Empirically Analyzing Knowledgebases Using Sparql-Queries - Guthub

August 1, 2022

Empirically Analyzing Knowledgebases Using Sparql-Queries

Universität Rostock

Fakultät für Informatik und Elektrotechnik

Institut für Informatik

Lehrstuhl für Wirtschaftsinformatik

Michael Doß

Maximilian Niese

1 Inhaltsverzeichnis

2 Einleitung

2.1 Motivation

Um den Austausch von strukturiertem Wissen zwischen menschlichen und computergestützten Akteuren zu ermöglichen, werden Wissensdatenbaken benötigt. Assistenzsysteme, wie beispielsweise "Amazon Alexa" oder "Siri", nutzen sehr wahrscheinlich strukturiertes Wissen, aus solchen Wissensdatenbanken, um Fragen wie "Wie viele Einwohner hat die Stadt Rostock?" beantworten zu können. Weil Wissensdatenbanken eine enorme Größe besitzen ist es oft nicht praktikabel, diese als Ganzes herunterzuladen und eine Analyse durchzuführen. Eine Qualitätskontrolle, ist somit keine einfache Aufgabe. Allerdings bieten sie, in den meisten Fällen, einen standardisierten Endpunkt für die Abfrage an: Die graphenbasierte Sprache SPARQL.

2.2 Zielstellung

Die Arbeit soll die folgenden Forschungsfragen beantworten:

Forschungsfrage 1:

Welche Ansätze gibt es, um Wissensbasen zu messen?

Es soll während der Literaturrecherche (siehe Methodik) untersucht werden, welche Möglichkeiten bestehen, um Aussagen über die Qualität von Wissensdatenbanken treffen zu können.

Forschungsfrage 2:

Wie können wir öffentliche Knowledgebases insbesondere mit SPARQL bewerten?

Aus dem Grund, wie in der Motivation zuvor beschrieben, besitzen diese meist einen standardisierten SPARQL-Endpunkt. Über diesen sollen verschiedene Anfragen an ausgewählte Wissensdatenbanken gestellt werden. Aus den Ergebnissen dieses Experimentes können schließlich ebenfalls Aussagen über die Qualität, der jeweiligen Wissensbasis, getroffen werden. Das Ziel besteht darin Qualitätsindikatoren aufzustellen. Ein Vergleich und eine Bewertung der genutzten Wissensdatenbanken wird damit ebenfalls ermöglicht. Die, für dieses Experiment ausgewählten öffentlichen Wissensbasen sind: DBPedia, WikiData, GovData, Europe, LinkedWiki und Uriburner.

3 Methodik

Im Folgenden, soll grob der Ablauf, der bibliografischen Onlinerecherche in der Literatur, geschildert werden. Hierzu wurde zuerst mit einer Stichwortsuche mittels Google Scholar gestartet. Ziel war es ein geeignetes Vokabular zu identifizieren, um später eine möglichst hohe „Precision“ zu erreichen. Herausgebildet haben sich dabei folgende Begriffe:

- knowledgebase
- sparql
- process
- system
- evaluation
- perfomance
- benchmark

An dieser Stelle, muss erwähnt werden, dass SPARQL als Anfragesprache, zwar im Fokus dieser Arbeit steht, jedoch bei der Stichwortsuche nicht verwendet wurde, um anderen Ansätzen gegenüber offen zu bleiben. Es hat sich aber bei der Recherche als der “Goldweg zum Ziel” erwiesen und ist in so gut wie jeder Arbeit, zu diesem Thema, zu Anwendung gekommen. Dadurch wurde im nachfolgenden für die iterative Literaturrecherche über Scopus SPARQL in den Query fest integriert. Die detallierte Übersicht, des Vorgehens bei der Suche, ist in Anlage A Iterative Suche Scopus zu finden. Die Auswertung der Treffer der Suche brachte leider nicht viel, für diese Arbeit verwendbares, Material hervor. So kam der zuerst ausgewertete Query (siehe Iterative Suche I Ergebnisse Query Nr 7) auf insgesamt 19 Treffer, von denen 8 relevant waren.

```
TITLE-ABS-KEY ( knowledgebase  AND  sparql  AND
( process*  OR  method*  OR  technique*  OR  system*  OR  evaluation
OR  performance  OR  benchmark  OR  procedure* ) )  AND  PUBYEAR  >  2016
```

Diese führte dazu, dass der Query weiterentwickelt werden musste. Der finale Query (siehe Iterative Suche Scopus) aus den oben aufgezählten Schlagworten, sieht wie folgt aus:

```
TITLE-ABS ((knowledgebase OR "linked open data" OR "semantic web")
AND sparql AND rdf* AND (technique* OR system* OR performance)
AND (evaluat* OR qualit* OR benchmark)) AND PUBYEAR > 2016
```

Im Rahmen dieser Suche wurden, Stand 06.07.2022, insgesamt 81 Dokumente durch Scopus ermittelt (siehe Iterative Suche II Ergebnisse Query Nr 17). Nach Auswertung des Titels sowie des Abstracts waren 20 Dokumente relevant. Da jedoch die Ergebnisse der Litereturrecherche sich immer auf spezielle Fälle beziehen, wie z.B. eine vom Autor selber entwickelte Knowledgebase, konnten die dort verwendeten Abfragen nicht für einen allgemeinen Vergleich von verschiedenen Knowledgebases, wie “dbpedia”, “wikidata”, “govdata”, “europa”, “LinkedWiki” oder “Uriburner” verwendet werden. Aus dem Grund, dass es sich um domainspezifische Knowledgebases handelte, wurden auch keine weiteren Ansätze, welche eine Allgemeingültigkeit besitzen, für die Bewertung der Qualität gefunden.

4 Definition

Im folgenden Abschnitt, soll kurz erklärt werden, was eine ***Knowlegdebase*** (deu: Wissensdatenbank) sowie ***SPARQL***, ***RDF***, ***RDFS*** und ***OWL*** sind und wofür es steht.

Knowledgebase

Für die betrachtete und untersuchte Wissensdatenbank, existiert keine einheitliche Definition. Eine Möglichkeit für die Definierung wäre: *„Wissensdatenbanken ermöglichen den Austausch von strukturiertem Wissen zwischen menschlichen und computergestützten Akteuren“* Eine andere wäre, nach dem Wirtschaftslexikon Gabler: *„Menge des in einem wissensbasierten System gespeicherten problemspezifischen Wissens.“* (4)

Resource Describtion Framework (RDF)

“RDF ist ein gerichtetes, beschriftetes Graphenformat, welches aus RDF-Tripeln besteht. Dieses setzt sich zusammen aus Subjekt, Prädikat sowie Objekt. Es dient der Darstellung von Informationen im Web und bildet somit die Grundlage für das Semantik Web. Die Syntax, als auch die Semantik der Abfragesprache SPARQL wird zudem durch RDF definiert.” (1) Abfragen, welche sich auf RDF, wie die Triple gerichtet sind, befinden sich in Abschnitt RDF.

Resource Describtion Framework Schema (RDFS)

“Bei RDF-Schema handelt es sich um eine semantische Erweiterung von RDF. Mechanismen für die Beschreibung von Gruppen mit verwandten Ressourcen und deren Beziehungen.” (2) Where-Clauses des Query, werden in Abschnitt RDFS gestellt.

Ontologie Web Language (OWL)

“OWL ist eine Sprache des Semantik Web, mit welcher sich umfangreiches sowie komplexes Wissen über Dinge, Gruppen von Dingen und Beziehungen zwischen Dingen darstellen lässt.” (3) SPARQL-Anfragen, welche OWL-Anfragen stellen, sind in Abschnitt OWL zu finden.

SPARQL

“Ist eine graphenbasierte Abfragesprache, welche in dieser Arbeit genutzt wird. Sie kann verwendet werden, um Abfragen über verschiedene Datenquellen auszudrücken. Dabei enthält sie Fähigkeiten erforderliche und optionale Graphenmuster gemeinsam mit ihren Konjunktionen und Disjunktionen abzufragen.” (1)

Definitionsquellen:

(1) W3C® (15.01.2008): SPARQL Query Language for RDF. URL: https://www.w3.org/TR/rdf-sparql-query/#basicpatterns [31.07.2022]

(2) W3C® (25.02.2014): RDF Schema 1.1. URL: https://www.w3.org/TR/rdf-schema/ [31.07.2022]

(3) W3C® (11.12.2012): OWL. URL: https://www.w3.org/OWL/ [31.07.2022]

(4) Prof. Dr. Lackes, Richard & Dr. Siepermann, Markus (19.02.2018): Wissensbasis. URL: https://wirtschaftslexikon.gabler.de/definition/wissensbasis-49358/version-272594 [31.07.2022]

5 Bezugsrahmen

Für die später folgenden Abfragen in SPARQL, wurde zunächts eine Abfragebasis geschaffen. Diese besteht aus immer wieder auszuführende Python-Funktionen. So enthält die Funktion “query” die Links zu den jeweiligen SPARQL Endpunkten, die Funktion “printbarlog1” und “printbarlog2” die Vorgehensweise zur graphischen und “table” zur tabellarischen Aufbereitung der Ergebnisse. Dadurch muss im folgenden nur noch die “serviceLabels”, ein “Titel” der SPARQL-Anfrage, die gewünschte “graphische Aufbereitung” und natürlich die “SPARQL-Anfrage” angegeben werden.

Abfragebasis: Queries und Knowledgebases

[ ]:
```
#!pip install pandas
#!pip install SPARQLWrapper
#!pip install seaborn
#!pip install matplotlib

import pandas as pd
from SPARQLWrapper import SPARQLWrapper2
import seaborn as sns, json, numpy
import matplotlib.pyplot as plt
import requests

services = ["https://dbpedia.org/sparql", "https://query.wikidata.org/sparql",
            "https://www.govdata.de//sparql",
            "https://publications.europa.eu/webapi/rdf/sparql",
            "https://linkedwiki.com/sparql","https://linkeddata.uriburner.com/
 ↪sparql/"]
serviceLabels = ["DBPedia",␣
 ↪"WikiData","GovData","Europe","LinkedWiki","Uriburner"]
services1 = ["https://dbpedia.org/sparql", "https://www.govdata.de//sparql",
             "https://publications.europa.eu/webapi/rdf/sparql",
             "https://linkedwiki.com/sparql","https://linkeddata.uriburner.com/
 ↪sparql/"]
serviceLabels1 = ["DBPedia","GovData","Europe","LinkedWiki","Uriburner"]

def query(query: str, services):
    resultList = []
    for service in (services):
        sparql = SPARQLWrapper2(service)
        sparql.setQuery(query)
        sparql.setTimeout(300) # Especially WikiData is very slow. Maybe first␣
 ↪test the queries somewhere else.
        try:
            res = sparql.query()
            if(len(res.bindings)> 0):
                resultList.append(int(res.bindings[0]["result"].value))
            else:
                resultList.append(-1)
```

```
        except json.JSONDecodeError: # If the value is None, the library rasies␣
↪a JSONDecodeError. In that case, append a -1
            resultList.append(-1)
    return resultList
```

Da die Ergebnis der Anfragen, auf die verschiedenen Knowledgebases, starke Unterschiede in der Ergebnismenge aufweisen, findet die graphische Auswertung in den Balkendiagramme, sowohl mit der absoluten Menge als auch mittels des dekadischen Logarithmus statt. (Anmerkung: Wenn die Ergebnismenge "0" beträgt, wird diese bei der logarithmischen Auswertung in "1" geändert, da der log10(0) = mathematisch nicht definiert, jedoch der log10(1) = 0 ist.)

Verglichen wird im folgenden die zurückgegebenen Treffermengen der Queries, durch die abgefragten Knowledgebases. Durch die Queries selbst, werden verschiedene logische Beziehungen zwischen den Elementen (z.B. Klassen oder Instanzen) abgefragt, um einen quanitativen Vergleich, in den Taxonomien und Ontologien, ziehen zu können.

Graphische und tabellarische Aufbereitung

[ ]:
```
def printbarlog1(list,titel,labels):
    list2 = [1 if x==0 else x for x in list]
    list2 = numpy.log10(list2)
    fig, ax = plt.subplots(figsize=(8, 4))
    for tick in ax.get_xticklabels():
        tick.set_rotation(0)
    ax.bar(labels, list, color="blue")
    ax.set_title(titel)
    ax.set_ylabel("Anzahl")
    fig, ax = plt.subplots(figsize=(8, 4))
    for tick in ax.get_xticklabels():
        tick.set_rotation(0)
    ax.bar(labels, list2, color="red")
    ax.set_title(titel+" log10(x)")
    ax.set_ylabel("Anzahl log10(x)")

def printbarlog2(list,titel,labels):
    list2 = [1 if x==0 else x for x in list]
    list2 = numpy.log10(list2)
    fig, ax = plt.subplots(1,2, figsize=(18, 4))
    for tick in ax[0].get_xticklabels():
        tick.set_rotation(0)
    ax[0].bar(labels, list, color="blue")
    ax[0].set_title(titel)
    ax[0].set_ylabel("Anzahl")
    for tick in ax[1].get_xticklabels():
        tick.set_rotation(0)
    ax[1].bar(labels, list2, color="red")
    ax[1].set_title(titel+" log10(x)")
    ax[1].set_ylabel("Anzahl log10(x)")
```

```
def table(list,serviceLabels,titel):
    fig, ax = plt.subplots(figsize=(10, 1)) # Diagramm und Achsen definieren
    table_data=[serviceLabels,list] # Werte für Tabelle erstellen
    table = ax.table(cellText=table_data, loc='center') #Tabelle erstellen
    table.set_fontsize(14) # Tabelle ändern
    table.scale(2,2), ax.axis('off')
    ax.set_title(titel)
    plt.show() #Tabelle anzeigen

def log(list):
    list2 = [1 if x==0 else x for x in list]
    list2 = numpy.log10(list2)
    return list2
```

6 Ergebnisse

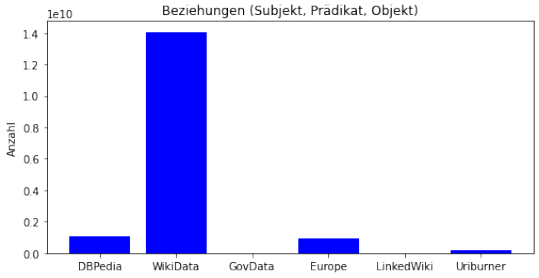
Im folgenden Abschnitt sollen die Ergebnisse der Anfragen vorgestellt und ausgewertet werden. Wie bereits im Kapitel Methodik (siehe Methodik) erläutert wurde, gab es in der Literatur keine allgemein anwendbaren Queries. Daher mussten diese neu entwickelt werden. Eine zeitliche Betrachung in den Anfragen wurde dabei außenvor gelassen (siehe Erläuterung Evaluation). Betrachtet und ausgewertet werden in dieser Arbeit nur die ABox, mit den dort enthaltenen Wissen um Individuen, Axiomen und konkreten Situationen (Daten). Die TBOx, mit dem terminilogisches Wissen und dem konzeptionellen Schema werden im Folgenden nicht weiter berücksichtigt.

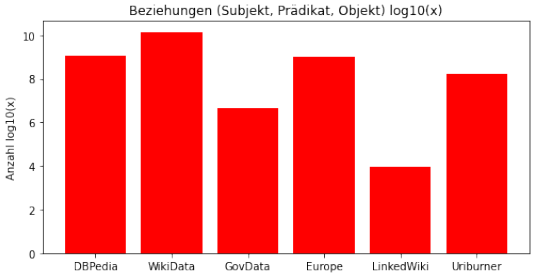
6.1 RDF

1. Beziehungen (Subjekt, Prädikat, Objekt)

Der folgende Query fragt die Subjekt-Prädikat-Objekt-Beziehung ab. Dabei wird gezählt, wie oft das RDF-Tripel (T(s,p,o), Ressource + Property + Wert (Literal oder Ressource)) oder der gerichtete Graph vorkommt. Schließen könnte man, an diser Stelle, aus der zugegebenen Antwort, auf die Größe der Knowledgebases.

[ ]:
```
a = query( """
    SELECT (COUNT (*) AS ?result)
        WHERE { ?sub ?pred ?obj }
    Limit 100""", services)
titel = ("Beziehungen (Subjekt, Prädikat, Objekt)")
printbarlog2(a,titel,serviceLabels)
table(a,serviceLabels,titel)
a.insert(0,"?sub ?pred ?obj")
```





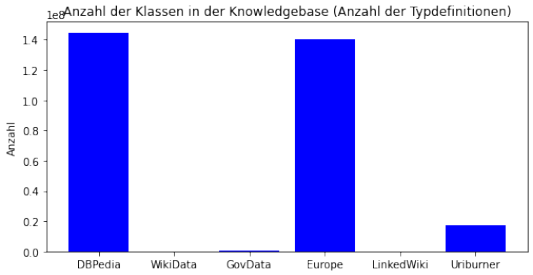
Beziehungen (Subjekt, Prädikat, Objekt)

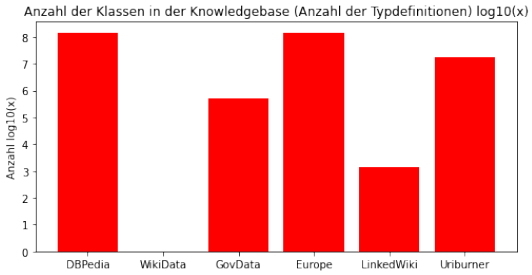
DBPedia	WikiData	GovData	Europe	LinkedWiki	Uriburner
1104871471	14067837578	4593931	968373591	9566	171376651

2. Anzahl der Klassen in der Knowledgebase

Dieser Query fragt die Anzahl der definierten Klassen (Anzahl der Typdefinitionen) in der Knowledgebase ab. Die Abfrage der Klassen, kann leider nur mit 5 von 6 Knowledgebases erfolgen (siehe "services1" der Abfragebasis), da die Abfrage in Wikidata keine Ergebnisse liefert, sondern folgenden Fehler: "EndPointInternalError: EndPointInternalError: The endpoint returned the HTTP status code 500" bzw. "Zeitüberschreitungsgrenze erreicht". Dieser Fehler erscheint ebenfalls bei der Abfrage direkt über den SPARQL-Endpoint von Wikidata (https://query.wikidata.org/).

[ ]:
```
b = query( """
    SELECT  (COUNT(?s) AS ?result)
      WHERE{ ?s a ?class }
    Limit 100""",services1)
titel = ("Anzahl der Klassen in der Knowledgebase (Anzahl der Typdefinitionen)")
b.insert(1,0)
printbarlog2(b,titel,serviceLabels)
b.remove(0),b.insert(1,"timeout")
table(b,serviceLabels,titel)
b.insert(0,"?s a ?class")
```





Anzahl der Klassen in der Knowledgebase (Anzahl der Typdefinitionen)

DBPedia	WikiData	GovData	Europe	LinkedWiki	Uriburner
144690227	timeout	507523	140221526	1384	17008109

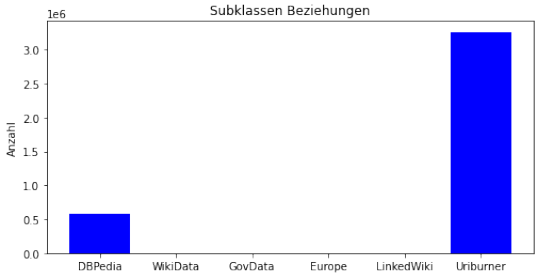
6.2 RDFS

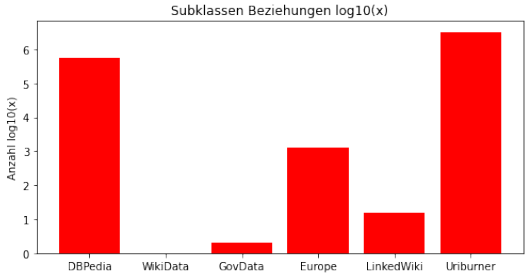
In den nun gestesteten Queries, wird RDFS (RDF-Schema) verwendet. Dies spezifiziert das Datenmodell, indem RDF Statements entworfen werden können. Erlaubt sind zusätzlich abstrakte Datentypen, Klassendefinitionen und -instanzierungen , hierarchische Klassenodelle und Vererbung (Subklassen, Superklassen) sowie das Festlegen von Eigenschaften und Restriktionen (Properties). Um dieses Vokabular verwenden zu können, muss in den Queries das "PREFIX rdfs" mit angegeben werden, welches auf w3.org verweißt.

3. Subklassen Beziehungen

Dieser Query fragt die möglichen Subklassen- und Klasseneziehungen in den Ontologien der Knowledgebases ab sowie die transitiven Eigenschaften zur Festlegung von Vererbungshierarchien von Klassen.

[ ]:
```
c = query( """
   PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        SELECT  (count(?c) AS ?result)
            WHERE { ?c rdfs:subClassOf ?b }
        Limit 100""",services)
titel = ("Subklassen Beziehungen")
printbarlog2(c,titel,serviceLabels)
table(c,serviceLabels,titel)
c.insert(0,"rdfs:subClassOf")
```





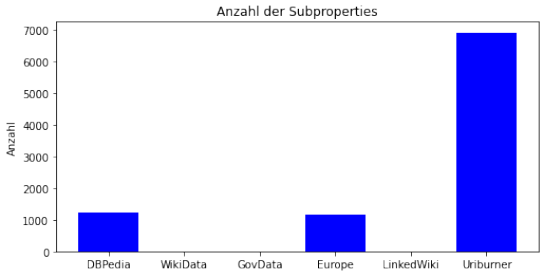
Subklassen Beziehungen

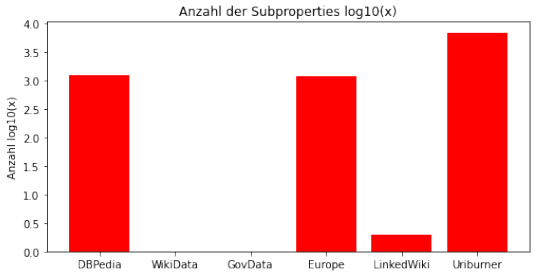
DBPedia	WikiData	GovData	Europe	LinkedWiki	Uriburner
572512	0	2	1244	15	3263879

4. Anzahl der Subproperties

Ähnlich wie Nr. 3 fragt dieser Query die Anzahl der Subproperties, sprich die transitiven Eigenschaften zur Festlegung von Vererbungshierarchien von Eigenschaften, ab.

[ ]:
```
d = query( """
    PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        SELECT (count(distinct (?c)) AS ?result)
            WHERE { ?c rdfs:subPropertyOf ?d }
        Limit 100""",services)
titel = ("Anzahl der Subproperties")
printbarlog2(d,titel,serviceLabels)
table(d,serviceLabels,titel)
d.insert(0,"rdfs:subPropertyOf")
```





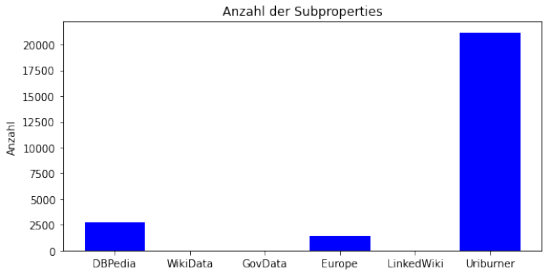
Anzahl der Subproperties

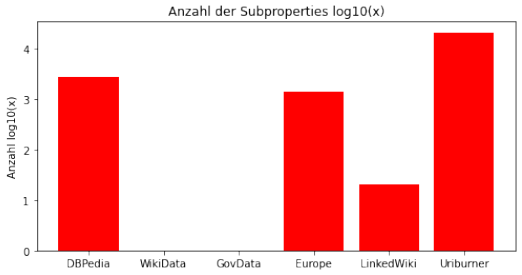
DBPedia	WikiData	GovData	Europe	LinkedWiki	Uriburner
1238	0	0	1168	2	6919

5. Anwendungsbereiche von Eigenschaften auf Klassen

Mit “rdfs:domain” werden Anwendungsbereiche von Eigenschaften in Bezug auf Klassen definiert. Der folgende Query fragt die Anzahl der Subjekte einer Klassenerweiterung, die zu einer bestimmten Klassenbeschreibung gehören ab.

[ ]:
```
e = query( """
    PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        SELECT (count(distinct (?c)) AS ?result)
            WHERE { ?c rdfs:domain ?d }
        Limit 100""",services)
titel = ("Anzahl der Subproperties")
printbarlog2(e,titel,serviceLabels)
table(e,serviceLabels,titel)
e.insert(0,"rdfs:domain")
```





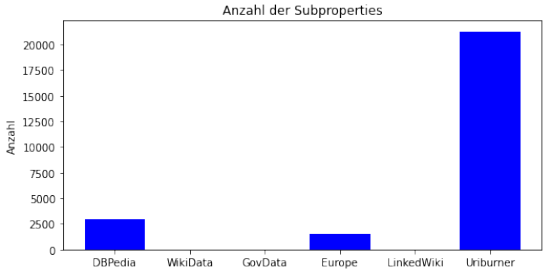
Anzahl der Subproperties

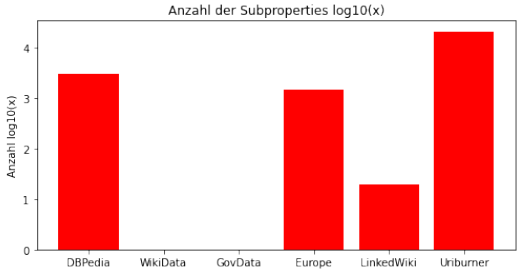
DBPedia	WikiData	GovData	Europe	LinkedWiki	Uriburner
2755	0	0	1423	21	21167

6. Wertebereiche von Eigenschaften

Mit Hilfe von “rdfs:range” kann der Wertebereich einer Eigenschaft festgelegt werden. Die Anzahl dieser Definitionen von Klassenerweiterung oder Datenwerten, die zu einem Wertebereich gehören, werden hier ausgewertet.

[ ]:
```
f = query( """
    PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        SELECT (count(distinct (?c)) AS ?result)
            WHERE { ?c rdfs:range ?d }
        Limit 100""",services)
titel = ("Anzahl der Subproperties")
printbarlog2(f,titel,serviceLabels)
table(f,serviceLabels,titel)
f.insert(0,"rdfs:range")
```





Anzahl der Subproperties

DBPedia	WikiData	GovData	Europe	LinkedWiki	Uriburner
2989	0	1	1481	20	21257

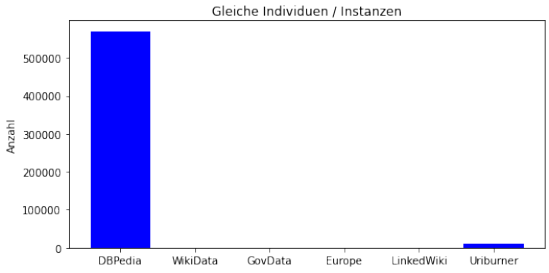
6.3 OWL

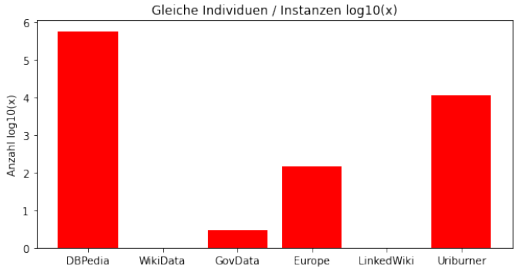
In diesem Abschnitt werden Queries verwendet, die OWL-Relationen verwenden. Mit OWL kommen nach RDFS (siehe RDFS) noch die Möglichkeiten hinzu Logiken (Konjuktionen UND (owl:intersectionOf), Disjunktionen ODER (owl:unionOf), Negationen (owl:complementOf)), Regeln (ungleich (owl:disjointwith) oder gleich (owl:equivalentClass)) sowie Constraints (z.B. Kardinalitäten) aufzustellen, da OWL auf RDF und RDFS aufbaut und eine Erweiterung zur formalen Beschreibung von Ontologien ist. Dem zufolge wird nun als “PREFIX” OWL mit dem Verweis auf w3.org mitangegeben.

7. Klassenerweiterungen enthalten genau dieselbe Menge von Individuen

Dieser Query vergleicht zwei Klassen miteinander auf Gleichheit und gibt die Häuhigkeit dieses Vorkommens aus. Dazu werden die Mengen von Individuen, in den Klassenerweiterungen, auf Gleichheit verglichen.

[ ]:
```
g = query( """
    PREFIX owl: <http://www.w3.org/2002/07/owl#>
        SELECT  (count(?c) AS ?result)
            WHERE { ?c owl:equivalentClass ?b }
        Limit 100""",services)
titel = ("Gleiche Individuen / Instanzen")
printbarlog2(g,titel,serviceLabels)
table(g,serviceLabels,titel)
g.insert(0,"owl:equivalentClass")
```





Gleiche Individuen / Instanzen

DBPedia	WikiData	GovData	Europe	LinkedWiki	Uriburner
570276	0	3	142	0	10968

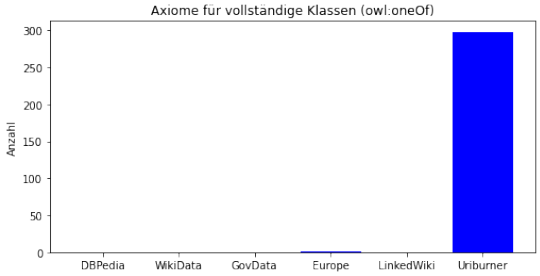
8.-11. Logische Beziehungen

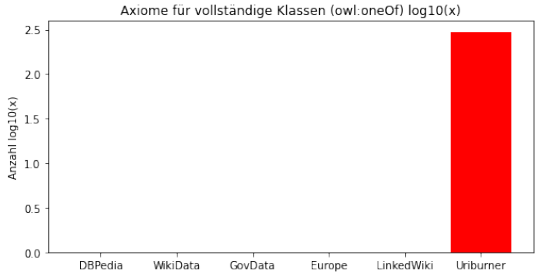
Die folgenden Queries, sind in einem Block zusammengefasst, da sie logische Klassenkonstruktoren bzw. Axiome für vollständige oder geschlossene Klassen (ohne Verwendung von owl:equivalentClass Gleichheit) sind. Die Bedeutung der logischen Konstruktoren sind die hier ausgelisteten:

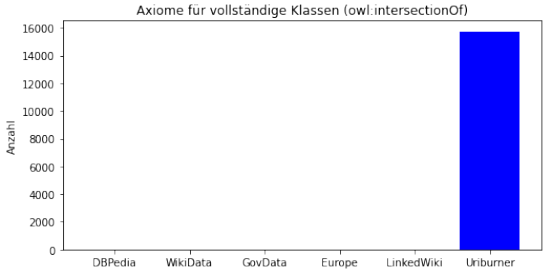
- owl:oneOf (Teilbeziehung)
- owl:intersectionOf (Konjunktion UND)
- owl:unionOf (Disjunktion ODER)
- owl:complementOf (Komplement)

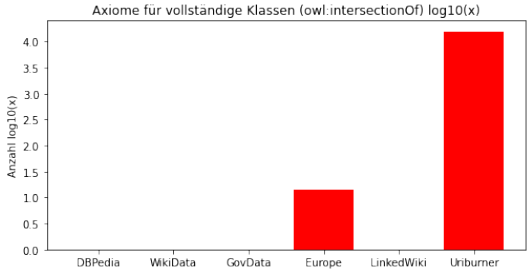
Somit wird verglichen wie viele Klassen existieren, die Überschneidungen haben, disjunkt sind, Teil- oder auch Nichtbeziehungen zueinander haben.

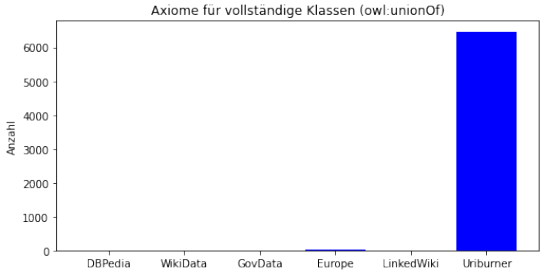
[ ]:
```
h = query( """
    PREFIX owl: <http://www.w3.org/2002/07/owl#>
        SELECT  (count(?c) AS ?result)
            WHERE { ?c owl:oneOf ?b }
        Limit 100""",services)
titel = ("Axiome für vollständige Klassen (owl:oneOf)")
printbarlog2(h,titel,serviceLabels)
h.insert(0,"owl:oneOf")

i = query( """
    PREFIX owl: <http://www.w3.org/2002/07/owl#>
        SELECT  (count(?c) AS ?result)
            WHERE { ?c owl:intersectionOf ?b }
        Limit 100""",services)
titel = ("Axiome für vollständige Klassen (owl:intersectionOf)")
printbarlog2(i,titel,serviceLabels)
i.insert(0,"owl:intersectionOf")

j = query( """
    PREFIX owl: <http://www.w3.org/2002/07/owl#>
        SELECT  (count(?c) AS ?result)
            WHERE { ?c owl:unionOf ?b }
        Limit 100""",services)
titel = ("Axiome für vollständige Klassen (owl:unionOf)")
printbarlog2(j,titel,serviceLabels)
j.insert(0,"owl:unionOf")

k = query( """
    PREFIX owl: <http://www.w3.org/2002/07/owl#>
        SELECT  (count(?c) AS ?result)
            WHERE { ?c owl:complementOf ?b }
        Limit 100""",services)
titel = ("Axiome für vollständige Klassen (owl:complementOf)")
printbarlog2(k,titel,serviceLabels)
k.insert(0,"owl:complementOf")

serviceLabels.insert(0,"Query")
fig, ax = plt.subplots(figsize=(10, 3)) # Diagramm und Achsen definieren
table_data=[serviceLabels,h,i,j,k] # Werte für Tabelle erstellen
table = ax.table(cellText=table_data, loc='center') #Tabelle erstellen
table.set_fontsize(14) # Tabelle ändern
table.scale(2,2), ax.axis('off')
ax.set_title("Axiome für vollständige Klassen ohne Verwendung von Gleichheit")
plt.show() #Tabelle anzeigen
serviceLabels.remove("Query")
```

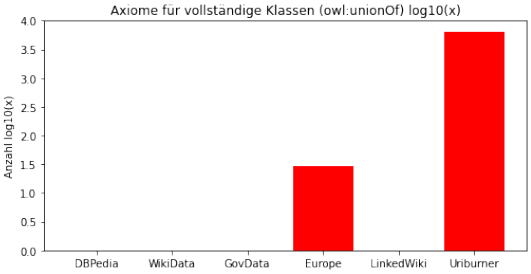

Axiome für vollständige Klassen (owl:oneOf)
Anzahl
DBPedia
WikiData
GovData
Europe
LinkedWiki
Uriburner

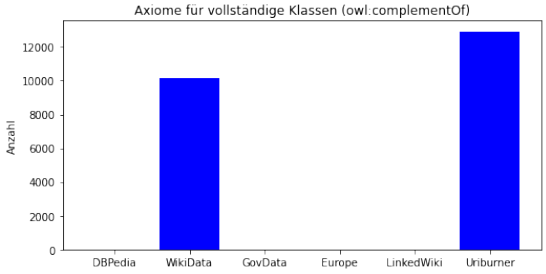

Axiome für vollständige Klassen (owl:oneOf) log10(x)
Anzahl log10(x)
DBPedia
WikiData
GovData
Europe
LinkedWiki
Uriburner

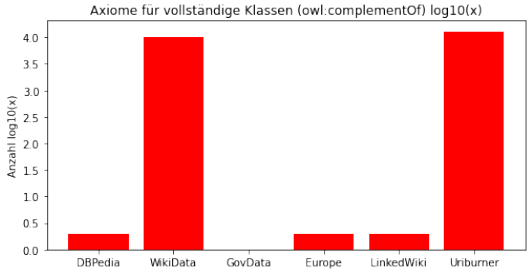

Axiome für vollständige Klassen (owl:intersectionOf)
Anzahl
DBPedia
WikiData
GovData
Europe
LinkedWiki
Uriburner


Axiome für vollständige Klassen (owl:intersectionOf) log10(x)
Anzahl log10(x)
DBPedia
WikiData
GovData
Europe
LinkedWiki
Uriburner


Axiome für vollständige Klassen (owl:unionOf)
Anzahl
DBPedia
WikiData
GovData
Europe
LinkedWiki
Uriburner


Axiome für vollständige Klassen (owl:unionOf) log10(x)
Anzahl log10(x)
DBPedia
WikiData
GovData
Europe
LinkedWiki
Uriburner


Axiome für vollständige Klassen (owl:complementOf)
Anzahl
DBPedia
WikiData
GovData
Europe
LinkedWiki
Uriburner


Axiome für vollständige Klassen (owl:complementOf) log10(x)
Anzahl log10(x)
DBPedia
WikiData
GovData
Europe
LinkedWiki
Uriburner

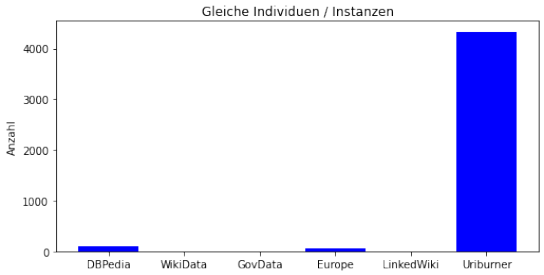
Axiome für vollständige Klassen ohne Verwendung von Gleichheit

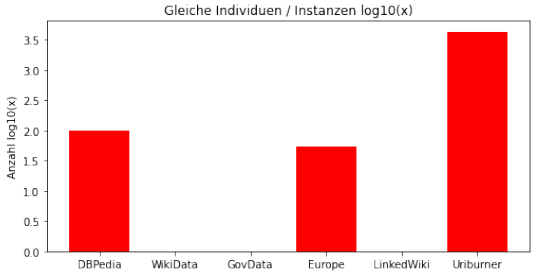
Query	DBPedia	WikiData	GovData	Europe	LinkedWiki	Uriburner
owl:oneOf	0	0	0	1	0	298
owl:intersectionOf	0	0	0	14	0	15737
owl:unionOf	1	0	0	29	1	6466
owl:complementOf	2	10180	0	2	2	12913

12. Nicht Beziehungen

Dieser Query ermittelt die Anzahl der Klassen oder Klassenerweiterungen, die keine Individuen gemeinsam haben (disjunkt sind) bzw. für welche eine "Ist-Nicht"-Beziehung definiert wurde.

[ ]:
```
l = query( """
    PREFIX owl: <http://www.w3.org/2002/07/owl#>
        SELECT  (count(?c) AS ?result)
            WHERE { ?c owl:disjointWith ?b }
        Limit 100""",services)
titel = ("Gleiche Individuen / Instanzen")
printbarlog2(l,titel,serviceLabels)
table(l,serviceLabels,titel)
l.insert(0,"owl:disjointWith")
```





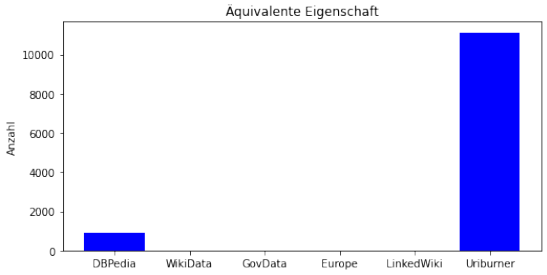
Gleiche Individuen / Instanzen

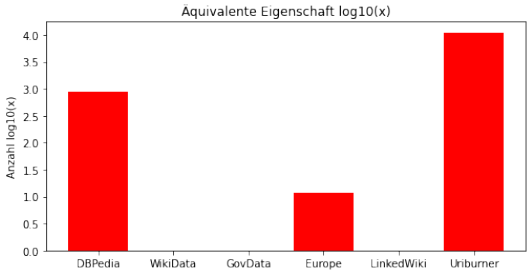
DBPedia	WikiData	GovData	Europe	LinkedWiki	Uriburner
100	0	1	54	0	4331

13. Gleiche Eigenschaftserweiterungen

Mit Hilfe von owl:equivalentProperty, wird erfragt wie viele Eigenschaften die gleiche Eigenschaftserweiterung haben.

[ ]:
```
m = query( """
    PREFIX owl: <http://www.w3.org/2002/07/owl#>
        SELECT  (count(?c) AS ?result)
            WHERE { ?c owl:equivalentProperty ?b }
        Limit 100""",services)
titel = ("Äquivalente Eigenschaft")
printbarlog2(m,titel,serviceLabels)
table(m,serviceLabels,titel)
m.insert(0,"owl:equivalentProperty")
```





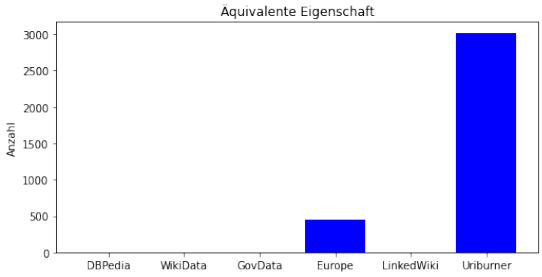
Äquivalente Eigenschaft

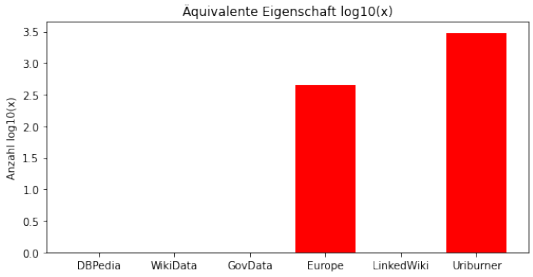
DBPedia	WikiData	GovData	Europe	LinkedWiki	Uriburner
879	0	0	12	0	11124

14. Inverse Beziehungen

Mit dem letzten Query, dieses Abschnittes, wird die Anzahl von inverse Beziehungen ermittelt, die zwischen Eigenschaften der Knowledgebase bestehen bzw. definiert wurden.

[ ]:
```
n = query( """
    PREFIX owl: <http://www.w3.org/2002/07/owl#>
        SELECT  (count(?c) AS ?result)
            WHERE { ?c  owl:inverseOf ?b }
        Limit 100""",services)
titel = ("Äquivalente Eigenschaft")
printbarlog2(n,titel,serviceLabels)
table(n,serviceLabels,titel)
n.insert(0,"owl:inverseOf")
```





Äquivalente Eigenschaft

DBPedia	WikiData	GovData	Europe	LinkedWiki	Uriburner
0	0	0	456	0	3021

7 Evaluation

Die Dimension der Zeit wurde in diesem Experiment, für die Bewertung nicht gewählt, da angefragte Queries im Cache zwischengespeichert werden, worduch ein Ergebnis direkt erhalten wird. Weitere Möglichkeiten, die das Ergebnis aus verschiedenen weiteren Gründen verfälschen können sind, z.B. die verwendete Hard- und Software, die Tageszeit in der ausgewählte Wissensdatenbank zum Anfragezeitpunkt stark genutzt wird oder die Möglichkeit von Problemen bei der Internetverbindung.

Eine Erkenntnis, die aus der Literaturrecherche (siehe Abschnitt Methodik) gewonnen werden konnte, ist die, dass alle dort erläuterten und vorgestellten Knowledgebases ein spezifisches, auf den jeweiligen Anwendungsfall angepasstes, Vokabular verwenden. Das macht ein Verwendung der Abfragen in anderen Knowledgebases unmöglich, da die Beziehungen oder Vokabeln dort anders definiert oder gar nicht bekannt sind. Jedoch die allgeinen Queries dieser Art, die auf w3.org verweisen, können in den speziellen Knowledgebases verwendet werden. Weiterhin lässt sich daraus ableiten, dass es nicht die eine Definition, sondern eine Vielzahl an Möglichkeiten dazu existiert (siehe RDFS und OWL).

Aus den Ergebnissen der Queries 1 und 2, die nur RDF benutzen, kann durch die Ergebnismenge, auf die Größe der der einzelnen Knowledgebases geschlossen werden. So kann man durch die quantive Auswertung sagen, dass WikiData die größte und LinkedWiki die kleinste der abgefragten Knowledgebases ist. Dies lässt aber keine Aussage über die Qualität zu, denn es kann durch aus vorkommen, dass die selbe Klasse mehrmals angelegt wurde, mit unterschiedlichen Namen in den verschiedenen Ontologien. Weiterhin können aber auch Synonyme und Homonyme die Auswertung verzerren.

An den Queries 3-6 kann man sehr schön sehen, dass nicht jede Knowledgebase, überhaupt oder im gleichen Maße, RDFS verwendet. Hier kann von den Betreibern eventuell nur ein vorgegebenes Vokabular erlaubt sein. Weitere in denen es offensicht erlaubt ist, aber nicht viel verwendet wird sind z.B. GovData oder LinkedWiki. Um an dieser Stelle dazu ein qulifizierte Aussage treffen zu können, müssen weitere Abfragen, auf Basis von RDFS, gemacht bzw. in den Beschreibungen der einzelen Knowledgebases nachgelesen werden.

Das Bild was bereits RDFS geliefert hat, spiegelt sich in OWL, in den Queries 7-12, wider. Denn die Knowledgebases, die RDFS nur wenig Verwenden, tun dies auch mit OWL. So wurden die logischen Beziehungen zwar von allen verwendet, jedoch mit Ausnahme von Uriburner und Europe (verwenden alle getesteten) nur selten in der Anzahl und auch nur sehr weniger der logischen Konstruktoren. WikiData verwendet nur “owl:complemetOf”, GovDate nur “owl:equivatentClass” (3x) und “owl:disjointWith” (1x) und LinkedWiki nur “owl:union” (1x) und “owl:complementOf” (2x). Dies kann aber, wie unter RDFS eine ganz bewusste Entscheidung, in den Knowledgebases sein.

Neben denen, im Abschnitt Ergebnisse betrachteten SPAQL-Anfragen, wurden noch weitere getestet. Diese haben jedoch für alle Knowledgebases als Ergebnismenge den Wert 0 geliefert. Getestet wurden dabei:

- owl:InverseFunctionalProperty
- owl:FunctionalProperty
- owl:AllDifferentFrom
- owl:AllDifferent
- owl:ObjectProperty
- owl:DatatypeProperty
- owl:SymmetricProperty
- owl:TransitiveProperty
- owl:distinctMembers
- owl:AnnotationProperty
- owl:Class

Daraus lässt sich schließen, dass diese logischen Verlinkungen (OWL-Referenzen) in der Definition, von keiner der betrachteten Knowledgebases verwendet wird.

Eine Aussage auf die Qualität, kann durch OWL getroffen werden. Da in der Wissensrepräsentation, dass formale Logiksystem der "Open-World-Annahme" Anwendung findet, in der man davon ausgeht, dass ein Aussage wahr sein kann, unabhängig davon, ob sie als wahr bekannt ist oder nicht. Um diese jedoch ordentlich Umsetzen zu können, müssen in jeder Ontologie ein Vielzahl von Definitinen durchgeführt werden wie z.B. "owl:intersektion" um zu beschreiben, dass die Eigenschaften gleich sind oder z.B. durch "owl:owl:disjointWith" um auszusagen das zwei Elemente nicht das Gleiche sind, wovon in der Open-World-Annahme zunächst ausgegangen werden könnte. Daher müssten die logischen Verbindungen in unglaublich großer Anzahl vorkommen. Diese Arbeit steht aber oftmals in keinen Aufwand-Nutzen-Verhältnis.

8 Fazit

Wie die Abschnitte Ergebnisse und Evaluation zeigen, existieren keine Vorgaben, wie eine Knowledgebase aufgebaut sein muss. Der Verfasser hat, an dieser Stelle alle Freiheit und entscheidet selbst welche Konstruktoren aus RDF, RDFS oder OWL er verwendet. Daher kann es keinen standardisietes Vorgehen geben, um Knowledgebases zu analysieren und die Qualität dieser zu bewerten wenn es unzälige Möglichkeiten gibt sie zu definieren und aufzubauen. Daher kann oft nur der Vergelich über die verscheiden Indikatoren durchgeführen werden, um einen indirekten Vergleich der Qualität ziehen zu können. Eine große Variable, die ebenfalls nicht eingeschrätzt werden kann, ist die verwendetet Soft- und Hardware am Backend, die einen großen Einfluss auf die Abfragegeschwindigkeit hat und deshalb hier nicht weiter berücksichtigt wurde.

Limitation und Ausbllick

In einem weiteren Schritt, der an diese Arbeit folgen kann, könnten weitere Knowledgebases und auch Abfragekonstruktoren getestet werden, um diese Ergebnisse weiter zu bekräftigen. Eine Auflistung, von möglichen weiteren Knowledgebases sind im Internet unter *https://www.wikidata.org/wiki/Wikidata:Lists/SPARQL_endpoints* oder *https://www.w3.org/wiki/SparqlEndpoints* zu finden. Aber auch eine genaue Betrachtung der Ontologien und Taxonomien, innerhalb der Knowledgebases, wären denkbar. Dazu würde ihr logischer Aufbau, als auch die Beziehungen getestet werden, wie es bereits in Grundzügen in Abschnitt OWL **8.-11. Logische Beziehungen** begonnen wurde. Sollte jedoch eine zeitlich Leistungsbewertung der Knowledgebases durchgeführt werden, muss die verwendete Technik, die Tageszeit sowie die Instratuktur berücksichtigt werden.

9 Anlage A Zusammenfassung Ergebnisse

[ ]:
```
serviceLabels.insert(0,"Query")
fig, ax = plt.subplots(figsize=(12, 10)) # Diagramm und Achsen definieren
rdf = (['RDF',None,None,None,None,None,None])
rdfs = (['RDFS',None,None,None,None,None,None])
owl = (['OWL',None,None,None,None,None,None])
table_data=[serviceLabels,rdf,a,b,rdfs,c,d,e,f,owl,g,h,i,j,k,l,m,n] # Werte für␣
 ↪Tabelle erstellen
table = ax.table(cellText=table_data, loc='center') #Tabelle erstellen
table.set_fontsize(14) # Tabelle ändern
table.scale(2,2), ax.axis('off')
ax.set_title("Zusammenfassung Ergebnisse")
plt.show() #Tabelle anzeigen
serviceLabels.remove("Query")
```

Zusammenfassung Ergebnisse

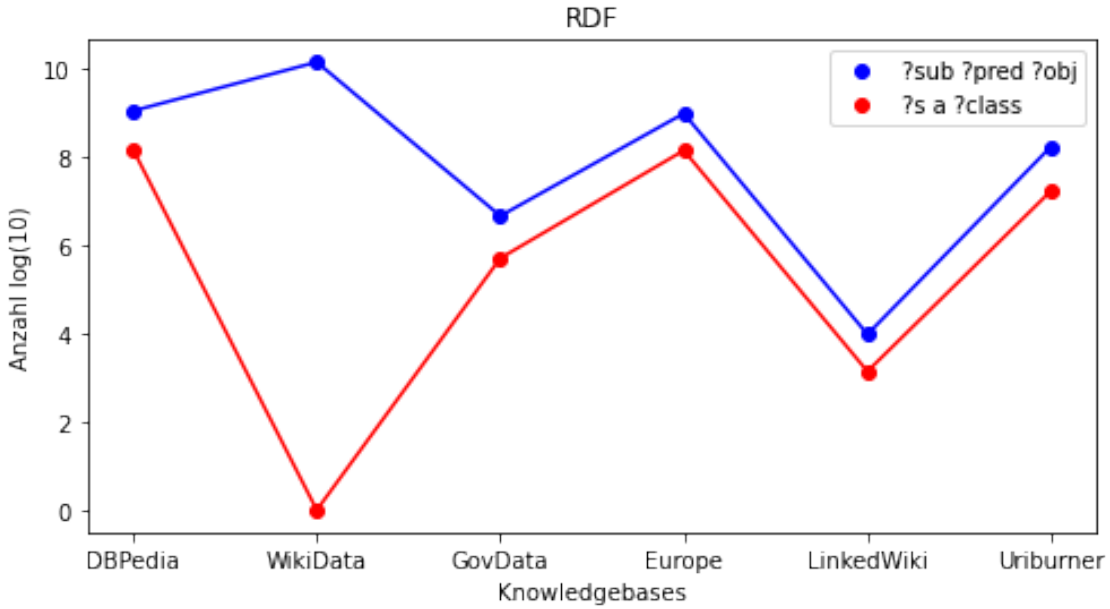
Query	DBPedia	WikiData	GovData	Europe	LinkedWiki	Uriburner
RDF						
?sub ?pred ?obj	1104871471	14067837578	4593931	968373591	9566	171376651
?s a ?class	144690227	timeout	507523	140221526	1384	17008109
RDFS						
rdfs:subClassOf	572512	0	2	1244	15	3263879
rdfs:subPropertyOf	1238	0	0	1168	2	6919
rdfs:domain	2755	0	0	1423	21	21167
rdfs:range	2989	0	1	1481	20	21257
OWL						
owl:equivalentClass	570276	0	3	142	0	10968
owl:oneOf	0	0	0	1	0	298
owl:intersectionOf	0	0	0	14	0	15737
owl:unionOf	1	0	0	29	1	6466
owl:complementOf	2	10180	0	2	2	12913
owl:disjointWith	100	0	1	54	0	4331
owl:equivalentProperty	879	0	0	12	0	11124
owl:inverseOf	0	0	0	456	0	3021

```
[ ]: serviceLabels.insert(0,"Query")
     x = [rdf,a,b,rdfs,c,d,e,f,owl,g,h,i,j,k,l,m,n]
     df = pd.DataFrame(x,columns= [serviceLabels])
     derived_df1 = df[['Query','DBPedia','WikiData','GovData']]
     derived_df2 = df[['Query','Europe','LinkedWiki','Uriburner']]
     print(derived_df1, "\n"),print(derived_df2)
     serviceLabels.remove("Query")
```

	Query	DBPedia	WikiData	GovData
0	RDF	NaN	None	NaN
1	?sub ?pred ?obj	1.104871e+09	14067837578	4593931.0
2	?s a ?class	1.446902e+08	timeout	507523.0
3	RDFS	NaN	None	NaN
4	rdfs:subClassOf	5.725120e+05	0	2.0
5	rdfs:subPropertyOf	1.238000e+03	0	0.0
6	rdfs:domain	2.755000e+03	0	0.0
7	rdfs:range	2.989000e+03	0	1.0
8	OWL	NaN	None	NaN
9	owl:equivalentClass	5.702760e+05	0	3.0
10	owl:oneOf	0.000000e+00	0	0.0
11	owl:intersectionOf	0.000000e+00	0	0.0
12	owl:unionOf	1.000000e+00	0	0.0
13	owl:complementOf	2.000000e+00	10180	0.0
14	owl:disjointWith	1.000000e+02	0	1.0
15	owl:equivalentProperty	8.790000e+02	0	0.0
16	owl:inverseOf	0.000000e+00	0	0.0

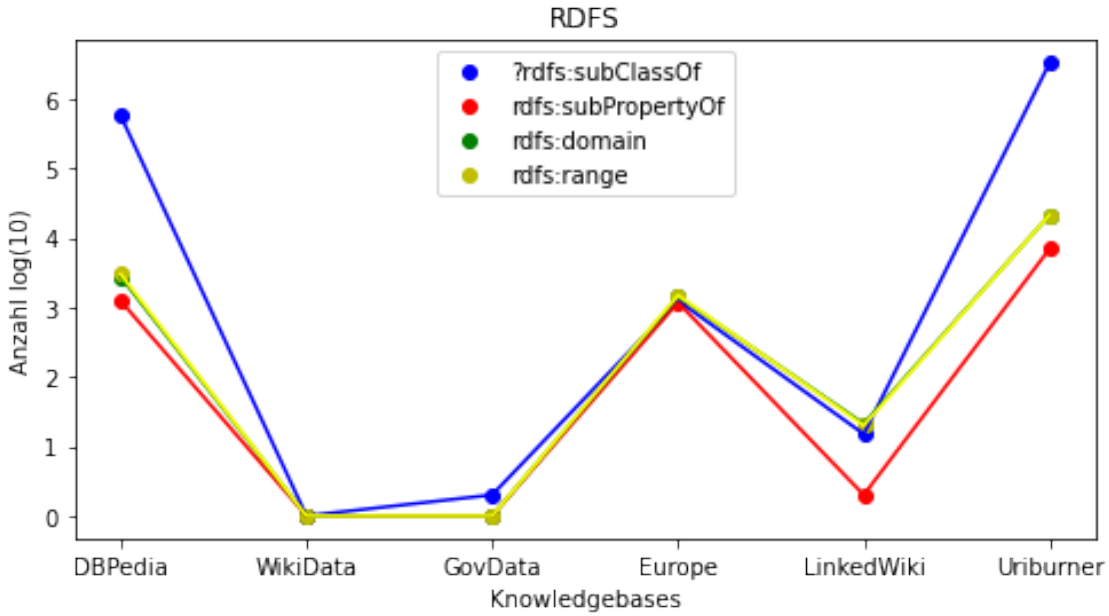
	Query	Europe	LinkedWiki	Uriburner
0	RDF	NaN	NaN	NaN
1	?sub ?pred ?obj	968373591.0	9566.0	171376651.0
2	?s a ?class	140221526.0	1384.0	17008109.0
3	RDFS	NaN	NaN	NaN
4	rdfs:subClassOf	1244.0	15.0	3263879.0
5	rdfs:subPropertyOf	1168.0	2.0	6919.0
6	rdfs:domain	1423.0	21.0	21167.0
7	rdfs:range	1481.0	20.0	21257.0
8	OWL	NaN	NaN	NaN
9	owl:equivalentClass	142.0	0.0	10968.0
10	owl:oneOf	1.0	0.0	298.0
11	owl:intersectionOf	14.0	0.0	15737.0
12	owl:unionOf	29.0	1.0	6466.0
13	owl:complementOf	2.0	2.0	12913.0
14	owl:disjointWith	54.0	0.0	4331.0
15	owl:equivalentProperty	12.0	0.0	11124.0
16	owl:inverseOf	456.0	0.0	3021.0

```
[ ]: a.remove("?sub ?pred ?obj")
     b.remove("?s a ?class"), b.remove("timeout"), b.insert(1,0)

     plt.figure(figsize=(8,4))
     plt.xlabel('Knowledgebases')
     plt.ylabel('Anzahl log(10)')
     plt.title("RDF")
     plt.plot(serviceLabels, log(a), "ob", label="?sub ?pred ?obj")
     plt.plot(serviceLabels, log(b), "or", label="?s a ?class")
     plt.legend()
     plt.plot(serviceLabels, log(a), color = "blue")
     plt.plot(serviceLabels, log(b), color = "red")
     plt.show()

     b.remove(0), b.insert(0,"?s a ?class"), b.insert(2,"timeout")
     a.insert(0,"?sub ?pred ?obj")
```

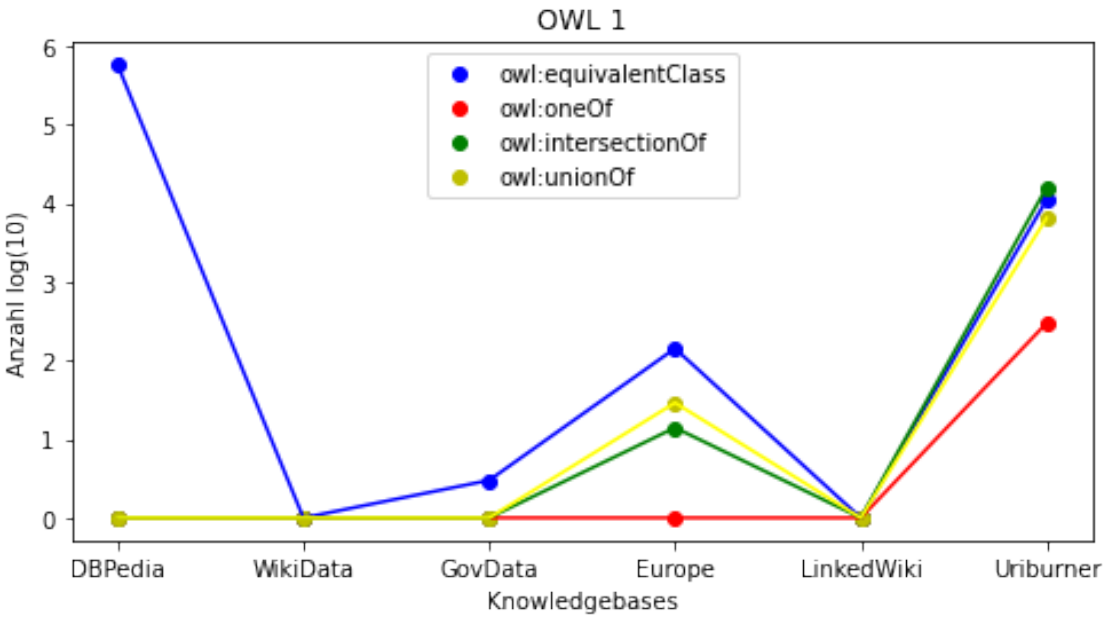


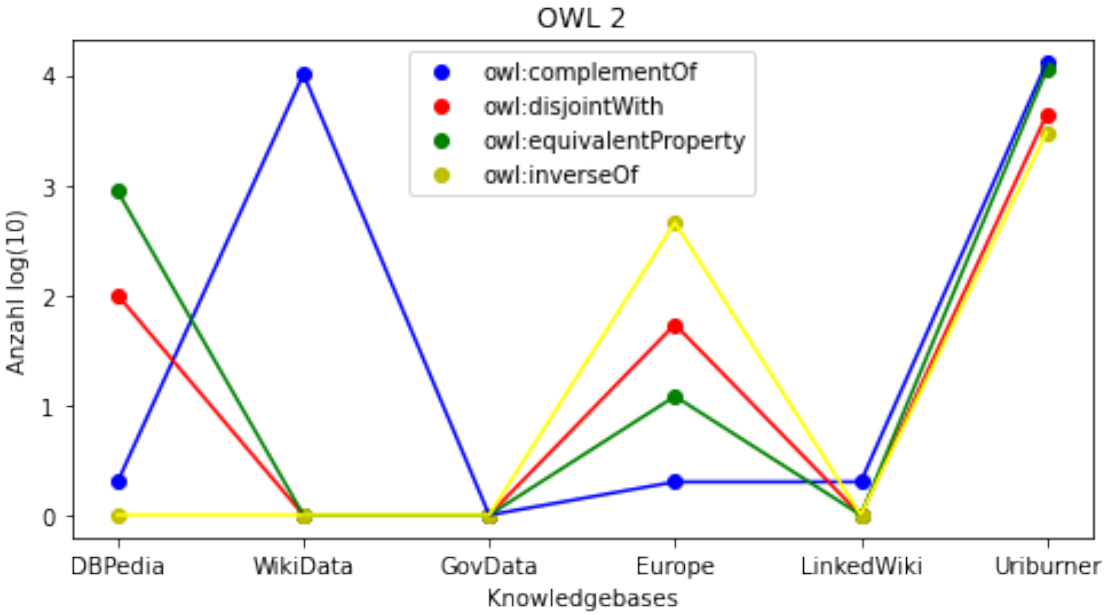

```
[ ]: c.remove("rdfs:subClassOf"), d.remove("rdfs:subPropertyOf")
     e.remove("rdfs:domain"), f.remove("rdfs:range")

     plt.figure(figsize=(8,4))
     plt.xlabel('Knowledgebases')
     plt.ylabel('Anzahl log(10)')
     plt.title("RDFS")
     plt.plot(serviceLabels, log(c), "ob", label="?rdfs:subClassOf")
     plt.plot(serviceLabels, log(d), "or", label="rdfs:subPropertyOf")
```

```
plt.plot(serviceLabels, log(e), "og", label="rdfs:domain")
plt.plot(serviceLabels, log(f), "oy", label="rdfs:range")
plt.legend()
plt.plot(serviceLabels, log(c), color = "blue")
plt.plot(serviceLabels, log(d), color = "red")
plt.plot(serviceLabels, log(e), color = "green")
plt.plot(serviceLabels, log(f), color = "yellow")

c.insert(0,"rdfs:subClassOf"), d.insert(0,"rdfs:subPropertyOf")
e.insert(0,"rdfs:domain"), f.insert(0,"rdfs:range")

plt.show()
```



[ ]:
```
g.remove("owl:equivalentClass"), h.remove("owl:oneOf")
i.remove("owl:intersectionOf"), j.remove("owl:unionOf")
k.remove("owl:complementOf"), l.remove("owl:disjointWith")
m.remove("owl:equivalentProperty"), n.remove("owl:inverseOf")

plt.figure(figsize=(8,4))
plt.xlabel('Knowledgebases')
plt.ylabel('Anzahl log(10)')
plt.title("OWL 1")
plt.plot(serviceLabels, log(g), "ob", label="owl:equivalentClass")
plt.plot(serviceLabels, log(h), "or", label="owl:oneOf")
plt.plot(serviceLabels, log(i), "og", label="owl:intersectionOf")
plt.plot(serviceLabels, log(j), "oy", label="owl:unionOf")
plt.legend()
plt.plot(serviceLabels, log(g), color = "blue")
plt.plot(serviceLabels, log(h), color = "red")
plt.plot(serviceLabels, log(i), color = "green")
plt.plot(serviceLabels, log(j), color = "yellow")
plt.show()

plt.figure(figsize=(8,4))
plt.xlabel('Knowledgebases')
plt.ylabel('Anzahl log(10)')
plt.title("OWL 2")
plt.plot(serviceLabels, log(k), "ob", label="owl:complementOf")
plt.plot(serviceLabels, log(l), "or", label="owl:disjointWith")
plt.plot(serviceLabels, log(m), "og", label="owl:equivalentProperty")
plt.plot(serviceLabels, log(n), "oy", label="owl:inverseOf")
plt.legend()
plt.plot(serviceLabels, log(k), color = "blue")
plt.plot(serviceLabels, log(l), color = "red")
plt.plot(serviceLabels, log(m), color = "green")
plt.plot(serviceLabels, log(n), color = "yellow")

g.insert(0,"owl:equivalentClass"), h.insert(0,"owl:oneOf")
i.insert(0,"owl:intersectionOf"), j.insert(0,"owl:unionOf")
k.insert(0,"owl:complementOf"), l.insert(0,"owl:disjointWith")
m.insert(0,"owl:equivalentProperty"), n.insert(0,"owl:inverseOf")

plt.show()
```


OWL 1
owl:equivalentClass
owl:oneOf
owl:intersectionOf
owl:unionOf
Anzahl log(10)
0
1
2
3
4
5
6
DBPedia
WikiData
GovData
Europe
LinkedWiki
Uriburner
Knowledgebases


OWL 2
owl:complementOf
owl:disjointWith
owl:equivalentProperty
owl:inverseOf
Anzahl log(10)
0
1
2
3
4
DBPedia
WikiData
GovData
Europe
LinkedWiki
Uriburner
Knowledgebases

10 Anlage B Literaturrecherche

10.1 Iterative Suche Scopus

Nr.	Query	Treffer
1	TITLE-ABS (knowledgebase AND (process OR method*))	972
2	TITLE-ABS (knowledgebase AND (process OR method*)) AND PUBYEAR > 2017	318
3	TITLE-ABS (knowledgebase AND sparql AND (process OR method*))	11
4	TITLE-ABS (knowledgebase AND sparql AND (process* OR method* OR technique* OR system*))	15
5	TITLE-ABS-KEY (knowledgebase AND sparql AND (process* OR method* OR technique* OR system*))	29
6	TITLE-ABS-KEY (knowledgebase AND sparql AND (process* OR method* OR technique* OR system* OR evaluation))	30
7	TITLE-ABS-KEY (knowledgebase AND sparql AND (process* OR method* OR technique* OR system* OR evaluation OR performance OR benchmark OR procedure*)) AND PUBYEAR > 2016	19
-	-	-
8	TITLE-ABS-KEY ((knowledgebase OR (“linked open data” AND “semantic web”)) AND sparql AND (process* OR method* OR technique* OR system* OR evaluation OR performance OR benchmark OR procedure*)) AND PUBYEAR > 2016	118
9	TITLE-ABS-KEY ((knowledgebase OR (“linked open data” AND “semantic web”)) AND sparql AND (quality OR method* OR technique* OR system* OR performance OR benchmark OR procedure*) AND evaluation) AND PUBYEAR > 2016	14
10	TITLE-ABS-KEY ((knowledgebase OR “linked open data” OR “semantic web”) AND sparql AND (method* OR technique* OR system* OR performance OR benchmark OR procedure*) AND evaluation AND qualit*) AND PUBYEAR > 2016	17
11	TITLE-ABS-KEY ((knowledgebase OR “linked open data” OR “semantic web”) AND sparql AND (method* OR technique* OR system* OR performance OR benchmark OR procedure*) AND (evaluation OR qualit*)) AND PUBYEAR > 2016	244
12	TITLE-ABS-KEY ((knowledgebase OR “linked open data” OR “semantic web”) AND sparql AND (method* OR technique* OR system* OR performance OR procedure*) AND (evaluation OR qualit* OR benchmark)) AND PUBYEAR > 2016	305
13	TITLE-ABS-KEY ((knowledgebase OR “linked open data” OR “semantic web”) AND sparql AND (method* OR technique* OR system* OR performance OR procedure*) AND evaluation AND qualit* AND benchmark) AND PUBYEAR > 2016	3
14	TITLE-ABS-KEY ((knowledgebase OR “linked open data” OR “semantic web”) AND sparql AND (method* OR technique* OR system* OR performance OR procedure*) AND (evaluat* OR qualit* OR benchmark)) AND PUBYEAR > 2016	391

Nr.	Query	Treffer
15	TITLE-ABS-KEY ((knowledgebase OR “linked open data” OR “semantic web”) AND sparql AND (method* OR technique* OR system* OR performance OR procedure*) AND evaluat* W/10 (qualit* OR benchmark)) AND PUBYEAR > 2016	36
16	TITLE-ABS ((knowledgebase OR “linked open data” OR “semantic web”) AND sparql AND (method* OR technique* OR system* OR performance OR procedure*) AND (evaluat* OR qualit* OR benchmark)) AND PUBYEAR > 2016	148
17	TITLE-ABS ((knowledgebase OR “linked open data” OR “semantic web”) AND sparql AND rdf* AND (technique* OR system* OR performance) AND (evaluat* OR qualit* OR benchmark)) AND PUBYEAR > 2016	81

10.2 Iterative Suche I Ergebnisse Query Nr 7

Nr.	Title	Authors	Year	DOI
1	Using SPARQL to unify queries over data, ontologies, and machine learning models in the PhenomeBrowser knowledgebase	Syed A., Kafkas Ş., Kulmanov M., Hoehndorf R.	2022	
2	SWAT4HCLS 2022 - 13th International Conference on Semantic Web Applications and Tools for Health Care and Life Sciences	[No author name available]	2022	
3	Knowledge discovery using SPARQL property path: The case of disease data set	Rajabi E., Sanchez-Alonso S.	2021	10.1177/ 0165551519865495
4	Extending ONTAgri with service-oriented architecture towards precision farming application	Fahad M., Javid T., Beenish H., Siddiqui A.A., Ahmed G.	2021	10.3390/ su13179801
5	Interaction network provides clues on the role of bcar1 in cellular response to changes in gravity	Bauer J., Gombocz E., Schulz H., Hauslage J., Grimm D.	2021	10.3390/ computation9080081
6	A method of knowledgebase curation using rdf knowledge graph and sparql for a knowledge-based clinical decision support system	Mattam X.J., Lourdusamy R.	2021	
7	GlyGen data model and processing workflow	Kahsay R., Vora J., Navelkar R., Mousavi R., Fochtman B.C., Holmes X., Pattabiraman N., Ranzinger R., Mahadik R., , et al.	2020	10.1093/ bioinformatics/btaa238

Nr.	Title	Authors	Year	DOI
8	Insight in adhesion protein sialylation and microgravity dependent cell adhesion—An omics network approach	Bauer T.J., Gombocz E., Wehland M., Bauer J., Infanger M., Grimm D.	2020	10.3390/ ijms21051749
9	A Comparison of the Cognitive Difficulties Posed by SPARQL Query Constructs	Warren P., Mulholland P.	2020	10.1007/ 978-3-030-61244-3_1
10	A hybrid similarity computing method for KBQA	Wang C., Zhang X., Feng Z.	2020	
11	The neXtProt knowledgebase in 2020: Data, tools and usability improvements	Zahn-Zabal M., Michel P.-A., Gateau A., Nikitin F., Schaeffer M., Audot E., Gaudet P., Duek P.D., Teixeira D., De Laval V.R., Samarasinghe K., Bairoch A., Lane L.	2020	10.1093/nar/ gkz995
12	Augmenting cancer cell proteomics with cellular images – A semantic approach to understand focal adhesion	Bauer T.J., Gombocz E., Krüger M., Sahana J., Corydon T.J., Bauer J., Infanger M., Grimm D.	2019	10.1016/ j.jbi.2019.103320
13	Semantic-based recommendation tool for library management system using domain specific ontology	Priya P., Velmurugan P.	2019	10.35940/ijrte. C5236.098319
14	Updates in Rhea: SPARQLing biochemical reaction data	Lombardot T., Morgat A., Axelsen K.B., Aimo L., Hyka-Nouspikel N., Niknejad A., Ignatchenko A., Xenarios I., Coudert E., Redaschi N., Bridge A.	2019	10.1093/ nar/gky876
15	OnTraNetBD: A knowledgebase for the travel network in Bangladesh	Islam M.R., Hossain B.A., Imteaj M.N., Akhter S., Jogesh H.S., Mostafa M.B.	2018	10.1109/ R10-HTC.2017. 8288931
16	A Question and Answering System for Management of Cloud Service Level Agreements	Mittal S., Gupta A., Joshi K.P., Pearce C., Joshi A.	2017	10.1109/CLOUD. 2017.92
17	An ontology-based approach to semantic health resource knowledge base development for crisis preparation decision support system	Zhu M., Qu J., Chen R., Wang X., Huang Q., Zhong S.	2017	10.18293/ SEKE2017-076

Nr.	Title	Authors	Year	DOI
18	UniProt: The universal protein knowledgebase	Bateman A., Martin M.J., O'Donovan C., Magrane M., Alpi E., Antunes R., Bely B., Bingley M., Bonilla C., Britto R., Bursteinas B., Bye-AJee H., Cowley A., , et al.	2017	10.1093/nar/gkw1099
19	The neXtProt knowledgebase on human proteins: 2017 update	Gaudet P., Michel P.-A., Zahn-Zabal M., Britan A., Cusin I., Domagalski M., Duek P.D., Gateau A., Gleizes A., Hinard V., De Laval V.R., Lin J., Nikitin F., Schaeffer M., Teixeira D., Lane L., Bairoch A.	2017	10.1093/nar/gkw1062

10.3 Iterative Suche II Ergebnisse Query Nr 17

Nr.	Title	Authors	Year	DOI
1	19th International Conference on European Semantic Web Conference, ESWC 2022	[No author name available]	2022	
2	Semantic Web for Meteorological and Oceanographic Data	Danyaro K.U., Liew M.S.	2022	10.1109/ICCIT52419.2022.9711621
3	An Efficient Distributed SPARQL Query Processing Scheme Considering Communication Costs in Spark Environments	Lim J., Kim, Lee H., Choi D., Bok K., Yoo J.	2022	10.3390/app12010122
4	Efficient Retrieval of Data Using Semantic Search Engine Based on NLP and RDF	Yadav U., Duhan N.	2021	10.13052/jwe1540-9589.2084
5	Qb4mobolap: A vocabulary extension for mobility olap on the semantic web	Wisnubhadra I., Baharin S.K., Emran N.A., Setyohadi D.B.	2021	10.3390/a14090265
6	An Effective Discrete Artificial Bee Colony Based SPARQL Query Path Optimization by Reordering Triples	Ozger Z.B., Uslu N.Y.	2021	10.1007/s11390-020-9901-y

Nr.	Title	Authors	Year	DOI
7	An automatic ontology-based approach to support logical representation of observable and measurable data for healthy lifestyle management: Proof-of-concept study	Chatterjee A., Prinz A., Gerdes M., Martinez S.	2021	10.2196/ 24656
8	A hybrid approach combining R*-tree and k-d trees to improve linked open data query performance	Sun Y., Zhao T., Yoon S., Lee Y.	2021	10.3390/ app11052405
9	Research and Application of Complex Event Processing Method Based on RDF Stream	Yu B., Wang H., Wang Q.	2021	10.1109/ CCDC52312.2021. 9602210
10	Schema-Based Mapping Approach for Data Transformation to Enrich Semantic Web	Natarajan S., Vairavasundaram S., Teekaraman Y., Kuppusamy R., Radhakrishnan A.	2021	10.1155/2021/ 8567894
11	18th Extended Semantic Web Conference, ESWC 2021	[No author name available]	2021	
12	Recommendation System Based on Semantic Web Approach	Belattar S., Abdoun O., Khatir H.E.	2021	10.1007/978-3-030-73882-2_53
13	HDT Bitmap Triple Indices for Efficient RDF Data Exploration	Wenzel M., Liebig T., Glimm B.	2021	10.1007/978-3-030-77385-4_7
14	Sequential linked data: The state of affairs	Daga E., Meroño-Peñuela A., Motta E.	2021	10.3233/ SW-210436
15	Ontop4theWeb: SPARQLing the Web On-the-fly	Bereta K., Papadakis G., Koubarakis M.	2021	10.1109/ICSC50631. 2021.00053
16	Web of Things Interoperability Using JSON-LD	Elsayed K.I., Elgamel M.S.	2020	10.1109/ICCTA52020. 2020.9477674
17	The Architecture of Farsi Knowledge Graph System	Sajadi M.B., Bidgoli B.M.	2020	
18	A novel tool for standardizing clinical data in a semantically rich model	Freedman H.G., Williams H., Miller M.A., Birtwell D., Mowery D.L., Stoeckert C.J., Jr.	2020	10.1016/ j.yjbinx.2020. 100086
19	S3QLRDF: Property Table Partitioning Scheme for Distributed SPARQL Querying of large-scale RDF data	Hassan M., Bansal S.K.	2020	10.1109/ SMDS49396.2020. 00023
20	Research on seismic performance assessment method for buildings based on BIM and ontology	Chen P., Shi J., Jiang L.	2020	
21	WODII: a solution to process SPARQL queries over distributed data sources	Rabhi A., Fissoune R.	2020	10.1007/s10586-019-03004-1

Nr.	Title	Authors	Year	DOI
22	A comparative study of dotnetrdf and SEmWeb.net semantic web libraries	Mahoro L.J., Fonou-Dombeu J.V.	2020	10.1109/ icABCD49160.2020. 9183808
23	Enabling ad-hoc reuse of private data repositories through schema extraction	Gleim L.C., Karim M.R., Zimmermann L., Kohlbacher O., Stenzhorn H., Decker S., Beyan O.	2020	10.1186/ s13326-020- 00223-z
24	Towards making sense of Spark-SQL performance for processing vast distributed RDF datasets	Ragab M., Tommasini R., Eyvazov S., Sakr S.	2020	10.1145/ 3391274. 3393632
25	17th Extended Semantic Web Conference, ESWC 2020		2020	
26	17th Extended Semantic Web Conference, ESWC 2020		2020	
27	Bidirectional transformation of MES source code and ontologies	Katti B., Plociennik C., Ruskowski M., Schweitzer M.	2020	10.1016/j.promfg. 2020.02.070
28	Random Forest Based Searching Approach for RDF	Soliman H.	2020	10.1109/ACCESS. 2020.2980155
29	Deep learning based searching approach for RDF graphs	Soliman H.	2020	10.1371/journal. pone.0230500
30	Querying massive RDF data using spark	Banane M., Belangour A.	2019	10.30534/ijatcse/ 2019/68842019
31	Building self-clustering RDF databases using Tunable-LSH	Aluç G., Özsu M.T., Daudjee K.	2019	10.1007/ s00778-018-0530-9
32	A semantic ontology for disaster trail management system	Ahmad A., Othman R., Fauzan M., Ilyas Q.M.	2019	10.21533/ PEN.V7I2.542
33	15th International Conference on Semantic Systems, SEMANTiCS 2019		2019	
34	Towards a Scalable Semantic-Based Distributed Approach for SPARQL Query Evaluation	Sejdiu G., Graux D., Khan I., Lytra I., Jabeen H., Lehmann J.	2019	10.1007/ 978-3-030- 33220-4_22
35	A Worst-Case Optimal Join Algorithm for SPARQL	Hogan A., Riveros C., Rojas C., Soto A.	2019	10.1007/978-3- 030- 30793-6_15
36	A semantic ontology for disaster trail management system	Ahmad A., Othman R., Fauzan M., Ilyas Q.M.	2019	
37	QuWeDa 2019 - Proceedings of the QuWeDa 2019: 3rd Workshop on Querying and Benchmarking the Web of Data, co-located with 18th International Semantic Web Conference, ISWC 2019	[No author name available]	2019	

Nr.	Title	Authors	Year	DOI
38	Benchmarking spark-SQL under alliterative RDF relational storage backends	Ragab M., Tommasini R., Sakr S.	2019	
39	Analysis of the effect of query shapes on performance over LDF interfaces	Montoya G., Keles I., Hose K.	2019	
40	Intuitive ontology-based SPARQL queries for RDF data exploration	Rodriguez Diaz A., Benito-Santos A., Dorn A., Abgaz Y., Wandl-Vogt E., Theron R.	2019	10.1109/ ACCESS.2019. 2948115
41	Semantic web query join optimization using modified grey wolf optimization algorithm	Jose R.T., Poulose S.L.	2019	10.22266/ ijies2019.1031.16
42	16th International Semantic Web Conference, ESWC 2019		2019	
43	New approach based on model driven engineering for processing complex SPARQL queries on hive	Banane M., Belangour A.	2019	10.14569/ ijacsa.2019.0100474
44	Benchmarking question answering systems	Usbeck R., Röder M., Hoffmann M., Conrads F., Huthmann J., Ngonga-Ngomo A.-C., Demmler C., Unger C.	2019	10.3233/ SW-180312
45	A prototype for semantic knowledge retrieval from educational ontology using RDF and SPARQL	Mahaboob Hussain S., Kanakam P., Suryanarayana D.	2019	10.1007/978-981-13-1708-8_50
46	The semantic web MIDI tape: An interface for interlinking MIDI and context metadata	Meroño-Peñuela A., De Valk R., Daga E., Daquino M., Kent-Muller A.	2018	10.1145/ 3243907.3243909
47	Efficiently processing and storing library linked data using apache spark and parquet	Sharma K., Marjit U., Biswas U.	2018	10.6017/ ital.v37i3.10177
48	RDF data storage techniques for efficient SPARQL query processing using distributed computation engines	Hassan M., Bansal S.K.	2018	10.1109/ IRI.2018.00056
49	Type-2 fuzzy ontology–aided recommendation systems for IoT–based healthcare	Ali F., Islam S.M.R., Kwak D., Khan P., Ullah N., Yoo S.-J., Kwak K.S.	2018	10.1016/ j.comcom.2017.10.005
50	Multi-query optimization via common sub query elimination for SPARQL	Zhou X., Luo J., He T.	2018	10.1109/ ISCID.2017.125

Nr.	Title	Authors	Year	DOI
51	SPedia: A central hub for the linked open data of scientific publications	Aslam M.A., Aljohani N.R.	2018	10.4018/978-1-5225-5191-1.ch071
52	A query language for semantic complex event processing: Syntax, semantics and implementation	Gillani S., Zimmermann A., Picard G., Laforest F.	2018	10.3233/SW-180313
53	17th International Semantic Web Conference, ISWC 2018		2018	
54	17th International Semantic Web Conference, ISWC 2018		2018	
55	A portable natural language interface to Arabic ontologies	Hakkoum A., Kharrazi H., Raghay S.	2018	10.14569/IJACSA.2018.090311
56	R2RS: Schema-based relational databases mapping to linked datasets	Kim J.R., Han S.-K.	2018	10.14419/ijet.v7i2.33.13868
57	MaSQue: An approach for flexible metadata storage and querying in RDF	Frey J., Hellmann S.	2018	
58	The compression of indexed data and fast search for large RDF graphs	Kaneiwa K., Fujiwara K.	2018	10.1527/tjsai.E-H43
59	Publication and usage of official Czech pension statistics Linked Open Data	Klímek J., Kučera J., Nečaský M., Chlapek D.	2018	10.1016/j.websem.2017.09.002
60	EpiK: A Knowledge Base for Epidemiological Modeling and Analytics of Infectious Diseases	Hasan S.M.S., Fox E.A., Bisset K., Marathe M.V.	2017	10.1007/s41666-017-0010-9
61	HPL algorithm for semantic information retrieval with RDF AND SPARQL	Kanakam P., Hussain S.M., Suryanarayana D.	2017	
62	Efficient Analytical Queries on Semantic Web Data Cubes	Etcheverry L., Vaisman A.A.	2017	10.1007/s13740-017-0082-y
63	A Question and Answering System for Management of Cloud Service Level Agreements	Mittal S., Gupta A., Joshi K.P., Pearce C., Joshi A.	2017	10.1109/CLOUD.2017.92
64	C-SWRL: A Unique Semantic Web Framework for Reasoning over Stream Data	Jajaga E., Ahmedi L.	2017	10.1142/S1793351X17400165
65	User-Configurable Semantic Data Stream Reasoning Using SPARQL Update	Rinne M., Nuutila E.	2017	10.1007/s13740-017-0076-9
66	Secure Semantic Web Application Development: Present and Future	Noor U., Rashid Z.	2017	10.1109/CSE-EUC-DCABES.2016.263

Nr.	Title	Authors	Year	DOI
67	Utilizing typed dependency subtree patterns for answer sentence generation in question answering systems	Perera R., Nand P., Naeem A.	2017	10.1007/s13748-017-0113-9
68	A design for additive manufacturing ontology	Dinar M., Rosen D.W.	2017	10.1115/1.4035787
69	An efficient and large-scale reasoning method for the semantic Web	Amir S., Aït-Kaci H.	2017	10.1007/s10844-016-0435-2
70	C-SWRL: SWRL for Reasoning over Stream Data	Jajaga E., Ahmedi L.	2017	10.1109/ICSC.2017.64
71	BioFed: Federated query processing over life sciences linked open data	Hasnain A., Mehmood Q., Sanae Zainab S., Saleem M., Warren C., Zehra D., Decker S., Rebholz-Schuhmann D.	2017	10.1186/s13326-017-0118-0
72	Evaluating open access journals using Semantic Web technologies and scorecards	Hallo M., Luján-Mora S., Maté A.	2017	10.1177/0165551515624353
73	Benchmarking RDF storage solutions with Iguana	Conrads F., Lehmann J., Saleem M., Ngomo A.-C.N.	2017	
74	Confederated International Conference On the Move to Meaningful Internet Systems, OTM 2017 held in conjunction with Conferences on CoopIS, CandTC and ODBASE 2017		2017	
75	Medical semantic question answering framework on RDF data cubes	Akhtar U., Hussain J., Lee S.	2017	10.1007/978-3-319-66188-9_23
76	SRDF: A novel lexical knowledge graph for whole sentence knowledge extraction	Nam S., Choi G., Choi K.-S.	2017	10.1007/978-3-319-59888-8_27
77	Medical information system based on support vector machine in web network	Tao R., Sun Z.	2017	
78	SM4MQ: A Semantic Model for Multidimensional Queries	Varga J., Dobrokhotova E., Romero O., Pedersen T.B., Thomsen C.	2017	10.1007/978-3-319-58068-5_28
79	Visual analytics system for LOD using sampling-based structure estimation	Takama Y., Yabe A., Ishikawa H.	2017	10.1527/tjsai.WII-B
80	SPedia: A central hub for the linked open data of scientific publications	Aslam M.A., Aljohani N.R.	2017	10.4018/IJSWIS.2017010108

Nr.	Title	Authors	Year	DOI
81	Enabling interoperability in the internet of things: A OSGi semantic information broker implementation	D'Elia A., Viola F., Roffia L., Azzoni P., Cinotti T.S.	2017	10.4018/IJSWIS.2017010109