

Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Bachelorarbeit Nr. 258

Extraktion und Visualisierung von Ontologie-Informationen aus Linked Data

Marc Weise

Studiengang: Softwaretechnik

Prüfer/in: Prof. Dr. Thomas Ertl

Betreuer/in: Dr. Steffen Lohmann,
Dipl.-Inf. Florian Haag

Beginn am: 8. Oktober 2015

Beendet am: 8. April 2016

CR-Nummer: H.3.3, H.5.2

Kurzfassung

Um das Potential von Linked Data nutzen zu können, muss der Nutzer sich zunächst einen Überblick über Inhalt und Struktur des zu untersuchenden Datensatzes verschaffen. Dazu wird in dieser Arbeit ein Ansatz für die Extraktion von Ontologie-Informationen aus Linked Data vorgestellt. Dabei werden die im Datensatz enthaltenen Entitäten Klassen zugeordnet. Ausgehend von den mächtigsten dieser Klassen werden durch generierte SPARQL-Abfragen hinterlegte Informationen und Beziehungen, sowie die im Datensatz verwendeten Vokabulare schrittweise aufgedeckt. Die extrahierten Ontologie-Informationen werden während der Extraktion sukzessive mittels einer modifizierten Form der Visual Notation for OWL (VOWL) grafisch dargestellt. Dieser Ansatz wird in Form einer Webanwendung prototypisch umgesetzt und im Rahmen einer Nutzerstudie evaluiert.

Inhaltsverzeichnis

1. Einleitung	11
2. Grundlagen	13
2.1. Linked Data	13
2.1.1. Prinzipien	13
2.1.2. Linking Open Data Projekt	14
2.2. Ontologien	14
2.3. Resource Description Framework (RDF)	14
2.4. Resource Description Framework Schema (RDFS)	15
2.5. OWL2 Web Ontology Language	15
2.6. SPARQL	16
3. Themenverwandte Arbeiten	19
3.1. Extraktion von Ontologien	19
3.1.1. Identifizieren von Schlüsselkonzepten in Ontologien	19
3.1.2. Extraktion von Kerngedanken aus Linked Data	20
3.1.3. Auf Zusammenfassungen basierendes Explorieren der RDF-Verwendung in der LOD Cloud	20
3.2. Visualisierung von Linked Data	22
3.2.1. RelFinder	22
3.2.2. ResXplorer	23
3.2.3. LodLive	24
3.3. Visualisierung von Ontologien	26
3.3.1. Jambalaya	26
3.3.2. OntoGraf	27
3.3.3. OntoViz	28
3.3.4. OntoRama	29
3.3.5. TGVizTab	30
3.3.6. IsaViz	30
3.3.7. OntoSphere	31
3.3.8. Visual Notation for OWL Ontologies (VOWL 2)	32
3.3.9. Implementierungen von VOWL	32
3.3.10. QueryVOWL	35

4. Visuelle Notation	37
4.1. Klassen	38
4.2. Relationen	38
4.3. Pfeilspitzen von Relationen	38
4.4. Rechtecke	39
4.5. Text	39
4.6. Farbpalette	39
4.7. Mehrfache Darstellung von Properties und Datentypen	42
4.8. Nicht enthaltene Elemente aus VOWL	43
5. Entwickeltes System	45
5.1. Voraussetzungen	45
5.1.1. SPARQL-Endpunkt	45
5.1.2. Browser	46
5.2. Anforderungen an das System	46
5.2.1. Funktionale Anforderungen	46
5.2.2. Nicht-funktionale Anforderungen	46
5.3. Software-Architektur	47
5.3.1. Verwendete Technologien	47
5.3.2. Komponenten	48
5.3.3. Konnektoren	50
5.4. Allgemeine Funktionsweise	50
5.4.1. Überblick	50
5.4.2. Auswahl des SPARQL-Endpunkts	50
5.4.3. Extraktion der wichtigsten Klassen	51
5.4.4. Erkennung von Äquivalenz- und Subklassenbeziehungen	52
5.4.5. Suchen nach Beziehungen zwischen den Klassen	53
5.4.6. Suche nach Datentypen	54
5.4.7. Blacklists für Klassen und Properties	56
5.4.8. Caching	57
5.5. Interaktion mit der Visualisierung	58
5.5.1. Wahl des Bildausschnitts (Zoom)	59
5.5.2. Selektion (Details on Demand)	59
5.5.3. Filterung der Darstellung (Filter)	60
5.5.4. Anpassung der Kantenlängen	61
5.5.5. Klassifikation von externen und internen Namensräumen	62
6. Evaluation	63
6.1. Entwurf der Nutzerstudie	63
6.1.1. Hypothesen	63
6.2. Fragen für die Nutzerstudie	64
6.2.1. Allgemeine Fragen	64

6.2.2.	Endpunktspezifische Fragen	64
6.2.3.	Auswahl der Endpunkte für die Nutzerstudie	65
6.3.	Versuchsaufbau	66
6.4.	Durchführung der Nutzerstudie	66
6.5.	Studienteilnehmer	66
6.6.	Ergebnisse der Nutzerstudie	67
6.6.1.	Fragebögen zu Datensätzen	67
6.6.2.	Qualitatives Feedback	69
6.7.	Untersuchung der Performance	70
6.7.1.	Vorgehen	70
6.7.2.	Untersuchte Endpunkte	70
6.7.3.	Ergebnisse	70
6.8.	Skalierbarkeit	73
7.	Fazit und Ausblick	75
A.	Dokumente für Nutzerstudie	77
A.1.	Einverständniserklärung	77
A.2.	LDVOWL Übersicht Notation	79
A.3.	Fragebögen	80
A.3.1.	Demografische Daten und Vorkenntnisse	80
A.3.2.	Endpunkt 1	81
A.3.3.	Endpunkt 2	83
A.3.4.	Feedback	85
B.	Begriffslexikon	87
	Literaturverzeichnis	89

Abbildungsverzeichnis

3.1.	Graphische Darstellung des RelFinders nach Abschluss der Suche nach Beziehungen zwischen den Mathematikern Alan Turing und Kurt Gödel auf DBpedia	22
3.2.	Beispiel für die Graphdarstellung der Anwendung „ResXplorer“ um den Eintrag für Tim Berners-Lee	24
3.3.	Beispiel für die Graphdarstellung der Anwendung „LodLive“ beim Durchsuchen von DBpedia ausgehend von Alan Turing	25
3.4.	Zwei Beispiele für verschiedene Darstellungen der Pizza-Ontologie mit Hilfe des Plug-ins „Jambalaya“ für Protégé 3.4	27
3.5.	Darstellungen von Klassen und Instanzen der Weinontologie in verschiedenen Layouts mit OntoGraf	27
3.6.	Beispiel für die grafische Darstellung einer Ontologie im Protégé Plug-in „OntoViz“ aus [Sin]	28
3.7.	Screenshot aus OntoRama bei der Darstellung einer Beispielontologie	29
3.8.	Darstellung der Newspaper-Ontologie mit dem TGVizTab Plug-in unter Protégé 3.4	30
3.9.	Dreidimensionale Darstellung eines Ausschnittes der Pizza-Ontologie in OntoSphere3D (Eclipse) aus [Ont]	31
3.10.	Darstellung der Ontologie „Modular Unified Tagging Ontology“ (MUTO) in ProtégéVOWL	33
3.11.	Darstellung der Ontologie „Friend of a Friend“ (FOAF) in WebVOWL	34
3.12.	Beispiel für die visuelle Darstellung einer SPARQL-Abfrage in QueryVOWL	35
5.1.	UML-Komponentendiagramm der Anwendung LDVOWL	48
5.2.	UML-Paketdiagramm für das Sidebar Paket	49
5.3.	UML-Diagramme zum Service-Paket der Anwendung LDVOWL	49
5.4.	Startseite der Webanwendung LDVOWL mit Eingabefeld zur Auswahl des zu untersuchenden SPARQL-Endpunktes	51
5.5.	Graphische Darstellungen der mächtigsten Klassen für den DBpedia Endpunkt	53
5.6.	Beispiele für Darstellungen von Klasse-Datentyp Relationen in LDVOWL . .	55
5.7.	Einstellungsseite der Anwendung LDVOWL mit editierbaren Blacklists . . .	57
5.8.	Darstellung von LDVOWL für den SPARQL-Endpunkt Transparency International Linked Data (http://transparency.270a.info/sparql)	59

5.9. Details für verschiedene ausgewählte Elemente in Auswahlgruppe der Sidebar in LDVOWL	60
5.10. Sidebar-Gruppe für Filterung der Graphansicht von LDVOWL	60
5.11. Sidebar-Gruppe für Grapheinstellungen in LDVOWL	61
5.12. Beispiel für Darstellung der Namensraumgruppe in der Sidebar von LDVOWL (auf DBpedia Endpunkt)	62
6.1. Darstellung der für die vollständige Extraktion der Ontologie-Informationen benötigten Zeit nach Endpunkt	71
6.2. Durchschnittliche Anzahl der von LDVOWL während der Extraktion gesendeten SPARQL-Abfragen pro Minute	72

Tabellenverzeichnis

4.1. Übersicht über die in LinkedDataVOWL verwendeten graphischen Primitive und deren Bedeutung	37
4.2. Übersicht über die für LDVOWL verwendeten Farben und deren Bedeutung	41
5.1. Klassen-Blacklist für RDF-, RDFS- und OWL-Namensräume	58
5.2. Property-Blacklist für RDF, RDFS und OWL-Namensräume	58
5.3. In VOWL verfügbare Filteroptionen sowie deren Voreinstellung	61
6.1. Liste der für die Untersuchung der Performance ausgewählten SPARQL-Endpunkte	71

Verzeichnis der Listings

2.1. Beispiel für eine SPARQL-Abfrage nach den zehn Ländern, in welchen die meisten Nobelpreisträger geboren wurden	16
5.1. SPARQL-Abfrage nach den zehn Klassen mit den meisten Instanzen	52
5.2. SPARQL-Abfrage für Anzahl gemeinsamer Instanzen der Klassen A und B	52

5.3.	Beispiel für eine SPARQL-Abfrage nach einer sortierter Liste verschiedener Properties zwischen den Instanzen der Klassen „agent“ und „Document“ auf DBpedia	54
5.4.	Beispiel für SPARQL-Abfrage nach verschiedenen Properties zwischen Instanzen der Klassen „agent“ und „human“ auf DBpedia	54
5.5.	Beispiel für SPARQL-Abfrage nach Datentypen, welche für Instanzen von „agent“ am häufigsten auftreten	55
5.6.	Beispiel für die Abfrage nach Properties zwischen Instanzen der Klasse „Agent“ und dem Datentyp „string“	56

1. Einleitung

Durch die Erweiterung der im Web hinterlegten Informationen um explizite und maschinenlesbare Bedeutungen sowie die wachsende Verbreitung der damit verbundenen Standards wie z. B. RDF gewinnen unter den Begriffen „Semantic Web“ „Web of Data“ oder „Linked Data“ (LD) einzuordnende Themen stetig an Bedeutung.

Mit einer wachsenden Zahl von Linking Open Data (LOD)-Projekten wie z. B. DBpedia oder GeoNames stehen bereits heute riesige Mengen von verknüpften Daten öffentlich zur Verfügung, welche für die Beantwortung diverser Fragestellungen frei genutzt werden können. Unternehmen sowie staatliche Stellen wie Regierungen veröffentlichen Daten zunehmend in dieser Form.

Um das Potential dieser Datensätze voll ausschöpfen zu können, ist es allerdings entscheidend, dass sich Nutzer schnell einen Überblick über die Inhalte eines Datensatzes verschaffen können. Da die Datensätze keinem einheitlichem Schema folgen, sondern Vokabulare aus verschiedenen Ontologien wiederverwenden, ist dies für beliebige Datensätze nur schwer möglich.

Zielsetzung

In dieser Arbeit soll ein Ansatz für die Extraktion von Ontologie-Informationen aus Linked Data Datensätzen vorgestellt werden, sowie eine interaktive Visualisierung der extrahierten Informationen. Mit Hilfe dieser Visualisierung soll der Nutzer einen Überblick über die Inhalte des untersuchten Datensatzes bekommen. Für die Darstellung der extrahierten Ontologie-Informationen soll die bereits existierende Visual Notation for OWL Ontologies (VOWL) erweitert und auf dieses neue Einsatzgebiet angepasst werden.

Die Extraktion und Visualisierung der Ontologie-Informationen sollen prototypisch in Form einer Webanwendung umgesetzt werden. Zur Evaluation des Ansatzes wird die entwickelte Anwendung im Rahmen einer Nutzerstudie untersucht.

Gliederung

Der verbleibende Teil dieser Bachelorarbeit gliedert sich wie folgt:

Kapitel 2 beschreibt die Grundlagen von Linked Data und führt in die verwendeten Technologien ein.

Kapitel 3 bietet einen Überblick über verwandte Arbeiten aus den Bereichen Extraktion von Kerngedanken sowie Werkzeuge und Notationen zur Visualisierung von Linked Data und Ontologien.

Kapitel 4 beschreibt die für die Visualisierung verwendete visuelle Notation.

Kapitel 5 beschreibt das implementierte System und dessen Funktionsweise.

Kapitel 6 evaluiert das implementierte System im Rahmen einer Nutzerstudie und untersucht die Performance des gewählten Ansatzes.

Kapitel 7 zieht ein Fazit und stellt einige Anknüpfungspunkte für mögliche Weiterentwicklungen des vorgestellten Ansatzes sowie dessen Implementierung vor.

Anhang A enthält alle im Rahmen der Nutzerstudie verwendeten Dokumente.

Anhang B führt die in dieser Arbeit verwendeten Fachbegriffe und Abkürzungen auf und erklärt diese kurz.

2. Grundlagen

Im Folgenden werden einige Grundlagen des Semantic Web vorgestellt. Dabei wird speziell auf Linked Data (LD), Ontologien sowie die dafür verwendeten Technologien wie RDF, RDFS, OWL und SPARQL eingegangen.

Zunächst wird jedoch der Begriff „Linked Data“ erklärt und die damit verbundenen Prinzipien näher beschrieben.

2.1. Linked Data

Bizer et al. definieren Linked Data als Menge von Best Practices zur typisierten Verknüpfung von strukturierten Daten im Web [BHB09]. Im Gegensatz zur herkömmlichen Darstellung von Daten im Web soll LD maschinenlesbar und selbstbeschreibend sein, sowie Bedeutung und Beziehungen der Daten explizit definieren. Hierbei entsteht durch die Verknüpfung von Daten aus verschiedenen Bereichen ein „Web of Data“. Die einzelnen Datensätze stellen typisierte Aussagen über beliebige Dinge dar, welche als Tripel bestehend aus Subjekt, Prädikat und Objekt ausgedrückt werden.

2.1.1. Prinzipien

Berners-Lee [BL06] nennt für die Veröffentlichung von LD die folgenden vier Regeln: Erstens sollen Universal Resource Identifier (kurz URI) für die Benennung von Dingen verwendet werden. Zweitens sollen die URIs per HTTP nachschlagbar sein. Drittens sollen auf Anfrage nützliche Informationen auf standardisierte Art und Weise zur Verfügung gestellt werden (z. B. mit SPARQL und RDF). Viertens sollen Verknüpfungen zu anderen URIs angeboten werden.

Auf die von Berners-Lee vorgeschlagenen Technologien wird später näher eingegangen.

2.1.2. Linking Open Data Projekt

Bizer et al. beschreiben das Linking Open Data Projekt in [BHB09]. Im Januar 2007 gegründet, verfolgt es das Ziel, Datensätze unter freien Lizenzen zu identifizieren, in RDF zu konvertieren und anschließend mit bereits vorhandenen Daten verknüpft zu veröffentlichen.

Während zu diesem Projekt vor allem wissenschaftliche Mitarbeiter und Entwickler beitragen, beteiligen sich mittlerweile ebenfalls große Unternehmen wie z. B. BBC. sowie staatliche Stellen wie Regierungen und Verwaltungen.

Die Inhalte der veröffentlichten Daten sind vielfältig: Neben geographischen Daten und öffentlich bekannten Personeninformationen sind u. a. auch Informationen über Bücher, wissenschaftliche Publikationen, Filme und Musik verfügbar. Da viele Daten in Form von Wrappern um relationale Datenbanken vorliegen, ist die Erfassung des genauen Umfangs schwierig.

2.2. Ontologien

Uschold und Gruninger beschreiben eine Ontologie als ein gemeinsames Verständnis einer bestimmten Domäne [UG96]. Damit werden Entitäten sowie deren Attribute und Beziehungen innerhalb dieser Domäne beschrieben. Ontologien enthalten ein Vokabular von Begriffen sowie Definitionen deren Bedeutung. Diese Definitionen können dabei verschieden formal ausgedrückt werden. Ontologien können damit als konzeptuelles Rahmenwerk für Domänenwissen dienen und werden unter anderem für die Kommunikation zwischen Menschen, die Kompatibilität zwischen Systemen oder für die Spezifikation und den Bau dieser eingesetzt.

2.3. Resource Description Framework (RDF)

Das 2004 in [KC04] erstmalig beschriebene Resource Description Framework (RDF) definiert eine Darstellung von Informationen im Internet. Hierbei werden Informationen aus einem bestimmten Fachbereich in Form eines RDF-Graphen dargestellt. Dieser Graph wird aus einer Menge von RDF-Tripeln gebildet, welche Subjekt, Prädikat und Objekt einer bestimmten Aussage darstellen. Die einzelnen Elemente dieser Tripel können hierbei Internationalized Resource Identifier (IRIs), leere Knoten oder typisierte Literale sein. [KC04] beschreibt die abstrakte Syntax für RDF, welche beispielsweise in XML konkretisiert wird.

2.4. Resource Description Framework Schema (RDFS)

Das in [GB14] beschriebene „Resource Description Framework Schema“ (RDFS) erweitert das RDF-Vokabular um Mittel zur Datenmodellierung. Mit RDFS können Gemeinsamkeiten unterschiedlicher Ressourcen ausgedrückt werden, indem diese Klassen zugeordnet oder der Definitions- (`rdfs:domain`) und Wertebereich (`rdfs:range`) von Properties definiert werden.

Wie in der objektorientierten Programmierung (OOP) können Klassen (`rdfs:Class`) definiert werden und diese über eine Kindklassenbeziehung (`rdfs:subClassOf`) in Verbindung gesetzt werden. Mit `rdfs:subPropertyOf` kann auch für Properties eine solche Hierarchie modelliert werden. Außerdem können mit den Kindklassen von `rdfs:Container` Mengen (`rdfs:Bag`), Listen (`rdfs:List`) und Alternativen `rdfs:Alt` dargestellt werden.

Um Aussagen modellieren zu können, bietet RDFS ein Vokabular zur Vergegenständlichung von Aussagen (`rdfs:Statement`) sowie deren Subjekt (`rdfs:subject`), Prädikat (`rdfs:predicate`) und Objekt (`rdfs:object`).

Schließlich definiert RDFS mit `rdfs:label` und `rdfs:comment` zwei Properties, mit welchen für den Mensch lesbare Bezeichnungen und Erläuterungen für Ressourcen zur Verfügung gestellt werden können.

2.5. OWL2 Web Ontology Language

Mit Hilfe von OWL 2 [Owl] können Ontologien beschrieben werden. Hierbei werden für diese Domäne wichtige Begriffe definiert und ihre Beziehungen untereinander beschrieben.

OWL 2 erweitert die im Jahr 2004 veröffentlichte OWL 1 Web Ontology Language.

Die drei wichtigsten Bestandteile von OWL 2 sind die Struktur der Ontologie als RDF-Graph, die Spezifikation der Bedeutung (Semantik) sowie eine Menge konkreter Syntaxen, welche für den Austausch von Ontologien benötigt werden (z. B. OWL/XML oder Turtle).

Die Semantik wird in zweierlei Art definiert: Zum einen durch die direkte Semantik, welche den einzelnen Strukturen der Ontologie eine bestimmte Bedeutung zuweist. Zum anderen durch eine RDF-basierte Semantik, welche dem RDF-Graphen eine Bedeutung zuweist und damit die Semantik von RDF erweitert.

OWL 2 steht in drei verschiedenen Profilen zur Verfügung, welche jeweils einer Teilmenge von OWL entsprechen: OWL 2 EL, OWL 2 QL und OWL 2 RL.

2.6. SPARQL

Die Abkürzung „SPARQL“ steht rekursiv für „SPARQL Protocol and RDF Query Language“. Diese in [PS08] spezifizierte Abfragesprache dient folglich der Abfrage von Graphmustern aus einem in einem Endpunkt hinterlegten RDF-Graphen.

Je nach Art der Abfrage wird als Ergebnis entweder eine Ergebnismenge oder ein RDF-Graph zurückgegeben. SPARQL erlaubt vier verschiedene Abfragearten:

Auf eine Abfrage mit SELECT werden die Ergebnisse tabellarisch zurückgegeben, die Syntax ähnelt dabei stark der Abfragesprache SQL für relationale Datenbanken. CONSTRUCT wird mit einem neuen RDF-Graphen beantwortet, in welchem die Variablen durch die gefundenen Ergebnisse ersetzt wurden. Mit Hilfe von ASK kann geprüft werden, ob ein bestimmtes Muster im hinterlegten RDF-Graphen existiert oder nicht. Das Ergebnis ist hier immer entweder ja oder nein. Die letzte Abfrageart wird mittels DESCRIBE eingeleitet und wird vom Endpunkt mit dem gefundenen RDF-Graphen beantwortet.

Da die in dieser Arbeit vorgestellte Ontologie-Extraktion auf der Verwendung von SELECT-Abfragen basiert, wird deren Aufbau im Folgenden näher beschrieben. Listing 2.1 zeigt ein Beispiel einer solchen SPARQL-Abfrage.

```
1 PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
2 PREFIX nobel: <http://data.nobelprize.org/terms/>
3 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
4
5 SELECT ?country (COUNT(?person) AS ?nobelPrizeCount)
6 WHERE {
7     ?person a foaf:Person .
8     ?price a nobel:NobelPrize .
9     ?country a dbpedia-owl:Country .
10    ?person nobel:nobelPrize ?price .
11    ?person dbpedia-owl:birthPlace ?country .
12 }
13 GROUP BY ?country
14 ORDER BY DESC(?nobelPrizeCount)
15 LIMIT 10
16 OFFSET 0
```

Listing 2.1: Beispiel für eine SPARQL-Abfrage nach den zehn Ländern, in welchen die meisten Nobelpreisträger geboren wurden

Zunächst erfolgt die Definition von Präfixen für die verwendeten Namensräume (Zeile eins bis drei). Darauf folgt eine durch Leerzeichen getrennte Auflistung der Variablen, welche zurückgegeben werden sollen (Zeile fünf). Anschließend wird in einem mit geschweiften Klammern umschlossenen WHERE-Block das zu suchende Tripelmuster definiert (Zeilen sechs bis zwölf). Dieses Muster besteht aus einer Liste von Subjekt, Prädikat und Objekt, welche

mit Leerzeichen und Punkten getrennt werden. Die einzelnen Ressourcen können IRIs, leere Knoten, Literale oder Variablen sein. Alle Tripel werden durch ein logisches „Und“ verknüpft, des weiteren können Teile des Graphmusters, welche nicht zwingend vorhanden sein müssen, mit dem Schlüsselwort `OPTIONAL` hinzugefügt werden. Nach dem Abschluss des `WHERE`-Blocks können die Ergebnisse noch sortiert und gruppiert werden, sowie mittels `OFFSET` und `LIMIT` wie in `SQL` eingeschränkt werden (Zeile 13 bis 16).

3. Themenverwandte Arbeiten

Im Folgenden werden einige themenverwandte Arbeiten aus verschiedenen Kategorien vorgestellt. Zunächst wird in Abschnitt 3.1 auf bereits existierende Ansätze zur Extraktion von Konzepten und Ontologien eingegangen, da dies eine wesentliche Herausforderung dieser Arbeit darstellt.

Die extrahierten Informationen sollen nach ihrer Beschaffung dem Benutzer in einer leicht verständlichen Form präsentiert werden, deshalb widmen sich anschließend die Abschnitte 3.2 und 3.3 verschiedenen bereits existierenden Ansätzen zur graphischen Darstellung von Linked Data bzw. Ontologien in Form eines Knoten-Kanten-Graphen. In diesen beiden Abschnitten werden verschiedene Ansätze und Werkzeuge für eine solche Darstellung vorgestellt, darunter unter anderem auch VOWL, welcher in leicht modifizierter Form für den in dieser Arbeit vorgestellten Ansatz und dessen Implementierung angewendet wurde.

3.1. Extraktion von Ontologien

Durch die Extraktion von Ontologie-Informationen aus einem Datensatz erhält der Benutzer einen guten Überblick über Inhalte und Struktur des Datensatzes. Im Folgenden werden bereits existierende Ansätze für die Identifikation und Extraktion von Kerngedanken und Schlüsselkonzepten vorgestellt.

3.1.1. Identifizieren von Schlüsselkonzepten in Ontologien

Peroni et al. stellen in [PMd08] einen Ansatz zur automatischen Identifikation der wichtigsten Konzepte in einer Ontologie vor. Dabei werden Kriterien aus Bereichen der Kognitionswissenschaft, lexikalischen Statistik und der Graphanalyse angewandt, um in einer gegebenen Ontologie die n Konzepte zu finden, welche die Ontologie am besten beschreiben. Diese Konzepte werden „key concepts“ genannt. In ihrem ersten Ansatz verwenden Peroni et al. die Länge der Konzeptnamen sowie deren Zentralität, um natürliche Kategorien in der Ontologie zu finden.

Die ausgewählten Konzepte sollen zum einen über eine hohe Informationsdichte verfügen, zum anderen jedoch auch einen großen Anteil aller Konzepte in der Ontologie abdecken.

Die Auswahl wurde mit der von Experten verglichen. Um die Übereinstimmung zu verbessern wurde in einem zweiten Ansatz das Vorgehen durch eine Gewichtung der Konzepte nach ihrer Popularität erweitert, welche sich über die Anzahl der Treffer einer Suchmaschine definiert.

Anders als in [PMd08] soll die in dieser Arbeit vorgestellte Anwendung keine bestehende Ontologie durch Extraktion von Schlüsselkonzepten zusammenfassen, sondern aus den ABox-Informationen eines LD-Datensatzes eine Ontologie extrahieren.

3.1.2. Extraktion von Kerngedanken aus Linked Data

Presutti et al. beschreiben einen Ansatz zur Extraktion von Kerngedanken aus LD-Datensätzen [Pre+11]. Dabei werden Datensätze, deren Vokabular zunächst nicht bekannt ist, untersucht und Knowledge Patterns (KPs) innerhalb dieser identifiziert. Um dies zu erreichen, wird eine Dataset Knowledge Architecture erstellt, welche die wichtigsten Eigenschaften des zu untersuchenden Datensatzes in Form von KPs und Typ-Property-Pfaden enthält. Dadurch wird neues Wissen über den untersuchten Datensatz generiert, indem zentrale Typen und Properties sowie Muster identifiziert werden.

Das Vorgehen wird in [Pre+11] wie folgt beschrieben: Zunächst werden Statistiken zur Verwendung von Properties erhoben. Anschließend werden Anfragen zur Pfadextraktion an den Datensatz gestellt und die extrahierten Pfade in der Knowledge Architecture hinterlegt. Zentrale Typen und Properties werden durch die Anzahl ihrer Instanzen bzw. ihre „between-ess“ ermittelt. Darauf aufbauend werden KPs extrahiert und schließlich nach einer Auswahl von Faktoren zur Clusterbildung zu einem Pfadcluster zusammengesetzt.

Die Knowledge Architecture stellt eine modulare Ontologie für das Datensatzvokabular dar. Ihre Kernelemente werden anhand bestimmter Metriken ermittelt. Diese können z. B. Anzahl der Tripel/Properties/Typen, Anzahl der Pfade oder die Zentralität von Typen oder Properties sein.

Während sich [Pre+11] primär mit der Datensatzanalyse beschäftigt, berücksichtigt der in dieser Arbeit vorgestellte Ansatz zusätzlich die Visualisierung der extrahierten Zusammenhänge mittels einer angepassten Form von VOWL sowie die praktischen Probleme bei der dynamischen Extraktion mittels SPARQL-Abfragen zur Laufzeit.

3.1.3. Auf Zusammenfassungen basierendes Explorieren der RDF-Verwendung in der LOD Cloud

Khatchadourian und Consens stellen mit „ExpLOD“ eine Java-Anwendung zur Erstellung von Zusammenfassungen der Verwendung von RDF sowie der Verknüpfungen zwischen

verschiedenen Datensätzen in der LOD Cloud vor [KC10]. Sie adressieren damit die Herausforderungen, LD und das zugrundeliegende Schema zu verstehen.

Dabei werden Bisimulation Labels (BLs) basierend auf der Verwendung von RDF erstellt. Bei der RDF-Verwendung wird zwischen Klassen- und Prädikatinstanziierung sowie der Verwendung von Klassen und Prädikaten unterschieden.

Der jeweilige Datensatz wird durch die sich daraus ergebenden Hierarchie in verschiedenen Granularitätsstufen beschrieben.

Durch die Berechnung der größten stabilen Partition (coarsest stable partition CSP) der BLs entsteht die Bisimulation Contraction, welche die Knoten des RDF Graphen auf Partitionen abbildet. Die CSP wird dabei entweder über einen Partitionsverfeinerungsalgorithmus oder durch SPARQL-Abfragen (SPARQL label retrieval query) berechnet.

Die so entstandenen Partitionsblöcke bilden als Graph mit beschrifteten Knoten und unbeschrifteten Kanten die Zusammenfassung für die jeweilige RDF-Verwendung.

3.2. Visualisierung von Linked Data

Im Folgenden werden verschiedene Ansätze zur Visualisierung von Linked Data vorgestellt. Hierbei liegt der Fokus auf Darstellungen für Graphstrukturen mit Hilfe verschiedener Layouts.

3.2.1. RelFinder

Heim et al. [Hei+09] stellen mit dem RelFinder ein Tool zur Extraktion von Beziehungen zwischen zwei oder mehr Objekten in einem LD-Datensatz vor, welches diese Beziehungen in einer graphbasierten Visualisierung darstellt. Nach der Eingabe zweier Begriffe werden diese auf Elemente in der Wissensbasis abgebildet. Diese stellen Start- und Endpunkt für die Suche nach Relationen zwischen diesen dar. Ausgehend von direkten Beziehungen über eine Property sucht die Anwendung sukzessive nach Verbindungen wachsender Länge und fügt diese der Visualisierung dynamisch hinzu. Abbildung 3.1 zeigt den Ergebnisgraphen einer Suche zwischen Alan Turing und Kurt Gödel.

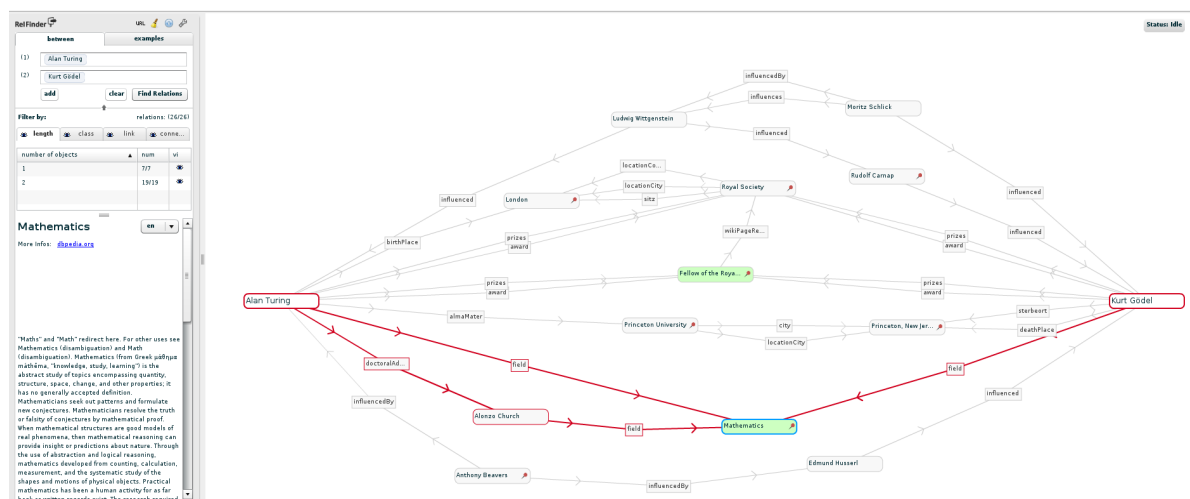


Abbildung 3.1.: Graphische Darstellung des RelFinders nach Abschluss der Suche nach Beziehungen zwischen den Mathematikern Alan Turing und Kurt Gödel auf DBpedia

In der Visualisierung werden Objekte und die Properties durch Knoten und beschriftete, gewichtete Kanten in einem kräftebasierten Layout dargestellt. Um die Lesbarkeit auch bei vielen Kanten zu erhalten, stoßen sich dabei die Beschriftungen der Kanten ab. Damit werden Überlappungen der Kantenbeschriftungen vermieden. Der Benutzer kann außerdem mit der Visualisierung des RelFinders interagieren, indem er Knoten per Drag-and-Drop verschieben und ihre Position fixieren kann (Pick-and-Pin). Außerdem werden weitere Informationen

zum jeweils ausgewählten Knoten in einer Sidebar auf der linken Seite angezeigt und die Pfade einer Verbindung farblich hervorgehoben.

Der RelFinder wurde als Webanwendung mit Adobe Flex implementiert und ist damit in jedem Browser mit Flash-Unterstützung ausführbar.

Im Gegensatz zu der in dieser Arbeit vorgestellten Anwendung LDVOWL dient der RelFinder nicht der Ontologie-Extraktion, sondern der gezielten Suche von Beziehungen zwischen zwei gegebenen Entitäten in einem Datensatz. Dennoch gibt es Ähnlichkeiten bei der abstrakten Funktionsweise der beiden Ansätze: Beide Anwendungen extrahieren mittels SPARQL-Abfragen nach und nach Informationen aus einem Datensatz und fügen diese dynamisch in einen Knoten-Kanten-Graph mit einem kräftebasierten Layout ein.

Während LDVOWL bei der Extraktion abstrakt mit einer Liste der mächtigsten Klassen beginnt, ist der Einstiegspunkt für den RelFinder durch konkrete, vom Nutzer bestimmte Entitäten vorgegeben. Der RelFinder sucht zwischen zwei Entitäten auch nach transitiven Beziehungen über weitere bisher unbekannte Entitäten, LDVOWL extrahiert dagegen nur direkte Beziehungen zwischen den Instanzen zweier Klassen. Dies liegt daran, dass bei einer Suche nach transitiven Beziehungen gefundene Klassen für den Überblick nicht ebenso relevant sein müssen wie die aufgrund ihrer Mächtigkeit bereits geladenen Klassen.

3.2.2. ResXplorer

De Vocht et al. stellen in [DV+13] die Anwendung „ResXplorer“ vor, welche das Durchsuchen von LD-Datensätzen mit Hilfe einer interaktiven Visualisierung ermöglicht. Die Anwendung fokussiert sich dabei auf Datensätze, welche wissenschaftliche Publikationen, Konferenzen und die daran beteiligten Personen behandeln.

Die Ergebnisse von Suchanfragen werden als Knoten verschiedener Form, Farbe und Größe in einem radialen Graphlayout dargestellt. Die Themenverwandtheit von Elementen wird dabei durch Nähe in der Darstellung ausgedrückt. Um dem Benutzer die Exploration von Datensätzen zu erleichtern, schlägt die Anwendung ausgehend von einem Suchergebnis weitere Abfragen vor. Abbildung 3.2 zeigt die Graphdarstellung der Anwendung beim Durchsuchen ausgehend vom Eintrag über Tim Berners-Lee.

Der ResXplorer wurde als Webanwendung in Javascript implementiert und verwendet jQuery UI für die Benutzeroberfläche und das Javascript InfoVis Toolkit¹ für die Darstellung des Graphen. Die Anwendung kann online² ausprobiert werden.

¹<http://philogb.github.io/jit/>

²<http://resexplorer.org/start/>

3. Themenverwandte Arbeiten

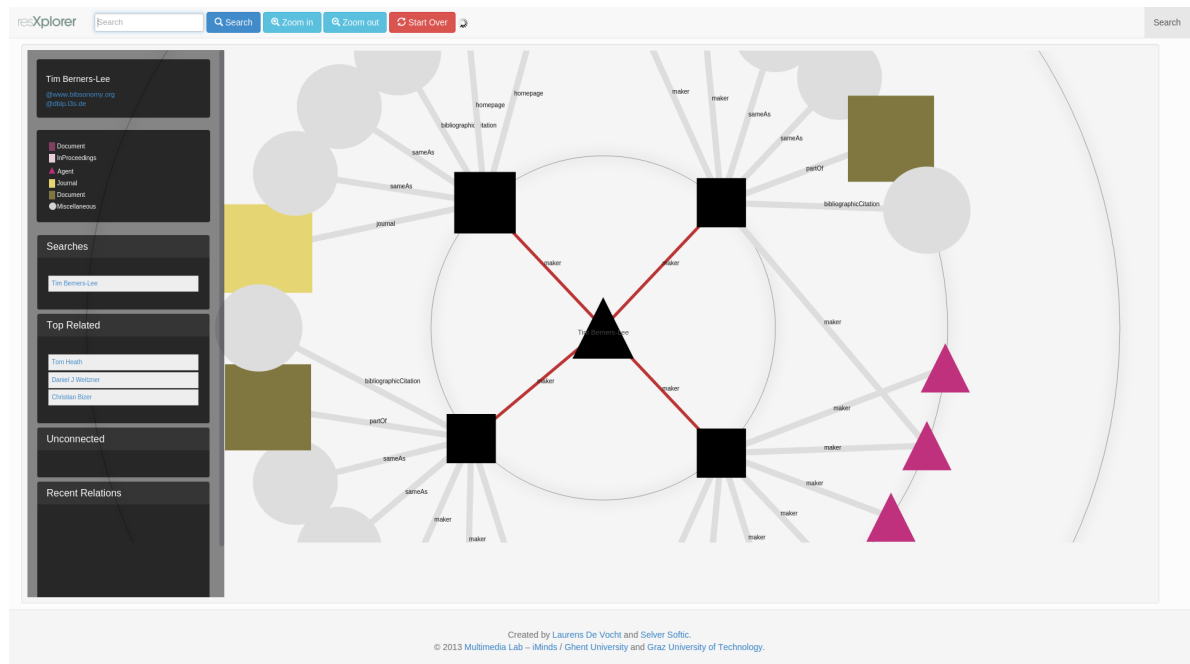


Abbildung 3.2.: Beispiel für die Graphdarstellung der Anwendung „ResXplorer“ um den Eintrag für Tim Berners-Lee

Der Fokus des ResXplorers liegt auf der Exploration von Datensätzen, wohingegen LDVOWL die Ontologie hinter einem Datensatz extrahiert und damit die darin enthaltenen Inhalte zusammenfasst. Außerdem ist der in dieser Arbeit vorgestellte Ansatz auf beliebige SPARQL-Endpunkte anwendbar. Auch technisch unterscheiden sich die beiden Webanwendungen bei der Wahl des eingesetzten Visualisierungstoolkits: Der ResXplorer setzt auf das einfache Javascript InfoVis Toolkit, LDVOWL auf das wesentlich mächtigere D3 [BOH11].

3.2.3. LodLive

Mit der in [CMA12] von Camarda et al. vorgestellten Webanwendung „LodLive“ können die LD-Datensätze mittels einer dynamischen Graphdarstellung durchsucht werden. Nach der Auswahl einer Ressource mit Hilfe einer Suchfunktion oder durch manuelle Eingabe der URI kann der Benutzer über Relationen von einer Ressource zur nächsten springen. Die benötigten Informationen werden dabei dynamisch über JSONP-Abfragen an den jeweiligen Endpunkt bezogen.

Ressourcen werden in LodLive als große Kreise dargestellt, um welchen herum eine Reihe kleiner Kreise angeordnet ist. Diese kleinen Kreise stellen Object Properties dar, welche einzeln aufgeklappt werden können, um damit verknüpfte Ressourcen zu laden und dem Graphen hinzuzufügen. Ist eine Relation geladen, so wird sie als gerichtete, beschriftete Kante

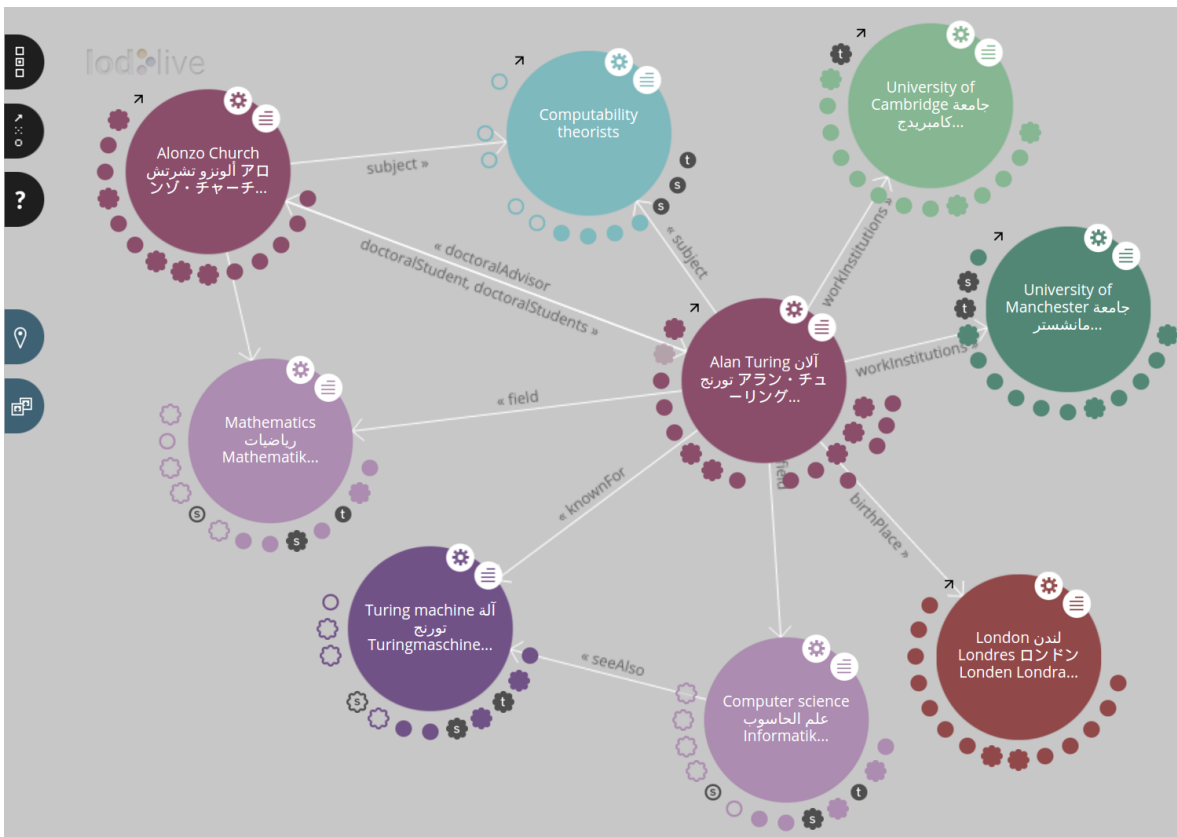


Abbildung 3.3.: Beispiel für die Graphdarstellung der Anwendung „LodLive“ beim Durchsuchen von DBpedia ausgehend von Alan Turing

angezeigt. Datatype Properties zu einer Ressource können bei Bedarf in einem Pop-up-Fenster aufgelistet werden. Abbildung 3.3 zeigt die Darstellung von LodLive beim Durchsuchen von DBpedia nach manueller Umpositionierung der Knoten ausgehend vom Eintrag über Alan Turing.

Weitere Funktionen der Anwendung sind zum einen die Darstellung von verknüpften Bildern in einer Galerie, zum anderen eine Anzeige ortsbezogener Daten auf einer Karte von Google Maps. Beide dieser zusätzlichen Ansichten können bei Bedarf eingeblendet werden.

Die unter der MIT Lizenz stehende Anwendung wurde in Javascript implementiert und kann online³ ausprobiert werden.

Auch wenn diese Art der Informationsbeschaffung der in dieser Arbeit vorgestellten Anwendung ähnelt, bietet LodLive im Gegensatz zu LinkedDataVOWL keinen Überblick über die Inhalte eines Datensatzes, sondern ermöglicht lediglich das schrittweise Durchsuchen der

³<http://en.lodlive.it/>

hinterlegten Daten von einem vom Nutzer gewählten Einstiegspunkt aus. Für den Graphen wird kein bestimmtes Layout verwendet, neu hinzugefügte Knoten werden an einer scheinbar zufälligen Stelle um den Ausgangsknoten herum positioniert, wobei teilweise bereits existierende Knoten überdeckt werden.

3.3. Visualisierung von Ontologien

Da die in dieser Arbeit vorgestellte Anwendung Ontologie-Informationen visuell darstellt, werden in diesem Abschnitt einige bereits existierende Ansätze und Werkzeuge zur Visualisierung von Ontologien präsentiert.

Bei den in Abschnitt 3.3.1, 3.3.2, 3.3.3 und 3.3.5 vorgestellten Anwendungen handelt es sich um Plug-ins für den Ontologie-Editor Protégé⁴. Das in Abschnitt 3.3.7 beschriebene Werkzeug „OntoSphere“ steht sowohl als ein Plug-in für die Entwicklungsumgebung Eclipse als auch für Protégé zur Verfügung. Die in Abschnitt 3.3.4 und 3.3.6 präsentierten Werkzeuge stellen dagegen eigenständige Anwendungen dar.

Bei der in Abschnitt 3.3.8 beschriebenen visuellen Notation für OWL-Ontologien (VOWL) handelt es sich um eine Spezifikation, für welche bereits mehrere Implementierungen existieren, die im darauf folgenden Abschnitt 3.3.9 näher betrachtet werden.

3.3.1. Jambalaya

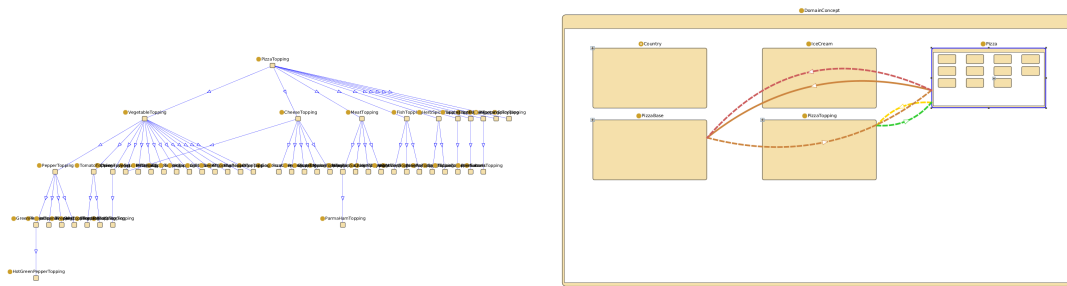
Storey et al. präsentieren in [Sto+01; Sto+02] mit „Jambalaya“ eine Integration des Visualisierungstools „SHriMP“ in den Ontologie-Editor Protégé.

Im Gegensatz zu dem in dieser Arbeit präsentierten Ansatz bietet Jambalaya gleich mehrere verschiedene Darstellungen für Ontologien, von welchen zwei beispielhaft in Abbildung 3.4 dargestellt sind. Klassen und Instanzen werden als verschiedenfarbige Knoten und Properties als Kanten zwischen diesen dargestellt. Klassenhierarchien können entweder als Baum (siehe Abbildung 3.4a) oder durch eine verschachtelte Darstellung (siehe Abbildung 3.4b) dargestellt werden.

Der Benutzer hat die Möglichkeit in der Darstellung zu zoomen und zwischen verschiedenen Ansichten und Layouts zu wechseln, welche Kindknoten verbergen oder anzeigen und außerdem die Bearbeitung von Klassen ermöglichen. Die Navigation in der Visualisierung wird durch Vor- und Zurück-Buttons sowie eine Knotensuche erleichtert.

Das in Java implementierte Werkzeug wird seit 2008 nicht mehr weiter entwickelt [Jam].

⁴<http://protege.stanford.edu/>

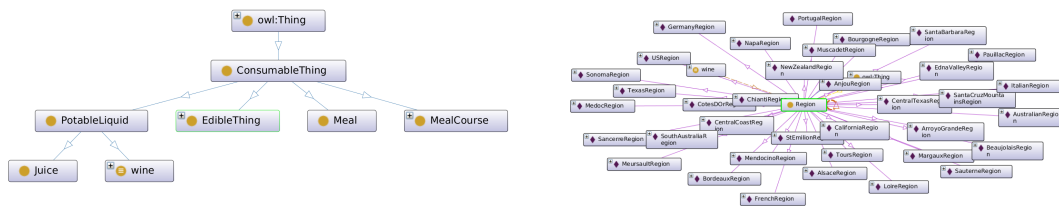


(a) Beispiel für Darstellung einer Klassenhierarchie als Baum (b) Beispiel für Darstellung von Klassen mit einem verschachtelten Layout

Abbildung 3.4.: Zwei Beispiele für verschiedene Darstellungen der Pizza-Ontologie mit Hilfe des Plug-ins „Jambalaya“ für Protégé 3.4

3.3.2. OntoGraf

OntoGraf [Fal] ist ein Plug-in zur Visualisierung von OWL-Ontologien für Protégé welches ab Version 4.1 bereits in der normalen Installation des Ontologie-Editors enthalten ist. Klassen und Instanzen werden mit OntoGraf als abgerundete Rechtecke mit verschiedenen Symbolen dargestellt, Beziehungen werden auf verschiedenfarbige Kanten abgebildet.



(a) Ausschnitt der Klassenhierarchie in einer vertikalen Baumdarstellung (b) Darstellung der hinterlegten Regionen (Instanzen der Klasse „Region“) in einem kräftebasierten Layout

Abbildung 3.5.: Darstellungen von Klassen und Instanzen der Weinontologie in verschiedenen Layouts mit OntoGraf

Abbildung 3.5 zeigt zwei mit OntoGraf erstellte Darstellungen von Teilen der Weinontologie⁵. Für die Anordnung der Knoten stehen bei OntoGraf mehrere verschiedene Layouts zur Wahl, darunter ein rasterförmiges, ein radiales, ein kräftebasiertes (siehe Abbildung 3.5b) sowie in verschiedene Richtungen orientierte Baumdarstellungen (siehe Abbildung 3.5a).

Das Plug-in unterstützt Subklassenbeziehungen und Äquivalenz sowie Object Properties.

⁵<http://www.w3.org/TR/owl-guide/wine.rdf>

3. Themenverwandte Arbeiten

Die Ansicht kann nach verschiedenen Arten von Beziehungen oder Knoten gefiltert werden. Die Darstellung kann vergrößert und verkleinert werden (Zooming) und der Bildausschnitt verschoben werden (Panning). Bewegt der Nutzer seine Maus über einen Knoten, so werden weitere Informationen in einem Tooltip angezeigt. Per Doppelklick können Subklassen einer Elternklasse ein- und ausgeblendet, mittels Pick-and-Pin können Knoten an einer bestimmten Position fixiert werden. Außerdem bietet OntoGraf eine Suchfunktion für Klassen und Instanzen.

3.3.3. OntoViz

Auch OntoViz [Sin] ist ein Plug-in für Protégé, welches Ontologien in einem einfachen Graph darstellt. Das Plug-in verwendet Graphviz⁶ für die grafische Darstellung und visualisiert damit Klassen und Instanzmengen als rechteckige Knoten sowie Properties als gerichtete Kanten.

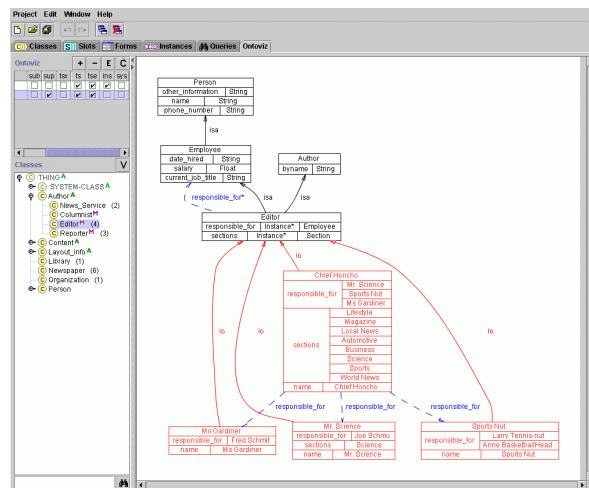


Abbildung 3.6.: Beispiel für die grafische Darstellung einer Ontologie im Protégé Plug-in „OntoViz“ aus [Sin]

Der Nutzer kann Klassen und Instanzen der Ontologie zur grafischen Ansicht hinzufügen, diese filtern sowie geometrisch zoomen. OntoViz steht unter der Mozilla Public License und wird seit Juli 2007 nicht mehr weiterentwickelt.

⁶<http://www.graphviz.org/>

3.3.4. OntoRama

Der quelloffene Ontologie-Browser „OntoRama“ wird in [ERG02] von Eklund et al. vorgestellt und verwendet für die Darstellung eine hyperbolische Baumansicht.

Die in Java implementierte Anwendung stellt die Konzept-Typen einer in RDF ausgedrückten Ontologie dar. Durch mehrfache Darstellung von Knoten mit mehreren Elternknoten wird ein planarer Graph mit radialem Layout erstellt, auf welchen anschließend eine Verzerrung angewendet wird. Abbildung 3.7 zeigt die daraus resultierende Darstellung.

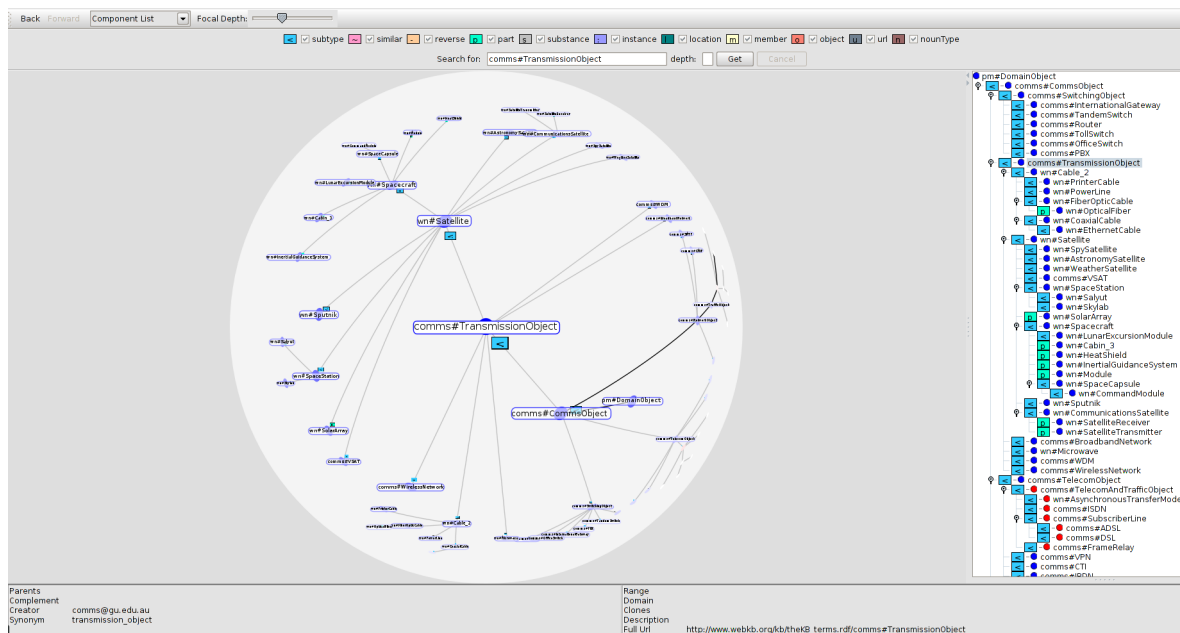


Abbildung 3.7.: Screenshot aus OntoRama bei der Darstellung einer Beispielontologie

Diese hyperbolische Ansicht wird durch eine aus Dateibrowsern bekannte, auf Einrückung basierende Baumansicht in einer Sidebar auf der rechten Seite ergänzt.

Der Benutzer kann den Fokus der hyperbolischen Darstellung per Mausklick auf einen beliebigen Knoten richten. Die Ansicht kann nach bestimmten Arten von Knoten gefiltert werden und innerhalb der Ansicht kann gesucht werden. Beim Bewegen der Maus über verbundene Knoten werden die Kanten zum momentan fokussierten Knoten hervorgehoben. Im Gegensatz zu den zuvor vorgestellten Werkzeugen kann in der Ansicht weder gezoomt, noch die Ansicht verschoben werden.

Die Anwendung steht unter der BSD Lizenz und wird seit 2013 nicht mehr weiter entwickelt [EB].

3.3.5. TGVizTab

Alani stellt mit „TGVizTab“ [Ala03] ein weiteres Plug-in für Protégé vor, welches Ontologien als Netzwerk-Graph in einem kräftebasierten Layout darstellt. TGVizTab liest die Ontologie-Informationen dazu aus der Protégé Java API und stellt die darin vorkommenden Klassen, Instanzen und Properties mit Hilfe der Java-Bibliothek „TouchGraph“ graphisch dar.

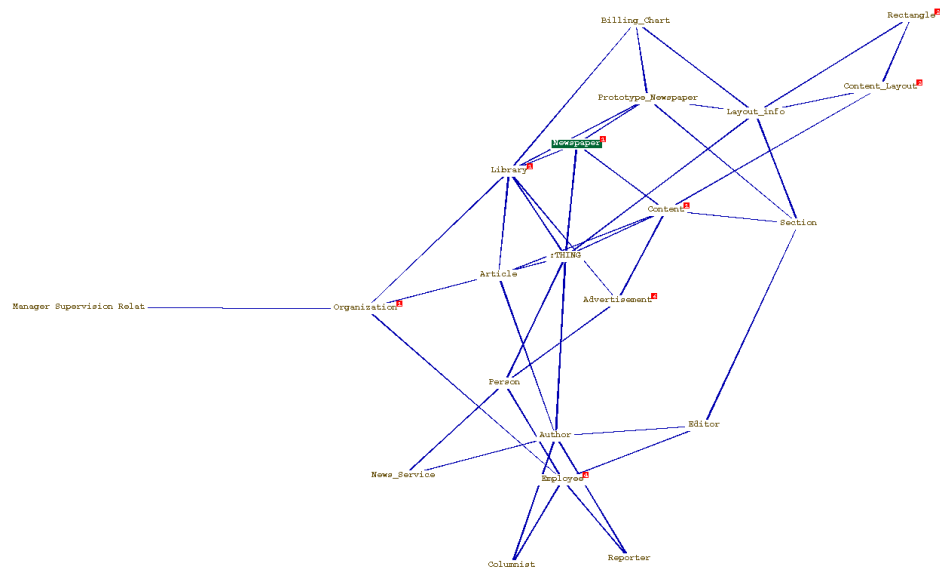


Abbildung 3.8.: Darstellung der Newspaper-Ontologie mit dem TGVizTab Plug-in unter Protégé 3.4

Die grafische Benutzeroberfläche von TGVizTab beinhaltet die Anzeige des Netzwerkgraphen, welche Klassen und Instanzen als Knoten in verschiedenen Farben und Properties als Kanten zwischen diesen darstellt sowie eine Sidebar mit einer Klassen- und Instanzliste. Die dargestellten Knoten sind mit den Klassen/Instanzen in der Sidebar verlinkt.

Der Benutzer kann die Anzeige über die Einstellungen filtern und mit geometrischem Zooming und Panning anpassen. Außerdem ermöglicht TGVizTab das Ein- und Ausklappen von Subgraphen und eine schrittweise Navigation durch den Graphen mittels Doppelklick.

3.3.6. IsaViz

Wie das in Abschnitt 3.3.3 vorgestellte OntoViz setzt auch IsaViz [Piea] auf die Verwendung von Graphviz für die Anzeige des Netzwerkgraphen. In die eigenständige Java Anwendung importierte RDF-Modelle können mit diesem Tool grafisch bearbeitet und anschließend wieder exportiert werden. Dazu bietet die Anwendung neben der Graphanzeige weitere Fenster für Attribute, Definitionen und mögliche Aktionen zur Bearbeitung [Pieb].

Ressourcen und Literale werden als elliptische bzw. rechteckige Knoten dargestellt und Properties als gerichtete Kanten zwischen diesen Knoten.

Der Nutzer kann die Ansicht vergrößern und den angezeigten Ausschnitt verschieben, sowie über Rechtsklicks oder die Verwendung der Pfeiltasten navigieren. Knoten können manuell selektiert werden, des weiteren bietet IsaViz eine Suchfunktion.

Die Anwendung wurde von Pietriga implementiert und ist zusammen mit dem Quellcode online⁷ verfügbar.

3.3.7. OntoSphere

Während die bisher beschriebenen Ansätze Ontologien zweidimensional visualisieren, stellt die in [BBP05] von Bosca et al. vorgestellte Anwendung „OntoSphere“ die Ontologien in einem dreidimensionalen Graphen dar. Ein Beispiel für die resultierende Darstellung findet sich in Abbildung 3.9.

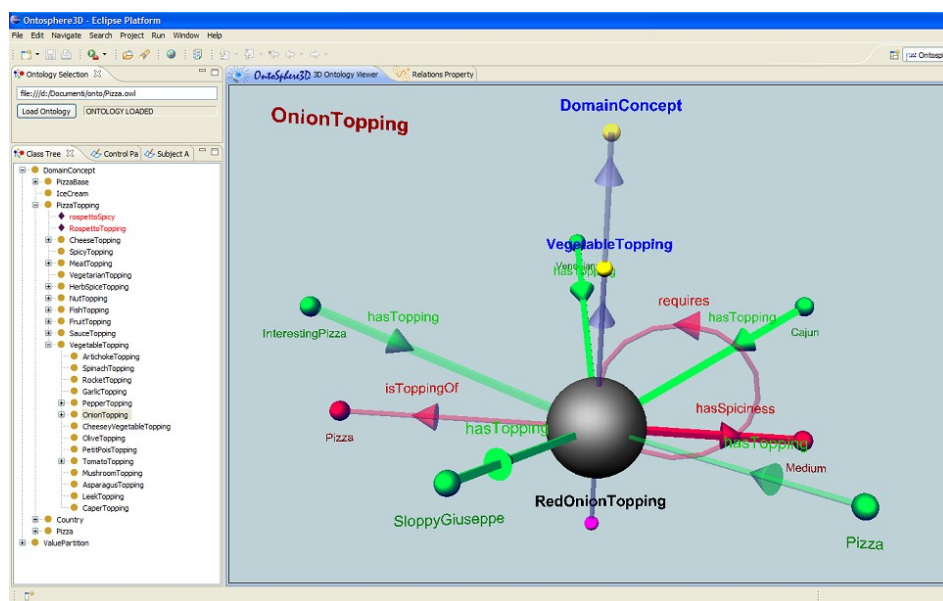


Abbildung 3.9.: Dreidimensionale Darstellung eines Ausschnittes der Pizza-Ontologie in OntoSphere3D (Eclipse) aus [Ont]

Ziel ist dabei eine vollständige Darstellung der in der Ontologie enthaltenen Konzepte in einer angemessenen Detailstufe, welche gut lesbar ist und innerhalb welcher intuitiv navigiert werden kann. Die mit der Java 3D API implementierte Anwendung erlaubt dem Benutzer

⁷<https://www.w3.org/2001/11/IsaViz/>

3. Themenverwandte Arbeiten

auf verschiedene Weise mit der Visualisierung zu interagieren, etwa durch Zooming und Panning, Drehen der Ansicht sowie Auswahl von Knoten. Die Anwendung erkennt den Fokus des Nutzers über die selektierten Knoten und bietet drei verschiedene Szenen: Die „Root Focus Scene“ zeigt semantische, jedoch keine taxonomische Informationen an. Für die Darstellung von Hierarchie und semantischen Beziehungen wird die „Tree Focus Scene“ verwendet. Eine oberflächliche Sicht auf ein Konzept und dessen Eltern- bzw. Kindknoten bietet die „Concept Focus Scene“.

Unterstützt werden Taxonomien und Ontologien in verschiedenen Formaten wie RDF(S), DAML oder OWL.

3.3.8. Visual Notation for OWL Ontologies (VOWL 2)

Im Gegensatz zu den zuvor beschriebenen Werkzeugen und Plug-ins handelt es sich bei VOWL [Loh+14a] zunächst lediglich um die Spezifikation einer visuellen Notation für OWL-Ontologien. Implementierungen dieser Notation werden in den folgenden Abschnitten näher betrachtet. VOWL richtet sich primär an Nutzer, welche sich einen Überblick über eine Ontologie verschaffen möchten, jedoch keine Experten auf dem Gebiet der Ontologien sind.

In VOWL werden OWL Elemente wie Klassen und Properties auf grafische Elemente abgebildet und diese zu einem Knoten-Kanten-Graph verbunden. Die wichtigsten Bestandteile von VOWL sind die grafischen Primitive, ein Farbschema, das kräftebasierte Layout für die Darstellung des Graphen sowie die Regeln für die mehrfache Darstellung bestimmter Elemente für eine klarere Darstellung.

In VOWL werden Klassen als Kreise, Datentypen sowie Bezeichnungen für Properties als Rechtecke und Properties selbst als gerichtete Kanten dargestellt. Die Farbe der Kreise bzw. Rechtecke gibt außerdem an, ob das jeweilige Element deprecated ist bzw. zu einem externen oder dem RDF-Namensraum gehört.

3.3.9. Implementierungen von VOWL

Mit ProtégéVOWL und WebVOWL unterstützen bereits zwei Tools die oben beschriebene Notation für die Darstellung von Ontologien. Der Quellcode der beiden Implementierungen ist jeweils unter der MIT-Lizenz auf GitHub verfügbar.

ProtégéVOWL

Bei dem in [LNB14] vorgestellten ProtégéVOWL handelt es sich um die erste Implementierung von VOWL 2.0. ProtégéVOWL ist ein in Java geschriebenes Plug-in für den Ontologie-Editor Protégé und nutzt das Visualisierungstoolkit „Prefuse“ für die Graphdarstellung.

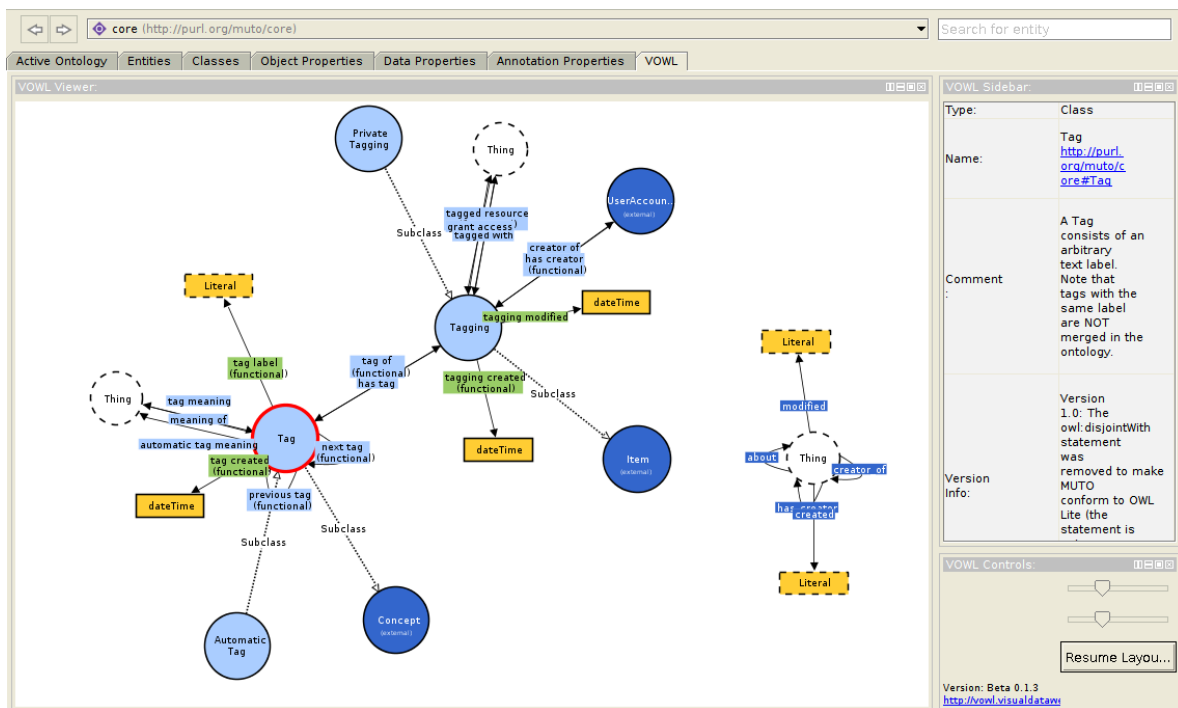


Abbildung 3.10.: Darstellung der Ontologie „Modular Unified Tagging Ontology“ (MUTO) in ProtégéVOWL

Die in Abbildung 3.10 abgebildete Anzeige von ProtégéVOWL im VOWL-Tab von Protégé ist dreigeteilt: Neben der Anzeige des Graphen auf der linken Seite gibt es eine Sidebar für weitere Informationen zum aktuell ausgewählten Element (rechts oben) und einen Bereich für die Anpassung und Steuerung des kräftebasierten Graph-Layouts (rechts unten).

WebVOWL

Im Gegensatz zu ProtégéVOWL handelt es sich bei WebVOWL [Loh+14b] um eine Standalone-Anwendung, welche Ontologien nach VOWL 2 darstellt und die grafischen Elemente unter der Verwendung von offenen Webstandards wie CSS und SVG im Webbrowser anzeigt. Dazu stützt sich WebVOWL für die Berechnung des kräftebasierten Graph-Layouts auf das Javascript Toolkit D3 [BOH11].

3. Themenverwandte Arbeiten

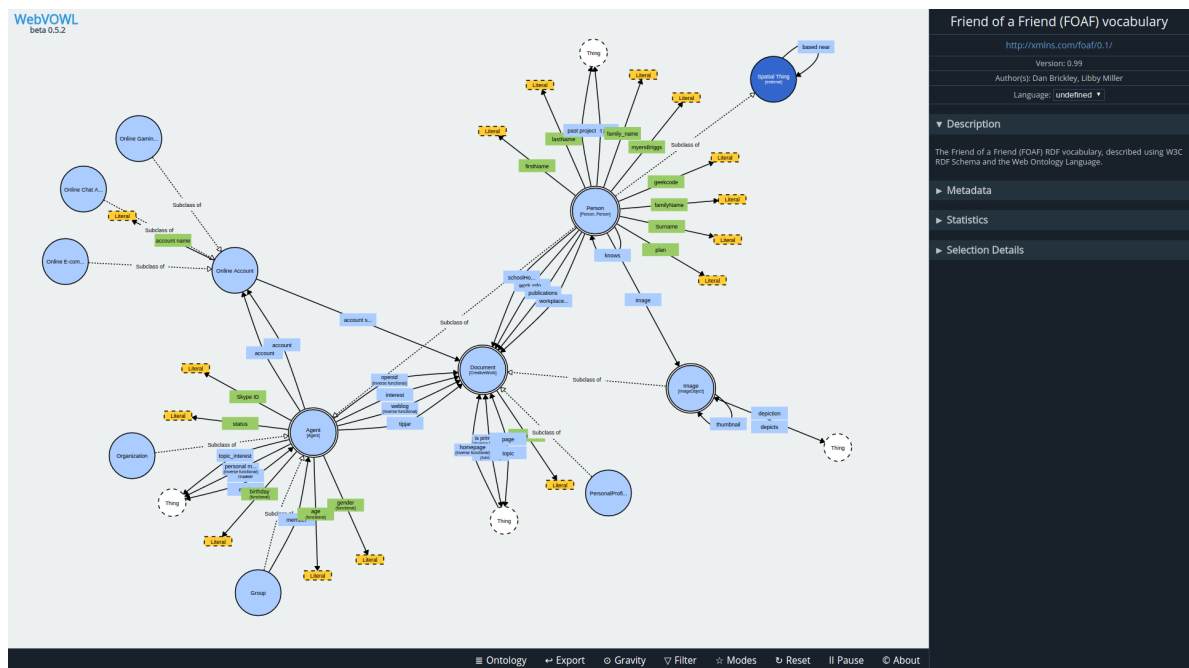


Abbildung 3.11.: Darstellung der Ontologie „Friend of a Friend“ (FOAF) in WebVOWL

Die Informationen über die darzustellenden Ontologien werden von WebVOWL zur Laufzeit aus einer JSON-Datei gelesen. Die anzuzeigenden Ontologien müssen deshalb zuvor mit einem separatem Tool in diese Form gebracht werden. Bei der Darstellung werden im Moment hauptsächlich TBox-Informationen wie Klassen, Datentypen und Properties berücksichtigt. Abbildung 3.11 zeigt die Benutzeroberfläche von WebVOWL für die Darstellung der Ontologie „Friend of a Friend“⁸.

Im Gegensatz zu dem in dieser Arbeit vorgestellten Ansatz dient WebVOWL lediglich der Anzeige einer bereits vorliegenden Ontologie, nicht aber der Extraktion einer Ontologie aus einem beliebigen LD-Datensatz.

Der Nutzer von WebVOWL hat die Möglichkeit in der Visualisierung zu zoomen sowie den angezeigten Bildausschnitt zu verschieben. Außerdem kann er Knoten beliebig positionieren und dort fixieren. Auch auf das Kräfte-Lay-out kann durch Pausieren oder Anpassung der Anziehungskräfte Einfluss genommen werden.

⁸<http://xmlns.com/foaf/spec/>

3.3.10. QueryVOWL

Bei QueryVOWL [Haa+15] handelt es sich um keine Visualisierung für Ontologien, sondern um eine auf VOWL basierende, visuelle Abfragesprache für Linked Data, welche der Vollständigkeit halber hier kurz erwähnt wird. QueryVOWL definiert eine Abbildung von VOWL auf SPARQL-Abfragen und ermöglicht es damit Benutzern ohne Kenntnisse von Technologien wie RDF und SPARQL, Abfragen für Linked Data grafisch zu erstellen.

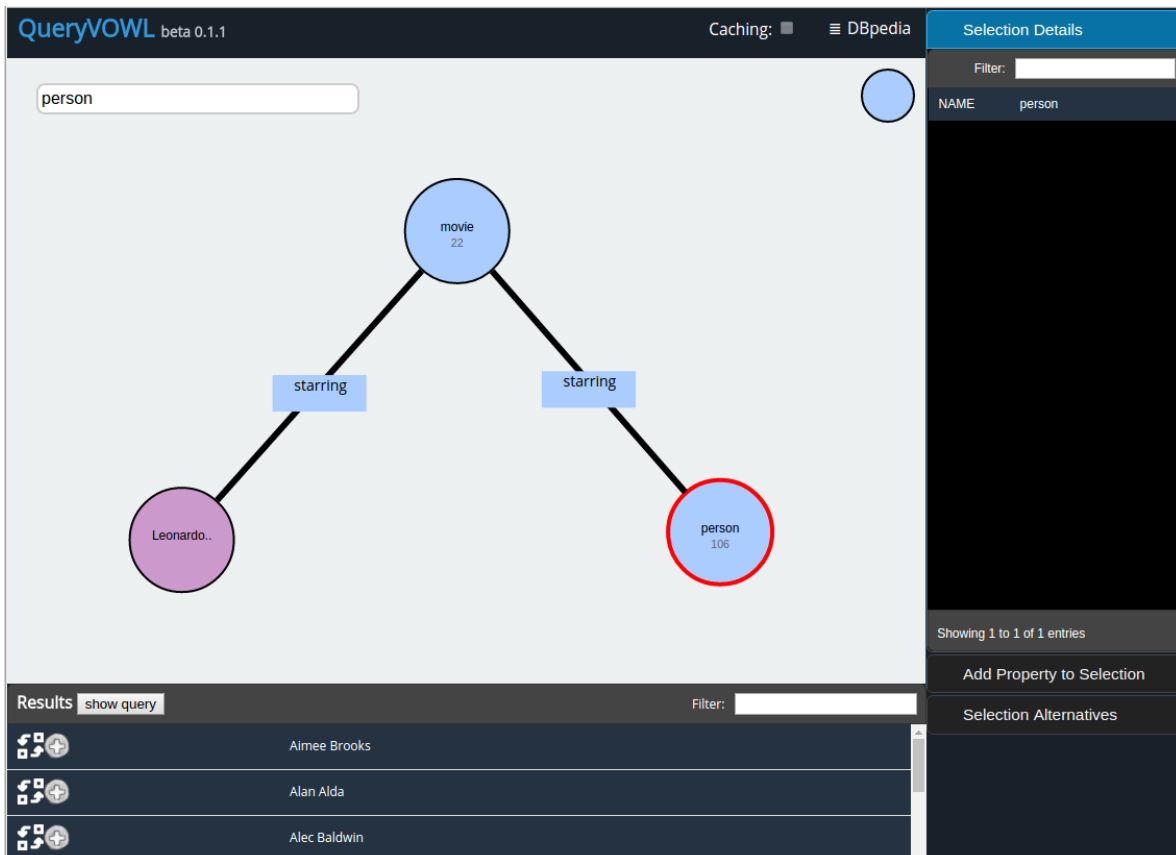


Abbildung 3.12.: Beispiel für die visuelle Darstellung einer SPARQL-Abfrage in QueryVOWL

Hierbei wird ein Teilgraph des gesuchten Datensatzes vom Benutzer visuell konstruiert, wobei Platzhalter für Variablen in SPARQL verwendet werden können. Die Ergebnisse der Abfrage haben keine feste Struktur wie in SPARQL, sondern der Nutzer kann die einzelnen Bestandteile der Treffer untersuchen. Die Abfrage kann interaktiv geändert werden, wobei der Nutzer die Auswirkungen auf die Anzahl der Treffer direkt angezeigt bekommt und von der Anwendung durch Vorschläge unterstützt wird.

3. Themenverwandte Arbeiten

Abbildung 3.12 zeigt beispielhaft die visuelle Darstellung einer SPARQL-Abfrage nach den Personen, welche zusammen mit dem Schauspieler Leonardo Di Caprio in einem Film aufgetreten sind.

Für QueryVOWL existieren zwei Implementierungen: Eine in C# geschriebene Desktop Anwendung sowie eine webbasierte Anwendung. Letztere wurde in einer qualitativen Nutzerstudie evaluiert, welche zeigte, dass die Benutzer die allgemeine Struktur der Abfrage schnell verstanden.

4. Visuelle Notation

Für die in dieser Arbeit vorgestellte Anwendung „LinkedDataVOWL“ (kurz LDVOWL) wird die VOWL-Spezifikation in der Version 2.0 angepasst und so erweitert, dass diese den Fokus auf die ABox-Informationen von Datensätzen legt.

Im Folgenden wird auf die einzelnen Bestandteile dieser visuellen Notation näher eingegangen. Der Aufbau orientiert sich dabei an der Spezifikation von VOWL 2.0. Eine tabellarische Übersicht über die einzelnen grafischen Primitive findet sich in Tabelle 4.1.

Name	Kurzbeschreibung	Darstellung von
Kreis	Kreise werden als Knoten im kräftebasierten Layout verwendet und stellen Klassen dar.	<ul style="list-style-type: none"> • Klassen
Linie	Eine Linie stellt eine Relation zwischen den Instanzen zweier Klassen dar.	<ul style="list-style-type: none"> • Relationen als Menge verschiedener Properties
Pfeilspitze	Die Pfeilspitze einer Linie zeigt die Richtung der Relation.	<ul style="list-style-type: none"> • Richtung von Relationen
Rechteck	Rechtecke werden zur Beschriftung von Relationen und für Datentypen verwendet. Wie Kreise sind auch Rechtecke Knoten im kräftebasierten Layout. Sie sind einfarbig und besitzen nur im Falle von Datentypen oder im selektierten Zustand einen Rand.	<ul style="list-style-type: none"> • Beschriftungen • Datentypen
Linienstil	Durchgezogene Linien stellen Relationen dar, gestrichelt bzw. gepunktete Linien werden für Mengenoperationen verwendet.	<ul style="list-style-type: none"> • Arten von Beziehungen
Text	Text wird zur Beschriftung von Klassen, Datentypen und Relationen verwendet.	<ul style="list-style-type: none"> • Beschriftungen

Tabelle 4.1.: Übersicht über die in LinkedDataVOWL verwendeten graphischen Primitive und deren Bedeutung

4.1. Klassen

Eine Klasse $C := \{i \mid i \text{ ist Instanz von } C\}$ ist eine Menge von Instanzen. Klassen stehen im Mittelpunkt der graphischen Darstellung und werden wie auch in VOWL als kreisförmige Knoten in einem kräftebasierten Layout dargestellt.

Die Größe eines Kreises steht für die Anzahl der Instanzen der jeweiligen Klasse. Um die Lesbarkeit zu verbessern, haben die Kreise einen bestimmten minimalen Radius r_{min} und werden über diesen hinaus logarithmisch skaliert. Dies stellt sicher, dass Klassen auch mit stark variierender Anzahl von Instanzen übersichtlich dargestellt werden können.

Die Spezialfälle für `owl:Thing` und `rdfs:Resource` werden bei LDVOWL nicht betrachtet, da diese als TBox-Informationen im Normalfall nicht dargestellt werden (siehe Abschnitt 5.4.7).

4.2. Relationen

Wie in VOWL spezifiziert, werden Properties zwischen Klassen als Linien zwischen den jeweiligen Knoten dargestellt. Die Darstellung einer Property P als Kante zwischen der Klasse C_1 und der Klasse C_2 bedeutet, dass mindestens eine Instanz $i_1 \in C_1$ und mindestens eine Instanz $i_2 \in C_2$ existieren, welche über die dargestellte Property P miteinander verbunden sind.

Da potentiell sehr viele verschiedene Properties zwischen den Instanzen zweier Klassen existieren können, werden diese in der angepassten Notation nicht als einzelne Linien dargestellt, sondern zu einer Linie zusammengefasst.

Wie in VOWL spezifiziert, werden für allgemeine Relationen durchgezogene Linien verwendet, für Subklassenbeziehungen wird eine gepunktete Linie verwendet. Die Disjunktheit zweier Klassen wird durch eine gestrichelte Linie zu einem speziellen Knoten dargestellt.

4.3. Pfeilspitzen von Relationen

Wie auch in VOWL stellen die Pfeilspitzen der Linien zwischen den Klassen die Richtung der Relation zwischen den Instanzen der jeweiligen Klassen dar. Zeigt eine Linie beispielsweise von Klasse C_1 auf Klasse C_2 , so sind Instanzen der Klasse C_1 über die beteiligten Properties P mit Instanzen der Klasse C_2 verbunden.

Relationen über Properties werden mit einer ausgefüllten Pfeilspitze dargestellt. Die Richtung einer Teilmengen-/Subklassenbeziehung zwischen einer Kindklasse und dessen Vaterklasse wird mit einer nicht ausgefüllten Pfeilspitze dargestellt.

4.4. Rechtecke

Wie auch in VOWL werden Rechtecke sowohl zur Darstellung von Property-Beschriftungen als auch für Datentypen verwendet. Die Rechtecke verwenden dafür die selben Farben wie sie in VOWL empfohlen werden. Darstellungen von Property-Beschriftungen verfügen im deselektierten Zustand über keinen Rand.

4.5. Text

Text wird zur Beschriftung von Klassen und Properties verwendet. Wie in VOWL 2.0 empfohlen, wird für diese Beschriftungen, falls vorhanden, der in `rdfs:label` angegebene Text gewählt. Ist dieser für die jeweilige Klasse oder Property nicht vorhanden, so wird auf den `skos:prefLabel` oder den letzten Teil der URI zurückgegriffen. Lange Beschriftungen werden auf eine fixe Maximallänge gekürzt, sodass die Darstellungen nicht zu groß werden.

Sind mehrere verschiedene Properties an einer Relation beteiligt, so wird die Beschriftung der am häufigsten vorkommenden Property gewählt. Außerdem wird die Anzahl verschiedener Properties als Zahl in eckigen Klammern an die Bezeichnung angehängt.

4.6. Farbpalette

Im Folgenden werden die verschiedenen Farben und deren Bedeutung näher erklärt. Eine Übersicht findet sich in Tabelle 4.2, im Falle mehrerer möglicher Farben ist das jeweilige Feld beispielhaft mit einer dieser hinterlegt.

Name	Farbe	Beschreibung	wird angewendet auf
Hintergrund	helles grau	Der Hintergrund der Graphdarstellung ist mit einem hellen grau gefüllt. Dadurch hebt sich der Bereich für die Darstellung des Graphen von der Sidebar ab ohne dabei die Aufmerksamkeit des Betrachters auf sich zu ziehen.	<ul style="list-style-type: none"> • Hintergrund der Graph-Darstellung
Vordergrund	schwarz	Diese Farbe wird für Beschriftungen auf hellen Farben sowie für die Ränder von Knoten verwendet.	<ul style="list-style-type: none"> • Knotenränder • Knotenbeschriftungen

4. Visuelle Notation

Name	Farbe	Beschreibung	wird angewendet auf
Linienfarbe	verschiedene dunkle Grautöne	Je nach Breite der Kante werden diese wie auch ihre zugehörige Pfeilspitze in verschieden dunklen Grautönen dargestellt. Beginnend mit schwarz werden für die Kanten mit zunehmender Breite immer hellere Grautöne verwendet. Allerdings ist durch die maximale Kantenbreite auch die Linienfarbe beschränkt, sodass die Kanten immer gut erkennbar sind.	<ul style="list-style-type: none"> • Kanten • Pfeilspitzen
Intern	Hellblau	Diese Farbe wird auf den Hintergrund der als intern klassifizierten Knoten angewendet.	<ul style="list-style-type: none"> • Hintergrund von Kreisen • Hintergrund von Rechtecken
Extern	verschiedene Farbtöne von blau bis violett	Verschiedene Farben zwischen Dunkelblau und violett werden als Hintergrund für Elemente aus einem als extern klassifizierten Namensraum verwendet. Außerdem wird die Farbe der Beschriftung von schwarz auf weiß invertiert. Diese Anpassungen dienen dazu, dem Benutzer eine schnelle Unterscheidung zwischen externen und internen Elementen zu ermöglichen, ohne dafür die Präfixe der URIs manuell vergleichen zu müssen.	<ul style="list-style-type: none"> • Hintergrund von Kreisen • Hintergrund von Rechtecken
Datentyp	Gelb	Diese Farbe wird auf Rechtecke angewendet, welche Datentypen darstellen.	<ul style="list-style-type: none"> • Hintergrund von Rechtecken

Name	Farbe	Beschreibung	wird angewendet auf
Datentyp Eigenschaft	Grün	Diese Farbe wird auf Datentyp-Properties (Properties zwischen Instanzen einer Klasse und einem Datentyp) angewendet und dient einer schnellen Abgrenzung zu Properties zwischen den Instanzen zweier Klassen.	<ul style="list-style-type: none"> • Hintergrund von Rechtecken
Hervorhebung	Rot	Für ausgewählte Elemente wie auch Elemente unter dem Cursor (Hover) wird diese Farbe verwendet. Die Farbe wird bei Auswahl eines Elements auf dessen Rand angewendet, beim Hovern wird die Farbe auf den gesamten Hintergrund des Elements angewendet.	<ul style="list-style-type: none"> • Ränder von selektierten Kreisen und Rechtecken • selektierte Linien und deren Pfeilspitzen • Hintergrund von Kreisen • Hintergrund von Rechtecken
indirekte Hervorhebung	Orange	Die Farbe wird analog zur Hervorhebungsfarbe bei Auswahlen angewendet, betrifft dabei jedoch nicht das selektierte Element selbst, sondern weitere Vorkommen des selben Elements im Graph.	<ul style="list-style-type: none"> • Ränder von Rechtecken • Ränder von Kreisen

Tabelle 4.2.: Übersicht über die für LDVOWL verwendeten Farben und deren Bedeutung

Für den Hintergrund von Klassen und Properties im internen Namensraum wird hellblau verwendet.

Anders als in VOWL 2.0 wird in der für Linked Data angepassten Notation violett nicht für RDF(S)-Elemente verwendet, da `rdfs:Class`, `rdfs:Resource` und `rdf:Property` zu den TBox-Informationen des jeweiligen Datensatzes gehören, welche in LDVOWL nicht berücksichtigt werden. Verschiedene Farbtöne von blau bis violett werden für die Darstellung von Elementen aus als extern klassifizierten Namensräumen verwendet.

Außerdem gibt es kein hellgrau für deprecated Class/Property, da es nicht möglich ist, allein aus ABox-Informationen festzustellen, ob eine Klasse oder Property veraltet/deprecated ist.

Für Linien werden statt schwarz Grautöne unterschiedlicher Helligkeit verwendet, um zusammen mit der Linienstärke die Anzahl verschiedener Properties zwischen den Instanzen zweier Klassen darzustellen.

Die Farben für Hervorhebung und indirekte Hervorhebung entsprechen den von VOWL empfohlenen (rot und orange).

Bei der Hervorhebung von Elementen wird zwischen Selektieren und Hovern unterschieden: Beim Selektieren wird das vom Benutzer per Mausklick oder Berührung des Touchscreens gewählte Element mit der Hervorhebungsfarbe umrahmt. Bewegt der Benutzer lediglich die Maus über ein Element (Hover), so ändert sich der Hintergrund zur Hervorhebungsfarbe.

Die indirekte Hervorhebung kommt bei mehrfacher Darstellung einer Property oder eines Datentyps zum Einsatz, wie im Folgenden näher beschrieben wird.

4.7. Mehrfache Darstellung von Properties und Datentypen

Während Klassen im Graphen maximal einmal dargestellt werden, kommen Properties und Datentypen in der Darstellung unter Umständen mehrfach vor. Dies dient einer besseren Lesbarkeit des Graphen, denn ohne eine mehrfache Darstellung würden beispielsweise Datentypen in den Fokus gerückt werden. Da Datentypen wie z. B. Strings oft mit vielen verschiedenen Klassen verbunden sind, würden sich die Darstellungen der Klassen um die Datentypen herum anordnen. Dies würde dem Benutzer der Visualisierung den Eindruck vermitteln, die Datentypen wären besonders wichtig für die Ontologie. Um dies zu vermeiden und eine baumartige Struktur zu begünstigen, werden sowohl Properties als auch Datentypen mehrfach dargestellt.

Wird eine Property oder ein Datentyp vom Benutzer selektiert, so wird das ausgewählte Element mit der Farbe der direkten Hervorhebung (rot) umrandet. Darüber hinaus werden alle weiteren Vorkommen der selben Property/des selben Datentyps mit der Farbe für indirekte Hervorhebung (orange) umrandet, um den Benutzer auf diese weiteren Vorkommen hinzuweisen.

Sobald zwei oder mehr Klassen über ihre Instanzen als äquivalent identifiziert wurden (siehe Abschnitt 5.4.4), werden sie in der Graphdarstellung als ein einzelner Knoten mit einem doppelten Rand dargestellt.

4.8. Nicht enthaltene Elemente aus VOWL

Die folgenden Elemente aus VOWL wurden aus verschiedenen Gründen in LDVOWL nicht implementiert und sind deshalb keine Bestandteile der hier beschriebenen Notation:

Da ein Datensatz die aufgrund einer verwendeten Ontologie maximal bzw. minimal zugelassenen Kardinalitäten nicht unbedingt ausschöpfen muss, könnten für die Kardinalitäten einer Property lediglich Minimal- bzw. Maximalwerte für den jeweiligen Datensatz bestimmt werden. Dieser geringen Aussagekraft steht jedoch eine Extraktion durch aufwändige SPARQL-Abfragen gegenüber. Diese Abfragen würden auf dem Endpunkt entweder gar nicht ausgeführt oder würden für ihre Beantwortung zumindest sehr viel Zeit benötigen. Aus diesem Grunde scheint eine Überprüfung dieser Eigenschaften wenig nützlich. Selbiges trifft auch für Super- und Subproperties sowie äquivalente Properties zu.

Die Erkennung von inversen Properties wäre zwar wünschenswert, da durch das Zusammenfassen von Properties mit ihrer jeweiligen Umkehrung unter bestimmten Umständen auch die graphische Repräsentation übersichtlicher werden würde. Aber auch hier setzt die Extraktion aufwändige Abfragen voraus und wurde deshalb nicht umgesetzt.

Die Transitivität von Properties sind hauptsächlich für Reasoner interessant und in der TBox hinterlegt, eine zuverlässige Erkennung aus ABox Informationen ist deshalb nicht möglich.

5. Entwickeltes System

In diesem Kapitel wird mit „LinkedDataVOWL“ ein im Rahmen dieser Arbeit entwickeltes System vorgestellt, welches die zuvor beschriebene Notation verwendet, um extrahierte Ontologie-Informationen aus beliebigen SPARQL-Endpunkten darzustellen. Hierbei wird auf Software-Architektur und Funktionsweise der Anwendung eingegangen sowie die möglichen Interaktionen mit diesem vorgestellt.

Zuvor werden jedoch Voraussetzungen an den Endpunkt sowie die Anforderungen an die Anwendung genauer spezifiziert.

5.1. Voraussetzungen

Zunächst werden einige Voraussetzungen definiert, welche erfüllt sein müssen, damit das beschriebene System funktioniert. Diese Voraussetzungen gliedern sich in zwei Teile, server-seitige und client-seitige Voraussetzungen. Die server-seitigen Voraussetzungen richten sich an den zu untersuchenden SPARQL-Endpunkt und dessen Umgang mit den dort eingehenden Abfragen. Die client-seitigen Voraussetzungen dagegen richten sich an den verwendeten Browser und dessen Unterstützung von bestimmten Webstandards. Zuerst werden die Voraussetzungen an den Endpunkt näher beschrieben.

5.1.1. SPARQL-Endpunkt

Um das implementierte System auf einem bestimmten SPARQL-Endpunkt verwenden zu können, muss dieser die folgenden Voraussetzungen erfüllen:

Zunächst sollte der SPARQL-Endpunkt auf mittels HTTP-GET gestellte SPARQL-Abfragen im JSON-Format antworten, Antworten in anderen Formaten wie beispielsweise XML werden momentan nicht unterstützt. Der Endpunkt muss des weiteren Cross-Origin Resource Sharing (CORS) durch das Senden eines entsprechenden HTTP-Headers erlauben. Ist dies nicht der Fall, so müssen die SPARQL-Abfragen über einen lokalen HTTP-Proxy geleitet werden, da der Browser die Antworten des Endpunkts sonst wegen eines Verstoßes gegen die Same-Origin-Policy verwirft. Der Endpunkt sollte außerdem alle in den Abfragen verwendeten SPARQL-Sprachkonstrukte unterstützen (darunter auch BIND und SAMPLE), ansonsten sind

die Ergebnisse unvollständig und es werden beispielsweise keine Verbindungen zwischen Klassen und Datentypen gefunden. Des weiteren wird angenommen, dass der angefragte Tripelstore, zumindest Tripel enthält, welche Instanzen über `rdf:type` mit deren Klassen verknüpft. Eine Beschreibung von Klassen durch die Aufzählung ihrer Instanzen mittels `owl:oneOf` wird nicht unterstützt.

5.1.2. Browser

Die Anwendung wurde unter Chromium 48 und Firefox 43 getestet, sollte darüber hinaus auch in jedem anderen modernen Browser, welcher Javascript ausführt sowie aktuelle Webstandards wie HTML5, CSS3 und SVG unterstützt, lauffähig sein.

5.2. Anforderungen an das System

Im Folgenden wird auf die Anforderungen an das implementierte System eingegangen. Diese werden in funktionale und nicht-funktionale Anforderungen unterteilt.

5.2.1. Funktionale Anforderungen

Ziel der Anwendung soll es sein, dem Nutzer einen groben Überblick über die Inhalte und Struktur des von ihm gewählten Datensatzes zu geben. Dazu gehören die wichtigsten Klassen sowie Subklassenbeziehungen zwischen diesen. Die Anwendung soll über die Instanzen feststellen können, ob zwei Klassen äquivalent oder disjunkt sind. Außerdem sollen Properties zwischen den Instanzen von Klassen erkannt werden. Darüber hinaus soll die Anwendung Beispiele für die zu den Instanzen der Klassen hinterlegten Informationen anzeigen und die Zugehörigkeit von Klassen und Properties zu den relevantesten Namensräumen im Datensatz darstellen.

5.2.2. Nicht-funktionale Anforderungen

Für die Extraktion der Ontologie-Informationen werden SPARQL-Abfragen an den entsprechenden SPARQL-Endpunkt gesendet. Da öffentliche Endpunkte wie z. B. von DBpedia¹ Abfragen mit einer erwarteten langen Laufzeit (beispielsweise über zwei Minuten) aus Gründen der Fairness und zum Schutz vor DDoS-Angriffen nicht ausführen und direkt mit einem Fehler (HTTP 500) beantworten, müssen die von der Anwendung gesendeten Abfragen schnell

¹<http://dbpedia.org/sparql>

zu beantworten sein. Folglich sind mehrere SPARQL-Abfragen mit geringem Rechenaufwand einer einzelnen Abfrage mit hohem Aufwand vorzuziehen. Dennoch soll darauf geachtet werden, die Anzahl der Abfragen zu begrenzen, einerseits da die Anzahl gleichzeitiger Verbindungen zum selben Host durch den Browser beschränkt ist, andererseits um Bandbreite zu sparen.

Die Ergebnisse der Abfragen sollen dem Nutzer möglichst schnell präsentiert werden, indem sie der Visualisierung hinzugefügt werden. Besonders wichtig ist, dass der Nutzer nach der Auswahl des zu untersuchenden Endpunkts innerhalb weniger Sekunden erste Ergebnisse zu sehen bekommt, damit klar ist, dass die Extraktion am gewählten SPARQL-Endpunkt funktioniert.

Im Folgenden wird auf die Architektur der Anwendung eingegangen, welche die beschriebenen Anforderungen erfüllen soll.

5.3. Software-Architektur

In diesem Abschnitt wird die Architektur der Anwendung näher beschrieben. Hierbei wird auf die verschiedenen Komponenten und Konnektoren eingegangen. Zunächst werden die verwendeten Technologien vorgestellt.

5.3.1. Verwendete Technologien

Das System wurde als Webanwendung in Javascript implementiert. Für die Umsetzung des Model-View-Controller Musters sowie einer Zwei-Wege-Datenbindung und Dependency Injection wurde das Framework AngularJS² in der Version 1.4 eingesetzt, für die Visualisierung der extrahierten Daten kommt das Visualisierungstoolkit D3 [BOH11] zum Einsatz.

Die Elemente der Benutzeroberfläche wurden mit Bootstrap und AngularUI erstellt, für die Slider wurde jQuery verwendet.

Da bei der Implementierung auch neue Sprachelemente aus ECMAScript 6 eingesetzt wurden, wird der Quellcode zunächst mit „Babel“ vorverarbeitet, damit die Anwendung in allen aktuellen Browsern lauffähig ist.

²<https://angularjs.org/>

5.3.2. Komponenten

Die Anwendung besteht aus Komponenten der Benutzeroberfläche sowie verschiedenen Services zur Datenextraktion und -haltung. Es handelt sich um eine reine Front-End Anwendung, die SPARQL-Endpunkte dienen dabei als Back-End.

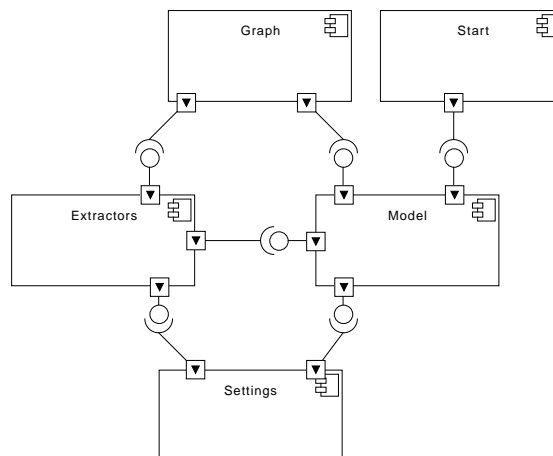


Abbildung 5.1.: UML-Komponentendiagramm der Anwendung LDVOWL

Komponenten der Benutzeroberfläche

Die wichtigsten Komponenten der Benutzeroberfläche sind die Graphdarstellung sowie die Sidebar. Letztere ist in verschiedene Gruppen aufgeteilt, welche die aktuelle Auswahl, Namensräume, Filter und Grapheinstellungen sowie Informationen über den Endpunkt anzeigen. Die einzelnen Gruppen und Darstellungen sind als Angular-Direktiven implementiert. Abbildung 5.2 zeigt die Struktur des Sidebar-Pakets.

Die darzustellenden Informationen erhalten die Komponenten von den Services der Datenhaltung.

Services

Die Services zur Extraktion von Klassen, Relationen, Datentypen und Details senden zunächst von der QueryFactory generierte SPARQL-Abfragen an den gewünschten Endpunkt. Nach Erhalt der Antwort speichern diese Extraktoren die aus den Antworten aggregierten Informationen asynchron in den jeweiligen Services zur Datenhaltung ab und schicken gegebenenfalls weitere Abfragen an den Endpunkt.

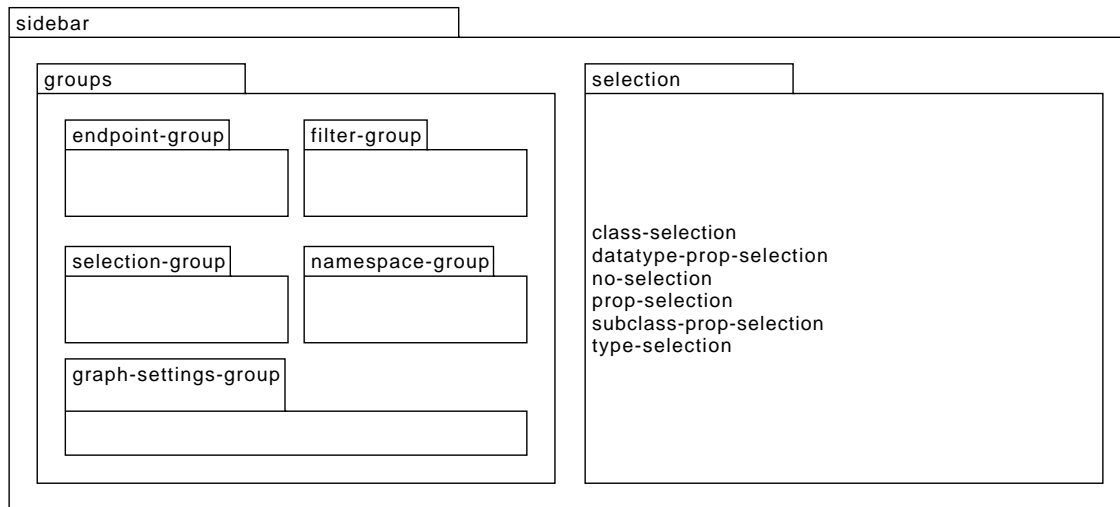
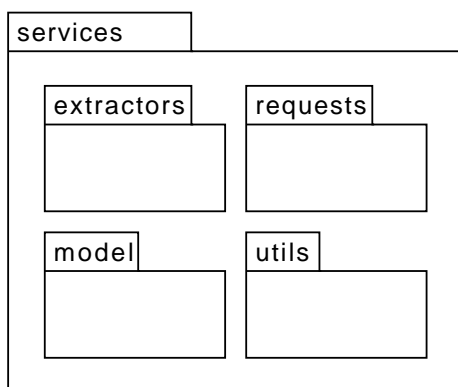
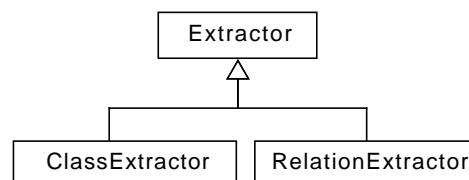


Abbildung 5.2.: UML-Paketdiagramm für das Sidebar Paket



(a) UML-Paketdiagramm der Services



(b) UML-Klassendiagramm der Extraktoren

Abbildung 5.3.: UML-Diagramme zum Service-Paket der Anwendung LDVOWL

Die Datenhaltungs-Services nutzen den HTML5 Sessionstorage bzw. Localstorage, um bereits extrahierte Information über ein Neuladen der Webseite hinweg zu speichern.

Alle Services sind nach dem Entwurfsmuster der Singletons implementiert, es existiert damit jeweils genau ein Service von jeder Art.

5.3.3. Konnektoren

Nach außen schickt die Anwendung mit Hilfe von HTTP-GET SPARQL-Abfragen an den vom Nutzer bestimmten Endpunkt.

Innerhalb der Anwendung kommunizieren die einzelnen Komponenten über gemeinsam genutzte Services sowie den von Angular bereitgestellten Ereignis-Bus, um auf Änderungen zu reagieren.

5.4. Allgemeine Funktionsweise

In diesem Abschnitt wird die allgemeine Funktionsweise der Anwendung beschrieben. Bei der Entwicklung wurde bei allen Einstellungsmöglichkeiten darauf geachtet, sinnvolle Standardwerte oder Vorauswahlen anzubieten, um den Benutzer nicht zu überfordern und ihm den Einstieg in die Verwendung der Anwendung so leicht wie möglich zu machen. Des Weiteren orientiert sich die Anwendung am Visual Information-Seeking Mantra von Ben Shneiderman [Shn96]: „Overview first, zoom and filter, then details-on-demand.“

5.4.1. Überblick

Zu Beginn wählt der Benutzer einen SPARQL-Endpunkt aus. Dann werden die wichtigsten Klassen extrahiert und auf Äquivalenz bzw. Subklassenbeziehungen überprüft. Anschließend werden Beziehungen zwischen den Instanzen der Klassen gesucht und nach und nach geladen und dargestellt. Außerdem wird nach Datentypen gesucht, welche über Properties mit den Instanzen der Klassen verbunden sind. Während und nach der Extraktion der Daten kann der Nutzer mit der Visualisierung interagieren und sich die Details anzeigen lassen, welche bei Bedarf dynamisch nachgeladen werden.

Im Folgenden werden die einzelnen Schritte im Detail erklärt, beginnend mit der Auswahl des zu untersuchenden Endpunkts.

5.4.2. Auswahl des SPARQL-Endpunkts

Auf der in Abbildung 5.4 dargestellten Startseite der Anwendung wählt der Benutzer einen SPARQL-Endpunkt aus, für welchen Ontologien extrahiert und dargestellt werden sollen.

Diese Auswahl geschieht über eine URL, welche der Benutzer manuell eintragen kann. Um den Start so einfach wie möglich zu gestalten, hat der Nutzer außerdem die Möglichkeit, einen aus einer Liste von vorgeschlagenen Endpunkten auszuwählen oder es bei der Vorauswahl von DBpedia zu belassen. Optional können sämtliche SPARQL-Abfragen auch über einen

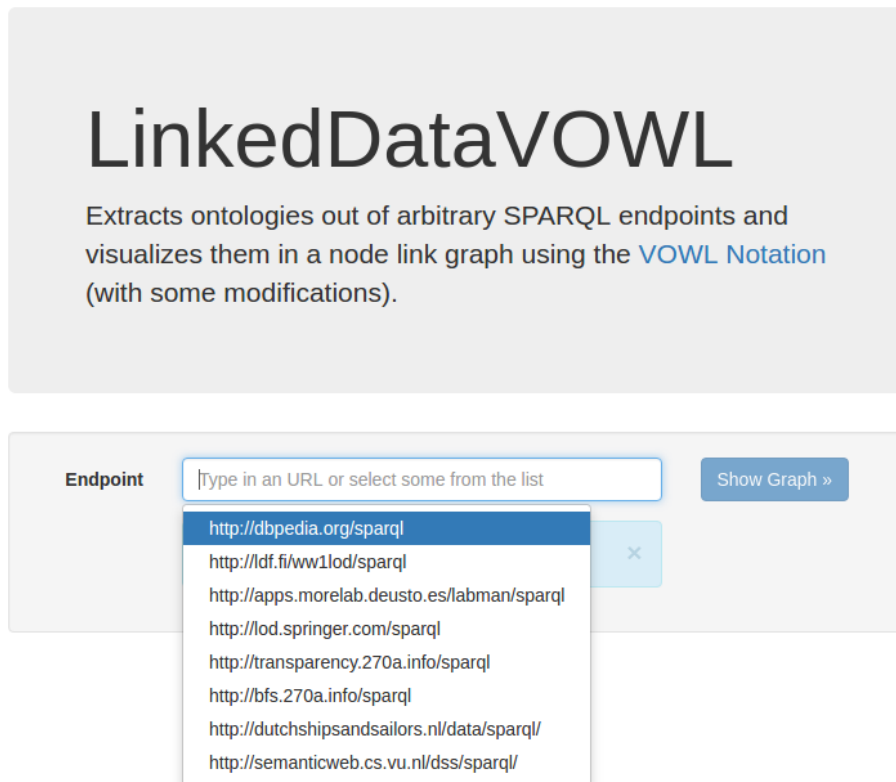


Abbildung 5.4.: Startseite der Webanwendung LDVOWL mit Eingabefeld zur Auswahl des zu untersuchenden SPARQL-Endpunktes

lokalen HTTP-Proxy geleitet werden, um damit Probleme mit Cross-Origin Resource Sharing (CORS) zu vermeiden. Dazu muss ein lokaler HTTP-Proxy ausgeführt werden und bei der Auswahl des Endpunktes das entsprechende Feld aktiviert sein. Mit der Bestätigung der Auswahl wird die Anzeige des Graphen aufgerufen und die Extraktion der Daten aus dem gewählten Endpunkt beginnt.

5.4.3. Extraktion der wichtigsten Klassen

Nach der Auswahl des Endpunkts extrahiert die Anwendung zunächst die wichtigsten Klassen. In diesem Ansatz gilt eine Klasse als wichtig, wenn im zugrundeliegenden Datensatz viele Instanzen von dieser existieren. Im Normalfall werden die zehn Klassen mit den meisten Instanzen geladen. Aufgrund der daraus resultierenden Zahl benötigter Abfragen stellt diese Beschränkung eine sinnvolle Vorauswahl dar, kann vom Benutzer jedoch jederzeit in den Einstellungen nach oben oder unten angepasst werden. Die zur Extraktion verwendete SPARQL-Abfrage ist in Listing 5.1 aufgeführt.

5. Entwickeltes System

```
1 SELECT DISTINCT ?class (COUNT(?sub) AS ?instanceCount)
2 WHERE {
3     ?sub a ?class.
4 }
5 GROUP BY ?class
6 ORDER BY DESC(?instanceCount)
7 LIMIT 10 OFFSET 0
```

Listing 5.1: SPARQL-Abfrage nach den zehn Klassen mit den meisten Instanzen

```
1 SELECT (COUNT(?commonInstance) AS ?commonInstanceCount)
2 WHERE {
3     ?commonInstance a <classURI1> .
4     ?commonInstance a <classURI2> .
5 }
```

Listing 5.2: SPARQL-Abfrage für Anzahl gemeinsamer Instanzen der Klassen A und B

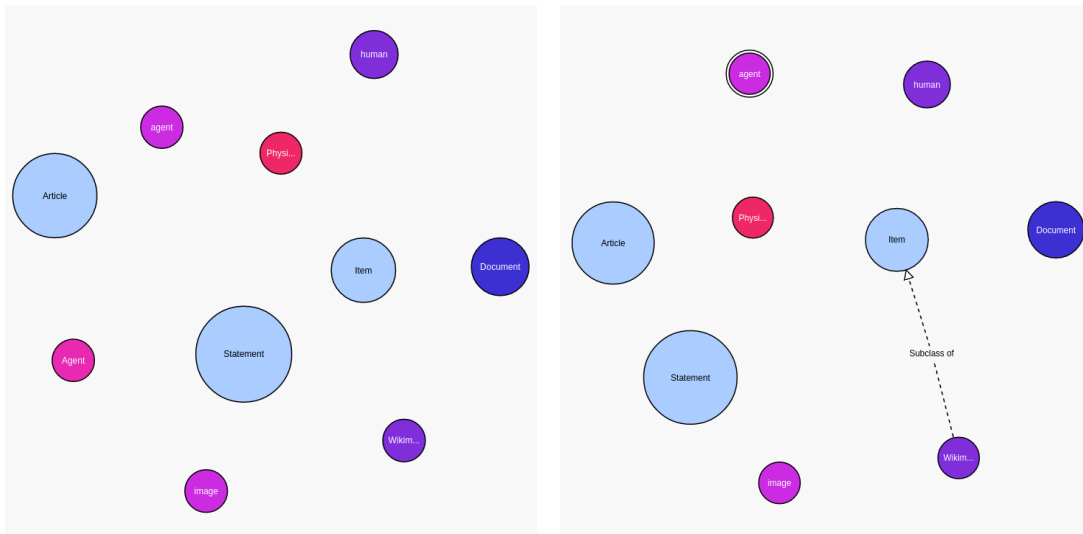
Anschließend werden Klassen, die sich auf der Blacklist befinden, client-seitig herausgefiltert und die übrigen Klassen der Visualisierung hinzugefügt. Da die Ausführungszeit für die SPARQL-Abfrage auf dem Endpunkt beschränkt ist und der Nutzer möglichst schnell erste Informationen zu sehen bekommen soll, wird die Filterung client-seitig ausgeführt und die erste SPARQL-Abfrage damit nicht unnötig komplizierter gemacht. Bleiben durch die Filterung weniger als 10 Klassen übrig, werden weitere Klassen nachgeladen. Ein Beispiel der daraus resultierenden grafische Darstellung für den DBpedia Endpunkt findet sich in Abbildung 5.5a.

5.4.4. Erkennung von Äquivalenz- und Subklassenbeziehungen

Um unnötige Abfragen nach Relationen zu vermeiden, werden die zuvor geladenen Klassen zunächst auf Äquivalenz geprüft.

Um Äquivalenz wie auch Kindklassenbeziehungen unter den identifizierten Klassen zu erkennen, wird die Anzahl der paarweise gemeinsamen Instanzen benötigt. Dies ergibt sich aus der Antwort auf die in Listing 5.2 aufgeführte Abfrage.

Durch den Vergleich der Anzahl gemeinsamer Instanzen mit denen der einzelnen Klassen werden die zuvor geladenen Klassen auf Subklassenbeziehungen und Äquivalenz geprüft: Falls die Anzahl der gemeinsamen Instanzen zweier Klassen C_1 und C_2 gleich der Instanzzahl der einzelnen Klassen ist, so sind die Klassen C_1 und C_2 äquivalent. Zwei als äquivalent identifizierte Klassen werden zusammengeführt, indem die URI der einen Klasse der anderen als Alternative hinzugefügt wird und anschließend verworfen wird. Ist die Anzahl der gemeinsamen Instanzen gleich der kleineren Instanzanzahl, so ist die Klasse mit der geringeren



(a) Darstellung vor der Überprüfung auf Äquivalenz- und Subklassenbeziehungen (b) Darstellung nach der Überprüfung auf Äquivalenz- und Subklassenbeziehungen

Abbildung 5.5.: Graphische Darstellungen der mächtigsten Klassen für den DBpedia Endpunkt

Anzahl an Instanzen eine Kindklasse der Mächtigeren. Zwei als Vater- und Kindklasse identifizierte Klassen werden in der Graphdarstellung durch eine entsprechende Kante verbunden. Im Falle einer gemeinsamen Instanzzahl von Null sind die Klassen C_1 und C_2 disjunkt.

Abbildung 5.5b zeigt die graphische Darstellung nach der Überprüfung der Klassen auf mögliche Äquivalenz oder Subklassenbeziehungen.

5.4.5. Suchen nach Beziehungen zwischen den Klassen

Nach der Überprüfung der Klassen auf Äquivalenz- oder Subklassenbeziehungen werden Properties zwischen den Instanzen zweier Klassen gesucht. Hierfür werden SPARQL-Abfragen für jedes mögliche Paar (inklusive Schlingen) der zuvor geladenen Klassen generiert und anschließend an den SPARQL-Endpunkt gesendet. Damit ist die Anzahl der Abfragen $N_{Requests} \in \mathcal{O}(n^2)$ für n Klassen. Um häufig vorkommende und damit repräsentative Properties an erster Stelle zu erhalten werden die Properties im Normalfall absteigend nach der Anzahl beteiligter Instanzen der Ausgangsklasse sortiert. Da dies je nach Datensatz und Klasse u. U. sehr lange dauern kann, können die Properties auch unsortiert abgerufen werden, indem die Sortierung für Properties in den Einstellungen deaktiviert wird.

Beispiele für sortierte und unsortierte Abfragen sind in Listing 5.3 und Listing 5.4 abgebildet.

5. Entwickeltes System

```
1 SELECT (COUNT(?originInstance) as ?count) ?prop
2 WHERE {
3     ?originInstance a <http://dbpedia.org/ontology/Agent> .
4     ?targetInstance a <http://xmlns.com/foaf/0.1/Document> .
5     ?originInstance ?prop ?targetInstance .
6 }
7 GROUP BY ?prop
8 ORDER BY DESC(?count)
9 LIMIT 10 OFFSET 0
```

Listing 5.3: Beispiel für eine SPARQL-Abfrage nach einer sortierter Liste verschiedener Properties zwischen den Instanzen der Klassen „agent“ und „Document“ auf DBpedia

```
1 SELECT distinct ?prop
2 WHERE {
3     ?originInstance a <http://dbpedia.org/ontology/Agent> .
4     ?targetInstance a <http://www.wikidata.org/entity/Q5> .
5     ?originInstance ?prop ?targetInstance .
6 }
7 LIMIT 10 OFFSET 0
```

Listing 5.4: Beispiel für SPARQL-Abfrage nach verschiedenen Properties zwischen Instanzen der Klassen „agent“ und „human“ auf DBpedia

Zwischen zwei Klassen werden unter Umständen sehr viele Properties gefunden, um lange Wartezeiten zu vermeiden werden zunächst maximal zehn dieser Properties geladen. Wird die Abfrage vom Endpunkt mit der maximalen Anzahl an Treffern beantwortet, wird eine neue Abfrage mit verdoppeltem Limit gesendet. Damit wird sowohl die Antwortzeit der Abfragen gering gehalten, als auch die Anzahl der Abfragen an den SPARQL-Endpunkt minimiert.

Die geladenen Properties werden als beschriftete und gerichtete Kanten dargestellt, dabei wird für die Kantenbeschriftung eine Property ausgewählt, die Namen aller anderen Properties kann der Benutzer sich durch Selektieren der Kante in der Sidebar anzeigen lassen. Im Falle einer sortierten Abfrage wird die Kante mit der Property mit den meisten Instanzen beschriftet, bei einer unsortierten Abfrage durch die, welche vom Endpunkt zuerst zurückgegeben wird. Die Stärke der Kante stellt die Anzahl verschiedener Properties dar, sie wird logarithmisch skaliert, um die Lesbarkeit des Graphen zu gewährleisten.

5.4.6. Suche nach Datentypen

Parallel zur Suche nach direkten Beziehungen zwischen den Instanzen zweier Klassen werden außerdem Beziehungen zwischen Instanzen und Datentypen gesucht. Hierfür werden zunächst die Datentypen gesucht, mit welchen die Instanzen einer bestimmten Klasse am

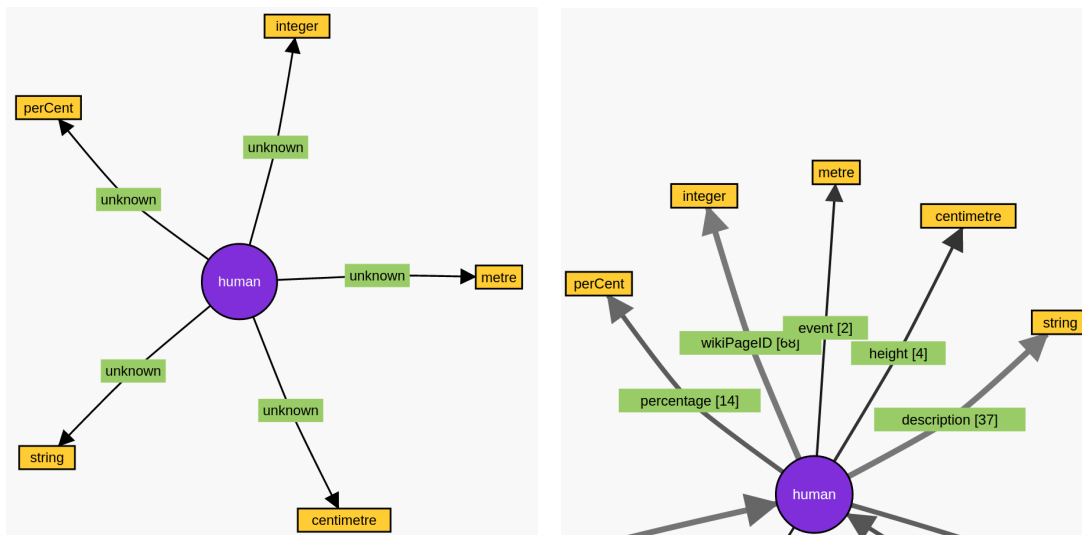
```

1 SELECT (COUNT(?val) AS ?valCount) ?valType
2 WHERE {
3     ?instance a <http://dbpedia.org/ontology/Agent> .
4     ?instance ?prop ?val .
5     BIND(DATATYPE(?val) AS ?valType) .
6 }
7 GROUP BY ?valType
8 LIMIT 10

```

Listing 5.5: Beispiel für SPARQL-Abfrage nach Datentypen, welche für Instanzen von „agent“ am häufigsten auftreten

häufigsten in Verbindung gebracht werden. Dies geschieht durch die Ausführung einer Abfrage wie sie für Instanzen der Klasse „Agent“ beispielhaft in Listing 5.5 dargestellt ist.



(a) Platzhalter-Properties zwischen Klasse und Datentypen (b) Nach Abschluss der Suche nach Klasse-Datentyp Relationen

Abbildung 5.6.: Beispiele für Darstellungen von Klasse-Datentyp Relationen in LDVOWL

Diese Abfrage wird einmal für jede Klasse gesendet, da ein Datentyp für jede Klasse einzeln und damit in der Visualisierung unter Umständen mehrfach dargestellt wird.

Erst nachdem diese Datentypen bestimmt wurden, wird in einer weiteren Abfrage nach der konkreten Verbindung zwischen diesen Datentypen und Instanzen der Klasse gefragt. Listing 5.6 stellt ein Beispiel für eine solche Abfrage dar.

Solange die konkreten Properties zwischen Instanzen der Klasse und dem jeweiligen Datentyp noch nicht extrahiert sind, wird eine Platzhalter-Property verwendet, um die Klasse mit den jeweiligen Darstellungen der Datentypen zu verbinden. Sobald die verbindende Property bestimmt wurde, wird der Platzhalter durch die extrahierte Beziehung ersetzt.

5. Entwickeltes System

```
1 SELECT DISTINCT ?prop
2 WHERE {
3     ?instance a <http://dbpedia.org/ontology/Agent> .
4     ?instance ?prop ?val .
5     FILTER (DATATYPE(?val) = <http://www.w3.org/2001/XMLSchema#string>)
6 }
7 LIMIT 1
```

Listing 5.6: Beispiel für die Abfrage nach Properties zwischen Instanzen der Klasse „Agent“ und dem Datentyp „string“

Abbildung 5.6a zeigt die vorübergehende Darstellung mit Platzhalter, Abbildung 5.6b die Darstellung nach Abschluss der Extraktion.

Der Grund für diesen zweiphasigen Ansatz liegt ein weiteres Mal bei der beschränkten Ausführungszeit für einzelne SPARQL-Abfragen wie auch dem Ziel, dem Benutzer die extrahierten Informationen möglichst schnell anzuzeigen und anschließend zu verfeinern. Die Platzhalter werden verwendet, um dem Nutzer die bereits extrahierten Informationen schneller präsentieren zu können.

5.4.7. Blacklists für Klassen und Properties

Da mit der Anwendung ABox-Informationen aus Linked Data extrahiert werden sollen, werden Klassen und Properties, die in RDF, RDFS oder OWL definiert sind und zur TBox zählen, im Normalfall ignoriert. Dies geschieht über zwei Blacklists, eine für Klassen und eine für Properties, mit welchen die extrahierten Informationen abgeglichen werden. Auch wenn die Blacklists per Voreinstellung sämtliche RDF-, RDFS- und OWL-Klassen und Properties enthalten, bietet die Anwendung dem Benutzer dennoch die Möglichkeit, Einträge aus der Liste zu entfernen oder weitere unerwünschte Klassen oder Properties hinzuzufügen. Dazu trägt der Benutzer deren URIs durch Kommas getrennt in die jeweilige Liste auf der in Abbildung 5.7 abgebildeten Einstellungsseite ein und bestätigt seine Änderung. Die Voreinstellungen der beiden Blacklists finden sich in Tabelle 5.1 und Tabelle 5.2.

SPARQL

Labels Limit

fetch properties ordered

Blacklists

Lists RDF RDFS OWL SKOS

⚠ Any manual changes below will be overwritten!

Classes

`http://www.w3.org/1999/02/22-rdf-syntax-ns#List,
http://www.w3.org/1999/02/22-rdf-syntax-ns#langString,
http://www.w3.org/1999/02/22-rdf-syntax-ns#HTML,
http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral,
http://www.w3.org/1999/02/22-rdf-syntax-ns#Property,
http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag,`

Properties

`http://www.w3.org/1999/02/22-rdf-syntax-ns#type,
http://www.w3.org/1999/02/22-rdf-syntax-ns#first,
http://www.w3.org/1999/02/22-rdf-syntax-ns#rest,
http://www.w3.org/1999/02/22-rdf-syntax-ns#value,
http://www.w3.org/1999/02/22-rdf-syntax-ns#subject,
http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate,`

Abbildung 5.7.: Einstellungsseite der Anwendung LDVOWL mit editierbaren Blacklists

5.4.8. Caching

Da vor allem für die Extraktion der Properties zwischen Instanzen der Klassen sehr viele Abfragen an den SPARQL-Endpunkt gestellt werden müssen und die Antwortzeiten auf diese Abfragen oft im Bereich von mehreren Sekunden liegen (siehe dazu auch Abschnitt 6.7), spielt das Caching der Antworten eine wichtige Rolle.

Alle Klassen und Properties werden client-seitig gespeichert, sodass diese z. B. beim Neuladen der Webseite im Browser nicht erneut abgefragt werden müssen. Je nach Konfiguration der Anwendung werden entweder durch das Schließen des Browserfensters bzw. -tabs oder das

5. Entwickeltes System

RDF	RDFS	OWL	
List	Resource	AllDifferent	Nothing
langString	Literal	AnnotationProperty	ObjectProperty
HTML	Class	Class	Ontology
XMLLiteral	Datatype	DataRange	OntologyProperty
Property	Statement	DatatypeProperty	Restriction
Bag	Container	DeprecatedClass	SymmetricProperty
Seq	ContainerMembershipProperty	DeprecatedProperty	Thing
Alt		FunctionalProperty	TransitiveProperty
		InverseFunctionalProperty	

Tabelle 5.1.: Klassen-Blacklist für RDF-, RDFS- und OWL-Namensräume

RDF	RDFS	OWL		
type	subClassOf	allValuesFrom	hasValue	priorVersion
first	subPropertyOf	backward- CompatibleWith	imports	sameAs
rest	domain	cardinality	incompatibleWith	someValuesFrom
value	range	complementOf	intersectionOf	unionOf
subject	label	differentFrom	inverseOf	versionInfo
predicate	comment	disjointWith	maxCardinality	
object	member	distinctMembers	minCardinality	
	seeAlso	equivalentClass	oneOf	
	isDefinedBy	equivalentProperty	onProperty	

Tabelle 5.2.: Property-Blacklist für RDF, RDFS und OWL-Namensräume

Leeren des HTML5 LocalStorage alle bereits geladenen Informationen über den aktuellen Datensatz verworfen und müssen vom Endpunkt neu abgerufen werden.

5.5. Interaktion mit der Visualisierung

In diesem Abschnitt werden die verschiedenen Möglichkeiten vorgestellt, mit welchen der Benutzer der Anwendung mit der Visualisierung interagieren kann. Dabei wird auch genau auf die einzelnen Ansichten und Bedienelemente innerhalb der Sidebar eingegangen.

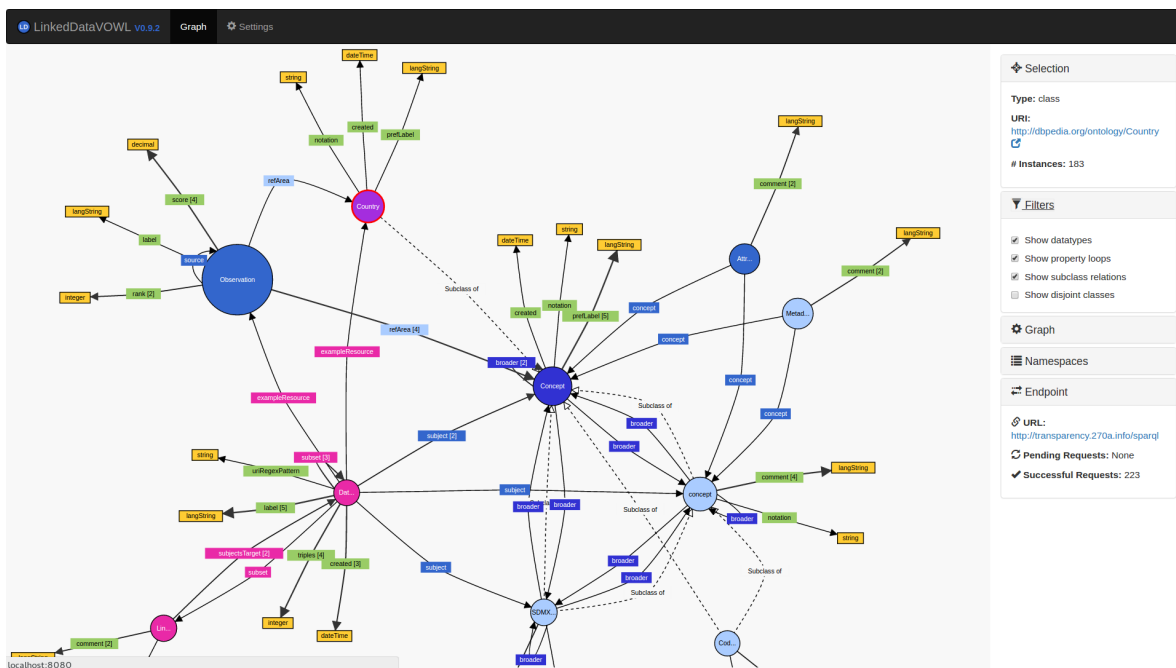


Abbildung 5.8.: Darstellung von LDVOWL für den SPARQL-Endpoint Transparency International Linked Data (<http://transparency.270a.info/sparql>)

5.5.1. Wahl des Bildausschnitts (Zoom)

Zu Beginn soll der Benutzer einen Überblick über die Inhalte des Datensatzes bekommen (Overview first). Um dies zu erleichtern kann der Benutzer den anzuzeigenden Bildausschnitt in der Visualisierung frei wählen, indem er die Anzeige mit Hilfe des Mausekkrads vergrößert (Zoom), Knoten bewegt (Drag-and-Drop) oder den angezeigten Ausschnitt verschiebt (Panning).

5.5.2. Selektion (Details on Demand)

Nachdem der Benutzer sich einen Überblick verschafft hat, kann er sich bei Bedarf mehr Informationen anzeigen lassen (Details on Demand). Durch einen Mausklick können Knoten im dargestellten Graphen selektiert werden. In der Sidebar öffnet sich dann automatisch die in Abbildung 5.9 dargestellte Auswahlgruppe, welche alle vorhandenen Informationen zum ausgewählten Element auflistet. URIs werden dabei direkt als Hyperlinks aufgeführt, sodass sich der Nutzer die Definitionen mit einem Klick in einem neuen Tab des Browsers anzeigen lassen kann.

Für ausgewählte Klassen oder Datentypen werden RDF-, SKOS- oder mittels regulärem Ausdruck aus der URI extrahierte Labels angegeben. Bei Klassen wird zusätzlich die genaue

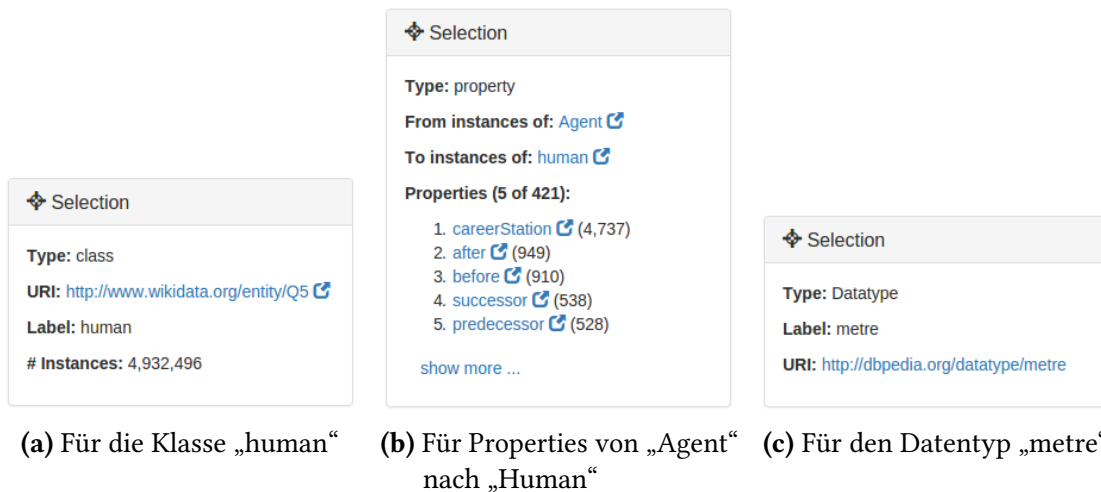


Abbildung 5.9.: Details für verschiedene ausgewählte Elemente in Auswahlgruppe der Sidebar in LDVOWL

Anzahl Instanzen angegeben. Für eine ausgewählte Beziehung werden hier die Klassen der Instanzen im Definitions- und Wertebereich der beteiligten Properties angezeigt sowie die beteiligten Properties aufgelistet.

5.5.3. Filterung der Darstellung (Filter)

Die Anwendung bietet dem Benutzer die Möglichkeit, bestimmte potentiell unerwünschte Bestandteile der Graphdarstellung auszublenden, um eine übersichtlichere Darstellung zu erhalten.

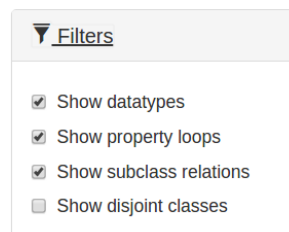


Abbildung 5.10.: Sidebar-Gruppe für Filterung der Graphansicht von LDVOWL

Dazu muss der Nutzer lediglich in der in Abbildung 5.10 dargestellten Filtergruppe der Sidebar die jeweiligen Felder abwählen. Die Graphansicht passt sich sofort an die neue Filterung an, wobei die zugrundeliegenden Informationen im Browser erhalten bleiben und nur ihre visuelle Repräsentation aus dem dargestellten Graphen entfernt wird.

Alle filterbaren Elemente des Graphen finden sich in Tabelle 5.3.

Filter-Bezeichnung	betroffene Elemente	Voreinstellung
Datentypen	Datentyp-Knoten sowie die Properties, welche diese als Ziel haben	anzeigen
Property-Schlingen	Properties, die von einer bestimmten Klasse zu selbiger verlaufen	anzeigen
Subklassenbeziehungen	Kanten welche Subklassen mit deren Elternklassen verbinden	anzeigen
Disjunkt-Knoten	Knoten welche die Disjunktheit zweier Klassen darstellen	verbergen

Tabelle 5.3.: In VOWL verfügbare Filteroptionen sowie deren Voreinstellung

Im Normalfall wird alles bis auf die Disjunkt-knoten dargestellt. Die Disjunktheit von Klassen ist deshalb per Vorauswahl ausgeblendet, da sie die Form des Graphen stark verändern kann indem sie z. B. nicht zusammenhängende Teile miteinander verbindet.

5.5.4. Anpassung der Kantenlängen

Liegen die Knotendarstellungen von Klassen und Properties in einem besonders dichten Graphen zu nah aufeinander bzw. sollen bestimmte Cluster dichter zusammengezogen werden, so kann dies der Benutzer über eine Anpassung der Kantenlängen in den Grapheinstellungen der Sidebar erreichen. In der Gruppe für Grapheinstellungen (siehe Abbildung 5.11) können die Abstände mit Hilfe von Slidern angepasst werden. Hierbei können die Kantenlängen zwischen Klassen und die zwischen Klassen und Datentypen vom Nutzer separat eingestellt werden.

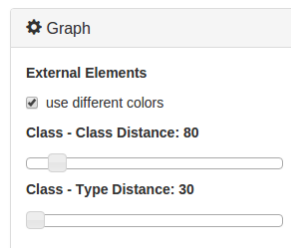
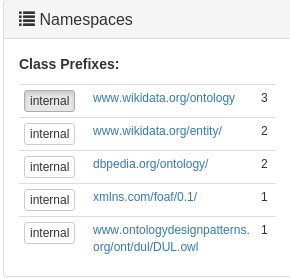


Abbildung 5.11.: Sidebar-Gruppe für Grapheinstellungen in LDVOWL

Diese separate Anpassung ist sinnvoll, da Datentypen im Graphen mehrfach dargestellt werden und deshalb nur eine eingehende Kante besitzen. Folglich können Properties zwischen Klassen und Datentypen stärker zusammengezogen werden, ohne dass dadurch die Lesbarkeit des Graphen leidet.

5.5.5. Klassifikation von externen und internen Namensräumen

Bei der Extraktion klassifiziert die Anwendung den Namensraum zu welchem die meisten extrahierten Klassen gehören als den internen Namensraum. Diese Heuristik basiert auf der Annahme, dass die am häufigsten verwendeten Klassen eines Datensatzes, welche nicht auf der Blacklist stehen, hauptsächlich zum eigenen Namensraum gehören.



Namespaces		
Class Prefixes:		
<input checked="" type="checkbox"/> internal	www.wikidata.org/ontology	3
<input type="checkbox"/> internal	www.wikidata.org/entity/	2
<input type="checkbox"/> internal	dbpedia.org/ontology/	2
<input type="checkbox"/> internal	xmlns.com/foaf/0.1/	1
<input type="checkbox"/> internal	www.ontologydesignpatterns.org/ont/dul/DUL.owl	1

Abbildung 5.12.: Beispiel für Darstellung der Namensraumgruppe in der Sidebar von LD-VOWL (auf DBpedia Endpunkt)

Ist diese Annahme für einen bestimmten Datensatz nicht zutreffend, so kann die automatisch vorgenommene Klassifikation der einzelnen Namensräume jederzeit durch den Benutzer manuell angepasst werden. Dazu müssen in der Namensraumgruppe der Sidebar, welche in Abbildung 5.12 dargestellt ist, lediglich die Buttons für den jeweiligen Namensraum an- oder abgewählt werden.

Damit sich die einzelnen externen Namensräume besser voneinander abheben, werden diese in verschiedenen Farben dargestellt. Auch dieses Verhalten kann in den Grapheinstellungen manuell deaktiviert werden, sodass die zugehörigen Klassen und Properties mit dem selben blauen Hintergrund dargestellt werden.

6. Evaluation

Die Evaluation des zuvor beschriebenen Ansatzes besteht aus zwei Teilen: Zum einen wird die in Kapitel 5 vorgestellte Anwendung mittels einer Nutzerstudie evaluiert. Zum anderen werden Performanzdaten zu verschiedenen Endpunkten erhoben, um daraus Rückschlüsse auf die Skalierbarkeit des Ansatzes ziehen zu können.

In den folgenden Abschnitten wird zunächst auf die Nutzerstudie eingegangen.

6.1. Entwurf der Nutzerstudie

Die Probanden untersuchen die in verschiedenen SPARQL-Endpunkten hinterlegten Informationen mit Hilfe der zuvor beschriebenen Anwendung und müssen dabei eine Reihe von Fragen zu Inhalt und Aufbau der jeweiligen Datensätze beantworten. Nach der Beantwortung der Fragen haben die Studienteilnehmer die Möglichkeit, qualitatives Feedback zur Benutzbarkeit zu geben, indem sie einige offene Fragen in einem weiteren Fragebogen beantworten.

Der zu untersuchende SPARQL-Endpunkt stellt dabei die unabhängige Variable, die von den Probanden gegebenen Antworten die abhängigen Variablen dar.

6.1.1. Hypothesen

Im Rahmen der Nutzerstudie sollen die folgenden Hypothesen überprüft werden:

1. Die Benutzer bekommen einen Überblick über die zentralen Inhalte des zu untersuchenden Datensatzes.
2. Die Benutzer können die relevantesten Namensräume in einem Datensatz benennen.
3. Die Benutzer können die mächtigsten Klassen im Datensatz benennen.
4. Die Benutzer finden mit der Anwendung die häufigste Property zwischen zwei Klassen.
5. Die Benutzer können mit Hilfe der Anwendung Beispiele für Relationen zwischen den Instanzen zweier Klassen nennen.

6. Die Benutzer können mit Hilfe der Anwendung Beispiele für die zu einer Klasse hinterlegten Daten geben.

6.2. Fragen für die Nutzerstudie

Um zu überprüfen, ob die extrahierten Ontologie-Informationen dem Nutzer einen guten Überblick über den zugrundeliegenden Datensatz geben und die Ontologie-Informationen verständlich darstellen werden, wird den Studienteilnehmern eine Reihe allgemeiner sowie endpunktspezifischer Fragen gestellt.

Zu Beginn werden für den aktuellen Endpunkt allgemeine Fragen gestellt, welche überprüfen sollen, ob sich der Proband mit der Anwendung einen Überblick über die Inhalte des dort hinterlegten Datensatzes verschaffen kann.

6.2.1. Allgemeine Fragen

Die folgenden Fragen wurden zu jedem Endpunkt in dieser Reihenfolge gestellt:

1. Um was geht es in diesem Datensatz?
2. Welches ist die Klasse mit den meisten Instanzen?
3. Wie viele verschiedene Namensräume (intern und extern) werden verwendet?
4. Wie heißt der Namensraum, welcher die meisten der angezeigten Klassen enthält?

Nachdem der Proband die allgemeinen Fragen zu einem Endpunkt beantwortet hat, werden ihm einige endpunktspezifische Fragen gestellt.

6.2.2. Endpunktspezifische Fragen

Zu den Klassen, Properties und Datentypen eines bestimmten Endpunktes sollen Fragen der folgenden Form beantwortet werden:

Frage zu Klassen

- Wie viele Instanzen hat die Klasse *C*?

Fragen zu Properties und Datentypen

- Welche Beziehung besteht zwischen Instanzen der Klassen C_1 und C_2 ?
- Wie viele verschiedene Properties gibt es zwischen den Instanzen der Klassen C_1 und C_2 ?
- Welches ist die am häufigsten auftretende Property P zwischen Instanzen der Klassen C_1 und C_2 ?
- Nennen Sie ein Beispiel für die zu Instanzen der Klasse C hinterlegten Daten.

Die im Rahmen der Nutzerstudie verwendeten Fragebögen mit den für den jeweiligen Endpunkt ausformulierten Fragen befinden sich in Anhang A.3.2 und A.3.3.

6.2.3. Auswahl der Endpunkte für die Nutzerstudie

Für die Nutzerstudie wurden Endpunkte gewählt, deren extrahierte Ontologie-Informationen in Graphen verschiedener Dichte dargestellt werden.

Ausgangspunkt für die Suche nach SPARQL-Endpunkten stellte die Webseite „SPARQL Endpoints Status“¹ der Wirtschaftsuniversität Wien dar. Dabei wurden Endpunkte, welche die in Abschnitt 5.1.1 beschriebenen Voraussetzungen nicht erfüllten oder sich im Test als sehr langsam oder unzuverlässig erreichbar herausstellten, bei der Suche nicht weiter berücksichtigt.

Schließlich wurden die folgenden Endpunkte für die Durchführung der Nutzerstudie ausgewählt:

1. Datensatz über Nobelpreisträger²
2. Datensatz über österreichische Skifahrer³

¹<http://sparql.es.ai.wu.ac.at/>

²<http://data.nobelprize.org/sparql>

³<http://vocabulary.semantic-web.at/PoolParty/sparql/AustrianSkiTeam>

6.3. Versuchsaufbau

Die zu untersuchende Anwendung LDVOWL wird im Browser Chromium 48 auf einem handelsüblichen Notebook (Lenovo Thinkpad T440p, Betriebssystem Ubuntu 15.10 GNOME) ausgeführt.

Für die Beantwortung der Fragen können die Probanden die Darstellung des jeweiligen Endpunkts betrachten sowie über das Touchpad des Notebooks oder eine externe Maus mit der Visualisierung interagieren. Die Ontologie-Informationen sind zu diesem Zeitpunkt bereits extrahiert und gespeichert, sodass alle Probanden die selben Informationen gezeigt bekommen und ohne Wartezeit sofort mit der Untersuchung des Datensatzes beginnen können.

Die zu untersuchende Anwendung wurde für die Nutzerstudie so konfiguriert, dass der Name des aktuell dargestellten Endpunkts nicht angezeigt wird, damit der Proband über den Namen keine Rückschlüsse auf die Inhalte ziehen kann. Die Beantwortung der Fragebögen geschieht mit Hilfe von Google Forms online von einem zweiten Computer aus.

6.4. Durchführung der Nutzerstudie

Zu Beginn der Nutzerstudie lesen die Teilnehmer die ihnen vorgelegte Einverständniserklärung (siehe Anhang A.1) und unterschreiben diese. Anschließend füllen sie den Fragebogen zu den demographischen Daten (Geschlecht, Alter, Beruf bzw. Tätigkeitsfeld) sowie bereits vorhandenen Vorkenntnissen aus den Bereichen Linked Data und Ontologien aus (siehe Anhang A.3.1).

Nachdem den Probanden die in Kapitel 4 verwendete visuelle Notation mit Hilfe einer in Anhang A.2 abgebildeten kleinen Übersicht erklärt wurde, beginnen sie mit der Untersuchung des ersten Endpunkts und der Beantwortung der dazugehörigen Fragen. Es folgt ein weiterer Durchlauf auf dem jeweils anderen Endpunkt, wobei die Reihenfolge, in welcher die Endpunkte abgearbeitet werden von Proband zu Proband getauscht wird, um einen Trainingseffekt auszuschließen.

Nach der Beantwortung der Fragen haben die Studienteilnehmer in einem dritten Fragebogen die Möglichkeit qualitatives Feedback zur Benutzbarkeit der Anwendung abzugeben.

6.5. Studienteilnehmer

An der Nutzerstudie nahmen insgesamt sieben Personen im Alter zwischen 16 und 25 Jahren teil (Mittelwert ca. 21,4). Davon waren sechs Männer und eine Frau. Die Teilnehmer für die

Nutzerstudie wurden hauptsächlich auf dem Campus angeworben, deshalb handelt es sich bei den Probanden überwiegend um Studenten aus dem Fachbereich Informatik/Softwaretechnik. Lediglich ein Teilnehmer gab an, bereits Erfahrungen mit Linked Data zu haben.

6.6. Ergebnisse der Nutzerstudie

In diesem Abschnitt werden die Ergebnisse der Nutzerstudie vorgestellt. Hierbei wird zunächst auf die Ergebnisse der Fragebögen, anschließend auf das qualitative Feedback eingegangen.

6.6.1. Fragebögen zu Datensätzen

Im Folgenden werden die Ergebnisse der Antworten auf die allgemeinen und endpunktspezifischen Fragen präsentiert. Dies erfolgt in der Reihenfolge, in welcher diese auch den Studienteilnehmern gestellt wurden.

Zentrale Inhalte

Alle Teilnehmer erkannten, dass sich der Datensatz hinter dem ersten Endpunkt mit Preisträgern befasste, fünf Studienteilnehmer gaben sogar an, dass es sich um den Nobelpreis handelt. Für den zweiten Datensatz erkannten alle Probanden, dass es sich um Wintersport/Skifahren handelte, vier davon gaben explizit Österreichische Skifahrer an. Dies steht im Einklang zur Hypothese, nach der die Benutzer durch die Visualisierung einen Überblick über die zentralen Inhalte des zu untersuchenden Datensatzes bekommen.

Mächtigste Klasse

Alle Teilnehmer erkannten die Klasse mit den meisten Instanzen im zweiten Datensatz, lediglich für den ersten Datensatz gab es eine falsche Antwort. Die Probanden erkannten die mächtigste Klasse meist allein über die Größe der Darstellung, ohne die einzelnen Anzahlen der Instanzen zu vergleichen. Das stützt stark die Hypothese, dass die Benutzer mit der Anwendung die mächtigste Klasse im Datensatz benennen können.

Anzahl verwendeter Namensräume

Die Frage nach der Anzahl der wichtigsten Namensräume wurde für den ersten Datensatz von lediglich zwei Teilnehmern richtig beantwortet. Drei Probanden nannten eine zu kleine Zahl, zwei eine zu große. Letztere haben unter Umständen die Farben von Datentypen sowie auf diese verweisenden Properties mitgezählt. Für den zweiten Datensatz gab es sogar nur eine richtige Antwort, für diesen Datensatz nannten alle sechs anderen Studienteilnehmer eine kleinere Zahl. Dieses Ergebnis zeigt, dass die Farbunterschiede für den gewählten Farbbereich zwischen rot und dunkelblau bereits bei vier Namensräumen zu klein sind, um diese sicher wahrnehmen zu können.

Relevante Namensräume

Für den ersten Datensatz erkannten die Teilnehmer den aus Sicht der angezeigten Klassen relevantesten Namensraum, beim zweiten Datensatz erkannte nur die Hälfte der Teilnehmer den Namensraum mit den meisten Klassen, die andere Hälfte nannte den Namensraum mit den meisten Properties. Dieser Fehler lässt sich dadurch erklären, dass der zweite Datensatz sehr viele Properties zwischen den zehn mächtigsten Klassen enthält und in der Darstellung deshalb die Farbe dieses Namensraums dominiert.

Beispiele für hinterlegte Daten

Die Studienteilnehmer gaben jeweils drei bis fünf korrekte Beispiele für oft zu Personen/Organisationen hinterlegten Daten an. Es wurden keine falschen Angaben gemacht. Dies stützt stark die Hypothese, die Anwendung befähige den Nutzer, Beispiele für die zu Instanzen bestimmter Klassen hinterlegten Daten geben zu können.

Anzahl Instanzen

Alle Probanden konnten die Fragen nach der Anzahl Instanzen einer bestimmten Klasse für beide Datensätze korrekt beantworten.

Erkennen von Beziehungen

Die Frage nach den Beziehungen zwischen den Instanzen wurde von allen Teilnehmern für beide Datensätze richtig beantwortet. Dies stützt stark die Hypothese, die Benutzer könnten mit Hilfe der Anwendung Beispiele für Relationen zwischen den Instanzen zweier Klassen nennen.

Anzahl verschiedener Properties

Während die überwiegende Mehrzahl der Teilnehmer die Frage nach der Anzahl verschiedener Properties für den zweiten Datensatz richtig beantworten konnte, so wurde die Frage für den ersten Datensatz mit eins statt zwei überwiegend zu niedrig angegeben. Möglicherweise handelt es sich hierbei um einen Flüchtigkeitsfehler und die an den Namen angehängte Zahl in eckigen Klammern wurde übersehen.

Häufigste Property

Bis auf eine Antwort für Datensatz zwei waren alle Antworten der Probanden auf die Frage nach der häufigsten Property zwischen den Instanzen zweier Klassen korrekt. Dies stützt die Aussage, die Probanden wären mit Hilfe der Anwendung in der Lage, die häufigste Property zwischen den Instanzen zweier Klassen zu bestimmen.

6.6.2. Qualitatives Feedback

Auf dem letzten Fragebogen für allgemeines Feedback zur Anwendung gaben zwei Teilnehmer an, ihnen habe besonders die flüssige Animation beim Verschieben der Knoten gefallen. Drei Teilnehmern mochten die Interaktion mit der Anwendung und nannten sie „intuitiv“ und „ansprechend“. Ein Proband gab außerdem an, das Bewegen der Knoten sei sehr hilfreich dabei gewesen, Überschneidungen im Graphen aufzulösen. Zwei Teilnehmern gefiel die Abbildung der Instanzzahlen auf die Größe der Klassen. Ein Teilnehmer gab an, die Darstellung des Datensatzes aus der Klassensicht helfe sehr dabei, die zugrundeliegende Ontologie zu verstehen.

Ein Teilnehmer gab an, ihn habe gestört, dass die Knoten nach dem Bewegen zum Teil in ihre Ausgangslage zurücksprangen. Durch die Erweiterung der Anwendung durch eine Pick-and-Pin Funktion ließe sich dieses Problem beheben. Zwei Teilnehmer empfanden das Nachverfolgen von Properties im dichteren Graphen als schwierig.

Die Problematik der geringen Farbunterschiede zwischen den Elementen verschiedener Namensräume fiel manchen Probanden während der Studie auf: Zwei Teilnehmer wünschen sich größere Farbunterschiede für die Kennzeichnung der verschiedenen Namensräumen. Ein Studienteilnehmer wünscht sich die Anzeige des vollen Namens für ausgewählte Klassen. Ein Proband schlägt eine Mehrfachauswahl für Klassen vor, welche anschließend die Verbindungen zwischen den selektierten Klassen hervorhebt.

6.7. Untersuchung der Performance

Nachdem im Rahmen der Nutzerstudie primär die Nützlichkeit der extrahierten Informationen sowie die Lesbarkeit der daraus erstellten Visualisierung untersucht wurde, soll im Folgenden die Performance der zuvor beschriebenen Anwendung bei der Extraktion evaluiert werden. Zunächst wird auf das Vorgehen sowie die für die Untersuchung ausgewählten Endpunkte eingegangen.

6.7.1. Vorgehen

Die Extraktion der Ontologie-Informationen wurde auf einer Reihe von zehn Endpunkten verschiedener Größe durchgeführt (siehe Tabelle 6.1). Dabei wurde sowohl die für die Extraktion benötigte Zeit gemessen, als auch die Anzahl der an den Endpunkt gerichteten Abfragen gezählt.

Um den Einfluss von etwaigen Lastspitzen auf den öffentlichen SPARQL-Endpunkten zu mindern, wurden die Messungen mehrmals und zu verschiedenen Zeiten durchgeführt. Alle Performance-Messungen wurden im Browser Chromium 48 unter Ubuntu 15.10 durchgeführt. Vor dem Beginn der Messung wurde sichergestellt, dass Cache sowie Local- und Sessionstorage des Browsers leer sind, sodass alle benötigten Informationen vom Endpunkt neu abgerufen werden müssen und keine Ergebnisse aus Cache oder Localstorage wiederverwendet werden können.

6.7.2. Untersuchte Endpunkte

Die für die Untersuchung der Performance ausgewählten SPARQL-Endpunkte finden sich in Tabelle 6.1. Die Liste enthält eine Reihe von SPARQL-Endpunkte verschiedener Größe, welche die in Abschnitt 5.1.1 genannten Voraussetzungen der Anwendung LDVOWL erfüllen und sich bei einigen Tests während der Entwicklung der Anwendung als zuverlässig erreichbar herausgestellt hatten.

6.7.3. Ergebnisse

Abbildung 6.1 zeigt die von LDVOWL durchschnittlich benötigte Zeit für die Extraktion der Ontologie-Informationen aus dem jeweiligen Endpunkt. Für Endpunkte mit kleinen Datensätzen wie etwa „Transparency International“ oder „Springer“ ist die Extraktion meist nach weniger als zwei Minuten abgeschlossen. Für größere Datensätze wie „DBpedia“ oder „LinkedGeoData“ wird mit fast 14 Minuten wesentlich mehr Zeit benötigt.

#	Name des Endpunkts	URL des Endpunkts
1	Archiveshub	http://data.archiveshub.ac.uk/sparql
2	DBCLS	http://data.allie.dbcls.jp/sparql
3	DBpedia	http://dbpedia.org/sparql
4	Dutch Ships and Sailors	http://semanticweb.cs.vu.nl/dss/sparql/
5	Ecuador Research	http://data.utpl.edu.ec/ecuadorresearch/lod/sparql
6	LinkedGeoData	http://linkedgeo.org/sparql
7	Nobelprize	http://data.nobelprize.org/sparql
8	Onto Mondis	http://onto.mondis.cz/openrdf-sesame/repositories/mondis-record-owlim
9	Springer	http://lod.springer.com/sparql
10	Transparency International	http://transparency.270a.info/sparql

Tabelle 6.1.: Liste der für die Untersuchung der Performance ausgewählten SPARQL-Endpunkte

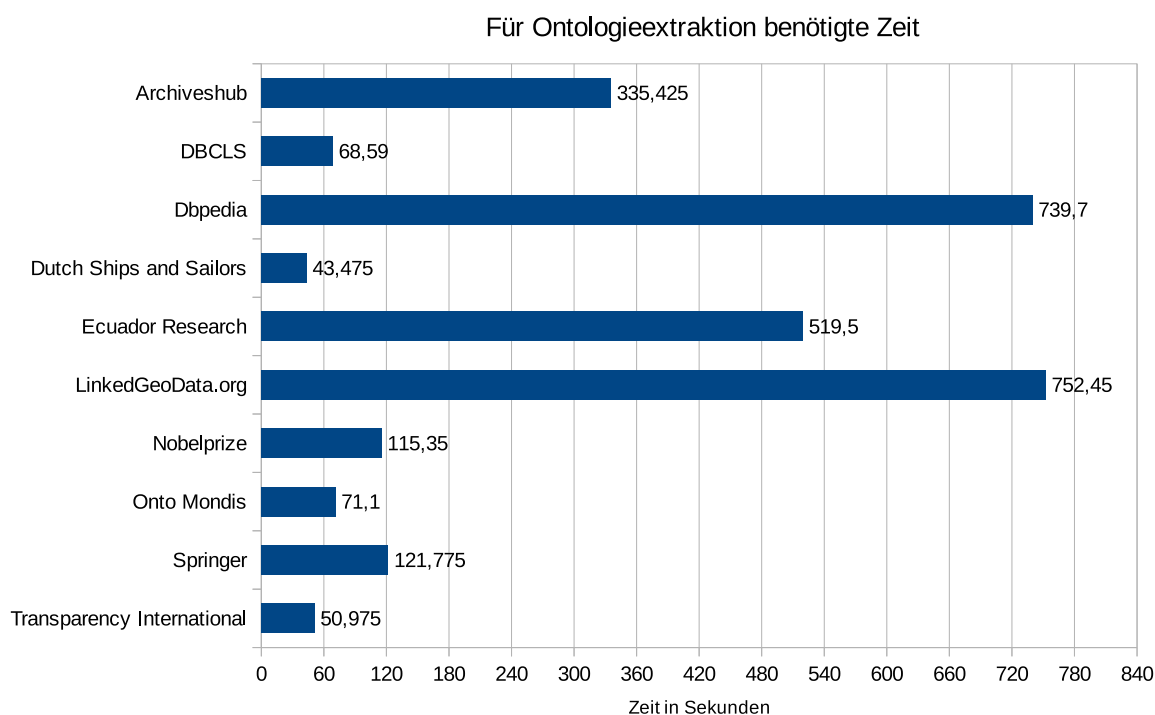


Abbildung 6.1.: Darstellung der für die vollständige Extraktion der Ontologie-Informationen benötigten Zeit nach Endpunkt

6. Evaluation

Allgemein fällt auf, dass der Fortschritt der Ontologie-Extraktion des Öfteren durch einige wenige SPARQL-Abfragen, deren Beantwortung durch den Endpunkt sehr lange dauert, aufgehalten wird. Besonders die letzten Abfragen benötigen meist viel Zeit und verlängern dadurch die Gesamtlaufzeit.

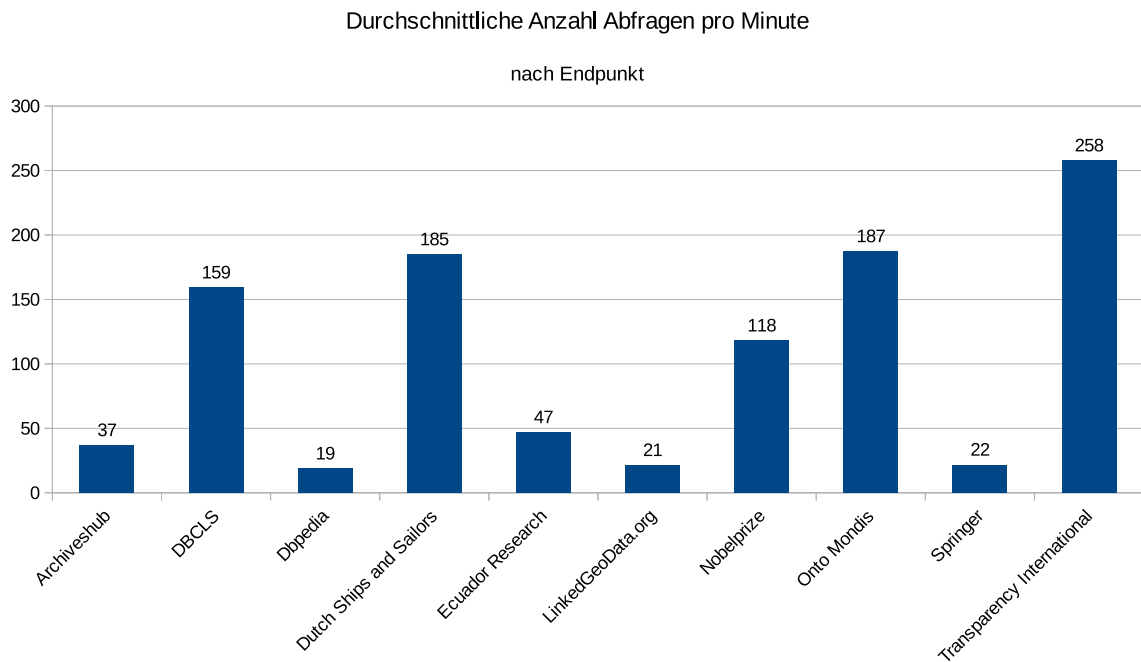


Abbildung 6.2.: Durchschnittliche Anzahl der von LDVOWL während der Extraktion gesendeten SPARQL-Abfragen pro Minute

Auf einigen Endpunkten wie z. B. AustrianSkiTeam werden mit fünf bis sechs Sekunden noch wesentlich geringere Laufzeiten für die Extraktion erreicht. Die kurzen Laufzeiten für diese Endpunkte werden für die Untersuchung der Performance nicht weiter berücksichtigt, da es wahrscheinlich ist, dass diese auf ein Caching der Ergebnisse seitens des Endpunkts zurückzuführen sind.

Abbildung 6.2 zeigt, dass die Anzahl der abgearbeiteten SPARQL-Abfragen pro Minute von Endpunkt zu Endpunkt stark variiert: Während von DBpedia durchschnittlich nur 19 SPARQL-Abfragen pro Minute beantwortet werden, liegt der Durchsatz für Abfragen auf kleinen Datensätzen wie etwa Transparency International mit 258 Abfragen pro Minute bei mehr als dem zehnfachen. Die Beantwortung der in diesem Ansatz verwendeten Abfragen ist für große Datensätze folglich sehr viel rechenintensiver als für kleine Datensätze.

6.8. Skalierbarkeit

Auch wenn der beschriebene Ansatz für große Datensätze wie z. B. DBpedia funktioniert, zeigen Nutzerstudie und Performance-Untersuchung dennoch, dass die Skalierbarkeit in mehrererlei Hinsicht beschränkt ist.

Sollen beispielsweise mehr als nur die zehn mächtigsten Klassen dargestellt werden, so dauert die Extraktion schnell sehr viel länger, da die Anzahl der benötigten SPARQL-Abfragen wie bereits erwähnt quadratisch mit der Anzahl der extrahierten Klassen wächst. Des Weiteren ist der Durchsatz für Endpunkte mit großen Datensätzen erheblich geringer als für kleine Datensätze.

Aber auch für kleinere Datensätze kann es zu Problemen kommen, sobald die Dichte des Graphen zunimmt und sich einem vollständigen Graphen annähert: Hier hat die Nutzerstudie gezeigt, dass die graphische Darstellung trotz der Zusammenfassung von Properties mit gleicher Ausgangs- und Zielmenge für den Benutzer schnell unübersichtlich wird.

7. Fazit und Ausblick

Der in dieser Arbeit vorgestellte Ansatz sowie dessen prototypische Umsetzung in der Webanwendung „LinkedDataVOWL“ zeigen, dass sich Ontologie-Informationen mittels generierter SPARQL-Abfragen innerhalb weniger Minuten aus LD-Datensätzen extrahieren lassen.

Diese Ontologie-Informationen können mit einer angepassten Form von VOWL visualisiert werden und geben dem Benutzer einen groben Überblick über die wichtigsten im untersuchten Datensatz enthaltenen Informationen sowie deren Struktur.

Mithilfe dieses Ansatzes erhält der Benutzer einen Einblick in den Datensatz und damit einen Ausgangspunkt für eine genauere Untersuchung. Außerdem erfährt er, welche Ontologien im Datensatz wiederverwendet werden und auf welche Weise dies geschieht.

Ausblick

Auch wenn die Ergebnisse vielversprechend sind, gibt es noch Potential zur Verfeinerung und Verbesserung des Ansatzes:

Die Extraktion der Ontologien lässt sich an mehreren Stellen noch erweitern, sodass weitere Ontologie-Informationen aus LD-Datensätzen gewonnen werden können. Eine Erkennung und Zusammenführung von inversen Properties wäre beispielsweise hilfreich. Außerdem könnte das Scheduling der SPARQL-Abfragen verbessert werden, sodass Abfragen mit kurzer Laufzeit nicht durch lang laufende Abfragen blockiert werden. Auch die zur Extraktion verwendeten Abfragen können hinsichtlich ihrer benötigten Ausführungszeit noch optimiert werden.

Seitens der Visualisierung hat die Nutzerstudie gezeigt, dass eine Hervorhebung der Beziehungen zwischen zwei oder mehr zuvor ausgewählten Klassen besonders für dichte Graphen wünschenswert ist. Auch weitere Interaktionsmöglichkeiten wie Pick-and-Pin für Knoten oder eine Suchfunktion für Klassen und Properties könnte die Benutzbarkeit der Anwendung verbessern.

A. Dokumente für Nutzerstudie

Für die Nutzerstudie wurden die folgenden Fragebögen und Dokumente verwendet. Um die Auswertung zu erleichtern wurden die Fragebögen mit Google Forms erstellt und von den Probanden direkt online ausgefüllt.



University of Stuttgart
Germany

Consent Form

DESCRIPTION: You are invited to participate in a **research study** on **the extraction and visualization of ontology information from linked data**.

TIME INVOLVEMENT: Your participation will take approximately **30 minutes**.

DATA COLLECTION: For this study you will explore different linked data sets using a web application and answer questions about them by filling out questionnaires.

RISKS AND BENEFITS: No risk associated with this study. The collected data is securely stored. We do guarantee no data misuse and privacy is completely preserved.

PARTICIPANT'S RIGHTS: If you have read this form and have decided to participate in this project, please understand your **participation is voluntary** and you have the **right to withdraw your consent or discontinue participation at any time without penalty or loss of benefits to which you are otherwise entitled**. **The alternative is not to participate**. You have the right to refuse to answer particular questions. The results of this research study may be presented at scientific or professional meetings or published in scientific journals. Your identity is not disclosed unless we directly inform and ask for your permission.

CONTACT INFORMATION: If you have any questions, concerns or complaints about this research, its procedures, risks and benefits, contact the following person:

Marc Weise (marc.weise@yahoo.de)

By signing this document I confirm that I agree to the terms and conditions.

Name: _____ Signature, Date: _____

LDVOWL Übersicht Notation

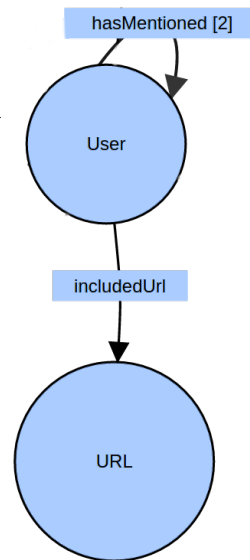
Klassen



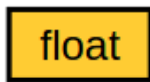
Klassen sind Mengen von Instanzen und werden als **Kreise** dargestellt. **Je mehr Instanzen** eine Klasse hat, **desto größer ist der Radius des Kreises**. Äquivalente Klassen werden als Kreis mit doppeltem Rand dargestellt. Kindklassen verweisen über einen gestrichelten Pfeil auf ihre Vaterklasse.

Properties

Properties zwischen den Instanzen zweier Klassen werden als **beschriftete Pfeile** dargestellt. Existieren mehrere verschiedene Properties zwischen den Instanzen zweier Klassen, so trägt der Pfeil den **Namen der häufigsten Property** und die **Anzahl verschiedener Properties** wird **in eckigen Klammern** an den Namen angehängt.



Datentypen



Datentypen wie z. B. Zahlen oder Zeitangaben werden als **gelbe Rechtecke mit schwarzem Rand** dargestellt. Properties, welche auf Datentypen verweisen haben eine Beschriftung mit grünem Hintergrund.

Namensräume

Die **Hintergrundfarbe von Klassen und Properties** weist auf den Namensraum des jeweiligen Elements hin. Die Bezeichnungen der Namensräume sind in der Sidebar aufgelistet.

Wichtigste Interaktionen

- **Zoomen** mit dem Mausrad
- **Knoten bewegen** per Drag-and-Drop
- vollständige Bezeichnungen und Details in Sidebar anzeigen durch **Auswählen per Mausklick**
- **Filterung der Darstellung** unter „Filters“ in der Sidebar
- **Anpassung der Darstellung** unter „Graph“ in der Sidebar

Demografische Daten

* Erforderlich

1. Geschlecht *

Markieren Sie nur ein Oval.

männlich

weiblich

2. Alter *

.....

3. Beruf *

.....

Vorkenntnisse

4. Ich weiß bereits etwas über Linked Data *

Markieren Sie nur ein Oval.

ja

nein

Bereitgestellt von



Endpunkt 1 - allgemeine Fragen

Machen Sie sich zunächst mit der Darstellung des Datensatzes vertraut und beantworten Sie anschließend die folgenden Fragen.

* Erforderlich

1. Um was geht es in diesem Datensatz? *

.....
.....
.....
.....
.....

2. Welches ist die Klasse mit den meisten Instanzen? (Bezeichnung oder letzter Teil der URI) *

.....

3. Wie viele verschiedene Namensräume (intern und extern) werden verwendet? *

.....

4. Wie heißt der Namensraum, welcher die meisten der angezeigten Klassen enthält? *

.....

Endpunkt 3 - spezifische Fragen

Versuchen Sie die folgenden Fragen mit Hilfe der Anwendung zu beantworten.

5. Welche Informationen sind zu einer Person hinterlegt? *

.....

6. Wie viele Preisträger (Laureate) sind im Datensatz erfasst? *

.....

A. Dokumente für Nutzerstudie

7. Welche Beziehung(en) bestehen zwischen Nobelpreisen (NobelPrize) und Auszeichnungen (Award)? *

.....

8. Über wie viele verschiedene Properties sind Personen und Städte verbunden? *

.....

9. Welche Property tritt zwischen Preisträgern (Laureate) und Ländern (Country) am häufigsten auf? *

.....

Bereitgestellt von



Endpunkt 2 - allgemeine Fragen

Machen Sie sich zunächst mit der Darstellung des Datensatzes vertraut und beantworten Sie anschließend die folgenden Fragen.

* Erforderlich

1. Um was geht es in diesem Datensatz? *

.....
.....
.....
.....
.....

2. Welches ist die Klasse mit den meisten Instanzen? (Bezeichnung oder letzter Teil der URI) *

.....

3. Wie viele verschiedene Namensräume (intern und extern) werden verwendet? *

.....

4. Wie heißt der Namensraum, welcher die meisten der angezeigten Klassen enthält? *

.....

Endpunkt 4 - spezifische Fragen

Versuchen Sie die folgenden Fragen mit Hilfe der Anwendung zu beantworten.

5. Welche Informationen sind zu einer Organisation hinterlegt? *

.....

6. Wie viele Unternehmen (Company) sind im Datensatz enthalten? *

.....

A. Dokumente für Nutzerstudie

7. Welche Beziehungen bestehen zwischen Personen? *

.....

8. Über wie viele verschiedene Properties sind lebende Personen und Unternehmen verbunden? *

.....

9. Welche Property tritt am häufigsten zwischen Unternehmen und Personen auf? *

.....

Bereitgestellt von



Feedback

Hier haben Sie die Möglichkeit ein generelles Feedback zur Anwendung abzugeben.

1. Was mir besonders gefallen hat...

.....
.....
.....
.....
.....

2. Was mich gestört hat...

.....
.....
.....
.....
.....

3. Verbesserungsvorschläge für die Anwendung oder verwendete Notation

.....
.....
.....
.....
.....

B. Begriffslexikon

Im Folgenden werden in dieser Arbeit verwendete Fachbegriffe und Akronyme kurz beschrieben oder auf den Abschnitt der Arbeit verwiesen, in welchem diese erläutert werden.

ABox bezeichnet die Instanzinformationen in einem LD-Datensatz.

BL steht für „Bisimulation Label“.

CORS steht für „Cross-Origin Resource Sharing“.

CSP steht für „Coarsest Stable Partition“.

CSS steht für „Cascading Style Sheets“. Diese werden für die Definition des Aussehens der HTML- und SVG-Elementen eingesetzt.

D3 steht für „Data Driven Documents“, ein in Javascript geschriebenes Visualisierungstoolkit [BOH11].

DAML steht für „DARPA Agent Markup Language“, und ist eine Auszeichnungssprache für Ontologien.

Eclipse ist eine Entwicklungsumgebung für Java und andere Programmiersprachen.

HTML steht für „Hypertext Markup Language“, eine Auszeichnungssprache für Webseiten.

HTTP steht für „Hypertext Transfer Protocol“.

IRI steht für „Internationalized Resource Identifier“.

JSON steht für „Javascript Object Notation“.

JSONP steht für „JSON with Padding“ und ist eine Übertragungsmethode für Daten im JSON Format.

LDVOWL steht für „LinkedDataVOWL“ und ist der Name der im Rahmen dieser Arbeit entwickelten Anwendung zur Extraktion von Ontologien aus Linked Data.

LOD steht für „Linked Open Data“ oder „Linking Open Data“.

OOP steht für „Object Oriented Programming“.

OWL steht für „Web Ontology Language“, siehe dazu Abschnitt 2.5.

Protégé ist ein Ontologie-Editor.

RDF steht für „Resource Description Framework“, siehe dazu Abschnitt 2.3.

RDFS steht für „Resource Description Framework Schema“, siehe dazu Abschnitt 2.4.

Reasoner sind Inferenzmaschinen, welche durch logisches Schließen aus einer Menge gegebenen Axiome und Schlussregeln folgende Regeln ableiten können.

SPARQL ist eine Abfragesprache für Graphmuster, siehe dazu Abschnitt 2.6.

SVG steht für „Scalable Vector Graphics“.

TBox bezeichnet die Typinformationen in einem LD-Datensatz.

URI steht für „Universal Resource Identifier“.

URL steht für „Uniform Resource Locator“.

VOWL steht für „Visual Notation for OWL“, siehe dazu Abschnitt 3.3.8.

XML steht für „Extensible Markup Language“ und ist eine Auszeichnungssprache für hierarchische Daten.

Literaturverzeichnis

- [Ala03] H. Alani. „TGVizTab: An ontology visualisation extension for Protégé“. In: *Knowledge Capture (K-Cap'03), Workshop on Visualization Information in Knowledge Engineering*. 2003. URL: <http://oro.open.ac.uk/20054/> (Zitiert auf S. 30).
- [BBP05] A. Bosca, D. Bonino und P. Pellegrino. „OntoSphere: more than a 3D ontology visualization tool“. In: *SWAP 2005 - Semantic Web Applications and Perspectives, Proceedings of the 2nd Italian Semantic Web Workshop, University of Trento, Trento, Italy, 14-16 December 2005*. Hrsg. von P. Bouquet und G. Tummarello. Bd. 166. CEUR Workshop Proceedings. CEUR-WS.org, 2005. URL: <http://www.ceur-ws.org/Vol-166/8.pdf> (Zitiert auf S. 31).
- [BHB09] C. Bizer, T. Heath und T. Berners-Lee. „Linked Data - The Story So Far“. In: *Int. J. Semantic Web Inf. Syst.* 5.3 (2009), S. 1–22. DOI: [10.4018/jswis.2009081901](https://doi.org/10.4018/jswis.2009081901). URL: <http://dx.doi.org/10.4018/jswis.2009081901> (Zitiert auf S. 13, 14).
- [BL06] T. Berners-Lee. *Linked Data Design Issues*. Juli 2006. URL: <http://www.w3.org/DesignIssues/LinkedData.html> (Zitiert auf S. 13).
- [BOH11] M. Bostock, V. Ogievetsky und J. Heer. „D³ Data-Driven Documents“. In: *IEEE Trans. Vis. Comput. Graph.* 17.12 (2011), S. 2301–2309. DOI: [10.1109/TVCG.2011.185](https://doi.org/10.1109/TVCG.2011.185). URL: <http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.185> (Zitiert auf S. 24, 33, 47, 87).
- [CMA12] D. V. Camarda, S. Mazzini und A. Antonuccio. „LodLive, exploring the web of data“. In: *I-SEMANTICS 2012 - 8th International Conference on Semantic Systems, I-SEMANTICS '12, Graz, Austria, September 5-7, 2012*. Hrsg. von V. Presutti und H. S. Pinto. ACM, 2012, S. 197–200. DOI: [10.1145/2362499.2362532](https://doi.org/10.1145/2362499.2362532). URL: <http://doi.acm.org/10.1145/2362499.2362532> (Zitiert auf S. 24).
- [DV+13] L. De Vocht, S. Softic, E. Mannens, R. Van de Walle und M. Ebner. „ResXplorer: Interactive Search for Relationships in Research Repositories“. In: *Semantic Web Challenge, part of the 12th International Semantic Web Conference*. 2013. URL: http://challenge.semanticweb.org/2013/submissions/swc2013_submission_14.pdf (Zitiert auf S. 23).
- [EB] P. Eklund und P. Becker. *OntoRama*. URL: <https://sourceforge.net/projects/ontorama/> (Zitiert auf S. 29).

- [ERG02] P. W. Eklund, N. Roberts und S. Green. „OntoRama: Browsing RDF Ontologies Using a Hyperbolic-Style Browser“. In: *1st International Symposium on Cyber Worlds (CW 2002), 6-8 November 2002, Tokyo, JAPAN*. IEEE Computer Society, 2002, S. 405–411. DOI: [10.1109/CW.2002.1180907](https://doi.org/10.1109/CW.2002.1180907). URL: <http://dx.doi.org/10.1109/CW.2002.1180907> (Zitiert auf S. 29).
- [Fal] S. Falconer. *OntoGraf - Protege Wiki*. URL: <http://protegewiki.stanford.edu/wiki/OntoGraf> (Zitiert auf S. 27).
- [GB14] R. Guha und D. Brickley. *RDF Schema 1.1*. W3C Recommendation. <http://www.w3.org/TR/2014/REC-rdf-schema-20140225/>. W3C, Feb. 2014. URL: <http://www.w3.org/TR/rdf-schema/> (Zitiert auf S. 15).
- [Haa+15] F. Haag, S. Lohmann, S. Siek und T. Ertl. „Visual Querying of Linked Data with QueryVOWL“. In: *Joint Proceedings of the 1st International Workshop on Summarizing and Presenting Entities and Ontologies and the 3rd International Workshop on Human Semantic Web Interfaces (SumPre 2015, HSWI 2015) co-located with the 12th Extended Semantic Web Conference (ESWC 2015), Portoroz, Slovenia, June 1, 2015*. Hrsg. von G. Cheng, K. Gunaratna, A. Thalhammer, H. Paulheim, M. Voigt und R. García. Bd. 1556. CEUR Workshop Proceedings. CEUR-WS.org, 2015. URL: <http://ceur-ws.org/Vol-1556/paper6.pdf> (Zitiert auf S. 35).
- [Hei+09] P. Heim, S. Hellmann, J. Lehmann, S. Lohmann und T. Stegemann. „RelFinder: Revealing Relationships in RDF Knowledge Bases“. In: *Semantic Multimedia, 4th International Conference on Semantic and Digital Media Technologies, SAMT 2009, Graz, Austria, December 2-4, 2009, Proceedings*. Hrsg. von T. Chua, Y. Kompatsiaris, B. Mérialdo, W. Haas, G. Thallinger und W. Bailer. Bd. 5887. Lecture Notes in Computer Science. Springer, 2009, S. 182–187. DOI: [10.1007/978-3-642-10543-2_21](https://doi.org/10.1007/978-3-642-10543-2_21). URL: http://dx.doi.org/10.1007/978-3-642-10543-2_21 (Zitiert auf S. 22).
- [Jam] *Jambalaya - Protege Wiki*. URL: <http://protegewiki.stanford.edu/wiki/Jambalaya> (Zitiert auf S. 26).
- [KC04] G. Klyne und J. Carroll. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation. <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>. W3C, Feb. 2004. URL: <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/> (Zitiert auf S. 14).
- [KC10] S. Khatchadourian und M. P. Consens. „ExpLOD: Summary-Based Exploration of Interlinking and RDF Usage in the Linked Open Data Cloud“. In: *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part II*. Hrsg. von L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral und T. Tudorache. Bd. 6089. Lecture Notes in Computer Science. Springer, 2010, S. 272–287. DOI: [10.1007/978-3-642-13489-0_19](https://doi.org/10.1007/978-3-642-13489-0_19). URL: http://dx.doi.org/10.1007/978-3-642-13489-0_19 (Zitiert auf S. 21).

- [LNB14] S. Lohmann, S. Negru und D. Bold. „The ProtégéVOWL Plugin: Ontology Visualization for Everyone“. In: *The Semantic Web: ESWC 2014 Satellite Events - ESWC 2014 Satellite Events, Anissaras, Crete, Greece, May 25-29, 2014, Revised Selected Papers*. 2014, S. 395–400. DOI: [10.1007/978-3-319-11955-7_55](https://doi.org/10.1007/978-3-319-11955-7_55). URL: http://dx.doi.org/10.1007/978-3-319-11955-7_55 (Zitiert auf S. 33).
- [Loh+14a] S. Lohmann, S. Negru, F. Haag und T. Ertl. „VOWL 2: User-Oriented Visualization of Ontologies“. In: *Knowledge Engineering and Knowledge Management - 19th International Conference, EKAW 2014, Linköping, Sweden, November 24-28, 2014. Proceedings*. 2014, S. 266–281. DOI: [10.1007/978-3-319-13704-9_21](https://doi.org/10.1007/978-3-319-13704-9_21). URL: http://dx.doi.org/10.1007/978-3-319-13704-9_21 (Zitiert auf S. 32).
- [Loh+14b] S. Lohmann, V. Link, E. Marbach und S. Negru. „WebVOWL: Web-based Visualization of Ontologies“. In: *Knowledge Engineering and Knowledge Management - EKAW 2014 Satellite Events, VISUAL, EKM1, and ARCOE-Logic, Linköping, Sweden, November 24-28, 2014. Revised Selected Papers*. 2014, S. 154–158. DOI: [10.1007/978-3-319-17966-7_21](https://doi.org/10.1007/978-3-319-17966-7_21). URL: http://dx.doi.org/10.1007/978-3-319-17966-7_21 (Zitiert auf S. 33).
- [Ont] *OntoSphere 3D - Screenshots*. URL: <http://ontosphere3d.sourceforge.net/screenshots.html> (Zitiert auf S. 31).
- [Owl] *OWL 2 Web Ontology Language Document Overview (Second Edition)*. URL: <http://www.w3.org/TR/2012/REC-owl2-overview-20121211/> (Zitiert auf S. 15).
- [Piea] E. Pietriga. *IsaViz: A Visual Authoring Tool for RDF*. URL: <http://www.w3.org/2001/11/IsaViz> (Zitiert auf S. 30).
- [Pieb] E. Pietriga. *IsaViz User Manual*. URL: <https://www.w3.org/2001/11/IsaViz/usermanual.html> (Zitiert auf S. 30).
- [PMd08] S. Peroni, E. Motta und M. d’Aquin. „Identifying Key Concepts in an Ontology, through the Integration of Cognitive Principles with Statistical and Topological Measures“. In: *The Semantic Web, 3rd Asian Semantic Web Conference, ASWC 2008, Bangkok, Thailand, December 8-11, 2008. Proceedings*. 2008, S. 242–256. DOI: [10.1007/978-3-540-89704-0_17](https://doi.org/10.1007/978-3-540-89704-0_17). URL: http://dx.doi.org/10.1007/978-3-540-89704-0_17 (Zitiert auf S. 19, 20).
- [Pre+11] V. Presutti, L. Aroyo, A. Adamou, B. A. C. Schopman, A. Gangemi und G. Schreiber. „Extracting Core Knowledge from Linked Data“. In: *Proceedings of the Second International Workshop on Consuming Linked Data (COLD2011), Bonn, Germany, October 23, 2011*. Hrsg. von O. Hartig, A. Harth und J. Sequeda. Bd. 782. CEUR Workshop Proceedings. CEUR-WS.org, 2011. URL: http://ceur-ws.org/Vol-782/PresuttiEtAl_COLD2011.pdf (Zitiert auf S. 20).

- [PS08] E. Prud'hommeaux und A. Seaborne. *SPARQL Query Language for RDF*. W3C Recommendation. <http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/>. W3C, Jan. 2008. URL: <http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/> (Zitiert auf S. 16).
- [Shn96] B. Shneiderman. „The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations“. In: *Proceedings of the 1996 IEEE Symposium on Visual Languages, Boulder, Colorado, USA, September 3-6, 1996*. IEEE Computer Society, 1996, S. 336–343. DOI: [10.1109/VL.1996.545307](https://doi.org/10.1109/VL.1996.545307). URL: <http://dx.doi.org/10.1109/VL.1996.545307> (Zitiert auf S. 50).
- [Sin] M. Sintek. *OntoViz*. URL: <http://protegewiki.stanford.edu/wiki/OntoViz> (Zitiert auf S. 28).
- [Sto+01] M.-A. Storey, M. Musen, J. Silva, C. Best, N. Ernst, R. Ferguson und N. Noy. „Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé“. In: *Workshop on Interactive Tools for Knowledge Capture (K-CAP-2001)*. Citeseer. 2001, S. 93 (Zitiert auf S. 26).
- [Sto+02] M. D. Storey, N. F. Noy, M. A. Musen, C. Best, R. W. Ferguson und N. A. Ernst. „Jambalaya: an interactive environment for exploring ontologies“. In: *IUI*. 2002, S. 239–239. DOI: [10.1145/502716.502778](https://doi.org/10.1145/502716.502778). URL: <http://doi.acm.org/10.1145/502716.502778> (Zitiert auf S. 26).
- [UG96] M. Uschold und M. Gruninger. „Ontologies: principles, methods and applications“. In: *Knowledge Eng. Review* 11.2 (1996). speziell middle-out Ansatz für den Ontologie Entwurf, S. 93–136. DOI: [10.1017/S0269888900007797](https://doi.org/10.1017/S0269888900007797). URL: <http://dx.doi.org/10.1017/S0269888900007797> (Zitiert auf S. 14).

Alle URLs wurden zuletzt am 23. 03. 2016 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift