Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Sémantický web v EEG/ERP doméně

Plzeň 2013 Jan Smitka

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 3. května 2013

Jan Smitka

Poděkování

Rád bych touto cestou poděkoval Ing. Romanu Moučkovi, Ph.D. za odborné vedení práce a podnětné připomínky. Dále bych chtěl poděkovat Ing. Vladimírovi Smitkovi a Mgr. Barboře Štětinové za pomoc při korektuře textu.

Abstract

Semantic web in the EEG/ERP domain
This thesis is focused on the Semantic Web technologies and storage of semantic data and metadata from neuroscience research. The thesis provides an overview of current semantic repositories. The main goal is to implement and deploy a solution for semantic annotation and search of neuroscience publications and discussions on social networks.

To accomplish the task, two utilities and a vocabulary of neuroscience research terms had to be created. The first utility, KIMBridge, is used to download publications from document repository, and discussion posts from social networks. The second utility, KIM-OWLImport, is a graphical tool that adjusts vocabularies in Web Ontology Language for semantic annotation. The created vocabulary contains common event-related potentials and signal processing methods.

Obsah

1 Úvod

Na katedře informatiky a výpočetní techniky Západočeské univerzity v Plzni působí neuroinformatická výzkumná skupina, která je zapojená do organizace International Neuroinformatics Coordinating Facility[1] (INCF).

Během neuroinformatických výzkumů vzniká množství dat a s nimi souvisejících metadat. Jedná se zejména o popis a výsledky elektrofyziologických experimentů. Tato data a metadata je nutné uchovávat a sdílet mezi jednotlivými výzkumnými skupinami.

Data a metadata o realizovaných experimentech jsou ukládána do EEG/ERP portálu.[2] Z něj je možné tato data publikovat ve formátech sémantického webu. Díky použití otevřeného standardu pro výměnu dat a integraci portálu do Neuroscience Information Framework[3] (NIF) jsou pak tato data dostupná i pro další výzkumné skupiny.

Kromě samotných dat z experimentů existuje celá řada dalších zdrojů, ze kterých je možné čerpat informace. Jedná se především o odborné publikace a články týkající se elektrofyziologických experimentů. Dalším často využívaným zdrojem jsou odborné diskuze v rámci vybraných skupin na sociálních sítích.

Vyhledávání v těchto zdrojích však není jednoduché, neboť jednotliví autoři používají různé terminologie a jednotlivá fakta označují různými klíčovými slovy. Často se tak stává, že při vyhledávání je nutné několikrát přeformulovat vyhledávací dotaz pro různá klíčová slova, než jsou požadované informace nalezeny.

Cílem této práce je nalézt vhodné řešení pro agregaci a uložení těchto dat, které usnadní následné vyhledávání relevantních informací. Snahou je využít již existující popis dat a znalostí z domény ve formátech sémantického webu.

Práce je rozdělena do několika kapitol. Následující kapitola popisuje principy sémantického webu, standardizované technologie a jejich vzájemné souvislosti. 3. kapitola pak srovnává repositáře vhodné pro ukládání sémantických dat, které umožňují fulltextové vyhledávání.

4. kapitola popisuje základní vlastnosti vybraného repositáře, možnosti definice znalostí o klíčových termínech a nahrávání dokumentů. 5. kapitola analyzuje ontologie, které má v současnosti výzkumná skupina k dispozici a které by mohly být potenciálně použity pro popis znalostí.

6. kapitola specifikuje požadavky výzkumné skupiny a poskytuje celkový přehled realizovaného řešení. 7. kapitola se zabývá instalací a konfigurací

[1] `http://www.incf.org/`

[2] `http://eegdatabase.kiv.zcu.cz/`

[3] `http://www.neuinfo.org/`

platformy pro sémantické anotování a prototypovou ontologií, která byla vytvořena pro ověření funkčnosti realizovaného řešení. Kapitoly 8 a 9 popisují nástroje, které byly vytvořeny pro usnadnění tvorby ontologie a pro import dokumentů ze vzdálených úložišť.

10. kapitola prezentuje výsledky sémantického anotování a vybraného vyhledávacího scénáře. Poslední kapitola shrnuje celou práci a nastiňuje další vývoj realizovaného řešení.

2 Technologie sémantického webu

Současný web obsahuje velké množství dat a informací. Celá síť je decentralizovaná a jednotlivé informace jsou uloženy v podobě textových dokumentů, které jsou případně obohaceny o multimediální podobu informací. Tyto dokumenty jsou dostupné prostřednictvím řetězce Uniform Resource Identifier (URI), který je jedinečně identifikuje daný dokument.

Tyto dokumenty jsou sice strukturované, ale jejich struktura nepřidává k jejich textové podobě žádnou dodatečnou informaci pro strojové zpracování. Význam jednotlivých slov a názvů v dokumentu tak lze odvodit jen pomocí metod přirozeného zpracování jazyka (NLP). To však činí velké problémy při vyhledávání ve velkém množství dokumentů.

Bez doplňujících informací o významu slov není v některých případech možné získat relevantní výsledky pro zadaný dotaz, zvláště pak pokud mohou mít klíčová slova více významů. [15] uvádí jako jeden z takových případů slovo „Marja“. To může znamenat jak křestní jméno osoby, tak finské slovo označující „bobule“.

Tento problém se snaží řešit sémantický web. Jeho cílem je poskytnout technologie a standardy, které umožní strojové zpracování a porozumění většímu množství dat dostupných na internetu. Výsledkem budou nejen přesnější výsledky vyhledávání, ale i usnadnění výměny dat, integrace jednotlivých služeb a automatizace získávání dat.

Sémantický web umožňuje k jednotlivým zdrojům na internetu připojit další doplňující informace – metadata. Je založen na následujících 6 principech [15]:

1. **Vše může být identifikováno pomocí URI.**
   Kromě stránek mají svůj jedinečný identifikátor i jednotlivé osoby, místa, události, věci a další objekty. Díky tomu se lze na každý objekt odkudkoliv odkazovat.

2. **Jednotlivé zdroje i odkazy mezi nimi mohou mít definovaný typ.**
   Současný web je změtí jednotlivých stránek a odkazů mezi nimi. U odkazů však není připojena žádná informace o tom, jaký je vztah mezi odkazovanými stránkami. Doplněním typu k těmto odkazům je možné tyto vztahy přesně popsat.

3. **Neúplné informace jsou tolerované.**
   U současného webu může jakákoliv stránka odkazovat na jakoukoliv jinou bez toho, aby se její autor musel starat o platnost tohoto odkazu.

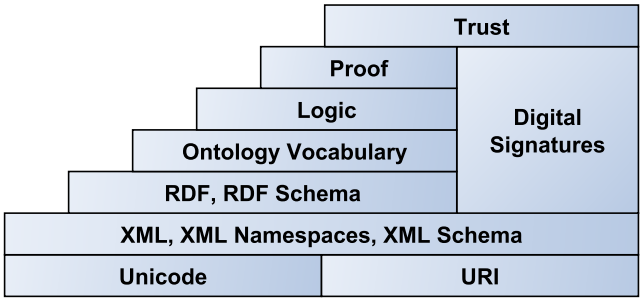
Pokud stránka bude přesunuta nebo přestane existovat, zobrazí se uživateli chyba 404. Tato vlastnost musí být zachována i u sémantického webu – nedostupnost části informace o objektu nesmí znemožnit využití zbytku informace.

4. **Absolutní pravda není nutná.**
   Ne všechny informace na webu jsou pravdivé a sémantický web toto musí zohledňovat. Každá aplikace bude pravdivost a důvěryhodnost tvrzení vyhodnocovat na základě jeho kontextu, tedy podle toho, kde a kdy bylo tvrzení uvedeno, kdo je jeho autorem a jaké měl k tvrzení pověření.

5. **Podpora vývoje.**
   Podobné koncepty jsou často vyvíjeny různými lidmi či skupinami. Sémantický web musí umožnit navázat na práci někoho jiného tak, aby bylo možné daná data zkombinovat a nevznikaly nejasnosti při jejich interpretaci.

6. **Minimalistický návrh.**
   Cílem je nestandardizovat více, než je nezbytně nutné.

Navržené technologické standardy jsou rozdělené do několika vrstev. Tyto vrstvy jsou znázorněny na obrázku 2.1.



Obrázek 2.1: Vrstvy sémantického webu.

Dvě nejnižší vrstvy, tedy „Unicode“ a „URI“, zajišťují použití mezinárodní znakové sady a odkazování na jednotlivé zdroje pomocí jedinečných identifikátorů. Vrstva „XML, XML Namespaces, XML Schema“ definuje XML standardy a zajišťuje, aby bylo možné technologie sémantického webu integrovat s dalšími technologiemi, které vycházejí z XML.

Pro současný sémantický web mají největší význam vrstvy „RDF[1], RDF Schema“ a „Ontology Vocabulary“. Vrstva „RDF, RDF Schema“ umožňuje

[1]Resource Description Framework

definovat datový model a popsat fakta o jednotlivých objektech. Právě v této vrstvě je možné jednotlivým zdrojům a odkazům přiřadit typ. Vrstva „Ontology Vocabulary“ pak podporuje spojování jednotlivých modelů a vytváření složitějších struktur – ontologií.

Vyšší vrstvy, tedy „Logic“, „Proof“ a „Trust“ nejsou v současnosti standardizovány.

Vrstva „Digital Signatures“ pak zajišťuje integritu dokumentu a ověření autenticity a důvěryhodnosti zdroje a jejího autora. Tato vrstva je definována ve standardu [13].

Kromě těchto vrstev existuje ještě několik dalších standardů, které se zaměřují na obohacení současných HTML stránek o sémantické informace. Jedním z nich je standard Microdata [11]. Umožňuje do stránky přidat sémantickou informaci o osobách, organizacích, událostech, produktech, multimediálních souborech a několika dalších typech objektů. Schéma definic těchto objektů je dostupné na webu schema.org. [2]

Na rozdíl od uvedených technologií se však jedná o nadstavbu nad současným standardem HTML5. Jeho cílem je pouze poskytnout vyhledávačům doplňující informace. Neposkytuje takové možnosti jako technologie, které jsou obecně označované jako „sémantický web“. Data je však možné transformovat do některého z formátů, které podporuje RDF. [12]

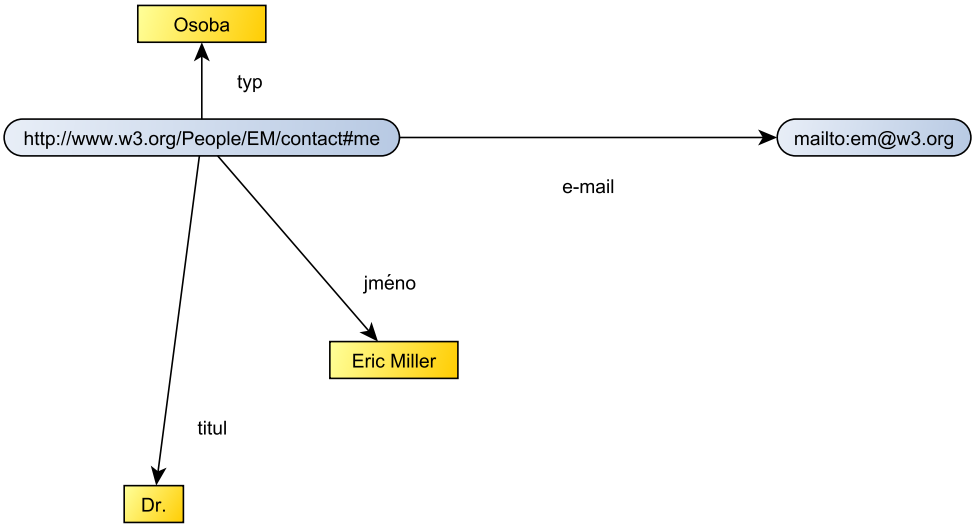
2.1 Resource Description Framework

Úvod do RDF [21] uvádí, že „RDF je jazyk pro reprezentaci informací o zdrojích dostupných v síti World Wide Web. Je určen zejména k popisu metadat zdrojů na webu, jakými jsou název, autor, datum modifikace webové stránky a informace o copyrightu a licencování daného dokumentu. Zobecněním konceptu zdroje však může být použit i pro popis objektů, které mohou být na webu identifikovány i přes to, že nejsou přímo dostupné. Jako příklad je možné uvést produkty dostupné z webového obchodu (např. informace o specifikaci, ceně a dostupnosti), nebo popis preferencí uživatele pro doručování informací.“

Jak již bylo dříve zmíněno, RDF používá myšlenku, že na všechny objekty je možné se odkazovat pomocí jejich URI. Vzniká tak graf, který reprezentuje jednotlivé objekty, jejich vlastnosti a vztahy s jinými objekty. Příklad takového grafu uvádí [21]. Jeho upravená varianta je znázorněna na obrázku 2.2.

Graf znázorňuje popis osoby jménem Eric Miller. Můžeme jej popsat

[2] `http://schema.org/docs/documents.html`



Obrázek 2.2: Graf popisující osobu jménem Eric Miller.

větou „Zdroj s URI `http://www.w3.org/People/EM/contact#me` je typu Osoba, jejíž celé jméno je Eric Miller, má e-mailovou adresu `em@w3.org` a má titul Dr.“

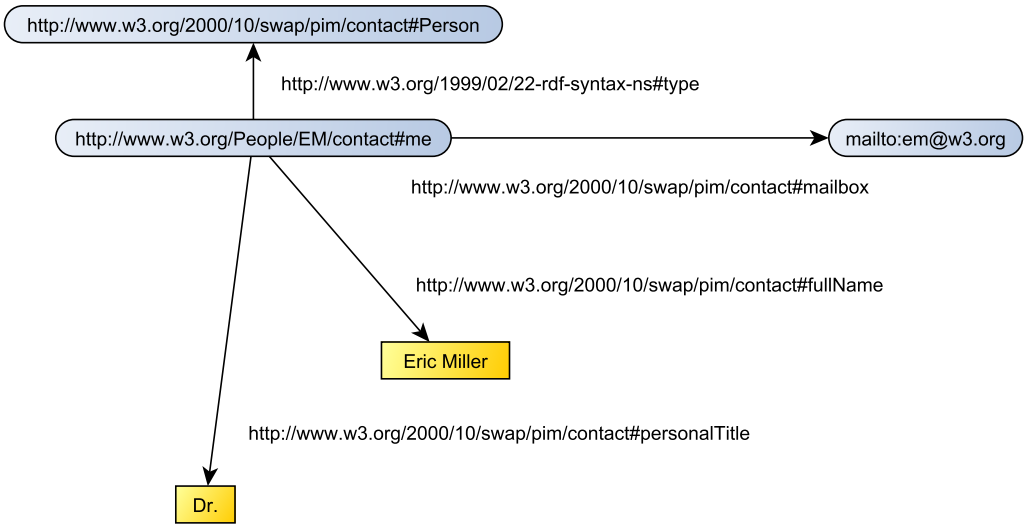
Tu samou informaci můžeme vyjádřit také pomocí jednotlivých dílčích tvrzení:

`http://www.w3.org/People/EM/contact#me` je typu osoba.
`http://www.w3.org/People/EM/contact#me` má celé jméno, jehož hodnotou je „Eric Miller“.
`http://www.w3.org/People/EM/contact#me` má e-mailovou adresu, jejíž hodnota je `em@w3.org`.
`http://www.w3.org/People/EM/contact#me` má titul, jehož hodnotou je „Dr.“

Tato tvrzení jsou v podstatě trojicemi (triplety) subjekt – predikát – objekt. Pomocí těchto tripletů lze reprezentovat informace obsažené v libovolném grafu. Subjekt vždy reprezentuje zdroj, který nabývá určité vlastnosti (predikát) s určenou hodnotou (objekt). Objektem může být buď přímo textová hodnota (např. „Eric Miller“), nebo jiný objekt identifikovaný pomocí URI (`em@w3.org`). Triplety jsou základním konceptem RDF.

Uvedený příklad je zapsaný v přirozeném jazyce. Při strojovém zpracování to však může činit potíže. Proto je možné pomocí URI identifikovat nejen jednotlivé zdroje (subjekty), ale i vlastnosti mezi nimi. Vztahy i hodnoty v grafu na obrázku 2.2 je tedy nutné identifikovat pomocí jejich URI.

Dostáváme graf na obrázku 2.3.



Obrázek 2.3: Graf popisující osobu Eric Miller s vlastnostmi reprezentovanými pomocí URI.

Kromě vlastností, které je možné definovat v konkrétních aplikacích, lze využít i některé vlastnosti, které definuje standard RDF. Ty jsou definovány ve jmenném prostoru `http://www.w3.org/1999/02/22-rdf-syntax-ns#`, typicky označovaném zkráceným prefixem `rdf:`.

Mezi nejdůležitější vestavěné vlastnosti patří `rdf:type`, která byla použita v grafu na obrázku 3. Tato vlastnost slouží k určení typu daného objektu. Díky ní je tedy možné jednotlivé zdroje typovat.

Specifikace RDF také definuje syntaxi RDF/XML [1], pomocí které je možné výše uvedený graf zapsat v jazyce, jenž je podmnožinou XML. Příklad výše uvedeného grafu je uvedený ve výpise 2.1.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
  <contact:Person
     rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>
</rdf:RDF>
```

Výpis 2.1: RDF/XML popis osoby Eric Miller

Pro uložení RDF tripletů je možné použít ještě další formáty. Mezi běžně podporované patří:

- N-Triples [2],
- Turtle, Terse RDF Triple Language [3] a
- Notation3 (N3) [5].

Tyto formáty se zaměřují na zápis samotných tripletů a neumožňují zápis ve strukturovaném XML. Oproti RDF/XML jsou však stručnější.

2.2 RDF Vocabulary Description Language

Prostřednictvím RDF můžeme definovat jednotlivé zdroje a jejich vlastnosti v podobě trojic. RDF Vocabulary Description Language, nebo také RDF Schema, zkráceně RDFS, přidává typový systém. S jeho pomocí je možné definovat třídy podobně jako v objektově orientovaných programovacích jazycích. Díky tomu lze spojit vlastnosti, u kterých se očekává, že budou použity současně. Přítomnost vlastnosti však nemůže být vynucena. [21]

Každý definovaný zdroj je pak instancí jedné nebo více tříd. Jednotlivé třídy je také možné sdružovat do hierarchií podobným způsobem jako v objektově orientovaných jazycích. Podobně jako zdroj může být instancí jedné nebo více tříd, může i třída být potomkem jedné nebo více tříd.

Definicí tříd a jejich hierarchie pak vzniká „slovník“, který lze v aplikaci používat. Výhodou RDFS je, že prostředky, které jsou použity pro popis jednotlivých tříd, jsou také slovníkem, jenž je popsán množinou trojic. Díky tomu je tak možné provádět dotazy nejen nad instancemi, které jsou definované, ale i nad samotnými definicemi tříd. Jednotlivé triplety popisující tento slovník jsou definovány ve jmenném prostoru `http://www.w3.org/2000/01/rdf-schema#` (dále označovaný prefixem `rdfs:`).

Pro vyjádření příslušnosti instance k třídě se využívá vlastnosti `rdf:type`. Jednotlivé definice tříd jsou pak instancí `rdfs:Class`. Pro vyjádření dědičnosti mezi dvěma třídami se používá vlastnosti `rdfs:subClassOf`. Jednotlivé vlastnosti je možné definovat ve třídě jako instance třídy `rdfs:Property`. Pro vyjádření příslušnosti vlastnosti k dané třídě lze potom využít vlastnost `rdfs:domain`, obor hodnot pomocí vlastnosti `rdfs:range`. Jako obor hodnot je možné použít buď URI třídy, nebo datové typy z XML Schema Definition Language (XSD). Další třídy a vlastnosti, které definuje RDFS, lze nalézt v [9].

V předchozím příkladu by bylo možné definovat třídu Osoba, která by definovala vlastnosti celé jméno, titul a e-mailová adresa. Díky tomu by byl zajištěn jednotný popis i dalších osob, které budou definovány.

Příklad definice této třídy v RDF/XML je uveden ve výpise 2.2.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd
  "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.w3.org/People/EM/contact">

<rdfs:Class rdf:ID="Osoba" />

<rdf:Property rdf:ID="fullName">
  <rdfs:domain rdf:resource="#Osoba"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Property>
<rdf:Property rdf:ID="title">
  <rdfs:domain rdf:resource="#Osoba"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Property>
<rdf:Property rdf:ID="mailbox">
  <rdfs:domain rdf:resource="#Osoba"/>
  <rdfs:range rdf:resource="&xsd;anyURI"/>
</rdf:Property>
</rdf:RDF>
```

Výpis 2.2: Definice třídy osoba.

Pokud bychom chtěli definovat například třídy Vědec a Filosof jako třídy podřízené třídě Osoba, musíme použít vlastnost `rdfs:subClassOf`, jak je vidět na výpise 2.3.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.w3.org/People/EM/contact">

<rdfs:Class rdf:ID="Osoba" />

<rdfs:Class rdf:ID="Vědec">
  <rdfs:subClassOf rdf:resource="#Osoba" />
</rdfs:Class>

<rdfs:Class rdf:ID="Filosof">
```

```
  <rdfs:subClassOf rdf:resource="#Osoba" />
</rdfs:Class>
</rdf:RDF>
```

Výpis 2.3: Použití `rdfs:subClassOf`.

2.3 Ontologie a Web Ontology Language

[30] definuje ontologie[3] jako „soubor konceptů a vztahů (označovaných jako termy), které popisují a reprezentují určitou oblast zájmu (doménu). Ontologie se používají pro klasifikaci termů, které mohou být použity v dané aplikaci, charakteristiku možných vztahů a definici omezení pro příslušné termy“. Ontologie je možné použít jako formát pro výměnu dat či pro uchování a organizaci znalostí.

Pro popis ontologií slouží Web Ontology Language, zkráceně OWL. Jedná se o výpočetně-logický jazyk, který umožňuje strojové ověření již definovaných znalostí, nebo odvození poznatků, které ze znalostí implicitně vyplývají. [16] Vychází ze standardů RDF a RDFS a přidává vlastnosti, díky kterým je možné popsat složitější definice a vztahy. Ontologie v jazyce OWL je tedy grafem v RDF.

V současné době jsou k dispozici dvě verze: OWL 1 a OWL 2. Druhá verze je plně zpětně kompatibilní – všechny OWL 1 ontologie jsou platnými OWL 2 ontologiemi. [32] Obě verze také používají stejné formáty pro uchování dat. OWL 2 přidává nové funkce, rozšiřuje například datové typy, přidává podporu pro asymetrické, tranzitivní a vzájemně vylučující se vlastnosti a definuje nové profily pro odvozování znalostí. Další text se bude zabývat novější verzí OWL 2.

Dokumenty v OWL je možné uložit v několika možných syntaxích. Tabulka 2.1 poskytuje přehled základních syntaxí, které jsou popsány v [16].

Ontologie se v jazyce OWL skládá z následujících částí: [16]

Axiomy: základní tvrzení, která jsou v ontologii vyjádřena. Takovým tvrzením může být například věta „všichni lidé jsou smrtelní“.

Entity: prvky, které reprezentují objekty a vztahy z reálného světa. Entity jsou pak součástí jednotlivých tvrzení v ontologii. Pokud budeme uvažovat

[3] W3C používá anglické výrazy „vocabulary“ (slovník) a „ontology“ (ontologie). Uvádí, že hranice mezi těmito dvěma výrazy je nejasná a většinou se používá výraz „ontology“ pro komplexnější a formálnější soubory termů, kdežto „vocabulary“ pro méně formální soubory. V českém prostředí se používá označení ontologie.

RDF/XML	Syntaxe určená pro výměnu dat, která musí být podporována všemi nástroji splňující specifikaci OWL 2.
OWL/XML	XML serializace ontologie, usnadňuje zpracování v nástrojích pracujících s OWL.
Functional Syntax	Umožňuje přehlednější zápis formální struktury ontologie.
Manchester Syntax	Jednodušší zápis ontologií využívajících deskripční logiky.
Turtle	Jednodušší zápis tripletů.

Tabulka 2.1: Seznam syntaxí OWL 2.

dvojici tvrzení „Jana je žena“ a „Jana a Petr jsou manželé“, tak jednotlivými entitami jsou konkrétní objekty (Jana, Petr), kategorie, do které je osoba řazena (žena), a vztah, který mezi těmito osobami je (jsou manželé). V jazyce OWL je konkrétní objekt (v našem případě osoby) nazýván individual (instance), kategorie je třída (class) a jednotlivé vztahy jsou jejich vlastnostmi (property).

Výrazy: Skládání jednotlivých entit do složitějších konstrukcí. Pokud například složíme třídy „žena“ a „rodič“, dostáváme třídu „matka“. Pomocí výrazů tak vznikají nové entity.

Z definovaných axiomů, entit a výrazů je možné odvozovat další znalosti, které nejsou v ontologii explicitně vyjádřené.

Jazyk OWL dále rozděluje vlastnosti objektů do následujících kategorií:

Object property popisuje vztah mezi dvěma objekty. Její hodnotou je tedy URI jiného objektu.

Datatype property je vlastnost objektu, jejíž hodnotou je literál, například řetězec nebo celočíselná hodnota.

Annotation property nepopisuje přímo vlastnost objektu, ale vlastnost ontologie (nebo její části) samotné. Jedná se například o autora a čas poslední úpravy určitého axiomu.

OWL umožňuje mezi jednotlivými třídami a vlastnostmi definovat další vztahy, například ekvivalenci, kardinalitu, vzájemnou výlučnost, případně další omezení. Podrobnější přehled vlastností poskytuje [16].

2.4 Sémantické repositáře

Sémantické repositáře slouží k uchovávání dat sémantického webu. Jsou podobné relačním SŘBD – umožňují data ukládat, modifikovat a provádět nad nimi dotazy. Oproti relačním SŘBD však data nejsou uložena v tabulkách, ale v RDF grafu. Do sémantického repositáře je tedy možné uložit ontologii definovanou v jazyce OWL a dále s ní pracovat. [24]

Sémantické repositáře ukládají jednotlivé RDF triplety. Ty jsou typicky vkládány v některém z formátů, které definují standardy RDF a OWL. Výstupem je opět některý z těchto formátů.

Alternativou k sémantickým repositářům je použití relačního SŘBD a mapování záznamů v něm uložených do RDF grafu. Tímto postupem se zabývá například [20].

Nad daty uloženými v sémantickém repositáři je možné se dotazovat dotazovacími jazyky, které jsou podobné jazyku Structured Query Language (SQL). W3C standardizuje jazyk SPARQL Protocol and RDF Query Language (zkráceně SPARQL).

2.5 Jazyk SPARQL

Všechny typy SPARQL dotazů obsahují tzv. vzor grafu. Jedná se o sadu RDF tripletů, jejichž každá část může být nahrazena za proměnnou. V datech je pak vyhledáván takový podgraf, ze kterého je možné do vzoru dosadit za proměnné hodnoty tak, aby výsledné grafy byly ekvivalentní. Kromě sady tripletů je možné definovat omezující podmínky pro jednotlivé triplety podobně jako je tomu u klauzule WHERE u SQL dotazů. [10]

Tyto triplety jsou zadávány v syntaxi Turtle [3]. Proměnné jsou uvozeny znakem ?, který je následován textovým identifikátorem dané proměnné.

Jazyk SPARQL definuje tři základní typy dotazů: `SELECT`, `CONSTRUCT` a `ASK`. Jednotlivé typy dotazů budou demonstrovány na množině tripletů, která je uvedena ve výpise 2.4. Prefix `foaf:` je použit pro jmenný prostor `http://xmlns.com/foaf/0.1/`.

```
_:osobaA foaf:name "Petr" .
_:osobaA foaf:nick "Pét'a" .
_:osobaB foaf:name "Lucie" .
_:osobaB foaf:nick "Lucka" .
_:osobaC foaf:name "Jakub" .
```

Výpis 2.4: Ukázková sada tripletů ve formátu Turtle.

Dotazy typu `SELECT` jsou funkčně i syntakticky podobné stejnojmennému

typu dotazu v jazyce SQL. Výsledkem dotazu je množina n-tic, které odpovídají zadaným podmínkám. Základní struktura dotazu vypadá následovně:

```
SELECT expr-list WHERE { search-graph-pattern }
```

`expr-list` je seznam výrazů, které budou tvořit výsledné n-tice. Výrazem může být konstantní hodnota, proměnná či některá z vestavěných funkcí. `search-graph-pattern` je seznam tripletů, které tvoří vzor grafu. Pro všechny nalezené podgrafy, jež odpovídají vzoru, je pak vrácena jedna n-tice, do níž jsou dosazeny proměnné ze vzoru.

Příklad jednoduchého dotazu `SELECT`:

```
SELECT ?name, ?nick
WHERE {
    ?x foaf:name ?name ;
       foaf:nick ?nick .
}
```

Výsledek uvedeného dotazu nad triplety z výpisu 2.4 je v tabulce 2.2. Ve výsledku není zahrnuta `osobaC`, neboť nemá v datech triplet s predikátem `foaf:nick`.

`name`	`nick`
Petr	Pét'a
Lucie	Lucka

Tabulka 2.2: Výsledek dotazu `SELECT`.

Pokud bychom chtěli do výsledků zahrnout i ty osoby, které nemají žádný `foaf:nick`, museli bychom příslušný triplet ve vzoru uvést v sekci `OPTIONAL`. Řádek ve výsledku, pro něž by nebyl `foaf:nick` nalezen, by obsahoval pouze jméno a sloupeček `nick` by neměl žádnou hodnotu. Dotaz by vypadal takto:

```
SELECT ?name, ?nick
WHERE {
    ?x foaf:name ?name .
    OPTIONAL {
        ?x foaf:nick ?nick .
    }
}
```

Dotazy typu `CONSTRUCT` jsou funkčně velmi podobné dotazům `SELECT`. Výsledkem však není množina n-tic, ale množina trojic, které tvoří graf. Základní struktura je podobná:

```
CONSTRUCT { result-graph } WHERE { search-graph-pattern }
```

`result-graph` je vzor výsledného grafu, do něhož jsou dosazovány proměnné z podgrafů, které odpovídají vyhledávanému vzoru. Pokud je výsledný graf stejný jako graf vyhledávaný, je možné část s výsledným grafem vynechat a použít zkrácenou formu

```
CONSTRUCT WHERE { search-graph-pattern }
```

Příklad jednoduchého dotazu `CONSTRUCT` (prefix `vcard:` označuje `http://www.w3.org/2001/vcard-rdf/3.0#`):

```
CONSTRUCT {
    ?x vcard:name ?name .
}
WHERE {
    ?x foaf:name ?name .
}
```

Výsledkem tohoto dotazu nad daty z výpisu 2.4 bude množina tripletů ve výpise 2.5.

```
_:osobaA foaf:name "Petr" .
_:osobaA foaf:nick "Pét'a" .
_:osobaB foaf:name "Lucie" .
_:osobaB foaf:nick "Lucka" .
_:osobaC foaf:name "Jakub" .
```

Výpis 2.5: Výsledek jednoduchého dotazu `CONSTRUCT`.

Výsledkem dotazů typu `ASK` je pouze logická hodnota, která říká, zda je možné zadaný vzor grafu v datech nalézt. Syntaxe dotazu je velmi jednoduchá:

```
ASK { search-graph-pattern }
```

Uvažujme následující dotaz nad množinou tripletů z výpisu 2.4:

```
ASK { ?x foaf:name "Jakub" }
```

Výsledkem bude logická hodnota `true`, protože existuje osoba, která se jmenuje „Jakub“. Výsledkem následujícího dotazu by byla hodnota `false`, protože osoba se jménem „Jakub“ nemá uvedenou žádnou přezdívku:

```
ASK { ?x foaf:name "Jakub"; foaf:nick ?y }
```

Další podrobnosti týkající se jednotlivých typů dotazů, jejich vyhodnocování a obecně jazyka SPARQL je možné nalézt ve specifikaci [10].

3 Přehled sémantických repositářů

Srovnáním výkonu jednotlivých repositářů se zabývá několik benchmarků, například Lehigh University Benchmark[1] (LUBM), Ontology Benchmark[2] (OUBM) či Berlin SPARQL Benchmark[3] (BSBM). Nejaktuálnější ze jmenovaných je BSBM, proto bude využit pro srovnání výkonu jednotlivých repositářů.

Při testování repositářů v BSBM jsou použity dvě sady dat:

- 100M s 100 000 748 triplety a
- 200M s 200 031 975 triplety.

U každého repositáře je měřen čas importu celého datasetu a následně počet vykonaných dotazů za jednotku času s různým počtem současně připojených klientů. Výsledky z poslední iterace benchmarku pro vybrané repositáře jsou uvedeny v tabulkách 3.1 a 3.2. Pro repositář OWLIM v edici SE je použit starší název BigOwlim.

Repositář	**Sada 100M**	**Sada 200M**
4store	26min 42s	1h 12min 04s
BigData	1h 03min 47s	3h 24min 25s
BigOwlim	**17min 22s**	**38min 36s**
TDB	1h 14min 48s	2h 45min 13s
Virtuoso	1h 49min 26s	3h 59min 38s

Tabulka 3.1: Doba načítání jednotlivých datových sad.

Kompletní souhrn výsledků poslední iterace BSBM poskytuje [6].

Následující přehled popisuje v současnosti nejvíce rozšířené a obecně podporované sémantické repositáře. Vhodný produkt pak bude vybrán na základě následujících kritérií:

1. možnosti fulltextového vyhledávání,

[1] `http://swat.cse.lehigh.edu/projects/lubm/index.htm`

[2] `http://link.springer.com/chapter/10.1007%2F11762256_12`

[3] `http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/`

Repositář	**Sada 100M**	**Sada 200M**
4store	5589	4593
BigData	2428	1795
BigOwlim	3534	1795
TDB	2274	1443
Virtuoso	**7352**	**4669**

Tabulka 3.2: Počet vykonaných dotazů za hodinu.

2. výkon a
3. poskytované funkce v oblasti RDF a OWL.

Kritéria jsou seřazena od nejdůležitějšího po nejméně důležité.

3.1 4store

4store[4] je databázový server, který je určen pro ukládání a dotazování RDF dat. Dotazovat se je možné jazykem SPARQL. 4store je určen hlavně pro nasazení do clusteru, avšak pro nasazení s menším množstvím dat lze použít jediný server. 4store byl původně vyvinut a používán interně ve společnosti garlik,[5] jako open-source projekt byl uvolněn v roce 2009. Je distribuován pod licencí GNU GPLv3.

Kromě SPARQL dotazování podporuje od verze 1.0.4 i základní fulltextové vyhledávání.

V Berlin SPARQL Benchmark V3 se umístil na 2. místě při načítání i dotazování dat. Při načítání dat byl 1,5× až 2× pomalejší než OWLIM-SE, avšak při dotazování byl v závislosti na množství dat téměř 1,5× až 2,5× rychlejší. Pro velký objem dat nebylo zpomalení dotazů natolik výrazné jako u jiných produktů. [6]

[4] `http://4store.org/`
[5] `http://www.garlik.com/`

3.1.1 Případy použití

Data Patrol

Data Patrol[6] je hlavní produkt společnosti garlik. Jeho cílem je zabránit zneužití osobních a firemních dat. Aktivně monitoruje velké množství webových stránek a sociální sítě a zjišťuje, zda se na nich neobjevily osobní, kontaktní či finanční informace (výpisy z účtů, čísla platebních karet a jiné). Pokud nalezne některé z těchto dat na veřejně přístupném místě (webu), kde existuje riziko jejich zneužití, upozorní uživatele, aby mohl podniknout případné kroky.

3.2 Jena

Jena[7] je Java framework pro vytváření aplikací sémantického webu. Poskytuje nástroje pro práci s RDF ve formátech XML, N-triples a Turtle, s ontologiemi OWL a RDFS, odvozovací engine pro RDFS a OWL, dotazovací engine odpovídající SPARQL specifikaci a úložiště dat. Komponenta LARQ[8] navíc přidává fulltextové vyhledávání.

V současnosti je již pro účely EEG/ERP portálu Jena používána – z relační databáze Oracle jsou vytvářeny OWL ontologie. To je uskutečněno pomocí frameworku Hibernate, který data z relační databáze mapuje do POJO objektů, a rozšíření JenaBeanExtension, které následně mapuje POJO objekty do RDF modelu Jeny. Více informací o nástroji JenaBeanExtension a jeho použití v EEG/ERP portálu popisuje [20].

Pro Jenu existuje i nativní úložiště TDB[9]. To ukládá na disku přímo RDF triplety, se kterými může Jena pracovat. Poskytuje tak vyšší výkon než ukládání objektů v relační databázi.

Podle Berlin SPARQL Benchmark V3 z února 2011 však Jena s TDB poskytuje nižší výkon než jiné sémantické repositáře. [6]

3.3 Sesame

Sesame[10] je, podobně jako Jena, Java framework pro zpracování sémantických dat a vytváření aplikací sémantického webu. Také poskytuje nástroje pro práci s RDF a ontologiemi, engine pro SPARQL dotazování a úložiště

[6] `http://www.garlik.com/datapatrol`
[7] `http://jena.apache.org/`
[8] Oficiální web neuvádí, zda se jedná o zkratku.
[9] Oficiální web opět neuvádí, jestli jde o zkratku.
[10] `http://www.openrdf.org/`

dat. Na rozdíl od Jeny však poskytuje odvozovací engine pouze pro RDFS. [28]

Sesame je tzv. quad-store. K tripletům je možné navíc definovat i kontext, ve kterém byl daný triplet uložený. Při výběru dat je pak možné dotaz omezit pouze na triplety s daným kontextem.

Architektura se skládá z několika vrstev:

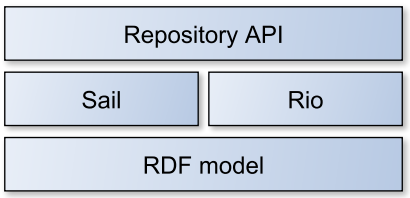
RDF model tvoří základ celého Sesame. Definuje a implementuje všechny základní RDF entity a pracují s ním všechny ostatní vrstvy.

Rio (RDF I/O) je sada parserů a zapisovatelů různých formátů RDF.

Sail (Storage And Inference Layer) je nízkoúrovňová sada rozhraní pro ukládání RDF a případné odvozování. Základní balíček Sesame obsahuje dvě implementace: MemoryStore pro ukládání dat v paměti a NativeStore pro ukládání RDF tripletů na disku.

Repository API poskytuje vysokoúrovňové API pro práci s RDF modelem nezávisle na použitém Sail rozhraní. Právě tuto vrstvu využívají aplikace.

Zjednodušená architektura je znázorněna na obrázku 3.1.



Obrázek 3.1: Architektura Sesame.

Kromě těchto vrstev obsahuje Sesame navíc implementaci HTTP serveru a klienta, která umožňuje komunikaci Sesame s jinými aplikacemi, případně komunikaci mezi jednotlivými instancemi Sesame. Pod HTTP serverem mohou běžet i webové aplikace pro správu dat, které jsou součástí distribuce.

Výkon je velmi závislý na použitém Sail rozhraní. Existuje i celá řada implementací třetích stran, z nichž nejvýznamnější a nejrozšířenější je komerční implementace OWLIM od společnosti Ontotext.

3.4 OWLIM a KIM Platform

OWLIM[11] je komerční sémantický repositář vyvíjený společností Ontotext AD. Jedná se o implementaci Sesame Sail rozhraní a přidává podporu pro

[11] `http://www.ontotext.com/owlim/`

OWL 2. K datům je možné přistupovat pomocí repository API ze Sesame, vyšší edice poskytují i adaptér pro Jenu.

Podle Berlin SPARQL Benchmark V3 má tento sémantický repositář nejrychlejší načítání dat – 200 milionů záznamů načetl za 39 minut, další produkt v pořadí téměř za dvojnásobek (72 minut). Ve zpracování dat se umístil na druhém místě za sémantickým repositářem Virtuoso. Podle stejného benchmarku také poskytuje v závislosti na počtu souběžných úloh 1,5× - 5× vyšší výkon než Jena s TDB. [6]

3.4.1 Edice

OWLIM je dodáván v následujících edicích:

Edice **OWLIM-Lite** je dostupná zdarma. Jedná se o implementaci Sesame Sail API a neposkytuje žádné další metody přístupu k datům. Podle oficiálních propagačních materiálů dokáže pracovat až s desítkami milionů záznamů, a to i na běžně dostupném hardwaru. Odvozování a dotazování probíhá v paměti, odkud jsou pak data ukládána na disk.

OWMLIM-SE je v ukládání a práci s RDF grafy funkčně shodný s OWLIM-Lite. Oproti volně dostupné verzi však používá algoritmy optimalizované pro práci s větší sadou dat a přidává podporu pro clusterování a hybridní dotazování, které kombinuje SPARQL, fulltextové vyhledávání, prostorové funkce a podporu pro řazení podle relevance.

OWLIM-Enterprise poskytuje vysoce dostupný cluster s replikací, distribucí zátěže, automatickým překonáním selhání (failover), monitoringem a dalšími vlastnostmi typickými pro řešení s vysokou dostupností.

3.4.2 KIM Platform

KIM Platform[12] je produkt vyvíjený společností Ontotext AD. Jedná se o platformu pro automatické anotování dokumentů a následné vyhledávání. Umožňuje nahrávání dokumentů, jejich automatickou anotaci proti připravené ontologii a následné vyhledávání v anotovaných dokumentech. Princip sémantického anotování a jeho realizaci v KIM uvádí jeho autoři v [14].

KIM Platform je postavený na sémantickém repositáři OWLIM. Zpracování textu a jeho anotace je realizováno pomocí nástrojů projektu GATE. Celý produkt je pro nekomerční použití poskytován zdarma (včetně úložiště OWLIM) a je možné jej nainstalovat na vlastní server. Licenci pro komerční

[12] `http://www.ontotext.com/kim/`

využití je možné pořídit od €3800, její cena se ale zvyšuje podle počtu serverů a procesorových jader, na kterých je produkt nasazen.[13]

Nahrané dokumenty jsou anotovány proti znalostní bázi uložené v sémantickém repositáři. Nad těmito dokumenty pak poskytuje komplexní vyhledávání. KIM Platform své funkce poskytuje pomocí webové SOAP služby a rozhraní pro Java RMI. K dispozici je také již hotové webové rozhraní, které tyto funkce využívá.

3.4.3 Případy použití

Tento oddíl popisuje některé významné aplikace OWLIM v praxi. Konkrétní implementační detaily nejsou zveřejňovány.

Web BBC Sports [27]

BBC na svém sportovním webu používá RDF ke zlepšení navigace a vytváření automaticky aktualizovaných stránek o sportovcích, týmech, sportovních událostech a dalších.

Jednotlivé články jsou při psaní anotovány pomocí RDF. Tato metadata jsou pak uložena v sémantickém repositáři (zde OWLIM-Enterprise).

Při vyhledávání jsou nad těmito metadaty pokládány SPARQL dotazy. Díky definované ontologii a odvozovacím pravidlům jsou pak nabídnuty vždy relevantní informace. Při sestavování stránky o týmu či sportovci je možné pouze na základě článků a jejich metadat generovat tabulky soupisek, zápasů, poskytovat aktuální informace o zraněních a mnoho dalších informací.

The National Archives [22]

The National Archives archivují dokumenty z webových stránek vládních organizací Spojeného království Velké Británie a Severního Irska. Tento archiv obsahoval 700 milionů dokumentů, z nichž velká část byla duplicitních. Vyhledávání v těchto dokumentech však bylo velmi obtížné, protože jednotlivé vládní organizace mají rozdílnou organizační strukturu (oddělení na podobné úrovni mohou mít různé zodpovědnosti) a tím i rozdílnou organizaci dokumentů.

Označením dokumentů sémantickými metadaty a následnou deduplikací se podařilo počet dokumentů snížit na 160 milionů unikátních dokumentů. Označení dokumentů bylo možné díky klasifikaci dokumentů a následnému rozpoznávání jednotlivých názvů v textu.

[13] `http://www.ontotext.com/kim/getting-started/download`

Byla také vytvořena ontologie, která zachycuje organizační strukturu vládních organizací a tím pomáhá při vyhledávání dokumentů.

Kromě OWLIM-Enterprise zde byl použit KIM Platform.

3.5 BigData

BigData[14] je, podobně jako OWLIM, implementací Sesame Sail rozhraní. Poskytuje odvozování nad RDFS a přidává omezenou podporu OWL. Je zaměřený převážně na nasazení v distribuovaných clusterech, má tedy dobrou podporu pro shardování dat a distribuci zátěže. Produkt BigData je šířen pod licencí GPLv2.

Kromě SPARQL dotazování poskytuje také jednoduché fulltextové vyhledávání.

V Berlin SPARQL Benchmark dosahuje horších výsledků než repositář OWLIM, a to jak při dotazování, tak při načítání dat. [6]

Oficiální web neuvádí žádné projekty, které BigData používají.

3.6 Virtuoso

Virtuoso[15] je databázový a aplikační server vyvíjený společností OpenLink Software. Umí ukládat klasická relační data i RDF data. Jako dotazovací jazyk používá SQL, je však schopný interpretovat také SPARQL dotazy, které jsou zabalené v SQL dotazu.

Aplikační server podporuje skriptování v PHP, ASP.NET a proprietární jazyk Virtuoso Server Pages[16] (VSP).

Virtuoso je možné pomocí oficiálních adaptérů použít i jako úložiště pro Jenu a Sesame.

V Berlin SPARQL Benchmark V3 zvítězil ve výkonu při zpracování dat. V mnoha případech byl více než 2× rychlejší než OWLIM. [6]

3.6.1 Edice

Virtuoso je distribuováno v open-source a komerčních variantách. Cena komerčních variant je odstupňována podle počtu použitých jader a dělí se na použití na pracovních stanicích a serverech. Nejlevnější varianta stojí $999

[14] `http://www.systap.com/bigdata.htm`

[15] `http://virtuoso.openlinksw.com/`

[16] `http://docs.openlinksw.com/virtuoso/vsp1.html`

a je určena pro použití na pracovních stanicích s maximálně 8 jádry a 10 vlákny. Serverová varianta se stejnými limity stojí $1999.[17]

Open-source varianta nemá oproti komerční variantě podporu pro prostorové indexy a dotazy, clusterování, replikaci a podporu pro načítání dat i z jiných datových zdrojů. [26] Kvůli chybějící podpoře jiných datových zdrojů je nutné mít všechna data uložena přímo v interní databázi Virtuosa a není možné pracovat s daty uloženými v jiném úložišti.

3.6.2 Případy použití

Na oficiální webu není k dispozici přehled konkrétních aplikací, které Virtuoso používají. [31] však uvádí, že Virtuoso používají projekty DBPedia, Bio2RDF a LOD Cloud Cache.

Web projektu DBPedia[18] uvádí, že Virtuoso používá jako backend ke svému veřejnému SPARQL přístupu. Projekt se zaměřuje na extrakci strukturovaných informací ze svobodné encyklopedie Wikipedia a propojení těchto informací s dalšími datovými zdroji na webu. [7]

Projekt Bio2RDF[19] agreguje v současnosti 19 různých heterogenních zdrojů dat z oblasti biologie a medicíny, ze kterých vytváří jednotný RDF výstup. Uvádí, že takto agreguje asi 1 miliardu tripletů. Více informací o projektu je dostupných v [4].

LOD Cloud Cache[20] je projekt společnosti OpenLink Software. Jedná se opět o agregátor několika zdrojů dat, ve kterých umožňuje vyhledávání.

3.7 Shrnutí

Po prozkoumání dostupných řešení a jejich použití se jako nejvhodnější ukazuje sémantický repositář OWLIM s KIM Platform. OWLIM patří podle aktuálních benchmarků k nejvýkonnějším sémantickým repositářům. KIM Platform pak poskytuje funkce rozšířeného vyhledávání a automatické anotování dokumentů. Oba tyto produkty jsou úspěšně využívány v projektech, které zpracovávají velké množství dat.

Díky otevřené architektuře (využití Sesame) a standardizovanému vnějšímu rozhraní (webové služby, Java RMI) je navíc toto řešení vhodné pro přizpůsobení požadavkům neuroinformatické skupiny ZČU.

[17] `http://virtuoso.openlinksw.com/pricing/`
[18] `http://dbpedia.org/`
[19] `http://bio2rdf.org/`
[20] `http://lod.openlinksw.com/`

4 KIM Platform

Následující přehled popisuje klíčové vlastnosti KIM Platform.

4.1 Znalostní báze

Základem ontologie v sémantickém repositáři, která tvoří znalostní bázi, je ontologie PROTON.[1] Ta tvoří ontologii nejvyšší úrovně a popisuje základní koncepty. Je rozdělena do 3 modulů:

System module je jediný vyžadovaný modul. Definuje koncept termínu, který se vyskytuje v textu a který je vyhledáván během anotace. Tento termín je v ontologii označován jako entita.

Top module definuje abstraktní koncepty, například agenta (činitele děje), roli, téma, datum, místo, skupinu, událost a další.

Upper module rozšiřuje předchozí modul o entity, které jsou specifické již pro konkrétní domény. Jedná se například o různé typy organizací (univerzita, vláda, pojišťovací společnost, banka...), dopravní prostředky (vozidlo, auto, letadlo...), pracovní pozice (zaměstnanec, manažer, úředník, pojišťovací agent...) a další.

Tyto moduly tvoří vhodný základ pro vlastní ontologie, které budou specifické pro požadovanou doménu.

Na této ontologii je také postavena KIM World Knowledge Base, jež je dodávána s platformou KIM. Jedná se o znalostní bázi fakt z reálného světa: geografická místa, významné organizace, lidé na významných pozicích a další. Celkem obsahuje více než 44 000 entit s 77 000 aliasy. [25]

Kromě těchto připravených entit je GATE schopen při anotování vytvářet instance některých základních entit – konkrétní data, čísla, částky. Zpracování těchto entit v anglickém jazyce zajišťuje modul ANNIE,[2] který je v KIM integrován.

4.2 Formát dokumentů

Nahrávané dokumenty mohou být v následujících formátech:

- prostý text,

[1] `http://proton.semanticweb.org/`

[2] A Nearly-New Information Extraction system

- HTML stránka,
- XML dokumenty,
- již anotované GATE XML dokumenty,
- formáty Microsoft Office 2003 - 2010: dokumenty Microsoft Word a Microsoft Powerpoint,
- Adobe PDF,
- dokumenty RTF a
- Open Document Format.

Některé formáty mohou vyžadovat pro správné zpracování ještě XML metadata o jazyce, názvu dokumentu, autorovi a datu publikace.

Nahrávané dokumenty musejí být v anglickém jazyce. Pro podporu dalších jazyků je nutné vyvinout jazykový modul pro GATE.

4.3 Ukládání a indexování dokumentů

V KIM je ukládán pouze text nahrávaných dokumentů s anotacemi. Pokud je tedy zdrojem dokumentu soubor typu PDF, nelze původní PDF z repositáře stáhnout.

Dokumenty je možné ukládat ve 3 podporovaných úložištích. Podle typu úložiště se mohou lišit možnosti a rychlost vyhledávání.

4.3.1 Apache Lucene

Lucene je běžně používaná knihovna pro indexování dokumentů a vyhledávání. Ukládá celý text dokumentů (včetně anotací) v indexu a poskytuje fulltextové vyhledávání. Toto úložiště je doporučované společností Ontotext AD a jedná se o výchozí nastavení.

4.3.2 Semantic Annotation Repository

Ukládá dokumenty do vestavěného sémantického repositáře ve formátu RDF. Poskytuje také běžné fulltextové vyhledávání, avšak vyžaduje další konfiguraci úložiště OWLIM.

4.3.3 MÍMIR

MÍMIR[3] je indexovací a vyhledávací engine, který kombinuje vlastnosti fulltextového indexování a sémantického repositáře. Je schopen vykonávat full-

[3]Multiparadigm Indexing and Retrieval, `http://www.ontotext.com/kim/mimir`

textové dotazy, SPARQL dotazy a kombinované dotazy obsahující anotační vzory.

Příkladem kombinovaného dotazu z poslední kategorie je například dotaz

```
{Organization} category:VBG {Organization} for {Money}
```

Tento dotaz vyhledá dokumenty, které obsahují následující sekvenci:

Organization: jméno organizace,
category:VBG: sloveso,
Organization: další jméno organizace,
for: slovo „for“,
Money: částka v jakékoliv rozpoznané měně.

Příkladem odpovídající věty může být „Oracle buying Sun Microsystems for $5.6 billion.“

Integrace tohoto nástroje v KIM je experimentální.

4.4 Doplnění znalostní báze a vytvoření ontologie

V sémantickém repositáři je uložena znalostní báze, kterou bude KIM Platform využívat při anotování dokumentů. Tato znalostní báze je definována formou ontologie. KIM Platform již má některé znalosti nadefinovány, pro použití ve vlastní aplikaci je však nutné dodefinovat doménovou ontologii.

V této ontologii by měly být definovány jak třídy, které danou doménu popisují, tak jejich instance, jež budou sloužit jako entity při anotaci a vyhledávání.

4.4.1 Ontologie

Aby mohly být tyto entity správně rozpoznány, musejí být jejich třídy odvozeny od třídy `Entity` z ontologie PROTON. V následujícím textu budu používat prefix `protons:` pro odkaz na jmenný prostor `http://proton.semanticweb.org/2006/05/protons`.

Pokud máme již existující ontologii, která toto kritérium nesplňuje, je nutné přidat vlastnost `rdfs:subClassOf` ke kořenové třídě ontologie. Příklad ve formátu RDF/XML je uveden ve výpise 4.1

```
<owl:Class rdf:ID="RootClass">
       <rdfs:subClassOf rdf:resource="http://proton.semanticweb.org
          /2006/05/protons#Entity"/>
```

```
</owl:Class>
```

Výpis 4.1: Příklad odvození kořenové třídy od ontologie PROTON v RDF/XML.

Můžeme však tuto definici vložit také do samostatného souboru a vyhnout se tak modifikaci původní ontologie. K definici těchto tříd se hodí notace Turtle. Ukázkový triplet je uveden ve výpise 4.2.

```
<http://example.com/ontology#RootClass> <http://www.w3.org/2000/01/
    rdf-schema#subClassOf> <http://proton.semanticweb.org/2006/05/
    protons#Entity> .
```

Výpis 4.2: Příklad odvození kořenové třídy od ontologie PROTON v notaci Turtle.

Třída `Entity` přidá pouze některé základní vlastnosti, které jsou nutné pro anotaci dokumentů, avšak doménovou ontologii nijak nenaruší. Nejdůležitější vlastnost, již je nutné zohlednit, je textový řetězec, který bude analyzován a vyhledáván v textu.

Direktiva `com.ontotext.kim.KIMConstants.ENTITY_DESCR` v konfiguračním souboru `config/install.properties` určuje, jaká vlastnost entity tento textový řetězec využívá. Jsou 3 možná nastavení této direktivy:

Aliases vyžaduje pro každý alias každé instance vytvořit novou instanci třídy `protons:Alias`. Tyto instance jsou pak s instancí entity svázány pomocí relace `protons:hasAlias`, popř. `protons:hasMainAlias`, pokud se jedná o preferovaný název.

Labels nevyžaduje definici nové instance pro každý alias. Textové řetězce jsou vyhledávány ve vlastnostech `rdfs:Label` a `protons:mainLabel`.

Custom umožňuje definovat seznam URI vlastností, ve kterých budou textové řetězce vyhledávány. Tento seznam je daný konfigurační direktivou `com.ontotext.kim.KIMConstants.MAIN_LABEL_PREDICATES`.

Rozdíl ve složitosti mezi prvními dvěma volbami bude nejlépe patrný na ukázce kódu.

Volba **Aliases** vyžaduje definici následujícím způsobem:

```
<rdf:Description rdf:about="http://example.com/ontology#Entity">
        <rdf:type rdf:resource="http://example.com/ontology#Class"/>
        <rdfs:label>Unused Entity Label</rdfs:label>
        <protons:hasMainAlias rdf:resource="http://example.com/
            ontology#Entity_1"/>
        <protons:hasAlias rdf:resource="http://example.com/ontology#
            Entity_2"/>
```

```
</rdf:Description>
<rdf:Description rdf:about="http://example.com/ontology#Entity_1">
       <rdf:type rdf:resource="http://proton.semanticweb.org
          /2006/05/protons#Alias"/>
       <rdfs:label>Entity Label</rdfs:label>
</rdf:Description>
<rdf:Description rdf:about="http://example.com/ontology#Entity_2">
       <rdf:type rdf:resource="http://proton.semanticweb.org
          /2006/05/protons#Alias"/>
       <rdfs:label>Entity Alias</rdfs:label>
</rdf:Description>
```

Entita `Entity` má definované vlastnosti `protons:hasMainAlias` a `protons:hasAlias`. Ty referencují instance `Entity_1` a `Entity_2`, ve kterých jsou popisky uloženy ve vlastnosti `rdfs:label`. Textový popisek samotné instance `Entity` nebude při anotování použit.

Oproti tomu u volby **Labels** stačí jen kratší definice:

```
<rdf:Description rdf:about="http://example.com/ontology#Entity">
       <rdf:type rdf:resource="http://example.com/ontology#Class"/>
       <protons:mainLabel>Entity Label</protons:mainLabel>
       <rdfs:label>Entity Alias</rdfs:label>
</rdf:Description>
```

Textové popisky entity Entity jsou uloženy přímo ve vlastnostech `protons:mainLabel` a `rdfs:label`.

Ontologie v tomto formátu je již připravena pro připojení do KIM Platform, ale zatím stále nebude rozpoznávána v textu. KIM sice nalezne v textu přidané entity, ale nemá žádná pravidla pro jejich zpracování.

Ta jsou v KIM definována až pro konkrétnější třídy: `protont:Object`, `protont:Person`, `protont:Organization`, nebo `protont:Location`. Prefix `protont:` je zde použit pro jmenný prostor `http://proton.semanticweb.org/2006/05/protont`.

Entita samotná také musí být označena, že pochází z důvěryhodného zdroje. Důvěryhodný zdroj je zavedený jako instance třídy `protons:Trusted`. Třída `protons:Entity` pro označení příslušnosti entity k danému zdroji definuje vlastnost `protons:generatedBy`. Několik předdefinovaných důvěryhodných zdrojů je uvedeno na začátku souboru `context/default/kb/wkb.nt`.

Popisky takto označených entit již budou v textu rozpoznávány, KIM ale ve výchozím nastavení rozlišuje velká a malá písmena. Toto nastavení lze změnit v souboru `context/default/resources/IE.gapp`.

Soubory `*.gapp` obsahují nastavení jednotlivých komponent, jenž se podílejí na sémantické anotaci. Vyhledávání entit v ontologii zajišťuje kom-

ponenta LKB Gazetteer, která má vlastnost `caseSensitive`. Pro vypnutí rozlišování velikosti písmen ji stačí nastavit na `false`.

V tomto výchozím nastavení KIM také hledá v dokumentu klíčová slova s vysokou frekvencí výskytu. Takto však může být označena i část termínu, jenž je definován v ontologii. V takovém případě není termín z ontologie rozpoznán. Nastavení `IE-no-keywords.gapp` ve složce `context/default/resources` tato klíčová slova neextrahuje.

Použitý soubor s nastavením sémantického anotování je možné změnit v konfiguračním souboru `config/nerc.properties` direktivou `com.ontotext.kim.KIMConstants.IE_APP`.

4.4.2 Zavedení ontologie

Ontologii může KIM načítat při svém prvním spuštění a prvotním vytváření databáze. Soubory s ontologiemi se umist'ují do složky `context/default/kb`, případně do kterékoliv z podsložek. V souboru `config/owlim.ttl` je pak nutné ontologie přidat do seznamu `owlim:imports` a jejich výchozí jmenný prostor do seznamu `owlim:defaultNS`:

```
owlim:imports
       "kb/owl/owl.rdfs;
       kb/owl/protons.owl;
       ...
       kb/ranks.nt;
       kb/example.com.owl;" ;
owlim:defaultNS
       "http://www.w3.org/2002/07/owl#;
       http://proton.semanticweb.org/2006/05/protons#;
       ...
       http://www.ontotext.com/kim/2006/05/wkb#;
       http://example.com/ontology#;"
```

Po přidání ontologie je nutné zastavit KIM, smazat všechny přidané dokumenty (složka `context/default/populated`) a importovat je znovu. Nové entity již budou v textech automaticky rozpoznány.

Jestliže není žádoucí mazat veškerá data, je možné ontologii importovat pomocí nástroje rdf import:

```
bin/tools/rdf import [dir]
```

Složka `[dir]` musí obsahovat všechny soubory s ontologiemi, které chceme importovat. Po dokončení importu je ještě nutné smazat cache, jež se nachází ve složce `context/default/populated/cache`. Nově přidané entity budou

rozpoznány až v nově přidaných dokumentech. Existující dokumenty je nutné přeindexovat.

Pokud navíc chceme, aby KIM zobrazoval třídy a vlastnosti z námi přidané ontologie ve svých přehledech a nabídkách, je nutné přidat jejich URI do souboru `context/default/kb/visibility.nt`:

```
<http://example.com/ontology#Class> <http://www.ontotext.com/kim
   /2006/05/kimso#visibilityLevel1> "" .
```

4.5 Import dokumentů

K importu dokumentů slouží nástroj `bin/tools/populater`. Při spuštění bez parametrů se otevře jeho grafické rozhraní. Pokud jej chceme spustit z konzole, musíme přidat parametr `console`:

```
bin/tools/populater console
```

Složku, odkud má nástroj importovat dokumenty, určuje konfigurační soubor `config/populater.xml`. Jedná se o element `INPUT_STORAGE_URL`. Ve výchozím nastavení se nové soubory hledají ve složce `corpus`, kde jsou i připravené testovací dokumenty.

KIM obsahuje také službu `populater-service`, která bude nově nahrané dokumenty automaticky importovat.

Tyto možnosti však vyžadují, aby soubory s dokumenty byly umístěny do souborového systému serveru. Pro automatizované nahrávání dokumentů je možné použít i API poskytované prostřednictvím webové služby (SOAP) a Java RMI.

4.6 Vyhledávání

Vyhledávání je dostupné pomocí webového rozhraní. To je součástí distribuce KIM Platform jako WAR archiv. Aktuální verze také poskytuje integrovaný Java Servlet Container Jetty, jenž umožňuje snadné spuštění webového rozhraní spolu s KIM Platform.

Možnosti vyhledávání, které poskytuje webové rozhraní, jsou uvedeny v příloze F.

5 Současné ontologie EEG/ERP domény

Ontologie používaná v EEG/ERP portálu v současnosti slouží hlavně jako formát pro uchování a sdílení dat. Pro sémantické anotování v KIM Platform je však potřeba ontologie, která bude popisovat znalosti z dané domény a bude definovat a klasifikovat používané termíny a klíčová slova.

Ontologie EEG/ERP portálu sice některá klíčová slova, jež by mohla být z ontologie extrahována, obsahuje v názvech tříd a vlastností, avšak při analýze ontologie byly zjištěny některé nedostatky.

Vlastnost `title` má jako `rdfs:domain`, tedy třídu, které tato vlastnost náleží, uvedenu třídu `ResearchGroup`, ale vlastnost samotná je chybně použita i u instancí několika dalších tříd, například u popisu hardwaru či scénářů. Tím se tyto instance při odvozování automaticky stávají i instancemi `ResearchGroup`, což zavádí nejednoznačnost a klíčová slova by tedy nebylo možné spolehlivě automaticky extrahovat a klasifikovat.

V rámci Electrophysiology Task Force[1] organizace INCF vzniká nová ontologie, která bude sloužit k popisu elektrofyziologických dat a metadat. Ontologie je rozdělená do dvou částí:

- část popisující zařízení a metody použité při elektrofyziologických experimentech a
- část popisující neurofyziologické koncepty.

Obě části jsou ve vývoji paralelně, první verze ontologie by však měla obsahovat převážně data z první části. Tato první verze však bude dostupná nejdříve v létě 2013 a prozatím jsou k dispozici pouze návrhy některých termínů, proto zatím nemůže být použita pro sémantické anotování.

Protože existující ontologie neobsahuje potřebné informace a nová ontologie je stále ve vývoji, je nutné pro zajištění základní funkčnosti vytvořit prototypovou ontologii, která bude popisovat alespoň část znalostí z EEG/ERP domény.

[1] `http://www.incf.org/programs/datasharing/electrophysiology-task-force`

6 Specifikace požadavků a přehled řešení

Jak již bylo uvedeno, cílem práce je navrhnout řešení pro agregaci vybraných dat z EEG/ERP domény a následné vyhledávání.

6.1 Požadavky

Indexovány budou následující typy dokumentů:

- Odborné články a další publikace, které budou dostupné typicky ve formátu PDF a
- diskuze v odborných neuroinformatických skupinách na sociální síti LinkedIn.

S členy výzkumné skupiny byly sestaveny následující příklady scénářů vyhledávání:

1. „Chci najít informace o všech experimentech, které se zaměřují na pozornost řidiče, P3 komponentu, poskytují seznam použitého hardwaru a byly provedeny na skupině nejméně 10 lidí.“
2. „Chci najít studii zaměřenou na pohybové poruchy dětí ve školním věku, ve které byla část výzkumu provedena s využitím EEP/ERG metod a vizuální stimulace.“
3. „Chci najít diskuzi o metodě matching pursuit vyšetřující existenci P3 komponenty.“

Cílem navrženého řešení je nalezení relevantních informací pro tyto typy vyhledávacích dotazů.

Z výsledků vyhledávání dále musí být možné snadno přejít k původnímu dokumentu.

6.2 Přehled řešení

Řešení lze rozdělit do tří částí:

1. instalace a konfigurace platformy umožňující sémantické anotování a vyhledávání,

2. sestavení znalostní báze (ontologie) EEG/ERP domény, která umožní rozšířené vyhledávání, a
3. stahování, ukládání a indexování dokumentů z vybraných zdrojů.

Produkt KIM Platform byl nainstalován na virtuální server s operačním systémem GNU/Linux, konkrétně jeho distribucí Debian. Webové rozhraní KIM Platform bylo nasazeno do Java Servlet containeru Tomcat a nebylo dále upravováno.

Protože ontologie, které jsou v současnosti výzkumné skupině dostupné, jsou v raném stádiu vývoje, nebo neobsahují požadovaná data, bylo nutné pro ověření funkčnosti řešení vytvořit prototypovou ontologii, jež bude popisovat část znalostí EEG/ERP domény.

Ontologie je navržena tak, aby umožnila realizaci 3. vyhledávacího scénáře. Ostatní scénáře vyžadují ontologii s obsáhlejším souborem termínů.

KIM Platform klade na vytvořené termíny požadavek, že musejí být označené jako vytvořené důvěryhodným zdrojem. Při tvorbě ontologie je však výhodnější se zaměřit na popis samotných znalostí a tyto doplňující definice vygenerovat až později.

K tomuto účelu byl vytvořen program KIM-OWLImport. Ten umožňuje pro vybrané ontologie vygenerovat příslušné definice, které budou importovány spolu s ontologií do KIM Platform. Po dalším rozšíření také může sloužit jako transformační nástroj, jenž bude z vybrané ontologie extrahovat klíčové termíny a vytvářet novou ontologii ve struktuře, která bude odpovídat ontologii PROTON.

Pro automatické stahování a anotování dokumentů byl vytvořen program KIMBridge. Ten přistupuje k vzdálenému úložišti dokumentů a službě LinkedIn, stahuje nové dokumenty a diskuze a s využitím Java RMI je importuje do KIM Platform.

KIMBridge je implementovaný jako služba, která poběží na stejném serveru jako KIM. Nové dokumenty a příspěvky v diskuzích jsou stahovány periodicky.

7 Instalace KIM Platform, prototypová ontologie

7.1 Instalace a konfigurace KIM Platform

KIM platform je možné provozovat na všech operačních systémech, na kterých je dostupná Java SE. V prostředí KIV ZČU jsme zvolili virtuální stroj s operačním systémem GNU/Linux, distribucí Debian.

Produkt je distribuován jako ZIP archiv. V něm je obsažen samotný server daemon s nakonfigurovaným sémantickým repositářem a webová aplikace v podobě WAR archivu. Server je po rozbalení možno ihned spustit. Je možné jej zastavovat a spouštět pomocí příkazu

```
bin/kim start|stop
```

Webová aplikace musí být nasazena do Java servlet containeru, jakým je například Tomcat. Od verze 3.6 obsahuje KIM Platform také integrovaný servlet container Jetty, s jehož zprovozněním jsem však měl při instalaci potíže. Proto bylo webové rozhraní nasazeno do containeru Tomcat 7.

Balíček bohužel neobsahuje žádné inicializační scripty, musel být tedy napsán základní initscript pro Debian. Script je k dispozici na přiloženém CD.

Uložené dokumenty a používaná ontologie je standardně uložená ve složce `context/default`. V ní se nachází i konfigurace sémantického repositáře OWLIM-SE.

Ve vytvořené ontologii byly textové popisky jednotlivých entit umístěny přímo ve vlastnostech `protons:mailLabel` a `rdfs:label`. Direktiva `com.ontotext.kim.KIMConstants.MAIN_LABEL_PREDICATES` tedy byla nastavena na hodnotu `Labels`. Ontolgie KIM World Knowledgebase, která počítá s nastavením na `Aliases`, nebyla do repositáře importována.

Pro sémantické anotování byla použita konfigurace `IE-no-keywords.gapp`, která neextrahuje klíčová slova s vysokou frekvencí výskytu v textu. Pro tuto konfiguraci bylo také vypnuto rozlišování velkých a malých písmen v názvech entit.

Více informací o instalaci a konfiguraci je k dispozici v oficiální dokumentaci. [23]

Během počátečních testů byly zaznamenány problémy se zpracováním textu některých PDF dokumentů. Ty jsou podle výjimek, které jsou vyvolány během jejich zpracování, způsobeny starší verzí knihovny PDFBox (1.4.0), jež je součástí KIM. Tato knihovna byla v instalaci aktualizována na verzi

1.8.1, avšak některé dokumenty stále nelze kvůli nevyřešené chybě v knihovně zpracovat.[1]

7.2 Doménová ontologie

Pro zajištění základní funkčnosti bylo proto nutné vytvořit novou ontologii, která bude popisovat část znalostí z EEG/ERP domény. Pro účely testování byly vybrány znalosti potřebné pro realizaci 3. vyhledávacího scénáře: „Chci najít diskuzi o metodě matching pursuit vyšetřující existenci P3 komponenty."

Jako základ této ontologie byl vytvořen souhrn komponent evokovaných potenciálů a jednotlivých metod, které jsou používány při zpracování naměřených signálů.

Z popisu jednotlivých komponent v [19] byl sestaven graf, jenž uvádí jednotlivé komponenty, jejich polaritu a zařazení do skupin. K jednotlivým komponentám také byly přiřazeny aliasy, pod kterými může být daná komponenta nalezena v odborné literatuře.

Při volbě těchto aliasů bylo přihlíženo ke zvyklostem, jež autoři v literatuře používají. [19] například uvádí, že komponentou P3 autoři téměř výhradně myslí komponentu P3b, proto je P3 přiřazeno jako alias k P3b, ačkoliv z hlediska formálního popisu jde o nadřazený pojem označující danou rodinu komponent.

Výsledný graf je k nahlédnutí v příloze A. Z tohoto grafu byla poté sestavena část ontologie v jazyce OWL, která strukturou odpovídá podmínkám, jež klade KIM Platform.

Graf druhé části ontologie, tedy jednotlivé metody používané při zpracování signálů, zpracoval Ing. Tomáš Řondík. Jednotlivé metody jsou rozděleny do kategorií podle jejich principu a podle typického použití. V grafu jsou také znázorněny základní vztahy mezi jednotlivými metodami.

Graf je k nahlédnutí v příloze B. Z grafu byla opět sestavena ontologie v jazyce OWL.

Před importem do KIM byla ontologie doplněna nástrojem KIM-OWLImport o triplety, které jednotlivé entity definují jako vytvořené důvěryhodným zdrojem.

Obě sestavené ontologie včetně ilustračních grafů jsou k dispozici na přiloženém CD.

[1] `https://issues.apache.org/jira/browse/PDFBOX-1512`

8 KIMBridge

KIMBridge je služba, která běží na stejném serveru jako KIM platform a periodicky stahuje nové dokumenty z vybraných zdrojů (repositářů) dat. Periodu stahování nových dokumentů lze nastavit v konfiguračním souboru. Tyto dokumenty jsou importovány do KIM, kde probíhá jejich anotace.

Do KIM jsou ukládány PDF dokumenty z Google Drive a texty diskuzí ve vybraných skupinách na sociální síti LinkedIn. Implementace repositářů s dokumenty je však obecná, takže je možné KIMBridge v budoucnu snadno rozšířit i o další zdroje dat.

Stručný popis instalace a spuštění služby je k dispozici v příloze G.

8.1 Analýza

Při importu dokumentů do KIM Platform je možné je uložit jako soubory na serveru. Tam je lze periodicky importovat pomocí nástroje `populater`, případně prostřednictvím služby `populater-service`. To však vede k duplicitě dat na serveru – text dokumentů je uložen jednou ve složce s dokumenty a jednou anotovaný ve fulltextovém indexu.

Dokumenty je také možné do KIM Platform nahrávat pomocí webových služeb, nebo prostřednictvím Java RMI. Podle dokumentace nelze pomocí webové služby manipulovat s již existujícími dokumenty v repositáři, proto bylo pro KIMBridge zvoleno rozhraní Java RMI.

Anotovány budou dva typy dokumentů:

- odborné články a publikace, které budou uloženy typicky ve formátu PDF, a
- odborné diskuze z profesně zaměřených skupin v sociální síti LinkedIn.

Jako úložiště dokumentů byla vybrána služba Google Drive, do níž je možné nahrávat libovolné dokumenty. Nahrané dokumenty jsou poté přístupné pomocí Google Drive API.

Odborné diskuze ze sociální sítě LinkedIn lze stahovat pomocí API, které služba poskytuje.

8.1.1 Google Drive API

Pro přístup k Drive API je nutné nejdříve vytvořit klientské klíče v Google APIs Console. Na výběr jsou tři typy klientských klíčů:

Web Application určený pro webové aplikace, do nichž se přihlašují uživatelé, kteří musejí aplikaci povolit přístup k jejich uživatelským datům,

Service Account, pomocí kterého aplikace do API nepřistupuje jako přihlášený uživatel, ale pod vlastním účtem, a

Installed Application, jenž slouží pro přístup z aplikací, které má uživatel nainstalované v PC nebo mobilních zařízeních.

Pro stahování dokumentů je nejvhodnější vytvořit Service Account. Pro aplikaci bude vytvořen účet Google s e-mailovou adresou v doméně `developer.gserviceaccount.com`. S tímto účtem je možné sdílet dokumenty či složky, do kterých má aplikace mít přístup.

API služby je zdokumentováno v oficiální dokumentaci [8]. Pomocí tohoto API je možné získávat buď seznamy souborů a složek, které jsou se službou sdíleny, nebo seznam upravených a přidaných dokumentů (obecně seznam změn v dokumentech). Každá změna je označena svým pořadovým číslem. Při stahování seznamu změn lze uvést nejnižší číslo změny, jež má být v seznamu zahrnuto. Uchováváním čísla poslední změny je tak možné stahovat seznam přidaných a upravených dokumentů od poslední kontroly.

Součástí každé změny je i URL souboru, ze kterého lze daný soubor dalším voláním API stáhnout. Dokumenty ve formátech Google Docs (jejich MIME-Type je ve formátu `application/vnd.google-apps.*`) není možné přímo stahovat, jsou určeny k zobrazení a editaci ve webové aplikaci Google Docs a je nutné je ze seznamu změn filtrovat. Protože je však naprostá většina odborných publikací ve formátu PDF, lze stahovat pouze dokumenty s MIME-Type `application/pdf`. Soubory PDF nejsou určeny k úpravám, proto stačí stahovat jen nové dokumenty a není nutné aktualizovat již importované dokumenty.

Google omezuje počet možných volání na jejich API, avšak pro službu Drive jsou tyto limity tak vysoké (10 000 000 požadavků za den), že pro aplikaci nepředstavují žádný problém.

Přístup do API je možný pomocí oficiálních klientských knihoven, které jsou k dispozici pro všechny jazyky používané při vývoji moderních aplikací, včetně jazyka Java.

8.1.2 LinkedIn API

Pro přístup k API služby LinkedIn je též potřeba vytvořit klientské klíče. Na portálu pro vývojáře je nutné vytvořit novou aplikaci. Aplikace pak ke službě přistupuje pomocí 4 klíčů: 2 jsou specifické pro uživatele, k jehož datům aplikace přistupuje, a zbylé dva jsou společné pro celou aplikaci.

Po vytvoření aplikace jsou vygenerovány přístupové klíče pro aktuálního uživatele. Je možné přidat další osoby jako vývojáře aplikace, kterým budou také vygenerovány přístupové údaje, případně si aplikaci může uživatel přidat v nastavení svého účtu.

K API lze přistupovat buď z JavaScriptové knihovny, jež je určena pro použití v prohlížeči, nebo pomocí REST API.

Toto API [17] poskytuje volání, pomocí něhož je možné stáhnout seznam posledních příspěvků ve skupině s daným ID. Můžeme také specifikovat datum, od kterého mají být příspěvky staženy.

Toto datum však udává pouze datum vytvoření příspěvku. Pokud byl příspěvek vytvořen před zadaným datem a po daném datu byl pouze modifikován, nebo k němu byl přidán komentář, nebude v seznamu uveden. U diskuzí je však nutné stahovat nejen nové příspěvky, ale i nové komentáře, které byly k příspěvku přidané.

API neposkytuje žádné volání, pomocí něhož by bylo možné stáhnout nové komentáře, případně příspěvky s novými komentáři. Je tak nezbytné pro každý příspěvek provést volání na získání seznam komentářů.

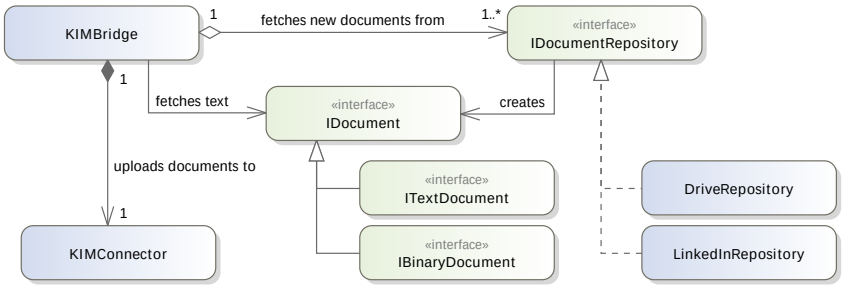
Protože služba LinkedIn omezuje počet volání, které je za den možné provést, nelze toto volání na seznam komentářů opakovat pro každý příspěvek v databázi. Podle [18] je možné provést za den 1000 požadavků na nové příspěvky ve skupině a 1000 požadavků na komentáře ke konkrétnímu příspěvku. Volání je tedy nutné omezit pouze na ty diskuze, u nichž je předpokládána aktivita diskutujících.

V aplikaci tedy bude pro každou skupinu udržována fronta diskuzí s omezenou délkou, ve které budou jednotlivé diskuze seřazeny podle aktivity. Nové a aktualizované diskuze budou zařazeny vždy na začátek fronty. Pokud některá z diskuzí nebude dlouho aktivní, propadne se na konec fronty, odkud bude posléze odstraněna. Tímto přístupem bude zajištěno, že počet volání API nepřesáhne stanovené limity.

8.2 Popis implementace

Zjednodušený diagram nejdůležitějších tříd a rozhraní v KIMBridge je znázorněn na obrázku 8.1. Podrobnější UML diagram tříd a vztahů mezi nimi, včetně signatur důležitých metod, je v příloze C.

Základní třídou programu je třída `KIMBridge`. Ta obsahuje reference na jednotlivé repositáře dokumentů (Google Drive, jednotlivé skupiny na LinkedIn) a po zavolání metody `annotateNewDocuments()` stáhne nové dokumenty a nahraje je do KIM Platform.
Spojení s KIM Platform zajišťuje třída `KIMConnector`. Ta využívá několika



Obrázek 8.1: Diagram základních tříd a rozhraní KIMBridge.

rozhraní z KIM a zajišťuje vytváření, ukládání a anotaci dokumentů. Je také schopná zajistit reanotaci určitého dokumentu, nebo aktualizaci jeho obsahu podle nově vytvořeného dokumentu. Pro připojení ke KIM využívá Java RMI.

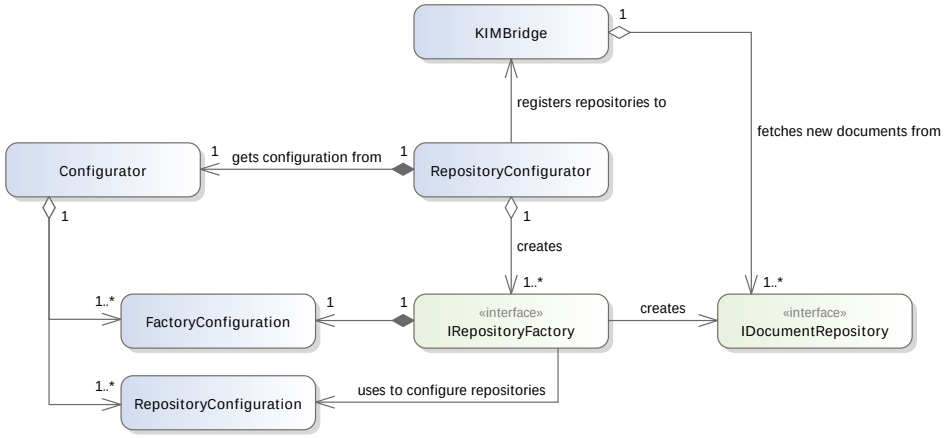
Repositář dokumentů je třída implementující rozhraní `IDocumentRepository`. Voláním metody `getNewDocuments()` repositáře dojde ke stažení nových dokumentů. Dokument je instance třídy, která implementuje rozhraní `ITextDocument`, nebo rozhraní `IBinaryDocument`. KIMBridge tyto dokumenty pomocí `KIMConnector` importuje do KIM Platform. Po úspěšné anotaci a uložení dokumentu je nad repositářem zavolána metoda `documentIndexed()`. Ta může být využita například pro vytvoření vazby mezi dokumentem v KIM Platform a dokumentem v repositáři dokumentů, aby dokument v KIM mohl být později aktualizován.

Vytváření jednotlivých repositářů při startu služby zajišťují třídy implementující rozhraní `IRepositoryFactory`. Konfigurace továren a repositářů dokumentů je uložena v konfiguračním XML souboru. Z něj jsou načítány třídou `Configurator`. Ta je schopná načítat více konfiguračních souborů a jejich direktivy slučovat. Konfigurace má 3 části: obecnou konfiguraci KIMBridge, konfiguraci továren repositářů a konfiguraci jednotlivých repositářů.

Vytvoření a registraci továren a repositářů z konfigurace zajišťuje třída `RepositoryConfigurator`. Její metoda `intitializeRepositories` má jeden parametr: instanci KIMBridge, do které mají být repositáře zaregistrovány.

Třída při volání této metody nejprve vytvoří všechny definované továrny. Továrna po vytvoření dostane svou konfiguraci ve třídě `FactoryConfiguration`. V ní jsou uvedeny například přístupové klíče k API a další parametry, jež jsou potřebné k navázání spojení s danou službou.

Vytvořené továrny jsou použity k vytvoření jednotlivých repositářů. Metodě `createRepository` je předán řetězec s interním názvem repositáře a kon-



Obrázek 8.2: Diagram tříd zajišťujících vytvoření repositářů dokumentů.

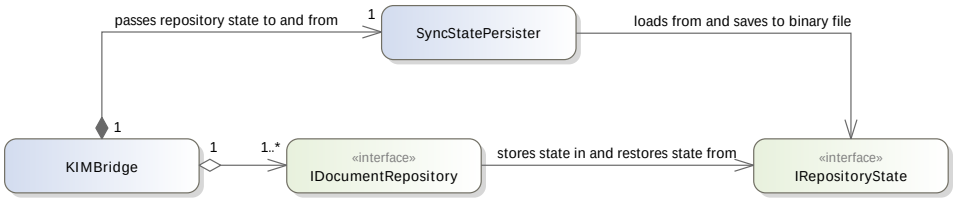
figurace repositáře ve třídě `RepositoryConfiguration`. Výsledkem volání je nový repositář s danou konfigurací, který je následně zaregistrován do KIMBridge.

Diagram tříd, jež zajišťují načítání konfigurace a vytváření repositářů, je na obrázku 8.2.

Po úspěšné synchronizaci a při ukončování KIMBridge je uchován stav všech repositářů. Repositář při ukládání vytvoří instanci třídy, která implementuje rozhraní `IRepositoryState`. V této třídě může být uloženo například datum a čas poslední synchronizace, seznam aktivních diskuzí, případně libovolná jiná data, jež je nutné uchovat při restartu služby. Po opětovném startu je stav repositáře opět obnoven.

Uložení a načítání stavu zajišťuje třída `SyncStatePersister`. Ta stav repositářů serializuje do binárního souboru. Všechny členy třídy implementující `IRepositoryState` proto musejí implementovat rozhraní `Serializable`.

Diagram tříd, které zajišťují ukládání a obnovu stavu repositářů, je znázorněn na obrázku 8.3.



Obrázek 8.3: Třídy zajišťující ukládání a načítání stavu repositářů.

8.2.1 Google Drive

Repositář pro Google Drive využívá Google Drive API Client s vygenerovaným modelem pro službu Google Drive. Tato knihovna je dostupná z adresy `https://code.google.com/p/google-api-java-client/wiki/APIs#Drive_API`.

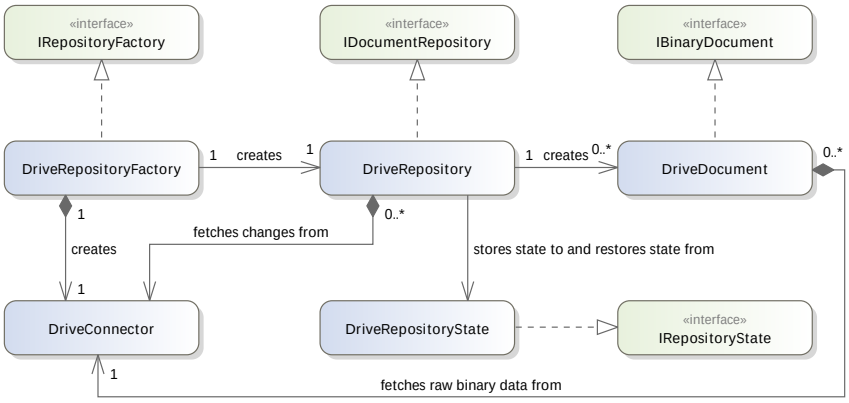
Pro tuto knihovnu byl vytvořen adaptér `DriveConnector`, jenž poskytuje metody potřebné pro stahování seznamu změn a souborů. Adaptér je vytvářen továrnou `DriveRepositoryFactory` a pro připojení potřebuje následující konfigurační direktivy:

appName – název aplikace, který bude hlášen službám Google,
accountId – ID vytvořeného service account a
privateKey – cesta k souboru service account.

ID účtu a soubor s privátním klíčem lze nalézt v Google APIs Console.

Dokument reprezentovaný třídou `DriveDocument` stahuje data až v okamžiku ukládání do KIM Platform, musí mu tedy při vytváření být předána instance `DriveConnector`. Při ukončování KIMBridge si Google Drive repositář ukládá ve třídě `DriveRepositoryState` pořadové číslo poslední změny.

Jednotlivé třídy Google Drive repositáře a vztahy mezi nimi jsou znázorněny na obrázku 8.4.



Obrázek 8.4: UML diagram tříd Google Drive repositáře.

8.2.2 LinkedIn

Repositář pro stahování diskuzí ze skupin na LinkedIn využívá knihovnu linkedin-j, která je dostupná z adresy
`https://code.google.com/p/linkedin-j/`.

Pro knihovnu byl opět vytvořen adaptér `LinkedInConnector`, jenž poskytuje metody pro stahování nových příspěvků ve skupině a stahování konkrétních příspěvků včetně jejich komentářů.

Adaptér je vytvářen továrnou `LinkedInRepositoryFactory` a vyžaduje v konfiguraci následující autorizační klíče:

consumerKey – klíč rozhraní API,
consumerSecret – tajný klíč,
token – autorizační uživatelský klíč a
tokenSecret – autorizační tajná věta.

Všechny tyto klíče je možné najít v nastavení aplikace na portálu pro vývojáře LinkedIn.

Pro každý stažený příspěvek je vytvořena instance třídy `LinkedInDocument`. Při ukládání dokumentu je z příspěvku vytvořen řetězec obsahující autora příspěvku, čas vytvoření, text příspěvku a všechny komentáře včetně jejich autora a textu.

Jednotlivé příspěvky jsou vkládány do fronty, která je reprezentovaná třídou `PostQueue`. Do ní se ukládá jen ID příspěvku ve službě LinkedIn, ID v repositáři KIM Platform a počet komentářů. Pokud při kontrole příspěvků počet komentářů vzroste, je příspěvek přesunut na začátek fronty a jeho obsah v KIM je aktualizován. Při ukončení kontroly nových příspěvků jsou z fronty vyjmuty prvky, jež přesahují její kapacitu.

Kontrola nových komentářů u sledovaných příspěvků probíhá nejdříve po 12 hodinách, aby nedošlo k vyčerpání limitu počtu volání, které je možné za den provést.

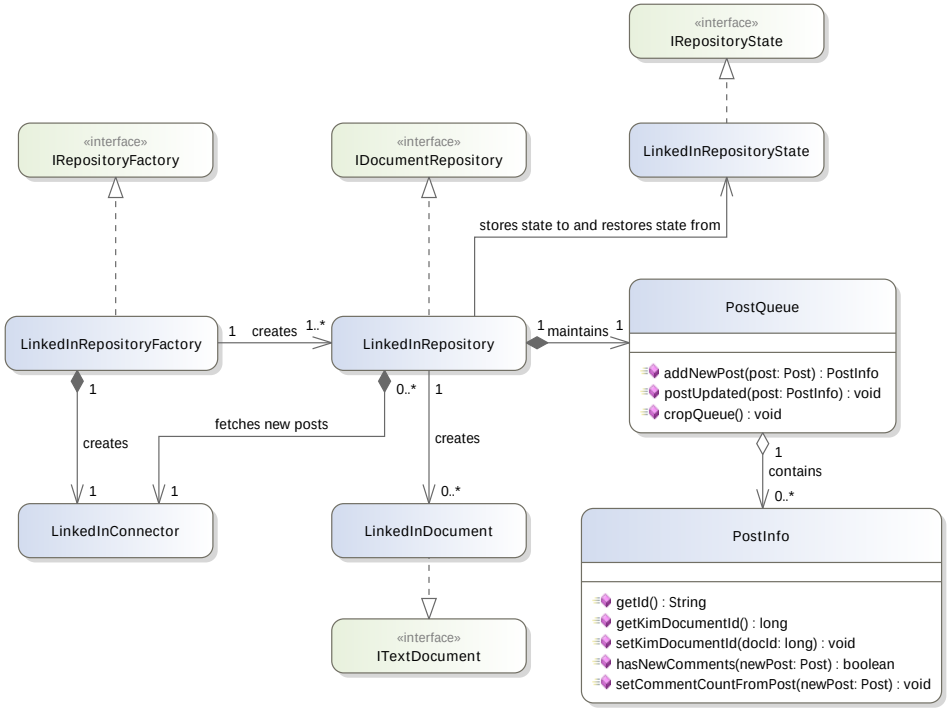
Při ukončování KIMBridge si LinkedIn repositáře ukládají datum a čas poslední kontroly nových příspěvků, datum a čas poslední kontroly nových komentářů a aktuální frontu příspěvků.

Jednotlivé třídy LinkedIn repositáře a vztahy mezi nimi jsou znázorněny na obrázku 8.5.

8.2.3 Služba KIMBridge

KIMBridge je možné s využitím knihovny Apache Commons Daemon a nástroje `jsvc` spustit jako službu, která bude běžet na pozadí a periodicky stahovat dokumenty z repositářů. Při tomto použití je nutné v konfiguračním souboru specifikovat všechny cesty absolutně.

Hlavní třída služby, `KIMBridgeDaemon`, implementuje rozhraní `Daemon`. Při inicializaci načte konfigurační soubor, vytvoří `KIMConnector`, `KIMBridge` a jednotlivé repositáře. Stav repositářů je obnoven z binárního souboru.



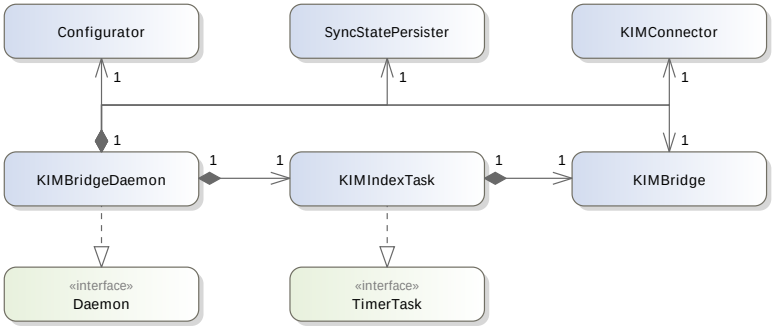
Obrázek 8.5: UML diagram tříd LinkedIn repositáře.

Při spuštění služby je spuštěn plánovač, jenž periodicky spouští úlohu definovanou ve třídě `KIMIndexTask`. Ta při svém spuštění nechá KIMBridge stáhnout a anotovat nové dokumenty. Při ukončení služby je plánovač zastaven a služba ukončena.

Diagram tříd pro službu KIMBridge je znázorněn na obrázku 8.6.

8.3 Formát konfiguračního souboru

Konfigurační soubor má několik sekcí. Obsahuje konfiguraci celého KIMBridge, jednotlivých továren repositářů a poté jednotlivých repositářů, ze kterých jsou stahovány nové dokumenty. Základní struktura konfiguračního souboru je uvedena ve výpise 8.1.



Obrázek 8.6: Diagram tříd služby KIMBridge.

```
<?xml version="1.0" encoding="UTF-8" ?>
<kimBridge>
    <configuration>
        <indexInterval>300</indexInterval>
        <syncFile>./state.bin</syncFile>
    </configuration>
    <factories>
    </factories>
    <repositories>
    </repositories>
</kimBridge>
```

Výpis 8.1: Prázdný konfigurační soubor s výchozím nastavením.

Struktura XML dokumentu je rozdělena do 3 sekcí:

configuration – globální konfigurace KIMBridge,
factories – konfigurace jednotlivých továren pro repositáře a
repositories – konfigurace jednotlivých repositářů.

Jednotlivé elementy v sekci `configuration` představují konfigurační direktivy. Název elementu odpovídá konfigurační direktivě, text uvnitř elementu její hodnotě.

Každá továrna repositářů má v sekci `factories` vlastní element. Název elementu je identifikátor továrny a povinný atribut `class` určuje třídu továrny. Továrna je vytvářena pomocí reflexe voláním výchozího konstruktoru bez parametrů. Třída implementující továrnu tedy musí mít veřejný bezparametrický konstruktor.

Jednotlivé elementy uvnitř konfigurace továrny představují její parametry. Příklad konfigurace továren je ve výpise 8.2. V tomto výpise má to-

várna `googleDrive` parametr `appName` s hodnotou KIMBridge. Druhá továrna, `linkedIn`, nemá žádné parametry.

```
<googleDrive class="cz.zcu.kiv.eeg.KIMBridge.repository.google.
   DriveRepositoryFactory">
       <appName>KIMBridge</appName>
</googleDrive>
<linkedIn class="cz.zcu.kiv.eeg.KIMBridge.repository.linkedin.
   LinkedInRepositoryFactory" />
```

Výpis 8.2: Příklad konfigurace továren.

V sekci `repositories` jsou pak definovány jednotlivé repositáře, které mají být vytvořeny pomocí továren. Název elementu je opět identifikátor repositáře, atribut type určuje název továrny, pomocí níž má být daný repositář vytvořen. Vnitřní elementy jsou opět parametry repositáře. Příklad je uveden ve výpise 8.3.

```
<drive type="googleDrive" />
<group1 type="linkedIn">
       <gid>123456</gid>
</group1>
<group2 type="linkedIn">
       <gid>654321</gid>
</group2>
```

Výpis 8.3: Příklad konfigurace repositářů.

Ve výpise jsou definovány celkem 3 repositáře. Repositář `drive` bude vytvořen továrnou, která byla ve výpise 8.2 definována v elementu `googleDrive`. Repositáře `group1` a `group2` budou vytvořeny továrnou linkedIn a každý má jako svůj jediný parametr ID skupiny, z níž budou stahovány diskuzní příspěvky.

Při spuštění z příkazové řádky program vyhledává konfigurační soubor `config.xml` ve svém pracovním adresáři. Při spuštění jako služby je nutné specifikovat celou cestu ke konfiguračnímu souboru jako první parametr služby.

9 KIM-OWLImport

KIM-OWLImport je nástroj, jenž dovoluje doplnit již existující ontologii tak, aby bylo možné ji použít v KIM Platform. Typicky se jedná o doplnění tripletů, které jednotlivé entity označují jako vytvořené důvěryhodným zdrojem.

Nástroj je ale navržený tak, aby jej šlo později rozšířit o další pravidla, jež umožní extrakci informací z již existující ontologie a vytvoření ontologie nové, která bude odpovídat struktuře, již používá KIM Platform.

Nástroj má grafické uživatelské rozhraní, v němž je možné přidávat ontologie z různých zdrojů. Pro jednotlivé ontologie pak lze definovat sadu pravidel, která budou na ontologii aplikována.

Stručná uživatelská dokumentace je k dispozici v příloze E. Grafické rozhraní i uživatelská dokumentace používají Farm-Fresh Web Icons[1] od společnosti FatCow Web Hosting, které jsou dostupné zdarma pod licencí Creative Commons Attribution 3.0.

9.1 Analýza

Aby ontologie mohla být použita v KIM Platform, je nutné s ní provést následující kroky:

1. Doplnit pro jednotlivé entity vlastnost `protons:generatedBy`, jejíž hodnotou bude důvěryhodný zdroj (instance třídy `protons:Trusted`). Je možné použít některý z již předdefinovaných zdrojů v KIM Platform, nebo tento zdroj vytvořit.

2. Všechny třídy musejí být potomky `protons:Entity`, případně některé konkrétnější třídy (`protont:Object`, `protont:Person`, `protont:Organization`, nebo `protont:Location`). Pro třídy, které tvoří kořeny hierarchie a nemají tedy v rámci ontologie definovanou rodičovskou třídu, je nutné tuto vazbu doplnit. Při automatickém doplnění je možné použít obecný `protont:Object`.

3. Aby se jednotlivé třídy a jejich vlastnosti zobrazovaly v nabídkách webového rozhraní, musejí mít v souboru `visibility.nt` uvedenou vlastnost `kimso:visibilityLevel1`.[2] Pro všechny definované třídy a vlastnosti je proto nutné záznamy do tohoto souboru vygenerovat.

[1] `http://www.fatcow.com/free-icons`

[2] `http://www.ontotext.com/kim/2006/05/kimso#visibilityLevel1`

Základním formátem, jenž musí nástroje pracující s ontologiemi v jazyce OWL[3] podporovat, je RDF/XML. Do tohoto formátu lze také ontologie převést z libovolného jiného formátu, který uvádí specifikace [32]. S ontologiemi by tedy bylo možné pracovat jako s libovolným XML dokumentem. Tento přístup by však byl poměrně obtížný a implementačně náročný.

Pro jazyk Java existuje celá řada knihoven, které práci s ontologiemi (a obecně RDF) usnadňují. Umožňují jednotlivé ontologie načíst do paměti a pracovat s nimi jako s grafem: iterovat nad jednotlivými triplety, vyhledávat triplety odpovídající danému vzoru a provádět SPARQL dotazy. Jedná se například o knihovny Jena a Sesame, jež už byly zmíněny v souvislosti se sémantickými repositáři.

S využitím těchto knihoven bude možné celou ontologii načíst do paměti a pomocí SPARQL dotazů `CONSTRUCT` pak jednotlivé triplety, které musejí být do ontologie doplněny, vygenerovat. Třídy a jejich vlastnosti jsou také definovány jako triplety, proto se lze dotazovat i nad hierarchií tříd.

Pomocí SPARQL dotazů pak také bude možné transformovat ontologie s různými strukturami tříd a instancí tak, aby je šlo použít i v KIM Platform. V tuto chvíli však není k dispozici žádná ontologie, která by poskytovala potřebná klíčová slova, proto jsou definována jen pravidla pro vytváření doplňujících tripletů.

Pro použití v nástroji KIM-OWLImport byl vybrán sémantický repositář OWLIM-Lite, jenž je dostupný zdarma. Uchovává data primárně v paměti a na disk jsou ukládána až dodatečně. Ukládání je však možné vypnout v konfiguraci repositáře.

K repositáři je zajištěn přístup pomocí API, které poskytuje Sesame. Kromě SPARQL podporuje také vlastní dotazovací jazyk SeRQL. Ten poskytuje podobné možnosti jako jazyk SPARQL, avšak jeho syntaxe se více podobá jazyku SQL a je tedy pro většinu vývojářů srozumitelnější. Kompletní popis jazyka je k dispozici v dokumentaci Sesame [29].

9.1.1 SeRQL dotazy

Výše zmíněné operace s ontologií je možné docílit pomocí následujících dotazů:

Doplnění `protons:generatedBy`

```
CONSTRUCT DISTINCT {Entity} protons:generatedBy {_sourceUri}
FROM {Entity} rdf:type {Type}, {Type} sesame:directType {ClassType}
```

[3] Platí pro obě verze – OWL 1 i OWL 2.

```
WHERE ClassType = owl:Class
USING NAMESPACE
       protons = <http://proton.semanticweb.org/2006/05/protons#>
```

Dotaz najde všechny entity, jejichž typ je přímou instancí (`sesame:directType`) typu `owl:Class`. Pokud by nebyla použita podmínka přímé instance, zahrnoval by výstup i definice vlastností, které mají omezení kardinality nebo hodnoty, protože třída `owl:Restriction` je podtřídou `owl:Class`.

Pro nalezené entity vygeneruje triplet ve tvaru uvedeném v klauzuli `CONSTRUCT`. Za parametr `_sourceUri` je nutné doplnit URI instance důvěryhodného zdroje.

Doplnění rodičovské třídy `protont:Object` ke kořenům hierarchie tříd

```
CONSTRUCT {Class} rdf:type {_protontObj}
FROM CONTEXT _context
       {Class} rdf:type {ClassType}; [sesame:directSubClassOf {
           Super}]
WHERE ClassType = owl:Class AND NOT BOUND(Super)
```

Dotaz vybere všechny třídy, jež v kontextu `_context` nemají definovanou žádnou rodičovskou třídu – nepovinná vazba `sesame:directSubClassOf` na třídu `Super` nebude naplněna. Všechny definované třídy jsou implicitně podtřídami `owl:Thing`, proto je nutné omezit kontext.

Vygenerování `visiblity.nt`

```
CONSTRUCT {Class} kimso:visibilityLevel1 ""
FROM CONTEXT _context
       {Class} sesame:directType {ClassType}
WHERE ClassType = owl:Class OR
              ClassType = owl:ObjectProperty OR
              ClassType = owl:DatatypeProperty
USING NAMESPACE
       kimso = <http://www.ontotext.com/kim/2006/05/kimso#>
```

Dotaz vygeneruje pro všechny třídy a vlastnosti definované v kontextu `_context` triplet s predikátem `kimso:visibilityLevel1`.

9.1.2 Uložení výstupu

Výstup z výše uvedených dotazů je možné uložit opět pomocí Sesame API. To poskytuje třídy pro zápis jednotlivých tripletů v různých formátech.

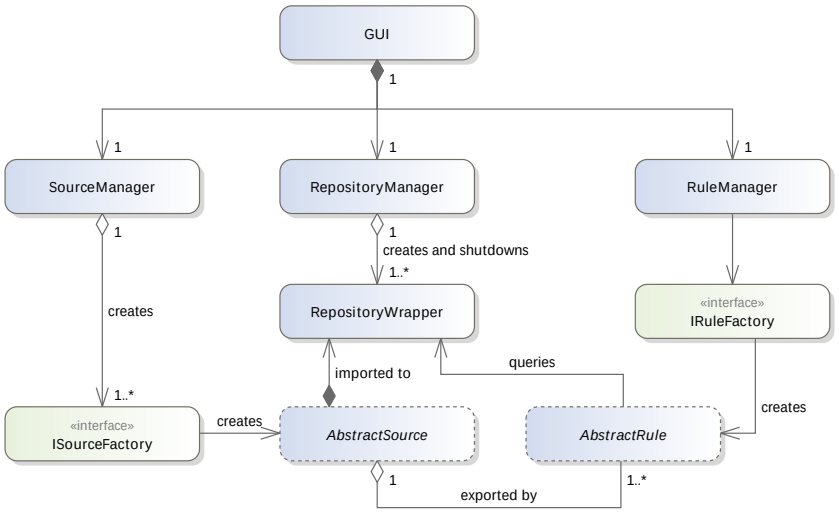
Problémem při uložení výstupu z několika dotazů mohou být jmenné prostory. Formáty založené na XML (například RDF/XML) využívají pro simulaci jmenných prostorů XML entit, které mohou být definovány pouze na začátku dokumentu v deklaraci DOCTYPE.

To může být problém, pokud by jednotlivé dotazy používaly různé jmenné prostory. Bylo by nutné nejprve vykonat všechny dotazy, zapsat všechny jmenné prostory najednou a teprve poté zapsat výsledky dotazů.

Pro proudový zápis jednotlivých výsledků je vhodnější formát, jenž nemá žádné omezení na umístění deklarace jmenného prostoru. Takovými formáty jsou N3 a Turtle. Pro KIM-OWLImport byl zvolen formát Turtle, jehož specifikace je ve vyspělejším stádiu.[4]

9.2 Popis implementace

Diagram nejdůležitějších tříd a rozhraní programu KIM-OWLImport jsou znázorněny na obrázku 9.1.



Obrázek 9.1: UML diagram základních tříd a rozhraní programu KIM-OWLImport.

Podrobnější UML diagram důležitých tříd a vztahů mezi nimi je v příloze D.

Jednotlivé soubory s ontologiemi jsou načítány do sémantických repositářů. Vytváření těchto repositářů zajišťuje třída `RepositoryManager`. Každý

[4]Specifikace formátu Turtle [3] je ve stádiu Candidate Recommendation, oproti tomu specifikace N3 [2] je pouze ve stádiu Team Submission a zatím nebyla zařazena mezi oficiální specifikace W3C.

vytvářený repositář musí mít vlastní konfiguraci. Sesame (t.j. i OWLIM-Lite) jsou konfigurovány pomocí RDF grafu.

Šablona tohoto grafu je uložena mezi zdrojovými soubory v souboru `repository/config/owlim.ttl`. Při vytváření repositáře je nutné do šablony dosadit pouze jeho název, ostatní parametry mohou být společné pro všechny repositáře. Konfigurace v šabloně například vypíná ukládání dat v repositáři na disk a vypíná odvozování nad načtenou ontologií. Popis jednotlivých konfiguračních direktiv je k dispozici v dokumentaci k OWLIM-Lite.[5]

Vytvořený repositář je pak k dispozici pomocí obálky `RepositoryWrapper`. Ta poskytuje metody na import ontologií ze souborů ve filesystému, import ze zadaného URL a pokládání SeRQL dotazů.

Aby bylo možné ontologii importovat jak ze souboru, tak z URL, je reprezentována abstraktní třídou `AbstractSource`. Její vytvoření zajišťuje třída realizující `ISourceFactory`. Parametry pro vytvoření daného zdroje (URL, cesta k souboru) jsou předávány ve třídě implementující `ISourceParams`. Z grafického rozhraní zajišťuje nastavení těchto parametrů třída `ISourceParamsComponent`.

Všechna tato rozhraní a abstraktní třída musejí být pro daný zdroj implementována. Import lokálních souborů tak zajišťují třídy `FileSource`, `FileSourceFactory` a `FileSourceParams`. Konfiguraci z uživatelského rozhraní zajišťuje třída `FileParamsComponent`. Obdobné třídy pak implementují i načítání ontologie z URL.

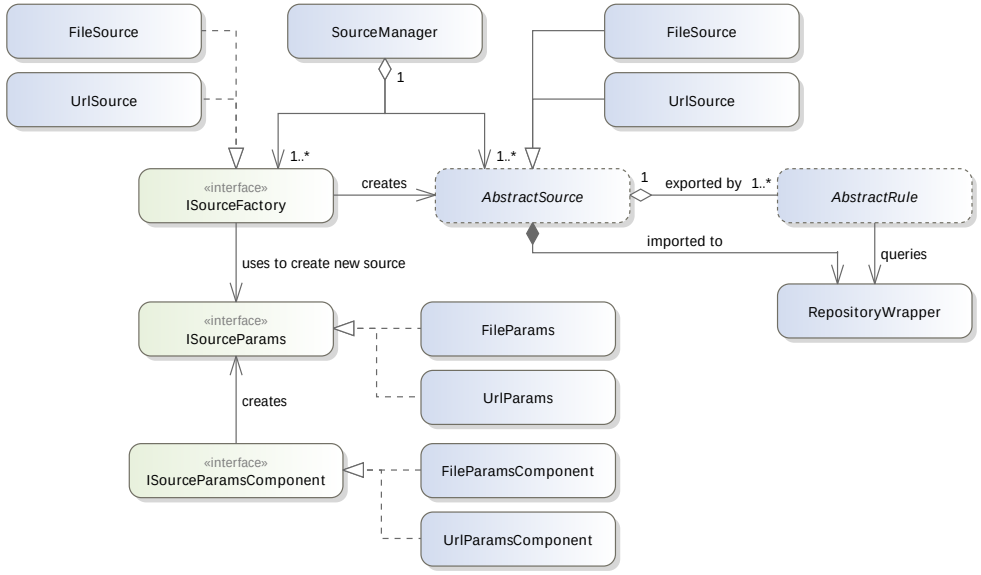
Všechny zdroje a jejich továrny jsou spravovány třídou `SourceManager`. Diagram těchto tříd je znázorněn na obrázku 9.2.

Ke každému zdroji může být přiřazeno několik pravidel, jež zajišťují vytváření nových tripletů. Architektura je velmi podobná správě jednotlivých zdrojů ontologií – pravidla implementující `AbstractRule` jsou vytvářena továrnami `IRuleFactory`. Parametry pravidla jsou reprezentovány třídami implementujícími rozhraní `IRuleParams`. Konfiguraci parametrů z uživatelského rozhraní zajišťuje třída `IRuleParamsComponent`.

Jednotlivé továrny jsou opět spravovány třídou `RuleManager`, pravidla jsou agregována v kolekci zdroje, kterému náleží. Oproti zdrojům ontologií nejsou parametry pravidla používány při jeho vytváření, ale jsou konfigurovány až po jeho vzniku. Při vytváření vzniká pouze instance s výchozím nastavením.

Metoda `getStatements()` pravidla provádí dotaz na sémantický repositář a vrací výsledek. Jednotlivé triplety jsou uloženy v kolekci implementující rozhraní `Iteration`, jež poskytuje Sesame. Ten také poskytuje další třídy pro

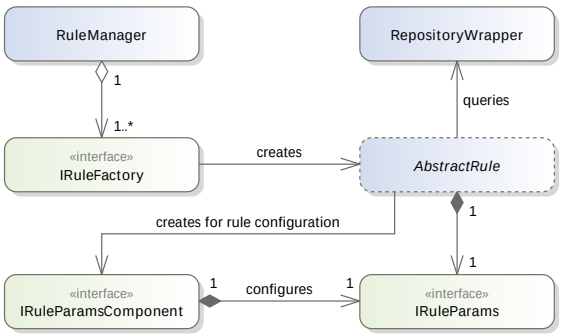
[5] `https://confluence.ontotext.com/display/OWLIMv53/OWLIM-Lite+Configuration`



Obrázek 9.2: Diagram tříd, které zajišťují načítání ontologie.

filtrování, spojování a skládání těchto kolekcí, takže je možné v rámci jednoho pravidla provést dotazů více, případně triplety vygenerovat nezávisle na repositáři a poté je přidat k výsledku dotazu.

Tyto třídy jsou opět znázorněny na obrázku 9.3.



Obrázek 9.3: Diagram tříd, které vytvářejí a reprezentují pravidla pro doplňování a transformaci ontologií.

KIM-OWLImport poskytuje následující pravidla:

ExportClassesRule exportuje všechny třídy, jež jsou v ontologii definovány. Slouží hlavně ke slučování několika ontologií do jednoho výsledného souboru.

ExportInstancesRule exportuje všechny instance, které jsou v ontologii definovány. Aby byla instance exportována, musí být její třída definovaná ve stejné ontologii. Nelze exportovat instanci třídy, jejíž definice je definována v jiné ontologii. Podobně jako předchozí pravidlo slouží hlavně ke slučování ontologií.

ExportTitleRule exportuje textové popisky instancí. Popisek je vybrán z uživatelem zvolené vlastnosti a je exportován jako vlastnost rdf:label. Toto pravidlo slouží jako ukázka transformace ontologií do struktury, jež je používána KIM Platform.

ProtonAlignRule najde všechny třídy, které nemají v ontologii definovanou rodičovskou třídu, a vytvoří triplet, jenž tyto třídy odvozuje od `protont:Object`.

TrustedSourceRule vytvoří definici důvěryhodného zdroje a pro všechny entity vygeneruje pravidlo, které je označí jako vygenerované tímto důvěryhodným zdrojem.

Další pravidla je možné poměrně snadno nadefinovat. Pro jejich použití stačí jejich továrny zaregistrovat do RuleManager.

Vykonání pravidel pro jednotlivé ontologie a jejich zapsání do výsledného souboru zajišťuje třída Exporter. Pro zadaný seznam ontologií vykoná všechna pravidla a jejich výstupy zapíše ve formátu Turtle do vybraného souboru.

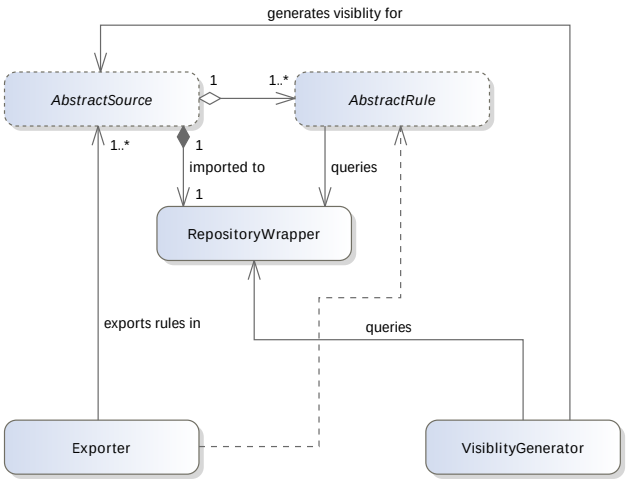
Vygenerování pravidel viditelnosti do souboru `visibility.nt` provádí třída `VisibilityGenerator`. Dotazem do repositáře vygeneruje pro všechny třídy, jež jsou v ontologii definované, seznam tripletů s predikátem `kimso:visibilityLevel1` a výsledek uloží do uživatelem vybraného souboru ve formátu N-Triple.

Diagram tříd, které zajišťují export ontologií, je na obrázku 9.4.

Všechny definované ontologie a pravidla je možné uložit ve formátu XML. Vytvoření a opětovné načtení tohoto souboru zajišťují třídy `ProjectWriter` a `ProjectReader`.

Mezi uživatelské rozhraní a aplikační a datový model není vložena žádná doplňující vrstva abstrakce.

Formuláře využívají layout manager `GridLayoutManager` od tvůrců IntelliJ. Pro překlad a spuštění aplikace je tak potřeba přibalená knihovna `forms_rt.jar`.



Obrázek 9.4: UML diagram tříd pro export ontologií.

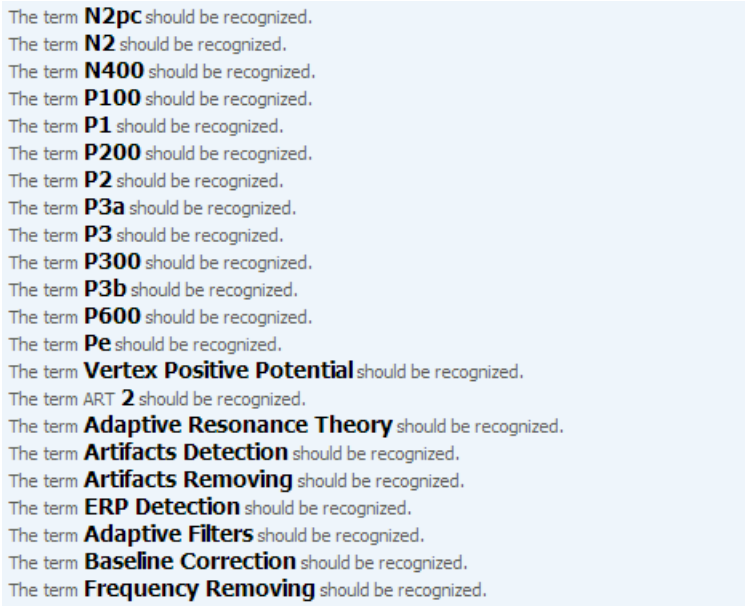
10 Výsledky

Funkčnost řešení je prezentována na dvou částech: ověření sémantické anotace podle vytvořené ontologie a realizaci vybraného vyhledávacího scénáře. Obě části byly provedeny jak s testovacími dokumenty, u kterých je předem znám očekávaný výsledek, tak se skutečnými dokumenty, jež byly do KIM Platform nahrány pomocí nástroje KIMBridge.

10.1 Sémantická anotace

Pro ověření funkčnosti sémantické anotace byl vytvořen testovací textový dokument, který obsahuje všechny termíny z ontologie včetně jejich aliasů. Všechny termíny byly umístěny do jednoduché věty, která simuluje okolní text termínu. Tento textový dokument byl do KIM Platform nahrán pomocí nástroje `populater`.

Ukázka anotovaného dokumentu je na obrázku 10.1.



Obrázek 10.1: Část anotovaného dokumentu zobrazená ve webovém rozhraní KIM UI. Anotované termíny jsou zvýrazněny tučně.

Až na jediný termín byla všechna klíčová slova správně rozpoznána. Problémem byl termín „ART2“, který ve svém názvu obsahuje několik písmen a číslovku. Z termínu bylo rozpoznáno pouze číslo, jež bylo označeno jako čís-

lovka, a zbytek termínu rozpoznán nebyl. Termíny obsahující jedno písmeno následované jednou nebo několika ciframi byly rozpoznány správně.

Dále byla spuštěna služba KIMBridge, jenž stahuje dokumenty z Google Drive a diskuze z vybraných skupin na sociální síti LinkedIn. Ve službě Google Drive bylo uloženo několik odborných publikací, které se týkají převážně ERP komponent.

Všechny tyto dokumenty byly anotovány v KIM Platform. Celkem bylo zpracováno 76 z 94 dokumentů, u zbylých dokumentů se nepodařilo extrahovat text kvůli dříve zmíněné chybě v knihovně PDFBox. Příklad anotovaného PDF dokumentu je na obrázku 10.2.

Obrázek 10.2: Anotovaný PDF dokument o komponentách P3a a P3b zobrazený ve webovém rozhraní KIM UI.

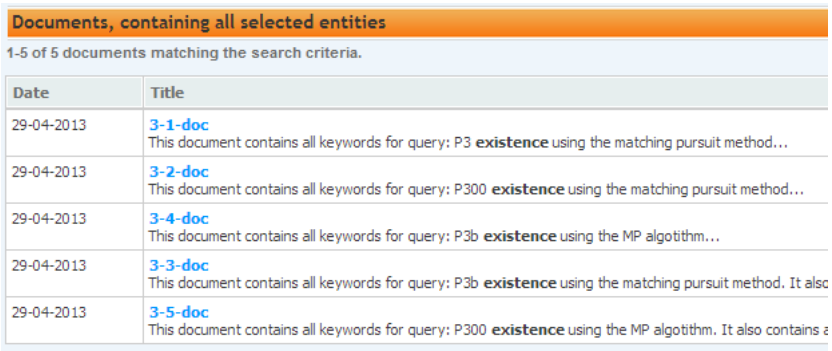
10.2 Vyhledávání

V rámci ověření funkčnosti vyhledávání byl realizován 3. vyhledávací scénář ze specifikace požadavků: „Chci najít diskuzi o metodě matching pursuit vyšetřující existenci P3 komponenty.“ Vyhledávací dotaz tedy bude obsahovat následující klíčová slova: „matching pursuit“, „P3“, „existence“.

Pro otestování byla vytvořena sada celkem 15 textových dokumentů. Tyto dokumenty jsou podle obsahu klíčových slov rozděleny do následujících 3 skupin:

1. Neobsahující žádná klíčová slova dotazu. Ve výsledku vyhledávání by se tyto dokumenty neměly objevit.
2. Obsahující část klíčových slov z dotazu, případně aliasy entit „P3“ a „matching pursuit“. Tyto dokumenty by do výsledku vyhledávání neměly být zahrnuty.
3. Obsahující všechna klíčová slova z dotazu a aliasy entit „P3“ a „matching pursuit“. Všechny tyto dokumenty by měly být zahrnuty do výsledku.

Dokumenty byly importovány do KIM Platform pomocí nástroje populater a přes webové rozhraní byl v sekci „Faceted Search“ zadán výše uvedený vyhledávací dotaz. Detail výsledku vyhledávání je na obrázku 10.3.



Obrázek 10.3: Detail výsledku vyhledávání entit „P3b“, „Matching Pursuit“ a klíčového slova „existence“.

Podle předpokladu byly ve výsledcích vyhledávání zahrnuty jen dokumenty, které obsahovaly všechna klíčová slova. Při zadávání dotazu je možné zadávat libovolný alias pro vybranou entitu. Kromě entit z ontologie je možné zadat libovolné klíčové slovo, které bude vyhledáváno fulltextově.

Vyhledávací dotaz byl položen také na dokumentech importovaných z Google Drive a z diskuzí na sociální síti LinkedIn. Žádný dokument vyšetřující existenci P3 komponenty metodou matching pursuit nebyl v dokumentech z Google Drive zahrnut. Při upřesňování dotazu však bylo nalezeno několik dokumentů řešících problematiku P3a a P3b komponent.

11 Závěr

První část práce se zabývá principy a technologiemi sémantického webu. Popisuje především struktury, které se používají pro definici ontologií v jazyce OWL, a jejich souvislosti se standardy RDF a RDF Schema. Dále se také zabývá možnostmi, jakými lze sémantická data uchovávat, a jazykem SPARQL, jímž je možné se na uložená sémantická data dotazovat.

Na první část navazuje přehled úložišť sémantických dat (sémantických repositářů), které jsou v praxi nejvíce rozšířené a které jsou srovnávané v aktuálních testech výkonu. Na základě vlastností a jejich použití v reálných projektech bylo vybráno úložiště OWLIM a KIM Platform od společnosti Ontotext AD.

Produkt KIM Patform umožňuje sémantické anotování dokumentů na základě ontologie, jež je uložena v sémantickém repositáři, který je součástí KIM Platform. V anotovaných dokumentech je možné vyhledávat a díky využití termínů z ontologie získat více relevantní výsledky než v případě běžného fulltextového vyhledávání. Použitá ontologie musí vycházet z ontologie PROTON a pro plnou funkčnost produktu musí splňovat několik dalších podmínek.

Sémantické anotování vyžaduje ontologii, jež obsahuje definice a klasifikaci termínů z EEG/ERP domény. Analýzou ontologie, která je v současnosti používaná v EEG/ERP portálu, bylo zjištěno, že tato ontologie potřebné termíny neobsahuje. Ontologie vytvářená výzkumnou skupinou v rámci organizace INCF bude dostupná až v průběhu léta 2013, proto bylo nutné pro ověření funkčnosti řešení vytvořit prototypovou ontologii obsahující část znalostí ERP domény.

Pro usnadnění tvorby ontologie, jež bude splňovat požadavky KIM Platform, byl vytvořen nástroj KIM-OWLImport. Tento nástroj je schopen načíst vybranou ontologii do sémantického repositáře v paměti a na základě definovaných pravidel ontologii doplnit tak, aby ji bylo možné použít pro sémantické anotování.

Pro import dokumentů do KIM Platform byl vytvořen program KIMBridge. Ten běží jako služba na serveru a periodicky stahuje nové dokumenty z vybraných zdrojů dat. KIMBridge podporuje stahování PDF dokumentů z Google Drive a stahování textu diskuzí na sociální síti LinkedIn.

Stažené dokumenty jsou anotovány a indexovány v KIM Platform. Následné vyhledávání je umožněno díky webovému rozhraní, které je součástí KIM. Funkčnost vyhledávání byla ověřena na testovací sadě dokumentů i na několika odborných publikacích o neuroinformatickém výzkumu.

V rámci práce tak bylo nasazeno a otestováno komplexní řešení pro sé-

mantické anotování a vyhledávání odborných dokumentů a diskuzí na sociálních sítích. Toto řešení odpovídá požadavkům výzkumné skupiny, a proto byl cíl práce splněn.

V rámci dalšího vývoje je nutné nahradit prototypovou ontologii ontologií, jež bude obsahovat větší množství klíčových termínů z EEG/ERP domény. Jako základ této ontologie je možné použít ontologii vznikající v rámci organizace INCF.

Pro automatizovaný převod libovolné ontologie do struktury, která odpovídá ontologii PROTON a kterou vyžaduje KIM Platform, je možné použít nástroj KIM-OWLImport. Ten je snadno rozšiřitelný o další pravidla a může se tak stát plnohodnotným transformačním nástrojem.

Seznam použitých zkratek

API	Application Programming Interface
ASP	Active Server Pages
BSBM	Berlin SPARQL Benchmark
EEG	Elektroencefalogram
ERP	Event-Related Potentials
GNU	GNU's Not Unix!
GPL	General Public License
HTML	HyperText Markup Language
INCF	International Neuroinformatics Coordinating Facility
LUBM	Lehigh University Benchmark
MIME	Multipurpose Internet Mail Extensions
NIF	Neuroscience Information Framework
NLP	Natural Language Processing
OUBM	Ontology Benchmark
OWL	Web Ontology Language
PHP	PHP: Hypertext Preprocessor
REST	Representational State Transfer
RDF	Resource Description Framework
RMI	Remote Method Invocation
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Qury Language
SQL	Structured Query Language
URI	Uniform Resource Identifier
VSP	Virtuoso Server Pages
WAR	Web application ARchive
XML	Extensible Markup Language
XSD	XML Schema Definition Language

Literatura

[1] BECKETT, D. RDF/XML Syntax Specification (Revised). Recommendation, W3C, únor 2004. Dostupné z: http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

[2] BECKETT, D. – GRANT, J. RDF Test Cases. Recommendation, W3C, únor 2004. Dostupné z: http://www.w3.org/TR/2004/REC-rdf-testcases-20040210/.

[3] BECKETT, D. et al. Turtle: Terse RDF Triple Language. Candidate Recommendation, W3C, únor 2013. Dostupné z: http://www.w3.org/TR/2013/CR-turtle-20130219/.

[4] BELLEAU, F. et al. Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics*. 2008, 41, 5, s. 706 – 716. Dostupné z: http://www.sciencedirect.com/science/article/pii/S1532046408000415.

[5] BERNERS-LEE, T. – CONNOLLY, D. Notation3 (N3), A readable RDF syntax. Team Submission, W3C, březen 2011. Dostupné z: http://www.w3.org/TeamSubmission/2011/SUBM-n3-20110328/.

[6] BIZER, C. – SCHULTZ, A. BSBM V3 Results. Technical report, Freie Universität Berlin, únor 2011.

[7] BIZER, C. et al. DBpedia: Querying Wikipedia like a Database, květen 2007. 16th International World Wide Web Conference Developers Track presentation. Dostupné z: http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/DBpedia-WWW2007-draft-slides.pdf.

[8] *Google Drive API Reference*. Google Inc., 2013. [cit. 3. 4. 2013]. Dostupné z: https://developers.google.com/drive/v2/reference/.

[9] GUHA, R. V. – BRICKLEY, D. RDF Vocabulary Description Language 1.0: RDF Schema. Recommendation, W3C, únor 2004. Dostupné z: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[10] HARRIS, S. – SEABORNE, A. SPARQL 1.1 Query Language. Recommendation, W3C, březen 2013. Dostupné z: http://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

[11] HICKSON, I. HTML Microdata. Technical Draft, W3C, říjen 2012. Dostupné z: http://www.w3.org/TR/2012/WD-microdata-20121025/.

[12] HICKSON, I. et al. Microdata to RDF: Transformation from HTML+Microdata to RDF. Interest Group Note, W3C, říjen 2012. Dostupné z: http://www.w3.org/TR/2012/NOTE-microdata-rdf-20121009/.

[13] HIRSCH, F. et al. XML Signature Syntax and Processing (Second Edition). Recommendation, W3C, červen 2008. Dostupné z: http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/.

[14] KIRYAKOV, A. et al. Semantic Annotation, Indexing, and Retrieval. In *2nd International Semantic Web Conference*. Springer-Verlag Berlin Heidelberg, 2003. Dostupné z: http://www.ontotext.com/sites/default/files/publications/SemAIR_ISWC169.pdf.

[15] KOIVUNEN, M.-R. – MILLER, E. *W3C Semantic Web Activity* [online]. 2001. [cit. 30. 3. 2013]. Dostupné z: http://www.w3.org/2001/12/semweb-fin/w3csw.

[16] KRÖTZSCH, M. et al. OWL 2 Web Ontology Language Primer. W3C Recommendation, W3C, prosinec 2012. http://www.w3.org/TR/2012/REC-owl2-primer-20121211/.

[17] *REST API – LinkedIn Developer Network*. LinkedIn Corporation, 2013. [cit. 3. 4. 2013]. Dostupné z: http://developer.linkedin.com/rest.

[18] *Throttle Limits – LinkedIn Developer Network*. LinkedIn Corporation, 2013. [cit. 3. 4. 2013]. Dostupné z: http://developer.linkedin.com/documents/throttle-limits.

[19] LUCK, S. J. *An Introduction to the Event-Related Potential Technique*. Cambridge, MA : MIT Press, 1st edition, 2005. ISBN 978-0262621960.

[20] MARKVART, F. EEG/ERP portál – transformace do sémantického webu. Bakalářská práce, Západočeská univerzita v Plzni, Plzeň, 2011.

[21] MILLER, E. – MANOLA, F. RDF Primer. Recommendation, W3C, únor 2004. Dostupné z: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[22] Ontotext AD. *The National Archives: Semantic Knowledge Base* [online]. [cit. 21. 10. 2012]. Dostupné z: http://www.ontotext.com/case/nationalArchives-skb.

[23] *KIM 3.6 Documentation*. Ontotext AD, 2013. [cit. 30. 3. 2013]. Dostupné z: https://confluence.ontotext.com/display/Kim36Docs.

[24] Ontotext AD. *Semantic Repository* [online]. [cit. 31. 3. 2013]. Dostupné z: http://www.ontotext.com/semantic-repository.

[25] Ontotext AD. *KIM Platform: Knowledge Base Statistics* [online]. listopad 2011. [cit. 21. 10. 2012]. Dostupné z: http://www.ontotext.com/sites/default/files/kim/KB_statistics.pdf.

[26] OpenLink Software. *Virtuoso Universal Server - Feature Comparison Matrix* [online]. [cit. 21. 10. 2012]. Dostupné z: http://virtuoso.openlinksw.com/features-comparison-matrix/.

[27] RAYFIELD, J. *Sports Refresh: Dynamic Semantic Publishing* [online]. 17. 4. 2012. [cit. 21. 10. 2012]. Dostupné z: http://www.bbc.co.uk/blogs/bbcinternet/2012/04/sports_dynamic_semantic.html.

[28] *Sesame User Guide: rdf:about Sesame 2*. Sesame developers, 2012. [cit. 21. 10. 2012]. Dostupné z: http://www.openrdf.org/doc/sesame2/users/ch01.html.

[29] *Sesame User Guide: The SeRQL query language (revision 3.1)*. Sesame developers, 2012. [cit. 4. 4. 2013]. Dostupné z: http://www.openrdf.org/doc/sesame2/users/ch09.html.

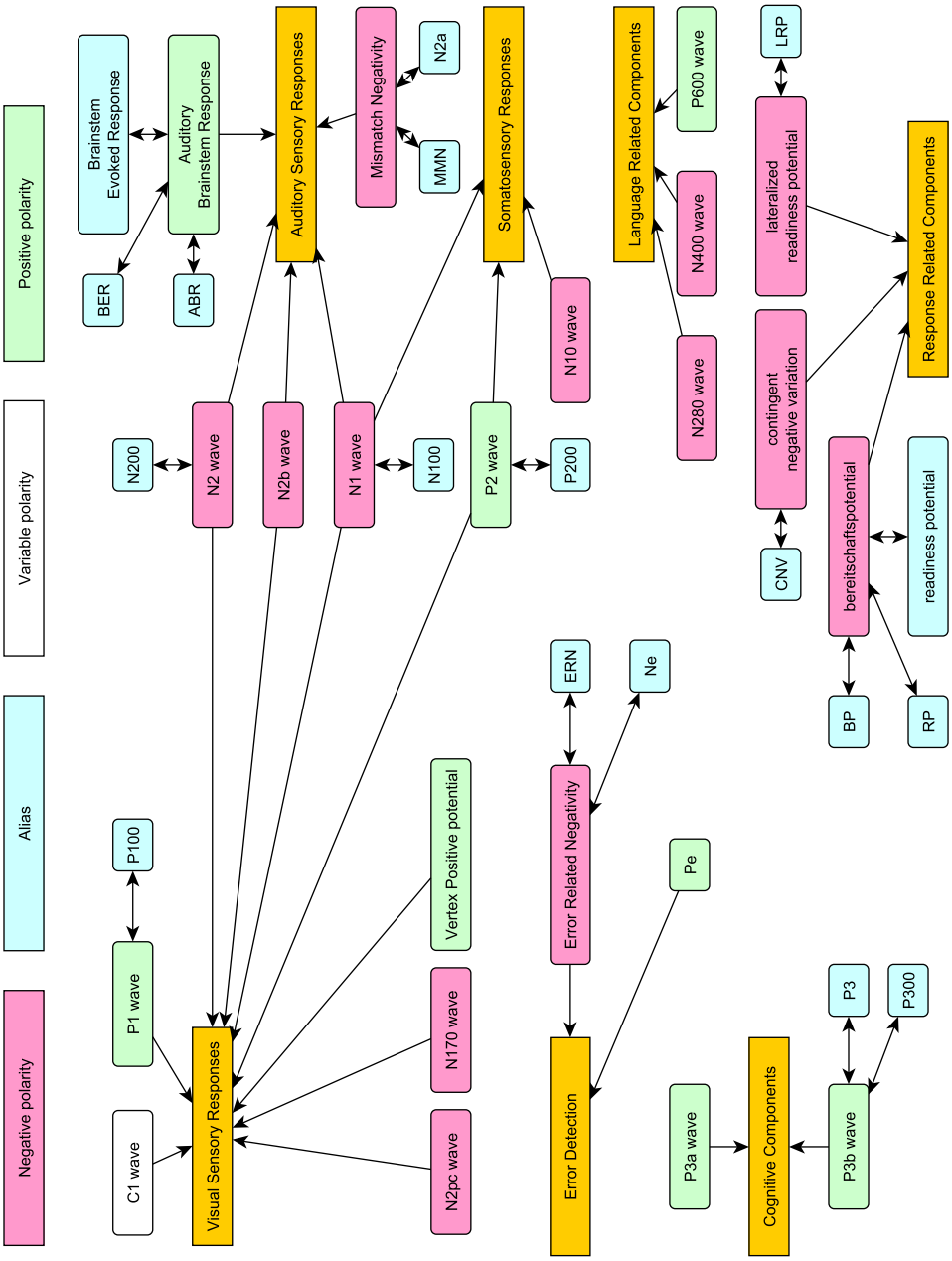
[30] W3C. *Vocabularies* [online]. [cit. 30. 3. 2013]. Dostupné z: http://www.w3.org/standards/semanticweb/ontology.

[31] *OpenLink Virtuoso – Semantic Web Standards wiki* [online]. 2012. [cit. 21. 10. 2012]. Dostupné z: http://www.w3.org/2001/sw/wiki/OpenLink_Virtuoso.

[32] W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview (Second Edition). W3C Recommendation, W3C, prosinec 2011. Dostupné z: http://www.w3.org/TR/2012/REC-owl2-overview-20121211/.

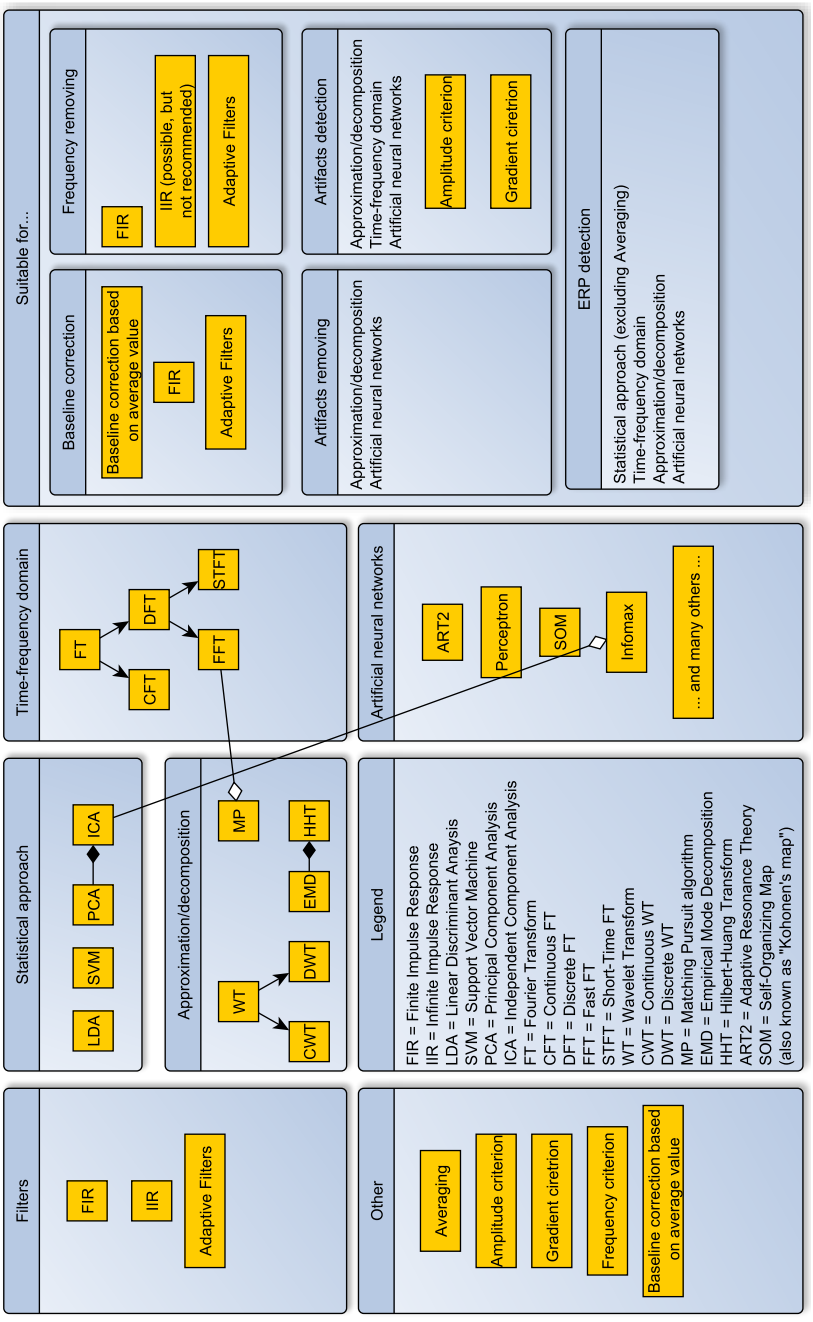
Přílohy

A Ontologie EEG/ERP komponent

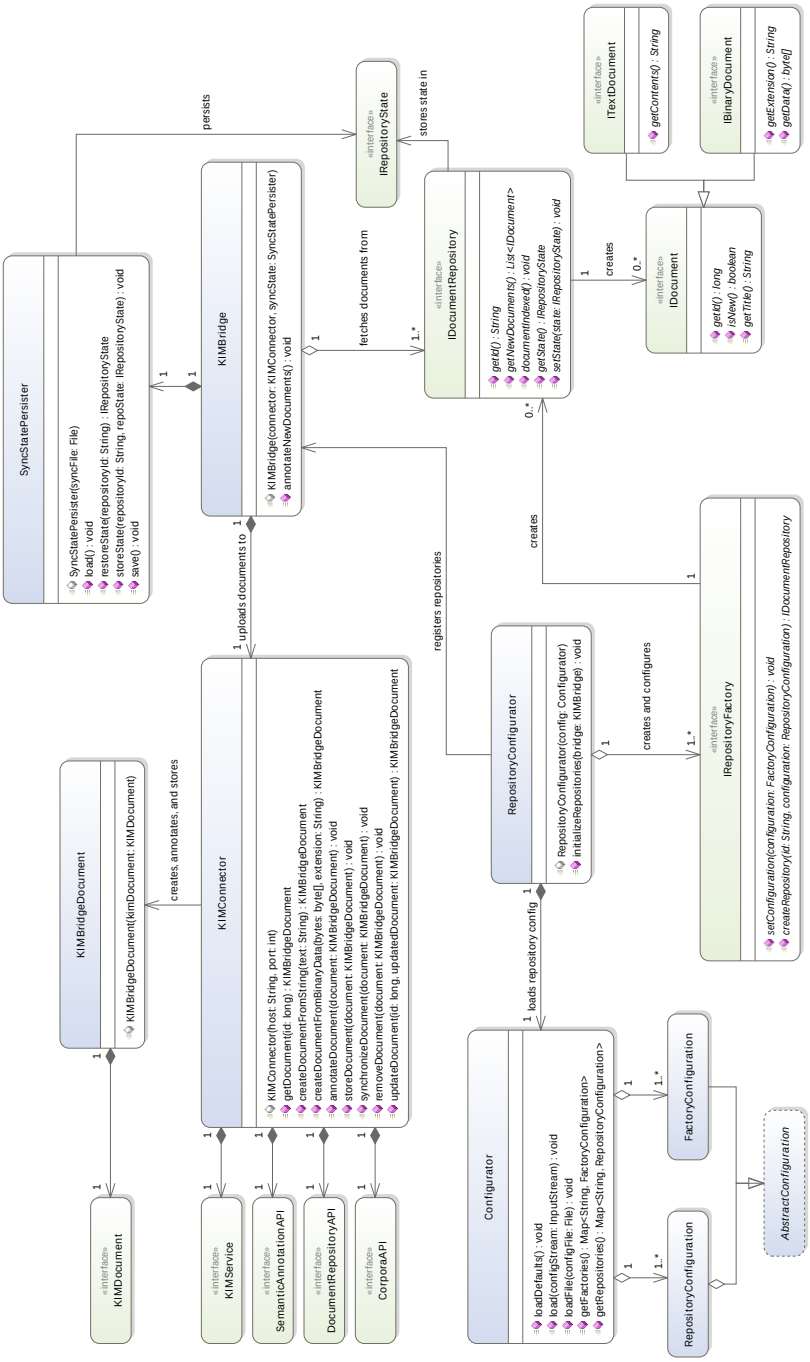


B Ontologie metod zpracování signálu

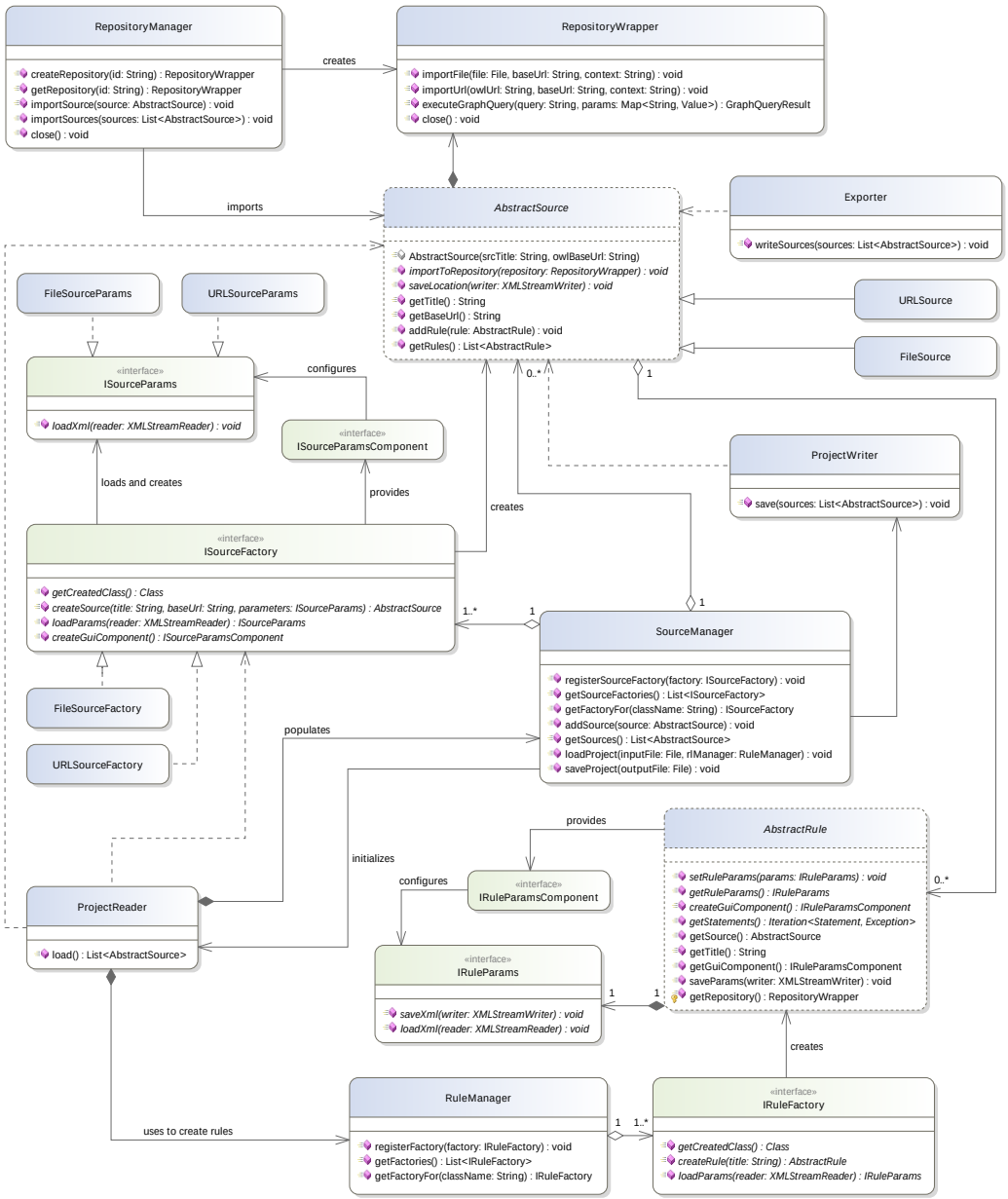
Autorem seznamu metod, jejich rozčlenění a grafu je Ing. Tomáš Řondík.



C UML diagram tříd v projektu KIMBridge



D UML diagram tříd v projektu KIM-OWLImport



E Uživatelská dokumentace k KIM-OWLImport

Zdrojové kódy KIM-OWLImport jsou k dispozici ve veřejném Git repositáři: `https://github.com/NEUROINFORMATICS-GROUP-FAV-KIV-ZCU/KIM-OWLImport`

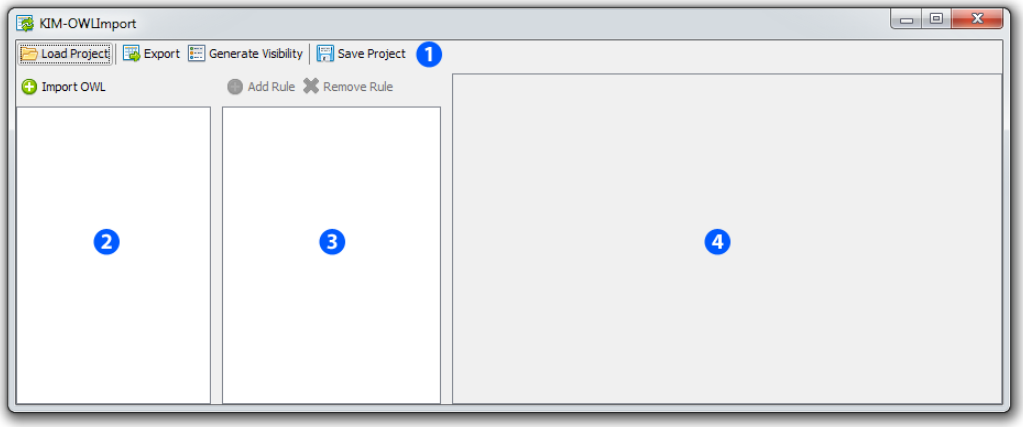
Program je možné přeložit nástrojem `ant`:

```
ant -f owlimport.xml
```

Ve složce `out/artifacts/OWLImport_jar` bude vytvořen spustitelný JAR soubor. Do stejného adresáře budou také nakopírovány všechny potřebné závislosti.

Hlavní okno programu (obrázek E.1) je rozděleno do 4 částí:

1. Nástrojová lišta,
2. seznam ontologií,
3. seznam pravidel aktuálně vybrané ontologie a
4. nastavení právě vybraného pravidla.



Obrázek E.1: Hlavní okno programu

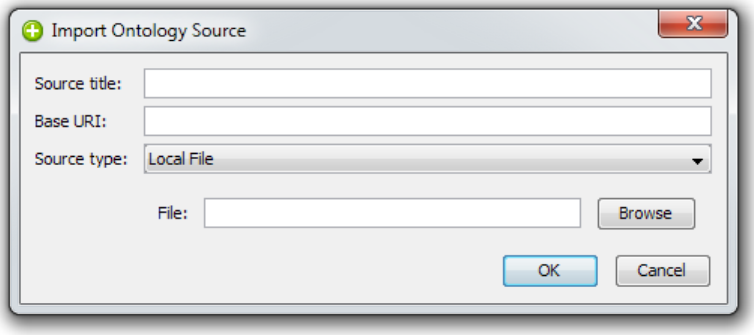
Práci s programem je možné začít buď načtením již dříve vytvořeného projektu tlačítkem **Load Project**, nebo rovnou přidáním ontologie tlačítkem **Import OWL**.

Při importu OWL souboru (obrázek E.2) je nutné zadat název ontologie (bude použit v nabídkách a pro vygenerování URL důvěryhodného zdroje entit), základní URL ontologie a vybrat její typ. V současné verzi je podporováno přidávání souborů uložených na disku a ontologií dostupných přes síť pomocí protokolu HTTP.

Ontologie může být uložená v jednom z následujících formátů:

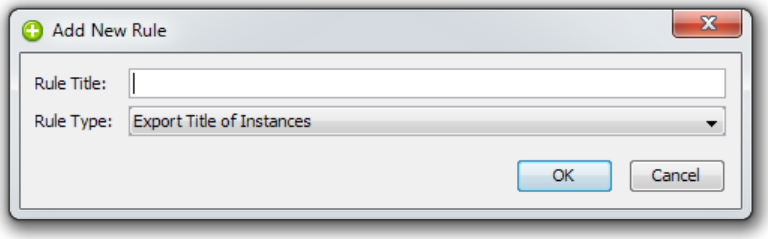
- RDF/XML,
- OWL/XML,
- Turtle,
- N-Triples a
- Notation-3 (N3).

V případě lokálního souboru i ontologie umístěné na webu je nutné, aby soubor měl příponu odpovídající danému formátu.



Obrázek E.2: Import ontologie.

Přidaná ontologie se objeví v seznamu ontologií. Jejím vybráním v seznamu je možné definovat pravidla pro export či transformaci ontologie pomocí tlačítka **Add Rule**. Při přidávání pravidla (obrázek E.3) je nutné zadat jeho název, který bude zobrazen v seznamu, a vybrat typ pravidla. Další volby konfigurace budou dostupné až po přidání pravidla.



Obrázek E.3: Přidání pravidla.

Po přidání pravidla se opět objeví v seznamu pravidel. Jeho vybráním je možné pravidlo konfigurovat v pravém panelu. Příklad konfigurace pravidla

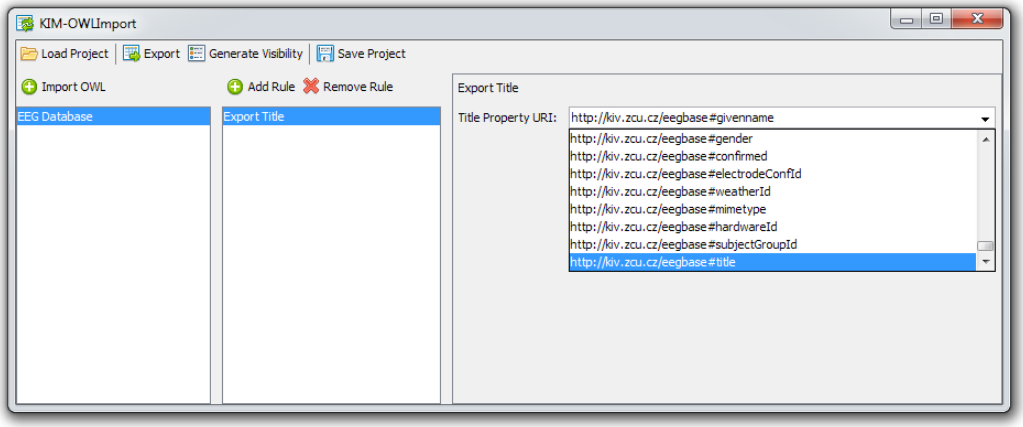
je znázorněn na obrázku E.4. Některá pravidla nemají žádné možnosti konfigurace.

Pro doplnění tripletů k ontologii, která již má strukturu vyžadovanou KIM Platform, je nutné přidat následující pravidla:

- Generate Trusted Source for All Entities a
- Align Root Classes to Proton Ontology (pouze pokud již třídy v ontologii nedědí od tříd definovaných v ontologii Proton).

Pravidla pro transformaci ontologií, jež ještě nejsou ve struktuře pro KIM, jsou závislá na formátu ontologie a je nutné pro ně dodefinovat pravidla programově.

Pravidlo je možné odstranit jeho vybráním v seznamu a následným stiskem tlačítka **Remove Rule**.



Obrázek E.4: Konfigurace pravidla.

Po přidání všech ontologií a definici pravidel pro export zbývá jen provést export. Tlačítkem **Export** lze zahájit k exportu všech přidaných ontologií podle k nim definovaných pravidel. Výsledný soubor je ve formátu Turtle.

Tlačítko **Generate Visibility** vygeneruje pravidla pro zobrazení tříd v ontologii v uživatelském rozhraní KIM Platform. Triplety z vytvořeného souboru je nutné přidat do `visibility.nt` v datech KIM.

Definovaná pravidla je možné uložit k pozdějšímu použití pomocí tlačítka **Save Project**. Do výsledného XML souboru jsou uloženy jen cesty k ontologiím, soubory proto musí i po opětovném otevření existovat.

F Uživatelská příručka k vyhledávání

Dokumenty jsou stahovány z Google Drive a z diskuzí v sociální síti LinkedIn nástrojem KIMBridge.

Pro stažení dokumentu ve formátu PDF z Google Drive jej stačí nasdílet s uživatelem `380431440078-079h1ai8dhiq3q57ks1og2fuu70f79k4@developer.gserviceaccount.com`.

Z LinkedIn jsou stahovány příspěvky z následujících skupin:

- Brain Pages – Where Professionals And Consumers Meet For Brain Health[1],
- Brain-Computer Interface group[2],
- Computational Neuroscience[3],
- EEG Signal Processing[4] a
- Neurofeedback[5].

Testovací webové rozhraní vyhledávání je k dispozici na adrese `http://eeg.kiv.zcu.cz:8080/KIM/`. Úvodní obrazovka obsahuje stručné informace o jednotlivých typech vyhledávání a seznam posledních dokumentů.

Webové rozhraní KIM poskytuje následující možnosti vyhledávání:

Structure search umožňuje vyhledat ve znalostní bázi entity, které splňují definované podmínky. Je možné určit třídu, omezení jednotlivých datových atributů či vztahy k jiným entitám. Vyhledávání také umožňuje vyhledat dokumenty, jež obsahují výskyty entit splňujících dané podmínky.

Tomuto vyhledávání je podobný **Pattern serach**, který obsahuje předdefinované podmínky a vztahy mezi jednotlivými entitami, do nichž jen uživatel vyplní hodnoty, které chce vyhledat. Tyto vzory vyhledávání jsou však nadefinovány přímo ve zdrojových kódech webové aplikaci a nelze je tedy snadno upravovat.

Boolean search umožňuje položit textový dotaz obsahující logické operátory AND a OR. Zadané termíny jsou vyhledány v dokumentech fulltextovým vyhledáváním a aliasy jednotlivých entit nejsou zohledněny.

Sekce **Ontology** zobrazuje ontologii, která je obsažena ve znalostní bázi. Je možné procházet jednotlivé třídy a jejich vlastnosti.

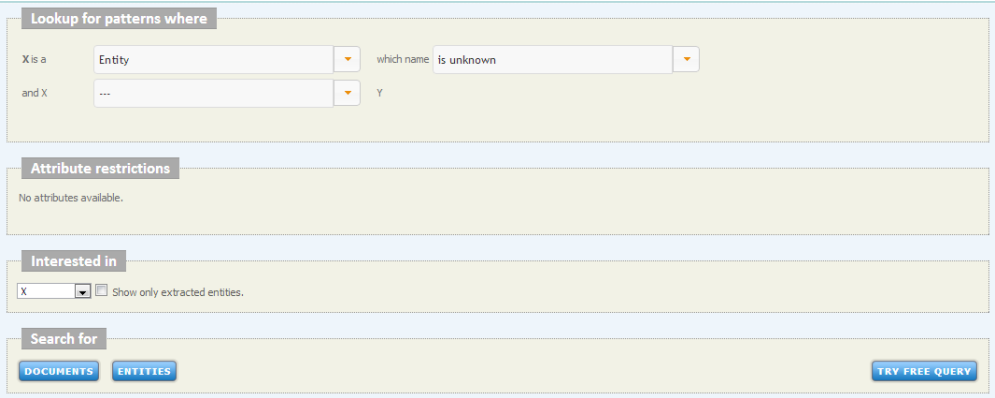
[1] `http://www.linkedin.com/groups?gid=4166025`
[2] `http://www.linkedin.com/groups?gid=1103077`
[3] `http://www.linkedin.com/groups?gid=1376707`
[4] `http://www.linkedin.com/groups?gid=959647`
[5] `http://www.linkedin.com/groups?gid=1825465`

Faceted search umožňuje postupně vybírat ze seznamů jednotlivé entity, jež chceme v dokumentech vyhledávat, a postupně tak zpřesňovat a filtrovat vyhledávací dotaz.

Pro uživatele jsou nejužitečnější sekce **Structure search** a **Faceted search**, proto budou nyní rozebrány podrobněji.

F.1 Structure search

Formulář na vyhledávání entit ve znalostní bázi podle zadaných podmínek je znázorněn na obrázku F.1. Umožňuje definovat podmínky podobně jako kdybychom skládali SPARQL dotaz.



Obrázek F.1: Formulář pro structure search.

Jeho funkci předvedeme na 2 příkladech vyhledávání: vyhledání dokumentů, které se týkají ERP komponent souvisejících se zpracováním jazyka s kladnou polaritou, a vyhledání filtračních metod zpracování signálu, jež je možné použít ke korekci baseline.

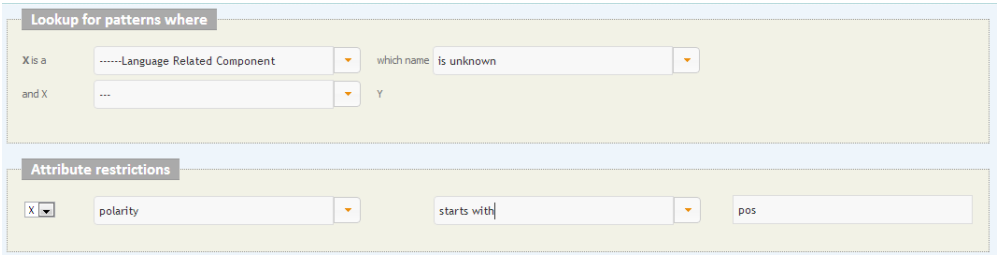
F.1.1 Vyhledání dokumentů týkajících se vybraných komponent

ERP komponenty v ontologii jsou rozděleny do několika tříd. Tyto třídy představují kategorie. Začneme výběrem „Language Related Component“ u políčka „X is an“. Tím určíme typ neznámé entity X.

Po výběru třídy dostaneme v sekci „Attribute restrictions“ možnost definovat omezení pro jednotlivé vlastnosti entity. ERP komponenty v ontologii mají jediný atribut: „polarity“. Ten vybereme v nabídce. V pravé části formuláře poté můžeme zadat hodnotu. Pokud bychom tak neučinili, byly by

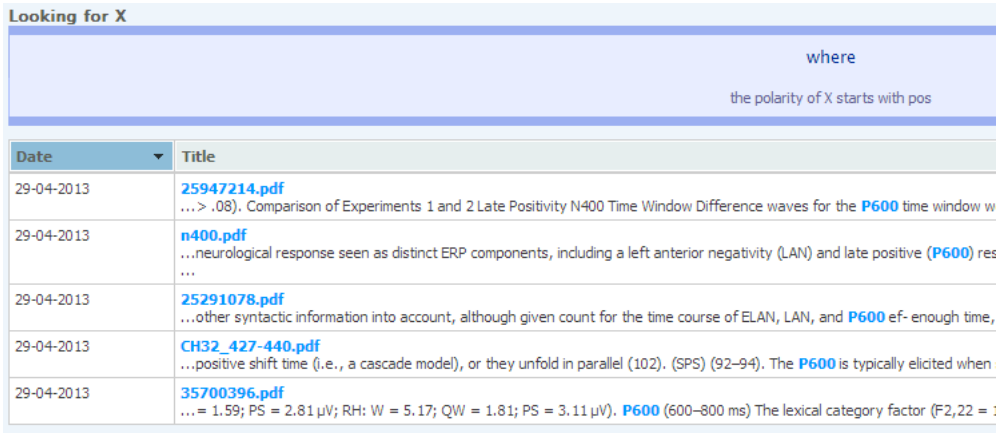
vyhledávány pouze komponenty, které mají uvedenou polaritu libovolné hodnoty. Webové rozhraní bohužel neumí pracovat s omezeními vlastností, proto nedostaneme na výběr seznam hodnot, jakých může nabývat. Lze však vybrat operátor „starts with“ a následně zadat jen začátek slova „positive“. Další možné hodnoty vlastnosti jsou „negative“ a „variable“.

Vyplněný formulář je znázorněn na obrázku F.2.



Obrázek F.2: Vyplněné hodnoty pro vyhledání kladných komponent souvisejících se zpracováním jazyka.

Vyhledání provedeme klepnutím na tlačítko „Documents“ v oddíle „Search for“. Výsledek je zachycen na obrázku F.3.



Obrázek F.3: Výsledek vyhledávání kladných komponent souvisejících se zpracováním jazyka.

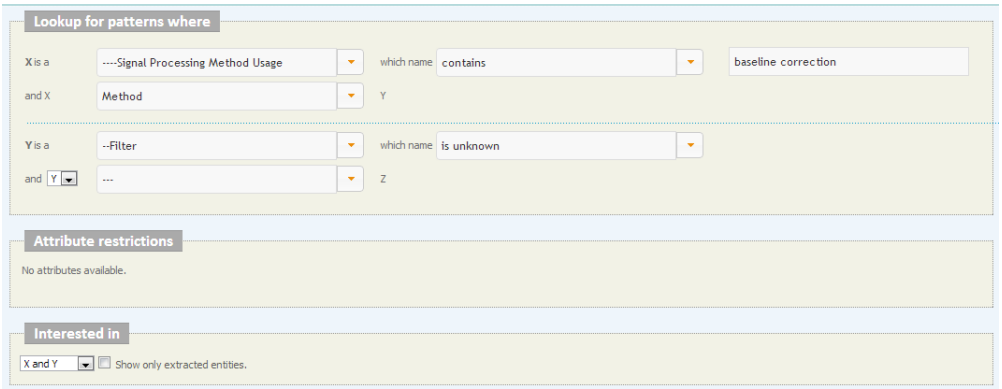
F.1.2 Vyhledání metod zpracování signálu podle zadaných podmínek

I metody pro zpracování signálu jsou rozděleny do tříd. Pokud chceme vyhledat filtrační metody zpracování, které lze použít ke korekci baseline, můžeme

postupovat dvěma způsoby:

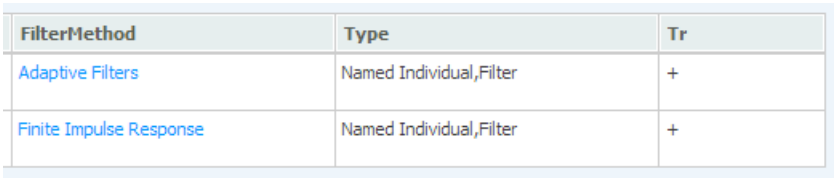
- Vybrat pro neznámou entitu X třídu „Filter“, jako vztah k neznámé entitě Y vybrat „Method Usage“ a jako název entity Y zvolit „baseline correction“.
- Vybrat pro neznámou entitu X třídu „Signal Processing Method Usage“, její název zvolit „baseline correction“, vztah k entitě Y „Method“ a třída entity Y „Filter“.

Výsledky obou dotazů jsou stejné, budou jen prohozeny entity X a Y. Ve druhém případě také musíme vybrat v sekci „Interested in“ hodnotu „X and Y“, jinak bychom ve výsledku nedostali požadovaný seznam metod. Vyplněný formulář pro druhý způsob zadání je znázorněn na obrázku F.4.



Obrázek F.4: Vyhledání metody zpracování signálu podle jejího možného použití.

Vyhledávání metod zahájíme tlačítkem „Entities“ v sekci „Search for“. Tlačítkem „Documents“ bychom opět vyhledávali dokumenty, jež obsahují vybrané entity. Výsledek vyhledávání je zachycen na obrázku F.5.



FilterMethod	Type	Tr
Adaptive Filters	Named Individual,Filter	+
Finite Impulse Response	Named Individual,Filter	+

Obrázek F.5: Výsledek vyhledání filtračních metod zpracování signálu, které je možné použít ke korekci baseline.

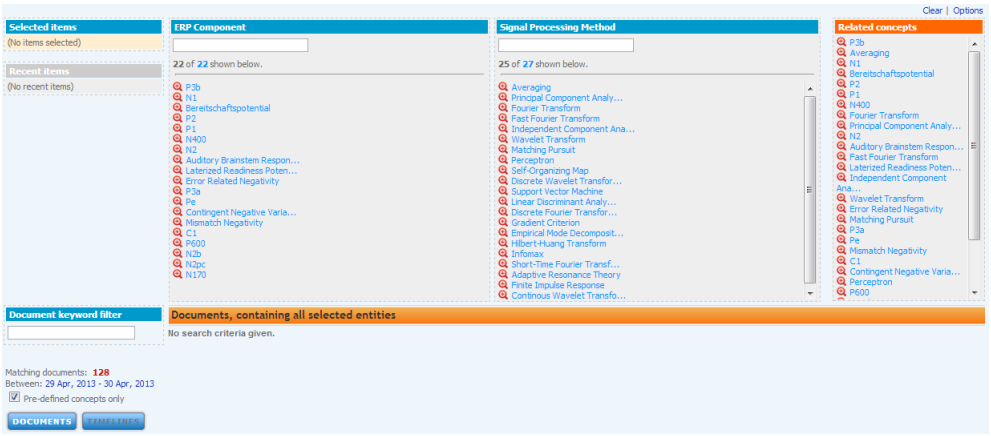
F.2 Faceted search

Tento typ vyhledávání umožňuje postupně filtrovat dokumenty, jež obsahují výskyty zadaných entit a klíčových slov. Hlavní obrazovka vyhledávání je znázorněna na obrázku F.6.

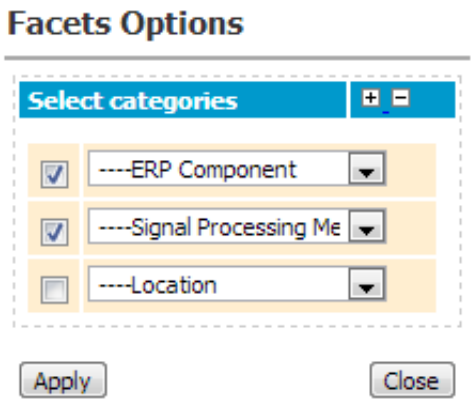
Ve výchozím nastavení umožňuje webové rozhraní filtrovat pouze entity typu osoba, organizace a místo. Pro filtraci entit typu „ERP Component“ a „Signal Processing Method“ je nutné v pravém horním rohu klepnout na odkaz „Options“ a vybrat vlastní kategorie pro filtraci. Možné nastavení je znázorněno na obrázku F.7.

Po nastavení kategorií je možné začít zadávat názvy entit, které chceme v dokumentech vyhledat. V seznamu pod textovým políčkem budou viditelné další entity, jež jsou ve zbývajících dokumentech ještě obsaženy. Jejich výběrem můžeme vyhledávání postupně upřesňovat.

V levé části obrazovky je políčko „Document keyword filter“. Do něj můžeme zadat klíčová slova, která chceme v dokumentech vyhledávat fulltextově. Příklad výsledku vyhledávání je znázorněn na obrázku 10.3 v sekci Výsledky.



Obrázek F.6: Faceted search.



Obrázek F.7: Nastavení kategorií pro filtraci.

G Instalace služby KIMBridge

Zdrojové kódy KIMBridge jsou k dispozici ve veřejném Git repositáři:
`https://github.com/NEUROINFORMATICS-GROUP-FAV-KIV-ZCU/KIMBridge`

Program KIMBridge je možné zkompilovat nástrojem `ant`:

```
ant -f kimbridge.xml
```

Ve složce `out/artifacts/KIMBridge_jar` bude vytvořen spusitelný JAR soubor. Do stejné složky budou také přibaleny všechny závislosti potřebné pro spuštění a příklad konfigurace v souboru `config.example.xml`.

G.1 Spuštění

Před spuštěním služby KIMBridge je nutné nastavit přístupové klíče k API vzdálených služeb a nastavit jednotlivé repositáře. Konfigurace je popsána v kapitole 8.

Služba se připojuje k instanci KIM Platform na lokálním stroji, proto musí být nastavena na stroj, kde služba KIM běží.

KIMBridge může běžet ve dvou režimech: konzolová aplikace a linuxová služba. V obou případech je nutné službu pouštět s následujícími parametry, jinak nebude Java RMI dostupné:

```
-Djava.security.manager -Djava.security.policy=security.policy
```

G.1.1 Konzolová aplikace

Před spuštěním konzolové aplikace je nutné vytvořit konfigurační soubor `config.xml` a umístit jej do pracovního adresáře aplikace. Aplikaci je možné spustit příkazem `java`:

```
java -Djava.security.manager -Djava.security.policy=security.policy
    -jar KIMBridge.jar
```

Veškeré záznamy o průběhu synchronizace budou vypisovány do konzole. Běh aplikace je možné ukončit klávesami Ctrl-C.

G.1.2 Linuxová služba

Pro spuštění v režimu linuxové služby je potřeba mít nainstalovaný nástroj `jsvc`. V distribuci Debian je k dispozici ve stejnojmenném balíčku. Spouštění a zastavování služby je možné inicializačním skriptem, který je přiložen ve

složce `init.d` se sestaveným JAR souborem. Před jeho použitím je nutné nastavit následující proměnné na začátku inicializačního skriptu:

`KB_HOME` – složka s KIMBridge,
`JDK_HOME` – cesta k instalaci Javy,
`JSVC` – cesta ke spustitelnému souboru `jsvc` a
`CONFIG` – umístění konfiguračního souboru.

Všechny cesty musejí být uvedeny absolutně.

Záznamy o běhu služby jsou vypisovány do souborů `kimbridge.out` a `kimbridge.err`.