pracovat na sémantickém webu, jaké technologie a nástroje stojí za sémantickým webem a jakým způsobem s ním můžeme pracovat.

2.1 Sémantický web obecně

První myšlenka, která byla veřejně vyslovena, se datuje k roku 2001 a pochází z úst Tima Bernse-Leeho, který tehdy prohlásil, že současný web je pouze změť webových stránek, která stále roste a najít v něm relevantní inforamce je čím dál více složitější. Proto přednesl jeho vizi o sémantickém webu: *"Mám vizi o webu, ve kterém jsou počítače schopny analizovat všechna data na webu - obsah, odkazy a transakce mezi lidmi a počítači. "Sémantický web", který toto umožňuje se musí nejprve objevit, ale jakmile to nastane budou se každodenní mechanismy obchodu, byrokracie a našeho každodenního života ovládat pomocí strojů, mluvících ke strojům. "Inteligentní agenti", na které lidé věky lákali, se konečně zhmotní."* [1]

Můžeme říci, že sémantický web je tedy nadstavba stávajícího webu, kde jsou však jednotlivé informace strukturovány podle standardizovaných pravidel, což umožňuje lepší strojové zpracování a tudíž lepší vyhledávání relevantních informací a jejich zpracovávání. Hlavním cílem sémantického webu je tedy umožnit počítačům, aby porozumněli významu jednotlivých dat.

Mezi základní stavební kameny sémantického webu patří zejména RDF neboli **R**esource **D**escription **F**ramework a ontologie. Pro sémantický web je klíčový standardizovaný popis všech zdrojů. Neodmyslitelnou součástí je zde i dotazovací jazyk, s jehož využitím lze nad RDF daty provádět dotazy.

2.2 RDF

O RDF lze prohlásit, že je jedním z nejdůležitějších nástrojů pro sémantický web. Jedná se o framework sloužící k popisu zdrojů vytvořený organizací W3C. Prostřednictvím RDF se popisují jednotlivé zdroje tak, aby byly čitelné pro lidi a zároveň pro stroje.

Je nutné podotknout, že RDF pouze určuje, jakým způsobem zapisovat popis zdroje, ale sám o sobě nemá definovanou syntaxi. Pro popis syntaxe lze použít různé jazyky, nicméně

nejvyužívanějším je značkovací jazyk XML[1]. Jazyk XML bude využit i v ukázkách v této práci.

2.2.1 RDF trojice

Popis zdrojů prostřednictvím RDF je založen na myšlence učinit prohlášení o zdrojích ve formě subjekt-predikát-objekt, též označováno jako RDF trojice, dále jen trojice. Subjektem je nazýván zdroj, který je popisován. Subjektem může být, s nadsázkou řečeno, téměř cokoliv, například celá webová stránka, či její část, kniha, osoba, místo atd. Je vždy popisován pomocí jednoznačného identifikátoru URI[2]. Predikát reprezentuje určitý atribut, vlastnost, aspekt, který popisuje daný zdroj, subjekt. Každý predikát má specifický význam. Určuje vztah mezi subjektem a objektem. Za pojmem objekt je potom skryta hodnota daného predikátu. Objekt může být již nějaký literál, nebo opět jednoznačný identifikátor URI. V případě, že se jedná o URI, může se objekt stát subjektem, zdrojem, v některé další RDF trojici. [6]

Shrneme-li si výše popsané poznatky o subjektu, predikátu a objektu, lze jednou větou říci, že subjekt má nějakou vlastnost (predikát), jejíž hodnota je objekt. Z faktu, že se objekt může stát subjektem v některé další trojici vyplývá, že se jednotlivé trojice mohou vzájemně provazovat, čímž se dostáváme k tomu, jak je možné trojice a jejich vazby reprezentovat. Jedna z možných reprezentací trojic je orientovaný graf, označován jako RDF graf. Subjekt je znázorněn jako elipsa do níž je vepsáno URI daného subjektu. Od subjektu vede orientovaná hrana, jež je nadepsána predikátem, směrem k objektu, který je reprezentován buď obdélníkem, v případě, že objektem je literál, nebo elipsou, jestliže se jedná o URI a tudíž o další subjekt.

Příklad

Nyní si uvedeme jednoduchý příklad (pro zjednodušení zápisu nebude uváděno celé URI). Vezmněme si větu: "*Shakespear je autorem Macbeth.*" Tuto větu si rozebereme na jednotlivé části podle schématu subjekt-predikát-objekt. Toto rozdělení je znázorněno v tabulce 2.1. Na obrázku 2.1 již vidíme grafickou reprezentaci dané trojice.

Subjekt	Macbeth
Predikát	Autor
Objekt	Shakespear

Tabulka 2.1: Rozdělení věty "*Shakespear je autorem Macbeth.*" na jednotlivé části trojice

Jestliže chceme uvést doplňující informace o objektu, musí být objektem zdroj, tedy musí mít jednoznačný identifikátor URI. Jako příklad si uvedeme následující dvě věty: "*Shakespear je autorem Macbeth. Anne Hathaway je manželka Shakespeara.*" Pro rozdělení těchto vět upravíme tabulku 2.1 následovně (grafická reprezentace těchto vět je znázorněna na obrázku 2.2):

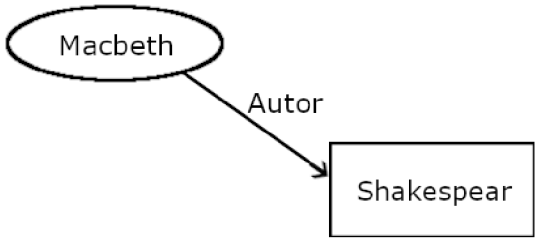
2.2.2 RDF/XML

Již výše bylo zmíněno, že RDF slouží k popisu metadat (dat o datech) zdrojů, ale nemá definovanou přesnou syntaxi a proto se pro popis syntaxe používá například jazyk XML.

[1] XML https://www.w3schools.com/xml/default.asp
[2] URI https://www.w3.org/Addressing/URL/uri-spec.html

	1. věta	2. věta
Subjekt	Macbeth	Shakespear
Predikát	Autor	Manželka
Objekt	Shakespear	Anne Hathaway

Tabulka 2.2: Rozdělení vět "*Shakespear je autorem Macbeth. Anne Hathaway je manželka Shakespeara.*" na jednotlivé části trojice



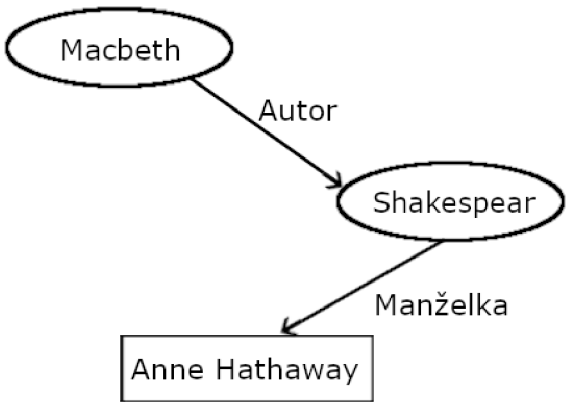
Obrázek 2.1: RDF graf trojice *Macbeth-Autor-Shakespear*

Organizace W3C[3], která stojí za RDF, přímo definovala syntaxi s využitím XML, konkrétně RDF/XML. Tato syntaxe je mnohdy chybně označována pouze jako RDF. S využitím syntaxe RDF/XML jsme schopni vyjádřit, serializovat, RDF graf jako XML dokument. RDF/XML syntaxe se vyskytuje ve dvou formách, *serializační* a *zkrácená*. Serializační syntaxe poskytuje všechny možnosti datového modelu, kdežto zkrácená pouze podmnožinu. Nicméně je však zkrácená verze kompaktnější.[6]

Zkrácená verze existuje ve třech formách. První verze je vhodná pro případ, kdy se predikáty v elementu `Description` neopakují a zároveň je hodnota predikátu, objekt, literál. Druhá verze zjednodušuje zápis, jestliže jsou v elementu `Description` vnořeny další elementy. Poslední forma zjednodušuje zápis, jestliže element `Description` obsahuje vlastnost `type`. [6]

Formální gramatika RDF/XML obsahuje celou řadu pravidel, podle kterých se provádí samotná serializace, nicméně tato pravidla si zde nebudeme uvádět. V případě zájmu

[3]W3C https://www.w3.org/



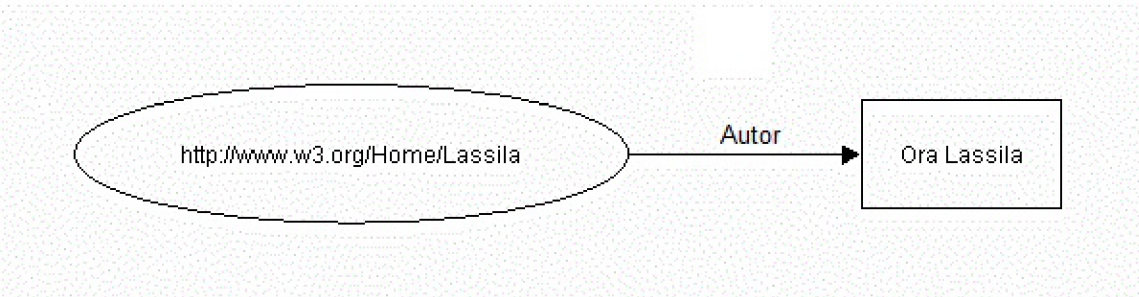
Obrázek 2.2: RDF graf trojice *Macbeth-Autor-Shakespear*, *Shakespear-Manželka-Anne Hathaway*

čtenáře jsou tato pravidla dostupná na https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/#grammar.

Důležitou roli zde, stejně jako v samostatném XML, hrají jmenné prostory. U RDF/XML slouží k přiřazení predikátu, pomocí kterého popisujeme příslušný zdroj, ke schématu, kde je daný predikát definován. Co přesně je zmíněné schéma bude vysvětleno v sekci 2.4.1.

Příklad

Vezměme si větu: "*Ora Lassila je autor zdroje http://www.w3.org/Home/Lassila.*" Grafická reprezentace této věty ve formátu RDF trojice je na orázku 2.3. Vidíme, že subjektem je zdroj *http://www.w3.org/Home/Lassila*, predikátem je *Autor* a objektem pak literál *Ora Lassila.* [6]



Obrázek 2.3: RDF graf věty "*Ora Lassila je autor zdroje http://www.w3.org/Home/Lassila.*" Převzato z [6].

Nyní si uvedeme, jak bude vypadat zápis v *serializační* a *zkrácené* formě (první forma zkrácení). Pro zkrácení zápisu nebudou v kódu deklarovány jmenné prostory, ale pro úplnost budou uvedeny nyní (platí pro serializační i pro zkrácenou formu):

xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:s="http://description.org/schema/"

Serializační forma:

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Autor>Ora Lassila</s:Autor>
  </rdf:Description>
</rdf:RDF>
```

Zkrácená forma:

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila"
        s:Autor="Ora Lassila"/>
</rdf:RDF>
```

[6]

2.2.3 Kontejnery

Kontejner je prostředek v RDF, pomocí kterého je možné odkazovat kolekci zdrojů. Jako příklad situace, kdy je toto nezbytné lze uvést případ, kdy se na jednom díle podílelo více autorů.

V RDF jsou definovány tři druhy kontejnerů:

- Bag
- Sequence
- Alternative

Bag

Jedná se v podstatě o multimnožinu. Tedy Bag se využívá v případě, kdy vlastnost má několik hodnot a nezáleží nám na jejich pořadí. Bag stejně jako multimnožina povoluje duplicitní hodnoty. [6] Uveďme si příklad použití kontejneru Bag i s jeho zápisem v RDF/XML:

Uvažme větu *Studenti v kurzu Matematika jsou Jan, Karel, Petr, Jana, Michaela.* Zápis v RDF/XML také věty by vypadal následovně:

```
<rdf:RDF>
  <rdf:Description about="http://univerzita.eu/kurzy/Matematika">
    <s:studenti>
      <rdf:Bag>
        <rdf:li resource="http://univerzita.eu/studenti/Jan"/>
        <rdf:li resource="http://univerzita.eu/studenti/Karel"/>
        <rdf:li resource="http://univerzita.eu/studenti/Petr"/>
        <rdf:li resource="http://univerzita.eu/studenti/Jana"/>
        <rdf:li resource="http://univerzita.eu/studenti/Michaela"/>
      </rdf:Bag>
    </s:studenti>
  </rdf:Description>
</rdf:RDF>
```

Sequence

Sequence představuje uspořádaný seznam. Sequence může obsahovat několik hodnot, u kterých je důležité jejich pořadí, například kvůli zachování abecedního pořadí. Stejně jako u Bag jsou povoleny duplicitní hodnoty. [6] Příklad si zde nebudeme uvádět, jelikož by se od předchozího příkladu na kontejner Bag moc nelišil. Jediný rozdíl by byl v syntaxi, kdy místo `<rdf:Bag>` by se použilo `<rdf:Seq>`. Výsledný seznam by poté byl stejný s tím rozdílem, že nyní by byly hodnoty (jména studentů) v abecedním pořadí.

Alternative

Alternative se využívá v případě, kdy chceme uvést k některé hodnotě dané vlastnosti další, alternativní hodnoty. [6] Například chceme-li název díla uvést i v dalších světových jazycích. V tom případě by ”hlavní hodnota” byla například název díla v anglickém jazyce a jako alternativy by byly přeložené názvy v italštině, češtině atd. Jako příklad si uvedeme větu:

"*Název jednoho z nějznámějších děl Satvatora Daliho je The Persistence of memory, nebo také La persistencia de la memoria.*" Zápis v RDF/XML by mohl vypadat následovně:

```
<rdf:RDF>
  <rdf:Description about="http://www.foo.cz/obrazy/SD1931">
    <t:Title>
      <rdf:Alt>
        <rdf:li xml:lang="en">The Persistence of memory</rdf:li>
        <rdf:li xml:lang="es">La persistencia de la memoria</rdf:li>
      </rdf:Alt>
    </t:Title>
  </rdf:Description>
</rdf:RDF>
```

2.3 Ontologie

Následující podkapitola je převzata z [11].

Pojem ontologie se nevztahuje pouze k informačním technologiím a sémantickému webu, ale vyskytuje se také jako filosofická disciplína, zabývající se bytím jako takovým. Je ale jasné, že filosofií se zde zabývat nebudeme. Nicméně některé rysy jsou společné. Jak ve filosofii, tak v informačních technologiích jde o představení entit, myšlenek a událostí se všemi vzájemně závislými vlastnostmi a vztahy, podle jistého systému kategorií. Dále již pod pojmem ontologie budeme uvažovat pouze onotologie z oblasti informačních technologií.

Ontologie formálně definuje pojmy modelovaného světa, jejich význam a jejich vzájemné vztahy. Tím, že se využívá formální popis, s jehož využitím se entitě dodá její význam (každé URI v sémantickém webu má nyní svůj význam) umožňujeme, aby danému významu bylo porozuměno napříč různými aplikacemi. Obrovskou výhodou onotologie je její znovupoužitelnost a sdílení. Není tedy pro každý případ nutné vytvářet novou ontologii, ale lze využít již existující onotologii, případně ji upravit, nebo na jejím základě vybudovat novou.

Ontologií je několik typů. Zde si uvedeme pouze pár příkladů.

Doménové ontologie

Doménová ontologie definuje termíny a jejich význam z konkrétní specifické domény, oblasti. Jako příklad můžeme uvést slovo *disk*. V ontologii o doméně *sport* by význam slova *disk* byl *sportovní náčiní*, zatímco v ontologii popisující počítačový hardware by byl význam slova *pevný disk*.

Terminologické ontologie

Terminologické či lexikální ontologie mají charakteristický rys, jímž je role *termínů*, které již dále nejsou definovány, ale jsou definovány jejich vzájemné vztahy jako je synonymie, meronymie či taxonomie. Nejznámější ontologií z této oblasti je *WordNet*.

Informační ontologie

Rozvíjejí databázová schémata. Jedná se tedy o jakousi nadstavbu konceptuálních databázových schémat, zabezpečující například vyšší kontrolu integrity či konceptuální abstrakci, která je potřebná pro pojmové dotazování

Znalostní ontologie

Koncepty a jejich vazby jsou zde formálně definovány s využitím logických formulí. Vznikly v návaznosti na výzkum v oblasti umělé inteligence.

2.3.1 Struktura ontologií

Ontologie spolu sdílejí mnoho podobností z pohledu na jejich strukturu a to bez ohledu na jazyk. Nyní si uvedeme podstatné prvky ze struktury ontologií.

Třídy, koncepty, kategorie

Oproti třídám jak je známe z objektově-orientovaných modelů neobsahují tyto třídy žádné procedurální metody. Třída z oblasti ontologií odpovídá relaci na dané doméně objektů. Existují dva typy tříd, *definované* a *primitivní*. Na množině tříd bývají definované jejich vzájemné vztahy, ve smyslu uspořádání do jisté hierarchie (taxonomie). Většina ontologických jazyků hojně podporuje vícenásobnou dědičnost tříd. Pojmy *koncept* a *kategorie* odpovídají pojmu *třída*, záleží na formalismu, ve kterém se daný pojem vyskytuje.

Individua, instance

Individuum reprezentuje konkrétní objekt reálného světa. Není nutné, aby individuum bylo instancí některé třídy, ale může být do ontologie vloženo bez vazby na třídu. S ohledem na fakt, že individua reprezentují konrétní data, je často některé jazyky vůbec jako součást ontologie nepodporují, jelikož ontologie slouží k popisu konceptů a nikoliv konkrétních objektů.

Relace, vlastnosti, atributy, sloty

Pojem *vlastnost (slot)* zde, na rozdíl oproti objektově-orientovanému modelování, vyjadřuje relaci, většinou binární. Vlastnosti v ontologii nemají pevnou vazbu s třídou a vazba na definiční obor (resp. obor hodnot) je zprostředkována skrze omezení. Relace mohou mít definovanou dědičnost, kdy nadřazená relace obsahuje všechny prvky podřazené relace. Jako speciální relace, zde bývá označena funkce, kde hodnota n-tého argumentu je jednoznačně určená předchozími n-1 argumenty.

Meta-sloty, omezení na sloty, facety

V ontologiích lze přiřazovat vlastnosti slotům. Hovoříme tedy o vlastnostech vlastností, což bývá označováno jako *meta-sloty*. Nejčastěji se jedná o vztah hierarchický, tedy nadřazená a podřazená vlastnost. Kromě obecných matematických vlastností relací mají sloty také *definiční obor* a *obor hodnot*. Tyto vlastnosti označujeme jako *globální omezení*. Někdy je však nutné vymezit takzvané *lokální omezení*, též označované jako *facet*, kterým může být například kardinalita.

Primitivní hodnoty, datové typy

Argumenty relací nemusí být nutně objekty, ale mohou to být i tzv. *primitivní hodnoty*. V tom případě už ale nehovoříme o *slotech*, jaké byly popsány výše, ale o *dato-typových* (datatype) slotech. Datový typ oboru hodnot dato-typového slotu může být jeden ze základních datových typů, jako je string, float, integer, ale i interval či výčet. Často se používá terminologie *dato-typová třída*, což jsou výše uvedené základní datové typy a *dato-typová instance*, což je přímo hodnota, jejíž datový typ je *dato-typová třída*. Například znak "a" je instance dato-typové třídy *string*. Dato-typové sloty se obvykle deklarují jako funkčí, aby nabývali pouze jedné hodnoty a odpovídali tak jednoznačným vlastnostem objektů. Jesliže by se nedeklarovali jako funkční, mohli by nabývat více hodnot současně.

Axiomy, pravidla

Kromě výrazů, pomocí nichž vymezujeme příslušnost ke třídám a relacím, lze v onotlogiích využít také logických formulí, skrze které můžeme vyjádřit například subsumpci, ekvivalenci disjunktnost tříd, či rozklad třídy na podtřídy. Tyto logické formule nejčastěji označujeme jako *axiomy*. *Axiomy* bývají obvykle součástí definice tříd a relací.

2.3.2 Existující ontologie

Jak již bylo zmíněno, tak velmi často dochází ke znovupoužití již existující onotlogie resp. existujících onotlogií, jejichž vlastnosti a koncepty je možné kombinovat.

Mezi ty nejznámější patří zcela jistě ontologie **Friend-of-a-friend (FOAF)**. Jedná se o ontologii sloužící pro popis osob a jejich vzájemných vztahů. Jako ukázkovou třídu a vlastnost můžeme uvést například třídu `foaf:Person` a vlastnost `foaf:name`.

Jedna z dalších velice známých onotlogií je **Dublin core**. Ta nachází využití zejména v knihovnictví, jelikož popisuje metadata dokumentů a definuje tak jejich vlastnosti.

Jako další příklady můžeme uvést ontologie s poměrně specifickým využitím, jako je **GAO**, otologii určenou pro automobilový průmysl, **Plant ontology**, onotlogii pro strukturu a růstovou fázi rostlin, **Foundational Model of Anatomy (FMA)**, ontologii pro lidskou anatomii. Toto je pouze malý výčet existujících ontologií, ale existuje samozřejmě obrovská spousta dalších ontologií, kdy využití některých z nich se může mnohdy zdát až bizarní.

2.4 Ontologické jazyky

Pojmem *ontologické jazyky* označujeme jazyky, pomocí kterých lze popisovat a tudíž vytvářet ontologie. Tyto jazyky bývají nejčastěji formální, ale mohou být i semiformální (jde o částečně strukturovaný volný text), či úplně neformální. Ontologické jazyky lze dělit do několika kategorií, nás však bude zajímat pouze kategorie *značkovacích onotlogických jazyků* a z této kategorie se zmíníme pouze o dvou jazycích, které souvisí s touto prací a to konkrétně RDFS a OWL.

2.4.1 RDFS

RDF Schema, zkráceně označováno jako RDFS, ale i jako RDF(S), RDF-S či RDF/S, je ontologickým jazykem, vytvořeným pod záštitou konsorcia W3C, který vznikl jako sémantické rozšíření RDF. Rozšiřuje tedy RDF o možnost vytvářet vlastní ontologie a RDF slovníky.

RDFS umožňuje definici tříd, binárních relací, hierarchie nad třídami a relacemi. Definice RDFS využívá opět principu RDF trojice. Syntaxe může být, stejně jako v RDF, v různých jazycích. I nyní se však zaměříme na XML. Jmenný prostor mývá obvykle prefix **rdfs:**. Tento jmenný prostor je identifikován pomocí IRI[4] http://www.w3.org/2000/01/rdf-schema#. Níže bude využíván také prefix **rdf:**, který odkazuje na RDF jmenný prostor, identifikovaný pomocí IRI http://www.w3.org/1999/02/22-rdf-syntax-ns#. [2]

Třídy definované v RDFS

Tříd definovaných v RDFS existuje hned několik. Jednotlivé třídy jsou přiřazovány ke zdroji pomocí **rdf:type**. Budeme zde hovořit i o tzv. *odvozených třídách (podtřídách)*, které jsou odvozeny z některé již existující třídy.

Uvedeme si třídy definované v RDFS (převzato z [2]):

- **rdfs:Class** - Třída zdrojů, které jsou třídami RDF. **rdfs:Class** je instancí **rdfs:Class**
- **rdfs:Resource** - Všechny věci popisované pomocí RDF se nazývají zdroje a jsou instancí **rdfs:Class**
- **rdfs:Literal** - Třída hodnot, které jsou literály jako je string či integer. Je to instace **rdfs:Class** a podtřída **rdfs:Resource**
- **rdfs:Datatype** - Třída datových typů. Instance i podtřída **rdfs:Class**. Každá instance **rdfs:Datatype** je podtřídou **rdfs:Literal**
- **rdf:langString** - Třída tzv. *language-tagged* řetězcových hodnot. Instance **rdfs:Datatype** a podtřída **rdfs:Literal**
- **rdfs:HTML** - Třída hodnot, které jsou HTML literály. Instance **rdfs:Datatype** a podtřída **rdfs:Literal**
- **rdfs:XMLLiteral** - Třída hodnot, které jsou XML literály. Instance **rdfs:Datatype** a podtřída **rdfs:Literal**
- **rdfs:Property** - Třída RDF vlastností. Instance **rdfs:Class**.

Vlastnosti v RDFS

Koncept RDF vlastnosti je popisován jako relace mezi subjektem a objektem. Všechny vlastnosti jsou instancí třídy **rdfs:Property**. V RDFS jsou definovány následující vlastnosti (u každé vlastnosti bude uveden příklad RDF trojice s jejím vysvětlením; převzato z [2]):

- **rdfs:range** - Určuje typ, obor hodnot, objektů. Trojice "**P rdfs:range T**" značí, že zdroje označené objekty trojic, kde predikátem je P (P je instance **rdf:Property**) jsou instancemi třídy T (T je instance třídy **rdfs:Class**).
- **rdfs:domain** - Určuje typ, definiční obor, subjektů. Trojice "**P rdfs:domain T**" značí, že zdroje označené subjekty trojic, kde predikátem je P (P je instance **rdf:Property**) jsou instancemi třídy T (T je instance třídy **rdfs:Class**).
- **rdfs:type** - Značí, že zdroj je instancí dané třídy. Trojice "**P rdfs:type T**" značí, že P je instance T (T je instance třídy **rdfs:Class**).

[4] IRI https://www.w3.org/International/O-URL-and-ident.html

- **`rdfs:subClassOf`** - Všechny instance jedné třídy jsou instancemi třídy druhé. Trojice "**`T1 rdfs:subClassOf T2`**" značí, že T1 je podtřídou T2 (T1 i T2 jsou instancemi třídy **`rdfs:Class`**). Vlastnost **`rdfs:subClassOf`** je tranzitivní. Jestliže řekneme: T1 *je podtřídou* T2, myslíme tím, že všechny instance T1 budou taky instancemi T2.

- **`rdfs:subPropertyOf`** - Zjednodušeně lze říci, že jedna vlastnost je "podvlastností" jiné vlastnosti. Tedy pokud jsou zdroje spojeny s jistou vlastností V1, která je "podvlastností" vlastnosti V2, pak všechny zdroje jsou také spojeny s vlastností V2. Trojice "**`P1 rdfs:subClassOf P2`**" značí, že P1 je "podvlastností" P2 (P1 i P2 jsou instancemi třídy **`rdf:Property`**).

- **`rdfs:label`** - Používá se k přidání popisku k názvu zdroje, aby byl pro člověka lépe pochopitelný. Trojice "**`R rdfs:label L`**" značí, že L je popisek názvu zdroje R. Definičním oborem **`rdfs:label`** je **`rdf:Resource`** a oborem hodnot **`rdfs:label`** je **`rdfs:Literal`**.

- **`rdfs:comment`** - Obdoba **`rdfs:label`** s tím rozdílem, že popisek se nevztahuje pouze k názvu zdroje, ale ke zdroji jako takovému. Trojice "**`R rdfs:comment C`**" značí, že C je popisek zdroje R. Definičním oborem **`rdfs:comment`** je **`rdf:Resource`** a oborem hodnot **`rdfs:comment`** je **`rdfs:Literal`**.

2.4.2 OWL

Web **O**ntology **L**anguage, zkráceně označován **OWL**, je jedním z dalších ontologických jazyků, jenž je využíván v oblasti sémantického webu. Oproti RDFS je rozšířen o pokročilejší vlastnosti, které lze přiřazovat zdrojům. OWL vznikl v roce 2004 a stejně jako RDFS ho vytvořilo konsorcium W3C. V roce 2009 přišlo W3C s druhou verzí jazyku OWL označovanou jako OWL2. Jmenný prostor OWL využívá prefix **`owl:`** a je idnetifikován pomocí IRI http://www.w3.org/2002/07/owl#. [10]

První verze OWL existovala v několika verzích (převzato z [10]):

- OWL Lite - OWL Lite vznikl zejména pro uživatele, kteří potřebují především hierarchii klasifikace a pouze jednoduchá omezení.

- OWL DL - OWL DL obsahuje veškeré konstrukční prvky z OWL, ale lze je použít pouze za jistých omezení

- OWL Full - OWL Full je maximálně kompatibilní s RDFS

Vztahy mezi jednotlivými verzemi byly následující (převzato z [10]):

- Každá legitimní OWL Lite ontologie je legitimní OWL DL ontologií

- Každá legitimní OWL DL ontologie je legitimní OWL Full ontologií

- Každý validní závěr v OWL Lite je validním závěrem v OWL DL

- Každý validní závěr v OWL DL je validním závěrem v OWL Full

V OWL lze definovat třídy několika způsoby (příklady zde nebudou uvedeny, v případě zájmu čtenáře jsou příklady dostupné na stránce https://www.w3.org/TR/owl-guide/):

- Identifikátorem

- Výčtem prvků
- Omezením vlastností
- Průnikem
- Sjednocením
- Doplňkem
- Ekvivalencí mezi třídami a vlastnostmi

OWL 2 přímo vychází z první verze OWL. OWL 2 přidává k OWL některé nové prvky, které budou uvedeny níže, nicméně co je důležité zmínit je fakt, že OWL 2 má zpětnou kompatibilitu s OWL. To znamená, že všechny platné onotlogie vytvořené prostřednictvím OWL budou platné i v OWL 2. Nyní budou uveden některé prvky, které byly přidány do OWL 2 (převzato z [5]).

- "Syntaktický cukr" - Usnadněný zápis některých běžných vzorů. Zatímco v OWL bychom museli nějaké prohlášení vytvořit kombinací několika axiomů, OWL 2 umožňuje zkrácený zápis, kdy axiomy spojíme do jediného axiomu.
- Nové konstruktory - Pro zvýšení vyjadřovací schopnosti zápisu
- Rozšířená podpora datových typů - Oproti OWL byly přidány některé nové datové typy, například `decimal, double, positiveInteger`, binární data, identifikátory IRI atd.
- Rozšířené anotace - OWL podporovala přidávání popisku k názvu zdroje nebo komentáře pro celý zdroj, ale nebylo možné přidat anotaci například k axiomu. OWL 2 umožňuje anotovat ontologie, entity, axiomy a třeba také anotovat samotné anotace.
- Další upravy týkající se například deklarací, IRI, anonimních individuí atd.

2.5 SPARQL

SPARQL, neboli **S**imple **P**rotocol **a**nd **R**DF **Q**uery **L**anguage, vznikl pod záštitou konsorcia W3C jako dotazovací jazyk nad daty RDF. Jelikož, se RDF skládá z trojic, tak i dotaz ve SPARQL je složen z množiny trojic. [8]

Podobnost s dotazovacími jazyky nad relačními databázemi, jako je například SQL, můžeme vidět zejména u některých klíčových slov, jako SELECT, WHERE či FROM. SPARQL pro dotazování nad RDF definuje čtyři druhy dotazů:

- SELECT
- CONSTRUCT
- ASK
- DESCRIBE

SELECT

Dotaz SELECT vrací všechny proměnné a jejich mapování, které vyhovují dotazu. Výsledek dotazu SELECT může být serializován do XML či RDF grafu. [8]

CONSTRUCT

Dotaz CONSTRUCT vrátí RDF graf specifikovaný šablonou. Ten je vytvořen tak, že jsou postupně brány jednotlivé výsledky dotazu a ty se propojují do jediného RDF grafu. [8]

ASK

ASK vrací hodnotu *True* resp. *False* podle toho, zda zadaný dotaz *má* resp. *nemá* řešení. [8]

DESCRIBE

Výsledek dotazu DESCRIBE je ve formátu RDF grafu, ale ten neobsahuje data vyhovující dotazu, jako v případě CONSTRUCT, ale obsahuje jejich popis. [8]

2.6 DBpedia

DBpedia vznikla jako projekt s účelem zpřístupnit data, dostupná na světově nejpoužívanější internetové encyklopedii Wikipedie ve formátu, ve kterém je možné tato data zpracovávat v oblasti sémantického webu, tj. RDF záznamy.

Wikipedie jako taková, existuje ve stovkách jazykových verzí. Některé články z různých jazykových verzí jsou vzájemně provázány, jelikož byly vytvořeny s využitím šablony, což usnadňuje zpracování dat z Wikipedie za účelem strojového "dolování" informací.

DBpedia získáva data z více než 100 jazykových verzí Wikipedie, které popisují přibližně 38.3 milionů věcí. V této práci nás bude zajímat pouze anglická verze DBpedie. Ta popisuje 4.58 milionu věcí zahrnující osoby, místa, umělecká díla (hudební alba. filmy, hry atd.), organizace, nemoci atd. [7]

2.6.1 Architektura DBpedie a princip extrakce informací

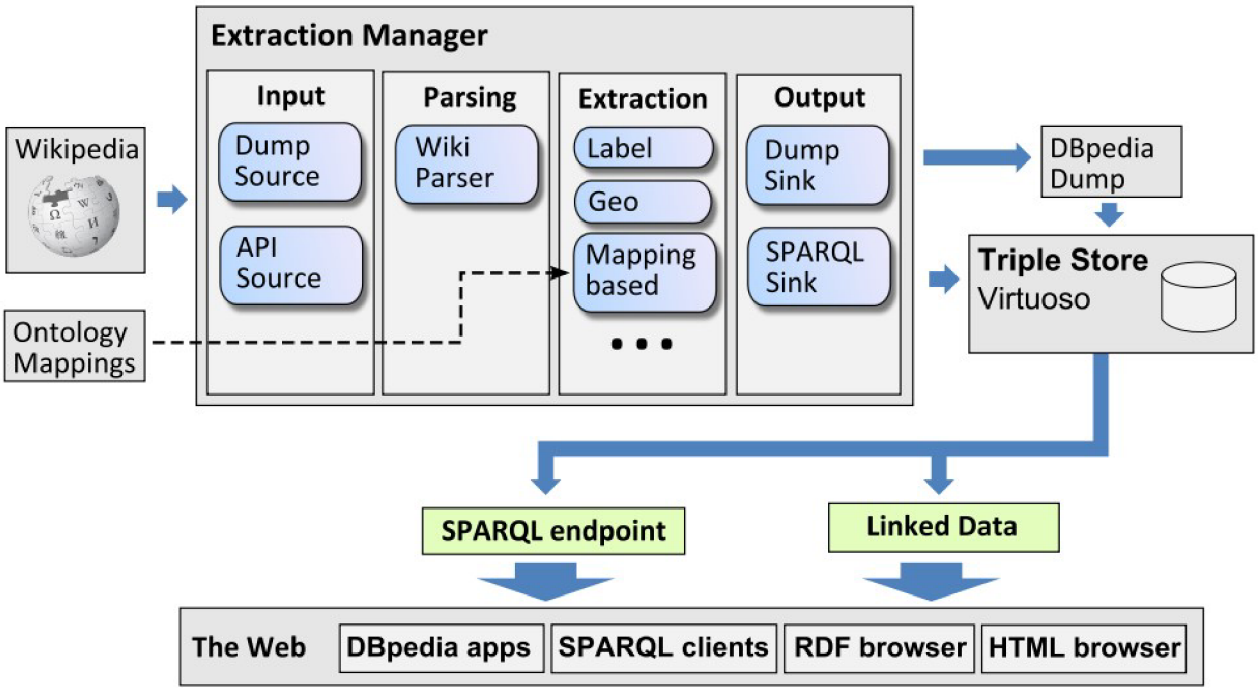
DBpedia získává z Wikipedie zejména strukturované informace jako jsou položky z informační tabulky (infoboxu), která bývá vytvořena na základě šablony, informace o kategorii, obrázky, odkazy na další webové stránky, které souvisí s tématem, geografické souřadnice nebo například odkaz na stránku Wikipedie, zabývající se stejným tématem, ale v jiné jazykové verzi. [7] Na obrázku 2.4 je znázorněna architektura DBpedie a postup extrakce stránek z Wikipedie, tak jak ji DBpedia provádí.

Proces extrakce informací je rozdělen do několika fází.

1. vstup
2. syntaktická analýza
3. extrakce
4. výstup

Vstup

Jednotlivé stránky Wikipedie jsou načteny z externího zdroje, případně mohou být načteny s využitím API, které poskytuje přímo Wikipedie, MediaWiki API.



Obrázek 2.4: Architektura DBpedie, s detailem na extrakci. Převzato z [7]

Syntaktická analýza

Nástroj provádějící synktatickou analýzu přetransformuje zdrojový kód stránky Wikipedie do abstraktního syntaktického stromu

Extrakce

DBpedia využívá různé extraktory podle daného účelu.

- **Hrubá extrakce infoboxů** - Jedná se o přímé mapování infoboxů na RDF. Kvalita extrakce je však nižší. Tento způsob extrakce je užitečný, jestliže daný infobox doposud nebyl mapován a tedy není možné využít extrakci na základě mapování.
- **Extrakce infoboxů na základě mapování** - Tato extrakce využívá mapování, které bylo vytvořeno ručně, mezi pojmy v infoboxech s pojmy v ontologii DBpedie. Každá vlastnost z infoboxu také získá na základě mapování datový typ.
- **Extrakce vlastnosti** - Extrakce vlastnosti využívá více extraktorů, které se specializují na získání jediné vlastnosti, prvku, ze článku, například nadpis či geografické souřadnice.
- **Extrakce za účelem získání statistik** - Tyto extraktory agregují data ze všech stránek Wikipedie, za účelem získání statistických údajů, jako je například počet slov.

Výstup

Získaná RDF data jsou uložena, například ve formátu N-Triples[5].

2.7 DBpedia-Spotlight

DBpedia-Spotlight, dále v textu pouze Spotlight, vznikl jako nástroj, který přímo pracuje s DBpedií a slouží k anotaci textu. Jeho vstupem je nestrukturovaný text, který Spotlight anotuje a anotované části textu propojí se zdroji z DBpedie. Spotlight v textu rozpozná název konceptu nebo entity a tento název propojí s jednoznačným idnetifikátorem zdroje. Využití anotace textu a následné přiřazení konkrétním zdrojům lze využít například pro zlepšení webových vyhledávačů.

Spotlight poskytuje uživatelům několik možností, jak s ním pracovat. Jednou z nich je webová demo aplikace, kdy uživatel pouze vloží text, vybere jazyk, v kterém je text napsán a může si také vybrat tématické oblasti, které ho zajímají (například pouze osoby a místa). Další prvek, který může uživatel ovlivnit je míra takzvané "důvěrnosti", její hodnota se pohybuje od 0 do 1 a čím více se blíží hodnotě 1, tím méně částí textu bude anotovaných, ale zato s vysokou přesností přiřazení ke zdroji v DBpedii. Výsledkem této aplikace je text, který uživatel zadal jako vstup, ve kterém jsou zvýrazněny slova či fráze, které Spotlight propojil s DBpedií. Anotované části jsou zároveň hypertextovým odkazem směřujícím na stránku příslušného zdroje v DBpedii, který byl dané části textu přiřazen.

Další možnosti jsou spíše pro vývojáře, kteří chtějí Spotlight zapojit do svého projektu. Pro tyto možnosti musí mít uživatel připravené prostředí pro překlad a spuštění Spotlightu lokálně, ať už jde o nainstalovanou Javu, Scalu a další nástroje. Spotlight však na svém GitHubu poskytuje návody, jakým způsobem docílit úspěšného překladu a spuštění Spotlight. Uživatel má možnost zprovoznění Spotlight prostřednictvím Mavenu[6], Dockeru[7] či stažením JAR balíku[8].

Spotlight poskytuje ještě jednu možnost, jak ho využívat ve svých projektech a to webovou službu, kdy prostřednictví WADL[9] lze s touto službou komunikovat. Uživateli stačí, aby měl klienta, prostřednictvím kterého bude s webovou službou komunikovat. Webová služba podporuje tři typy příkazů (převzato z [4]):

- spot - Provádí tzv *spotting*, kdy ve vstupním textu vybere vhodné kandidáty k anotaci
- annotate - Provede spotting a poté jednotlivé kandidáty propojí se zdroji v DBpedii. V případě více kandidátů na propojení vybere "nejvhodnějšího".
- candidate - Obdoba annotate, ale s rozdílem, že vrací list kandidátů na propojení se zdrojem, včetně jejich ohodnocení.

Všechny tři příkazy podporují více výstupních formátů jako je XML, JSON, HTML, NIF či RDFa. Pomocí paramteru při zasílání příkazu lze formát výstupu vynutit. Dalším velmi užitečným prvkem je zahrnutí SPARQL dotazu přímo do příkazu zasílaného webové službě, ta poté filtruje výsledky podle toho, zda splňují zadaný dotaz. Je tedy možné omezit anotace pouze na jisté téma, či anotovat výsledky, které zapadají do daného tématu, ale mají navíc spojení s konkrétním místem. [4]

[5] N-Triples https://www.w3.org/TR/n-triples/
[6] Maven https://maven.apache.org/
[7] Docker https://www.docker.com/
[8] JAR https://en.wikipedia.org/wiki/JAR_(file_format)
[9] WADL https://en.wikipedia.org/wiki/Web_Application_Description_Language

Kapitola 3

Segmentace webových stránek

Cílem této kapitoly je představení jedné z disciplín z oboru informačních technologií využívanou k předzpracování webových stránek za účelem strojového získávání informací, segmentace webových stránek. Jelikož se následující text bude věnovat pouze segmentaci webových stránek, bude dále uváděn pouze pojem segmentace, ale rozumnějme za ním právě segmentaci webových stránek. Dále si přiblížíme metody a principy, které se k segmentaci využívají.

3.1 Co je segmentace webových stránek?

Dříve než si představíme způsoby, jakým lze segmentaci provést, je vhodné si nejprve říci co vůbec stojí za pojmem segmentace.

S ohledem na fakt, že internet, webové stránky a jejich obsah je v dnešní době jedním z největších a nejdůležitějších zdrojů informací, je nakládáno značné úsilí v oblasti získávání relevantních informací právě z těchto zdrojů. Jelikož však neexistuje žádný jednotný standart či šablona, podle které by se webové stránky tvořili, je velice obtížné z nich požadované informace získat. Velký problém nastává právě tehdy, kdy jediná webová stránka vypadá tak, že se na ní nachází několik oblastí, z nichž každá oblast může poskytovat informace o zcela jiných odvětvích. Z lidského pohledu se může jevit zcela jednoznačně, že jednotlivé oblasti spolu nijak nesouvisí, jelikož dané oblasti mohou být nějak vizuálně odděleny, případně je pro člověka zcela evidentní, že jedna část obsahuje menu celé webové stránky a jiná představuje již odstavec, z kterého bychom chtěli získat informace. Problém ale nastává, chceme-li aby toto rozdělení stránky uvažovali i počítače a případně informace vyhledávali pouze z oblasti, která obsahuje relevantní text. Této problematice se věnuje právě segmentace webových stránek. [13]

Můžeme tedy říci, že segmentace je strojové rozdělení webové stránky na oblasti, které spolu nějakým způsobem souvisí, například vizuálně či obsahově. Existuje několik způsobů, jak přistupovat k segmentaci. Lze nahlížet na strukturu zdrojového kódu webové stránky, případně hledat ve skupině stránek jednotné rysy (šablony) nebo využít principu, který využívá člověk, čímž je rozdělení stránky podle vizuálního vjemu. Spousta prací, se snaží o kombinaci principů, které byly zmiňovány, aby bylo dosaženo co nejlepšího výsledku segmentace v co možná nejkratším čase.

V následující části kapitoly budou blíže představeny některé přístupy k segmentaci webové stránky.

3.2 Segmentace založená na DOM stromu

Metody segmentace, které jsou založené na DOM[1] stromu jsou také někdy označováno jako metody textové. Je to dáno tím, že tyto metody jsou založeny na analyzování webové stránky bez nutnosti jejího vykreslení, tudíž se analyzují čistě textová data webové stránky, zdrojový kód. Některé textové metody pracují pouze s HTML kódem, mnohem častější přístup je však ten, kdy se zpracovává DOM strom, který odpovídá zdrojovému kódu dané stránky.

Jelikož u této metody není nutnost danou stránku vykreslovat, bývají algoritmy, pracující na základě této metody, zpravidla rychlejší. Rychlost je však také ovlivněna heuristikou, na základě které se provádí analyzování DOM stromu. Zvolená heuristika do značné míry ovlivňuje také kvalitu výsledku segmentace (do jaké míry je rozdělení stránky na jednotlivé oblasti správně provedeno). [13]

Heuristiky pro tuto metodu se mohou navzájem velice odlišovat. Některé mohou pouze analyzovat text, zatímco další mohou být velice komplexní a mohou zahrnovat obrovskou škálu faktorů, které ovlivní dosažený výsledek, ale pravděpodobně i čas potřebný pro analýzu stránky pomocí takové heuristiky, zpravidla negativně.

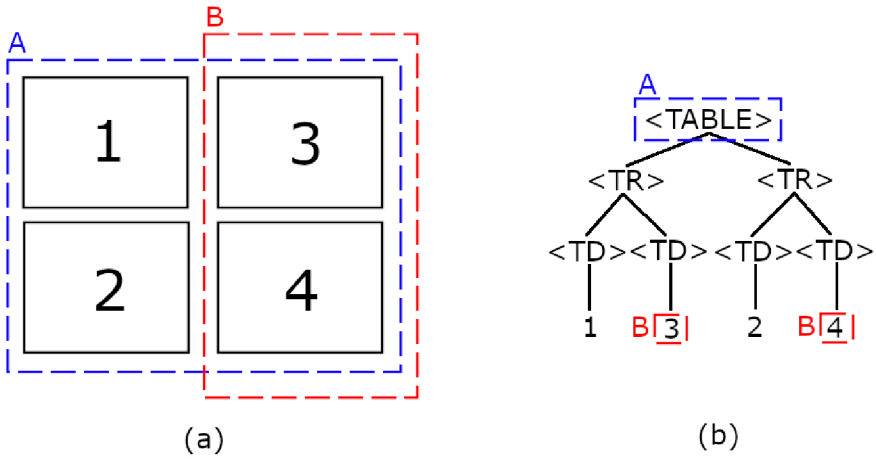
Textové metody však mají jeden obrovský nedostatek. Ať je heuristika jakkoliv složitá, pouze analyzuje strukturu stránky z pohledu na její zdrojový kód, ať už na HTML kód nebo na DOM strom. S ohledem na to, jakým způsobem jsou v dnešní době tvořeny webové stránky, kdy se mnohdy samotné rozvržení a rozmístění jednotlivých prvků stránky provádí až v rámci CSS, se stávají tyto metody značně neefektivními, jelikož se neprovádí samotné vykreselní a tudíž změny rozložení provedeny prostřednictvím CSS vůbec nebudou brány v úvahu. Pomocí CSS lze docílit například toho, že prvek se bude nacházet v úplně jiné části stránky, než by se zdálo z pohledu analýzy DOM stromu. Nehledě na fakt, že v CSS jediným atributem lze prvek stránky zcela zneviditelnit, tedy z pohledu uživatele takový prvek neexistuje a neměl by tedy možnost v takovém prvku hledat relevantní informace. Kdežto při strojovém zpracování takové stránky pomocí textové metody segmentace by takový prvek byl uvažován jako relevantní a byl by nadále zpracováván. [13]

Princip metody

Jak již z názvu vyplývá, algortimy založené na DOM stromu vychází právě z DOM stromu dané stránky, kdy uzly takového stromu jsou reprezentovány HTML tagy[2]. Významnými tagy pro segmentaci jsou `<TITLE>`, `<P>`, `<H1>` - `<H6>`, `<META>`, `<TABLE>` a další které ovlivňují strukturu stránky. Strom se postupně prochází od kořenového uzlu a na základě jisté heuristiky se analyzují jednotlivé uzly stromu a určuje se zda daný podstrom bude již reprezentovat samostatný blok stránky, či bude dále dělen na menší podstromy, respektive bloky. Problém této metody je ale ten, kromě již představeného výše, že například tabulka (`<TABLE>`) se mnohdy nepoužívá pouze na organizaci obsahu, ale také jako prostředek pro rozvržení a zarovnání jednotlivých prvků v rámci stránky. Pokud segmentační algoritmus narazí na stránku kde tomu tak je, není možné, aby daná stránka byla správně segmentována. Tento problém je ilustrován na obrázku 3.1, kde část (a) reprezentuje rozložení stránky pomocí tabulky a část (b) odpovídající DOM strom. Oblast 'A' představuje blok, který bude dále segmentován. Vezmeme-li v úvahu, že oblast 'B' reprezentuje jediný obsahový blok, který byl kvůli rozvržení rozdělen do dvou buněk tabulky, pak nastává problém.

[1] DOM https://www.w3schools.com/whatis/whatis_htmldom.asp
[2] HTML tagy https://www.w3schools.com/tags/

Aby segmentace byla správná, museli by oblasti '3' a '4' být sloučeny. Segmentace založená na DOM stromu však nedokáže označit oblast 'B' jako jediný obsahový blok. [9]



Obrázek 3.1: Problém segmentace založené na DOM stromu

3.3 Segmentace založená na šablonách

Ohledně metody segmentace založené na šablonách je zde velký otazník, zda ji vůbec zařadit mezi plnohodnotné metody, které segmentaci provádí. Důvodem je zejména výstup této metody. Vezmeme-li v úvahu ostatní metody pro segmentaci bude jejich výstup, s jistým nadhledem, stejný, tedy stránka rozdělená na menší oblasti, které spolu nějak souvisí. Metoda stavěná na detekci šablon také vrátí stránku, která je rozdělena na oblasti, ale ty oblasti jsou pouze dvě, šablona a zbylý obsah. [13]

Vzhledem k faktu, že výstup této metody je takto specifický, tak se ve většině případů, kdy je využívána tato metoda, provádí kombinace s dalšími metodami pro segmentaci, kde detekce šablon slouží jako prostředek pro předzpracování stránky.

Metoda detekce šablon je vhodná zejména u webů, kdy jejich autoři využili pro jeho tvorbu techniky, kdy část kódu je předdefinovaná, tzv. šablona, a na základě interakce s uživatelem je zbylý obsah vykreslován dodatečně. Algoritmus pro detekci šablony potom porovnává jednotlivé stránky webu a hledá shodné části stránky. Ty lze s jistou pravděpodobností považovat právě za šablonu. [13]

Jak již bylo zmíněno v úvodu této kapitoly, webové stránky obsahují mnoho prvků, které neobsahují žádné relevantní informace. Ať už jde o navigační prvky, oblast stránky určenou pro příhlášení, reklamní bloky atd. Všechny tyto oblasti jsou většinou společné pro více stránek v rámci webu a proto jsou identifikovány jako součást šablony.

Shrneme-li informace, které byly uvedeny o mětodě detekce šablon i s ohledem na fakt, že ve většině případů nejde nijak ovlivnit míra granularity s jakou jsou prvky stránky filtrovány, je zřejmé za jakým účelem lze tuto metodu využít. Detekce šablon je vhodná právě k oddělení irelevantní části stránky a části s obsahem, který je vhodný k dalšímu zpracování.

3.4 Segmentace založená na vizuálním vnímání

Tato metoda je, dle jejího principu, nejblíže k segmentaci, jak by ji provedl člověk. Tedy využívá vizuální reprezentace stránky a na jejím základě provádí segmentaci. Aby bylo možné tuto metodu použít, je samozřejmě nutné, aby celá stránka byla předem vykreslena z důvodu zahrnutí CSS specifikací. Tím, že je nutné předem celou stránku vykreslit, roste výpočetní náročnost a hlavně čas, potřebný k segmentaci stránky. S ohledem na časovou náročnost vzniklo mnoho projektů, kdy se metoda založená na vizuálním vnímání kombinuje s jinými metodami, čímž bývá dosaženo zkrácení času potřebného pro segmentaci stránky, aniž by došlo ke ztrátě kvality segmentace. [13]

Nejznámějším a nejvyužívanějším algortimem pro vizuální segmentaci stránky je algoritmus **Vi**sion-based **P**age **S**egmentation Algorithm neboli **VIPS**. Spousta projektů, které využívají vizuální segmentaci staví právě na algoritmu VIPS. Buď jej modifikují nebo ho využívají tak, jak byl vytvořen a pouze vylepšují jeho výstup. V následující části textu bude nastíněn princip vizuální segmentace stránky a stručně představen algoritmus VIPS.

Z pohledu algoritmu pro vizuální segmentaci je webová stránka vnímána jako trojice (převzato z [3]):

$$\Omega = (O, \Phi, \delta) \tag{3.1}$$

,kde:

- $O = \Omega^1, \Omega^2, ..., \Omega^N$ je konečná množina bloků, které se vzájemně nepřekrývají a každý blok může být rekurzivně vnímán jako podstránka definována stejnou strukturou jako celá stránka.
- $\Phi = \varphi^1, \varphi^2, ..., \varphi^T$ je konečná množina horizontálních i vertikálních oddělovačů, kdy každý oddělovač má přidělenou váhu, která indikuje jeho viditelnost. Všechny oddělovače v jedné množině Φ mají stejnou váhu.
- $\delta = O \times O \rightarrow \Phi \cup NULL$ je zobrazení popisující vzájemný vztah bloků z množiny O. Například uvažujme dva objekty Ω_i a Ω_j z O, zápis $\delta(\Omega_i, \Omega_j) \neq NULL$ značí, že bloky Ω_i a Ω_j jsou přímo odděleny pomocí oddělovače $\delta(\Omega_i, \Omega_j)$. Nebo také lze říci, že bloky Ω_i a Ω_j jsou sousední. V opačném případě se se mezi bloky Ω_i a Ω_j nacházejí další objekty.

Jak bylo v definici řečeno, tak každý blok lze uvažovat jako samostatnou podstránku, kterou lze dále dělit. S tím souvisí pojem *stupeň konzistence*, který je definován pro každý vizuální blok a uvádí, jak moc je daný blok konzistentní. Stupeň konzistence lze předdefinovat. To už ale hovoříme o *povoleném stupni konzistence*. Pomocí povoleného stupně konzistence jsme schopni ovlivnit úroveň granularity, tedy do jaké míry se jednotlivé bloky budou dále rekurzivně dělit. [3]

Výstupem vizuální segmentace je hierarchický strom vizuálních bloků, kde jednotlivé uzly představují vizuální bloky stránky. Jestliže byl daný blok stránky dále dělen, větví se i odpovídající uzel stromu, kdy jeho potomci představují vnořené vizuální bloky v daném bloku. Listové uzlu reprezentují nejmenší vizuální bloky stránky, které už nebyly dále děleny. Kořenovým uzlem je pak celá stránka. [3]

3.4.1 Postup metody VIPS

Algoritmus VIPS vychází z principu a definic popsaných výše. VIPS se skládá ze tří jednotlivých částí, které na sebe navazují. Extrakce vizuálních bloků, detekce vizuálních oddělovačů, konstrukce vizuální struktury obsahu.

Extrakce vizuálních bloků

Úkolem této fáze je určit z DOM stromu na základě jeho procházení a skupinou jistých rozhodovacích pravidel, které uzly DOM stromu jsou vizuální. Každému extrahovanému uzlu je přidělen stupeň konzistence a dále je posuzováno, zda daný uzel bude dále segmentován. Posuzování probíhá podle vlastností (barva pozadí, HTML tag uzlu, či velikost a tvar bloku) daného uzlu a vlastností jeho potomků. [3]

Detekce vizuálních oddělovačů

Jakmile jsou extrahovány všechny vizuální bloky z DOM stromu, zahájí se dektekování jednotlivých oddělovačů. Ty mohou být horizontální či vertikální, vzájemně se nekříží a jsou reprezentovány dvojicí bodů, které představují počáteční a koncový pixel daného oddělovače. Nejprve probíhá detekce horizontálních resp. vertikálních oddělovačů a až po jejím dokončení proběhne detekce vertikálních resp. horizontálních oddělovačů. Detekce probíhá v několika iteracích. Jednotlivé oddělovače mohou být zpětně upraveny na základě nově detekovaných oddělovačů, případně mohou být úplně odstraněny. Každý oddělovač má navíc přidělenou váhu. [3]

Konstrukce vizuální struktury obsahu

Fáze konstrukce vizuální struktury obsahu pracuje s přidělenými váhami jednotlivých oddělovaču. Začíná u oddělovačů s nejnižší vahou. Bloky, oddělené tímto oddělovačem jsou sloučeny. Proces slučování probíhá dokud není dosaženo oddělovače s nejvyžší váhou. Nejvyžší váha oddělovače také určuje stupeň konzistence nového bloku. Poté proběhne kontrola, zda každý koncový uzel splňuje povolený stupeň konzistence. Jestliže nesplňuje, proběhne opět fáze extrakce vizuálních bloků a tím dojde k jemnějšímu dělení daného bloku. Jakmile všechny koncové uzly daný povolený stupeň konzistence splňují, proběhne sestavení vizuální struktury obsahu. [3]

Kapitola 4

Návrh extrakce témat za účelem segmentace

V rámci této kapitoly bude představen návrh algoritmu, který jsem navrhl, využívající prvky a nástroje sémantického webu, za účelem segmentace webové stránky na základě sémantické analýzy DOM stromu. Tento algoritmus je založen na myšlence analyzování DOM stromu konkrétní webové stránky, kdy je pro textové elementy DOM stromu provedena sémantická analýza, při které se každému z těchto elementů přiřadí témata. Porovnáním těchto témat je následně možné provést segmentaci dané stránky.

Sekce 4.1 přiblíží problematiku určení tématu na základě zdroje. V sekci 4.2 představuje návrh upravení metody pro segmentaci webové stránky založené na analýze DOM stromu tak, že budou elementy zkoumány z hlediska sémantiky.

4.1 Určení tématu zdroje s využitím DBpedia-Spotlight

Jak již bylo v sekci 2.7 řečeno, tak DBpedia-Spotlight poskytuje uživateli možnost anotovat určitý text, kdy jednotlivá anotovaná slova či fráze jsou přímo propojeny se zdroji v rámci DBpedie. Jestliže existuje více kandidátních zdrojů na propojení, lze nechat samotný Spotlight vybrat nejvhodnějšího kandidáta. Lze však přimět Spotlight vrátit celý seznam kandidátů, včetně jejich ohodnocení. Pro tuto práci je klíčový druhý způsob, kdy Spotlight vrátí seznam kandidátů, kteří budou dále zpracováváni.

Nejdříve je ale nutné stanovit si seznam hlavních témat (kategorií), která budou vyhledávána a budou brána jako stěžejní v této práci. Tato témata byla zvolena jako přůřez obsahující nejběžnější a nejvyskytovanější témata. Tato témata jsou Ekonomika, Filmy, Filosofie, Historie, Hry, Hudba, Jídlo, Kultura, Literatura, Móda, Náboženství, Politika, Příroda, Sport, Technologie, Umění, Věda, Vzdělání, Zábava a Zeměpis.

Na základě zkoumání zdroje a jeho vlastností v DBpedii, jsme schopni zařadit daný zdroj do kategorie a ke každé kategorii získat její nadřazenou kategorii. Například vezmeme-li pojem *Fotbal*, tak jednou z jeho přímo nadřazených kategorií je *Týmový sport*. Tímto se přímo nabízí tvorba hierarchického uspořádní ve formě orientovaného grafu, kdy počátečním uzlem grafu bude daný zdroj (pojem) a od tohoto uzlu povedou hrany směrem k jeho nadřazeným kategoriím. V dalším kroku bychom jako počáteční uzly vzali nadřazené kategorie, získané v předchozím kroku a k nim získali opět jejich nadřazené kategorie. Tímto je postupně vytvořena hierarchie kategorií. Přitom se snažíme dosáhnout toho, aby alespoň jedna z ka-

tegorií náležela seznamu hlavních témat, který je uveden výše. Jak rozsáhlý daný graf bude, respektive kdy zastavit rozvoj grafu se řídí následujícím postupem:

- Stanoví se hodnota l_{max}, která určuje maximální uroveň každé větve. Například pokud nastavíme $l_{max} = 1$, znamená to, že vyhledáme pouze nadřazené kategorie zdroje.
- Jestliže je alespoň jedna kategorie z grafu při dané l_{max} ze seznamu hlavních témat, ukončí se rozvoj.
- V případě, že při dané l_{max} nebylo dosaženo uzlu, který by odpovídal jednomu z témat ze seznamu, zvýší se l_{max} o 1 a rozvoj pokračuje.
- Jakmile je dosaženo v některé z větví grafu uzlu, který odpovídá tématu ze seznamu, v rozvoji dané větve se již nadále nepokračuje, ikdyž uroveň zanoření je menší než l_{max}, a rozvíjí se pouze zbylé větve grafu. Tato podmínka je zde z toho důvodu, že například pojem *Umění* je nadřazenou (nemusí být přímou) kategorií pojmu *Hudba* a z důvodu jemnějšího, přesnějšího určení tématu chceme, aby se bralo v úvahu pouze téma *Hudba* a nikoliv *Umění*.

S využitím těchto podmínek bychom měli být schopni sestavit orientovaný graf, kdy alespoň jeden z uzlů odpovídá tématu ze seznamu.

V případě, že bude nalezeno pouze jediné téma získáme tím jediného kandidáta. Mnohdy však získáme několik kandidátních témat a poté je nutné určit míru příslušnosti daného zdroje ke všem těchto kandidátům. Míru příslušnosti bude ovlivňovat několik faktorů:

- Nejkratší cesta z cest, vedoucích ke kandidátním uzlům. Kandidátní uzel, ke kterému vede nekratší cesta by měl mít největší váhu.
- Počet uzlů na nejkratší cestě ke kandidátnímu uzlu
- Počet uzlů vedoucích ke kandidátnímu uzlu

Jakmile pomocí výše uvedeného postupu, uvedených pravidel a podmínek určíme témata jednotlivých zdrojů v textu (částí, které byly anotovány s využitím Spotlight), jsme schopni, na základě poměrů měr příslušnosti jednotlivých témat, určit kandidátní témata celého bloku textu.

Tento pricip vychází z projektu [12], kdy autoři aplikovali tento postup pro analýzu profilů uživatelů na sociálních sítích.

4.2 Využití sémantické analýzy k segmentaci stránky

V sekci 3.2 byl představen algoritmus, který využívá analýzu elementů DOM stromu a na jejich základě provádí segmentaci. Nabízí se zvážit přístup, kdy by se segmentace neprováděla podle HTML tagů, které DOM strom obsahuje, ale provedla se analýza obsahu jednotlivých elementů z hlediska sémantiky a poté se provedla samotná segmentace.

Jednalo by se tedy o modifikaci algortimu pro segmentaci na základě DOM stromu. Postup segmentace by byl do značné míry podobný, tedy procházeli by se jednotlivé elementy DOM stromu, které by byly analyzovány, nicméně by se elementy analyzovali s využitím prvků sémantického webu a postupu popsaného výše. Výsledkem této analýzy by byl DOM strom, u kterého bychom znali témata obsahu jednotlivých elementů.

Dalším krokem by byla analýza těchto témat a určování do jaké míry jsou si jednotlivé elementy podobné z hlediska sémantiky. V případě jisté míry podobnosti by se tyto elementy považovali jako jeden blok, segment. V případě, že by rozdílnost témat obsahu daných elementů překročila jistou hranici, došlo by k rozdělení těchto elementů do samostatných bloků.

4.3 Návrh řešení z hlediska použitých technologií

Pro implementaci aplikace bylo uvažováno mezi dvěma programovacími jazyky, Javou[1] a Pythonem[2]. Jelikož je DBpedia-Spotlight vytvořen v programovacím jazyce Java, přímo se nabízí tento jazyk využít i pro tento projekt, nicméně s ohledem na dostupné moduly pro jazyk Python, které by značně usnadnili implementaci aplikace, byl zvolen právě Python.

Syntaktická analýza webové stránky lze provést například s využitím modulu *lxml*[3], nebo například pomocí modulu *BeautifulSoup*[4].

Pro komunikaci se Spotlight lze využít buď REST API[5], v případě, že by se využívalo webové služby Spotlight, jak bylo popsáno v sekci 2.7 nebo využít zdrojového kódu poskytnutého přímo vývojáři a provozovat Spotlight lokálně.

Více informací ohledně využitých technologií a nástrojů pro implementaci je uvedeno v následující kapitole.

[1] Java https://www.java.com/en/
[2] Python https://www.python.org/
[3] lxml https://lxml.de/
[4] BeautifulSoup https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[5] REST https://en.wikipedia.org/wiki/Representational_state_transfer

Kapitola 5

Implementace segmentační metody

V rámci této kapitoly budou představeny použité nástroje pro implementaci aplikace, bude představena struktura aplikace a v jednotlivých podkapitolách bude představen postup implementace části zabývající se anotací a následně postup implementace segmentační části.

5.1 Implementační jazyk a další použité nástroje

Jak již bylo výše zmíněno, pro implementaci aplikace byl zvolen programovací jazyk Python, konkrétně jeho nejnovější verze 3.8.3. Pro práci se Spotlight byl využit modul *PySpotlight*[1], který poskytuje rozhraní, skrze které lze jednoduše konfigurovat Spotlight a následně pomocí poskytnutých funkcí zahájit komunikaci, kdy *PySpotlight* zformuluje ze zadaných parametrů odpovídající příkaz a po provedení anotace vrátí skrze návratovou hodnotu výsledek anotace ve zvoleném formátu.

V další fázi bylo nutné zajistit komunikaci se SPARQL serverem, aby bylo možné na základě výsledku anotace sestavit hierarchické uspořádání témat a jim nadřazených témat. Pro Python existuje modul *SPARQLWrapper*[2], prostřednictvím kterého je navázána komunikace se serverem, který zprostředkovává dotazy v rámci databáze DBpedie. Stačí tedy pomocí parametru zadat end-point daného serveru a následně zformulovat SPARQL dotaz ve formátu *String*, který je následně skrze *SPARQLWrapper* odeslán na server. Výhodou modulu *SPARQLWrapper* je také to, že lze nastavit formát výsledku vyhledávání. Z důvodu přehlednosti a jednoduché syntaktické analýzy je využit formát JSON.

Z důvodu, že je implementovaná aplikace určena k segmentaci webových stránek, bylo nutné zajistit syntnaktickou analýzu DOM stromu dané webové stránky. Pro tento problém byl využit modul *lxml*, který umožňuje syntaktickou analýzu jednak XML dokumentů, tak i HTML dokumentů. Ze zdrojového kódu, který je obsahem daného dokumetu vytvoří DOM strom, jehož kořen je uložen v rámci proměnné. S DOM stromem lze pracovat velice podobně jako například v jazyce JavaScript[3]. Lze tedy v daném stromu vyhledávat jednotlivé elementy na základě jejich id, případně třídy či atributů. Jednotlivé elementy jsou objekty typu *HtmlElement*.

Jako další nástroj je v práci použit modul *BeautifulSoup*, který lze stejně jako modul *lxml* využít pro syntaktickou analýzu DOM stromu. Nicméně poskytuje další funkcionalitu, která není implementována v rámci modulu *lxml*. *BeautifulSoup* je v práci využit k jedinému účelu

[1] PySpotlight https://github.com/ubergrape/pyspotlight
[2] SPARQLWrapper https://github.com/RDFLib/sparqlwrapper
[3] JavaScript https://www.w3schools.com/js/default.asp

a to konkrétně k předzpracování HTML dokumentu za účelem odstranění nadbytečných HTML tagů, jelikož poskytuje funce, které na základě zadaného HTML tagu vyhledají odpovídající element a tento element nahradí pouze jeho obsahem, tedy odstraní HTML tag a zanechá pouze obsah. Důvod tohoto předzpracování bude více vysvětlem v kapitole 5.3 v sekci 5.3.2.

Jelikož je aplikace navržena, aby pracovala s HTML dokumentem, jenž je uložen v některém z adresářů počítače, byla implementována funkce, jenž umožní uživateli nejprve stáhnout HTML dokument z webu na základě URL a dále již aplikace pracuje se staženým dokumentem. Aplikace pro stažení využívá modul *Selenium*[4] a webový prohlížeč *Firefox*[5]. Více o spuštění aplikace i pro stažení webové stránky naleznete v příloze A.

Mezi další využité nástroje patří již standardní moduly pro Python, sloužící k manipulaci se soubory, výpisu na standardní výstup či modul s matematickými funkcemi.

5.2 Struktura aplikace

Aplikace byla navržena jako konzolová aplikace, tedy skript v Pythonu, které se při jejím spuštění předávají argumenty prostřednictvím příkazového řádku. Více o těchto parametrech v příloze A. Nejprve byla aplikace vyvíjena tak, aby se po spuštění skriptu provedla anotace vstupního dokumentu a následně ihned segmentace. V průběhu vývoje a testování však bylo zjištěno, že zatímco segmentace, která pracuje pouze s DOM stromem na úrovni syntaktické analýzy a zpracování atributů, trvá pro celý HTML dokument zlomky sekund, tak anotační část, kde probíhá zpracování textu s využitím Spotlight a následně komunikace se SPARQL serverem, trvá řádově desítky sekund.

S ohledem na to, že po dokončení implementace budou prováděny experimenty, kdy se pro různé vstupní parametry segmentace bude vyhodnocovat její výstup, byla implementována možnost zadat parametr při spuštění skriptu, který definuje požadovanou akci, tedy zda se má provést anotace, segmentace, případně obě akce společně. Více o spuštění skriptu v příloze A.

Po spuštění skriptu proběhne kontrola vstupních parametrů, se kterými byl daný skript spušten. Proběhne kontrola, zda byla skrze příkazový řádek zadána cesta k souboru, nebo URL. Jestliže byla zadána URL webové stránky, provede se její stažení. Vstupní soubory se ověří, zda existují, zda má skript oprávnění pro jejich čtení a zda mají správný formát. Na základě požadované akce se zavolá příslušná funkce, která provede odpovídající akci. V případě, že je požadováno provést jednak anotaci i segmentaci během jednoho běhu se volají postupně funkce pro anotaci a segmentaci.

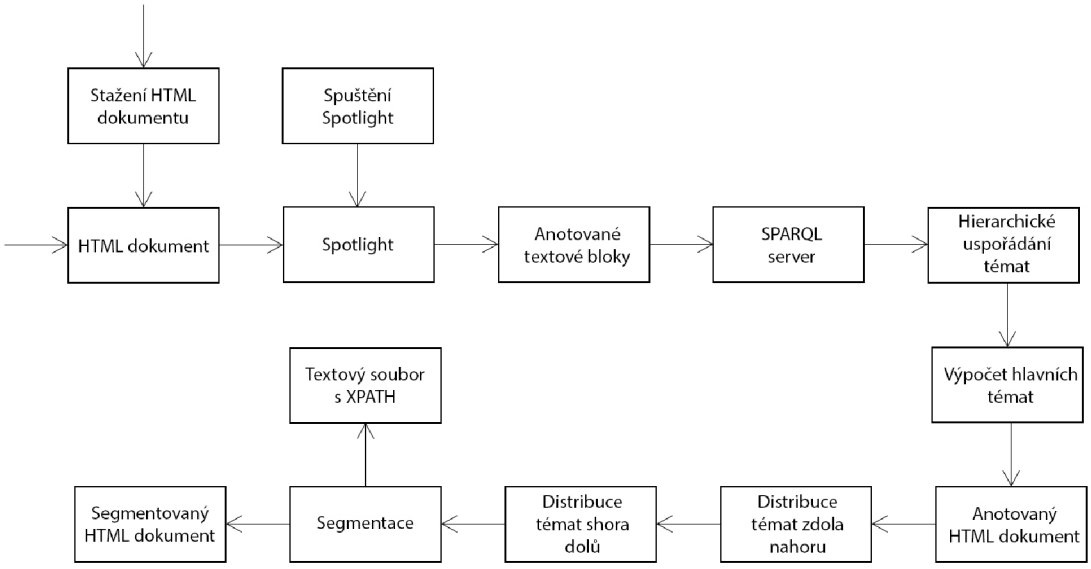
Anotační část zabezpečí navázání komunikace se Spotlight a SPARQL serverem a provede anotaci textových elementů DOM stromu příslušného HTML dokumentu pomocí Spotlight. Díle provede sestavení hierarchického uspořádání témat pro každý anotovaný zdroj, výpočet procentuálního podílu hlavních témat v rámci celého textového bloku a v poslední řadě vytvoření výstupního souboru, který odpovídá vstupnímu HTML dokumentu, kdy jsou příslušné textové bloky doplněny o atribut nesoucí informace o tématech daného bloku. Podrobněji se implementací anotační části bude zabývat sekce 5.3.

Segmentační část nejprve zkontroluje, zda již byl vstupní soubor anotován a následně provede distribuci témat textových bloků napříč DOM stromem. Poté provede rozdělení, segmentaci, DOM stromu na základě těchto témat na jedotlivé bloky, jejichž obsah je té-

[4]Selenium https://www.selenium.dev/
[5]Firefox https://www.mozilla.org/cs/firefox/

maticky podobný. Po provedení segmentace je vytvořen výstupní soubor, který stejně jako v anotační části vychází se vstupního HTML dokumentu, kdy jsou nyní doplněny atributy u daných elementů, které definují, že příslušný element je hraniční a jedná se tedy o tématicky samostatný podstrom, jehož kořenem je daný element. Hodnota, která ovlivňuje granularitu segmentace je zadávána prostřednictvím parametru příkazové řádky při spuštění skriptu. Z důvodu, že s touto hodnotou byly prováděny experimenty, kdy byla snaha o nalezení ideální hodnoty, při které segmentace dosahuje co možná nejlepšího výsledku pro konkrétní stránku, bylo umožněno spustit skript pouze pro segmentaci, zatímco anotovaný soubor zůstával nezměněn a experimenty tak mohli být prováděny v kratším čase. Více podrobností o implementaci segmentační části se nachází v sekci 5.4.

Pro lepší pochopení struktury aplikace bylo vytvořeno jednoduché blokové schéma 5.1, na kterém je ilustrován chod aplikace. Ve schématu nejsou zahrnuty podrobné kroky jednotlivých částí aplikace, ale pouze návaznost hlavních částí aplikace. Podrobněji budou jednotlivé kroky popsány v následujících částech textu.



Obrázek 5.1: Blokové schéma návaznost základních kroků aplikace

5.3 Implementace anotace

V následujícím textu bude podrobně popsán postup implementace anotování HTML dokumentu. Budou představeny jednotlivé části anotace, jako je předzpracování vstupního souboru, spuštění a konfigurace Spotlight, výpočet procentuálního podílu hlavních témat a nakonec vytvoření výstupního souboru.

Anotační část je implementována ve funkci *annotation*, ze které jsou volány další podpůrné funkce implementující úpravu vstupního souboru, anotaci Spotlight, výpočet procentuálního podílu daných témat a dalších.

5.3.1 Problémy související se Spotlight

Jak již bylo v části 2.7 zmíněno, DBpedia-Spotlight je nástroj sloužící k anotaci textu a propojení anotovaných částí se zdroji v DBpedii.

Původní plán jak využívat Spotlight byl prostřednictvím poskytované webové služby a prostřednictvím REST API. Bohužel v průběhu implementace a prvotních testování bylo zjištěno, že poskytovaná webová služba je schopna zpracovat pouze prvních pár požadavků a poté již vrací prázdné výsledky. S ohledem na fakt, že mohou být zpracovávány webové stránky obsahující desítky textových bloků, je tato služba nepoužitelná.

Bylo tedy nutné využít některý ze způsobu, jak spustit Spotlight lokálně. Tuto fázi provázely značné problémy. Jelikož anglický slovník, který Spotlight využívá je rozsáhly a Spotlight nahrává celý slovník do operační paměti počítače, je nutná opravdu velká kapacita operační paměti. První pokusy o spuštění Spotlight byly prováděny na operačním systému *Windows 10*, kde spuštění bylo vždy ukončeno chybou. Proto byl vývoj aplikace přesunut na operační systém Ubuntu 18.4., kde již s využitím Dockeru bylo spuštění Spotlight úspěšné. Nicméně Spolight zabíral 99% kapacity operační paměti a další práce tudíž byla nereálná.

Jediným východiskem bylo rozšíření operační paměti na dvojnásobek (16GB). Poté již Spotlight fungoval dle očekávání.

Jelikož Spotlight využívá operační paměti jako úložiště slovníku, ve kterém vyhledává, způsobuje, že odezva a rychlost anotace textu je závislá na velikosti a rychlosti operační paměti. Dalším parametrem, který výrazně ovlivňuje délku anotace je rychlost zpracování SPARQL požadovku serverem. Je-li SPARQL server příliš vytížený, prodlouží se doba zpracování jediného požadavku několikanásobně. Z tohoto důvodu bylo provedeno oddělení anoační a implementační části, jako bylo popsáno v předchozí kapitole.

Při spuštění Spotlight pomocí Dockeru se definuje, prostřednictvím argumentu, port, na kterém Spotlight lokálně poběží a na kterém s ním tudíž bude probíhat komunikace. Více o spuštění Spotlight naleznete v příloze A.

5.3.2 Vstup a jeho zpracování

Vstupním souborem anotační části aplikace je HTML dokument, obsahující zdrojový kód webové stránky, u níž je požadována segmentace. Anotace i segmentace pracuje s lokálními HTML dokumenty. Webovou stránku, jež je určena k anotaci, lze stáhnout s využitím webového prohlížeče, aby se provedlo stažení všech doplňujících souborů, jako jsou obrázky, CSS soubory či skripty, a tudíž bylo možné provést vizualizaci segmentované stránky tak, aby odpovídala původní webové stránce i vzhledově.

Aby uživatel nemusel webovou stránku stahovat ručně, byla implementována funkce, jenž na základě zadané URL webové stránky provede její stažení. V případě, že tuto funkci uživatel vyžaduje, je nutné spustit aplikaci s požadovanými argumenty. Více v příloze A.

Ze zadaných argumentů je získána jednak adresa dané stránky, tak i cesta k adresáři, kam má být webová stránka stažena, včetně všech doplňujících souboru. Funkce pro stažení webové stránky využívá modul *Selenium*, modul *PyAutoGui*[6] a webový prohlížeč *Firefox*. S využitím modulu *Selenium* je načtena požadovaná webová stránka skrze prohlížeč *Firefox* a následně je vyvolán dialog pro stažení stránky, kam je vložena cesta k adresáři, kam má být stránka stažena. Aby se zabránilo problémům s rozložením klávesnice, bylo nutné vložení cesty pro stažení stánky vložit do dialogového okna s vyvoláním klávesové zkratky

[6] PyAutoGui https://pyautogui.readthedocs.io/en/latest/

”ctrl + v” (pro operační systém Mac ”command + v”). Pro tuto operaci je však nutné mít nainstalovanou utilitu *XSEL*[7].

V případě, že se nepodaří stránku stáhnout, ať už z důvodu nedostupnosti dané stránky, či nedostatečných práv k zápisu do adresáře, aplikace je ukončena s příslušnou chybovou hláškou. Podaří-li se však stránku úspěšně stáhnout, je cesta ke staženému HTML dokumentu uložena do proměnné a pokračuje se anotací daného souboru.

Jestliže je již HTML dokument stažen ručně, nebo jiným způsobem, je cesta k příslušnému souboru zadána prostřednictvím argumentu příkazového řádku při spuštění skriptu. Nejprve dojde k ověření, že se jedná o HTML dokument a skript má veškerá práva k otevření daného souboru. V případě chyby je skript ukončen s příslušnou chybovou hláškou.

Ze vstupního souboru je vytvořena ”pracovní” kopie, u které je upravován zdrojový kód. Vstupní soubor, repektive jeho pracovní kopie, je přečten s využitím modulu *lxml*. Modul *lxml* načte zdrojový kód z HTML dokumentu a převede ho do stromové struktury reprezentující DOM strom daného HTML dokumentu a umožní tak přistupovat k jednotlivým elementům webové stránky, jejich argumentům, vyhledávat jednotlivé elementy na základě *id* daného elementu či s využitím *XPATH*[8].

Nyní je nutné vstupní soubor zbavit nadbytečných HTML tagů. Tento krok je nutné provést z důvodu, aby se blok textu později zpracovával a anotoval jako celek. Jestliže bychom tento krok neprovedli, budou části textu uzavřené v HTML tagu, reprezentující řádkový element, brány jako samostatné elementy, což by z pohledu na DOM strom bylo správně, ale z pohledu sémantiky špatně. Pod označením nadbytečné HTML tagy jsou, v tomto případě, myšleny jednak řádkové tagy upravující jednak vzhled textu jako jsou například tagy <*b*> (tučné písmo), <*i*> (kurzíva), <*u*> (podtržené písmo) a mnoho dalších, tak i řádkové tagy, sloužící k logickému formátování textu, jako je například tag <*span*>. Mezi tagy, které je nutné odebrat, náleží i tag <*a*>, tedy odkaz.

Odebráním tagů je samozřejmě myšleno odstranění samotného tagu, s tím, že text uzavřený v daném tagu zůstane zachován a bude tak začleněn do okolního textu. V případě odkazů (tag <*a*>) bude zachován zobrazovaný text, který daný odkaz zastupuje. Pro odstranění nadbytečných tagů je v této práci využíván modul *BeautifulSoup*[9], který umožňuje elegantně odstranit požadované tagy. Na figuře 5.1 je části HTML kódu sloužící jako vstup a následně na 5.2 je výsledný, upravený kód.

```
<div>
    <p>
        <i>Ukazkovy</i> <b>text</b> s odkazem na
        <a href="https://www.wikipedia.com">Wikipedii</a>.
    </p>
</div>
```

Výpis 5.1: Odebrání nadbytečných HTML tagů - vstup

Po odstranění nadbytečných tagů je již HTML dokument připraven pro anotovaní prostřednictvím Spotlight. Z HTML dokumentu, který byl upraven odebráním nadbytečných

[7] XSEL https://linux.die.net/man/1/xsel
[8] XPATH https://www.w3schools.com/xml/xpath_intro.asp
[9] BeautifulSoup https://www.crummy.com/software/BeautifulSoup/bs4/doc/

```
<div>
    <p>
        Ukazkovy text s odkazem na
        Wikipedii.
    </p>
</div>
```

Výpis 5.2: Odebrání nadbytečných HTML tagů - výstup

tagů se nyní extrahují elementy obsahující text. Seznam extrahovaných elementů slouží jako vstupní atribut funkce provádějící anotaci textových elementů.

5.3.3 Anotace textových elementů s využitím Spotlight

Ve fázi, kdy je vstupní dokument upraven pro anotaci, je zavolána metoda pro extrakci textových elementů. DOM strom je procházen metodou pre-order[10] a analyzují se jednotlivé elementy z hlediska toho, jestli se jedná o textový element, tedy o element jehož text je viditelný a zároveň je relevantní z hlediska sémantiky na stránce. Elementy typu *comment*, *style* a *script* nejsou tudíž brány v úvahu i přesto, že obsahují text. Relevantní textové elementy jsou přidávány do datové struktury typu *seznam*, která slouží jako návratová hodnota funkce.

Pro komunikaci se Spotlight, jak již bylo výše zmíněno, je využíván modul *PySpotlight*. S využitím rozhraní, které *PySpotlight* implementuje, se na základě vstupních parametrů, jako je end-point (url kde běží Spotlight), vstupní text, jenž má být anotován a paramtery sloužící ke konfiguraci důvěryhodnosti anotace, naváže komunikace se Spotlight a je vytvořen dotaz pro anotaci, který je odeslán do Spotlight. *PySpotlight* implementuje funkce *annotate* a *candidates*, které odpovídají příkazům pro Spotlight uvenedých v sekci 2.7. V této práci je použita funkce *candidates*, což odpovídá příkazu *candidate*. Parametry ovlivňující důvěryhodnost zdroje je možné definovat při spuštění skriptu, více v příloze A. Jestliže pro definované paramatery nedokáže Spotlight nalézt důvěryhodný zdroj, proběhne snížení hodnot těchto parametrů a provede se opětovný pokus o anotaci. Toto snižování probíhá pouze do určité meze, jinak by bylo dosáhnuto nesmyslných výsledků.

Jestliže Spotlight úspěšně anotuje daný text, vrátí výsledek ve formátu JSON, obsahující *label* (popisek daného zdroje v DBpedii), hodnotu určující s jakou přesností propojil Spotlight daný zdroj s textem (dále bude uváděna jako skóre), a další parametry, jež však pro tuto práci nejsou důležité. Jednotlivé popisky zdrojů, včetně skóre jsou ukládány do datové struktury slovník a budou následně použity pro výpočet hlavních témat.

Tímto způsobem jsou anotovány všechny textové elementy. Na konci této části tak existuje pro každý textový element slovník, obsahující popisky zdrojů v DBpedii, jako klíče a odpovídající skóre jako hodnoty.

V následující části bude popsán a ilustrován způsob, jakým se z popisků zdrojů a skóre zjistí tématické okruhy daných textových bloků a jak se vypočítá jejich procentuální přínos.

5.3.4 Určení a výpočet hlavních témat

Jak již bylo zmíněno v sekci 4.1, princip určení tématu zdroje vyhází z velké části z projektu [12], kde používali Spotlight k profilování uživatelských účtů na sociálních sítí. V sekci 4.1

[10] Pre-order průchod stromem https://en.wikipedia.org/wiki/Tree_traversal

byl však uveden obecný princip metody, proto bude nyní popsán podrobněji včetně ilustrací pro lepší pochopení dané metody.

Pro lepší orientaci bude rozdělen popis metody na část věnovanou určení hlavních témat a část zaměřenou na výpočet procentuálního podílu jednotlivých tématických okruhů.

Určení témat na základě zdroje

Budeme-li zkoumat zdroj v rámci DBpedie, budeme schopni o daném zdroji prohlásit například jakého je typu, datum narození, jestliže daný zdroj je typu osoba a spoustu dalších informací. Pro nás jsou podstatné dvě informace. *Label*, na jehož základě propojíme zdroj s výstupem, který nám poskytne Spotlight a *Subject*, který nám určuje v podstatě téma, kategorii zdroje. Těchto témat je zpravidla několik. Každé z těchto témat má ve většině případů svoje nadřazené téma, kategorii. A takto lze rekurzivně pokračovat dále, přičemž se vytváří hierarchické uspořádání, v němž dříve či později narazíme na jedno z hlavních témat, tak jak je popsáno v sekci 4.1.

Na obrázku 5.2 je znázorněno zmíněné hierarchické uspořádání, jehož kořenem je zdroj, který je určen pomocí Spotlight. Jako příklad je zde uveden zdroj "Thor"[11]. Další úroveň tvoří témata *Bůh (God)* a *Norská mytologie (Norse mythology)*[12]. Jestliže budeme rozvíjet téma *God*, již na 3. úrovni zanoření dosáhneme na jedno z hlavních témat, *Náboženství (Religion)*. Budeme-li rozvíjet téma *Norse mythology*, nejdříve na hlavní téma narazíme až na 4. úrovni. Pokud by takto vypadala úplná hierarchie, mohli bychom prohlásit, že téma *Religion* má větší procentuální podíl na celkovém tématu, jelikož na něj narazíme na nižší úrovni a obě témata se oběvili pouze jednou.

Podstatným faktem, jenž je na obrázku 5.2 také ilustrován, je fakt, že budeme-li dále rozvíjet hlavní téma, můžeme narazit na jedno z dalších hlavních témat. Konrétně je zde ilustrováno rozvíjení tématu *History*, kdy dosáhneme faktu *Culture*. To pro nás ale znamená problém. Při určování témat se totiž vždy snažíme dosáhnout co možná nejkonkrétnějšího tématu a takto by téma *Culture* mělo procentuální podíl na celkovém tématu. Je tedy nutné, aby se v případě, kdy se na nějaké úrovni objeví hlavní téma toto téma již dále nerozvíjelo. Správně by tedy mělo být dosaženo pouze tématu *History* a *Religion*, zatímco tématu *Culture* by nikdy nemělo být dosaženo (vztaženo k této ilustraci).

Může také nastat situace, kdy je v různých větvích v hierarchii dosaženo stejných hlavních témat. To bude mít vliv hlavně při výpočtu procentuálního podílu tématického okruhu. Jelikož dosáhneme-li jednoho z hlavních témat několikanásobně víckrát, tak je logické, že toto téma musí mít razantní vliv na celkové téma a tudíž bude mít větší procentuální podíl.

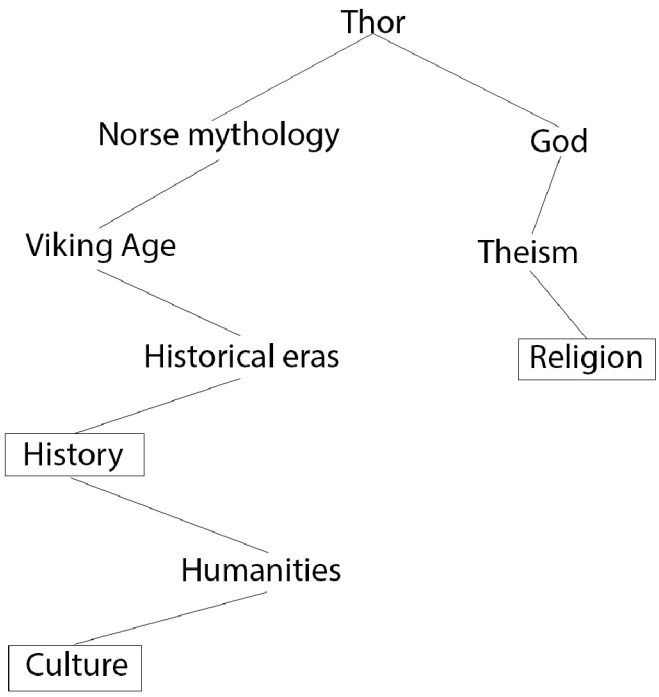
O aspektech ovlivňujících podíl daného tématu jako je například počet výskytů daného tématu, úroveň na které daného tématu bylo dosaženo atd. bude větší zmínka v sekci 5.3.4, kde budou tyto aspekty zařazeny přímo do výpočtů.

Vytvoření výše popisované hierarchie a následná analýza této hierarchie (na jaké úrovni se objevují hlavní témata, v jakém počtu se hlavní témata objevují atd.) je jedním z klíčových kroků metody pro určení témat daného zdroje.

Jedním z postupů, jakým lze vytvářet hierarchické uspořádání témat je systematicky procházet všechny témata daného zdroje, následně jejich nadřazené témata a postupně vše ukládat do vnitřní struktury odpovídající stromu a následně v této struktuře prová-

[11] Jelikož se v této práci používá anglický slovník DBpedie, jsou na obrázku zachovány původní názvy témat a zdrojů

[12] Na této i dalších úrovni je pochopitelně daleko více dosažitelných témat, nicméně v rámci přehlednosti ilustrace byla vynechána



Obrázek 5.2: Hierarchie témat na základě zdroje

dět vyhledávání. V rámci experimentů, byl částečně tento postup implementován, nicméně se ukázal jako velice neefektivní a proto byl nahrazen jiným postupem, který využívá sílu dotazovacího jazyka nad RDF záznamy, jazyka *SPARQL*.

Využití SPARQL

Hlavním negativním aspektem postupu, kdy se postupně vytváří hierarchie témat je větvení témat, která na žádné úrovni nedosáhnou žádného z hlavních témat a tudíž by tato větev byla zpracovávána zcela zbytečně. I s ohledem na to, že některé zdroje mohou mít desítky témat a tato témata opět desítky svých nadřazených témat dochází k "explozi" větvení a následné zpracování této hierarchie vytváří velké časové a paměťové nároky. Tímto se dostáváme k myšlence využít dotazovací jazyk SPARQL a celou práci ohledně vytváření hierarchie a následného vyfiltrování pouze relativních výsledků nechat na SPARQL serveru.

Je tedy nutné zformulovat dotaz, na jehož základě SPARQL určí, zda se na nějaké úrovni nachází alespoň jedno z hlavních témat pro daný zdroj. Koncept dotazu sloužící k určení témat na základě zdroje je znázorněn na 5.3[13]. Nyní bude tento koncept podrobněji popsán.

Klíčovým slovem *PREFIX* definujeme jmenný prostor. Samotný dotaz by se dal slovně popsat takto: "Pro daný zdroj (?concept), jehož atribut *rdfs:label* má hodnotu Label v angličtině (@en), vem hodnoty atributu *dct:subject*, jejichž nadřazené téma (*skos:broader*) je *?theme*, přičemž *?theme* nabývá některé z hodnot *Arts, Cuisine, ... , Technology* a vrať hodnotu daného *?theme*." Tímto dotazem bychom pro daný *Label* dokázali nalézt hlavní téma v případě, že by se nacházelo maximalně na 2. úrovni zanoření (1. úroveň tvoří sa-

[13]Nejedná se o celý dotaz, ale pouze o koncept

```
PREFIX dct: <http://purl.org/dc/terms/>
SELECT ?theme WHERE {
    ?concept rdfs:label Label@en; dct:subject ?subject.
    ?subject skos:broader ?theme.
    FILTER ?theme IN (
        <http://dbpedia.org/resource/Category:Arts>,
        <http://dbpedia.org/resource/Category:Cuisine>,
        <http://dbpedia.org/resource/Category:Culture>,
        <http://dbpedia.org/resource/Category:Economics>,
        <http://dbpedia.org/resource/Category:Education>,
        <http://dbpedia.org/resource/Category:Entertainment>,
        <http://dbpedia.org/resource/Category:Fashion>,
        <http://dbpedia.org/resource/Category:Film>,
        <http://dbpedia.org/resource/Category:Games>,
        <http://dbpedia.org/resource/Category:Geography>,
        <http://dbpedia.org/resource/Category:History>,
        <http://dbpedia.org/resource/Category:Literature>,
        <http://dbpedia.org/resource/Category:Languages>,
        <http://dbpedia.org/resource/Category:Music>,
        <http://dbpedia.org/resource/Category:Nature>,
        <http://dbpedia.org/resource/Category:Philosophy>,
        <http://dbpedia.org/resource/Category:Politics>,
        <http://dbpedia.org/resource/Category:Religion>,
        <http://dbpedia.org/resource/Category:Science>,
        <http://dbpedia.org/resource/Category:Sport>,
        <http://dbpedia.org/resource/Category:Technology>)
}
```

Výpis 5.3: Koncept SPARQL dotazu

motný *subject*). Je tedy nutné dotaz upravit tak, aby dokázal nalezt hlavní téma i na vyšších úrovních zanoření, jelikož jen ve výjimečných případech je nalezeno hlavní téma již do dvou úrovní zanoření.

Nejprve je nutné určit počáteční maximální úroveň zanoření l_{max}, do které SPARQL bude provádět větvení. Pro tuto práci byla zvolena hodnota $l_{max} = 6$. Určení této hodnoty vycházelo z experimentů, kdy byla hledána hodnota, při které bude pro většinu zdrojů nalezeno jedno z hlavních témat a zároveň bude co nejmenší větvení hierarchie. V případě, že při aktuální hodnotě l_{max} nebude dosaženo hlavního tématu, zvýší se l_{max} o 1, dotaz bude upraven a provede se znovu. Zvyšování l_{max} však nemůže být prováděno do nekonečna, jelikož při zvyšování úrovně zanoření rostou časové nároky a proto by celkový čas nutný pro anotaci celého dokumentu mohl být neúnosný a také by se razantně zvýšila vytíženost SPARQL serveru. Hranice, do které bude l_{max} navyšováno byla zjištěna opět experimentálním způsobem a byla tak stanovena na hodnotu 13, tedy větvení bude v nejhorším případě probíhát až do úrovně 13.

Kvůli možnosti zvyšování l_{max} je nutné celý dotaz ve SPARQL vytvářet dynamicky. Dotaz je ve formátu string a aby byla možná jeho dynamická tvorba, byl rozdělen na části, které se dynamicky mění a na části, které zůstávají statické. Části, které se mění byly

nahrazeny proměnnými, jejichž obsah je editován dle potřeby. Upravený dotaz je znázorněn na 5.4.

V dotazu figurují čtyři proměnné (uvedeno s defaultními hodnotami):

```
PREFIX dct: <http://purl.org/dc/terms/>
SELECT dynamic_select WHERE {
    ?concept rdfs:label dynamic_label@en; dct:subject ?subject.
    dynamic_where.
    FILTER dynamic_filter IN (
        <http://dbpedia.org/resource/Category:Arts>,
        <http://dbpedia.org/resource/Category:Cuisine>,
        <http://dbpedia.org/resource/Category:Culture>,
        <http://dbpedia.org/resource/Category:Economics>,
        <http://dbpedia.org/resource/Category:Education>,
        <http://dbpedia.org/resource/Category:Entertainment>,
        <http://dbpedia.org/resource/Category:Fashion>,
        <http://dbpedia.org/resource/Category:Film>,
        <http://dbpedia.org/resource/Category:Games>,
        <http://dbpedia.org/resource/Category:Geography>,
        <http://dbpedia.org/resource/Category:History>,
        <http://dbpedia.org/resource/Category:Literature>,
        <http://dbpedia.org/resource/Category:Languages>,
        <http://dbpedia.org/resource/Category:Music>,
        <http://dbpedia.org/resource/Category:Nature>,
        <http://dbpedia.org/resource/Category:Philosophy>,
        <http://dbpedia.org/resource/Category:Politics>,
        <http://dbpedia.org/resource/Category:Religion>,
        <http://dbpedia.org/resource/Category:Science>,
        <http://dbpedia.org/resource/Category:Sport>,
        <http://dbpedia.org/resource/Category:Technology>)
}
```

Výpis 5.4: Upravený SPARQL dotaz s proměnnými

- dynamic_select = "?subject ?1 ?2 ?3 ?4 ?5"
- dynamic_label = hodnota proměnné se mění podle aktuálně zpracovávaného popisku zdroje
- dynamic_where = "?subject skos:broader ?1. ?1 skos:broader ?2. ?2 skos:broader ?3. ?3 skos:broader ?4. ?4 skos:broader ?5."
- dynamic_filter = "?subject || ?1 || ?2 || ?3 || ?4 || ?5"

Každá z těchto proměnných má nastavenou defaultní hodnotu pro počáteční $l_{max} = 6$. Pro jednoduchost byly pro názvy proměnných v rámci dotazu (*?proměnná*) zvoleny čísla. Usnadňuje to přidávání proměnných do dotazu, jelikož nová proměnná bude mít název odpovídající aktuální hodnotě $l_{max} - 1$ (-1 proto, aby název proměnné odpovídal aktuální úrovni zanoření, jelikož 1. úroveň tvoří *subject*). Jestliže je tedy nutné zvýšit hodnotu l_{max} o 1, proběhne úprava proměnných následovně:

- **dynamic_select** = "?subject ?1 ?2 ?3 ?4 ?5 ?6"
- **dynamic_label** = zůstane stejná, jelikož je zpracováván stále stejný zdroj
- **dynamic_where** = "?subject skos:broader ?1. ?1 skos:broader ?2. ?2 skos:broader ?3. ?3 skos:broader ?4. ?4 skos:broader ?5. ?5 skos:broader ?6."
- **dynamic_filter** = "?subject || ?1 || ?2 || ?3 || ?4 || ?5 || ?6"

Když už víme, jak je SPARQL dotaz tvořen, shrneme si postup hledání hlavních témat. Z předchozí fáze, kdy Spotlight provedl anotaci, existuje pro každý textový blok slovník obsahující záznamy, jejichž klíče jsou jednotlivé popisky zdrojů a jejich hodnoty tvoří skóre s kterým Spotlight propojil zdroj se záznamem v DBpedii. Nyní se budou postupně procházet všechny popisky zdrojů pro daný textový blok a na jejich základě bude vytvářen SPARQL dotaz. Tento dotaz je s využitím modulu *SPARQLWrapper* odeslán na end-point, na kterém běží server vyhodnocující dotazy v rámci databáze DBpedie. Pro tuto práci je využit end-point `http://dbpedia.org/sparql`. Jestliže pro dané l_{max} nebyl nalezen žádný výsledek, zvýší se jeho hodnota o 1, přeformuluje se dotaz a tento dotaz je opět zaslán ke zpracování. Pakliže l_{max} dosáhne hodnoty 13 a stále nebyl nalezen výsledek, končí hledání jako neúspěšné a pokračuje se zpracováním dalšího popisku zdroje.

Jakmile je úspěšně nalezen výsledek, který je ve formátu JSON, přejde se k jeho zpracování. Pro další výpočty je nutné zjistit z výsledku dva podstatné faktory (u obou faktorů se z jednotlivých výsledků vyhledávání bere v potaz pouze hlavní téma, které bylo nalezeno na nejnižší úrovni, aby se zabránilo započítání tématu, které vzniklo větvením jiného hlavního tématu):

- Na jakých úrovních bylo téma nalezeno - odpovídá názvu proměnné, jelikož, jak již bylo výše popsáno, tak názvy odpovídají úrovním zanoření
- Kolikrát se dané téma vyskytlo - jakmile se nalezne hlavní téma, provede se navýšní jeho výskytu o 1

Tímto způsobem jsou zpracovány všechny záznamy z výsledku dotazování napříč všemi textovými bloky.

Na konci této fáze existuje pro všechny popisky zdrojů daného textového elementu informace o tom, jakých témat nabývá (kandidátní témata), na jakých úrovních zanoření byla témata nalezena a v jakém počtu se jednotlivá témata vyskytla.

Výše získané informace jsou naprosto klíčové pro další krok, výpočet procentuálního podílu tématického okruhu.

Výpočet procentuálního podílu tématického okruhu

Rovnice, které se vyskytují v této části, byly převzaty z práce [12] s úpravami, aby proměnné odpovídali problematice v této práci (v [12] se zabývali profilováním uživatelských účtů na sociálních sítích). Pro každý textový element známe jeho kandidátní témata, včetně informací o jejich výskytu v rámci hierarchie. Tudíž lze z těchto informací výpočítat, jaký procentuální přínos mají jednotlivá témata.

Stanovme si několik proměnných:

- MT - hlavní téma[14]

[14] Pro rekapitulaci si tato hlavní témata připomeneme: Ekonomika, Filmy, Filosofie, Historie, Hry, Hudba, Jídlo, Kultura, Literatura, Móda, Náboženství, Politika, Příroda, Sport, Technologie, Umění, Věda, Vzdělání, Zábava a Zeměpis

- L_{MT} - seznam všech hlavních témat
- N_{MT} - celkový počet hlavních témat, tedy $L_{MT} = (MT_i, MT_{i+1}, ..., MT_{N_{MT}})$
- T - text, který je aktuálně zpracováván (textový blok)
- s - zdroj, jenž Spotlight propojil s částí textu

S využitím těchto proměnných jsou sestaveny rovnice pro výpočet procentuálního podílu tématického okruhu následovně.

Nechť T je textový blok, jenž chceme zpracovat a nechť MT_i je i-té hlavní téma ze seznamu hlavních témat L_{MT}. Procentuální podíl pp (v %) tématu MT_i v rámci textového bloku T vypočítáme následovně:

$$pp(T, MT_i) = \left(\sum_{j=1}^{N_s} (cnt_s(j) \times w[s_j, MT_i]/N_R(T)) \times score \right) \times 100 \tag{5.1}$$

, kde:

- N_s je celkový počet všech zdrojů v daném textovém bloku
- $cnt_s(j)$ je počet výskytů j-tého zdroje v rámci textového bloku
- $w[s_j, MT_i]$ je váha definující sílu vztahu mezi zdrojem s_j a hlavním tématem MT_i
- $N_R(T)$ je normalizační hodnota pro celý textový blok
- *score* je hodnota určující s jakou přesností propojil Spotlight daný zdroj s textem (tuto hodnotu vrátí přímo Spotlight při anotaci)
- konstanta 100 je zde, aby byl výsledek v %

Normalizační hodnota $N_R(T)$ je vypočtena na základě této rovnice:

$$N_R(T) = \sum_{i=1}^{N_{MT}} \sum_{j=1}^{N_s} cnt_s(j) \times w[s_j, MT_i] \tag{5.2}$$

, kde:

- N_{MT} je celkový počet hlavních témat
- N_s je celkový počet všech zdrojů v daném textovém bloku
- $cnt_s(j)$ je počet výskytů j-tého zdroje v rámci textového bloku
- $w[s_j, MT_i]$ je váha definující sílu vztahu mezi zdrojem s_j a hlavním tématem MT_i

Váhu definující sílu vztahu mezi zdrojem s_j a hlavním tématem MT_i vypočítáme následovně:

$$w(s, MT_i) = \left(\frac{n_p(s, MT_i)}{n_{tot}(s)} e^{-\alpha[l_p(s,MT_i) - l_{MIN}(s)]} \right) / N_R(s) \tag{5.3}$$

, kde:

- $n_p(s, MT_i)$ je celkový počet cest vedoucích v hierarchickém uspořádání od zdroje s k hlavnímu tématu MT_i, neboli kolikrát se hlavní téma MT_i vyskytlo v hierarchickém uspořádání, jehož kořenem byl zdroj s
- $n_{tot}(s, MT_i)$ je počet všech cest vedoucích v hierarchickém uspořádání od zdroje s k některému z hlavních témat MT, neboli součet výskytů všech hlavních témat
- e je Eulerovo číslo
- α je korekční konstanta jejíž hodnota je > 0, tato konstanta umožňuje korekci výsledku vyhodnocení procentuálního podílu jednotlivých témat tak, aby výsledek byl nejuspokojivější (její hodnota byla zjištěna experimentálně v práci [12])
- $l_p(s, MT_i)$ je cesta vedoucí v hierarchickém uspořádání od zdroje s k hlavnímu tématu MT_i. V případě, že existuje více cest od zdroje s k hlavnímu tématu MT_i, vybere se ta nejkratší z nich. Neboli na jaké nejmenší úrovni zanoření se v hierarchickém uspořádání, jehož kořenem byl zdroj s, vyskytlo hlavní téma MT_i
- $l_{MIN}(s)$ je nejkratší cesta vedoucí v hierarchickém uspořádání od zdroje s k některému z hlavních témat MT, neboli na jaké nejmenší úrovni zanoření se v hierarchickém uspořádání, jehož kořenem byl zdroj s, vyskytlo některé z hlavních témat MT
- $N_R(s)$ je normalizační hodnota pro zdroj s

Normalizační hodnota $N_R(s)$ je dána následující rovnicí:

$$N_R(s) = \sum_{i=1}^{N_s} \frac{n_p(s, MT_i)}{n_{tot}(s)} e^{-\alpha[l_p(s,MT_i) - l_{MIN}(s)]} \tag{5.4}$$

, kde:

- N_s je celkový počet všech zdrojů v daném textovém bloku
- $n_p(s, MT_i)$ je celkový počet cest vedoucích v hierarchickém uspořádání od zdroje s k hlavnímu tématu MT_i, neboli kolikrát se hlavní téma MT_i vyskytlo v hierarchickém uspořádání, jehož kořenem byl zdroj s
- $n_{tot}(s, MT_i)$ je počet všech cest vedoucích v hierarchickém uspořádání od zdroje s k některému z hlavních témat MT, neboli součet výskytů všech hlavních témat
- e je Eulerovo číslo
- α je korekční konstanta jejíž hodnota je > 0, tato konstanta umožňuje korekci výsledku vyhodnocení procentuálního podílu jednotlivých témat tak, aby výsledek byl nejuspokojivější (její hodnota byla zjištěna experimentálně v práci [12])
- $l_p(s, MT_i)$ je cesta vedoucí v hierarchickém uspořádání od zdroje s k hlavnímu tématu MT_i. V případě, že existuje více cest od zdroje s k hlavnímu tématu MT_i, vybere se ta nejkratší z nich. Neboli na jaké nejmenší úrovni zanoření se v hierarchickém uspořádání, jehož kořenem byl zdroj s, vyskytlo hlavní téma MT_i
- $l_{MIN}(s)$ je nejkratší cesta vedoucíc v hierarchickém uspořádání od zdroje s k některému z hlavních témat MT, neboli na jaké nejmenší úrovni zanoření se v hierarchickém uspořádání, jehož kořenem byl zdroj s, vyskytlo některé z hlavních témat MT

Celý výpočet je implementován ve funkci *count_score_of_text_block*, která jako vstupní parametr obdrží seznam slovníků, kdy klíče jednotlivých záznamů ve slovníku odpovídají zdrojům a hodnoty záznamů obsahují informace o hierarchickém uspořádání vytvořeném pro daný zdroj. Funkce poté prochází jednotlivé zdroje, z kterých jsou extrahovaný požadované informace a ty jsou následně vloženy jako proměnné do rovnic.

Jednotlivé textové bloky jsou prostřednictvím cyklu procházeny a nad každým blokem je zavolána výše uvedená funkce *count_score_of_text_block*, jíž je předán seznam zdrojů pro daný blok. Tato funkce vrací jako výstupní hodnotu slovník, jehož klíči jsou hlavní témata a hodnoty tvoří číselná hodnota odpovídající procentuálnímu podílu daného tématu v rámci textového bloku.

5.3.5 Výstup

Při plánování, v jaké podobě bude výstup anotační části, byla vize, kdy elementy obsahující text v původním HTML dokumentu budou doplněny o atribut, jehož hodnota bude obsahovat řetězec ve formátu JSON, jehož prvky budou ve formě dvojic (klíč, hodnota), příčemž klíče budou hlavní témata a hodnoty budou procentuální podíl daného tématu.

Jelikož se během anotace zpracovávala pracovní kopie originálního souboru, ve které byly odstraněny některé tagy a tudíž pracovní kopie již plně neodpovídala originálnímu souboru bylo nutné zvolit složitější přístup než pouhé přepsání obsahu původního dokumentu novým obsahem.

Pro vytvoření výstupního dokumentu byl opět využit modul *lxml*. Ten totiž umožňuje zjistit XPATH libovolného elementu v rámci DOM stromu. Proto byl vytvořen seznam elementů, kterým v rámci anotace byly přiřazeny témata s jejich procentuálním podílem, tedy textové bloky u nichž anotace proběhla úspěšně. Na základě tohoto seznamu byl vytvořen nový seznam, který obahuje řetězce odpovídající textovým podobám XPATH daných elementů.

Jakmile již známe XPATH elementů, které mají přiřazené témata, stačí tato témata překopírovat do originálního dokumentu k elementům s odpovídajícím XPATH.

Vytvoří se tedy nová kopie originálního dokumentu. Pomocí cyklu se prochází seznam obsahující XPATH elementů s přiřazenými tématy a v nové kopii se na základě aktuálního XPATH vyhledá odpovídající element, do kterého jsou překopírována přiřazená témata.

Po dokončení kopírování témat je ještě přidán k elementu *body* atribut, který značí, že daný HTML dokument byl již úspěšně anotován. Tento krok umožní ve fázi segmentace jednoduchou kontrolu, zda byl již soubor anotován. Následně je daný soubor uložen pod upraveným názvem originálního souboru, který je doplněn o příponu "_annotated". Na závěr procesu anotace je odstraněna "pracovní" verze vstupního souboru.

5.4 Implementace segmentace

V rámci této sekce bude představen princip a způsob implementace části zajišťující segmentaci HTML dokumentu na základě témat elementů v DOM stromu.

Bude popsáno zpracování vstupního souboru, princip a odůvodnění distribuce témat napříč DOM stromem, způsob, jakým je rozhonuto, že se daný element a jeho podstrom segmentuje a nakonec vytvoření a podoba výstupních souborů segmentace.

5.4.1 Vstup a jeho zpracování

Vstupní soubor, respektive cesta ke vstupnímu souboru, je zadávána prostřednictvím argumentu příkazové řádky při spuštění skriptu. Je provedena kontrola, zda daný soubor má správnou příponu a zda má skript práva k otevření souboru a k jeho čtení.

Další podmínkou je, že daný soubor již musí být anotován. Proto se nejprve provede kontrola, zda element *body* obsahuje atribut značící, že soubor byl anotován. Tato kontrola se provádí pouze v případě, že byl skript spuštěn pouze pro segmentaci. Spustí-li se s požadavkem provést anotaci a segmentaci zároveň, tak se daná kontrola neprovádí, jelikož výstupní soubor z anotace je automaticky zvolen jako vstupní soubor pro segmentaci.

Jestliže vstupní soubor splňuje veškeré podmínky, vytvoří se "pracovní" kopie daného souboru, z důvodu, aby byl vstupní soubor zachován v nezměněné podobě.

S využitím modulu *lxml* se, stejně jako v případě anotace, vytvoří interní reprezentace DOM stromu daného HTML dokumentu. Kořenový element je uložen v proměnné, skrze kterou se k danému stromu přistupuje.

5.4.2 Distribuce hlavních témat napříč DOM stromem

Po provedení anotační části mají elementy obsahující textové bloky přiřazená hlavní témata včetně procentuálního ohodnocení. Aby bylo možné provést segmentaci na základě těchto témat, je nutné, aby každý element měl přiřazená hlavní témata. Toho docílíme procesem, kdy se provede distribuce těchto témat napříč celým DOM stromem.

Hlavní témata jsou přiřazena zejména listovým elementům, proto je nutné nejprve provést distribuci témat zdola nahoru. V DOM stromě však existují podstromy, kde žádný z elementů nemá přiřazená témata. Proto je nutné provést distribuci témat i shora dolů, aby se témata rozšířila po celém stromě. Během distribuce je nutné kontrolovat typ elementu, kterému mají být témata předána, jelikož pokud by daný element byl například typu *comment*, distribuce by způsobila pád aplikace, protože by se provedl pokus přiřadit komentáři atribut, což není možné.

Nyní budou popsány principy a situace, které mohou nastat jednak při distribuci témat zdola nahoru, tak i shora dolů.

Zdola nahoru

V rámci distribuce hlavních témat ze zdola nahoru je nejprve zavolána funkce, jenž s využitím modulu *lxml* vyhledá všechny elementy, které již mají hlavní témata přiřazena. Tato funkce má návratovou hodnotu seznam, který obsahuje všechny tyto elementy.

Následně se prostřednictvím cyklu prochází jednotlivé elementy, kdy se daný element bere jako počáteční pro následnou distribuci. Distribuce se provede zavoláním funkce, které je skrze parametr předán aktuálně zpracovávaný uzel a to jako první i druhý parametr funkce. Je to z toho důvodu, že jeden parametr zůstane neměnný (aktuálně zpracovávaný uzel ze seznamu) a druhý se bude při rekurzi nahrazovat rodičovským uzlem. Funkce vyhledá, s využitím funkce *getparent*, z modulu *lxml*, rodičovský element, kterému se témata budou distribuovat. Během distribuce mohou nastat tři různé situace:

- **rodičovský element nemá žádná témata** - V tomto případě se provede pouhé překopírování témat z distribuovaného elementu do rodičovského elementu. Rodičovskému elementu je nastaven příznak (atribut), že svoje témata zdědil a také je mu nastaven atribut, indikující od kolika potomků témata zdědil (v tomto případě je hodnota nastavena na 1). Poté se postupuje rekurzivně dále, kdy je opět zavolána funkce

pro distibuci témat zdola nahoru a první parametr funkce je nahrazen rodičovským elementem pro uzel na aktuální úrovni.

- **rodičovský element již má témata, která však zdědil** - Témata se sloučí. Postupně se tedy prochází odpovídající dvojice témat a jejich procentuální podíl se sčítá. Předem je nutné zajistit, aby elementy obsahující témata obsahovala naprosto všechna témata i v případě, že dané téma vůbec v elementu nefiguruje. V tom případě bude mít nulový procentuální podíl. Ještě je nutné o 1 zvýšit hodnotu atributu indikující počet potomků, od kterých byla témata děděna. Výsledný procentuální podíl daných témat u rodičovského elementu bude vypočten jako aritmetický průměr ze zděděných témat. Tento výpočet však bude proveden až při distribuci shora dolů. Poté se rekurzivně pokračuje dále ke kořeni DOM stromu, stejně jako v předchozím případě.

- **rodičovský element již má témata, která mu byla přiřazena při anotaci** - Rodičovský element je tudíž textový. Kopírování témat nebude provedeno, rodičovský uzel se přeskočí a opět se rekurzivně postupuje v distribuci dále ke kořeni.

Tímto způsobem jsou distribuovány všechny elementy, jenž obdržely svá témata již při anotaci, směrem ke kořeni. Kořenový uzel tak má nyní témata, která postupně zdědil od všech textových elementů, jenž byly anotovány. Po provedení distribuce shora dolů a tedy přepočtu procentuálního podílu jedontlivých témat, bude kořenový element obsahovat témata, jenž jsou shrnutím pro celou webovou stránku, včetně procentuálního podílu.

Shora dolů

Podobně jako v případě distribuce zdola nahoru, zabezpečuje distribuci shora dolů funkce, které je předán jako parametr element, který je aktuálně zpracováván.

Při distribuci shora dolů se postupuje od kořenového elementu DOM stromu, jelikož kvůli předchozí distribuci zdola nahoru je jisté, že kořenový element již témata obsahuje. Tato témata však zdědil. Proto je nejprve nutné přepočítat procentuální podíl jednotlivých témat. Jednotlivá témata se tedy postupně prochází a hodnota jejich procentuálního podílu je vždy vydělena hodnotou atributu definujícího, od kolika potomků daný element témata zdědil. Kontrola, zda aktuální element témata zdědil a případné přepočítání procentuálního podílu u jeho témat se provede vždy na začátku funkce provádějící distribuci témat shora dolů, tedy vždy když je aktualizován zpracovávaný element.

Po přepočítání témat se provede, s využitím funkce *getchildren* z modulu *lxml*, vyhledání všech elementů, jenž jsou potomky aktuálně zpracovávaného elementu (aktuálně je to kořenový element celého DOM stromu) a provede se distribuce shora dolů pro každého potomka.

Během distribuce mohou nastat dvě situace:

- **potomek nemá žádná témata** - Provede se překopírování témat od aktuálního uzlu k potomkovi a je zavolána funkce pro distribuci shora dolů, které je předán jako parametr potomek, do kterého byla distribuována témata a postupuje se tak rekurzivně až k listovým elementům. Tímto je v podstatě prováděn průchod DOM stromem metodou pre-order.

- **potomek již témata má** - Kopírování témat se neprovádí a je tedy pouze zavolána funkce pro distribuci shora dolů, které je předán jako parametr jeden z potomků, stejně jako v předchozím případě.

Po dokončení distribuce shora dolů je již zajištěno, že každý element v rámci DOM stromu má přiřazena témata, ať už mu byla přiřazena přímo anotací, nebo je zdědil od svých potomků či od rodičovského elementu. Proto je možné přistoupit k samotné segmentaci DOM stromu.

5.4.3 Segmentace DOM stromu

Na začátku segmentace se provede vyhledání všech listových elementů zavoláním příslušné funkce. Tato funkce prochází DOM strom metodou pre-order od kořenového elementu a nad každým elementem je zavolána funkce *getchildren* z modulu *lxml*, která pro daný element navrátí seznam jeho potomků. V případě, že je tento seznam prázdný se jedná o listový element, který je přidán do seznamu sloužícího jako návratová hodnota funkce. Takto se projde celý DOM strom. Na konci průchodu tak existuje seznam všech listových elementů, který funkce vrací skrze návratovou hodnotu.

Cyklem je procházen seznam listových elementů a nad každým elementem je zavolána funkce, která pro podstrom DOM stromu, kde se vyskytuje aktuálně zpracovávaný element provede segmentaci.

Segmentace je založena na principu porovnávání témat aktuálního elementu s tématy jeho rodiče. Pro aktuální element se tedy získájí témata jeho rodiče a zavolá se funkce pro porovnání dvojice seznamů témat.

Při porovnání jsou nejprve oba seznamy témat seřazeny a s využitím cyklu a funkce *zip*, se postupně prochází odpovídající dvojice témat. Rozdíl dvojice témat je vypočítán jako absolutní hodnota rozdílu jejich procentuálního podílu. Vypočítané rozdíly z jednotlivých dvojic se postupně sčítají.

Jakmile jsou projity oba seznamy, známe celkový rozdíl všech témat. Aby tato hodnota dávala smysl, je nutné ji podělit konstantou 2, z důvodu přepočtu na rozdíl v procentech (jestliže jeden seznam témat bude značit, že odpovídající textový blok je na 100% o historii, zatímco druhý seznam na 100% o filmu, celkový rozdíl bude roven číslu 200, podělením konstantou 2 dostaneme reálnou hodnotu rozdílu, tedy 100%).

Po zjištění rozdílu témat aktuálního elementu s rodičovským je hodnota rozdílu porovnána s hodnotou, jež byla zadána jako parametr přikazového řádku při spuštění skriptu. Jestliže rozdíl překročí zadanou hodnotu (jinak řečeno se témata elementu s rodičovským elementem liší více než bylo nastaveno), označí se aktuální element jako hraniční (má být segmentován) a provede se přiřazení atributu *top_segment*, který tento fakt indikuje. Zadaná hodnota tak značně ovlivňuje granularitu celé segmentace.

Tímto způsobem jsou porovnána témata všech elementů s tématy jejich rodičů a je tedy provedena segmentace celého DOM stromu. Každý element, jenž má být kořenem segmentovaného podstromu, má tak přiřazený atribut *top_segment*.

5.4.4 Výstup

K vytvoření výstupního souboru byl využit velice podobný přístup jako v případě anotace. Nejprve jsou tedy vyhledány, s využitím modulu *lxml*, všechny elementy, jenž mají atribut *top_segment*. Pro tyto elementy je zjištěno jejich *XPATH* a ukládáno do seznamu.

Poté se provede znovuotevření a přečtení vstupního souboru, který je převeden modulem *lxml* do vnitřní struktury odpovídající DOM stromu. Následně se cyklem prochází seznam obsahující *XPATH* hraničních elementů a stejné *XPATH* je vyhledáváno v DOM stromu vstupního souboru. Do odpovídajícího elementu je překopírován jednak atribut *top_segment*, tak i seznam hlavních témat včetně procentuálního ohodnocení.

Aby provedená segmentace byla po otevření segmentovaného HTML dokumentu vizualizována, jsou s využitím modulu *lxml* vloženy do hlavičky HTML dokumentu (prozatím do interní reprezentace DOM stromu) následující řádky kódu:

```
<link rel=''stylesheet'' href=''segmentation.css''>
<script
   src=''https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js''>
</script>
```

Výpis 5.5: Kód vkládaný do hlavičky HTML dokumentu

Dále je vložen do elementu *body* nový element, skrze nějž jsou zobrazována témata jednotlivých elementů. Tento element má následující podobu:

```
<div id=''segmentation_result_div_themes''>
    <span id=''segmentation_result_span_themes''></span>
</div>
```

Výpis 5.6: Element vkládaný do těla HTML dokumentu

Nakonec je provedena ještě poslední úprava, při které je na konec elementu *body* vložen kód, který připojí externí script:

```
<script src=''segmentation.js''></script>
```

Výpis 5.7: Propojení externího scriptu

Výše uvedené řádky kódu, jenž jsou vloženy do výstupního souboru, umožní zobrazení rámečků okolo hraničních elementů a po najetí myší na příslušný element se zobrazí jeho hlavní témata. Ta se zobrazí jako obsah nově vloženého elementu, který je pozicován do horního levého rohu a jeho obsah je dynamicky měněn, na základě témat elementu, na němž se nachází kurzor.

Po provedených úpravách je DOM strom převeden do textové podoby a uložen do výstupního souboru, jenž nese stejný název jako vstupní soubor, doplněný o příponu "_segmented". V rámci zrychlení provádění experimentů se za příponu "_segmented" ještě doplní hodnota proměnné *difference*, s kterou byla segmentace spuštěna. Toto doplnění umožní provést během krátké chvíle spuštění segmentace pro stejný vstupní soubor s různými hodnotami *difference*, aniž by se musel výstupní soubor zálohovat, aby nedošlo k jeho přepsání.

Jestliže byl při spuštení skriptu zadán argument "-x", vytvoří se ještě jeden výstupní, textový soubor, do kterého se zapíší *XPATH* všech hraničních elementů. Tímto je umožněno následné strojové zpracování. Na závěr procesu segmentace je odstraněna "pracovní" verze vstupního souboru.

V příloze A se nachází také návod, na spuštění aplikace tak, aby fungoval výše popsaný způsob vizualizace segmentace.

Kapitola 6

Testování uživatelského rozhraní

V rámci této kapitoly budou popsány testy, jenž byly prováděny za účelem ověření funkčnosti uživatelského rozhraní.

6.1 Způsob testování

K testování bylo přistupováno principem dílčích testů, kdy se během implementace prováděli testy právě implementované části, či menších úseků kódu. S rostoucí velikostí kódu a celé aplikace byly prováděné testy více komplexnější. Testy dílčích částí aplikace byly však prováděny opakovaně, aby se případně odhalily chyby zanesené do aplikace novou částí kódu.

6.2 Příklady testů

V následující sekci budou uvedeny příklady některých dílčích testů, jako ukázka toho, na co se dané testy zaměřovali.

6.2.1 Vstupní argumenty

Jelikož byla, v rámci této práce, vytvořena konzolová aplikaci, které jsou požadované argumenty předávány skrze příkazový řádek, bylo nutné vyzkoušet různé variace chybně zadaných vstupních argumentů a tím otestovat robustnost části aplikace zajišťující syntaktickou analýzu vstupních argumentů.

Bylo tedy testováno například zadání neexistujícího souboru, ohledně argumentů, jejichž hodnota je číselná bylo testováno, zda aplikace správně ověřuje rozsah možných hodnot a nenastane tak situace, kdy se na vstupu objeví nesmyslné hodnoty. Dále byla testována kontola, zda pro požadovanou akci (anotace, segmentace, kombinace anotace a segmentace) byly zadány všechny požadované argumenty.

6.2.2 Zpracování vstupního souboru

Testy týkající se vstupního souboru zkoumaly, zda se aplikace správně zachová, bude-li například zadán vstupní soubor jenž nelze otevřít (například kvůli právům ke čtení souboru).

Aplikace se zabývá segmentací HTML dokumentu. Proto bylo nutné otestovat chování aplikace v případě, kdy zadaný soubor, HTML dokument, nebude splňovat některá kritéria. Tedy například, že bude obsahovat chybný zdrojový kód a nebude jej tedy možné správně

převét do podoby DOM stromu. V případě spuštění skriptu pouze za účelem segmentace bylo také testováno zadání na vstup HTML dokumentu, jenž splňuje všechny požadavky na HTML dokument, ale doposud nebyl anotován.

Víceméně odpovídající testy byly prováděny jednak pro část určenou pro anotaci, tak pro segmentační část a také pro část provádějící anotaci i segmentaci. Jelikož ve všech třech případech musí vstupní soubor splňovat jistá kritéria, která jsou velice podobná.

6.2.3 Anotace

Testování anotační části se zaměřovalo na ověření chování aplikace v několika situacích, které by mohli nastat před nebo při anotaci.

Příkladem lze úvest pokus o navázání komunikace se Spotlight, který není spuštěn, případně komunikuje na jiném portu, než je nastaveno. Dále pokus o komunikaci se SPARQL serverem, jenž neodpovídá. Ohledně anotace byly také testovány případy, kdy Spotlight i SPARQL server komunikuje, ale buď Spotlight nedokáže vstpní text anotovat a tudíž vrací prádnou odpověď a nebo SPARQL server nedokáže vyhledat pro daný zdroj odpovídající téma do nastavené úrovně zanoření.

Právě toto testování odhalilo problémy vzniklé s neomezením úrovně zanoření a tudíž nekontrolovatelným větvením vytvářené hierarchie.

V poslední řadě těchto testů bylo ověřováno chování v situaci, kdy je vytvářen výstupní soubor do adresáře, pro který nemá skript práva zápisu a tudíž výstupní soubor nelze vytvořit.

6.2.4 Segmentace

V rámci testů segmentační části bylo využíváno z větší části stejných testů jako bylo popsáno výše. Tedy testy týkající se vstupního souboru z hlediska existence, formátu a struktury.

Navíc byl testován případ, kdy atribut definující, že pro vstupní soubor již proběhla anotace byl vložen ručně a vstupní soubor se tak tvářil jako anotovaný, přičemž nebyl. Jelikož v rámci aplikace je ale tento případ ošetřen dvěma způsoby, jednak je testován daný atribut a také je ověřována existence alespoň jednoho elementu, jenž má přiřazená témata, prošel tento test úspěšně.

Závěrem bylo, stejně jako v případě anotace, testováno vytvoření výstupního souboru do adresáře, kam nemá skript právo zápisu.

6.2.5 Stažení webové stránky

Během testů funkce, jenž provádí stažení webové stránky na základě URL, byla objevena řada problému. Jako hlavní jmenujme ten, kdy se při zadávání cesty k adresáři do dialogového okna pro stažení příslušné stránky bralo v úvahu aktuální rozložení klávesnice. Jednotlivé znaky byly do okna totiž zadávány, jako kdyby byly opravdu stisknuty na klávesnici, tudíž například se při českém rozložení klávesnice znak ”z” zobrazoval jako znak ”y” při anglickém rozložení klavesnice. Z tohoto důvodu byla provedena změna zadávání cesty a byl proveden přechod na simulaci stisknutí klávesové zkratky ”ctrl + v”. Což ale způsobovalo další problémy, které však byly vyřešeny instalací utility *XSEL*.

6.2.6 Vizualizace

Při provádění testů části provádějící vizualizaci segmentovaného HTML dokumentu bylo zjištěno, že původní plán vytvoření pouze CSS souboru, v kterém bude využíváno selektoru a funkce ”hover”, je nedostatečný. Ukázalo se, že se styly propagují také do rodičovských elementů. Proto bylo nutné přejít na kombinaci CSS a JavaScriptu.

6.3 Shrnutí

Výše uvedené příklady testů byly použity i v případě spuštění skriptu pro anotaci a segmentaci zároveň. Jelikož je tento případ implementován tak, že se postupně zavolá funkce pro anotaci a poté funkce pro segmentaci, šlo pouze o formalitu, jelikož po provedení testů jednotlivých částí bylo očekáváno bezproblémové chování i v tomto případě.

Jednotlivé testy byly různě modifikovány, aby byly odhaleny veškeré nedostatky týkající se implementace aplikace a aby případné chyby v průběhu běhu aplikace nezpůsobily pád aplikace, nýbrž aby byl uživatel informován o nastalém problému a aplikace byla řádně ukončena.

Kapitola 7

Experimenty a jejich vyhodnocení

V následujícím textu bude představena metoda provádění experimentů s aplikací implementující výše popsanou metodu a na základě které lze vyhodnotit provedené experimenty. Následně budou uvedeny experimenty na některých vybraných webových stránkách, včetně podrobnější analýzy.

7.1 Princip metody pro provádění experimentů

Obecný princip metody, na základě které byly prováděny experimenty a jejich následné vyhodnocení, je založen na porovnávání výstupu aplikace s referenčním výsledkem, korekci vstupních parametrů, s kterými je aplikace spouštěna a opětovného porovnávání s referenčním výsledkem.

7.1.1 Referenční výsledek - Ground truth

Referenčním výsledkem, neboli "Ground truth" (dále v textu bude označován zkratkou GT) je termín využívaný v mnoha oborech a označuje v podstatě vzorový výstup, s jehož využitím se provádí vyhodnocování různých algoritmů, metod či aplikací.

GT může být například jiná metoda, která je brána jako tzv. "zlatý standard" výstupy takové metody jsou brány jako vzorky GT. Další možností jak získat GT pro vyžadované účely je ho vytvořit ručně.

Jestliže se zamyslíme nad principem této práce, tedy návrh a implementace metody pro segmentaci webové stránky na základě sémantiky jednotlivých elementů, a to konkrétně nad tím, že webová stránka je segmentována z hlediska sémantiky, tedy významu obsahu jednotlivých elementů, může být zhodnocení výsledku segmentace poměrně subjektivní. Jelikož obsah některého článku se jednomu jedinci může jevit tak, že je čistě o politice, zatímco druhý může tvrdit, že je o historii. A přitom mohou mít pravdu oba, protože se článek může zabývat politickými událostmi v historii.

Fakt, že se do jisté míry dá výsledek navržené metody posuzovat subjektivně, vedl k závěru, že GT pro webovou stránku, na které budou prováděny experimenty bude vytvářen ručně. Tedy daná webová stránka bude segmentována ručně, dle lidského uvážení a výsledek této segmentace bude považovát za GT, s kterým bude porovnáván výsledek segmentace, jenž vznikl jako výstup implementované aplikace.

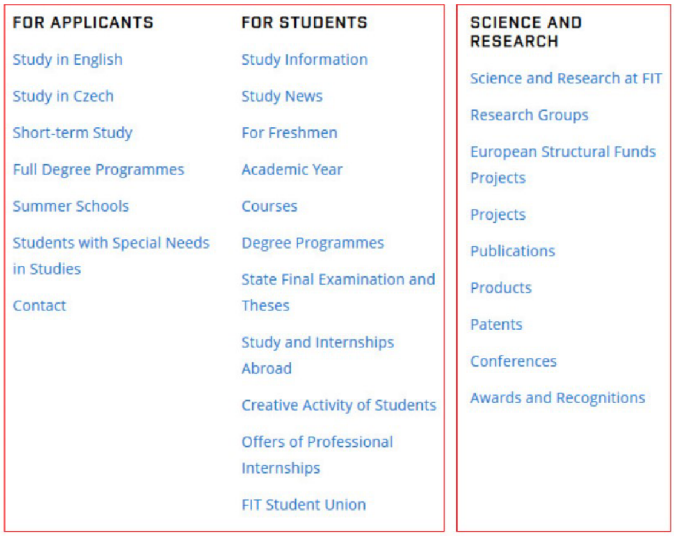
Pravděpodobně nejkorektnější cesta, pro vytvoření GT pro konkrétní webovou stránku, by byla vytvořit větší skupinu lidí na základě jistých aspektů, jako je například různý věk jedinců, různé úrovně vzdělání atd., a nechat tuto skupinu lidí provést ruční segmentaci

dané stránky. Z dosažených výsledků by byl vytvořen průměrný výsledek a ten by byl poté považován jako GT pro danou stránku. Nicméně pro účely této práce bylo vytvoření GT pro testované stránky ponecháno na uvážení autora práce.

Výstup segmentace provedené implementovanou aplikací vypadá jako původní webová stránka, která má červený rámeček okolo elementů, jenž jsou hraniční. Pro vytvoření GT dané stránky byl využit grafický program, skrze který byl editován screenshot webové stránky stránky tak, že do něj byly přidány červené rámečky okolo elementů, jenž by dle uvážení měly být segmentovány.

Při ruční segmentaci bylo postupováno tzv. od nejhrubší granularity. Nejprve byla tedy webová stránka (její screenshot) rozdělena na větší oblasti jako je například hlavička obsahující menu dané stránky, patička s informacemi o copyrightu, oblast obsahujicí prvky pro přihlášení a registraci, či hlavní část dané stránky s relevantním obsahem. Následně byla stránka segmentována jemněji. Například pokud relevantní část stránky obsahovala několik bloků, které spolu nijak nesouvisely a byly například odděleny i vizuálními oddělovači či barvou pozadí, byly tyto bloky segmentovány. V poslední řadě proběhla nejjemnější segmentace, kdy se posuzovala i sémantika jednotlivých bloků či elementů.

Poslední fáze ruční segmentace, tedy fáze, kdy se zohledňovala sémantika obsahu, byla nejnáročnější, zejména z časového hlediska. Byla brána v úvahu jednak sémantika konkrétního elementu, ale i sémantika okolních elementů a tudíž sémantika celého bloku, ve kterém se daný element nacházel. Jako příklad lze uvést menu v oblasti zápatí jedné z testovaných stránek, konkrétně úvodní stránku webu Fakulty informačních technologií v Brně. Pro ilustraci je přiložen obrázek 7.1.



Obrázek 7.1: Ukázka ruční segmentace menu na webu https://www.fit.vut.cz/.en

Na obrázku obrázek 7.1 lze vidět, že jednotlivé bloky menu byly zachovány jako celky a dokonce dva bloky, "For applicants" a "For students", byly sloučeny, jelikož bylo posouzeno, že oba bloky se tykají zejména studia a informací o něm. Blok "Science and research" je oddělen z toho důvodu, že pokud na tento blok budeme nahlížet samostatně, lze prohlásit, že z hlediska sémantiky je odlišný od ostatních. Jak i sám název tohoto bloku napovídá, jedná se sématnticky spíše o téma věda a výzkum, tudíž proběhlo segmentování daného bloku. Pokud bychom však nahlíželi samostatně na všechny jednotlivé elementy obsažené

v bloku "Science and research", pravděpodobně bychom provedli jemnější segmentaci. Jako příklad uveďme element s textem "Publications". Tento element by z hlediska sémantiky šlo vyhodnotit tak, že bychom mu spíše přiřadili téma "literatura" a tudíž by byl segmentován z důvodu, že celý blok je tématicky zaměřen spíše na téma "věda".

Jak však dále v této kapitole bude ukázáno, tak aplikace vytvořená v rámci této práce nahliží na obsah jednotlivých elementů a tudíž v jistých případech provede jemnější segmentaci, než byla provedena v rámci tvorby GT pro danou stránku. Fakt, že je nahlíženo pouze na sémantickou složku elementů povede v některých případech naopak k situaci, že bloky, jenž byly v rámci GT segmentovány z vizuálního důvodu, zůstanou sloučeny, jelikož jejich obsah bude sémanticky velmi podobný. Na tyto skutečnosti i s ukázkami bude poukázáno v dalších částech v rámci této kapitoly.

Po dokončení ruční fáze segmentace, kdy je brán ohled na sémantiku elementů, či bloků elementů, vznikne GT pro danou webovou stránku, na základě kterého bude prováděno porovnávání.

7.1.2 Postup experimentování

Jak již bylo výše zmíněno, je princip provádění experimentů založen na porovnání výstupu aplikace s GT pro danou stránku. V rámci kapitoly 5.4 bylo popsáno, že rozhodování, zda daný element bude segmentován, je prováděno na základě porovnávání s jistou hodnotou, jenž je aplikaci předána při spuštění prostřednictvím příkazového řádku. Tato hodnota významně ovlivňuje granularitu segmentace. Nelze však jednoznačně prohlásit, jaké je optimální nastavení této hodnoty. Optimální nastavení se pro každou webovou stránku totiž může lišit. Dalším aspektem je, že pro každého uživatele může být ideální různá úroveň granularity segmentace a tudíž různá hodnota, jenž je zadávána při spuštění aplikace.

Na základě výše uvedeného faktu byla zvolena metoda, kdy je pro každou testovanou webovou stránku provedena segmentace opakovaně s různým nastavením úrovně granularity (cca 25 různých úrovní granularity) a ze všech výsledků segmentace jsou vybrány ty, u kterých se projevuje zajímavé chování segmentační metody a na které bude dále v textu poukazováno.

V případě porovnávání výsledku s GT mohou v podstatě nastat 3 situace:

- segmentované bloky se shodují - tedy segmentace konkrétního bloku stránky provedená aplikací, odpovídá GT
- aplikace provedla jemnější segmentaci - konkrétní blok v rámci stránky byl aplikací více segmentován ve srovnání s GT
- aplikace provedla hrubší segmentaci - v rámci GT je konkrétní blok segmentován jemněji, tedy je segmentován na více částí, zatímco aplikace daný blok zachovala celý, případně ho segmentovala na méně částí v porovnání s GT

V rámci experimentů a jejich vyhodnocení budou blíže zkoumány zejména poslední dva případy, tedy případy, kdy se segmentace provedená aplikací úplně neshoduje s GT a v některých případech bude poukázáno na důvod, proč tato situace nastala, včetně konrétních případů a ilustrací.

7.2 Experimenty

Nyní budou následovat experimenty na vybraných webových stránkách, kdy bude poukazováno, jak již bylo výše popsáno, na odlišnosti výsledků segmentace aplikací oproti GT. Přímo v textu budou ilustrovány pouze části webové stránky, jenž budou blíže popisovány.

Jelikož je segmentace vizualizována červeným rámečkem přidaným s využitím CSS stylů a JavaScriptu, tak se tyto rámečky pro sousední elementy mohou překrývat nebo v případě rozbalovacích menu se mohou zobrazovat lehce odlišným způsobem oproti GT pro danou webovou stránku. Z tohoto důvodu byly, pro lepší ilustraci, některé ilustrace výstupu segmentace aplikací upraveny grafickým editorem, aby byla segmentace lépe rozpoznatelná. Ale vždy upravený výsledek koresponduje se zdrojovým kódem dané stránky.

7.2.1 Titulní stránka webu Fakulty informačních technologií

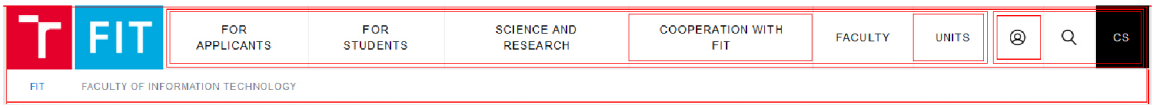
Jako jedna z testovacích webových stránek byla zvolena domovská stránka webu Fakulty informačních technologií dostupná na adrese `https://www.fit.vut.cz/.en`. Z důvodu, že Spotlight využívá slovník v anglickém jazyce, je testovaná stránka přepnuta do anglického jazyka. Ukázka GT a výstupu segmentace aplikací pro celou stránku se nachází v příloze B.

Stránka je strukturována do několika oblastí, které jsou vizuálně odděleny ať už barvou pozadí či horizontálními, respektive vertikálními oddělovači. To usnadnilo práci při tvorbě GT pro tuto stránku, jelikož pro hrubou segmentaci byly využity právě tyto oddělovače. Pro jemnější segmentaci byla již zohledňována sémantika konkrétních elementů, ale s ohledem na celý blok.

Navigační menu a hlavička stránky

Zaměříme-li se hned na úvodní část stránky, kde se nachází navigační menu, které je vytvořené s využitím seznamu, dále logo fakulty, část pro vyhledávání, přihlášení a změnu jazyka, objevíme odlišnosti výstupu aplikace s GT pro tuto stránku. Na obrázku 7.2 je zobrazeno GT pro uvodní část stránky a na obrázku 7.3 je pro porovnání zobrazena segmentace této části vytvořená aplikací.

Obrázek 7.2: GT segmentace úvodní části stránky



Obrázek 7.3: Výstup aplikace pro úvodní části stránky

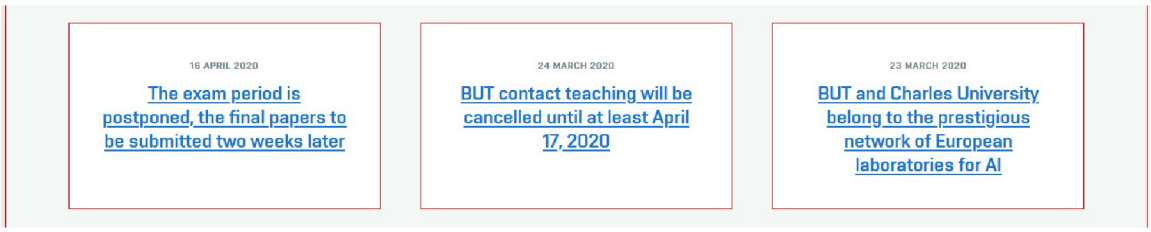
Jak lze při porovnání vidět, tak logo fakulty zůstalo nesegmentováno. Příčinou nesegmentování loga je fakt, že se jedná o obrázek, který však nemá žádný popis. Jedná se tedy sice o samostatný element, který však neobsahuje žádný text, který by mohl být zpracován během anotace. Výsledkem tedy je, že element, do kterého je vložen obrázek dědí témata

od rodičovského elementu a při segmentaci, kdy se provádí porovnání témat elementu s tématy jeho rodiče dochází k výsledku, že tato témata jsou totožná a vnitřní element tak není segmentován. Tento jev se vyskytuje ve všech případech, kdy se na webové stránce vyskytuje obrázek bez slovního popisu.

Další odlišnost je možné vidět při segmentaci navigačního menu. Jednotlivé položky menu obsahují odkazy vedoucí do jednotlivých částí celého webu. Tyto odkazy jsou však zastoupeny heslovitým popisem, tedy text zastupující daný odkaz obsahuje zpravidla 1-3 slova. S ohledem na to, že k jednotlivým elementům je přistupováno individuálně, mohou vzinkat problémy během anotace textu pomocí Spotlight. Ten v některých případech buď vůbec nedokáže dané slovo anotovat a nebo ho anotuje tak, že následným výpočtem témat pro daný element (v tomto případě pro dané slovo) je mu přiřazen úplně jiný sémantický význam, než bychom očekávali. To může mít za následek, že je chybně přiřazený sémantický význam pro celé sub-menu. Jestliže se provede chybná anotace a elementům jsou přiřazena témata, avšak ne úplně správná, má to ve většině případů za následek jemnější segmentací (sub-menu má odlišné témata oproti ostatním sub-menu v rámci navigačního menu). Pokud však položky sub-menu nejsou anotovány vůbec, tak témata pro dané sub-menu jsou přiřazena dědičností od rodičovského elementu. To má za následek nesegmentování elementu, což v některých případech může vyhovovat a v jiných nikoliv.

Oblast stránky obsahující články o fakultě

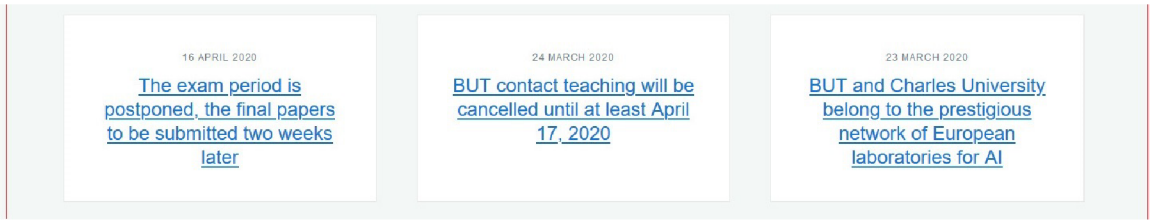
Na obrázcích 7.4 a 7.5 vidíme segmentovaný úsek prostřední části webové stránky, kde se nachází novinky týkající se fakulty, nejprve GT a následně výsledek segmentace aplikací. Zde si lze povšimnout, že v rámci vytváření GT byly jednotlivé články segmentovány. Hlavní důvod pro segmentaci je fakt, že články jsou vzájemně vizuálně odděleny a proto jsou vnímany samostatně. Jak již však bylo mnohokrát zmiňováno, tak metoda, jenž byla navržena v rámci této práce, vůbec nebere v potaz vizuální vlastnosti dané stránky, ale zaměřuje se pouze na sémantiku. Po anotaci s využitím Spotlight, byla jednotlivým blokům (článkům) přiřazena témata, jenž spolu vzájemně souvisejí, zejména šlo o téma "vzdělání" a "technologie". Na základě těchto témat rozhodl algoritmus pro segmentaci, že tyto elementy nebude segmentovat. Z hlediska segmentace jako takové, by se zřejmě jednalo o chybu, jelikož vizuálně jsou bloky odděleny, ale z hlediska sémantiky jsou si bloky velice blízké a tedy neprovedení segmentace lze považovat za správné.



Obrázek 7.4: GT segmentace části novinek o fakultě

Navigační menu ve spodní části stránky

V dolní části stránky se nachází další navigační menu, které je více méně totožné s navigačním menu v úvodu stránky, s rozdílem, že menu v horní části je rolovací a jednotlivé



Obrázek 7.5: Výstup aplikace pro část novinek o fakultě

položky sub-menu jsou tak skryty. Zde si však blíže, na konkrétním příkladu, přiblížíme problém, kdy jsou elementy segmentovány příliš jemně z důvodu délky textu.

Obrázky 7.6 a 7.7 vyobrazují část segmentovaného sub-menu (GT vlevo a výstup aplikace vpravo). Zatímco v GT jsou jednotlivé položky sub-menu nesegmentovány, jelikož bylo bráno v úvahu umístění v rámci bloku (sub-menu), aplikace provedla segmentaci některých dílčích elementů sub-menu. Důvod je stejný jako již bylo zmiňováno výše, tedy ten, že jednotlivé elementy jsou zpracovávány samostatně. Nicméně nyní si ukážeme konkrétní příklady.

Obrázek 7.6: GT segmentace submenu

Obrázek 7.7: Výstup aplikace pro submenu

Vezmněme v úvahu například položku s názvem "Products". Budeme-li na ni nahlížet jako na součást celku, pravděpodobně určíme, že se jedná o *produkty*, které vznikly v rámci výzkumů na fakultě a tudíž bychom prohlásili, že z hlediska témat se bude jednat o *vědu*, *techniku*, případně *vzdělání*. Spotlight však jako vstup obdrží pouze slovo "Products" a pracuje pouze s ním. To má za následek, že je na slovo "Products" nahlíženo jako na předmět obchodu a prodeje a tudíž největší procentuální podíl na celkovém tématu má téma *ekonomika*.

Méně patrné může být vyhodnocení položky s názvem "Patents". Patenty jsou logicky opět výstupem vědeckého výzkumu a tudíž bychom slovu "Patents" pravděpodobně přiřadili stejná témata jako v případě slova "Products" a tentokrát i v případě, že bychom stejně jako Spotlight nebrali v úvahu ostatní položky menu. Proč tedy proběhla segmentace tohoto elementu? Vyhledáme-li si termín "Patent" v anglické verzi Wikipedie (Spotlight pracuje se

slovníkem DBpedie, která vznikla na základech Wikipedie, jak již bylo popsáno výše v části 2.6) uvidíme, že velká část stránky je tvořena historií, kde se objevuje velké množství termínu, spojených se zákony. To má za následek, že Spotlight určí jako hlavní téma pro slovo "Patents" téma *politika*. Tudíž je element s tímto názvem určen jako tématicky odlišný a je segmentován. Obdobně je to i s ostatními položkami tohoto sub-menu, ale i s položkami všech ostatních sub-menu.

Zbývající část stránky

Ve zbývajících částech této webové stránky se vyskytují odlišnosti, které odpovídají těm, které již zde byly popsány a tudíž už na ně nebude znovu poukazováno.

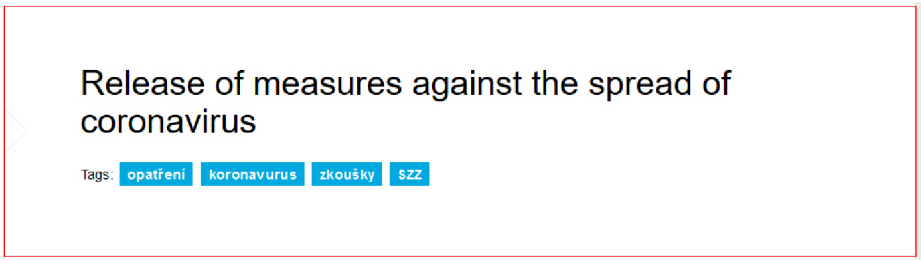
7.2.2 Zpráva na webu Fakulty informačních technologií

Předchozí experiment se zabýval titulní stránkou webu Fakulty informačních technologií. V rámci tohoto webu bylo prováděno hned několik experimentů, kdy byly využity stránky obsahující tiskové zprávy, informace o fakultě a další. Nicméně ve většině experimentů byla zjištěna stejná fakta, jako v předchozím experimentu. Je tomu tak z důvodu, že fakultní stránky mají vždy stejnou strukturu. V úvodní části stránky se nachází navigační menu a v zápatí se nachází stejné navigační menu s rozdílem, že se nejedná o rozbalovací menu. Prostřední, obsahová čast se již poté mění v závislosti na aktuální stránce.

Nyní bude představen ještě jeden experiment v rámci fakultního webu, kdy pro experimentování byla vybrána stránka obshaující zprávu s titulkem "Uvolnění opatření proti šíření koronaviru". Tato stránka nebyla vybrána z důvodu tématu, ale byla náhodně vybrána ze stránek, které obsahují souvislý text, který je však členěn pomocí odstavců, tudíž se jednotlivé části textu nachází v oddělených elementech.

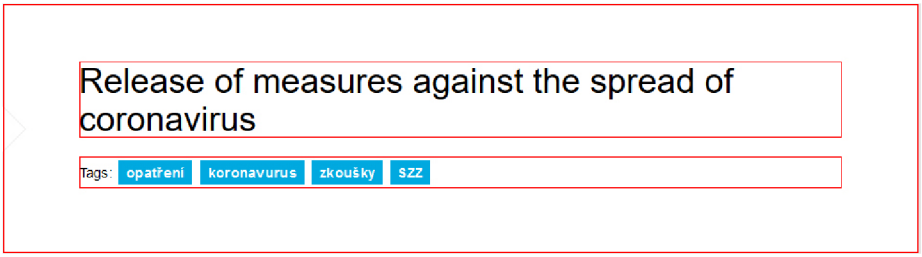
Předmětem zkoumání bude nyní pouze obsahová část stránky, jelikož ostatní části by byly totožné s předchozím experimentem.

Nejprve se zaměříme na část obsahující nadpis daného článku. Na obrázku 7.8 vidíme GT pro tuto část stránky, zatímco na obrázku 7.9 je výstup segmentace provedené aplikací.



Obrázek 7.8: GT segmentace nadpisu článku

Jak lze vidět aplikace provedla i segmentaci elementu, jenž obsahuje přímo text nadpisu a elementu obsahujícího klíčová slova. Důvod, proč tomu tak je, vychází z faktu, že rodičovské elementy, jenž neobsahují žádný text a tudíž nemohou mít přiřazena anotací svá vlastní témata, dědí témata od svých potomků. Následně se procentuální podíl zděděných témat dělí počtem potomků, od kterých rodič dědil. I přesto, že element obsahující klíčová slova článku, obshauje některá slova v českém jazyce, přiřadil Spotlight tomuto elementu zdroj a následně i témata, která jsou však z jistého důvodu odlišná od témat, která byla přiřazena elementu obsahujícího text nadpisu. To má za následek, že celému bloku obsahu-



Obrázek 7.9: Výstup aplikace pro část nadpisu článku

jícího nadpis a také klíčová slova jsou přiřazena témata, která se z 50% shodují s tématy nadpisu a z 50% s tématy klíčových slov. Pro účely tohoto experimentu však byla nastavena granularita, na jejímž základě se provede segmentace elementů na hranici nižší jak 50%. Pokud bychom zvýšili hrubost granularity a nastavili tedy vyšší hodnotu, odpovídal by výstup aplikace očekávanému výstupu. Nicméně pro popis chování algoritmu v jistých momentech, byla úmyslně nastavena jemnější granularita.

Zaměříme-li se nyní na oblast, kde se nachází již samotný text článku vypadá výstup aplikace naprosto stejně jako v případě GT. Tedy i přesto, že text byl rozdělen do odstavců, které jsou z pohledu aplikace samostatné textové bloky a jsou tudíž zprácovávány samostatně. Tématicky byly totiž jednotlivé odstavce vyhodnoceny jako velice podobné a proto nebyl důvod k segmentaci. Za pozornost však stojí zápatí tohoto článku, kde jsou informace o autorovi a časové razítko identifikující čas a datum poslední úpravy. Na obrázku 7.10 vidíme GT pro zápatí článku a na obrázku 7.11 výstup aplikace.

Obrázek 7.10: GT segmentace zápatí článku

Obrázek 7.11: Výstup aplikace pro část zápatí článku

Opět vidíme, že aplikace provedla jemnější segmentaci, než bylo očekáváno. Při zkoumání důvodu této segmentace bylo zjištěno, že Spotlight se snaží anotovat také časová razítka a vyhledat pro ně vhodný zdroj. A to takovým způsobem, že se nejprve pokouší anotovat celé časové razítko a v případě neuspěchu se pokusí anotovat pouze některé části, například pouze rok. To má za následek, že pro časová razítka, tedy i pro elementy ve kterých se objevuje nějaké datum, přiřadí Spotlight nějaký zdroj a na jeho základě jsou poté danému elementu s časovým razítkem přiřazena témata. To však může způsobovat problémy při segmentaci, jelikož celý článek může mít hlavní téma *věda*, zatímco na základě elementu s časovým razítkem může být přiřazeno zcela odlišné hlavní téma, například *politika*. Vliv těchto faktorů může mít v některých případech podíl na neočekávané segmentaci jednak daného elementu s časovým razítkem, ale také elementů, majících stejný rodičovský element.

Jelikož určená témata pro konkrétní element ovlivňují témata rodičovských elementů, jak již bylo popsáno výše.

Důvod proč byla také provedena segmentace u elementu obsahujícího jméno autora má společný základ s problémem týkajícího se časových razítek. Tedy fakt, že Spotlight se snaží anotovat části dílčí části textu. V tomto případě, kde se v textu nachází jméno *Jaroslav Dytrych* včetně jeho titulů, se Spotlight zaměří právě na zkratky titulů. Výsledkem je, že nejvetší podíl pro tento element má téma *vzdělání*. Téma vzdělání se však objevuje i v rámci celého článku, tedy v rámci elementu, jenž má společného rodiče s elementem obsahujícím jméno autora. Proto téma *vzdělání* má jistý podíl v rodičovském elementu. Pokud bychom tedy nastavili granularitu na hrubší hodnotu, element se jménem autora by nebyl segmentován.

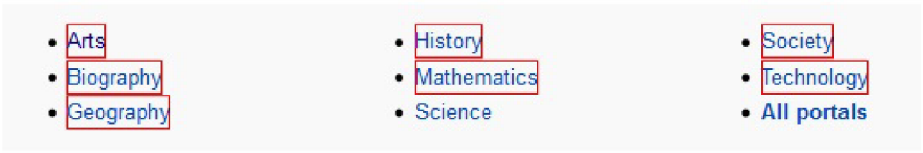
7.2.3 Titulní strana anglické verze webu wikipedia.org

Tato stránka byla vybrána zejména z důvodu, že obsahuje větší množství tématicky odlišných bloků a také je stránka přehledně členěna s využitím vizuálních oddělovačů, což usnadňuje tvorbu GT pro následné porovnávání.

Navigační menu v horní části stránky

V úvodu stránky se nachází menu, které obsahuje odkazy, které směřují do dalších částí webu na základě tématu. Na obrázku 7.12 můžeme vidět GT segmentaci pro toto menu, zatímco na obrázku 7.13 výstup segmentace provedené aplikací.

Obrázek 7.12: GT segmentace menu



Obrázek 7.13: Výstup aplikace pro část menu

Prvním rozdílem, který je ihned patrný, je neprovedená segmentace u položek ”Science” a ”All portals”. Nesegmentování položky ”Science” má poměrně překvapivý důvod. I přesto, že ”Science” je přímo jedno z hlavních témat, tak Spotlight nedokázal toto slovo anotovat a tudíž element obsahující slovo ”Science” zdědil témata od svého rodičovského elementu a tudíž při porovnání, na jehož základě se rozhoduje o segmentaci elementu nebyl žádný rozdíl v tématech tohoto elementu s tématy jeho rodiče. To mělo za následek rozhodnutí, že element nebude dále segmentován.

Elementu, jenž je rodičem elementů na obrázku byla témata přidělena na základě průměrné hodnoty. Témata rodiče ovlivňujíi další jeho potomci, kteří se nenachází na obrázku. Tento fakt zapříčinil, že element s textem ”All portals” není segmentován. Spotlight tomuto

elementu sice přiřadil jistá témata, která jsou však velice podobná tématům rodiče a proto nebyla provedena segmentace.

Pokud bychom nastavili jemnější granularitu při segmentaci této stránky, provedla by se segmentaci i elementu s textem "All portals". Nicméně element s textem "Science" by stále zůstal nesegmentován, jelikož jak už bylo popsáno výše jsou jeho témata totožná s jeho rodičem.

Rozdíl v podobě rámečků okolo elementů je zapříčiněn strukturou HTML kódu. GT byl vytvářen grafickým editorem, zatímco aplikace pracuje přímo se zdrojovým kódem dané stránky. Toto menu je vytvořeno jako nečíslovaný seznam, kde se odrážky přidávají implicitně. Daný element je však tvořen pouze odkazem, který je zastoupen zobrazovaným textem. Proto kdybychom dodržovali skutečnou velikost elementu při vytváření GT, byla by podoba jednotlivých rámečků totožná s výstupem aplikace.

Tabulka s vedlejšími projekty Wikipedie

Ve spodní části stránky se nachází sekce, kde jsou odkazy na sesterské projekty wikipedie s krátkým popiskem. Originální podobu této tabulky lze vidět na obrázku 7.14.

Obrázek 7.14: Originální podoba tabulky sesterských projektů Wikipedie

Na tuto sekci je zde poukazováno zejména kvůli obtížnosti rozhodnutí, zda v rámci vytváření GT provést segmentaci jednotlivých elementů. Z jednoho úhlu pohledu by se teoreticky mohli elementy nechat nesegmentované, jelikož obsah elementů je zastřešen jedním společným prvkem a tím je Wikipedia. Tím druhým pohledem je zohlednění faktu, že jednotlivé elementy jsou od sebe výrazně odděleny mezerou a tudíž se vizuálně jeví jako samostatné prvky.

Nakonec bylo zohledněno vizuální oddělení jednotlivých prvků a tudíž v rámci GT jsou elementy segmentovány samostatně. Implementovaná metoda se však vizuální stránkou vůbec nezabývá. Bylo tedy otázkou jaký bude výstup aplikace.

Jak již bylo výše popsáno, tak seznam témat je pevně daný. Pokud by se v tomto seznamu nacházelo téma *Wikipedia*, je možné, že by elementy zůstaly nesegmentovány. Prvním faktem, který ovlivňuje výsledek segmentace je právě omezenost seznamu témat. Dalším faktem je, že text je zpracováván strojově s využitím Spotlight a tudíž je určení témat silně závislé na úspěšnosti a přesnoti Spotlight. Kombinace těchto faktů zapříčiní, že pro jednotlivé elementy jsou určena témata, která jsou si do jisté míry podobná (ve většině elementů figuruje téma "Technologie"). V jistých případech však některá témata zcela vynikají, například pro element s názvem "Wikiversity" bylo přiřazeno téma "Education" a to s podílem 99%, což lze v tomto případě považovat za správné. Tyto výkyvy velice ovlivňují průměrnou hodnotu témat rodičovského elementu, což má za následek ovlivnění celé segmentace. Ten je v tomto případě takový, že jednotlivé elementy jsou segmentovány samostatně a tudíž výsledek odpovídá GT.

Zbývající část stránky

Ve zbývajících částexh stránky, které nebyly výše zmíněny se nachází například postraní menu, kde lze přepínat zobrazení stránky v různých jazykových verzích, nebo zobrazit různé informace a nástroje a také zobrazit sesterské projekty Wikipedi. Názvy položek v tomto menu jsou zpravidla jednoslovné a chování segmentace je tak velice podobné tomu, jenž bylo popsáno v prvním experimentu v části 7.2.1, kde se segmentovalo menu na stránce. Tedy nastavíme-li jemnější granularitu, je většina položek menu segmentována samostaně, zatímco při vyšší granularitě jsou segmentována pouze sub-menu a jejich položky jsou v rámci jednoho segmentu.

Dále se na stránce nachází jednotlivé úseky článků a zajímavostí souvisejících s aktuálním datem. V těchto případech však segmentace provedená aplikací odpovídá očekávanému výstupu a proto zde nebudou další příklady uváděny.

7.2.4 Stránky o konkrétních tématech na webu Wikipedia

Kromě titulní stránky Wikipedie, byly prováděny experimenty na stránkách Wikipedie, které však jsou o konkrétním tématu, například témata jako "Česká republika", "Druhá světová válka", "Mona Lisa" a mnoho dalších.

Experimenty na těchto stránkách však neodhalili nijak zvláštní chování segmentace. Avšak jedna věc, na kterou lze poukázat jsou nadpisy jednotlivých odstavců.

Jelikož nadpisy jsou zpravidla tvořeny jako blokové elementy, které se nachází před odstavcem, ke kterému naleží, jsou z pohledu DOM stromu brány jako samostatné elementy, které s odstavcem obsahujícím příslušný text pouze sousedí. Zpracování nadpisů je tak prováděno samostatně. Nadpisu tak může být při anotaci, pomocí Spotlight přiřazeno například pouze jediné téma (záleží na délce nadpisu, avšak u článků na Wikipedii se jedná zpravidla o krátké nadpisy typu "Historie", "Turismus", "Vznik" a podobně). Odstavci, ke kterému napdis patří, bývá ve většině případů přiřazeno více témat a tudíž se podíl jednotlivých témat rozloží (platí hlavně v případech kdy je odstavec tvořen alespoň jednou větou). To má za následek, že v drtivé většině případů, je nadpis vždy segmentován samostatně.

Nadpisy na těchto stránkách však bývají také odděleny vizuálně, například horizontální čárou. Tudíž z hlediska segmentace, kdy jsou brány vizuální prvky v potaz, lze segmentaci nadpisů, tak jak ji provede aplikace, považovat za správné.

7.3 Shrnutí experimentů

Výše byly popsány experimenty, během kterých se projevily jisté odlišnosti v segmentaci oproti GT. Na základě těchto odlišností byly zkoumány důvody, kvůli kterým dané chování nastává. Pro každou testovanou webovou stránku bylo připraveno množství výstupních souborů s různou úrovní granularity segmentace, což usnadnilo samotné provádění experimentů, jelikož bylo možné porovnávat GT pro danou stránku hned s několika výstupními soubory, s různou granularitou naráz.

Během experimentování byla prováděna celá řada experimentů na různých webových stránkách. Ve většině případů byly však odchylky oproti GT způsobeny stejnými aspekty, jako v případech uvedených výše, proto zde již nebyly uvedeny.

Pokud bychom se pokusily shrnout výsledky provedených experimentů, mohli bychom prohlásit, že ve většině testovaných případů odpovídají výstupy segmentace očekávanému

výsledku. Je však nutné najít optimální úroveň granularity segmentace, při které dosahuje segmentační metoda požadovaného výstupu.

Nicméně byly také experimentální metodou objeveny případy, projevující se na několika, či na všech, testovaných webových stránkách, při kterých se výstupy segmentační metody odlišují od očekávaného výstupu.

Nejčastějším problémem je situace, kdy se v elementu nachází velice krátký text, který má být anotován (například pouze jediné slovo). V takovém případě nastávají 3 různé situace:

- Spotlight úspěšně anotuje daný text a správně mu přiřadí zdroj. Tato situace je nejideálnější, jelikož výsledek odpovídá očekávání. Nastává hlavně v případech , kdy text obsahuje klíčová slova jenž jsou jednoznačná.
- Spotlight úspěšně anotuje text, ale přiřadí mu chybný zdroj. Tato situace způsobí, že jsou danému textu přiřazena témata, jenž neodpovídají očekávaným tématům a tato témata ovlivňují jednak témata rodičovského elementu, tak i výsledek celé segmentace.
- Spotlight nedokáže daný text anotovat. Důsledkem je, že příslušný element zdědí témata od svého rodiče a tudíž není segmentován, jelikož má s rodičem totožná témata.

Dalším prvkem, jenž může v některých případech způsobovat neočekáváné chování segmentace úzce souvisí s výše popsaným problémem týkajícím se krátkého textu. Tímto prvkem jsou nadpisy. Pokud je nadpis tvořen krátkým textem, jedná se o stejný případ jako v případě předchozího problému.

Pro nadpisy také záleží na struktuře zdrojového kódu a tedy podobě DOM stromu. Jestliže je zdrojový kód strukturován tak, že pro každý blok textu, například článek, existuje samostatný element, ve kterém jsou již jednotlivé ostavce a také nadpis, jsou témata nadpisu porovnávána především s tématy daného bloku textu, ke kterému nadpis náleží. Tudíž pokud jsou správně určena témata pro nadpis i příslušný text, nebude provedena segmentace nadpisu.

Jakmile je však zdrojový kód strukturován tak, že jednotlivé bloky (články) nejsou v samostatných elementech, ale pouze se střídají odstavce a nadpisy a tedy všechny odstavce i nadpisy jsou z hlediska DOM stromu sourozenecké elementy (takto jsou například strukturovány články na Wikipedii), provádí se v podstatě porovnávání témat nadpisu s tématy všech ostatních nadpisů i s tématy všech ostatních odstavců. Tento aspekt se však netýká pouze nadpisů ale v podstatě všech ostatních elementů.

Problémy mohou také způsobovat elementy obsahující časové údaje jako je poslední modifikace článku, datum vytvoření článku aktuální datum a čas, či jiné časové údaje, které však nijak nesouvisí s obsahem textu. Spotlight se pokouší anotovat i jednotlivé části časového údaje, například pouze rok. To způsobuje přiřazení témat elementu, jenž by teoreticky žádná témata mít neměl. Na základě toho mohou být negativně ovlivněna témata rodičovského elementu a tudíž může být chybně segmentován některý z dalších elementů, jenž sousedí s časovým údajem.

Podstatným faktem je, že se stále jedná o metodu, jenž vychází z metody segmentace založené na analyzování DOM stromu a s tím si nese i některé z problémů této metody. Bude-li webová stránka nevhodně strukturována nebo některé elementy budou použity pouze pro rozvržení elementů na stránce bude docházet k chybám segmentace. Můžeme vycházet z příkladu, jenž byl uveden v teoretické části této práce.

Podíváme-li se na obrázek 3.1, na kterém byl popisován problém metody založené na analýze DOM stromu, kdy nebude správně provedená segmentace, jelikož je pro rozložení

elementů na stránce použita tabulky a představíme si, že oblasti '3' a '4' mají z hlediska sémantiky totožná témata, tak i přesto nemusí být tyto oblasti sloučeny ani s použitím navržené metody. Pokud totiž oblasti '1' a '2' budou mít zcela odlišná témata oproti oblastem '3' a '4', dojde při jisté míře granularity k segmentování všech jednotlivých oblastí, tedy oblastí '1', '2', '3' a '4', ale může také dojít k segmentaci celých řádků.

Ne vždy však tento problém nastane. V některých případech dojde alespoň k částečné eliminaci tohoto problému. Záleží však na odlišnosti jednotlivých témat. Jestliže nebudou témata oblastí '1', '2' zcela odlišná oproti tématům '3' a '4', nebo pokud nastavíme hrubší granularitu, mohou být navrženou metodou oblasti '3' a '4' sloučeny.

Během provádění experimentů byl také vypozorován vliv obsahu zpracovávaného textu na délku provádění anotace. Jestliže budou zpracovávány stejně rozsáhlé textové bloky, ale v jednom z textů se bude nacházet velké množství klíčových slov, zatímco v druhém bude text obecnějšího charakteru, bude se blok obsahující klíčová slova zpracovávát déle. Tento rozdíl vzniká hlavně proto, že Spotlight v obecnějším textu anotuje mnohem méně částí, než v rámci textu s velkým množstvím klíčových slov. Během samotné anotace pomocí Spotlight nevznikne takový časový rozdíl, ten vznikne až při určování témat, kdy je prováděna komunikace se SPARQL serverem, který vyhodnocuje dotazy a pro každou anotovanou část textu je sestavováno hierarchické uspořádání témat. Pokud je tedy větší rozdíl v počtu anotovaných částí, bude časový rozdíl při určování témat velice znatelný.

Experimenty odhalily jisté nedostatky navržené metody, které vznikají zejména při určování témat pro textové elementy ať už z důvodu délky textu, nebo z některých dalších důvodů které byly uvedeny výše. Bylo však také zjištěno, že při jistých podmínkách dochází alespoň k částečné eliminaci problému týkajícího se základní metody segmentace s využitím DOM stromu, kdy je pro rozložení elementů na stránce použita například tabulka. Jak bylo v kapitole zabývající se implementací navržené metody popsáno, využívá se pro určení témat pro jednotlivé elementy SPARQL server, kterému jsou zasílány dotazy. Rychlost zpracování dotazu tímto serverem je silně závislá na jeho vytíženosti. Experimenty byly prováděny v rozmezí několika dní a proto bylo vypozorováno, že čas provedení anotace pro totožnou webovou stránku se některé dny výrazně prodloužil. Proto při provádění budoucích experimentů může nastávat situace, kdy se čas anotace konkrétní webové stránky bude lišit několikanásobně.

Kapitola 8

Závěr

V rámci této práce bylo nutné nastudovat problematiku spojenou s dvěma rozsáhlými tématy. První téma je oblast informačních technologií označovaná jako sémantický web. Z knižních i elektronických zdrojů byly nastudovány základní prvky a nástroje spojené se sémantickým webem, ať už se jedná o RDF trojice, ontologie a s nimi spojené ontologické jazyky, či dotazovací jazyk nad RDF daty, jazyk SPARQL. Dalé byly studovány a zkoumány nástroje a projekty, které byly vytvořeny v rámci sémantického webu. Stěžejními projekty byly DBpedia a DBpedia-Spotlight, které velice úzce souvisí s Wikipedií. V úvodních kapitolách této práce byly přiblíženy zmíněné nástroje a prvky sémantického webu i s některými ilustracemi pro lepší pochopení dané problematiky.

Oblast sémantického webu je velice rozsáhlá, nicméně v rámci této práce byly přiblíženy pouze základní stavební kameny sémantického webu, nutné pro pochopení problematiky a také nástroje, jenž byly přímo využity v pozdější fázi této práce.

Druhým rozsáhlým tématem je problematika týkající se segmentování webových stránek. Pro účely této práce byly nastudovány různé metody, jenž jsou využívány k segmentaci webových stránek. Především byly nastudovány a také představeny základní metody, z kterých vznikají nové metody pro segmentaci tak, že se základní metoda modifikuje, či se provede kombinace více metod. Byly představeny i výhody a nevýhody základních metod.

Hlávní částí této práce bylo však navrhnout metodu pro segmentaci webových stránek, která modifikuje základní metodu pro segmentaci za využití nástrojů z oblasti sémantického webu.

Základní metoda segmentace byla zvolena metoda založená na analýze DOM stromu webové stránky. Tato metoda pracuje především s HTML tagy jednotlivých elementů a na jejich základě provádí segmentaci. Hlavní myšlenka, jak modifikovat tuto metodu, byla neprovádět analýzu DOM stromu z hlediska typů jednotlivých elementů, ale z hlediska obsahu elementů, přesněji řečeno z hlediska sémantiky obsahu elementů. První fází návrhu metody bylo tedy docílit výsledku, kdy se ze vstupního textu provede určení sémantiky.

Nástroj, jenž byl zkoumán v oblasti sémantického webu, DBpedia-Spotlight, umožňuje jistým způsobem definovat sémantiku nějakého textu. Inspirací a vzorem pro tuto problematiku byl projekt [12], kdy byl Spotlight využit pro profilování účtů na sociálních sítí. Hlavní inspirací byla část, kdy ze zdroje v DBpedii, jenž Spotlight propojí s částí textu, se provede určení témat, včetně procentuálního podílu, jenž se v textu vyskytují. Pro tyto účely byl vytvořen seznam hlavních témat.

Po dokončení a otestování této fáze bylo možné aplikavat tento proces při analýze DOM stromu. Jelikož Spotlight provádí anotaci textu, bylo zřejmé, že nejprve bude nutné analyzovat elementy, jenž obsahují text, jenž je relevantní pro webovou stránku, tedy textu, jenž

je přímo zobrazen na stránce. Elementů, jenž obsahují relevantní text však může být pouze zlomek v rámci celé webové stránky. Bylo tedy nutné navrhnout způsob, jak bude možné na základě sémantiky pouze textových elementů provést segmentaci celé webové stránky.

Způsob segmentace byl založen na myšlence porovnávat témata elementu s tématy jeho rodičovského elementu a na základě podobnosti, či rozdílnosti těchto témat rozhodnout, zda bude element segmentován. Bylo tedy nutné vymyslet způsob, jak zajistit, aby měl rodičovský element přiřazená témata i přesto, že se nejedná o textový element a nebylo tedy možné mu přiřadit témata s využitím Spotlight. Pro tyto účely byl navržen způsob distribuce témat, kdy se nejprve provede distribuce témat od textových elementů, přes rodičovské elementy, až ke kořeni celého DOM stromu. Během tohoto procesu jsou rodičovským elementům přiřazována témata od jeho potomků. Jestliže je potomků více, bude rodiči přiřazena průměrná hodnota. Během distribuce směrem ke kořeni DOM stromu tak probíhá sčítání témat jeho potomků.

Dále bylo nutné, aby témata měly i ty elementy, jenž nejsou textové a ani žádný z potomků daného elementu není textový. Proto bylo nutné navrhnout způsob, jak přiřadit témata i těmto elementům. Pro tento účel byla využita reverzní metoda vůči metodě distribuce témat směrem ke kořeni. Tedy témata se nyní distribuovala od kořenového elementu směrem k listovým uzlů.

V této fázi tak bylo zajišteno určení témat pro všechny elementy v rámci DOM stromu. Nyní bylo možné přistoupit k návrhu metody, jak se budou porovnávat jednotlivá témata. Byl tedy navržen způsob, kdy jsou porovnávány dva seznamy témat, kdy jeden náleží aktuálnímu elementu a druhý jeho rodiči. V rámci těchto seznamů jsou porovnávána témata, jenž si odpovídají. Celkový rozdíl je pak tvořen součtem absolutních hodnot rodílů odpovídajících si dvojic témat. Po výpočtu tohoto rozdílu je možné rozhodnout, zda dojde k segmentaci či nikoliv, na základě porovnání se zadanou hodnotou při spuštění aplikace.

Takto byl navržen hlavní princip metody segmentace webové stránky, na základě sématiky obsahu elementů. Následně byl navržen způsob testování uživatelského rozhraní, při kterém byly odhaleny některé chyby při implementaci navržené metody a mohlo tak dojít k jejich opravě.

V rámci této práce byla také vytvořena demonstrační aplikace, jenž implementuje navrženou metodu a bylo tak možné ověřit chování této metody. Aplikace byla vytvořena v programovacím jazyce Python jako konzolová aplikace a byla vyvíjena a také testována v prostředí operačního systému Ubuntu.

Pro účely vizualizace výsledku segmentace implementovanou aplikací, byl vytvořen jednoduchý skript v ajzyce JavaScript a soubor kaskádových stylů, jenž společně způsobí zobrazení rámečků okolo segmentovaných elementů a také po najetí myší na daný segment zobrazí témata, jenž byla segmentu přiřazena.

Jelikož se jedná o experimentální metodu, kdy nebylo předem známo, jak se navržená metoda bude chovat v praxi v určitých případech, byl navržen způsob provádění experimentů, na jehož základě dojde k ověření chování této metody.

Experimenty byly navrženy tak, že se nejprve pro testovanou webovou stránku určil referenční výsledek segmentace, tzv. Ground truth (GT), s kterým bude možné porovnat výsledek segmentace, vytvořený navrženou metodou. Vytvoření GT pro testovanou stránku bylo velice subjektivní, jelikož dva lidé by mohli mít různý názor na to, zda z hlediska sémantiky nechat dva elementy sloučené, či je segmentovat. I z tohoto důvodu byly experimenty navrženy tak, že se zkoumaly zejména anomálie, tedy rozdíly mezi GT a výstupem, jenž nebyly před provedením experimentů očekávány.

Na základě provedených experimentů bylo odhaleno několik případů, při kterých se navržená metoda chová odlišně oproti očekávání a to napříč různými webovými stránkami. Některé problémy souvisí s tím, že se stále jedná o metodu, jenž analyzuje DOM strom a respektuje tak hierarchii uspořádání jednotlivých elementů. V jistých případech jsou však některé problémy spojené s touto metodou alespoň částečně eliminovány. Může tomu tak však být na úkor jiných elementů a jejich segmentace.

Během experimentů byla také zjištěna silná závislost vytíženosti SPARQL serveru, jenž zpracovává dotazy do databáze DBpedie, na času trvání implemenotvané aplikace při fázi anotace. Při analýze rozsáhlejších webových stránek, kde je větší množství textu je zasíláno SPARQL serveru velké množství dotazů. V případě, že daný server je již před anotací hodně vytížen, může se provádění anotace velice zpomalit.

Aplikace byla navržena tak, aby výstup segmentace bylo možné zpracovávat i strojově. Je však nutné tento výstup povolit při spuštění aplikace pomocí parametru příkazového řádku. Tento parametr povolí vytvoření textového souboru, jenž obsahuje XPATH všech elementů, jenž mají být segmentovány. Tento fakt umožňuje, jak již bylo řečeno, strojové zpracování a dává tak možnost dalšího využití a provádění dalších experimentů s navrženou metodou.

Literatura

[1] Berners Lee, T. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. Harper San Francisco, 1999. ISBN 1402842937.

[2] Brickley, D., Guha, R. a McBride, B. *RDF Schema 1.1* [online]. 2004. Aktualizováno 25. 2. 2014 [cit. 6. ledna 2020]. Dostupné z: https://www.w3.org/TR/rdf-schema/.

[3] Cai, D., Yu, S., Wen, J.-R. a Ma, W.-Y. *VIPS: a Vision-based Page Segmentation Algorithm*. MSR-TR-2003-79. November 2003. 28 s. Dostupné z: https://www.microsoft.com/en-us/research/publication/vips-a-vision-based-page-segmentation-algorithm/.

[4] Daiber, J., Jakob, M., Hokamp, C. a Mendes, P. N. Improving Efficiency and Accuracy in Multilingual Entity Extraction. In: *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*. 2013.

[5] Golbreich, C., Wallace, E. K., Patel Schneider, P. F. et al. New Features and Rationale (Second Edition). *OWL 2 Web Ontology Language* [online]. 2012. Aktualizováno 11. 12. 2012 [cit. 6. ledna 2020]. Dostupné z: https://www.w3.org/TR/owl2-new-features/.

[6] Lassila, O. a Swick, R. R. *Resource Description Framework (RDF) Model and Syntax Specification* [online]. 1999. Aktualizováno 10. 2. 2004 [cit. 6. ledna 2020]. Dostupné z: https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[7] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D. et al. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*. 2015, roč. 6, č. 2, s. 167–195. Dostupné z: http://jens-lehmann.org/files/2015/swj_dbpedia.pdf.

[8] Prud'hommeaux, E. a Seaborne, A. *SPARQL Query Language for RDF* [online]. 2006. Aktualizováno 15. 1. 2008 [cit. 6. ledna 2020]. Dostupné z: https://www.w3.org/TR/rdf-sparql-query/.

[9] Sano, H., Swezey, R. M. E., Shiramatsu, S., Ozono, T. a Shintani, T. A Web Page Segmentation Method by using Headlines to Web Contents as Separators and its Evaluations. In:. 2013.

[10] Smith, M. K., Welty, C. a McGuinness, D. L. Guide. *OWL Web Ontology Language* [online]. 2004. Aktualizováno 10. 2. 2004 [cit. 6. ledna 2020]. Dostupné z: https://www.w3.org/TR/owl-guide/.

[11] Svátek, V. *Ontologie a WWW* [online]. Říjen 2002 [cit. 6. ledna 2020]. Dostupné z: https://nb.vse.cz/~svatek/onto-www.pdf.

[12] Torrero, C., Caprini, C. a Miorandi, D. A Wikipedia-based approach to profiling activities on social media. *CoRR*. 2018, abs/1804.02245. Dostupné z: http://arxiv.org/abs/1804.02245.

[13] Zelený, J. a Burget, R. Cluster-based Page Segmentation - a fast and precise method for web page pre-processing. In: *The Third International Conference on Web Intelligence, Mining and Semantics*. Association for Computing Machinery, 2013, s. 1–12. Dostupné z: https://www.fit.vut.cz/research/publication/10252. ISBN 978-1-4503-1850-1.

Příloha A

Spuštění skriptu a vizualizace segmentace

Následující kapitola bude sloužit jako návod pro spuštění skriptu, provádějícího anotaci i segmentaci vstupního HTML dokumentu, na základě cesty k danému dokumentu, či na základě URL webové stránky. Budou uvedeny i veškeré potřebné prerekvizity a moduly, které musí být předem nainstalovány, aby bylo umožněno spuštění skriptu a jeho bezproblémový chod. Také zde budou uvedeny podmínky, pro vizualizaci provedené segmentace.

A.1 Spuštění Spotlight

Aby bylo možné provést anotaci či segmentaci s využitím přiloženého skriptu je nutné spustit lokální verzi Spotlight. V kapitole 5.3, kokrétně v sekci 5.3.1 byly uvedeny počáteční problémy provázené spuštění Spotlight lokálně. Spotlight se nakonec tedy podařilo spustit na operačním systému Ubuntu 18.4. Každopádně pro samotné spuštění Spotlight je vyžadována velká kapacita operační paměti, tato práce byla implementována a testována s operační pamětí o velikosti 16GB. Pro spuštění Spotlight byl využit Docker[1] a bylo postupováno dle návodu poskytnutého vývojáři Spotlight, který lze najít na webu `https://github.com/dbpedia-spotlight/spotlight-docker`. Příkaz pro spuštění Spotlight s anglickou verzí slovníku vypadá následovně:

```
docker run -itd -restart unless-stopped -p 2222:80 dbpedia/spotlight-english spotlight.sh
```

, kde 2222 je číslo portu, na kterém bude Spotlight komunikovat.

V úvodní části skriptu pro anotaci a segmentaci je nastavena proměnná `end_point_for_annotation`, skrze kterou je definován end-point, na kterém Spotlight komunikuje. Její defaultní hodnota je nastavena na `"http://localhost:2222/rest/candidates"`. Tudíž pokud bude provedeno spuštění Spotlight s jiným číslem portu, je nutné změnit číslo portu také v proměnné `end_point_for_annotation` v rámci skriptu **wpas.py**.

Po úspěšném spuštění Spotlight je možné spustit skript **wpas.py**.

[1] Docker `https://www.docker.com/`

A.2 Spuštění skriptu pro anotaci a segmentaci

Implementovaný skript pro anotaci a segmentaci nese název **wpas.py** (název je zkratka od **W**eb **P**age **A**notation and **S**egmentation). Byl implementován a zároveň testován v jazyce Python verze 3.8.3. Běh skriptu pro starší verze Pythonu nebyl testován, nicméně by dle očekávání měl fungovat i pod stašími verzemi Pythonu 3. Skript byl testován v operačním systému Ubuntu 18.4.

Pro spuštění skriptu wpas.py je nutné nejprve spustit Spotlight a další podmínkou je mít nainstalované následující moduly (vše pro Python 3):

- `lxml`
- `urllib`
- `requests`
- `PySpotlight`
- `SPARQLWrapper`
- `BeautifulSoup`
- `Selenium`
- `pyautogui`
- `pyperclip`

Pro usnadnění instalace výše uvedených modulů byl přiložen soubor "requirements.txt", s jehož pomocí lze tyto moduly automaticky nainstalovat. Nutností však je mít nainstalován `pip3`. To lze provést následujícími příkazy:

- `sudo apt update`
- `sudo apt install python3-pip`

Jakmile je pip3 nainstalován, lze spustit následující příkaz, jenž spustí instalaci modulů, jejichž názvy, včetně verzí, se nachází v souboru "requirements.txt":

- `pip3 install -r requirements.txt`

Případně:

- `sudo pip3 install -r requirements.txt`

Mimo tyto moduly, jenž využívá Python 3, je nutné nainstalovat utilitu jménem `XSEL` a webový prohlížeč Firefox. Jelikož, jak již bylo popsáno v kapitole 5.3, je při stažení webové stránky simulováno zadávání názvu souboru do dialogového okna, jako by byla stisknuta kombinace kláves "ctrl + v" (případně "command + v" pro operační systém Mac), aby se zabránilo problémům týkajících se různých rozložení klávesnic. Proto je nutné mít tuto utilitu nainstalovanou, jelikož v opačném případě nepůjde stránka stáhnou z důvodu, že není povoleno využít klávesovou zkratku "ctrl + v" ("command + v").

Instalaci utility XSEL lze spustit příkazem:

- `sudo apt install xsel`

Všechny ostatní moduly, jenž jsou využívány při implementaci jsou již součástí instalace Pythonu 3.

Jak již bylo dříve řečeno, skript wpas.py je možné spustit buď pouze pro anotaci HTML dokumentu, nebo pouze k jeho segmentaci (musí být již dříve anotován) a nebo lze skript pustit jak pro anotaci tak i pro segmentaci zároveň. Nyní budou uvedeny jednotlivé varianty spuštění včetně popisu jednotlivých argumentů.

A.2.1 Spuštění anotace

K spuštění skriptu wpas.py pro anotaci, jejímž vstupem má být HTML dokument, který se již nachází v některém adresáři počítace, lze využít následující příkaz :

```
python3 wpas.py -a annotation -i <inputfile> [-c <confidence>] [-s <support>]
```

,kde

- -a (–action) - Definuje jaká akce má být provedena, v tomto případě, kdy vyžadujeme pouze anotaci, je hodnota argumentu ”annotation”.
- -i (–inputfile) - Název, případně cesta k HTML dokumentu, jenž má být anotován.
- -c (–confidence) - Volitelný argument, jímž lze definovat počáteční parametr *confidence*, jenž je předán Spotlight a který definuje míru důverihodnosti s jakou Spotlight propojí část textu se zdrojem v Dbpedii . Defaultní hodnota je nastavena na 0,5.
- -s (–support) - Volitelný argument, jenž stejně jako argument -c slouží ke konfiguraci anotace pomocí Spotlight, konkrétně parametru *support*. Support definuje kolik daný zdroj musí mít minimálně odkazů na Wikipedii, aby byl brán jako relevantní. Zdroje jenž této hodnoty nedosahují jsou ignorovány. Pomocí support lze omezit využití neinformativních a nedůležitých zdrojů. Defaultní hodnota je nastavena na 20.

Vyžaduje-li uživatel nejprve stáhnout danou webovou stránku a poté na staženém HTML dokumentu provést anotaci, lze skript spustit následujícím příkazem:

```
python3 wpas.py -a annotation -u <url> -f <file> [-c <confidence>] [-s <support>]
```

,kde

- -a (–action) - Definuje jaká akce má být provedena, v tomto případě, kdy vyžadujeme pouze anotaci, je hodnota argumentu ”annotation”.
- -u (–url) - URL webové stránky, jejíž stažení a následná anotace je požadováno.
- -f (–file) - Název souboru, respektiva cesta do adresáře včetně názvu souboru, kterým bude pojmenován stažený HTML dokument a v případě zadání celé cesty bude uložen do příslušného adresáře.
- -c (–confidence) - Volitelný argument, jímž lze definovat počáteční parametr *confidence*, jenž je předán Spotlight a který definuje míru důverihodnosti s jakou Spotlight propojí část textu se zdrojem v Dbpedii . Defaultní hodnota je nastavena na 0,5.
- -s (–support) - Volitelný argument, jenž stejně jako argument -c slouží ke konfiguraci anotace pomocí Spotlight, konkrétně parametru *support*. Support definuje kolik daný zdroj musí mít minimálně odkazů na Wikipedii, aby byl brán jako relevantní. Zdroje jenž této hodnoty nedosahují jsou ignorovány. Pomocí support lze omezit využití neinformativních a nedůležitých zdrojů. Defaultní hodnota je nastavena na 20.

Po spuštění skriptu jsou všechny argumenty ověřeny a v případě, že zadané argumenty jsou v pořádku spustí se anotace vstupního HTML dokumentu, případně je nejdříve stažena webová stránka. O průběhu anotace je uživatel informován prostřednictvím procentuální hodnoty značící množství již zpracované části HTML dokumentu. V případě jakékoliv chyby je uživatel o dané chybě informován prostřednictvím chybového výstupu na příkazové řádce. Jakmile anotace proběhne úspěšně skript je ukončen a uživatel je informován o úspěšném dokončení anotace.

A.2.2 Spuštění segmentace

Pro spuštění skriptu wpas.py za účelem segmentace, lze využít následující příkaz:

```
python3 wpas.py -a segmentation -i <inputfile> [-d <difference>] [-x <xpath>]
```

,kde

- -a (–action) - Definuje jaká akce má být provedena, v tomto případě, kdy vyžadujeme pouze segmentaci, je hodnota argumentu ”segmentation”.
- -i (–inputfile) - Název, případně cesta k HTML dokumentu, jenž má být segmentován
- -d (–difference) - Volitelný argument, jímž lze ovlivnit granularitu segmentace. Defaultní hodnota je nastavena na 20.
- -x (–xpath) - Nastavením volitelného argumentu -x je na konci segmentace vytvořen textový soubor, jehož název je roven hodnotě argumentu -x z příkazového řádku, který obsahuje XPATH všech elementů, jenž jsou hraniční. Název je nuné zadat včetně cesty do požadovaného adresáře. V případě, že je zadán pouze název souboru, je tento soubor vytvořen v adresáři, kde je umístěn skript wpas.py.

Jakmile je skript spuštěn a vstupní argumenty ověřeny, provede se segmentace HTML dokumentu. Uživatel je v případě chyby informován výpisem na chybový výstup a po dokončení segmentace je informován o úspěšném dokončení procesu.

A.2.3 Spuštění anotace i segmentace

Jestliže je požadováno provést anotaci a ihned segmentaci HTML dokumentu, skript lze spustit následujícím příkazem:

```
python3 wpas.py -a anseg -i <inputfile> [-c <confidence>] [-s <support>] [-d
<difference>] [-x <xpath>]
```

,kde

- -a (–action) - Definuje jaká akce má být provedena. ”anseg” značí kombinaci anotace a segmentace.
- -i (–inputfile) - Název, případně cesta k HTML dokumentu, jenž má být anotován a následně segmentován.
- -c (–confidence) - Volitelný argument, jímž lze definovat počáteční parametr *confidence*, jenž je předán Spotlight a který definuje míru důverihodnosti s jakou Spotlight propojí část textu se zdrojem v Dbpedii . Defaultní hodnota je nastavena na 0,5.
- -s (–support) - Volitelný argument, jenž stejně jako argument -c slouží ke konfiguraci anotace pomocí Spotlight, konkrétně parametru *support*. Support definuje kolik daný

zdroj musí mít minimálně odkazů na Wikipedii, aby byl brán jako relevantní. Zdroje jenž této hodnoty nedosahují jsou ignorovány. Pomocí support lze omezit využití neinformativních a nedůležitých zdrojů. Defaultní hodnota je nastavena na 20.

- -d (–difference) - Volitelný argument, jímž lze ovlivnit granularitu segmentace. Defaultní hodnota je nastavena na 20.
- -x (–xpath) - Nastavením volitelného argumentu -x je na konci segmentace vytvořen textový soubor, jehož název je roven hodnotě argumentu -x z příkazového řádku, který obsahuje XPATH všech elementů, jenž jsou hraniční. Název je nuné zadat včetně cesty do požadovaného adresáře. V případě, že je zadán pouze název souboru, je tento soubor vytvořen v adresáři, kde je umístěn skript wpas.py.

Stejně jako v případě anotace lze daný skript spustit s argumenty, které způsobí nejprve stažení požadované webové stránky a následně anotaci a segmentaci příslušného HTML dokumentu. Aby byla nejdříve webová stránka stažena lze spustit skrip následujícím příkazem:

```
python3 wpas.py -a anseg -u <url> -f <file> [-c <confidence>] [-s <support>] [-d <difference>] [-x <xpath>]
```

,kde

- -a (–action) - Definuje jaká akce má být provedena. ”anseg” značí kombinaci anotace a segmentace.
- -u (–url) - URL webové stránky, jejíž stažení a následná anotace je požadováno.
- -f (–file) - Název souboru, respektiva cesta do adresáře včetně názvu souboru, kterým bude pojmenován stažený HTML dokument a v případě zadání celé cesty bude uložen do příslušného adresáře.
- -c (–confidence) - Volitelný argument, jímž lze definovat počáteční parametr *confidence*, jenž je předán Spotlight a který definuje míru důverihodnosti s jakou Spotlight propojí část textu se zdrojem v Dbpedii . Defaultní hodnota je nastavena na 0,5.
- -s (–support) - Volitelný argument, jenž stejně jako argument -c slouží ke konfiguraci anotace pomocí Spotlight, konkrétně parametru *support*. Support definuje kolik daný zdroj musí mít minimálně odkazů na Wikipedii, aby byl brán jako relevantní. Zdroje jenž této hodnoty nedosahují jsou ignorovány. Pomocí support lze omezit využití neinformativních a nedůležitých zdrojů. Defaultní hodnota je nastavena na 20.
- -d (–difference) - Volitelný argument, jímž lze ovlivnit granularitu segmentace. Defaultní hodnota je nastavena na 20.
- -x (–xpath) - Nastavením volitelného argumentu -x je na konci segmentace vytvořen textový soubor, jehož název je roven hodnotě argumentu -x z příkazového řádku, který obsahuje XPATH všech elementů, jenž jsou hraniční. Název je nuné zadat včetně cesty do požadovaného adresáře. V případě, že je zadán pouze název souboru, je tento soubor vytvořen v adresáři, kde je umístěn skript wpas.py.

V případě spuštění skriptu pro anotaci i segmentaci s argumentem -u je nejprve stažena webová stránka a následně se automaticky spustí anotace a následně segmentace pro stažený HTML dokument. Jestliže byla stránka již dříve stažena a vyžaduje se tedy anotace

a segmentace souboru jenž je dostupný offline, je zadáván pouze název, respektive cesta k danému. Vstupní soubor pro segmentační část je předán interně a nemusí být tedy zadán explicitně.

A.3 Vizualizace výsledku segmentace

Výstupní dokument po provedení segmentace obsahuje přidané atributy u příslušných elementů, které definují, že na daném elementu má být provedena segmentace. Aby byla segmentace viditelná je ke skriptu přiložen soubor *segmentation.css* s kaskádovými styly a soubor *segmentation.js* s kódem v jazyce JavaScript. Kombinace těchto souborů s výstupním souborem segmentace umožní zobrazení rámečků okolo elementů, jenž byly segmentovány a také zobrazení témat včetně procentuálního podílu pro daný element. Tato témata se zobrazí v levém horním rohu okna, po najetí kurzorem na příslušný element. Aktuální element je pro lepší orientaci podbarven šedou barvou. Po načtení stránky jsou již některá témata zobrazena. Jedná se o témata pro celou webovou stránku.

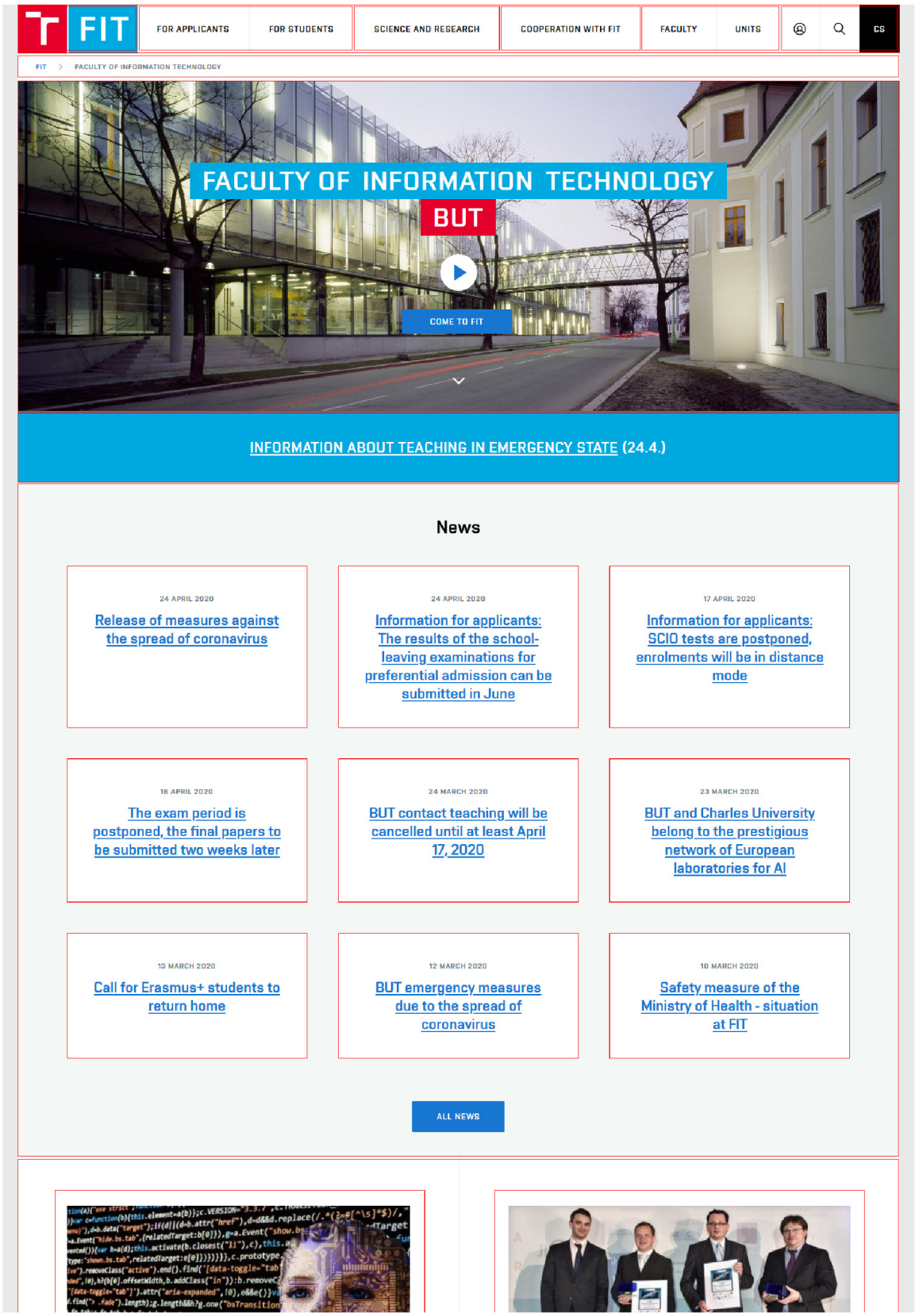
Soubor *segmentation.css* a *segmentation.js* je nutné vložit do stejného adresáře, ve kterém se nachází segmentovaný HTML dokument. Propojení CSS souboru a JavaScriptového souboru s HTML dokumentem je již provedeno automaticky, jak již bylo vysvětleno v kapitole 5.4 v sekci 5.4.4.

Jestliže uživatel vyžaduje jiný způsob vizualizace segmentace, stačí aby upravil tyto dva soubory.

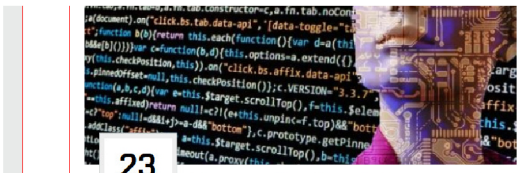
Příloha B

Ukázky segmentace úvodní strany webu Fakulty informačních technologií

V rámci této přílohy budou představeny Ground truth segmentace pro webovou stránku dostupnou na adrese https://www.fit.vut.cz/.en.



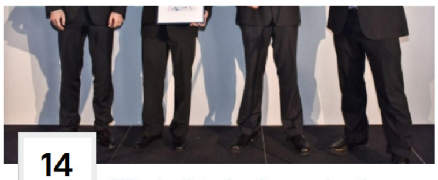
Obrázek B.1: Ground truth část 1



23 MARCH 2020

Charles University and Brno University of Technology belong to the prestigious network of European laboratories for AI

Two Czech laboratories working in automatic speech and language processing have achieved significant international success. The results of the European program supporting the creation of artificial intelligence centers were announced in March, and the institutions involved in the HumanE-AI-Net project also include the Faculty of Mathematics and Physics, Charles University in Prague and the Faculty of Information Technology, BUT.



14 FEBRUARY 2020

FIT scientists develop acceleration technologies for high-speed networks. Their probe also helps with lawful interception

The ANT@FIT research team from BUT has created one of the world's first acceleration cards with a bandwidth of 100 Gb/s.In contrast to other devices available, the new card is much more powerful, allowing deployment in high-speed networks. Working on acceleration of time-critical operations used in network infrastructure equipment and in network monitoring and security, the research group also participated in several other commercially successful projects.

ALL PRESS RELEASES

Events at FIT

27 MAY 2020	Excel@FIT 2020 online
13 - 28 JULY 2020	2020 Brno International Summer School in Information Technology
24 - 28 AUGUST 2020	Letní škola (F)IT nejen pro holky

ALL FIT EVENTS

FOR APPLICANTS
- Study in English
- Study in Czech
- Short-term Study
- Full Degree Programmes
- Summer Schools
- Students with Special Needs in Studies
- Contact

FOR STUDENTS
- Study Information
- Study News
- For Freshmen
- Academic Year
- Courses
- Degree Programmes
- State Final Examination and Theses
- Study and Internships Abroad
- Creative Activity of Students
- Offers of Professional Internships
- FIT Student Union

SCIENCE AND RESEARCH
- Science and Research at FIT
- Research Groups
- European Structural Funds Projects
- Projects
- Publications
- Products
- Patents
- Conferences
- Awards and Recognitions

COOPERATION WITH FIT
- Industry Cooperation
- Partners
- Research Partnership
- Foreign Cooperation
- Collaboration with Schools
- Faculty Services
- Job Offers in IT

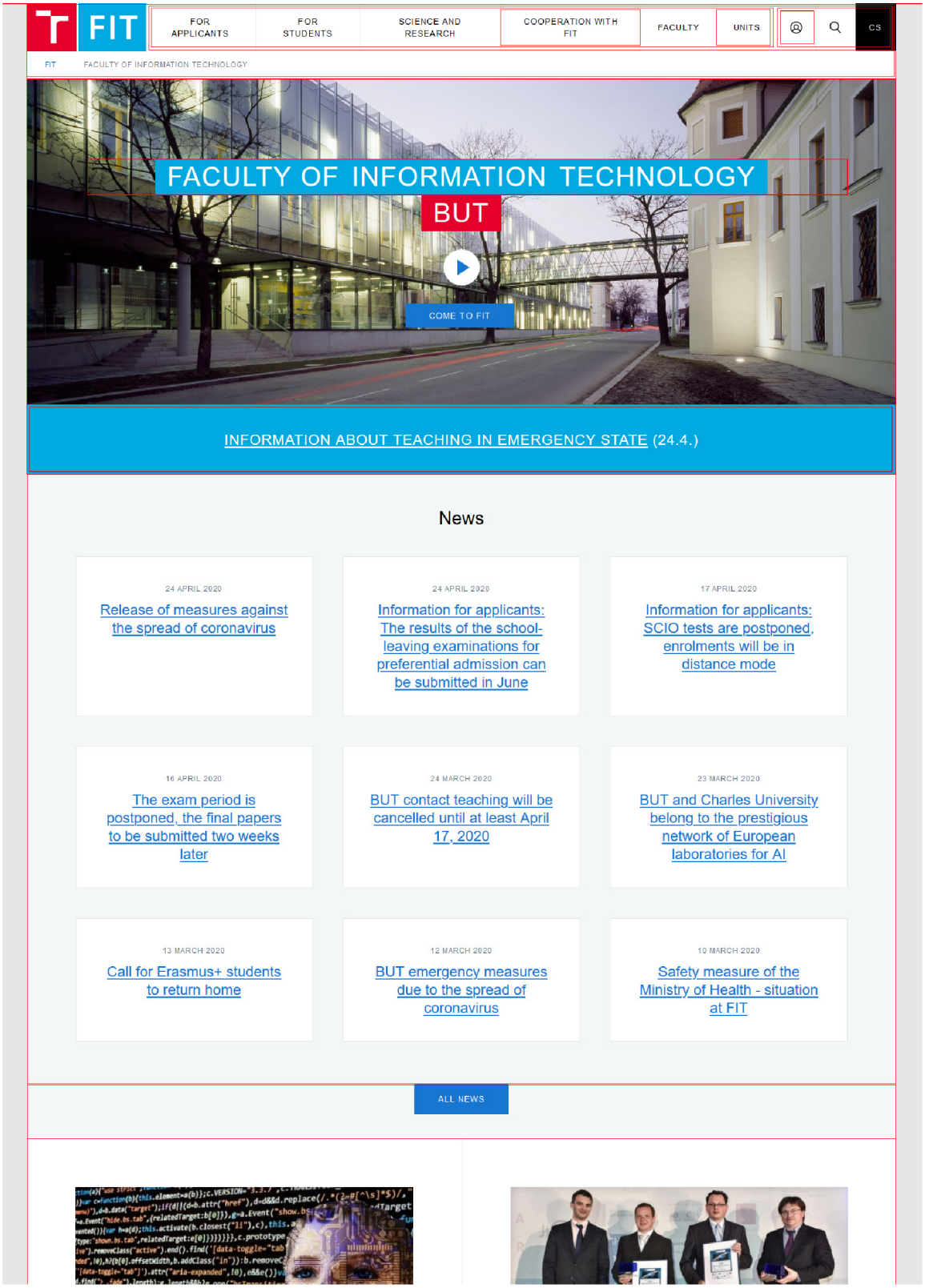
FACULTY
- News
- Event Calendar
- Campus Map
- Faculty Campus
- Organizational Structure
- Official Notice Board
- History and Now
- For Media
- Staff
- Contacts
- Protection of Personal Data

UNITS
- Deans Office
- Department of Information Systems
- Department of Intelligent Systems
- Department of Computer Graphics and Multimedia
- Department of Computer Systems
- Computer Centre
- Research Centre of Information Technology
- Library
- Museum

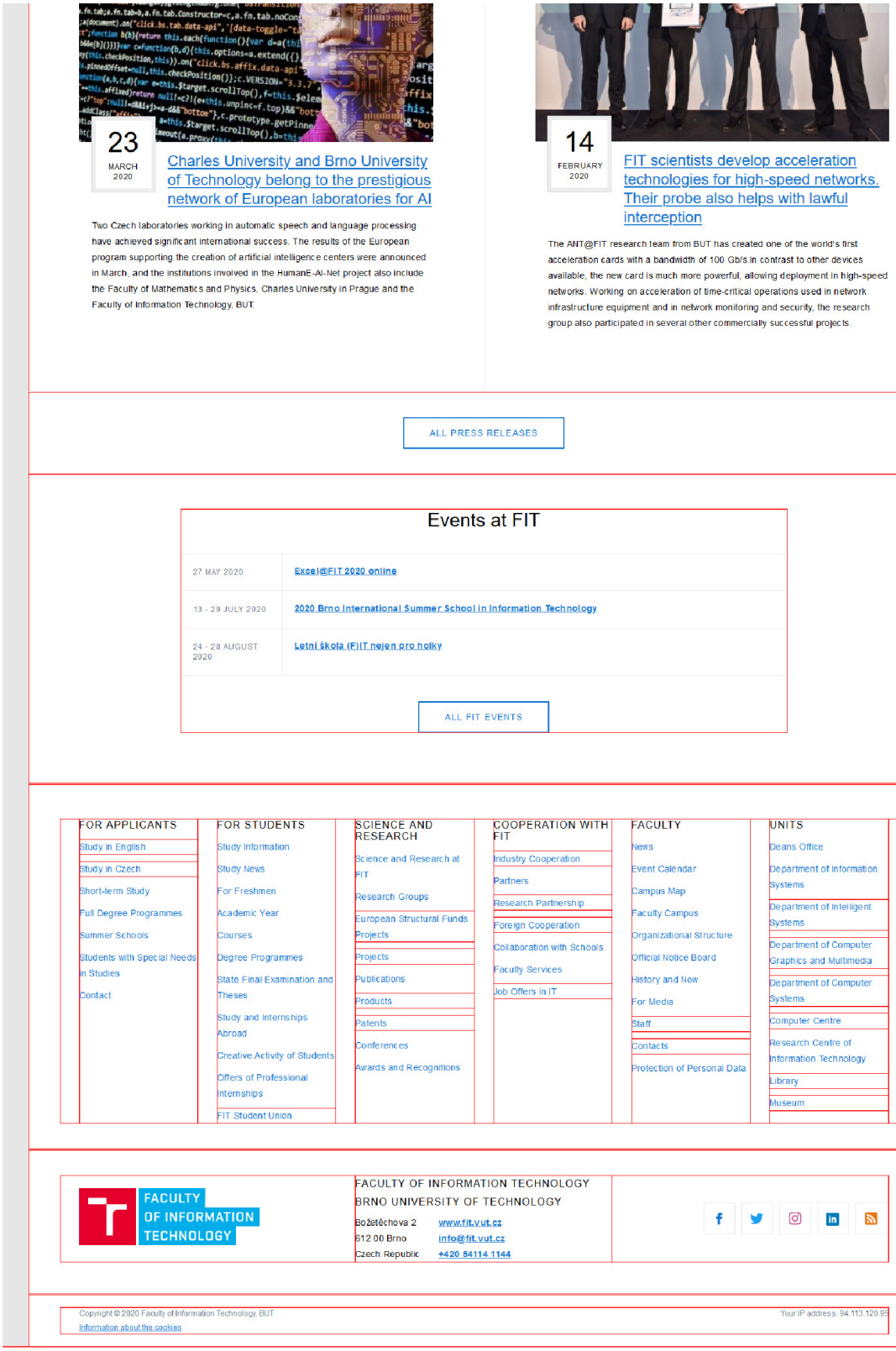
FACULTY OF INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY
BRNO UNIVERSITY OF TECHNOLOGY
Božetěchova 2
612 00 Brno
Czech Republic
www.fit.vut.cz
info@fit.vut.cz
+420 54114 1144

Obrázek B.2: Ground truth část 2



Obrázek B.3: Výstup aplikace část 1



Obrázek B.4: Výstup aplikace část 2