Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-5220-56475

Bc. Peter Macko

ZJEDNOTENÉ VYHĽADÁVANIE NAD PREPOJENÝMI DÁTAMI NA WEBE

Diplomová práca

Študijný program: Softvérové inžinierstvo
Študijný odbor: 9.2.5 Softvérové inžinierstvo
Miesto vypracovanie: Ústav informatiky a softvérového inžinierstva, FIIT STU Bratislava
Vedúci práce: Ing. Michal Holub

máj, 2013

S T U
F I I T
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta informatiky a informačných technológií

Zadanie diplomovej práce

Meno študenta: **Bc. Macko Peter**

Študijný program: Softvérové inžinierstvo

Študijný odbor: Softvérové inžinierstvo

Názov práce: **Zjednotené vyhľadávanie nad prepojenými dátami na webe**

Samostatnou výskumnou a vývojovou činnosťou v rámci predmetov Diplomový projekt I, II, III vypracujte diplomovú prácu na tému, vyjadrenú vyššie uvedeným názvom tak, aby ste dosiahli tieto ciele:

Všeobecný cieľ:

Vypracovaním diplomovej práce preukážte, ako ste si osvojili metódy a postupy riešenia relatívne rozsiahlych projektov, schopnosť samostatne a tvorivo riešiť zložité úlohy aj výskumného charakteru v súlade so súčasnými metódami a postupmi študovaného odboru využívanými v príslušnej oblasti a schopnosť samostatne, tvorivo a kriticky pristupovať k analýze možných riešení a k tvorbe modelov.

Špecifický cieľ:

Vytvorte riešenie, zodpovedúce návrhu textu zadania, ktorý je prílohou tohto zadania. Návrh bližšie opisuje tému vyjadrenú názvom. Tento opis je záväzný, má však rámcový charakter, aby vznikol dostatočný priestor pre Vašu tvorivosť.

Riaďte sa pokynmi Vášho vedúceho.

Pokiaľ v priebehu riešenia, opierajúc sa o hlbšie poznanie súčasného stavu v príslušnej oblasti alebo o priebežné výsledky Vášho riešenia alebo o iné závažné skutočnosti, dospejete spoločne s Vaším vedúcim k presvedčeniu, že niečo v texte zadania a/alebo v názve by sa malo zmeniť, navrhnite zmenu. Zmena je spravidla možná len pri dosiahnutí kontrolného bodu.

Miesto vypracovania: Ústav informatiky a softvérového inžinierstva FIIT STU v Bratislave

Vedúci práce: **Ing. Michal Holub**

Termíny odovzdania

podľa harmonogramu štúdia platného pre semester, v ktorom máte príslušný predmet (Diplomový projekt I, II, III) absolvovať podľa Vášho študijného plánu

Predmety odovzdania:

V každom predmete dokument podľa pokynov na www.fiit.stuba.sk v časti:
home > Informácie o > štúdiu > organizácia štúdia > diplomový projekt

V Bratislave dňa 13. 2. 2012

prof. Ing. Pavol Návrat, PhD.
riaditeľ Ústavu informatiky a softvérového inžinierstva

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta informatiky a informačných technológií

Návrh zadania diplomovej práce

Finálna verzia do diplomovej práce [1]

Študent:

Meno, priezvisko, tituly:	Peter Macko, Bc.
Študijný program:	Softvérové inžinierstvo
Kontakt:	mr.petermacko@gmail.com

Výskumník:

Meno, priezvisko, tituly:	Michal Holub, Ing.

Projekt:

Názov:	Zjednotené vyhľadávanie nad prepojenými dátami na webe
Názov v angličtine:	Unified search of linked data on the Web
Miesto vypracovania:	Ústav informatiky a softvérového inžinierstva, FIIT STU, Bratislava
Oblasť problematiky:	Webové inžinierstvo

Text návrhu zadania[2]

Vyhľadávanie informácií na dnešnom webe je čoraz zložitejšie vďaka jeho enormnému rastu a faktu, že dáta sú publikované neštruktúrovane. Čoraz viac údajov sa začína publikovať v štruktúrovanej podobe, čoho dôkazom je aj Linked Data iniciatíva. Takéto dáta umožnia presnejšie zodpovedanie informačných potrieb používateľov, písanie dopytov pre prepojené dáta však nie je triviálna úloha.

Analyzujte problematiku prepojených dát v kontexte webu so sémantikou. Zamerajte sa na možnosti prepojenia rôznych datasetov za účelom ich priameho prehľadávania. Analyzujte možnosti vyhľadávania na webe so sémantikou prívetivým spôsobom pre používateľov. Navrhnite metódu, ktorá umožní vyhľadávať vo viacerých štruktúrovaných datasetoch, bez nutnosti poznania ich presnej štruktúry alebo jazykov určených na dolovanie dát z nich (napr. SPARQL). Metóda poskytne používateľom výsledky vzťahujúce sa k zadanému dopytu sformulovanému v semi-prirodzenom jazyku. Metóda tiež zjednoduší zadávanie dopytov v porovnaní s vytváraním dopytu v jazyku SPARQL pomocou vopred definovaných šablón, ktoré budú jednoducho konštruovateľné.

Analyzujte a navrhnite vylepšenie pre dnes používané vyhľadávanie vo vybranej problémovej doméne na webe (napr. digitálne knižnice). Navrhnuté riešenie experimentálne overte prostredníctvom proxy servera upravujúceho existujúci portál alebo vytvorením nového portálu umožňujúceho vyhľadávanie informácií v zvolenej doméne.

---

[1] Vytlačiť obojstranne na jeden list papiera

[2] 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

Literatúra[3]

- D. Tümer, M. A. Shah and Y. Bitirim, "An Empirical Evaluation on Semantic Search Performance of Keyword-Based and Semantic Search Engines: Google, Yahoo, Msn and Hakia," 2009 Fourth International Conference on Internet Monitoring and Protection, IEEE, pp. 51-55, May 2009.
- R. Valencia-Garcia, F. Garcia-Sanchez, D. Castellanos-Nieves and J. T. Fernndez-Breis, "OWLPath: An OWL Ontology-Guided Query Editor," IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, Vol. 41, No. 1, pp. 121-136, Jan. 2011.
- G. Zou, B. Zhang, Y. Gan and J. Zhang, "An Ontology-Based Methodology for Semantic Expansion Search," 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery, IEEE, pp. 453-457, Oct. 2008.

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Peter Macko, konzultoval(a) a osvojil(a) si ho Ing. Michal Holub a súhlasí, že bude takýto projekt viesť v prípade, že bude pridelený tomuto študentovi.

V Bratislave dňa 9.1.2012

Podpis študenta

Podpis výskumníka

Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schválený: áno / ~~nie~~[4]

Dňa: 7.2.2012

Podpis garanta predmetov

[3] 2-3 vedecké zdroje, každý na samostatnom riadku a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho aktuálnosť (uveďte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

[4] Nehodiace sa prečiarknite

ANOTÁCIA

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Študijný program: SOFTVÉROVÉ INŽINIERSTVO

Autor: Bc. Peter Macko

Diplomový projekt: Zjednotené vyhľadávanie nad prepojenými dátami na webe

Vedenie diplomového projektu: Ing. Michal Holub

máj, 2013

Vyhľadávanie informácií na webe je dnes stále náročnejšie pre jeho obrovský rast. Aby toho nebolo málo, väčšina údajov zverejnených na webe sa nachádza v neštruktúrovanej podobe. Napriek tomu sa na webe objavuje stále viac štruktúrovaných dát, čo je hlavne zásluhou iniciatívy Linked Data. V tomto čase je však iba niekoľko vyhľadávačov schopných prehľadávať takéto dáta s použitím všetkých ich možností. Väčšina vyhľadávačov vyhľadáva pomocou kľúčových slov. Pre využitie prepojených dát je nutné použiť špeciálny dopytovací jazyk, akým je napr. SPARQL.

My tento fakt zmeníme vytvorením komplexného vyhľadávača, ktorý bude rozumieť pseudoprirodzenému jazyku používateľa. Tieto dopyty budú následne transformované na SPARQL a vykonané nad ontologickou databázou. Naša metóda rieši vďaka predspracovaniu aj slovníkový problém, a preto sa používateľ môže dopytovať sebe prirodzenými výrazmi.

Kľúčové slová: *Sémantický web, SPARQL, dopyty v prirodzenom jazyku, Liked Data, prepojené dáta, RDF, vyhľadávanie, vyhľadávač*

ANNOTATION

Slovak University of Technology Bratislava

FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Degree Course: SOFTWARE ENGINEERING

Author: **Bc. Peter Macko**

Diploma project: Unified Search of Linked Data on the Web

Supervisor: Ing. Michal Holub

2013, May

Searching information on the Web is difficult because of its enormous size. To make matters worse, most of the data published on the Web is in unstructured format. However, more and more structured data is being published, what is also evident from the emergence of unifying initiatives like Linked Data. Nowadays, there are only few search engines able to search in structured data using the full power of the provided semantics. The majority of the search engines look for information using keywords. To utilize the full power of the structured data a special query language, like SPARQL, has to be used.

We would like to change this fact by creating a complex search engine which should be able to interpret user queries in pseudo-natural language. Queries will be transformed to SPARQL language and executed on an ontological database. Our method solves the vocabulary problem, too. Therefore user can use phrases typical for him.

Keywords: *Semantic web, SPARQL, natural language query, Linked Data, RDF, searching, search engine*

Prehlasujem, že som túto prácu vypracoval samostatne, na základe získaných vedomostí, s využitím uvedenej literatúry.

Bratislava, máj 2013

Obsah

Kapitola 1

Úvod

Na dnešnom webe sa nachádza veľké množstvo stránok. Väčšina z týchto stránok potrebuje ukladať dáta a dynamicky ich načítavať, pričom na tieto účely využívajú prevažne relačné databázy. Tieto databázy umožňujú veľmi rýchly a spoľahlivý prístup k dátam. Čo im však chýba, je istá prepojenosť, najmä na sémantickej úrovni. Webové stránky, resp. webové aplikácie pracujú nad svojimi dátami s istými výnimkami, kedy jednotlivé aplikácie spolupracujú pomocou webových služieb. Takisto, dáta uložené v relačných databázach o sebe nehovoria takmer nič, a tak je ich sémantické prepojenie veľmi náročné.

Keď sa pozrieme na samotnú podstatu dát na týchto serveroch, dochádza tu k veľkým redundanciám. Jednotlivé webové stránky hovoria napríklad, o tých istých osobách, geografických objektoch, projektoch, veciach, atď. Pritom však tieto entity nie sú prepojené, a teda sa medzi nimi napr. nedá vyhľadávať s využitím týchto prepojení, čo sa nazýva sémantické vyhľadávanie.

Predstavme si modelovú situáciu, že chceme vyhľadať pána *Nováka*, ktorý pracuje v spoločnosti *Spoločnosť s.r.o.* Keď dáme tieto informácie do vyhľadávača, dostaneme veľké množstvo výsledkov, keďže ľudí s menom *Novák* je veľa. Vyhľadávač však nezohľadňuje väzbu medzi zamestnancom a firmou, a tak nami hľadaný *Novák* sa s veľkou pravdepodobnosťou nebude nachádzať medzi prvými výsledkami vyhľadávania.

Keby vyhľadávač použil dodatočné informácie, ktoré môže poskytnúť sémantický web, identifikoval by v ňom väzbu *zamestnávateľ* a nami hľadaný *Novák* by bol zobrazený na prvých miestach. Práve týmto problémom, teda prepojenými dátami (angl. Linked Data),

sa venuje čím ďalej tým viac výskumníkov v oblasti webu. V prepojených dátach sú entity jednoznačne pomenované a identifikované a takisto obsahujú aj vzájomné prepojenia rôznych typov. Preto sú s ich pomocou vyhľadávacie dopyty konkrétnejšie.

Sémantický web je však v ranných fázach vývoja a tak sa stretáva s množstvom problémov, ktoré ešte len čakajú na svoje vyriešenie. Ukladanie sémantického obsahu nie je jednoduché a úložiská, ktoré takéto spôsoby reprezentácie dát podporujú, sa len pomaly vyvíjajú. Takisto sa takéto úložiská stretávajú s problémami s výkonom podobne, ako to bolo pri príchode objektových úložísk.

Vytváranie a úprava dát je pre sémantický web tiež problematická. Existujú viaceré formáty zápisu sémantického obsahu, ktoré sú často navzájom nekompatibilné. Úprava takýchto dát bola do nedávnej doby veľkým problémom, ktorému sa nevenoval žiaden štandard v tejto oblasti. Momentálne sa však situácia mení a jednotlivé úložiská začínajú implementovať štandardizované spôsoby úpravy dát.

Keďže sémantické dátové zdroje vznikajú na viacerých miestach, je náročné udržať konzistenciu pomenovania. Takisto, nie všetky vzťahy sú v sémantických dátach podchytené, ale veľké množstvo vzťahov medzi dátami je možne odvodiť z existujúcich vzťahov.

Problém, ktorému sa venujeme v tejto práci, je vyhľadávanie v prepojených dátach. Vyhľadávanie v tejto oblasti je momentálne možné pomocou sady príkazov, ktoré bežný používateľ webu nie je schopný jednoducho zostaviť a napísať. Presadenie sémantického obsahuje je však závislé na čo najväčšej používateľskej základni, ktorá bude schopná tieto dáta využívať. Práve z týchto dôvodov sme sa zamerali práve na problém komunikácie s databázou pomocou prirodzeného jazyka.

Podobný problém sa v dávnejšej minulosti viacerí výskumníci pokúšali vyriešiť v oblasti klasických, objektovo relačných databázach. Tieto riešenia sú známe pod pojmom rozhrania prirodzeného jazyka nad databázou (angl. *Natural Language Database Interface*). Tieto riešenia však boli obmedzené vlastnosťami objektovo relačných databáz.

Pri samotnom vyhľadávaní preto využívame vlastnosti sémantických dát. Najjednoduchším spôsobom získania informácií pre bežného používateľa je prirodzený jazyk, ktorým ľudia denne komunikujú. Webové vyhľadávače naučili síce ľudí písať dopyty v podobe kľúčových slov, nie je to však úplne prirodzený spôsob komunikácie. Okrem toho, dopyty v podobe kľúčových slov strácajú pôvodnú sémantiku otázok. Práve preto sa zameriavame na spracovanie dopytov od používateľa v prirodzenom jazyku. Jazykom, ktorý sme si na tento účel vybrali, je angličtina, a to hlavne kvôli jej rozšíreniu v celom svete. Angličtina má aj veľkú výhodu v tom, že absolútna väčšina dnes existujúcich sémantických dátových zdrojov má názvy elementov práve v tomto jazyku.

Spracovanie prirodzeného jazyka nie je triviálne, pretože bežný jazyk obsahuje veľké množstvo foriem, ktorými sa dá vyjadriť tá istá myšlienka, príp. otázka. Naše riešenie je jedinečné v tom, že navádza používateľa pri kladení otázky tak, aby použil pojmy, ktoré dokážeme spracovať a poskytnúť mu tak žiadané informácie.

Kapitola 2

Prepojené dáta

Nasledujúca kapitola sa venuje všeobecnému úvodu do problematiky prepojených dát. Kapitola popisuje spôsoby zápisu dát v sémantickej podobe pomocou formátov:

- RDF/XML
- Turtle

Ďalej sa v kapitole sústreďujeme na jazyk SPARQL, ktorý slúži na dopytovanie prepojených dát. V záverečnej časti kapitoly ešte skúmame možnosti získavania výstupu zo sémantickej databázy, konkrétne vo formáte:

- XML.SPARQL
- JSON

2.1 Reprezentácia prepojených dát

Prepojené dáta, ktoré sme spomínali v predchádzajúcej časti, je nutné reprezentovať, a to vo forme prístupnej pre spracovanie počítačom. Na tento účel sa využíva technika definície *trojíc* subjekt-predikát-objekt. V tomto zápise *subjekt* reprezentuje dátovú entitu, *objekt* je hodnota, pričom *predikát* reprezentuje vzťah medzi subjektom a objektom tak, ako je znázornené na obrázku 1.

Tento spôsob zápisu by sa dal vysvetliť na nasledujúcom príklade vzťahu *„Jozef je otec Evy“*. Tu je možné povedať, že *„Jozef“* je subjekt, *„Eva“* je objekt a *„je otec“* je predikát. Vďaka takémuto zápisu vo forme trojíc sú vzťahy medzi entitami využiteľné pri počítačovom spracovaní dát. Okrem toho vzťah *„je otec“* sa môže vyskytnúť aj v podobe objektu.

Napríklad sa dá vytvoriť trojica „je otec“ *„je podtrieda“* *„je rodič“*. Zo vzťahov sa následne dajú odvodiť vzťahy v opačnom smere, teda *„je dieťa“* a *„je dcéra“*.

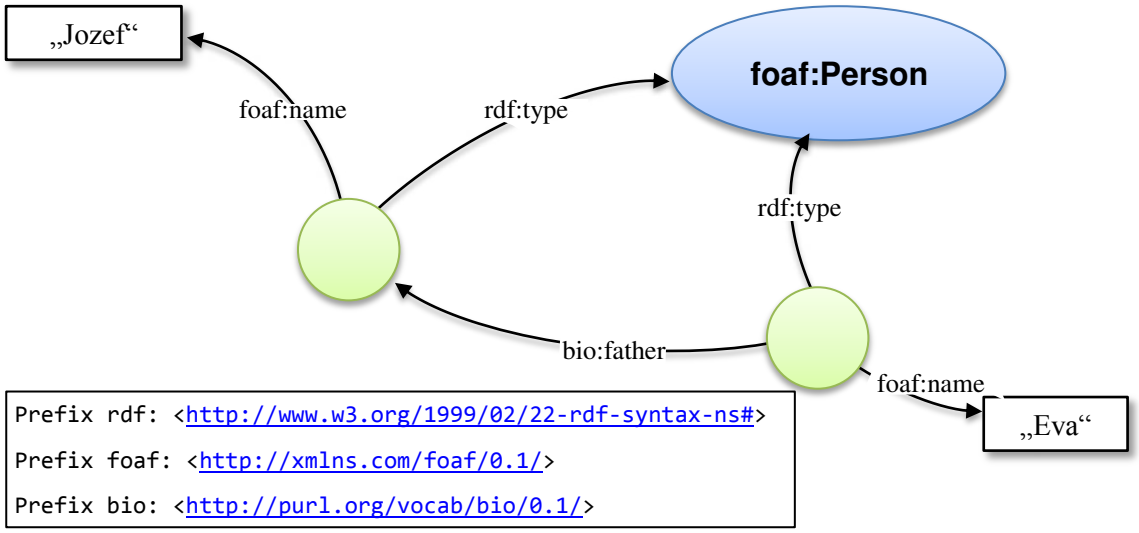
Obrázok 1 Reprezentácia trojíc v prepojených dátach

2.2 Zápis prepojených dát

Hlavným mechanizmom v oblasti zápisu týchto dát je RDF (angl. Resource Description Framework)[1]. Je to jazyk pre reprezentáciu informácií o zdrojoch v prostredí webu. Tento štandard je špecifikáciou formátu pre zápis entít pomocou abstraktného modelu metadát. Prácu na tomto štandarde zastrešuje konzorcium W3C, ktoré pokrýva väčšinu štandardov súvisiacich s vývojom webu.

RDF používa na identifikovanie zdrojov URI (jednotný identifikátor zdroja[2], angl. Uniform resource identifier[3]) alebo unikátny identifikátor zdroja. Ten vyzerá podobne ako URL, aj keď nemusia reprezentovať webovú stránku. Toto URI hovorí o príslušnosti entity k autorovi, ale takisto prostredníctvom HTTP protokolu poskytuje informácie o samotných objektoch.

Reprezentácia dát prostredníctvom RDF dovoľuje ich zobrazenie prostredníctvom grafovej štruktúry. Na predchádzajúcom príklade väzby *„Jozef je otec Evy“* to vieme znázorniť obrázkom 2.



Obrázok 2 Grafová reprezentácia väzby *"Jozef je otec Evy"*

[1] http://www.w3.org/RDF/
[2] http://aleph.nkp.cz/
[3] http://www.w3.org/TR/uri-clarification/

2.2.1 Spôsoby zápisu trojíc v RDF

V súčasnosti sa používajú prevažne dva základné spôsoby (formáty) zápisu RDF trojíc pre strojové spracovanie:

- RDF/XML
- Turtle

RDF/XML

Základnou možnosťou zápisu RDF dát je zápis v podobe XML, ktorý sa nazýva aj RDF/XML[4]. Pri tejto forme zápisu sú jednotlivé subjekty obalené v elemente *Person,* čo určuje ich definíciu. Ich vlastnosti (predikáty) sú ich podelementami, pričom je možné používať aj zápis, kedy sú atribútmi.

Na začiatku súboru sa potom nachádza definícia skratiek pre zdroje v danom RDF dokumente. Jednotlivé zdroje potom nie je nutné špecifikovať celou URI cestou k zdroju, ale je možné použiť iba jej skrátený tvar. Príklad takéhoto dokumentu na entitách Jozef a Eva z predchádzajúceho textu je znázornený na obrázku 3.

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:bio="http://purl.org/vocab/bio/0.1/">
  <foaf:Person
    rdf:about="http://priklad.fiit.stuba.sk/osoby/Jozef">
    <foaf:name>Jozef</foaf:name>
  </foaf:Person>
  <foaf:Person
    rdf:about="http://priklad.fiit.stuba.sk/osoby/Eva">
    <foaf:name>Eva</foaf:name>
    <bio:father
      rdf:resource="http://priklad.fiit.stuba.sk/osoby/Jozef"/>
  </foaf:Person>
</rdf:RDF>
```

Obrázok 3 Príklad dokumentu vo formáte RDF. Dokument obsahuje dva objekty typu *Person* s menami *Adam* a *Eva*. V príklade sú tiež znázornené aj ich vzťahy cez väzbu *father*.

Turtle

Druhým, často používaným spôsobom zápisu, je Turtle[5]. Tento vznikol zjednodušením jazyka Notation3 (N3) a využíva možnosti N-Triples formátu. Na začiatku dokumentu sú

[4] http://www.w3.org/TR/REC-rdf-syntax/
[5] http://www.w3.org/TR/turtle/

uvedené predpony (prefix). Príklad takéhoto dokumentu je na obrázku 4. Tento formát je takisto jednoducho strojovo spracovateľný, ale oproti formátu RDF/XML je menej náročný na dátové prenosy, keďže neobsahuje také množstvo redundancií.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix bio: <http://purl.org/vocab/bio/0.1/> .

<http://priklad.fiit.stuba.sk/osoby/Jozef>
  foaf:name "Jozef" .

<http://priklad.fiit.stuba.sk/osoby/Eva>
  foaf:name "Eva" ;
  bio:father <http://priklad.fiit.stuba.sk/osoby/Jozef> .
```

Obrázok 4 Príklad dokumentu vo formáte Turtle

2.3 Získavanie obsahu z prepojených dát

V dnešnej dobe sú 3 možnosti, ktorými je možné získať dáta zo sémantických databáz. Prvou možnosťou je sťahovanie celej dátovej množiny vo formátoch, ktoré sme spomínali vyššie. Táto možnosť je síce najjednoduchšia, ale nie najpohodlnejšia, ak chceme pracovať s dátami.

Druhá možnosť, ktorú poskytujú autori niektorých dátových množín, je prístup cez nimi vytvorené rozhranie. Tu sú často obľúbené možnosti navigácie v podobe faziet alebo vyhľadávania pomocou kľúčových slov.

Spôsob, ktorý sa pre naše ďalšie využitie najviac hodí, je dopytovanie dátových zdrojov pomocou špeciálneho jazyka SPARQL[6].

2.3.1 Dopytovanie prepojených dát

SPARQL nadväzuje na klasické relačné databázy, v ktorých sa už niekoľko rokov používa jazyk SQL. Úpravou jazyka SQL vznikol jazyk SPARQL, ktorý je prispôsobený pre dopytovanie dát nad sémantickým webom.

Jeho pomocou je možne získavať dáta prostredníctvom jednoduchých výberových klauzúl, ktoré sú obdobou SQL príkazu SELECT. SPARQL podporuje filtrovanie záznamov, ich zoraďovanie a ďalšie možnosti. Vznikajú takisto aj rozšírenia jazyka, ktoré umožňujú aktualizáciu prepojených dát. Príklad takéhoto dopytu je na obrázku 5.

V schéme dopytu vidieť použitie formát zápisu Turtle, ktorý sme opísali v predchádzajúcej kapitole. Jeho hlavnou prednosťou je deklarácia predpôn (angl. Prefix), ktoré slúžia ako skratky ku kompletným URI k danej schéme. V SPARQL dopyte nie je potom nutné písať

[6] http://www.cambridgesemantics.com/semantic-university/sparql-by-example

URI pred každým atribútom alebo zdrojom, ale namiesto nich je možné použiť definovanú predponu.

```
# deklarácia predpôn
PREFIX foo: <http://example.com/resources/>
...
# definícia datasetu
FROM ...
# klauzula výsledku
SELECT ...
# model klauzule
WHERE {
  ...
}
# modifikátor klauzule
ORDER BY ...
```

Obrázok 5 Schéma SPARQL dopytu

Klauzula *from* potom definuje, ktoré RDF grafy budú dopytované v danom SPARQL dopyte. *Select* určuje, ktoré dáta z jednotlivých grafov budú získavané. *Where* určuje, čo konkrétne ideme získavať z danej dátovej množiny. Nakoniec modifikátory klauzuly určujú preskupenia dát vo výsledku. V tomto jazyku sa klauzula *from* využíva iba v prípade, ak je nutné vybrať dáta iba z časti podgrafu. Príklad takéhoto dopytu sa nachádza na obrázku 6.

```
PREFIX akt: <http://www.aktors.org/ontology/portal#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT
  ?title ?name
WHERE {
  ?article  akt:has-title   ?title.
  ?article  akt:has-author  ?author.
  ?author   akt:full-name   ?name.
  FILTER (
    ?name = "Mária Bielik" ||
    ?name = "Michal Holub"
  )
}
GROUP BY ?title ?name
```

Obrázok 6 Príklad SPARQL dopytu. Príklad vracia názov článku a meno autora pre skupinu článkov ktorých autorom je Michal Holub alebo Mária Bielik

Uvedený príklad získava všetky články, ktoré napísali autori *Mária Bielik* alebo *Michal Holub*. Vo výsledkoch sa v tomto prípade bude nachádzať meno článku, ktoré je k článku

pripojené pomocou predikátu *has-title* a meno autora, ktorý článok napísal. Autor je pritom spojený s článkom predikátom *has-author* a meno autora je k autorovi pripojené pomocou predikátu *full-name*.

2.3.2 Výsledky dopytu

Výsledok SPARQL dopytu môže byť vo viacerých formátoch podľa danej implementácie a výberu používateľa. Základným spôsobom, založeným na formáte XML [7], je XML.SPARQL[8]. Tento formát výstupu je odporúčaný W3C konzorciom ako základný spôsob výstupu dát z ontologickej databázy. Príklad takéhoto výstupu sa nachádza na obrázku 7.

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="hpage"/>
    <variable name="name"/>
    <variable name="age"/>
  </head>
  <results>
    <result>
      <binding name="hpage">
         <uri>http://work.example.org/bob/</uri>
      </binding>
      <binding name="name">
         <literal xml:lang="en">Bob</literal>
      </binding>
      <binding name="age">
         <literal
        datatype="http://www.w3.org/2001/XMLSchema#integer">
           30
         </literal>
      </binding>
    </result>
    ...
  </results>
</sparql>
```

Obrázok 7 Príklad výsledkov dopytu v XML.SPARQL formáte

[7] http://www.w3.org/XML/
[8] http://www.w3.org/TR/rdf-sparql-XMLres/

Tento výstup samostatne definuje hlavičku, kde uvádza jednotlivé mená stĺpcov. Následne sa vo výstupe nachádzajú jednotlivé dáta, ktoré majú na seba naviazané meno stĺpca. Vo výstupe sa môžu nachádzať nielen jednoduché objekty, ako je text alebo čísla, ale aj komplexné objekty. Každý stĺpec má definovaný typ, poprípade iné atribúty, ako je napríklad jazyk pri textových premenných.

Vďaka tomu, že ide o XML formát, je jednoducho čitateľný a je ho možné prehľadávať vo väčšine programovacích jazykov. Taktiež však je menej šetrný k prenášanému množstvu dát, a to vďaka redundanciám v XML formáte.

Dnes sa na webe s veľkou obľubou začína používať formát JSON[9], hlavne pre jeho skromnejší spôsob zápisu. Konzorcium W3C začalo pracovať na tomto spôsobe výstupu SPARQL dopytov[10]. Formát je priamo odvodený od predchádzajúceho XML formátu. Takisto preto obsahuje definíciu hlavičky, teda stĺpcov výstupu a samotného výstupu. Pri každom prvku výstupu je potom definovaný typ a hodnota. Príklad takéhoto výstupu je na obrázku 8.

```
{
  "head": { "vars": [ "hpage" , "name", "age" ]
  } ,
  "results": {
    "bindings": [
      {
        "hpage": {
          "type": "uri",
          "value": "http://work.example.org/bob/"
        },
        "name": {
          "type": "literal",
          "value": "Bob"
        },
        "age": {
          "type": "literal",
          "value": "30"
        }
      },
      ...
    ]
}}
```

Obrázok 8 Príklad výstupu SPARQL dopytu v JSON formáte

[9] http://www.json.org/
[10] http://www.w3.org/TR/rdf-sparql-json-res/

Samotný výstup je vďaka JSON dobre čitateľný, ale niektorými programovacími jazykmi menej podporovaný. Jeho forma však prináša výhodu väčšej šetrnosti k prenášaným dátam.

Výstupom SPARQL dopytu môže byť okrem iného aj samotné RDF. Toto môže byť vo viacerých formách ako napríklad RDF/XML, N-Triples, Turtle, atď.

Pre používateľa je najlepšie čitateľný formát typu HTML. Tu vidí používateľ dáta naformátované v podobe grafických prvkov. Pre takýto spôsob výstupu sa často používajú XSL Transformácie, ktoré vedia upravovať zadaný zdroj v XML podobe do výstupu v HTML.

2.3.3 SPARQL 1.1

Keďže SPARQL prišiel s menším počtom možností, bolo nutné v krátkom čase rozšíriť jeho možnosti. Z tohto dôvodu vznikol projekt SPARQL 1.1[11], ktorý má priniesť vylepšenia protokolu predstavené v nasledujúcej časti. SPARQL 1.1 je veľmi mladý štandard a práce na ňom stále prebiehajú pod záštitou W3 konzorcia.

V novom štandarde sa počíta s vybudovaním nového protokolu pre SPARQL na báze REST. Vďaka nemu bude možné využívať možnosti protokolu HTTP pre prácu s dátami. Ďalej bude nová verzia obsahovať možnosti pre popis služby, a teda získavanie informácií o štruktúre dát. SPARQL 1.1 bude možné využiť na prácu naprieč viacerými dátovými zdrojmi.

Aktualizácia dát (Update)

SPARQL 1.1 obsahuje aj možnosti aktualizácie dát[12]. V štandarde je definované, že pri vytvorení prvej dátovej jednotky je nutné, aby vznikol graf. Na rozdiel od toho pri mazaní poslednej dátovej jednotky konkrétna implementácia môže, ale nemusí vymazať celý graf.

Pri vytváraní dát sa používa klauzula INSERT DATA nasledovaná grafom (subjekt), do ktorého sa dáta vkladajú, následne predikát a objekt. Takým istým spôsobom sa zapisuje aj príkaz DELETE.

Ďalšou časťou štandardu je DELETE/INSERT klauzula. Tá je vhodná na modifikáciu dát, pričom v časti DELETE sa definujú dáta, ktoré treba zmeniť a v časti INSERT sa definuje hodnota, ktorou sa má zmeniť. V tejto časti je definovaná aj klauzula DELETE WHERE, v ktorej sa definuje, ktoré dáta treba vymazať podľa zadaných parametrov.

Jazyk okrem toho obsahuje možnosti pre manažment grafov. Tu sú definované nasledujúce funkcie:

- CREATE – vytvorí graf, ak sú podporované prázdne grafy
- DROP – vymaže graf a všetky jeho súčasti
- COPY – modifikuje graf tak, aby obsahoval kópiu iného
- MOVE – presunie všetky dáta zo zadaného grafu na iný
- ADD – reprodukuje všetky dáta z jedného grafu na iný

[11] http://www.w3.org/TR/sparql11-query/
[12] http://www.w3.org/TR/sparql11-update/

2.4 Zhrnutie

V tejto kapitole sme analyzovali spôsoby, akými je možné vytvárať a dopytovať sémantické dáta. Tieto spôsoby sú dobre prepracované a v ďalšej práci budeme na ich možnostiach stavať. Využijeme možnosti zapisovanie dát v RDF trojiciach. Na ich získavanie využijeme jazyk SPAQRL ako dnešný štandard na dopytovanie ontologických dát.

Treba však povedať, že tento dopytovací jazyk je pre bežných používateľov značne komplikovaný, čo bolo možné vidieť aj v ukážke dopytu v tomto jazyku. Pri jeho využití je nutné pracovať s mennými priestormi a písať netriviálne množinové operácie. Pri písaní dopytu je nutné okrem poznania správnej syntaxi poznať aj štruktúru dát. Práve z týchto dôvodov sa v práci sústredíme na vyvinutie mechanizmu, ktorým by bežný používateľ vedel jednoduchšou formou získavať informácie zo sémantických dát.

Kapitola 3

Vyhľadávanie pomocou dopytov v prirodzenom jazyku

Táto kapitola sa venuje spôsobom dopytovania relačných dátových úložísk pomocou prirodzeného jazyka. V ďalšej časti kapitoly sa venujeme nástrojom napomáhajúcim pri spracovaní prirodzeného jazyka, ktorými sú:

- WordNet
- Standford Parser

Problém komunikácie so strojmi pomocou prirodzeného jazyka je tu už od samotného začiatku a rozšírenia počítačov v spoločnosti. Prvým pokusom, ktorý je dodnes používaný, je príkazový riadok. Prostredníctvom neho používateľ komunikuje s počítačom a posiela mu príkazy o tom, čo má vykonať.

Prirodzený jazyk na získavanie dát z veľkých úložísk je tu tiež už dlhšiu dobu. Tento problém sa v literatúre označuje ako *rozhranie prirodzeného jazyka nad databázou* (angl. Natural language interface to databases) [1]. Jadrom problému je teda to, aby počítačový program dokázal odpovedať (získať tie dáta, ktoré používateľ žiada) na podnety od používateľa v prirodzenom jazyku.

3.1 Rozhranie prirodzeného jazyka nad relačnou databázou

Prvé zmienky o riešení problému rozhrania v prirodzenom jazyku pre relačné databázy priniesli výskumníci pre organizáciu NASA so svojím reportom k projektu Lunar [2].

V tomto projekte autori popisujú spôsob dopytovania databázy prirodzeným jazykom. Ďalšou zaujímavou prácou v tejto oblasti bolo použitie vyberacích menu [3] na dopytovanie v databáze. Autori v tomto príspevku prišli s riešením, pri ktorom si používateľ vyberá postupne príkazy (Find, Delete,...), atribúty, tabuľky, spojenia (and, or) a modifikátory (whose name is, whose address is,...). Takýmto spôsobom si používateľ vie vytvoriť dopyt, ktorý nimi navrhnuté riešenie dokáže preložiť do SQL jazyka.

V momentálnej dobe je tento problém stále nevyriešený a viacerí výskumníci sa tejto tematike venujú. Príkladom je projekt rozhrania prirodzeného jazyka založený na konverzácii [4]. Riešenie pozostáva z dvoch hlavných častí a to konverzačného agenta (angl. conversation agent) a stromu vedomostí (angl. knowledge tree). Riešenie konverzačného agenta vychádza z projektu ADAM [5], ktorý je systémom pre komunikáciu so študentmi. Agent má na starosti identifikáciu vzorov. Kľúčovou časťou riešenia je strom vedomostí, v ktorom sú usporiadané vedomosti o danom dátovom zdroji. Pri dopytovaní sa potom riešenie pohybuje postupne v strome, pričom zisťuje, ktorému nasledujúcemu podstromu vyhovuje zadaná veta viac. Tento strom vedomostí je získaný od expertov v danej oblasti, čo znamená, že fáza predspracovania vyžaduje ľudský zásah. Následkom tohto faktu je aj to, že strom vedomostí nevie odpovedať iba na definované otázky.

Ďalšou prácou v tejto oblasti je MASQUE/SQL [6]. Tu ide o nadviazanie na systém MASQUE, ktorý bol vytvorený na dopytovanie sa prirodzeným jazykom v angličtine na databázu prologu. MASQUE/SQL robí podobný prevod na klasickú SQL databázu. Jej predpokladom je spracovanie štruktúry dopytu pomocou gramatického spracovania. Výstupom tohto spracovania je takzvaný LQL dopyt, ktorý sa neskôr transformuje do SQL dopytu. Problémom riešenia pre zložité dopyty bola nutnosť vytvorenie hierarchických stromov, ktoré hovorili o príbuznosti entít. Teda napríklad, ako uvádzajú autori v článku, tak zamestnanec ako aj a zákazník sú priamo odvodené z entity osoba.

Svoj prístup k tejto problematike priniesol článok spoločnosti Microsoft [7]. Autor v tomto článku hovorí o spôsobe transformácie prirodzeného jazyka na SQL dopyty pomocou OLAP kocky (OLAP Cube). OLAP kocka je istým vyjadrením hlavných entít a vzťahov medzi týmito entitami. Článok sa teda zameriava na využitie týchto vzťahov na dopytovanie dát. V článku sa uvádza, že automatické nástroje založené na OLAP kocke, vedeli v experimentoch zachytiť až 85 % vzťahov daného dátového zdroja automaticky.

3.2 Nástroje používané v doméne dopytovania prirodzeným jazykom

Pri dopytovaní v prirodzenom jazyku sa často využívajú aj nasledujúce nástroje:

- StandfordParser[13] – analýza viet
- WordNet[14] – databáza anglických slov so vzájomnými prepojeniami

[13] http://nlp.stanford.edu/software/lex-parser.shtml
[14] http://wordnet.princeton.edu/

3.2.1 StandfordParser

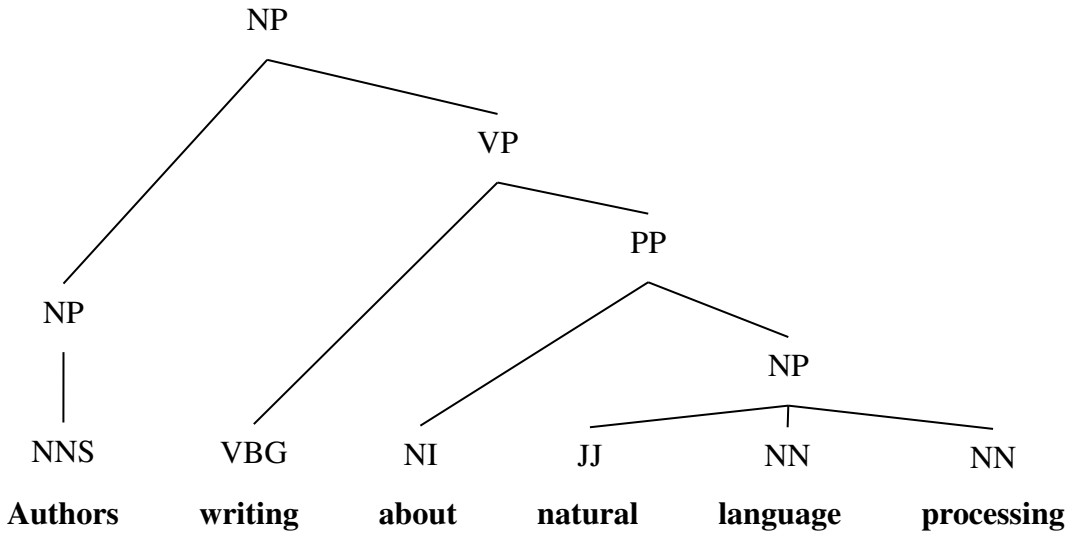
Nástroj StandfordParser [8] je vytvorený na analýzu vety. Nástroj po zadaní vety zistí väzby medzi slovami. Takisto označuje hlavné slová danej vety a slová, ktoré ich rozširujú. Na obrázku 9 je rozobraný dopyt v tvare: *„authors writing about natural language processing“*. Táto veta je na ďalšom obrázku 10 reprezentovaná v stromovej štruktúre.

```
(ROOT
  (S
    (NP
      (NP
        (NNS autors))
      (VP
        (VBG writing)
        (PP
          (IN about)
          (NP
            (JJ natural)
            (NN language)
            (NN processing)
          )
        )
      )
    )
  )
)
```

Obrázok 9 Výstup programu StandfordParser pre vetu *„Authors writing about natural language processing“*.

V tomto dopyte sú identifikované menné frázy (angl. Noun Phrases), ďalej už len NP. Prvým je *„autors“*, ktorý je identifikovaný aj ako podmet (angl. subject) a druhým je takisto správne identifikovaná skupina slov *„natural language procesing“*. V tejto fráze máme ešte rozlíšené podstatné a prídavné mená. Takisto vieme identifikovať slovesné frázy (angl. Verb Phrases), pričom sa identifikuje prísudok.

Tento nástroj používajú viaceré riešenia na analýzu vety. V takto zanalyzovanej vety je ľahšie nájsť entity a vzťahy. Napríklad v našej vete vieme, že hľadáme väzbu pisateľa (autora) medzi autorom a dielom. Takisto vieme že hľadáme diela, ktoré sú o téme spracovania prirodzeného jazyka.



Obrázok 10 Strom dopytu "Authors writing about natural language processing"

3.2.2 WordNet

WordNet [9] je rozsiahla databáza anglických slov. Táto databáza obsahuje veľké množstvo informácií o angličtine. Základom databázy je zoznam anglických slov. Tieto slová sú rozdelené do kategórií podľa slovných druhov (podstatné, prídavné mená, slovesá atď.) Ďalej sa v databáze nachádzajú vzťahy medzi slovami. Tie sú na úrovni toho či ide o sloveso, ale aj relácie na úrovni podmnožín.

Z dát je možné zistiť, že slovo autor je vo vzťahu so slovom človek, pričom človek je nadmnožina autora. Všetky vzťahy sú tranzitívne, takže ak *„autor obrazu“* je podmnožina autora, tak je aj podmnožinou človeka. Takisto sú v databáze definované aj vzťahy typu „*z čoho sa skladá*“. Teda, ak je definované, že človek má dve ruky a dve nohy, takisto je definované aj to, že autor má dve ruky a dve nohy. Vzťahy podobnosti sú aj medzi slovesami, teda napríklad komunikácia, rozhovor a šepot majú spoločnú väzbu podobnosti. Pre jednotlivé slová obsahuje nim prislúchajúce slovné druhy (angl. Part of Speach, POS).

Viaceré riešenia, ktoré pracujú s prirodzeným jazykom a jeho prevodom na iné formy, používajú túto databázu. Vďaka nej zisťujú, aké entity je možné v dátovej množine hľadať namiesto používateľom zadaného slova. To prináša možnosti písania dopytov v prirodzenom jazyku vo svojom vlastnom slovníku a nie v slovníku danej dátovej množiny, ktorý nemusí používateľ poznať.

3.3 Zhrnutie

Z analýzy riešení, ktoré sme opísali, vyplýva fakt, že v klasických relačných databázach je hlavným problémom extrakcia relácií. Dáta sú umiestnené v tabuľkovej štruktúre, záznamy sú v tabuľkách previazané podľa primárnych kľúčov, relácií. Tieto relácie však o sebe hovoria len veľmi málo.

Práve preto sa viacerí autori sústredili na to, ako tieto relácie interpretovať a obohacovať. Riešenia opísané v tejto kapitole využívali prístupy OLAP kocky alebo prístupy stromov. Práve z tohto dôvodu sa pri riešení nášho problému sústredíme práve na prepojené dáta

alebo ontologické databázy. Tieto dáta sú postavené na inom princípe, entity sú jasne identifikované a väzby medzi nimi je možné podrobne opisovať. Takéto dátové zdroje sú preto ideálnym kandidátom na dopytovanie pomocou prirodzeného jazyka.

Nástroje, ktoré sa pri spracovaní dátových zdrojov používajú, sú užitočné aj v prostredí prepojených dát. Práve preto sú nástroje WordNet a StandfordParser použité aj v našej metóde.

Kapitola 4

Problémy spracovania sémantického obsahu

V tejto kapitole uvádzame problémy spojené so spracovaním sémantického obsahu, ktoré sa týkajú nášho riešenia. Opisujeme základné formy a spôsoby dopytovania:

- Kľúčové slová
- Šablónové dopyty
- Pseudo/úplne prirodzený jazyk

Okrem toho sa v kapitole venujeme spôsobom extrakcie trojíc z webových sídiel (napr. Wikipédia), ktoré sú využité pri budovaní dátovej množiny, akými sú napr. YAGO a DBpedia. Na záver kapitoly sa venujeme problémom fixnej dátovej množiny a predspracovania sémantického úložiska.

Téma sémantického obsahu je veľmi mladá, a preto sa dnes spracovanie takýchto dát stretáva s množstvom problémov. Tieto problémy momentálne rieši viacero výskumníkov, čo napomáha ďalšiemu rozširovaniu takéhoto spôsobu reprezentácie a ukladania dát. O tom svedčí aj množstvo dátových zdrojov, ktoré sa nachádzajú v LOD (linking open data project cloud)[15].

[15] http://lod-cloud.net/

4.1 Komunikácia so sémantickou databázou

Sémantické dáta sú uložené v grafoch, čo je presne problém, prečo bežní používatelia nevedia využiť ich potenciál. Tieto dáta sú v tejto podobe veľmi ťažko dopytovateľné, a preto používatelia siahajú radšej po jednoduchších metódach, ako zistiť svoju odpoveď. Práve preto je kľúčovým problémom pre ďalší vývoj priblížiť spôsoby práce so sémantickým obsahom bežným používateľom.

Väčšina dnes rozšírených prístupov k dopytovaniu takéhoto druhu dát sú založené na vyhľadávaní pomocou kľúčových slov [10, 11]. Tieto spôsoby sú veľmi jednoduchým riešením spracovania dát v sémantickom úložisku, pričom vôbec nevyužívajú možnosti, ktoré takýto druh úložísk poskytuje.

Prístup, ktorý sa viac približuje práci s ontologickým úložiskom, je založený na písaní šablónových dopytov [12, 13, 14]. Pri takomto spôsobe riešenia používateľ píše dopyt v jazyku, ktorý je blízky prirodzenému jazyku. Používateľ je však obmedzený na vyberanie z konkrétneho množstva operátorov, ktoré pri písaní môže použiť. Takýto spôsob sa používa na rozširovanie dátových množín v metóde GINO (A Guided Input Natural Language Ontology Editor) [12].

Používateľ teda skonštruuje dopyt typu: „*The depth of lake Tahoe is 1645 feet*“, pričom GINO ho usmerní, aby použil tie správne slová pre konkrétne objekty, predikáty a subjekty. Na základe takéhoto dopytu potom GINO doplní údaj o hĺbke jazera *Tahoe* do sémantického úložiska. Tento spôsob je veľmi pohodlný pre používateľa. Ak však ten chce urobiť s úložiskom niečo, na čo autori nevytvorili šablónu, musí sa vrátiť k písaniu klasického SPARQL.

Iným prístupom je komunikácia s používateľom pomocou pseudo-prirodzeného alebo úplne prirodzeného jazyka [13, 15, 16]. Tieto prístupy sú založené na extrakcii trojíc z prirodzeného jazyka, čím vytvárajú dopyty pre sémantické úložisko.

4.2 Získavanie trojíc z prirodzeného jazyka

Proces konverzie prirodzeného jazyka na trojice v sémantickom úložisku nie je jednoduchý. Práve z tohto dôvodu sa viacerí autori tomuto problému venovali. Veľmi dobrým zdrojom dát pre takéto pokusy je webová encyklopédia Wikipedia[16]. Prepisu tejto encyklopédie plnej vzájomných vzťahov sa venujú hlavne výskumné tímy tvoriace dátových množín YAGO[17] a DBpedia[18].

4.2.1 DBpedia

Autori tohto projektu [17] sa venujú extrakcii štruktúrovaných dát z webového sídla Wikipedia do podoby sémantického úložiska, čo v konečnom dôsledku prináša úplne nové možnosti pre dopytovanie tejto encyklopédie.

[16] http://www.wikipedia.org/
[17] http://www.mpi-inf.mpg.de/yago-naga/yago/
[18] http://dbpedia.org/About

Spracovanie stránok Wikipedie riešia autori s využitím informačných boxov, tzv. infobox. Tieto infobox-y sú štruktúrovaným spôsobom, ako vyjadrovať dáta o danej entite. Každý typ entity (krajiny, osoby, predmety) má svoju šablónu pre infobox s vlastnými parametrami a typmi hodnôt. Takýmto spôsobom je teda jednoducho možné získať štruktúrované dáta o danej entite. Tieto dáta sa potom vkladajú do ontologickej databázy, kde je už možné pracovať s týmito dátami v podobe grafov.

Ďalším prínosom DBpedie je prepojenie dátovej sady s inými množinami. DBpedia sa stala jadrom iniciatívy LinkedData.

4.2.2 YAGO

Metóda, uplatnená v projekte YAGO [11], používa odlišný spôsob spracovania entít ako DBpedia, aj keď pracuje nad rovnakou dátovou množinou článkov z Wikipedie. YAGO sa sústredí na spracovanie kategórií jednotlivých článkov. Spracúva pri tom hlavne kategórie vo Wikipedii, ktoré obohacujú článok o vzťahy. Príkladom takýchto kategórií sú: *1882_births*, *1239_establishment*... Tieto kategórie YAGO konvertuje na vzťahy *bornInYear*, *establishedIn* a podobne. Takýmto spôsobom postupuje aj pre niekoľko ďalších kategórií. Taktiež pracuje s kategóriami typu *singer*, ktoré hovoria iba o cieľovej entite a vzťah *is* v relácii *?object is singer* je tu zamlčaný.

YAGO sa tiež podrobnejšie venuje tomu, či je daný vzťah tranzitívny, teda platí nielen smerom od objektu k subjektu, ale aj naopak. Pri takýchto vzťahoch definuje aj tranzitívne pomenovania, a teda vytvára aj opačné vzťahy.

YAGO na lepšie pochopenie vzťahov využíva slovník WordNet a špeciálne synonymá jednotlivých slov tak, aby vylúčil pomenovanie rovnakého vzťahu rôznymi slovami. V neskoršej fáze vývoja prepojili autori YAGO aj s dátovou súpravou GeoNames obsahujúcou geografické informácie. V dnešnej dobe je dátová množina YAGO priamo prepojená s dátovou sadou DBpedia, a tak je možné využívať spoločné poznatky oboch dátových množín.

4.3 Predspracovanie vs. fixná dátová množina

Hlavným problémom metód, ktoré využívajú šablónové dopytovanie je to, že šablóny sú dopredu stanovené. To znamená, že pri zmene dátového zdroja je nutné takéto šablóny ručne prepracovať [14]. Pri týchto riešeniach je teda zmena dátovej sady veľmi nepohodlná a vyžaduje veľké úsilie.

Ďalším druhom riešení sú tie, ktoré pracujú iba so samotnou štruktúrou dát [13, 16]. Tieto riešenia síce nie sú závislé na dátovej množine, nad ktorou operujú, no problémom je, že dopyty nie je možné písať v prirodzenom jazyku, ale v jazyku cieľového dátového zdroja. V takýchto dátových zdrojoch sa potom môžu nachádzať aj pre používateľa neprirodzené entity a väzby. Tými myslíme slová, ktoré by bežne používateľ pre takúto entitu nepoužil. Napríklad, ak používateľ hovorí o autorovi článku, je malá pravdepodobnosť, že použije slovo *osoba*, ktoré však tohto autora v databáze môže reprezentovať.

Riešenia, ktoré sú pre používateľov najprirodzenejšie, sú založené na predspracovaní s obohatením slovníka dátového zdroja [15]. Problém, ktorý tieto metódy riešia, sa nazýva

slovníkový problém (angl. Vocabulary problem) [18]. Pri takomto riešení má používateľ veľmi flexibilné možnosti písania dopytov. Popritom výmena dátovej množiny je jednoduchá a vyžaduje iba opätovné spustenie predspracovania dátového zdroja.

4.4 Zhrnutie

V tejto kapitole sme identifikovali niektoré problémy, ktoré sa týkajú prepojených dát a špeciálne vyhľadávania v nich. Prvým poznatkom je nevyhnutnosť predspracovania dátového zdroja, bez ktorého nie je možné pohodlné písanie dopytov ani výmena dátových zdrojov. Práve tento krok je kľúčovým prvkom nášho riešenia.

Okrem toho vylepšujeme proces tvorby dopytu tým, že používateľovi napovedáme ďalšie výrazy, ktoré môže zadať. Robíme to podobne, ako je tomu pri šablónových metódach. Pokiaľ ide o samotné šablóny, našou jedinou šablónou je v riešení prirodzený jazyk v angličtine. Následne nám pri jeho transformácii pomáha nástroj na určenie štruktúry vety. Kombináciou písania v prirodzenom jazyku a napovedania používateľom dosahujeme presnejšie písanie dopytov, čo má za dôsledok lepšiu kvalitu výstupu.

Kapitola 5

Existujúce metódy vyhľadávania v prepojených dátach

V tejto kapitole sa venujeme riešeniam, ktoré sa zaoberajú dopytovaním prepojených dát. V kapitole postupne prejdeme od jednoduchších riešení až po tie komplexné. V závere kapitoly potom zhodnotíme klady a zápory týchto metód v prehľadných tabuľkách.

Ako sme už načrtli v predchádzajúcej kapitole, spôsob dopytovania je momentálny problém prepojených dát. Bežní používatelia nemajú dostatočné znalosti nielen o grafových štruktúrach tu používaných, ale aj o štruktúre dopytovaných dátových množín.

V dnešnej dobe sa na dopytovanie používa jazyk SPARQL, ktorý sme opísali v úvodnej časti tejto práce. SPARQL je však jazyk, ktorý je prístupný pre expertov v oblasti sémantického webu. Bežní používatelia bohužiaľ nie sú schopní písať dopyty v tomto jazyku nielen pre štruktúru dát, ale aj pre jeho zložitosť.

Aj z toho dôvodu je nutné pre rozšírenie sémantického webu vytvoriť spôsob, ktorý by používateľom priniesol požadované možnosti dopytovania nad sémantickými dátami. Tento spôsob musí byť dostatočne jednoduchý a zrozumiteľný pre nich, a preto je snaha priblížiť ho klasickej ľudskej reči. V nasledujúcej kapitole sa preto pozrieme na vlastnosti dnes prístupných metód na konštruovanie dopytov v oblasti prepojených dát.

5.1 Sig.ma

Metódu Sig.ma [10] sme už spomínali v predchádzajúcej kapitole, v ktorej sme hovorili o spájaní grafov z viacerých zdrojov. Tento projekt obsahuje taktiež aj možnosti vyhľadávania nad sémantickými dátami. Jeho veľkou výhodou je to, že hľadá nad spojenými zdrojmi nástrojov Sindice a OKKAM.

Samotné vyhľadávanie začína zadaním kľúčového slova do vyhľadávača. V tejto chvíli začne Sig.ma prehľadávať všetky poskytnuté zdroje od zhromaždovacích služieb. Tieto zdroje spája dohromady a pomocou heuristickej normalizácie a ďalším preddefinovaným termínom zhlukuje tieto zdroje dohromady. Používateľ v konečnom dôsledku vidí zoznam všetkých predikátov a objektov spojených so zadaným kľúčovým slovom.

Rozšírením vyhľadávania pomocou kľúčových slov je presné definovanie atribútov, ktoré chceme o zadanom kľúčovom slove zistiť. Teda napríklad, ak zadáme do vyhľadávania „*name@Barak Obama*“, získame iba atribúty dopytu na *Baraka Obamu*, ktoré obsahujú výraz „*name*“.

5.1.1 Zhodnotenie metódy

Nevýhodou tohto vyhľadávania je fakt, že Sig.ma nepodporuje zložitejšie dopyty a nesnaží sa ich sémanticky pochopiť. Každý dopyt vníma ako jedno kľúčové slovo, ktoré sa následne snaží nájsť v dátových zdrojoch. To znamená napríklad to, že ak zadáme dopyt „Barak Obama’s wife“, nedostaneme profil Michelle Obama. Dátový zdroj Sig.ma má podrobnú databázu vlastností pre všeobecné zdroje a tak by pomocou jednoduchých dopytov, hoci aj v podobe trojíc, vedel vyťažiť z dát oveľa viac ako to momentálne robí.

5.2 Sindice

Službu Sindice [19] sme takisto spomínali v predchádzajúcej časti. Aj ona však obsahuje možnosti vyhľadávania v spojených zdrojoch. Základné vyhľadávanie prebieha podobne ako to bolo v predchádzajúcom prípade, pomocou kľúčových slov. Práve preto má rovnaké nedostatky a nesnaží sa dopyt analyzovať v podobe prepojení.

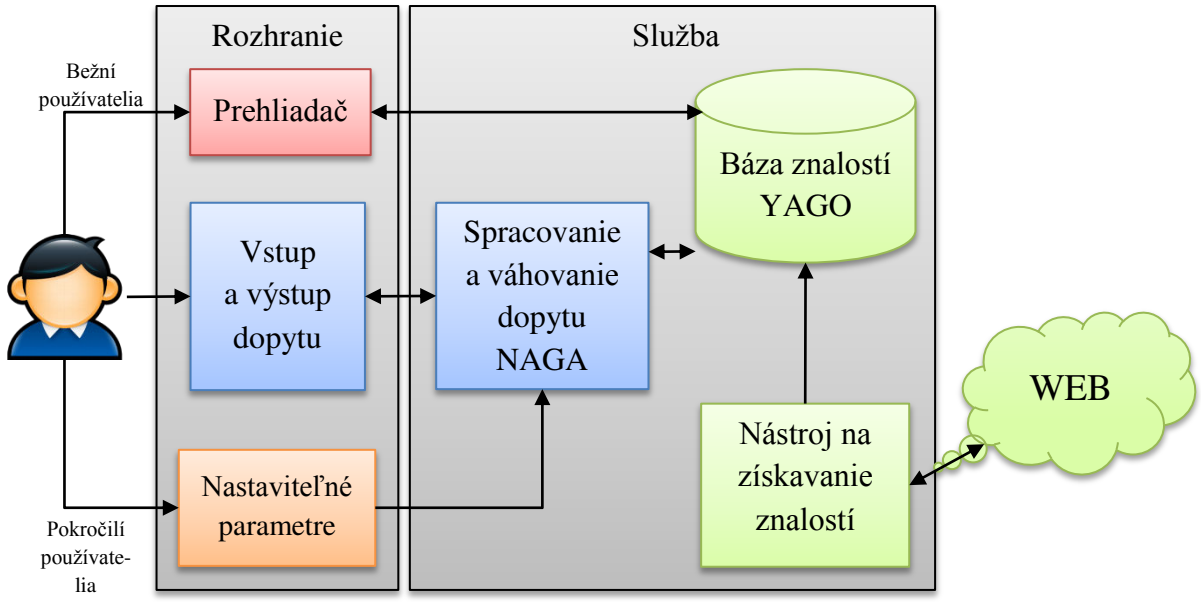
Sindice okrem toho obsahuje aj formu vyhľadávania pomocou výrazov v jazyku SPARQL. V tejto časti je veľmi pozitívnou vlastnosťou to, že samotné rozhranie obsahuje aj niekoľko preddefinovaných dopytov v tomto jazyku. Používateľ si preto môže vybrať dopyt, ktorý sa najviac približuje jeho požiadavke, a následne ho môže upraviť. Okrem toho si môže zvoliť aj formu výstupu, ktorá môže byť klasickým HTML, JSON, RDF ...

5.2.1 Zhodnotenie metódy

Takýto spôsob dopytovania je vhodný najmä pre expertov, ktorí poznajú jazyk SPARQL, pretože ich navádza k rýchlejšiemu vytvoreniu dopytu. Problémom však ostáva to, že bežný používateľ nedokáže takýto dopyt upraviť do svojej podoby, a preto mu táto metóda vyhľadávanie neuľahčí.

5.3 NAGA

V predchádzajúcej kapitole sme hovorili o metóde YAGO, zásluhou ktorej bola vytvorená prepracovaná databáza prepojených dát. Presne tento dátový zdroj využíva aj metóda NAGA [20]. Architektúra výsledného systému YAGO-NAGA je zachytená na obrázku 11.



Obrázok 11 Architektúra nástroja YAGO-NAGA [20]

Systém pozostáva z jednoduchého rozhrania, kam môže používateľ písať dopyty (o nich budeme bližšie hovoriť v ďalšej časti textu). Následne po odoslaní ide daný dopyt na spracovanie pomocou metódy NAGA, ktorá vo výsledku vytvorí SPARQL dopyt nad dátovou množinou YAGO.

Používateľ má okrem toho aj ďalšie možnosti upravovania dopytu, ktorý zadal pomocou ďalších parametrov, a takisto si môže pozrieť čisté dáta v báze znalostí. Na pravej strane obrázku je ešte znázornený nástroj na získanie bázy znalostí, ktorý tieto znalosti získava z prostredia Webu (Wikipedia, WordNet, GeoNames).

NAGA sa vyznačuje dvoma základnými metódami:

- spracovanie,
- váhovanie dopytov.

Vďaka nim dostáva používateľ podrobné výsledky podľa ich relevantnosti.

5.3.1 Spracovanie dopytu

Spracovanie dopytov prebieha na základe analýzy trojíc. Používateľ má definovaný zoznam niekoľkých desiatok vlastností, ktorý môže pri vytváraní dopytu používať. Okrem toho je možné používať premenné, za ktoré analyzátor považuje všetky slová so znakom $ na ich začiatku.

V NAGA sú definované 3 druhy dopytov [21]:

- objavovacie (angl. discovery),
- regulárne (angl. regular),

- súvislostné (angl. relatedness).

Objavovacie dopyty

Objavovací (angl. Discovery) dopyt pomáha používateľovi nájsť chýbajúcu informáciu. Používateľ poskytne istý fakt, ktorý vie o entite a vďaka nej entitu NAGA objaví. Toto vysvetlíme na nasledujúcom príklade:

```
Max_Planck bornOnDate $x
$y bornOnDate $z
$z sameYear $x
```

Hore uvedený príklad definuje príkaz na vyhľadanie osoby, ktorá bola narodená v ten istý rok ako sa narodil Max Planck. Sú tu použité 3 premenné, pričom jednotlivé príkazy na seba nadväzujú a ich postupným vykonaním je možné získať hľadanú entitu.

Regulárne dopyty

Ďalším typom dopytov sú regulárne (angl. regular) dopyty. Týmito dopytmi je možné získať odpoveď na jednoduchú otázku. Napríklad:

```
Zidane isa $x
```

Tento príkaz sa dotazuje na to, čím je osoba pod menom „*Zidane*“. Samotný výraz má niekoľko možností ako odpovedať, a teda zistiť premennú x. X môže byť „osoba“, „futbalista“ a mnoho ďalších. O riešení takýchto situácií v metóde NAGA budeme hovoriť v nasledujúcej podkapitole *váhovanie*.

Súvislostné dopyty

Posledným typom dopytov sú dopyty súvislostí. Tieto hľadajú podobné súvislosti medzi entitami. Tieto výrazy sa zapisujú pomocou výrazu *related* alebo *conected*. Napríklad:

```
Niels_Bohr related Albert_Einstein
```

V tomto dopyte sa teda pýtame, čím sú prepojené entity Albert Einstein a Niels Bohr.

NAGA poskytuje aj ďalšie možnosti pre skúsených používateľov. Títo majú možnosť upravovať parametre dopytu a tak prispôsobiť dopyt tomu, čo chcú vo výsledku vidieť.

5.3.2 Váhovanie

Keď sa pozrieme na posledný príkaz, jeho spracovanie by nemuselo byť jednoznačné. Tu je dôležité váhovanie výsledkov ako ďalšia pomôcka. Keďže všetky predikáty majú v systéme svoju váhu, vieme zistiť, aká je relevancia týchto dvoch entít. Vďaka váhovaniu sa teda dostane na popredné priečky to, že obaja dostali *Nobelovu cenu* pred faktom, že *Einstein* sa narodil vtedy, keď *Bohr* zomrel.

YAGO-NAGA je zaujímavý projekt v tom, že stavia na dobre prepracovanej báze znalostí YAGA. Takisto aj písanie dopytov formou, ktorú prináša NAGA, je oveľa jednoduchšie ako písanie dopytov v jazyku SPARQL.

5.3.3 Zhodnotenie metódy

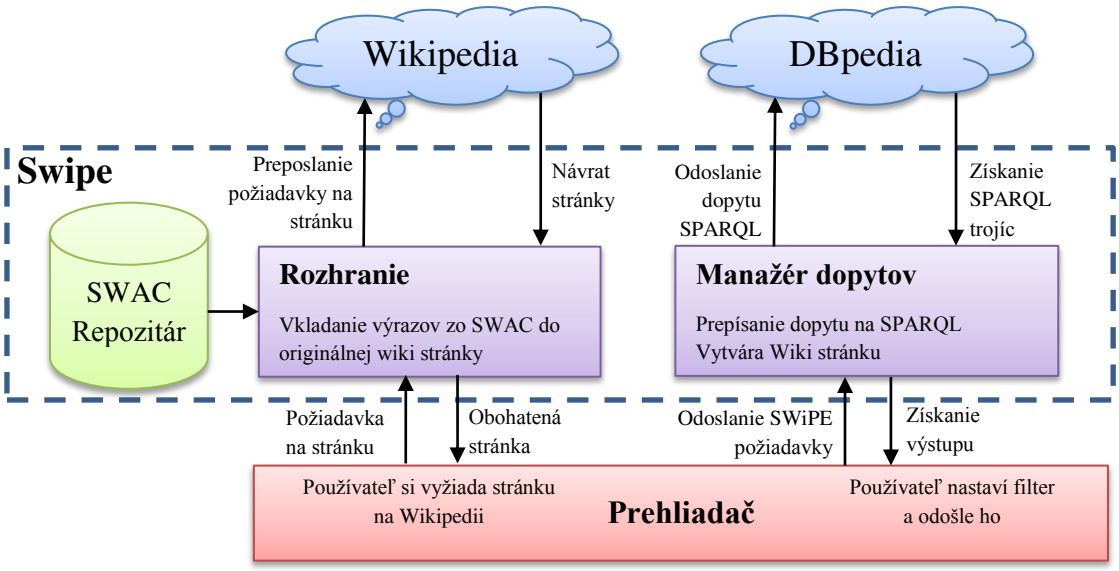
V samotnej metóde NAGA sa pozitívne dá hodnotiť aj možnosť následného upravovania zadaného výrazu používateľom. Veľmi pozitívnou vlastnosťou je aj použitie váhovania na zoradenie výsledkov, čo prináša výhody v podobe zobrazenia relevantných výsledkov na horných pozíciách výsledku vyhľadania.

Stále sú však tieto dopyty náročné pre bežného používateľa webu, ktorý by chcel takéto dopyty písať. Nástroj je teda v konečnom dôsledku určený pre odbornú verejnosť, ktorá má už skúsenosti s prepojenými dátami. Istá nevýhoda prispôsobivosti sa skrýva aj v tom, že v dopyte pre NAGA sa dá použiť iba výraz zo zoznamu predikátov. Toto môže byť prekážkou aj v prípade použitia inej dátovej množine, kde sa nachádzajú iné predikáty.

5.4 SWiPE: Searching Wikipedia By Example

Úplne iný spôsob vyhľadávania v prepojených dátach ponúka metóda SWiPE [22]. Táto metóda pracuje priamo so stránkami známeho portálu Wikipedia. Tieto stránky najskôr obohacuje o výrazy zo svojej internej databázy (SWAC) a následne zobrazuje používateľovi.

Používateľ dostáva teda na výstupe klasickú stránku Wikipedie, na ktorú je zvyknutý. Následne je možné na tejto obohatenej stránke prepísať fakty. Fakty sa líšia podľa objektu, na ktorom sa používateľ nachádza. Napríklad v profile osoby môže prepísať meno, dátum narodenia, počet detí a podobné atribúty. Následne SWiPE dovoľuje vyhľadať dáta pomocou DBpedii podľa zadaných kritérií. SWiPE v tejto chvíli vytvára SPARQL dotaz, v ktorom zadáva limit na zadané parametre. Celý tento scenár fungovania je zobrazený na obrázku 12.



Obrázok 12 Architektúra systému SWiPE obohacujúceho klasickú stránku wikipédie o možnosti sémantického vyhľadávania [22]

V prvom kroku si teda používateľ vyžiada stránku Wikipédie. Tá prechádza cez metódu SWiPE a je obohatená o možnosti editovania predikátov, ktoré sa nachádzajú v databáze

SWAC. V druhom kroku môže používateľ prepísať predikáty nájdené na stránke. Príkladom je prepísanie krstného mena na „Mike“ a zahrnutie do dopytu.

Po odoslaní takejto požiadavky prichádza na rad jadro metódy, ktoré jednoduchým spôsobom skonštruuje SPARQL dopyt zo zahrnutých predikátov. Následne je poslaný tento dopyt na DBpediu, ktorá odpovie a SWiPE z nej vykonštruuje zoznam výsledkov.

5.4.1 Zhodnotenie metódy

Tento spôsob prehliadania je pre používateľa veľmi jednoduchý na použitie. Metóda SWiPE je však veľmi mladá a vývoj na nej je v ranných fázach. Preto je prototyp momentálne menej použiteľný. Tento spôsob sa však môže dobre uplatniť v istých situáciách, kedy používateľ vyhľadáva podľa zadaných parametrov. Napríklad pre vyhľadávanie vo Wikipedii je tento spôsob veľmi dobre uplatniteľný. My však chceme riešiť problém dopytovania pomocou pseudo-prirodzeného jazyka, zatiaľ čo táto metóda skôr pripomína fazetové vyhľadávanie. Práve preto by ju bolo možné využiť ako obohatenie nášho vyhľadávania v budúcnosti.

5.5 OWLPath

Ďalšou metódou zaoberajúcou sa vyhľadávaním nad prepojenými dátami je OWLPath [13]. Tento je založený na myšlienke postupného vytvárania dotazu používateľom. OWLPath ponúkne používateľovi možnosť písania dopytu, pričom pomocou návrhov ho usmerňuje k napísaniu správneho výrazu. Návrhy, ktoré doručuje k používateľovi, formuje z vlastností alebo objektov, ktoré sa nachádzajú v danom dátovom zdroji. Používateľ tak formuje dopyt z mien jednotlivých vlastností a objektov.

Samotný OWLPath sa skladá zo štyroch základných častí:

- rozhranie,
- navrhovateľ,
- SPARQL generátor,
- kontrolór gramatiky.

Okrem toho pravdaže spolupracuje s repozitárom prepojených dát pomocou manažéra ontológie, ktorého funkcie využívajú ostatné časti systému.

5.5.1 Rozhranie

Rozhranie OWLPath je veľmi dôležitá súčasť, ktorá pomáha používateľovi sformulovať dopyt. Následne ho takisto odosiela a spracúva jeho výstup. Tvorba dopytu je však plne v jeho réžii, pretože dovoľuje používateľovi vyberať nasledujúcu časť dopytu. Rozhranie okrem toho dovoľuje aj používať porovnávacie funkcie, ako väčší, menší, rovný alebo pracovať so zložitejšími dátovými typmi, ako sú dátumy.

5.5.2 Navrhovateľ

Navrhovateľ je zodpovedný za vytvorenie zoznamu možností, ktoré sa zobrazia používateľovi v rozhraní. Tieto sa generujú podľa kontextu aktuálne vytváraného dopytu. V prípade, ak napríklad vyberá predikát, zobrazia sa mu všetky vlastnosti daného objektu.

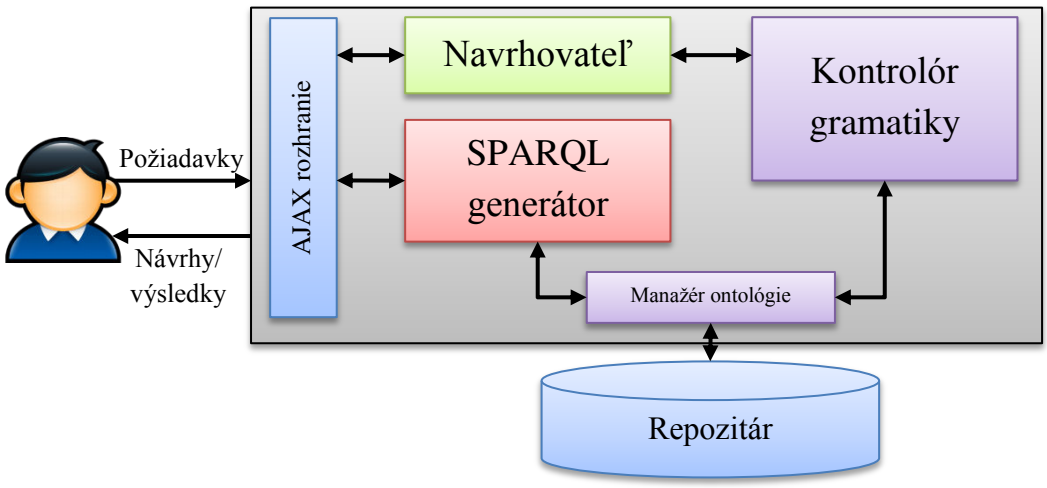
5.5.3 Kontrolór gramatiky

Kontrolór overuje správnosť zadávaných slov v dopyte. Okrem toho prináša navrhovateľovi možné pokračovania vety. Výsledný dotaz takisto overuje a stará sa o to, aby bol spracovateľný SPARQL generátorom. Dotazy preto overuje cez databázu prepojených dát, nad ktorou prebieha vyhľadávanie.

5.5.4 SPARQL generátor

Je hlavnou častou systému, ktorá zo zadaného vstupu vytvára SPARQL dotaz na ontologickú databázu. Keďže navrhovateľ celý čas viedol používateľa, dotaz je momentálne spracovaný v podobe trojíc. Generátor tieto trojice spracuje a vytvára z nich dotaz v jazyku SPARQL.

Schéma celého tohto systému je zachytená na obrázku 13. Metóda vo svojej implementácii využíva možnosti ontologickej databázy Jena.



Obrázok 13 Schéma systému OWLPath [13]

Autori overili navrhnutú metódu v experimente, v ktorom porovnávali rýchlosť tvorby dopytov pomocou OWLPath a pomocou samotného SPARQL jazyka. Keďže jazyk SPARQL bežní používatelia vo väčšine prípadov neovládajú, tento experiment bol realizovaný na vzorke niekoľkých expertov v danej oblasti.

Pri experimente bol porovnávaný čas, za aký sa podarilo expertom vytvoriť dotaz v jazyku SPARQL a koľko trvala jeho príprava pomocou metódy OWLPath. OWLPath bol vo väčšine prípadov úspešnejší a preto autori hovoria o úspešnosti blízko 100%.

5.5.5 Zhodnotenie metódy

Hlavnou výhodu metódy OWLPath je písanie dopytov v jazyku, ktorý sa viac približuje bežnému používateľovi. Dobre navrhnutý je aj spôsob tvorenia dopytu, kedy systém usmerňuje používateľa tak, aby vytvoril správny dopyt. Toto pomáha používateľovi spoznať dátový zdroj, ktorý dopytuje a používa tak jeho názvoslovie.

Systém má však aj niekoľko nedostatkov. Väčšina objektov a predikátov má mená, ktoré bežnému používateľovi nemusia veľa hovoriť. Takisto je systém pri vytváraní dopytu tak-

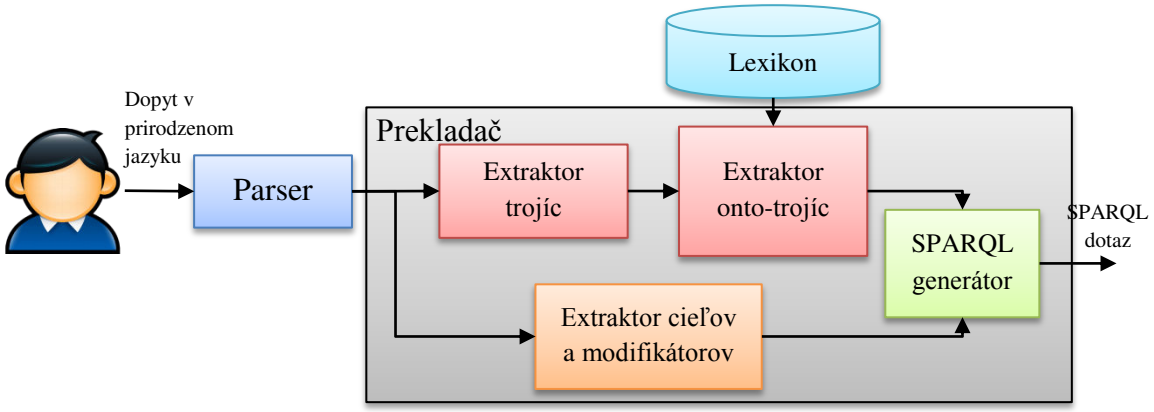
povediac direktívny a nedovolí používateľovi konštruovať dopyty v prirodzenejšom jazyku.

5.6 PANTO

Metóda PANTO (angl. A Portable Natural Language Interface to Ontologies) [15] je už z oblasti pokročilejších metód pre dopytovanie prepojených dát. Jej základnou charakteristikou je využitie riešenia StandfordParser [8] a analýza zadaných viet. Okrem toho, keďže výrazy môžu mať rôznu podobu, využíva aj slovník WordNet.

Metóda PANTO je prenositeľná nad rôznymi ontológiami a preto je jej neoddeliteľnou časťou predspracovanie. V tejto fáze sa vytvára Lexikón, ktorý sa následne využíva pri vytváraní výsledného dopytu. Schému toho, na akom princípe daný systém funguje, je možné vidieť na obrázku 14.

Po zadaní vety do prekladača sa veta analyzuje pomocou nástroja StandfordParser. Následne sa vďaka extraktoru vyťahujú trojice podľa definovaných pravidiel. Tieto trojice sú momentálne reprezentované v prirodzenom jazyku a je ich nutné modifikovať na aktuálnu doménu. To sa robí vďaka ontologickému extraktoru trojíc, ktorý na tento účel využíva Lexikón.



Obrázok 14 Schéma fungovania systému Panto

V lexikóne sa nachádza niekoľko prípustných významov daného slova a vďaka tomu je možné konvertovať zadané trojice na doménu úložiska. Ďalším krokom je vstup do generátora SPARQL dopytov. Do tohto generátora okrem toho vstupujú aj ciele a modifikácie daného výrazu. Samotný generátor potom zo zadaných zdrojov vygeneruje výstupný SPARQL dotaz, ktorý je možné vykonať nad prepojenými dátami a získať tak požadovaný výstup.

5.6.1 Lexikón

Pri vytváraní Lexikónu sa do neho zapisujú záznamy pre entity a ich väzby. Každý záznam obsahuje doplňujúce údaje zo slovníka WordNet. Následne, aby poskytol používateľovi možnosti použitia vlastných slovných spojení, dovoľuje do lexikónu pridávať aj ďalšie výrazy a ich synonymá.

5.6.2 Extraktor trojíc

Extraktor trojíc je jadro samotnej metódy. Jeho hlavná prednosť je v použití nástroja StandfordParser [8] na spracovanie vety. Nástroj sa snaží identifikovať trojice v zadanej vete. To robí vďaka identifikácii *nominálnej fázy* (Nominal Phase), ďalej už iba NP.

Všetky trojice, ktoré získava, sú vo forme <subjekt, predikát, objekt>. Práve preto dve NP v zadanej vete sú označené ako subjekt a objekt v danej trojici. Následne sa z vety extrahuje predikát, ktorý je tvorený prečistenými slovami medzi dvoma NP.

5.6.3 Zhodnotenie metódy

Táto metóda sa radí medzi pokročilejšie riešenia v oblasti dopytovania ontologických dát. Jej hlavnou výhodou je možnosť písania dopytov v skutočne prirodzenom jazyku používateľa. Vďaka tomu vie dopyt pre takýto nástroj písať naozaj aj menej skúsený používateľ.

Nástroj by však stále bolo možné ďalej rozšíriť o ďalšie možnosti. Písanie takéhoto dopytu je veľmi jednoduché, jeho spracovanie je však značne náročné. Preto by bolo vhodné používateľa viesť pri formulovaní takéhoto dopytu, aby bola znížená námaha pre samotné spracovanie. Ďalším vylepšením by bolo umožnenie úpravy vytvoreného dopytu, čo by v konečnej fáze pomohlo vylepšiť aj spracovanie extraktora.

5.7 NLION

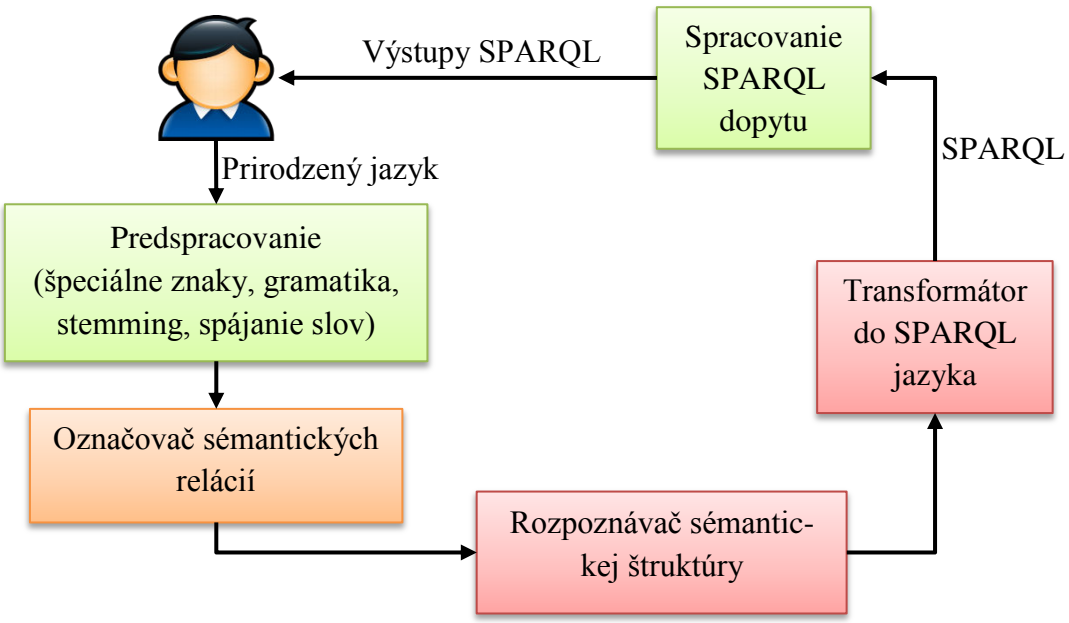
Metóda NLION (angl. Natural Language Interface for querying ONtologies) [16] ja založená na písaní dopytov pre ontologický web prirodzeným jazykom. Na tento účel však nevyužíva StandfordParser, ani žiadne podobné riešenie ako ostatné projekty z tejto oblasti. Vetu sa snaží analyzovať samotná metóda.

Na obrázku 15 sa nachádza schéma zachytávajúca princíp fungovania metódy NLION. Prvým krokom tejto metódy je predspracovanie, ktoré pozostáva z nasledujúcich častí:

- odstránenie špeciálnych znakov,
- gramatická kontrola,
- úprava slov na základný tvar,
- spájanie súvisiacich slov.

V tejto fáze sa teda na začiatku odstraňujú špeciálne znaky a kontroluje sa gramatika danej vety. Následne sa jednotlivé slová premenia na ich základný tvar pomocou hrubej sily (Brut-Force Stemmer). V metóde sa na tento účel nachádza tabuľka všetkých tvarov slov so základným tvarom. Posledná časť predspracovania je spájanie slov, v ktorej sa zisťuje súvislosť slov a tieto sa spájajú dohromady.

Následne sa v metóde označujú sémantické relácie v danej vete a rozpoznáva sa sémantická štruktúra. Tu sa ohodnocujú jednotlivé predpokladané relácie. Posledná časť danej metódy spočíva už v samotnej transformácii vytvorených relácií na SPARQL dopyt, ktorý sa následne vykoná nad ontologickou databázou.



Obrázok 15 Schéma fungovania metódy NLION [16]

V nasledujúcej časti sa pozrieme na podstatné časti tohto riešenia, ktorými sú *označenie sémantických relácií* a *rozpoznanie sémantickej štruktúry*.

5.7.1 Označenie sémantických relácií

V metóde sa sémantická relácia chápe ako označenie relácie medzi slovom v používateľom zadanom dopyte a ontológiou. Tu NLION rozlišuje dva druhy relácie:

- možné označenie konceptu (possible concept tag),
- možné označenie vlastnosti (possible property tag).

V tejto chvíli sa vytvárajú viaceré verzie stromu dopytu. To z dôvodu, že dané slovo môže byť označené aj za možný koncept aj vlastnosť. Napríklad vo vete „*Uveď definíciu Spájaného zoznamu*“ môžeme identifikovať slovné spojenie „*Spájaný Zoznam*“ (angl. Linked List) ako koncept „*SpájanýZoznam*“, ale aj vlastnosť „*máVKódeSpájanýZoznam*“.

5.7.2 Rozpoznanie sémantickej štruktúry

Keďže v predchádzajúcom kroku môže metóda vytvoriť viacero stromov, je nutné tieto stromy upraviť a vytvoriť tak jeden strom riešenia. Táto časť sa preto skladá z nasledujúcich krokov:

- označenie konceptu,
- úprava možných konceptov,
- označenie hodnoty,
- úprava možných vlastností,
- riešenie dvojznačností.

Označenie konceptu

Tu sa rieši zhodnosť konceptov v daných dopytoch. Napríklad, ak používateľ napísal dopyt „*Uveď definíciu zásobníku alebo LIFO*“, tak v tejto časti metóda overí, že „*zásobník*“ je

rodič „*LIFO*“ v prepojených dátach. Preto preformuluje *LIFO* na *zásobník,* čím získa dva rovnaké stromy, z ktorých jeden odstráni.

Úprava možných konceptov

Po predchádzajúcom kroku sa v riešení môžu nachádzať dva stromy s rovnakým základom. Jeden z nich je potom odložený na budúce spracovanie.

Označenie hodnoty

To isté, čo sa môže stať pri koncepte, sa môže stať aj s vlastnosťou. Teda napríklad, ak používateľ zadá výraz „*Uveď zdrojový kód implementácie Zásobníka*“, tak tu figuruje aj *implementácia* aj *zdrojový kód*, ktoré sú zhodné.

Úprava možných vlastností

Následný krok spočíva v preverení vlastností, ktoré sú identifikované v stromoch. Tieto vlastnosti sa preverujú proti ontológii, kde sa zisťuje, či daná entita obsahuje vlastnosť uvedenú v dotaze. Tie, ktoré sa tu nachádzajú, sú označené, ostatné sa odznačujú.

Riešenie dvojznačností

V poslednom kroku sa riešia dvojznačnosti v krokoch. Napríklad, ak sa jeden koncept nachádza ako vlastnosť iného konceptu, tak sa tento koncept a celý jeho strom odstraňuje. Ďalej ak napríklad máme vlastnosť „*aplikácia*“ a „*aplikácia v reálnom čase*“, tak vlastnosť *aplikácia* je zo zoznamu odstránená.

Výstup týchto krokov je následne vložený do SPARQL generátora a je z neho vytvorený samotný SPARQL dotaz. Táto časť je už triviálnou úlohou, lebo jednotlivé stromy sú prevedené na trojice a tie zase na výsledný SPARQL dotaz.

5.7.3 Zhodnotenie metódy

Keďže používateľ môže písať dotazy pomocou tejto metódy v absolútne prirodzenom jazyku, je táto metóda dobre prístupná. Metóda používa veľmi dobrý spôsob reprezentácie jednotlivých výrazov v podobe stromov. Takisto nadväzujúca časť úprav týchto stromov má výhody v dobrom odstránení nesúvisiacich dopytov.

Nevýhodu však je absencia slovníka na odhaľovanie synonymických výrazov, čo komplikuje zadávanie dopytov. Tie preto môžu byť v prirodzenom jazyku, ale takisto musia využívať slovník daného dátového zdroja. Ďalej by bolo dobré do metódy zakomponovať spôsob navádzania používateľa na použitie správnych slov obsiahnutých v dátových zdrojoch.

5.8 Querix

Hlavná myšlienka metódy Querix [14] je v získavaní doplnených informácií od používateľa. Základom celého postupu metódy je najskôr získanie dopytu od používateľa. Ten následne putuje do manažéra ontológie, ktorý využíva nástroje WordNet a StandfordParser [8]. Pomocou nástroja WordNet sa snaží zistiť, o ktorom objekte ontológie používateľ hovorí. StandfordParser slúži na analýzu zadaného dopytu.

Querix na konštrukciu dopytu využíva hlavne predspracované schémy, ktoré vie spoľahlivo pretransformovať do podoby SPARQL a ktoré vie vykonať nad databázou.

Okrem tohto jednoduchého spracovania ešte obsahuje spomínané objasňovanie dopytu používateľom. Querix sa v prípade, ak nevie presne vyhodnotiť dopyt, opýta používateľa, ako má presne chápať daný dopyt.

Napríklad, ak používateľ zadá nasledujúci dopyt „Ktorý štát je v USA najväčší“, používateľ dostane doplňujúcu otázku. V nej sa metóda pýta na metriku, ktorú má na tento účel použiť. V tejto chvíli dáva na výber používateľovi viacero vlastností, ako „najväčšia hodnota populácie“, „najväčšia hodnota veľkosti štátu“, ... Následne objasnený dopyt vykoná na ontologickej databáze a vráti používateľovi požadovanú hodnotu.

V experimente táto metóda dosiahla úspešnosť okolo 87%, pri teste má 215 dopytov. Väčšina z týchto dopytov mala pritom podobnú štruktúru, a teda bol Querix na ňu pripravený.

5.8.1 Zhodnotenie metódy

Querix má zaujímavý spôsob získavania ďalších informácií od používateľa, čím vie vyriešiť veľa nejasností. Takisto je v ňom možné písať dotazy v prirodzenom jazyku. Problémom je, že je pripravený iba na isté druhy dopytov, pre ktoré má šablóny spracovania. Práve preto používateľ nemôže písať dopyty v absolútne prirodzenom jazyku, ale iba v jeho podmnožine, ktorú vie metóda spracovať.

5.9 Porovnanie metód

V nasledujúcej kapitole uvádzame tabuľky, ktoré zobrazujú porovnanie vlastností jednotlivých metód. Tabuľka 1 zobrazuje porovnanie metód, ktoré nezahŕňajú spracovanie prirodzeného alebo pseudo-prirodzeného jazyka používateľa. Tabuľka 2 na druhej strane porovnáva riešenia, ktoré využívajú prirodzený alebo pseudo-prirodzený jazyk v nejakej podobe.

Tabuľky majú rovnakú štruktúru riadkov, a teda v prvých dvoch stĺpcoch definujú spôsob, akým sa konštruuje dopyt a akým sa zadaný dopyt vykonáva nad dátovou vzorkou. Nasledujú stĺpce, ktoré hodnotia samotnú metódu, a teda:

- či (ako) metóda dokáže spracovať prirodzený alebo pseudo-prirodzený jazyk,
- či (ako) napomáha metóda používateľovi pri konštrukcii dopytu alebo ho vedie,
- či (ako) je možné upravovať výsledný dopyt po jeho vytvorení,
- či (ako) sú riešené nejednoznačnosti dopytov, ktoré používateľ konštruuje,
- či je metóda naviazaná na konkrétny slovník dátového zdroja, ako rieši synonymá,
- či (na akej adrese) sa nachádza spustiteľný prototyp a jeho obmedzenia.

Tabuľka 1 Porovnanie metód, ktoré nevyužívajú prirodzený alebo pseudoprirodzený jazyk

	Sig.ma	Sindice	NAGA	SWiPE
Spôsob písania dopytu	Kľúčové slová s možnosťou obmedzenia jednotlivých atribútov.	1. Kľúčové slová. 2. SPARQL dopyt + obsahuje príklady najčastejších dopytov	Vytváranie dopytu v podobe trojíc s použitím premenných s predponou $.	Editovanie hodnoty predikátov na stránke Wikipedie.
Analýza dopytu	Vyhľadanie v názvoch jednotlivých objektov.	1. Vyhľadávanie v názvoch. 2. Vykonanie SPARQL dopytu.	Transformácia trojíc na SPARQL dopyty.	Vytvorenie SPARQL dopytu z editovaných predikátov.
Využitie prirodzeného jazyka	Nie (kľúčové slová)	Nie (kľúčové slová, SPARQL)	Nie (trojice)	Nie (hodnoty predikátov)
Napomáhanie používateľovi pri konštrukcii dopytu	Nie (niekoľko príkladov pri vyhľadávaní)	Čiastočne (SPARQL šablóny)	Nie (niekoľko príkladov pri vyhľadávaní)	Áno (jasné definovanie vlastností, ktoré môže používateľ dopytovať)
Možnosť úpravy dopytu	Nie	Nie	Áno (atribúty vyhľadávania)	Nie
Riešenie nejednoznačností	Nie	Nie	Áno (váhovanie)	Nie
Väzba na slovník dátového zdroja	Áno	Áno	Áno (vymenované možné väzby, ktoré sa dajú použiť medzi entitami)	Áno
Prototyp	Áno[19]	Áno[20]	Áno[21] (väčšinou však nefunkčný)	Áno[22] (v rannom štádiu vývoja)

[19] http://sig.ma/
[20] http://sindice.com/
[21] http://www.mpi-inf.mpg.de/yago-naga/naga/demo.html
[22] http://swipe.i-mozart.com/wiki

Tabuľka 2 Porovnanie metód, ktoré využívajú prirodzený alebo pseudoprirodzený jazyk

	OWLPath	PANTO	NLION	Querix
Spôsob písania dopytu	Vytváranie dopytu pomocou nápovedí relevantných k písanému textu.	Vytváranie dopytu v prirodzenom jazyku.	Vytváranie dopytu v prirodzenom jazyku.	Šablónové dopyty v prirodzenom jazyku.
Analýza dopytu	Už počas konštrukcie dopytu. Výsledok vytvára SPARQL.	Na analýzu sa používa StandfordParser, čím sa zisťuje základ vety. Používa aj WordNet ako slovník názvoslovia.	Vytváranie stromov dopytu, postupná úprava až k vytvoreniu jednoznačných trojíc. Konvertovanie trojíc na SPARQL.	Podľa šablóny rozpoznanie konkrétneho typu dopytu.
Využitie prirodzeného jazyka	Čiastočne (polo-prirodzený jazyk)	Áno (dopyty sa píšu v úplne prirodzenom jazyku)	Áno (dopyty sa píšu v úplne prirodzenom jazyku)	Áno (iba v konkrétnych prípadoch)
Napomáhanie používateľovi pri konštrukcii dopytu	Áno (každá časť dopytu sa dá vybrať iba z nápovede)	Nie	Nie	Nie
Možnosť úpravy dopytu	Nie	Nie	Nie	Áno (zaznamenáva nejednoznačnosti a opýta sa ako upraviť dopyt)
Riešenie nejednoznačností	Nie	Nie	Nie	Áno (v prípade nejednoznačností poskytuje zoznam možných pokračovaní)
Väzba na slovník dátového zdroja	Áno (často menej zrozumiteľné väzby s _ v notácii)	Nie (využitie WordNet)	Áno	Áno
Prototyp	Nie	Nie	Nie	Nie

5.10 Zhodnotenie metód vyhľadávania v prepojených dátach

V tejto kapitole sme porovnali metódy pre dopytovanie prepojených dát, ktoré sú na dnešnom webe známe. Jednotlivé metódy majú zakomponované riešenie problémov viac, či menej vhodnými spôsobmi. Tieto fakty sme zachytili v tabuľkách v predchádzajúcej podkapitole.

Z porovnania nám vyšlo ako najzaujímavejšie riešenie Querix, ktoré má prepracované možnosti úpravy dopytov. Toto riešenie je zaujímavé aj tým, že samo identifikuje nejednoznačnosti v dopytoch a dokáže používateľovi ponúknuť možnosti ďalšieho postupu. Problém riešenia je však pevná väzba na konkrétne dopyty pomocou šablón, čo pri využití v rámci iného dátového zdroja s iným charakterom dát komplikuje jeho použitie.

Z pohľadu písania dopytov a výmeny dátových zdrojov je najflexibilnejšie riešenie PANTO, ktoré umožňuje písanie dopytov v prirodzenom jazyku. Jeho druhou výhodou je riešenie synonymických problémov slovníkom WordNet.

Z analýzy z porovnávacích tabuliek vyplynuli nasledujúce podmienky na riešenie problému dopytovania sémantického obsahu prirodzeným jazykom:

- väčšia voľnosť pre používateľa pri písaní dopytov,
- nefixovanie slovníka dopytu na slovník dátového zdroja,
- napomáhanie používateľovi pri písaní dopytov,
- riešenie nejednoznačností.

Všetky tieto podmienky sme zohľadnili pri návrhu metódy transformácie dopytov v prirodzenom jazyku do podoby vykonateľnej nad sémantickým úložiskom dát.

Kapitola 6

Ciele projektu

Pre túto prácu sme si stanovili nasledujúci hlavný cieľ:

- Vytvorenie metódy pre vyhľadávanie nad ontologickou databázou pomocou dopytov v prirodzenom jazyku

Na naplnenie tohto cieľa sme stanovili tieto čiastkové ciele:

- Vytvorenie metódy pre predspracovanie dátového zdroja (v anglickom jazyku)
  - Vytvorenie lexikónu tried a vlastností
  - Vytvorenie lexikónu hodnôt
- Spracovanie dopytov v anglickom jazyku
- Prevod prirodzeného jazyka v slovníku používateľa na slovník vybranej domény
- Prevod dopytu v slovníku danej domény na dopyt v jazyku SPARQL
- Spracovanie modifikátorov v dopyte

Okrem nich sme ešte stanovili vedľajšie ciele, ktoré má naše riešenie spĺňať:

- Navrhovanie ďalších možností pri písaní dopytu pre metódu
- Vytvorenie SPARQL koncového bodu (angl. *endpoint*) nad dátovým zdrojom, aby bolo možné vykonávať nad ním dopyty

Kapitola 7

Vyhľadávanie v prepojených dátach pomocou dopytov v prirodzenom jazyku

V tejto kapitole uvádzame návrh metódy pre prevod dopytu zadanom v prirodzenom jazyku na dopyt v jazyku SPARQL. V prvej časti kapitoly opisujeme metódu na spracovanie dátového zdroja do podoby lexikónov. Následne uvádzame metódu na preklad dopytu v prirodzenom jazyku používateľa na SPARQL dopyt. V posledných častiach kapitoly uvádzame fungovanie navrhovateľa a upravovača dopytov.

Pri riešení definovaného problému vychádzame z viacerých vlastností existujúcich riešení. V rámci analýzy sme ako najpokročilejšie riešenie sme identifikovali systém PANTO. Podobne ako v systéme PANTO, aj my používame na analýzu viet zadaných používateľom externý nástroj. S jeho pomocou analyzujeme štruktúru vety, čo využívame pri preklade.

Okrem toho navrhujeme dodatočný krok, ktorým je predspracovanie dopytu pre vyčistenie štruktúry vety. Okrem toho do procesu prevodu dopytov zapájame synonymá a podobné slová, ktoré spresnia zadaný výraz.

Ďalším unikátnym krokom nášho riešenia je spôsob napovedania, ktorý usmerňuje používateľa pri písaní dopytu tak, aby používal podobný slovník, v akom sú reprezentované dáta, čo skracuje spracovanie výrazu a prináša lepšie výsledky vyhľadávania.

Naše riešenie sa skladá z dvoch hlavných častí:

1. predspracovanie,
2. samotná metóda odpovedania.

7.1 Predspracovanie

Ako bolo možné vidieť na metódach uvedených v predchádzajúcej kapitole, kľúčovým bodom takýchto riešení je predspracovanie. Riešenia, ktoré nevyužívajú predspracovanie, pracujú buď s fixnou dátovou množinou alebo sa obmedzujú na prácu so slovníkom, v ktorom je napísaný daný dátový zdroj.

Obidve možnosti prinášajú viacero nevýhod. Pri fixnej dátovej množine je to uplatniteľnosť iba na jeden problém. Pri používaní slovníka dátovej sady je to písanie dopytov v málo prirodzenom jazyku. Je to tak hlavne z toho dôvodu, že dátové sady obsahujú nie úplne prirodzené pomenovania parametrov, ako napríklad *full-name* alebo *has-author*, ktoré sa v bežnom jazyku nevyskytujú.

Práve z týchto dôvodov sme sa v našom riešení rozhodli využiť predspracovanie. Vďaka predspracovaniu vytvoríme z názvov prvkov viac prirodzené názvy, ktoré následne obohatíme o ďalšie alternatívne popisy. Týmto postupom vytvárame nasledujúce lexikóny:

- lexikónu entít a vlastností,
- lexikónu hodnôt.

Lexikóny v neskorších fázach slúžia na rýchle vyhľadávanie v štruktúre s alternatívnymi pomenovaniami a hodnotami dátovej sady.

7.1.1 Lexikón entít a vlastností

Tento lexikón sa vytvára z dátového zdroja, pre ktorý budeme v nasledujúcich fázach chcieť odpovedať na dopyty používateľa. Z dátového zdroja sa extrahujú všetky mená entít a ich vlastností. Entity a vlastnosti budeme ďalej v texte označovať ako *termy*. Termy sa ukladajú do lexikónu spolu s informáciami, z ktorých elementov dátového zdroja pochádzajú.

V nasledujúcej fáze predspracovania je nutné tieto termy obohatiť o možnosti, akými sa dajú v danom jazyku vyjadriť. Tieto alternatívne názvy budeme ďalej v texte označovať ako *popisy*. Popisy sa prvotne získavajú zo samotnej dátovej sady. Popisy sa získavajú v dvoch fázach:

- získanie popisov zo samotného dátového zdroja,
- získanie popisov z externej databázy slov.

Získanie popisov zo samotného dátového zdroja

Vďaka sémantickej štruktúre dát je možné extrahovať veľké množstvo informácií zo samotných dát. Každá trieda a vlastnosť v sémantickom webe obsahuje totiž atribút *označenie* (angl. *label*), ktorý popisuje daný element. Druhým bežným atribútom elementov v sémantickom úložisku je *popis* (angl. *description*), z ktorého extrahujeme kľúčové slová. Okrem toho nám pomáhajú aj informácia o nadtriede daného elementu.

Získanie popisov z externej databázy slov

Ďalšie popisy pre daný element sa získavajú na základe jeho *label* atribútu. Pomocou neho sa hľadá daný element v externej databázy slov. Z tejto databázy sa následne vyberajú všetky synonymá daného slova. Okrem toho sa získavajú aj nadradené a podradené slová do určitej hĺbky.

Na to, aby sme dokázali nájsť ďalšie popisy v externých databázach, je nutné, aby bol daný dátový zdroj dobre pomenovaný. V prípade, ak by náš dátový zdroj obsahoval neurčité väzby typu: väzba1, väzba2, entita1 atd., nie je možné z takýchto väzieb extrahovať ďalšie alternatívne názvy. Používateľ by v takomto prípade musel poznať presné významy väzieb v dátovej množine.

Priradenie váh pomenovaniam

Keďže nie všetky popisy rovnako významne vystihujú daný element z dátovej sady, zadefinovali sme váhy týchto popisov. Váha popisu sa vypočítava podľa toho, z akého zdroja je daný popis získaný. Nami zadefinované váhy sú naznačené v tabuľke 3. Tieto váhy sme najskôr určovali pomocou jednoduchých experimentov nad našou dátovou sadov. Neskôr sme tieto váhy overili v experimente popísanom v kapitole 8.

Tabuľka 3 Váhy priradené ku zdrojom pomenovania entít

Zdroj pomenovania	Váha
Upravený popis entity	1,0
Popis z *label* atribútu	0,9
Popis zo slovníka	0,85
Popis z nadtriedy	0,7
Kľúčové slovo z *description*	0,5

V prípade, ak popis spadá do viacerých kategórií, váhy sa medzi sebou násobia. Napríklad teda, ak ide o popis zo slovníka pre nadtriedy, váha sa vypočíta ako $v_{popis\_zo\_slovníka} \times v_{popis\_z\_nadtriedy} = 0{,}85 \times 0{,}7 = 0{,}595$. Ak sa nájde jeden popis vo viacerých zdrojoch, napríklad v nadtriede, a zároveň v slovníku, vkladá sa do lexikónu slovo s vyššou váhou. Popisy, ktoré majú váhu nižšiu ako 0,4, sa do slovníka nevkladajú, keďže ich relevancia s danou entitou je veľmi nízka. Túto hranicu sme stanovili takisto na základe experimentu, ktorý bude spomínaný v kapitole 8.

Algoritmus 1 znázorňuje hľadanie opisov pre jeden element dátovej sady. Algoritmus najskôr hľadá popisy v samotnom elemente (*označenie* (angl. label), *popis* (angl. description)) pomocou funkcie *getAllElementDecorators*. Následne vďaka funkcii *getAllWordnetDecorators* vyhľadáva popisy v externej databáze slov. Tieto sa hľadajú vďaka väzbám medzi slovami typu nadradený pojem, synonymum a podobne. Po hľadaní prechádza do nadtriedy, pomocou rekurzívneho volania metódy *getAllDecorators*, kde hľadá ďalšie relevantné opisy. Ak je vstupná hodnota váhy nižšia ako nami stanovená hodnota minima, hľadanie končí a vracia sa zoznam opisov s váhami.

Algoritmus 1 Rekurzívne hľadanie opisov pre zadaný element

```
function getAllDecorators(element, iniWeight){
  if(initWeight > STOP_WEIGHT)
  {
    return nil
  }
  else
  {
    def currWeight to initWght * PARENT_WEIGHT
    decorators << getAllElementDecorators(element, currWeight)
    decorators << getAllWordnetDecorators(element, currWeight)
    decorators << getAllDecorators(element.parent, currWeight)
  }
  return decorators
}
```

7.1.2 Lexikón hodnôt

Keďže ontologická databáza má grafovú štruktúru, cesta ku konkrétnej hodnote je relatívne dlhá (2 skoky). Napríklad, keby používateľ chcel zistiť niečo o objekte *historická budova* typu *budova,* musel by pri využití celej štruktúry napísať dopyt „budova s názvom historická budova“.

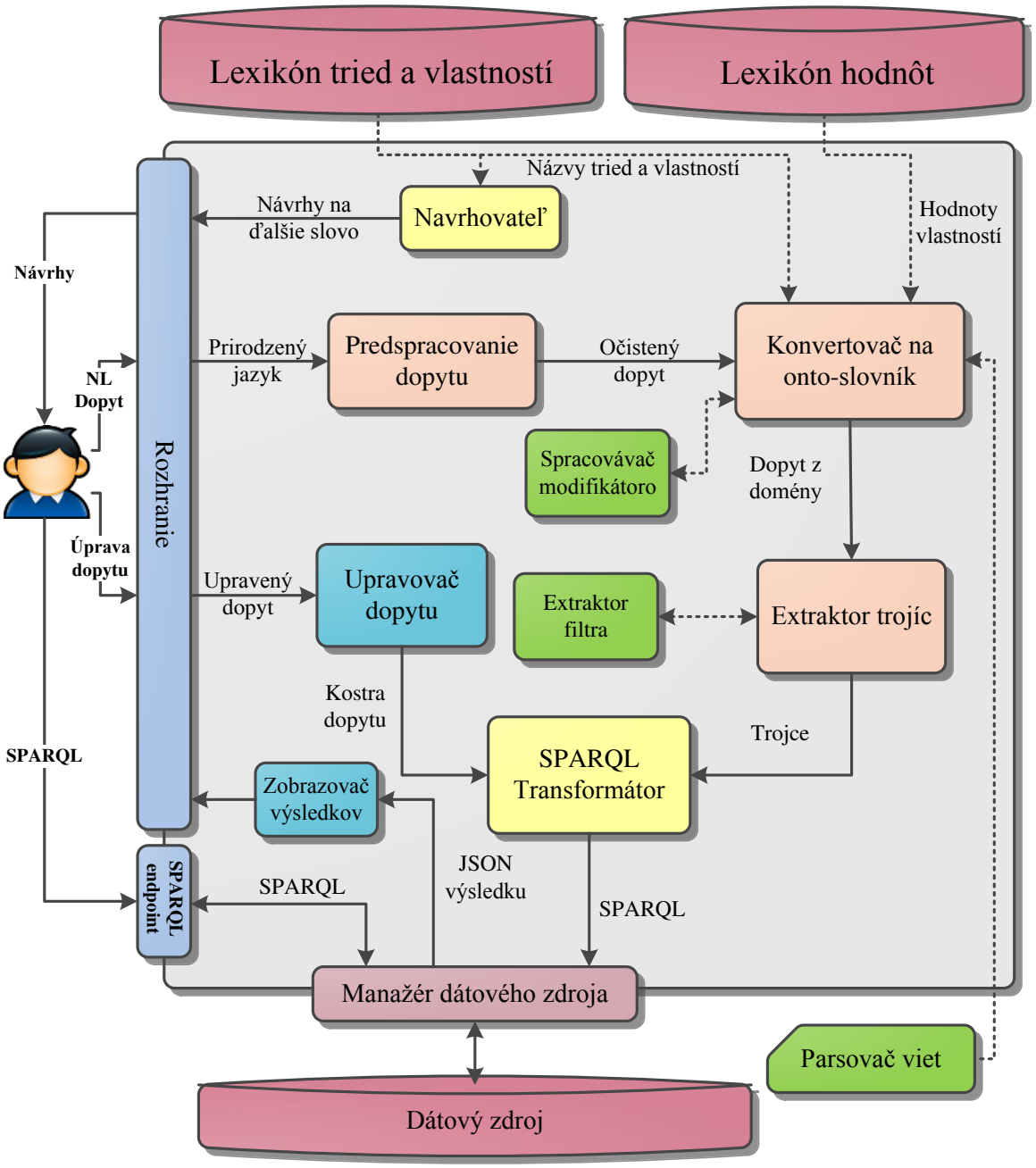
Preto vykonávame extrahovanie hodnôt z jednotlivých objektov v dátovej sade a ich spracovanie do prirodzenej podoby. Hodnoty sa vyberajú z názvu (angl. *label)* a popisu (angl. *description*) objektov.

V dátových súpravách sa okrem toho nezriedka vyskytujú entity, ktoré nemajú žiaden popis v prirodzenom jazyku. Zadefinujme príklad objektu historickej budovy s URI *http://fiit.stuba.sk/priklad.owl#HistorickaBudova*. Tento objekt nemá zadefinované žiadne popisné názvy. Názov sa však dá odvodiť spracovaním jej URI prevodom z maďarskej notácie[23] na prirodzený jazyk, teda v našom prípade „*historická budova*“.

7.2 Spracovanie dopytu v prirodzenom jazyku

Po dokončení predspracovania je možné začať pracovať s dátovým zdrojom pomocou našej metódy. Metóda získava na vstupe dopyt v prirodzenom jazyku. Tento postupnými transformáciami, opísanými ďalej v tejto kapitole, pretransformuje na SPARQL dopyt vykonateľný na ontologickej databáze. Výstupom tohto SPARQL dopytu a zároveň aj metódy je množina dát vrátená klientovi. Nami navrhnuté riešenie je znázornené na obrázku 16.

[23] http://msdn.microsoft.com/en-us/library/aa260976(v=vs.60).aspx



Obrázok 16 Schéma fungovania riešenia (plné šípky symbolizujú presun dát medzi jednotlivými komponentmi, prerušované šípky znázorňujú získavanie pomocných dát z lexikónov a parsovača)

7.2.1 Rozhranie

Rozhranie je pripravené na získanie dopytu od používateľa v prirodzenom jazyku. Po zadaní celého dopytu ho rozhranie posiela našej metóde. Po spracovaní dopytu rozhranie zabezpečuje zobrazenie výsledku dopytu, ktorým je zoznam entít. Pre zvýšenie čitateľnosti výstupu získava rozhranie ďalšie parametre získaných entít, ktoré zobrazuje používateľovi.

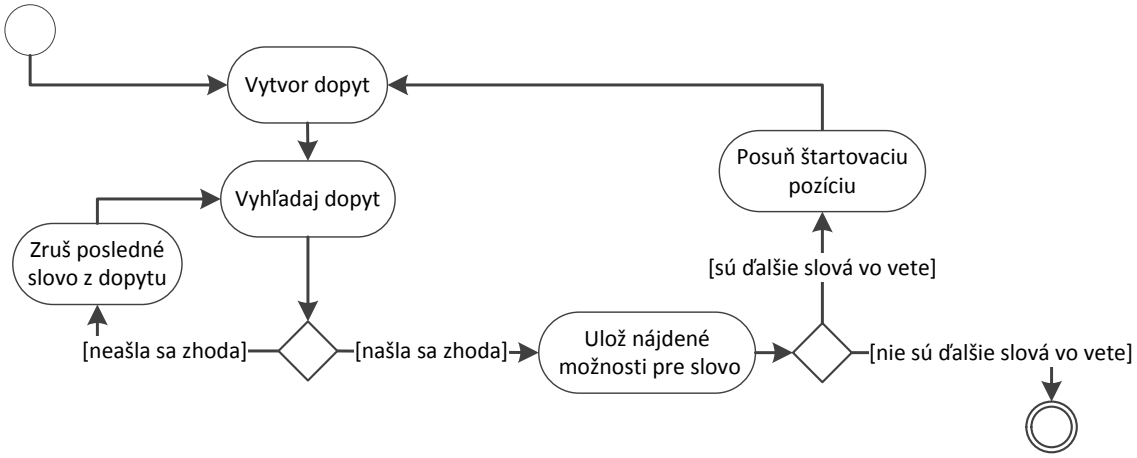
Ďalšou funkciou rozhrania je aj sprostredkovanie návrhov na pokračovanie vety. Tvorba a zobrazenie návrhov bude opísané v podkapitole 7.3.

7.2.2 Predspracovanie dopytu

Predspracovanie je založené na niekoľkých jednoduchých pravidlách. Dopyt je očistený od informácií, ktoré doplnil navrhovateľ dopytov. Takisto sa odstraňujú štylistické znaky a slová bez významu.

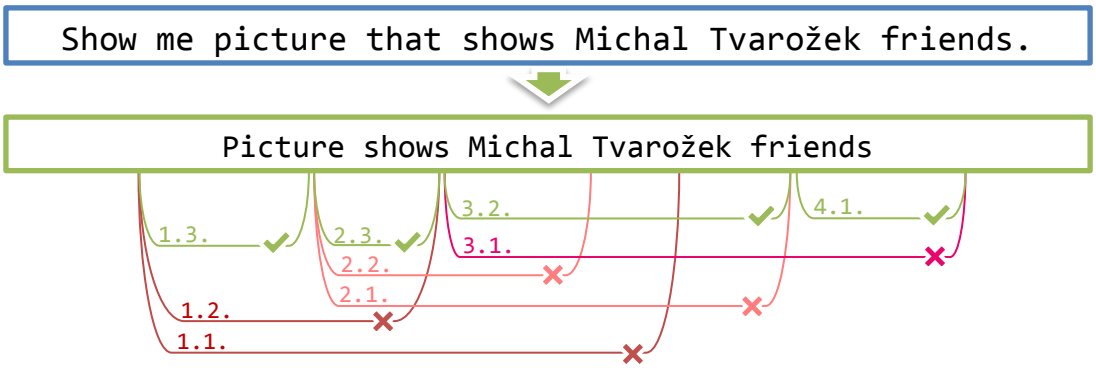
7.2.3 Konvertor na onto-slovník

Konverzia spočíva v postupnom hľadaní adeptov pre jednotlivé frázy v dopyte. Hľadanie prebieha nad lexikónom popisov dátovej sady aj lexikónom hodnôt. Dopyt sa postupne prechádza, pričom sa najskôr snaží nájsť väčšie množstvo slov a následne sa znižuje počet hľadaných slov. Diagram pre tento proces je naznačený na obrázku 17.



Obrázok 17 Diagram znázornuje aktivity pri hľadaní adeptov na preklad jednotlvých čati vety. V prípade neúspešného hľadania sa znižuje počet hladaných slov. Po úspešnom hladaní sa posúva začiatok hladaného dopytu o počet slov nájdeých posledným hľadaním.

Pre lepšie pochopenie procesu sa na obrázku 18 nachádza podrobnejší príklad takéhoto hľadania. Zelené a červené svorky znázorňujú hľadania. Číselná verzia hľadania nad svorkou je rozdelená na majoritnú a minoritnú zmenu. Majoritná zmena označuje posun v používateľom zadanej vete. Minoritná zmena označuje odstránenie jedného slova z dopytu.



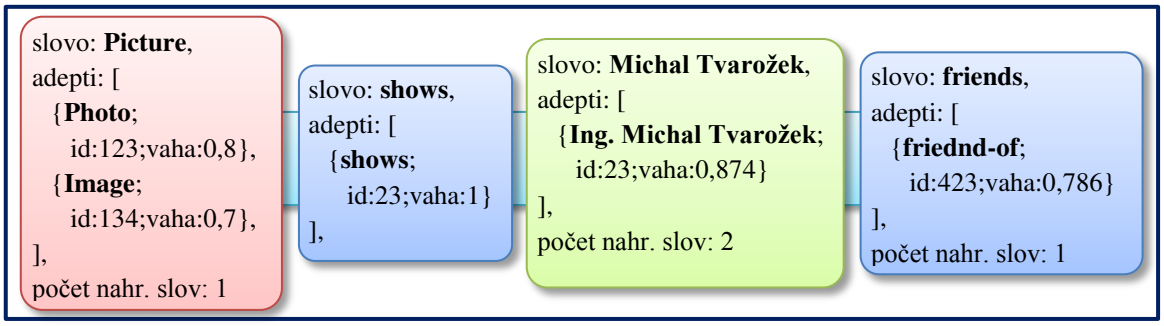
Obrázok 18 Priebeh predspracovania dopytu. Prvá fáza odstraňuje štylistické a bezvýznamové slová. V druhej fáze je znázornené postupné hľadanie slov vo vete. Číslami je označená postupnosť hľadania. Červené × znázorňuje neúspešné hľadanie, zelená ✓ znázorňuje úspešné hľadanie.

Pri tomto preklade sa môže vyskytnúť niekoľko nejednoznačností. Tieto plynú z pomenovaní dátovej sady. V našom príklade je to napríklad slovo „Picture“ (slov. *obrázok*). Toto slovo sa v dátovej sade nevyskytuje, ale lexikón našiel dve podobné entity „Photo“ (slov. *fotka*) a „Image“ (slov. *obrázok*).

V tejto chvíli rozhoduje o výbere konkrétnej entity jej váha. Váha sa vypočítava kombináciou váh z lexikónu, spomínaného v predchádzajúcej podkapitole, a hodnoty zhody vyhľadávaného výrazu s nájdeným slovom. Výstupom tohto procesu je kostra dopytu, ktorá slúži ako vstup do ďalšej fázy.

Kostra dopytu

Kostra dopytu v sebe uchováva spracovaný dopyt. Jednotlivé časti kostry dopytu obsahujú nájdené popisy spojené s prvkami dopytu. Okrem najlepšieho výrazu z lexikónu sa uchovávajú aj ďalšie nájdené alternatívy, ako podkladové dáta pre hľadanie alternatívnych dopytov. Pre priblíženie štruktúry dopytu slúži obrázok 19.

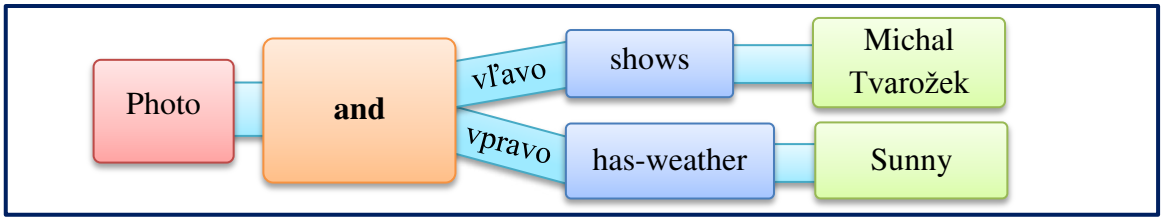


Obrázok 19 Kostra dopytu, s údajmi o jednotlivých častiach dopytu. Farebné znázornenie nahradzuje typ prvku kostry. Červenou sú znázornené entitné časti, modrou väzobné časti a zelenou sú hodnoty.

Spracovanie logických spojok

V samotnom dopyte sa prihliada aj na použitie logických spojok. Tu sme definovali dve spojky a to *a* a *alebo* (keďže metóda pracuje s angličtinou pracujeme so slovami *and* a *or*). Takéto spojky rozdeľujú dopyt na dve časti (ľavú a pravú), čím vytvárajú stromovú štruktúru v dopyte. Spôsob zápisu do kostry vysvetlíme na nasledujúcom príklade znázornenom na obrázku 20:

Picture shows Michal Tvarožek and weather is sunny



Obrázok 20 Schéma kostry dopytu pre logickú operáciu

Odhalenie skrytých prepojení

V prirodzenom jazyku sa často vyskytujú zamlčané väzby. Príkladom takéhoto správania sú hlavne privlastňovacie spojenia, ako *Mária Bielik's articles*, alebo *Michala Tvarožek's photos*. V prípade rozsiahlych dát sa takéto dopyty dajú preložiť rôzne. Napríklad, prvé slovné spojenie sa dá preložiť, ako „*Články, ktoré napísala Mária Bieliková*“, „*Články*

ktoré recenzovala“, „Články, ktoré sú o Márii Bielikovej“, alebo aj *„Články, ktoré patria Márii Bielikovej“*.

Naša metóda rieši tento problém v dvoch rovinách. Prvou rovinou je samotné identifikovanie množiny vyhovujúcich väzieb. Druhá rovina je zase nájdenie správnej väzby. Druhú rovinu naša metóda odstraňuje upravovačom dopytov, ktorý v prípade takejto nejednoznačnosti dá používateľovi na výber, ktorú z väzieb touto otázkou myslel.

Riešenie prvej roviny je realizované v podobe identifikácie nesprávneho vzoru. Nesprávnym vzorom je pre nás kombinácia hodnoty a mena triedy. Takáto dvojica je odhalená v čase spracovávania dopytu. Následne metóda hľadá väzbu medzi entitou a hodnotou pričom obohacuje dopyt touto väzbou.

7.2.4 Extraktor trojíc

V ďalšej fáze spracovania máme teda kostru, ktorá je pretransformovaná do slovníka našej databázy a z nej potrebujeme vytvárať trojice (to vyplýva z povahy jazyka SPARQL). Trojica sa skladá z:

- subjekt,
- predikát,
- objekt.

V subjekte a objekte sa môžu nachádzať premenné, vďaka čomu sú navzájom jednotlivé trojice previazané. V našej metóde sme navrhli spôsob tvorenia premenných pomocou cyklického nabaľovania znakov.

Na obrázku 21 je znázornený postup vytvorenia názvu pre tretiu premennú v poradí. Verzia premennej sa využíva pre logické operácie a jej využitie bude znázornené na konci tejto podkapitoly. Využitie nabaľovania mena premennej vychádza z toho, že jazyk SPARQL nemá obmedzenie pre dĺžku premenných a v prirodzených dopytoch je predpoklad priemerného výskytu štyroch rôznych premenných. Premenná takisto nemá žiaden význam pre koncového používateľa a preto nemusí byť dobre čitateľná.

Obrázok 21 Tvorba premenných pre dopyt nabaľovaním znakov

Trojice sa generujú na základe nami definovaných pravidiel. Tieto pravidlá sú definované vzhľadom na typy častí kostry dokumentu. Týmito časťami sú:

- trieda z dátového zdroja,
- väzba z dátového zdroja (znázornená v algoritme 2),
- hodnota (znázornená v algoritme 3),
- modifikátor.

Pravidlá hovoria o tom, aká trojica sa má vytvoriť alebo kde je nutné vložiť nasledujúcu hodnotu, ak sa nájde daný typ prvku kostry. Tieto pravidlá sú závislé, okrem samotného prvku, aj na trojici definovanej v predchádzajúcom kroku.

Generovanie väzobných trojíc

Postup generovania väzobných trojíc je znázornený v algoritme 2. Algoritmus najskôr vytvára novú časť klauzuly (trojicu) funkciou *add_bind_triple*, pričom funkcia získava URI identifikátor danej väzby. Následne sa do objektu klauzuly nastavuje hodnota naposledy použitej premennej. V prípade, ak sa v trojici nachádzajú hodnoty z predchádzajúceho spracovania, vkladá sa do subjektu táto hodnota, inak sa vkladá nová premenná. V poslednom kroku sa nastavuje samotná väzba danej trojice.

Algoritmus 2 Znázornenie generovania časti SPARQL klauzuly pre väzbu.

```
set new_triple = clause.add_triple()
set new_triple.predicat  = bind_uri
set new_triple.object    = clause.last_used_variable
if (clause.has_future_value){
  set new_triple.subject = clause.pop_future_value()
} else {
  set new_triple.subject = clause.next_variable
}
```

Vkladanie hodnôt do klauzuly

Postup spracovania prvku kostry dopytu sa nachádza v algoritme 3. V postupe sa overuje, či v klauzule exituje nejaká trojica. Ak neexistuje, ukladá sa hodnota na budúce použitie. Naopak v prípade, ak sa v klauzule trojica nachádza, vkladá sa hodnota namiesto poslednej premennej.

Algoritmus 3 Znázornenie spracovania hodnoty. hodnota sa buď ukladá na budúce použitie, alebo sa vkladá miesto poslednej premennej.

```
if(clause.isEmpty){
  clause.set_variable_value(last_variable, this.value)
} else {
  clause.save_future_value(this)
}
```

Extraktor logických operácií

V predchádzajúcom kroku sme ukázali, akým spôsobom zakomponujeme logické operácie do našej kostry. V čase prekladu na trojice je nutné preložiť aj tieto logické operácie. Pri preklade sa môžeme stretnúť s dvoma spôsobmi:

- jedna väzba, dve hodnoty (príklad: *image shows Michal Tvarožek and building*),
- dve väzby, dve hodnoty (príklad: *image shows building and weather is sunny*).

Algoritmus 4 znázorňuje spôsob riešenia pre operáciu *and*. Algoritmus najskôr zisťuje, o aký typ operácie sa jedná. Ak ide o prvý typ postup je nasledovný:

1. kopíruje posledná trojica s väzbou,

2. spracuje sa ľavá strana operácie,
3. vloží sa skopírovaná trojica s väzbou,
4. spracuje sa pravá strana.

V prípade, že ide o druhý typ, postup je nasledovný:

1. spracováva ľavú stranu,
2. zmení verziu premenných,
3. spracováva pravú stranu.

Algoritmus 4 Spracovanie operácie and v dopyte

```
if (right.first.type is BindTriple) {
  left.process()
  clause.variable_version++
} else {
  set triple = clause.clone_last
  left.process()
  clause.variable.next()
  clause.<< triple
}
right.process()
```

7.2.5 SPARQL transformátor

Keďže výstupom z predchádzajúceho kroku je skupina trojíc a *filter,* vygenerovanie SPARQL dopytu je triviálnou operáciou. V tomto kroku sa teda iba všetky časti riešenia dávajú do výsledného dopytu. Samotné trojice sa vkladajú do WHERE klauzuly dopytu, do ktorej sa doplní FILTER klauzula.

Samostatnou časťou je však určenie SELECT klauzuly, a teda predmetu, na ktorý sa používateľ pýta. Z tohto pohľadu sme identifikovali dva rôzne typy dopytov:

1. Klasické dopyty
2. Dopyty cez väzbu

Klasický dopyt vyzerá nasledovne:

Pictures shows Michal Tvarožek

Preklad našou metódou bude vyzerať nasledovne:

```
?photo   rdf:type     image:Photo
?photo   image:shows  ?person
?person  rdfs:label   „Michal Tvarožek“
```

V dopyte sa používateľ pýta na obrázky, teda identifikovanú premennú z prvej časti dopytu. A teda výstupom sú premenné s názvom ?person.

Môže však nastať aj druhý typ dopytu:

Michal Tvarožek friends

Po preklade:

```
?person  rdfs:label       „Michal Tvarožek“
?person  image:friedn-of  ?friend
```

Druhý príklad očakáva teda výstup s hodnotami z premennej ?firiend, ktorá sa nachádza za hodnotou. Identifikácia sa preto robí na základe toho, či ide o prepojovaciu premennú. Prepojovacia premenná je taká, ktorá vystupuje vo viac ako jednej trojici. Ak sa v dopyte nájde neprepojovacia premenná, je táto označená za premennú, ktorá sa vracia používateľovi, ako môžeme vidieť v druhom príklade. Ak sa žiadna premenná nenájde, vracia sa používateľovi prvá vrátená premenná, ako to bolo v prvom príklade.

7.2.6 Záverečná fáza

V záverečnej fáze vystupuje manažér dátového zdroja, ktorý má za úlohu spustiť daný dopyt nad definovaným dátovým zdrojom. Takisto je jeho úlohou vrátiť výsledky tohto dopytu zobrazovaču výsledkov, ktorý sa stará o zobrazenie dát používateľovi.

7.3 Navrhovateľ

Keďže počas písania dopytov môžu v slovníku používateľa vznikať nejednoznačnosti, prvotnou záchranou je navrhovateľ. Tento počas písania dopytu ponúka používateľovi možnosti toho, čo by práve chcel napísať.

Navrhovateľ sa primárne sústredí na štruktúru práve písaného dopytu a podľa nej napovedá používateľovi. Ma sadu pravidiel, ktoré mu hovoria, v akej fáze má vyberať konkrétne typy popisov z lexikónov. Pravidlá napovedania sú znázornené v tabuľke.

Tabuľka 4 Pravidlá napovedania, v hlavičke je typ predchádzajúceho elementu vety podľa, ktorého sa napovedá jej pokračovanie

Predchádzajúce slovo	Začiatok dopytu	Entita	Väzba	hodnota	Logická spojka
návrh typu	- Entita - Hodnota	- Väzba - Hodnota - Entita	- Hodnota	- Logická operácia - Väzba	- Väzba - Hodnota

Pri napovedaní sa navrhovateľ sústredí na dva stavy. Prvý stav je v prípade, ak napovedá po zadaní predchádzajúceho slova ďalšie možné pokračovania. V tejto chvíli vracia zoznam zoradený podľa váh v lexikóne a podľa toho, ako často sa dané slovo vyskytuje v dopytoch. Tu vracia iba vzorku niekoľkých návrhov.

Druhý stav je v prípade, ak používateľ začal už písať časť textu. V tomto prípade sa vracajú iba návrhy, ktoré vyhovujú zadanému textu, zoradené podľa relevancie k práve písanému dopytu.

7.4 Upravovač dopytu

Upravovač dopytu má dve funkcie:

- úprava dopytu na popud používateľa,
- vyriešenie nejednoznačností zobrazením možností prekladu dopytu.

V prípade, ak chce používateľ upraviť dopyt, ponúkneme mu jednoduché rozhranie, v ktorom si bude môcť vyberať entity a väzby medzi nimi pomocou metódy založenej na menu vyberačoch. Touto metódou si teda jednoducho upraví dopyt na takú podobu, akú potrebuje.

V prípade, ak nám nastala nejednoznačnosť, pri ktorej majú obe možnosti podobne vysoké váhy, zobrazujem používateľovi aj alternatívnu možnosť. Alternatívy sa skladajú z kostry dopytu, o ktorej sme hovorili v predchádzajúcej časti. V nej sú uložené alternatívne výrazy pre práve písaný dopyt. Pomocou tejto kostry sa spätne skladajú dopyty s alternatívnymi slovami a dávajú sa používateľovi na schválenie.

7.5 Učiaca sa metóda

V prípade, ak začne používateľ upravovať dopyt, vždy sa ho nenútenou formou snažíme opýtať, či je upravená verzia lepšia ako tá pôvodná. V prípade, ak nám teda povie, že ním upravený dopyt odpovedal na otázku lepšie, upravujú sa váhy pri jednotlivých entitách tak, aby sme v prípade ďalšieho podobného dopytu vedeli odpovedať správne. Do učenia je zahrnutá aj omylnosť alebo zlý úmysel používateľa. Danú novú stratégiu preto musí potvrdiť niekoľko ďalších používateľov na to, aby sa váhy začali používať všeobecne.

7.6 Diskusia k metóde

Nami navrhnutá metóda teda dokáže fungovať na ľubovoľnom dátovom zdroji. Jediné, čo na to potrebuje, je prítomnosť dobre definovaných nazvaných entít v dátovom zdroji (ako sme spomenuli v podkapitole 7.1.1) a spustenie predspracovania.

Metóda je navrhnutá pre anglický jazyk, ak by sme ju však chceli uplatniť na iný jazyk, potrebovali by sme niekoľko nasledujúcich nástrojov:

1. databázu slov a ich vzťahov v danom jazyku,
2. lematizér pre daný jazyk,
3. parsovač pre vety v danom jazyku.

Rozdielom oproti iným riešeniam je, že pomocou nami navrhnutej metódy vytvárame počas predspracovania podrobné lexikóny hodnôt a entít spolu s váhami, vďaka čomu je prvotné spracovanie vety oveľa podrobnejšie. Vďaka týmto lexikónom vieme používateľovi napovedať nasledujúce možné výrazy z dátového zdroja, čím mu umožňujeme písať konkrétnejšie dopyty.

Metódu by bolo možné ešte vylepšiť, keby sa zvolená dátová množina najskôr predspracoval niektorou z metód na hľadanie skrytých väzieb v dátových súpravách. Takýmto spôsobom by sme získali ďalšie väzby, ktoré by pri dopytovaní pomáhali ešte voľnejšiemu písaniu dopytov. Hľadanie skrytých väzieb v dátových súpravách však tvorí samostatný výskumný problém, ktorý je mimo rámca našej práce.

Kapitola 8

Overenie a experimenty

Pre overenie sme navrhli softvérový nástroj, ktorý vykonáva jednotlivé kroky našej metódy. Overenie sme realizovali na viacerých dátových množinách:

- digitálna knižnica ACM[24],
- galéria fotiek Mirai[25],
- anotačný nástroj Annota[26].

V úvodných fázach overenia sme využili voľne dostupnú dátovú sadu vytvorenú z digitálnej knižnice ACM. Na tejto sade sme robili niekoľko prvých experimentov zameraných na nastavenie parametrov navrhnutej metódy.

Obsahom dátovej množiny boli:

- publikácie,
- autori.

Neskôr sme metódu overovali na dátovej sade fotiek pod názvom Mirai. Na túto sadu sme prešli z dôvodu, že množina dát z ACM obsahovala veľmi málo atribútov a väzieb medzi entitami a neobsahovala dostatočne pestrú štruktúru dát. Dátová množina Mirai na rozdiel od toho má pestrú štruktúru a dovolil nám písanie oveľa zložitejších dopytov.

[24] http://datahub.io/dataset/rkb-explorer-acm
[25] http://mirai.fiit.stuba.sk/meia/
[26] http://annota.fiit.stuba.sk/

V databáze sú teda dáta typu:

- fotografie,
- autori,
- obsah fotiek,
    - osoby,
    - budovy,
    - pamiatky,
    - zvieratá,
    - ...
- miesta, z ktorých fotky pochádzajú,
- počasie na fotografiách,
- ...

Nakoniec sme naše riešenie nasadili aj na dátovú množinu pochádzajúcu z projektu TraDiCe. Tieto dáta pochádzali z nástroja Annota a obsahovali nasledujúce dáta:

- autori článkov,
- články.

Všetky nami testované dátové sady obsahujú zložitú dátovú štruktúru, ktorá je rozdelená do niekoľkých ontológií. Napríklad v dátovej sade TraDiCe sa vyžíva ontológia foaf[27] na zápis parametrov autora, zatiaľ čo články a organizácie využívajú ontológiu bibo[28].

8.1 Implementácia rozhrania s navrhovateľom

Pre pohodlné písanie používateľských dopytov sme implementovali rozhranie našej metódy zobrazené na obrázku 22. Toto rozhranie má v sebe implementovaného aj navrhovateľa, ktorý bol navrhnutý v predchádzajúcej kapitole 7.3.

8.1.1 Navrhovateľ

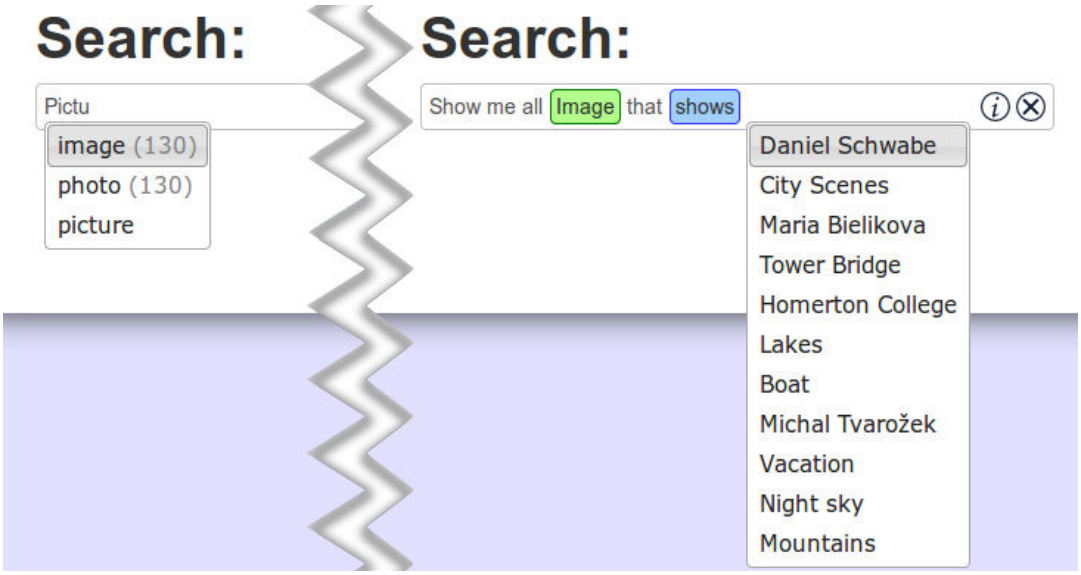
Počas písania dopytu používateľom navrhovateľ zbiera informácie o dopyte a zisťuje, čo práve daný používateľ píše. Zisťovanie prebieha na základe vyhľadávania v lexikónoch. Návrhy sú potom používateľovi ponúkané v podobe vyberacích menu pod vstupným polom pre zapísanie dopytu.

Navrhovateľ zoraďuje výrazy v dopyte primárne podľa relevancie s práve písaným textom. Výnimku tvoria iba slová, ktoré nemajú dostatočnú váhu alebo vystupujú ako popisy pre viaceré elementy z dátovej sady. Príklad je znázornený na obrázku 22 vľavo.

Na obrázku používateľ napísal text *Pictu*. Navrhovateľ vyhodnotil, že najviac sa toto slovo podobá slovu *Picture*. Toto slovo sa však priamo v dátovej sade nenachádza. Nachádza sa tam iba v podobe popisu pre entity *image* a *photo*. Navrhovateľ tieto dva elementy dátovej

[27] http://xmlns.com/foaf/spec/
[28] http://bibliontology.com/

sady zaradil pred samotné slovo *Picture*, aby používateľ vedel vybrať konkrétnejší výraz pre daný prvok a vyhol sa nejednoznačnému dopytu.



Obrázok 22 Rozhranie pre písanie dopytov v prirodzenom jazyku. Na ľavej strane riešenie nejednoznačností, na pravej strane napovedanie pokračovania vety.

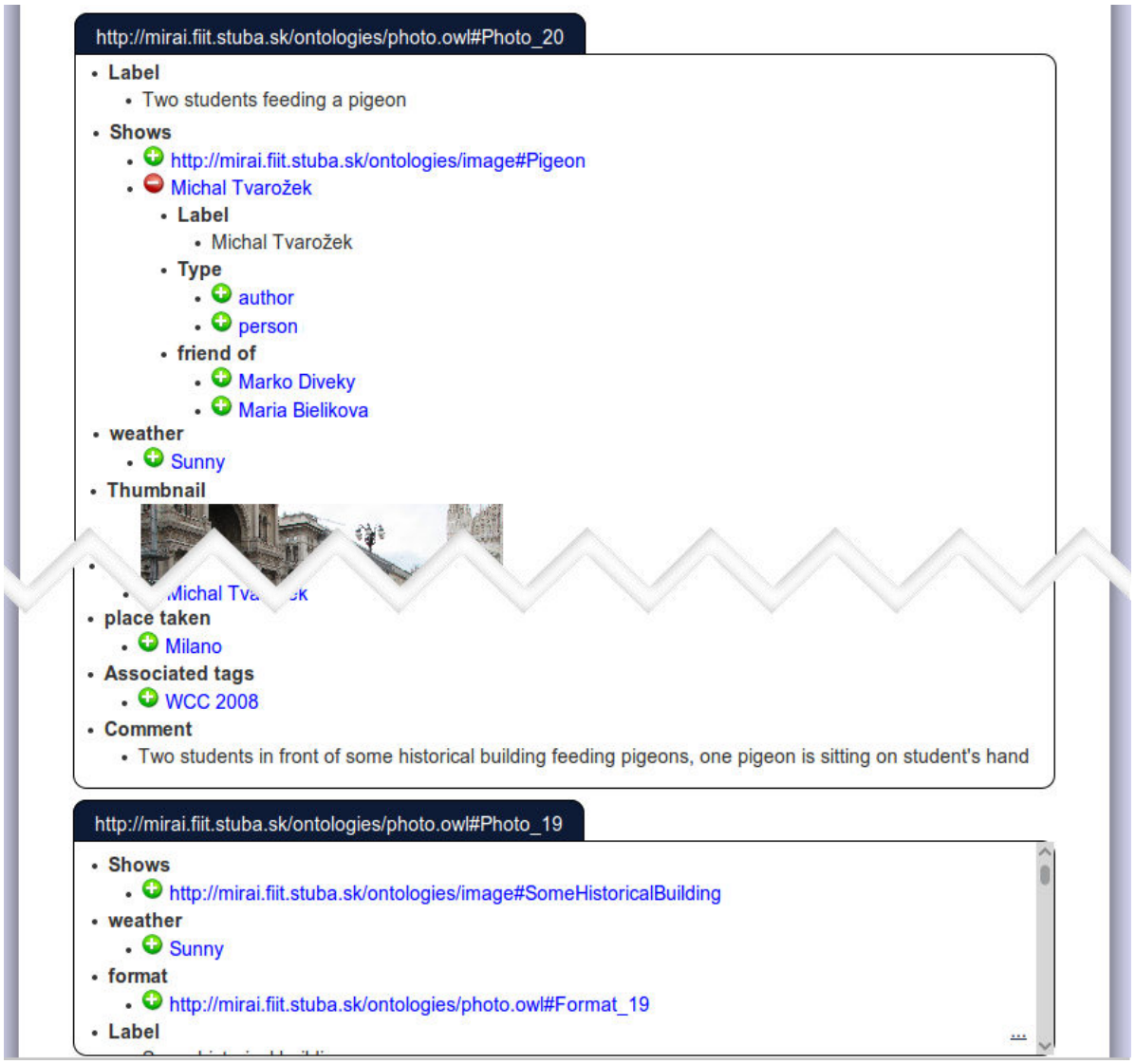
Navrhovateľ okrem toho identifikuje aj časti vety, ktoré by mohli nasledovať bezprostredne za napísaným textom. Po zadaní slova sa preto používateľovi zobrazia možné pokračovania danej vety, ako je znázornene na obrázku 22 vpravo.

8.1.2 Výsledky dopytu

Po vytvorení odoslaní požiadavky s našou metódou sa používateľovi pod vyhľadávacím oknom zobrazia výsledky spracovania tak, ako je znázornené na obrázku 23. Výsledkami je vždy zoznam objektov vyhovujúcich dopytu. Následne naše rozhranie získava pre každý objekt z databázy ďalšie podrobnosti.

Naša metóda vracia výsledky vo forme čo najviac prijateľnej pre bežného používateľa. To sa týka hlavne zobrazenia väzieb, ktoré sa nezobrazujú v podobe URI identifikátorov, ale popisov z *label* atribútov. Podobný postup uplatňujeme aj pri objektoch, avšak vo všeobecnosti platí, že nie všetky objekty dátovej sady sú popísané. Tieto nepopísané objekty sú potom znázornené iba pomocou URI identifikátorov.

Keďže v parametroch objektu sa nezriedkavo nachádzajú ja ďalšie komplexné objekty, vytvorili sme metódu na ich prehliadanie. Ďalšie parametre objektu sa preto zobrazia po stlačení tlačidla + pri niektorom z nich. Táto metóda je ukázaná na príklade atribútu *shows* (slov. zobrazuje), objektu *Photo_20* na obrázku 23. Hodnotou tohto atribútu je objekt reprezentujúci osobu s názvom *Michal Tvarožek*. Po rozbalení tohto objektu sa nám zobrazili ďalšie podrobnosti, ako jeho typ a priatelia.



Obrázok 23 Zobrazené výsledky dopytu s podrobnosťami o jednotlivých objektoch z dátovej sady.

8.2 Fázy overenia našej metódy

Samotné overenie navrhnutej metódy sme realizovali v dvoch fázach:

1. Overenie predspracovania dátovej sady.
2. Testovanie metódy v rámci nástroja umožňujúceho vyhľadávať nad prepojenými dátami.
    a. Testovanie schopnosti používateľov vytvoriť dopyt.
    b. Testovanie prívetivosti používateľskej interakcie.
    c. Testovanie rýchlosti písania dopytov oproti SPARQL.

8.2.1 Overenie predspracovania dátovej sady

Keďže fáza predspracovania bola kľúčovou časťou nášho riešenia, bolo nutné aj to, aby sme jej správnosť overili. Pred experimentom sme teda na dátovej množine Mirai spustili našu metódu predspracovania, ktorá nám vytvorila popisy s váhami pre náš dátový zdroj. Následne sme pripravili jednoduché rozhranie znázornené na obrázku 24. Rozhranie slúži-

lo na zobrazenie slova zvýrazneného modrou farbou a následne k nemu maximálne piatich premiešaných alternatívnych slov.

Obrázok 24 Rozhranie pre experiment overenia váh stanovených našou metódou

Úlohou používateľa bolo alternatívy zoradiť podľa relevancie k hore uvedenému slovu. V žiadnej fáze zoraďovania sme používateľovi nezobrazili naše hodnotenie. V spodnej časti obrazovky bol pripravený malý textový blok, do ktorého bolo možné vpísať názory alebo postrehy používateľa.

K jednému výrazu bolo väčšinou priradených viac ako päť zobrazených alternatívnych slov. V takomto prípade sme popisy zaradili do skupín podľa veľkosti hodnoty a náhodne sme vyberali niekoľko výrazov z každej skupiny. Takto sme chceli zabezpečiť, aby používateľ nemusel zoraďovať všetkých päť výrazov s rovnakou váhou (či už extrémne nízkou, alebo extrémne vysokou), čo by viedlo k deprimovanosti a výsledky by boli skreslené.

Toto rozhranie sme využili na dva experimenty:

1. Zoradenie viacerých popisov pre jeden výraz z dátovej sady.
2. Zoradenie viacerých výrazov z dátovej sady pre jeden popis.

Viacero popisov pre jeden výraz z dátovej sady

V prípade tohto experimentu bolo modré slovo v hornej časti výrazom z dátovej sady a bočný panel obsahoval viacero popisov patriacich k danému popisu. Tento experiment sa týkal hlavne overenia samotných váh, ktorých stanovenie sme opisovali v kapitole 7.1.1.

V spomínanom experimente sme mali 17 zúčastnených používateľov, ktorí pre 30 výrazov vytvorili 336 zoradení (približne 11 zoradení na jeden výraz). Vo výslednom hodnotení sa nám podarilo zistiť, že používatelia sa s naším hodnotením zhodli na 89,3 %. Pri vyhodnocovaní sme za správne zoradenie prehlásili iba to, ktoré bolo presne podľa našich váh, ostatné boli nesprávne.

Viacero výrazov z dátovej sady pre jeden výraz z dátovej sady

Druhý experiment bol opačný ako predchádzajúci experiment. V tomto bolo modré slovo v hornej časti popisom a bočný zoznam obsahoval entity ktoré sa viažu s daným popisom (toto vystihuje aj príklad z predchádzajúcej podkapitoly popisu *picture*, ktorý je priradený aj k *image* aj k *photo*). Takýmto spôsobom sme chceli overiť, či nami stanovené váhy odstraňujú aj problémy nejednoznačných popisov.

V tomto experimente bolo 17 účastníkov, ktorí vytvárali 79 zoradení pre päť popisov (približne 16 zoradení na jeden popis). Vo výsledku sme takisto brali za správne iba tie zoradenia, ktoré boli presne podľa našich váh. V tomto experimente sa nám podarilo dosiahnuť zhodu 68 %. Medzeru, ktorú neodstraňujú naše váhy, dokáže však odstrániť náš navrhovateľ, ktorý, ako sme ukázali v predchádzajúcej podkapitole, napovedá predvolene slová z dátovej sady, čo odstraňuje nejednoznačnosti.

Ďalšie závery z experimentu

Keďže používateľom sme dali dostatočnú šancu sa vyjadriť v podobe voľného textu pod zoraďovaním, dostali sme od nich širokú spätnú väzbu. Tá sa týkala hlavne popisov, ktoré boli s pôvodným výrazom späté veľmi slabo. Z poznámok používateľov sme teda stanovili spodnú hranicu váhy, pri ktorej sa už výraz z predspracovania nedostáva do nášho lexikónu. Táto hranica bola pred experimentom stanovená na hodnotu 0,2 po zrealizovaní experimentu sme ju posunuli na limit 0,4.

8.2.2 Overenie celej metódy

Celkové overenie sme realizovali takisto na dátovej sade Mirai obsahujúcej fotky osoby a predmety na fotkách. Táto časť overenia bola zameraná na vyhodnocovanie:

- navrhovateľa,
    - komfortnosť práce,
    - správnosť prekladu,
- zobrazovača výsledkov,
    - čitateľnosť výstupu,
- výsledného prekladu,
    - správnosť výsledku.

Overovanie prebiehalo prostredníctvom individuálnych rozhovorov s používateľmi. Na začiatku experimentu sme používateľa zoznámili s obsahom dátovej sady. Zoznámenie prebiehalo v slovenskom jazyku, zatiaľ čo obsah dátovej sady je v angličtine. Vďaka rozdielnemu jazyku nebol používateľ ovplyvnený slovami, ktoré sme použili pri opisovaní dátovej vzorky. Používatelia boli počas experimentu rozdelený na dve skupiny, podľa toho, či im pri písaní pomáhal navrhovateľ alebo nie.

Overenie sa skladalo z dvoch experimentov:

1. experiment s bežnými používateľmi,
2. experiment s expertmi na jazyk SPARQL.

Experiment s bežnými používateľmi

Tento experiment sa týkal hlavne správnosti výsledkov a práce s používateľským rozhraním. Jedno sedenie s používateľom trvalo v priemere 25 minút a skladalo sa z nasledujúcich častí:

- získavanie výstupov podľa zadania,
- voľný spôsob dopytovania nad dátovou vzorkou,
- rozhovor s používateľom.

V experimente sme mali 15 používateľov v nasledujúcom zastúpení:

- 5 používateľov pochádzalo z externého prostredia,
- 10 používateľov bolo z informatického prostredia.

Pre prvú časť experimentu sme používateľovi definovali sadu ôsmich otázok, ktoré sú uvedené v prílohe **Error! Reference source not found.**-1.V druhej časti bolo používateľom umožnené písať ľubovoľné dopyty. Tu sme sa snažili zachytiť, aké otázky kládli používatelia. Túto časť využili najviac požívatelia z informatického prostredia. Bolo to spôsobené hlavne databázou fotografií, ktorá obsahovala fotografie z rôznych školských akcií. Na fotografiách boli, ako sme spomínali na začiatku, označené aj osoby, ktoré hľadanie zatraktívnili. Zaujímali ich preto hlavne dopyty na fotografie s konkrétnymi osobami, pričom veľké nadšenie spôsobilo hľadanie kombinácie zvierat a ľudí.

Poslednou časťou rozhovoru bolo interview s používateľom. Používateľa sme sa pýtali na otázky spojené s navrhovateľom a zobrazovačom výsledkov. Každú otázku mohol hodnotiť na stupnici od 1 po 4, kde 4 bolo najlepšie a 1 najhoršie. Otázky a ich priemerné hodnotenie sa nachádza v tabuľke 5.

Tabuľka 5 Otázky a priemerné hodnotami výsledkov od používateľov

Otázka	Modus	Priemer
Myslíte si, že boli návrhy od navrhovateľa relevantné?	4	3,53
Bola práca s navrhovateľom podľa vás intuitívna?	3	2,93
Boli výsledky relevantné k vašej otázke?	4	3,93
Bolo vytvorenie otázky pre vás intuitívne?	3	3,26
Boli podľa vás výsledky dopytu zobrazené prehľadne?	3	2,8

Nižšie hodnotenie v prípade intuitívnosti práce s navrhovateľom sa vyskytol hlavne v prípade používateľov z informatického prostredia. Títo chceli dopyt vyhľadať stlačením klávesu *Enter*. Tento kláves sa však v prípade navrhovateľa používal na výber nasledujúcej časti dopytu.

Keďže objekty typu fotka mali v databáze veľké množstvo parametrov, ich zobrazenie sme riešili zbalenými objektmi, ktoré sa dali v prípade potreby rozbaliť. V otázke prehľadnosti zobrazených dopytov nám používatelia nízke hodnotenie zdôvodnili hlavne tým, že samot-

ná fotografia, ktorá bola jedným z parametrov objektu „*Photo*“, bola v zozname parametrov výsledku zobrazená nízko a bolo nutné rozbalenie objektu fotky.

Rozhranie však bolo navrhnuté, ako univerzálny zobrazovač výsledkov, a preto sme fotografie nijakým spôsobom nepresúvali v rámci výsledkov dopytu.

Experiment s navrhovateľom

Keďže sme v rámci experimentu chceli vyhodnotiť aj navrhovateľa, štyrom používateľom z predchádzajúceho experimentu sme pre prvé štyri otázky vypli napovedanie. Pritom sme merali ich časy a zisťovali sme, ako pomáha navrhovateľ pri písaní dopytov. Výsledky experimentu sú znázornené v tabuľke 6.

Tabuľka 6 porovnanie rýchlosti tvorby dopytu s návrhmi a bez nich

	S navrhovateľom		Bez navrhovateľa	
Otázka	Priemerný čas	Potrebná pomoc	Priemerný čas	Potrebná pomoc
1	29 sec.	Áno (1×)	20 sec.	Nie
2	59 sec.	Nie	74 sec.	Áno (2×)
3	29 sec.	Nie	24 sec.	Nie
4	57 sec.	Nie	107 sec.	Áno (1×)

Tabuľka na ľavej strane zobrazuje priemerné časy písania dopytov jedenástich používateľov, ktorým pomáhal navrhovateľ. Na pravej strane sú časy štyroch používateľov, ktorým nepomáhal navrhovateľ.

Vedľa jednotlivých časov je znázornené, koľkokrát sme používateľom museli pri tvorbe dopytu pomôcť. V skupine s navrhovateľom sme mali jedného používateľa, ktorý sa snažil vyhľadať všetky obrázky dopytom „Image all“.

V skupine bez navrhovateľa sme mali problém pri písaní dopytu pre získanie všetkých fotiek, na ktorých je „Michal Tvarožek“. Tento dopyt písali používatelia ako „*Michal Tvarožek photos*“. Naša metóda tento dopyt preložila ako dopyt na „*Fotky, ktoré vytvoril Michal Tvarožek*“. V dátovej sade sú medzi objektom *Michal Tvarožek* a entitou fotka (angl. *Photo*) dve väzby:

- zobrazuje (angl. shows),
- vytvoril (angl. created by).

Naša metóda vyhodnotila väzbu „*vytvoril*“ ako pravdepodobnejšiu, pretože sa vyskytuje vo väčšom počte trojíc.

Takýto typ problému je však možné odstrániť implementáciou upravovača dopytov, ktorý by používateľovi dovolil zmeniť väzbu v dopyte na väzbu *zobrazuje*. Pri väčšom počte

dopytov na takúto zmenu je metóda schopná naučiť sa a používať predvolene väzbu *zobrazuje*.

Časy písania jednoduchých dopytov boli kratšie v prípade skupiny bez navrhovateľa. Bolo to spôsobené hlavne tým, že v tomto prípade používateľom nebola zobrazená kontextová ponuka a tak nemuseli rozmýšľať nad nasledujúcim pojmom.

Pri písaní dlhších dopytov na tom bola lepšie skupina s navrhovateľom. To bolo spôsobené napovedaním nasledujúcej väzby. Tento spôsob napovedania dovolil používateľovi vybrať väzbu z kontextového menu namiesto toho, aby nad pokračovaním vety musel sám rozmýšľať. Takisto v tomto prípade pomohlo zoraďovanie nápovedí, kedy vkladáme elementy z dátovej sady pred popisy z externého slovníka. Používatelia sa tak hneď opravili a nedošlo k nejednoznačnostiam.

Experiment s expertmi

Na overenie rýchlosti prekladu sme uskutočnili ďalší experiment. Tento experiment bol realizovaný s expertmi z oblasti sémantického webu. Požiadavka na účastníkov tohto experimentu bola schopnosť používať SPARQL jazyk. V experimente sme pre používateľov pripravili štyri dopyty uvedené v prílohe **Error! Reference source not found.**-2.

Experti museli zostaviť dva dopyty:

1. SPARQL dopyt
2. Dopyt v prirodzenom jazyku

Poradie vytvárania dopytov sa menilo. Napr. pre prvý dopyt expert najskôr tvoril dopyt v prirodzenom jazyku a následne v SPARQL jazyku, zatiaľ čo pre druhý dopyt najskôr vytváral SPARQL dopyt a následne dopyt v prirodzenom jazyku. Takýmto postupom sme sa snažili eliminovať vzájomný vplyv metód. Okrem toho experti nepoznali danú dátovú množinu a preto nevedeli ako sa v nej nazývajú entity a väzby. Práve preto neboli touto znalosťou ovplyvnení ani pri jednom druhu dopytov. Pre uľahčenie písania SPARQL dopytov dostali experti jednoduchú kostru dopytu, ktorá obsahovala všetky menné priestory a základnú kostru dopytu:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>...
SELECT
WHERE {}
```

Od experimentu sme očakávali dva závery:

1. Vyhodnotenie experta, či sa prirodzený dopyt preložil na správny SPARQL dopyt.
2. Porovnanie časov písania SPARQL dopytu a dopytu v prirodzenom jazyku.

Všetky dopyty napísané v prirodzenom jazyku sa preložili na správny dopyt. V druhej časti expertom trvalo niekoľkonásobne dlhšie vytvoriť SPARQL dopyt ako dopyt v prirodzenom jazyku. Výsledné časy sú zobrazené v tabuľke 7.

Tento experiment sme realizovali pre porovnanie našej metódy s metódu OWLPath [13]. Experti v overení tejto metódy však pravdepodobne poznali dátovú sadu, keďže sa im podarilo SPARQL dopyt napísať v priebehu 20 až 70 sekúnd. Ich metódou sa okrem toho podarilo v niekoľkých prípadoch napísať dopyt pomalšie ako SPARQL jazykom, čo sa pri našom experimente nestalo.

Tabuľka 7 porovnanie časov písania dopytov v prirodzenom jazyku a písania SPARQL dopytov

Otázka	Priemerný čas písania dopytu v SPARQL jazyku	Priemerný čas písania dopytu v prirodzenom jazyku	Zlepšenie
1	2 min. 20 sek.	32 sek.	76,9 %
2	5 min 23 sek.	46 sek.	85,8 %
3	1 min. 20 sek.	32 sek.	59,8 %
4	3 min. 54 sek.	53 sek.	77,9 %

Porovnanie percentuálneho zrýchlenia našej metódy a metódy OWLPath oproti časom vytvárania SPARQL dopytu sa nachádza v tabuľke 8. Z tabuľky je vidieť, že naša metóda dosiahla oveľa väčšie zrýchlenie písania dopytov, ako to bolo v prípade metódy OWLPath.

Tabuľka 8 Zrýchlenie času našej metódy a metódy OWLPath oproti klasickému spôsobu písania dopytov pre sémantický web v SPARQL jazyku.

	Zrýchlenie času písania
OWLPath	47 % (približne)
Naša metóda	75 %

8.3 Diskusia k overeniu a experimentom

V tejto kapitole sme popísali základné možnosti nami implementovaného experimentu. Okrem toho sme zdokumentovali realizované experimenty a ich výsledky. Z týchto výsledkov sa nám ukazuje, že naša metóda je životaschopná a výrazným spôsobom uľahčuje dopytovanie sémantického úložiska. Predspracovanie napomáha odstraňovaniu problému slovníku a náš spôsob hľadania zamlčaných prepojení taktiež ukazuje výhody v podobe prirodzenejších dopytov.

Okrem toho sme ukázali aj navrhovateľa, ktorý podľa realizovaných experimentov výrazne zjednodušuje hlavne písanie náročnejších dopytov. Okrem toho odstraňuje niektoré z nejednoznačností.

Zlepšeniu nášho riešenia by pomohla implementácia upravovača dopytov a učiacej sa metódy. Upravovač dopytov by pomohol v prípade nejednoznačností používateľovi v zmene dopytu do správnej podoby. Učiaca sa metóda by vedela zachytiť časté upravovania dopytov rovnakým spôsobom a používateľovi priamo vrátiť výsledky, ktoré si želal.

Kapitola 9

Záver

V tejto práci sme sa zaoberali vytváraním a spracovávaním dopytov v prirodzenom jazyku, pričom sme sa zamerali na dopytovanie dátových množín vo forme prepojených dát. V analýze sme sa venovali možnostiam písania dopytov pre klasické relačné, ako aj ontologické databázy.

Identifikovali sme problémy existujúcich riešení používaných v kontexte sémantických úložísk, ktoré sme sa podujali vyriešiť. V tejto práci sme si ako cieľ stanovili umožniť používateľovi písať dopyty nad dátovou množinou vo forme prepojených dát bez nutnosti poznania špecifického dopytovacieho jazyka. Navrhli sme metódu, ktorá pomocou niekoľkých krokov konvertuje prirodzený dopyt používateľa v angličtine na dopyt pre sémantickú databázu, čím sme hlavný cieľ práce naplnili. Naše riešenie je navrhnuté tak, aby ho bolo možné nasadiť, s vynaložením minimálneho úsilia, na akýkoľvek sémantický dátový zdroj.

Takéto nasadenie si vyžaduje iba spustenie prvotnej fázy predspracovania. Nami navrhnutá metóda predspracovania odstraňuje pomocou spomínaného lexikónu slovníkový problém. Používatelia môžu písať dopyt svojimi slovami a naša metóda dokáže nimi použité slová mapovať na elementy v dátovom úložisku. Pri mapovaní metóde pomáha nami navrhnutý systém váh, ktorý dokáže v prípade nejednoznačností rozhodnúť o najviac pravdepodobnom mapovaní.

Metóda počíta aj s riešením nejednoznačností pri dopytovaní, ktoré rieši napovedaním v čase písania dopytu a modifikovaním dopytov po získaní výsledkov. Vďaka zaznamená-

vaniu týchto zmien, ktoré používateľ vykoná, je metóda schopná učiť sa, a tak zlepšovať poskytované výsledky vyhľadávania.

Navrhnutú metódu sme overili implementáciou prototypu. Ten obsahuje rozhranie pre preklad dopytov v prirodzenom jazyku. V rozhraní je implementovaný navrhovateľ, ktorý pomáha používateľovi pri písaní dopytov.

S použitím prototypu sme vykonali niekoľko experimentov na overenie správnosti navrhnutej metódy. Experimenty boli realizované na nami vytvorenom webovom portáli. V prvotných experimentoch sme overili fázu predspracovania a hlavne navrhnutý systém váh. Toto sme vykonali na viacerých dátových zdrojoch. Následne sme overili aj navrhovateľa pokračovania dopytu, ktorý používateľom napomáhal pri písaní dopytov a bol hodnotený z ich strany kladne. Následne sme overili aj metódu prekladu na bežných používateľoch, ktorí nám pomohli overiť správnosť výsledkov, ktoré metóda vracia. Podobné overenie sme realizovali aj s expertmi na jazyk SPARQL, vďaka ktorým sme overili správnosť prekladu na SPARQL dopyt.

Naše riešenie je možné nasadiť na akúkoľvek dátovú sadu s prepojenými dátami. Predpokladom metódy je však rozumné pomenovanie štruktúry dátovej sady a využívanie štandardných atribútov rdfs:label.

Do nášho riešenia sme zahrnuli dátovú sadu pochádzajúcu z projektu TraDiCe. Pomocou tejto dátovej sady sme našu metódu prepojili s existujúcim portálom Annota, z ktorého táto dátová množina pochádza. Vďaka tomuto prepojeniu môžu používatelia portálu Annota vyhľadávať obsah na portáli pomocou našej metódy, prirodzeným jazykom.

Použitá literatúra

1. ANDROUTSOPOULOS, I. G. RITCHIE a P. THANISCH. Natural language interfaces to databases-an introduction. Cambridge: arXiv, 1995, s. 29-81. ISSN 1351-3249.
2. WOODS, W.A. R.M. KAPLAN a B. NASH-WEBBER. *Natural Language Information System: Final Report*. Cambridge: 1972. Final Report. Bolt Beranek and Newman inc. The Lunar Sciences.
3. THOMPSON, C. a S. MARTIN. Using a menu-based natural language interface to ask map-and graph-valued database queries. New York: ACM, 1985, s. 328-38. ISBN 0-89791-170-9.
4. OWDA, M. Z. BANDAR a K. CROCKETT. Conversation-Based Natural Language Interface to Relational Databases. Silicon Valley: IEEE Computer Society, 2007, s. 363-67. ISBN 0-7695-3028-1.
5. *Adam: Student Depth Advisor*. Manchester: Convagent Ltd, 2001. Dostupné také z: http://www.convagent.com/convagent/adam3.aspx
6. ANDROUTSOPOULOS, I. G. RITCHIE a P. THANISCH. Masque/sql-an efficient and portable natural language query interface for relational databases. Edinburgh: Gordon and Breach Science Publishers, 1993, s. 327-30. ISBN 2-88124-604-4.
7. BLUM, A. Microsoft English Query 7.5: Automatic Extraction of Semantics from Relational Databases and OLAP …. In: *Proceedings of the 25th International Conference on Very Large Data Bases*. San Francisco: Morgan Kaufmann Publishers Inc. 1999, s. 247-48. ISBN 1-55860-615-7.
8. UNIVERSITY, S. The Stanford Parser: A statistical parser. In: UNIVERSITY, S. *The Stanford Natural Language Processing Group* [online]. 2012.
9. MILLER, G. A. WordNet: a lexical database for English. New York: ACM, 1995, s. 39-40.
10. TUMMARELLO, G. et al. Sig.ma: Live Views on theWeb of Data. Amsterdam: Elsevier Science Publishers, 2010, s. 355–64.
11. HOFFARTA, J. et al. YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. Saarbrücken: Max-Planck-Institut Informatik, 2010, s. 28–61. MPI-I-2010-5-007.
12. BERNSTEIN, A. a E. KAUFMANN. GINO–a guided input natural language ontology editor. Athens: Springer-Verlag, 2006, s. 144-57. ISBN 978-3-540-49029-6.
13. VALENCIA-GARCÍA, R. et al. OWLPath: An OWL Ontology-Guided Query Editor. IEEE Systems, Man, and Cybernetics Society, 2011, s. 121-36. ISSN 1083-4427.
14. KAUFMANN, E. A. BERNSTEIN a R. ZUMSTEIN. Querix: A Natural Language Interface to Query Ontologies Based on Clarification Dialogs. Athens: Springer, 2006, s. 980-81. ISSN 0302-9743.

15. WANG, C. et al. PANTO: A Portable Natural Language Interface to Ontologies. Innsbruck, Austria: Springer-Verlag, 2007, s. 473-87.
16. RAMACHANDRAN, V. A. a D.I. KRISHNAMURTHI. NLION : Natural Language Interface for querying ONtologies. Bangalore: ACM, 2009, s. 17:1-17:4. ISBN 978-1-60558-476-8.
17. Dbpedia: A nucleus for a web of open data. Busan: Springer-Verlag, 2007, s. 722-35. ISBN 3-540-76297-3.
18. FURNAS, G. W. et al. The vocabulary problem in human-system communication [Commun. ACM]. New York: ACM, 1987, s. 964-71.
19. TUMMARELLO, G. R. DELBRU a E. OREN. Sindice.com: Weaving the Open Linked Data. Busan: Springer-Verlag, 2007, s. 552-65 [cit. 2011-12-2012]. ISBN 3-540-76297-3. Dostupné z: http://www.springerlink.com/index/528H15634Q332P02.pdf
20. KASNECI, G. et al. NAGA: Searching and Ranking Knowledge. Cancun: IEEE Computer Society, 2008, s. 953-62. ISBN 978-1-4244-1836-7.
21. KASNECI, G. et al. NAGA: Harvesting, Searching and Ranking Knowledge. Vancouver: ACM, 2008, s. 1285-88. ISBN 978-1-60558-102-6.
22. ATZORI, M. a C. ZANIOLO. SWiPE: SearchingWikipedia by Example. Lyon: ACM, 16. Apríl 2012, s. 309-12. ISBN 978-1-4503-1230-1.

Zoznam príloh

A. Opis vytvoreného prototypu
B. Článok z vedeckej konferencie IIT.SRC
C. Inštalačná príručka
D. Používateľská príručka
E. Doplňujúce informácie k experimentom
F. Obsah elektronického média
G. Elektronické médium

A. Opis vytvoreného prototypu

B. Článok z vedeckej konferencie IIT.SRC

C. Inštalačná príručka

D. Používateľská príručka

E. Doplňujúce informácie k experimentom

F. Obsah elektronického média

G. Elektronické médium