Visualisierung Semantischer Relationen am Beispiel des Semantic MediaWikis Surgipedia

Seminararbeit von Tim Stein, 1538898

Institut für Angewandte Informatik und Formale Beschreibungsverfahren des
Karlsruher Instituts für Technologie
Referent: Prof. Dr. Rudi Studer
Betreuer: Dipl.-Inform. Benedikt Kämpgen

Zusammenfassung. Im Rahmen dieser Seminararbeit wurde ein technischer Prototyp für die Visualisierung semantischer Daten eines Semantic Mediawikis als Netzwerkgraph entwickelt. Besonderes Merkmal des Prototypen ist, dass sowohl die Knoten als auch die Kanten vom Benutzer per SPARQL-Query individuell konfiguriert und gefiltert werden können. Des weiteren können Kontenfarbe, -größe und -form ebenfalls über das SPARQL-Query individuell eingestellt werden.
Die Lösung basiert auf HTML5/Javascript und dem Javascript-Framework D3 zur Darstellung des Graphen.
Insbesondere für eine überschaubare Anzahl an Knoten konnte das Ziel erreicht werden, eine übersichtliche Darstellung individueller SPARQL-Anfragen zu schaffen. Bei einer größeren Anzahl Knoten und Kanten ist noch weitere Arbeit in Richtung Filterung und Hervorhebung einzelner Knoten und Kanten notwendig.

Inhaltsverzeichnis

1 Einführung und Problem

In der SFB Wissensbasis werden Daten rund um das Thema Cognition-Guided Surgery in semantischer Form abgespeichert. Ein Teil dieser Wissensbasis stellt das SurgiPedia dar, ein semantisches Mediawiki, das primär der Schaffung eines einheitlichen Vokabulars am SFB dient ([vgl. Sur14], [SFB]).
Mit diesem Hintergrund ist es beispielsweise von Interesse, welche Ontologien wie stark im Surgipedia verwendet werden, um wichtige bzw. weit verbreitete Ontologien einfach sichtbar zu machen. Andere Fragestellungen betreffen die Darstellung aller Teilprojekte und die mit den Projekten verbundenen Relationen, z.B. alle Mitarbeiter die an diesen Projekten arbeiten.
Aus diesen unterschiedlichen Fragestellungen, die nur Beispiele darstellen und durch viele weitere Fragestellungen ergänzt werden können, entsteht der Wunsch, diese Fragen einfach und schnell verständlich zu beantworten.
Um beliebige Abfragen dieser Art in Surgipedia zu ermöglichen, wurde bereits ein SPARQL Endpoint (http://aifb-ls3-vm2.aifb.kit.edu:8890/sparql) eingerichtet, über den mit Hilfe der semantischen Abfragesprache SPARQL Daten ausgegeben werden können. Die genannten Fragen lassen sich prinzipiell über SPARQL modellieren, die Ergebnisdaten können dann in verschiedenen textbasierten Formaten wie z.B. HTML oder CSV ausgegeben werden.
Diese textliche Ausgabe der Abfragedaten ist der Ausgangspunkt dieser Seminararbeit und stellt das Problem dar, das gelöst werden soll: Die Daten sollen für Menschen einfacher interpretierbar und lesbar sein, als die genannte Textausgabe.
Die Daten und die Antworten auf genannte Fragen sind also bereits vorhanden, das Problem liegt vielmehr in der Darstellungsweise.

1.1 Zielsetzung dieser Arbeit

Eine grafische Visualisierung der Abfrage-Ergebnisse ist eine vielversprechende Möglichkeit, der Problemstellung zu begegnen. Diese bietet insbesondere den Vorteil, die mächtigen menschlichen kognitiven und wahrnehmenden Fähigkeiten zu nutzen, was mit reiner textlicher Darstellung nicht möglich ist [vgl. FCS10].
Ziel der Seminararbeit ist es, einen Prototypen zur Visualisierung von SPARQL-Abfragen des semantischen Mediawikis Surgipedia zu erstellen.
Um dieses abstrakte Ziel zu konkretisieren, sollen nun im Folgenden einige technische Anforderungen an den Prototypen definiert werden:

1.1.1 Darstellung als Netzwerk Semantische Daten lassen sich als sogenannte „Triples“ (Subjekt, Objekt und eine Eigenschaft) darstellen. Beispielsweise würde sich die Relation „Max Mustermann arbeitet an Teilprojekt A“ darstellen mit einem Subjekt „Max Mustermann“, der Eigenschaft „arbeitet an“ und dem Objekt „Teilprojekt A“. Daher bietet sich die Darstellung als Netzwerk an, da sich Subjekt, Objekt und Eigenschaft direkt als gerichtete Kante („Link“) mit Quelle und Ziel interpretieren lassen.

1.1.2 Darstellung beliebiger SPARQL-Abfragen als Knoten Es soll möglich sein, über eine frei definierbare SPARQL-Abfrage die angezeigten Knoten auszuwählen. Dies bietet eine hohe Flexibilität für die zur Zeit noch nicht genau zu definierenden Anwendungsmöglichkeiten der Visualisierung.

1.1.3 Darstellung beliebiger SPARQL-Abfragen als Knoten-Verbindungen (Links) Außerdem soll es möglich sein, die angezeigten Verbindungen ebenfalls als Ergebnis einer beliebigen SPARQL-Abfrage frei zu wählen.

1.1.4 Färbung, Form, Größe und Beschriftung der Knoten Um die Ergebnisse der Abfragen einfacher visuell auswerten zu können, ist es praktisch, den Knoten ein selbst definierbares Erscheinungsbild zu geben. Dies soll ebenfalls möglichst flexibel sein und daher am besten über SPARQL funktionierten.

1.1.5 Zeitersparnis gegenüber textlicher Auswertung Als wichtiges Kriterium gilt die Zeitersparnis in der Auswertung (durch einen menschlichen Betrachter) gegenüber der bisherigen, textlichen Ausgabe der SPARQL-Ergebnisse. Dieser Punkt hängt mit der Forderung nach einer benutzerfreundlichen Bedienung, einer übersichtlichen Darstellung und einer performanten Applikation zusammen.

1.1.6 Performanz Auch große Netzwerke sollen performant dargestellt werden können. Dies betrifft sowohl die serverseitigen Laufzeiten als auch die clientseitigen Browserdarstellung.

1.1.7 Open Source Der Prototyp soll nach Fertigstellung veröffentlicht werden (z.B. unter Github oder Google Code). Um den Quellcode möglichst komplett vollständig veröffentlichen zu können, aber ebenso um die Anpassbarkeit bei Bedarf bis in Detail zu ermöglichen, sollen Open Source Komponenten während der Entwicklung bevorzugt werden.

2 Technisches Fundament

Im Folgenden werden kurz einige technische Begriffe eingeführt und bei Bedarf die Wahl der jeweiligen Technik begründet.

2.1 Semantic Mediawiki

Surgipedia ist ein Mediawiki mit der Semantic Mediawiki (SMW) Extension, die das Abspeichern semantischer Relationen innerhalb des Wikis ermöglicht. Entsprechend stellt eine mit SMW kompatible Lösung eine grundlegende Anforderung dar.

2.2 RDF

RDF ist ein standardisiertes Datenaustauschformat im Internet und eignet sich gut für die Speicherung semantischer Relationen. Dabei werden Beziehungen in Form von Triples dargestellt, das heißt Subjekt, Objekt und Eigenschaft, welche die beiden verbindet [vgl. W3C13]. Dieses Datenformat lässt sich demnach gut zur Darstellung als Netzwerk nutzen. Da RDF ein standardisiertes Format ist und die im Semantic Mediawiki gespeicherten semantischen Daten im RDF-Format ausgegeben werden können, eignet sich das RDF-Format als Grundlage für den Datenaustausch.

2.3 Triple Store

Triple Stores sind Datenbankmanagementsysteme um in RDF abgespeicherte Daten effizient zu verwalten. Triple Stores verwenden im Gegensatz zu relationalen Datenbanksystemen die speziell für RDF entwickelte Abfragensprache SPARQL ([vgl. Jua13]). Surgipedia setzte die OpenVirtuoso Triple Store Software ein ([vgl. Jua13]).
[Git14]

2.4 SPARQL

SPARQL ist eine Abfragesprache für RDF, mit der verschiedenste Anfragen wie die genannten Beispiele (vgl. 1) formuliert werden können. Mit Hilfe des SPARQL-Endpoints (vgl. http://aifb-ls3-vm2.aifb.kit.edu:8890/sparql) können SPARQL-Abfragen (im Folgenden auch SPARQL-„Query“ genannt) an die SFB-Wissensbasis benutzerfreundlich ausgeführt werden. Die Ergebnisdaten können dabei u.a. in RDF ausgegeben werden.
Da diese nützliche Infrastruktur bereits existiert, wurde sie auch für die folgende Arbeit verwendet. Prinzipiell können jedoch auch für andere semantische Mediawikis SPARQL-Endpoints eingerichtet werden. Der hier verwendete Endpoint basiert auf der OpenVirtuoso Tripe Store Software.

2.5 D3 Data-Driven Documents

„D3 Data-Driven Documents“ ist ein Open Source JavaScript Framework, welche die Darstellung von Daten auf HTML5-Basis ermöglicht. Mit D3 lassen sich große Datenmengen in verschiedensten Arten visualisieren [vgl. Git14]. Das D3 Framework wird zur Darstellung des Netzwerkes genutzt, da es die notwendigen Funktionen zur Darstellung eines Netzwerks von Haus aus mitbringt, gleichzeitig jedoch die Darstellung über Javascript individuell konfiguriert werden kann. Des weiteren ist es für große Datenmengen ausgelegt und unabhängig von proprietären Systemen wie z.B. Adobe Flash. Die Entwicklung wird aktiv vorangetrieben, weshalb man von einem zukunftssicheren Framework ausgehen kann [vgl. Git14].

2.6 VisualRDF

VisualRDF ist ein Prototyp, der die ansprechende grafische Darstellung von RDF-Daten als Netzwerk zum Ziel hat [vgl. Alv14]. VisualRDF verwendet u.a. D3 (vgl. 2.5). Es eignet sich hervorragend als Grundlage für die angestrebte Netzwerk-Visualisierung, muss jedoch für die gewünschten Zwecke noch weitreichend angepasst werden, da zur Zeit alle Attribute, die vom Wiki in RDF exportiert werden, auch angezeigt werden. So würden beispielsweise um einen einzelnen Mitarbeiter zu visualisieren, über 10 Knoten mit Pfeilverbindungen (z.B. Typ, Bezeichnung, Seite im Wiki, Definition, ..., zuletzt geändert, wikiPageSortKey, uvm.) angezeigt werden, was selbst bei einer geringen Anzahl Knoten den Browser überlastet und sehr unübersichtlich wird.

3 Prototyp-Design

3.1 Prototyp-Name

Der erstellte Prototyp soll der Einfachkeit halber „Wissensbasis-Visualierer“ (oder englisch: KnowledgeBaseVisualizer) heißen, da er genau diesen Zweck erfüllen soll.

3.2 Architektur

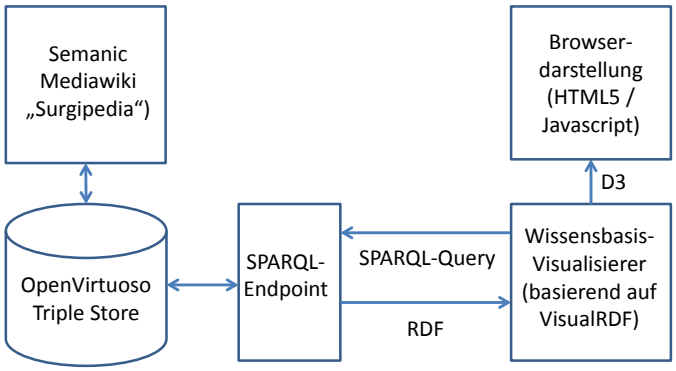


Abb. 1. Verknüpfung der eingesetzten Technologien, eigene Darstellung

Das unter 2 eingeführte technische Fundament wird in Abbildung 1 in der zum Einsatz kommenden Struktur dargestellt. Das Semantic Mediawiki Surgipedia speichert seine Daten im OpenVirtuoso Triple Store. Über den SPARQL-Endpoint kann der Wissensbasis-Visualisierer auf diese Daten zugreifen und bekommt die Ergebnisse im RDF-Format zurück. Die Darstellung der Daten erfolgt dann mit Hilfe von D3 im Browser.

3.3 Frontend-Design

Das Frontend-Design ist mit Hilfe von Bootstrap aufgebaut und sehr schlicht und übersichtlich gehalten. Links oben befindet sich das Logo (das auch als „Zurücksetzen“-Link verwendet werden kann), rechts daneben das Funktionsmenü, darunter die Visualisierung und unten die Eingabemasken für die SPARQL-Queries sowie die Text-Ausgabe der Ergebnisse.
Das Layout ist responsive, d.h. es passt sich an die Bildschirmgröße des Anzeigegeräts an. Es wurde aus Usability-Sicht darauf geachtet, dass der Graph möglichst groß dargestellt wird, gleichzeitig jedoch die Überschrift unterhalb des Graphen sichtbar bleibt, damit dem Benutzer klar ist, dass er nach unten scrollen kann.

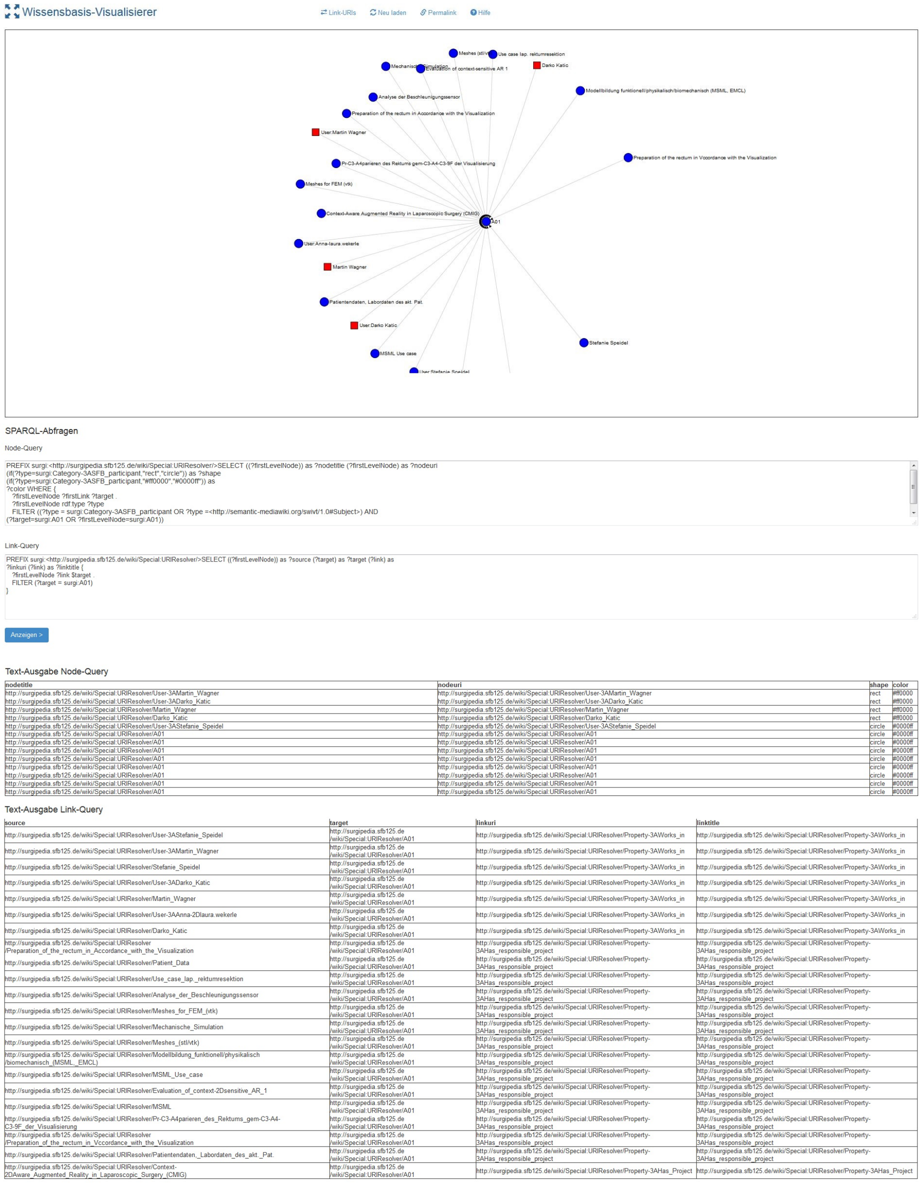
3.4 Frontend-Aufbau und Funktionen

3.4.1 SPARQL-Abfragen Wie bei den technischen Anforderungen (vgl. 1.1) formuliert, kommt den SPARQL-Eingabefeldern eine zentrale Bedeutung zu: Die selektierten Knoten und deren Darstellungsweise werden über das Node-Query gesteuert. Das zweite SPARQL-Query, das Link-Query, steuert, welche Relationen zwischen den Knoten dargestellt werden.
Die beiden Eingabefelder sind als Textfelder realisiert, sodass die Abfragen einfach eingesehen und editiert werden können.
Ein Klick auf „Absenden“ startet die Anfrage und aktualisiert die Visualisierung.

3.4.2 Graph Der Graph wird aus den beiden SPARQL-Abfragen generiert: Es werden alle Knoten angezeigt, die mittels des Node-Query selektiert wurden, und alle Kanten zwischen diesen Knoten, die mittels des Link-Query abgefragt wurden.

3.4.3 Funktionsmenü Rechts oben befindet sich das Menü, das folgende Funktionen ermöglicht:
Mit einem Klick auf **Link-URIs** werden die für die Kanten hinterlegten URIs ein- bzw. ausgeblendet. Standardgemäß sind die Link-URIs ausgeblendet, um die Übersichtlichkeit zu steigern.
Mit **Neu laden** wird dieselbe Abfrage nochmals gestartet und neu erzeugt. Da der Graph nicht direkt vom User beeinflusst werden kann, kann es passieren, dass sich Schriften überlagern und so nicht gut lesbar sind (vgl. 6.2). In diesem Falle kann ein Neu-Laden der Visualisierung hilfreich sein.
Permalink erzeugt einen permanenten Link zur aktuellen Abfrage, so dass man diese leicht per Mail versenden kann. Man beachte, dass die Visualisierung beim Empfänger neu generiert wird.
Hilfe öffnet eine Hilfe-Seite, welche kurz die Variablen der beiden SPARQL-Abfragen erläutert und ein Beispiel zum einfacheren Einstieg gibt.

3.4.4 Zurücksetzen Ein Klick auf das Logo links oben löscht die Visualisierung und setzt die Abfragen zurück.



SPARQL-Abfragen

Node-Query

```
PREFIX surgi:<http://surgipedia.sfb125.de/wiki/Special:URIResolver/>SELECT ((?firstLevelNode)) as ?nodetitle (?firstLevelNode) as ?nodeuri
(if(?type=surgi:Category-3ASFB_participant,"rect","circle")) as ?shape
(if(?type=surgi:Category-3ASFB_participant,"#ff0000","#0000ff")) as
?color WHERE {
  ?firstLevelNode ?firstLink ?target .
  ?firstLevelNode rdf:type ?type
  FILTER ((?type = surgi:Category-3ASFB_participant OR ?type =<http://semantic-mediawiki.org/swivt/1.0#Subject>) AND
(?target=surgi:A01 OR ?firstLevelNode=surgi:A01))
```

Link-Query

```
PREFIX surgi:<http://surgipedia.sfb125.de/wiki/Special:URIResolver/>SELECT ((?firstLevelNode)) as ?source (?target) as ?target (?link) as
?linkuri (?link) as ?linktitle {
  ?firstLevelNode ?link $target .
  FILTER (?target = surgi:A01)
}
```

Text-Ausgabe Node-Query

nodetitle	nodeuri	shape	color
http://surgipedia.sfb125.de/wiki/Special:URIResolver/User-3AMartin_Wagner	http://surgipedia.sfb125.de/wiki/Special:URIResolver/User-3AMartin_Wagner	rect	#ff0000
http://surgipedia.sfb125.de/wiki/Special:URIResolver/User-3ADarko_Katic	http://surgipedia.sfb125.de/wiki/Special:URIResolver/User-3ADarko_Katic	rect	#ff0000
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Martin_Wagner	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Martin_Wagner	rect	#ff0000
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Darko_Katic	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Darko_Katic	rect	#ff0000
http://surgipedia.sfb125.de/wiki/Special:URIResolver/User-3AStefanie_Speidel	http://surgipedia.sfb125.de/wiki/Special:URIResolver/User-3AStefanie_Speidel	circle	#0000ff
http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	circle	#0000ff
http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	circle	#0000ff
http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	circle	#0000ff
http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	circle	#0000ff
http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	circle	#0000ff
http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	circle	#0000ff
http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	circle	#0000ff
http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	circle	#0000ff

Text-Ausgabe Link-Query

source	target	linkuri	linktitle
http://surgipedia.sfb125.de/wiki/Special:URIResolver/User-3AStefanie_Speidel	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AWorks_in	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AWorks_in
http://surgipedia.sfb125.de/wiki/Special:URIResolver/User-3AMartin_Wagner	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AWorks_in	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AWorks_in
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Stefanie_Speidel	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AWorks_in	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AWorks_in
http://surgipedia.sfb125.de/wiki/Special:URIResolver/User-3ADarko_Katic	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AWorks_in	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AWorks_in
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Martin_Wagner	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AWorks_in	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AWorks_in
http://surgipedia.sfb125.de/wiki/Special:URIResolver/User-3AAnna-2Dlaura.wekerle	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AWorks_in	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AWorks_in
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Darko_Katic	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AWorks_in	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AWorks_in
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Preparation_of_the_rectum_in_Accordance_with_the_Visualization	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Patient_Data	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Use_case_lap._rektumresektion	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Analyse_der_Beschleunigungssensor	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Meshes_for_FEM_(vtk)	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Mechanische_Simulation	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Meshes_(stl/vtk)	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Modellbildung_funktionell/physikalisch/biomechanisch_(MSML,_EMCL)	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project
http://surgipedia.sfb125.de/wiki/Special:URIResolver/MSML_Use_case	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Evaluation_of_context-2Dsensitive_AR_1	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project
http://surgipedia.sfb125.de/wiki/Special:URIResolver/MSML	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Pr-C3-A4parieren_des_Rektums_gem-C3-A4-C3-9F_der_Visualisierung	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Preparation_of_the_rectum_in_Vccordance_with_the_Visualization	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Patientendaten,_Labordaten_des_akt._Pat.	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_responsible_project
http://surgipedia.sfb125.de/wiki/Special:URIResolver/Context-2DAware_Augmented_Reality_in_Laparoscopic_Surgery_(CMIG)	http://surgipedia.sfb125.de/wiki/Special:URIResolver/A01	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_Project	http://surgipedia.sfb125.de/wiki/Special:URIResolver/Property-3AHas_Project

Abb. 2. Die Benutzeroberfläche des Wissensbasis-Visualisierer

3.4.5 Textausgabe Node- und Link-Query Am Ende der Seite werden sowohl die Ergebnisse des Node- als auch des Link-Queries in tabellarischer Textform angegeben. Dies kann z.B. zur Überprüfung, ob die Visualisierung korrekt ist, genutzt werden.

3.5 Input

Um Daten abzufragen, werden zwei Queries erwartet: Das Node-Query und das Link-Query. Das Node-Query bestimmt, welche Knoten auf welche Art und Weise angezeigt werden sollen. Dieser Input ist verpflichtend, da sonst nichts angezeigt werden kann. Das Link-Query bestimmt, welche Verbindungen zwischen den einzelnen Knoten angezeigt werden. Dieser Input ist optional, wenn leer, werden die Knoten ohne Verbindung angezeigt. Verbindungen zwischen Knoten, die nicht im Node-Query abgefragt wurden, werden im Sinne der Robustheit ignoriert.

3.5.1 Node-Query Das Node-Query unterstützt folgende Variablen:

Variable	Beschreibung	Werte
nodeuri	Die URI des Knotens	String
nodetitle	Angezeigter Knoten-Name	String
shape	Angezeigte geometrische Form	rect,circle,ellipse
color	Angezeigte Farbe	Hexadezimale Farbangabe, z.B. #00ff00
size	Angezeigte Größe	Integer (in Pixeln, Standard: 10)
priority	Da gleiche Knoten (URIs) mehrfach selektiert werden können, und somit unterschiedliche Stylinganweisungen gelten können, gilt immer die Stylinganweisung mit der höchsten Priorität.	Integer

Beispiel: Zeige alle Objekte vom Typ „Subject“ oder „ASFB-Participant“,die mit dem Projekt A01 verbunden sind (oder mit denen A01 verbunden ist). ASFB-Participants sollen als rotes Rechteck, die Subjects als blauer Kreis angezeigt werden.

```
PREFIX surgi:<http://surgipedia.sfb125.de/wiki/Special:URIResolver/>
SELECT ((?firstLevelNode)) as ?nodetitle
(?firstLevelNode) as ?nodeuri
(if(?type=surgi:Category-3ASFB_participant,"rect","circle")) as ?shape
(if(?type=surgi:Category-3ASFB_participant,"#ff0000","#0000ff")) as
```

```
?color WHERE {
   ?firstLevelNode ?firstLink ?target .
   ?firstLevelNode rdf:type ?type
   FILTER ((?type = surgi:Category-3ASFB_participant OR
?type =<http://semantic-mediawiki.org/swivt/1.0#Subject>) AND
(?target=surgi:A01 OR ?firstLevelNode=surgi:A01))
}
```

3.5.2 Link-Query Das Link-Query unterstützt folgende Variablen:

Variable	Beschreibung	Werte
source	Die URI des Startknotens	String
target	Die URI des Zielknotens	String
linkuri	Die URI, die die Verbindungsart beschreibt	String
linktitle	Angezeigter Verbindungsname	String

Beispiel: Zeige alle Verbindungen mit dem Knoten A01.

```
PREFIX surgi:<http://surgipedia.sfb125.de/wiki/Special:URIResolver/>
SELECT ((?firstLevelNode)) as ?source
(?target) as ?target
(?link) as ?linkuri
(?link) as ?linktitle {
   ?firstLevelNode ?link $target .
   FILTER (?target = surgi:A01)
}
```

3.6 Output

Auf Basis der Inputdaten werden die Ergebnisse (Nodes und ggf. Links) zum einen in tablellarischer Form als auch als Graph entsprechend der Spezifikation im SPARQL-Query (vgl. 3.5) dargestellt.
Beispiel: Die grafische Ausgabe der genannten Beispiel-Abfragen:

3.7 Installation und Konfiguration

3.7.1 Installation Die technischen Abhängigkeiten des Skriptes sind gering: eine PHP-Umgebung (z.B. auf Apache-Basis, getestet auf PHP 5.3.28) genügt, keine Datenbank wird verwendet. Das restliche JavaScript läuft im Browser und ist auf Firefox Version 30, Google Chrome Version 36 und Internet Explorer Version 11 getestet.
Die Anforderungen an den SPARQL-Endpoint sind, dass er Queries per GET-Befehl unterstützt und die SPARQL-Ergebnisse in zwei Formaten: zum einen JSON (application/sparql-results+json), zum anderen als TEXT/HTML (text/html) ausgeben kann. Das erstere Datenformat wird für die Darstellung des Graphs verwendet, das zweite für die tabellarische Ansicht der Ergebnisse.

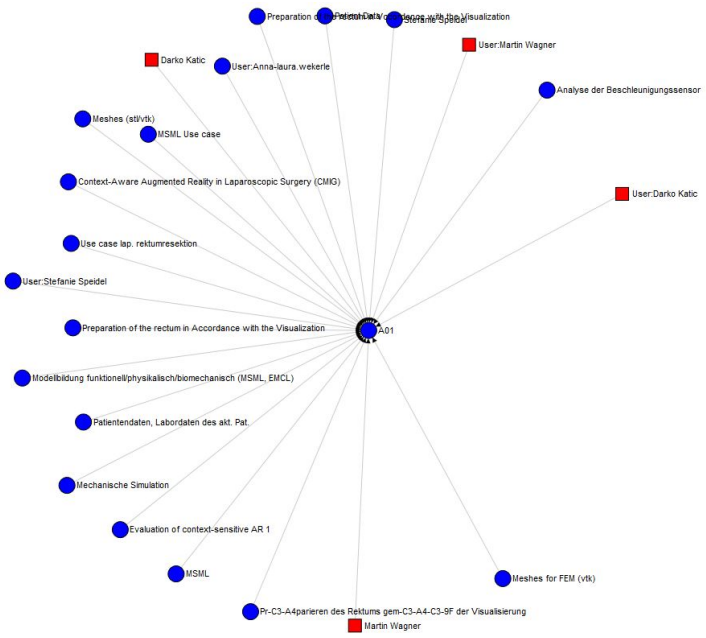


Abb. 3. Beispiel-Visualisierung

Um das Tool, das bei Github veröffentlicht ist, zu installieren, genügt es also, die Dateien auf ein entsprechendes Serversystem zu installieren und danach bei Bedarf die Konfigurationsdatei config.php im Stammverzeichnis anzupassen.

https://github.com/creffect/knowledgebase-visualizer/tree/master

3.7.2 Konfiguration Im Folgenden sind die Konfigurationsvariablen in der Konfigurationsdatei config.php erklärt:

$graphUri
Der Default Data Set Name (Graph URI)
(vgl. http://aifb-ls3-vm2.aifb.kit.edu:8890/sparql)

Beispiel:

`http://wissensbasis.sfb125.de/`

$requestUri
Die Abfrage-URL zum Endpoint mit den Platzhaltern GRAPHURI für die url-kodierte GraphURI, QUERY für das SPARQL-Query und FORMAT für das gewünschte Ausgabeformat (application/sparql-results+json oder text/html)

Beispiel:

```
http://aifb-ls3-vm2.aifb.kit.edu:8890/sparql?
default-graph-uri={GRAPHURI}&query={QUERY}&format={FORMAT}&timeout=0&debug=on
```

$uriResolver
Die URI zum URI-Resolver des semantischen Mediawikis.
Wird verwendet, um die angezeigten Titel der Knoten automatisch zu kürzen (da ansonsten eine URL angezeigt werden würde).

Beispiel:

```
http://surgipedia.sfb125.de/wiki/Special:URIResolver/
```

4 Evaluation

Unter den Zielen der Arbeit (vlg. 1.1) wurden Anforderungen an den Prototypen genannt. Nun soll überprüft werden, welche davon erfüllt werden konnten und bei welchen noch Verbesserungsbedarf besteht.
Um die Evaluation greifbarer zu machen, wird im Folgenden mit einer Beispielabfrage gearbeitet, die hier kurz eingeführt werden soll:

Beispiel: Alle Verbindungen eines Teilprojektes, drei Ebenen tief
Es sollen alle Verbindungen des Teilprojekts A01 bis auf drei Hierarchieebenen angezeigt werden. Die direkt und die indirekt verbundenen Knoten sollen unterschiedlich dargestellt werden.

4.1 Darstellung als Netzwerk

Die Visualisierung erfolgt als gerichteter Netzwerkgraph. Die semantischen Relationen lassen sich einblenden (über den Menüpunkt Link-URI). Diese Anforderung kann also vollständig erfüllt werden.
Der Graph des genannten Beispiels sieht wie folgt aus:

4.2 Darstellung beliebiger SPARQL-Abfragen als Knoten und Links

Prinzipiell kann diese Anforderung bis auf unten genannte Einschränkungen als erfüllt gelten.
Es können sehr spezielle Anfragen in SPARQL realisiert und visualiert werden.
Die beiden Queries des genannten Beispiels lauten beispielweise:
Node-Query

```
PREFIX surgi:<http://surgipedia.sfb125.de/wiki/Special:URIResolver/>
SELECT (?firstSource) as ?nodetitle
(?firstSource) as ?nodeuri
?type as ?type
(if(?firstTarget=surgi:A01,"1",
```

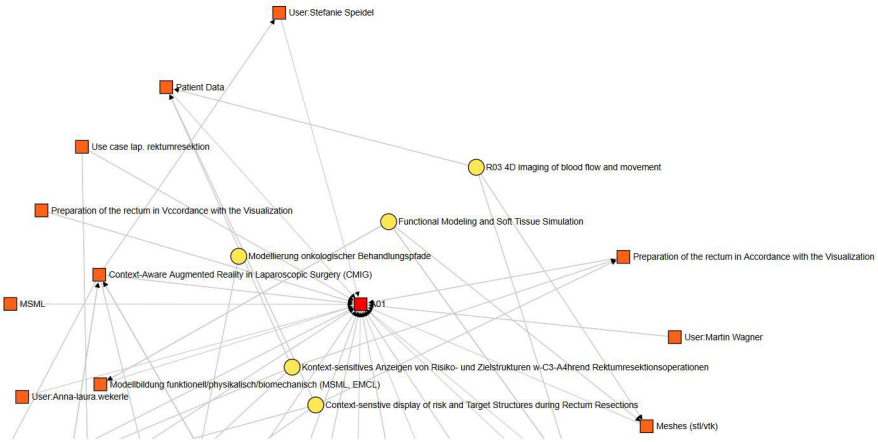


Abb. 4. Ausgabe (Ausschnitt) der obenstehenden Anfrage

```
    if(?secondTarget=surgi:A01,"2","3"))) as ?priority
(if(?firstSource=surgi:A01,"rect", if(?firstTarget=surgi:A01,"rect",
    if(?secondTarget=surgi:A01,"circle","ellipse")))) as ?shape
(if(?firstSource=surgi:A01,"#ff0000",
    if(?firstTarget=surgi:A01,"#FF631A",
       if(?secondTarget=surgi:A01,"#FFE855","#ADFF55")))) as ?color
(if(?firstSource=surgi:A01,"30", if(?firstTarget=surgi:A01,"20",
    if(?secondTarget=surgi:A01,"15","10")))) as ?size
WHERE {
{
?firstSource ?firstLink ?firstTarget .
?firstSource rdf:type ?type
FILTER ((?firstTarget=surgi:A01 OR ?firstSource=surgi:A01))
}
UNION {
?firstSource ?firstLink ?firstTarget .
?firstTarget ?secondLink ?secondTarget .
?firstSource rdf:type ?type
FILTER ((?secondTarget=surgi:A01))
}
UNION {
?firstSource ?firstLink ?firstTarget .
?firstTarget ?secondLink ?secondTarget .
?secondTarget ?thirdLink ?thirdTarget .
?firstSource rdf:type ?type
FILTER ((?thirdTarget=surgi:A01))
}
}
```

Link-Query

```
PREFIX surgi:<http://surgipedia.sfb125.de/wiki/Special:URIResolver/>

SELECT ((?firstSource)) as ?source (?firstTarget) as ?target
  (?firstLink) as ?linkuri (?firstLink) as ?linktitle {
{
   ?firstSource ?firstLink ?firstTarget .
   ?firstSource rdf:type ?type
   FILTER (?firstTarget = surgi:A01)
}
UNION
{
?firstSource ?firstLink ?firstTarget .
?firstTarget ?secondLink ?secondTarget .
?firstSource rdf:type ?type
FILTER (?secondTarget = surgi:A01)
}
UNION {
?firstSource ?firstLink ?firstTarget .
?firstTarget ?secondLink ?secondTarget .
?secondTarget ?thirdLink ?thirdTarget .
?firstSource rdf:type ?type
FILTER (?thirdTarget = surgi:A01)
}
}
```

Bedingung für funktionierende SPARQL-Queries ist, dass die für die Darstellung benötigten Variablen bereitgestellt werden (vgl. 3.5).
Außerdem gibt es Einschränkungen bei Abfragen, die eine große Zahl an Knoten zurückgeben. Dies kann schnell insbesondere clientseitig zu Performance-Verlusten führen und an die Grenzen eines gewöhnlichen Client-Gerätes führen. Das Thema wird unter der Anforderung der Performanz noch näher beschrieben. Eine weitere technische Grenze ergibt sich in diesem Kontext im Bezug auf den Permalink: Sehr lange URLs können zu Schwierigkeiten führen. Da die Permalinks die Queries in der URL abspeichern, kann es bei umfangreicheren Queries (zum Beispiel über insgesamt 2000 Zeichen) zu Problemen kommen.

4.3 Färbung, Form, Größe und Beschriftung der Knoten

Auch diese Anforderung kann im Kern erfüllt werden: Die Farbe der Knoten ist frei wählbar, es werden mehrere - wenn auch eine beschränkte Anzahl (Vgl. 3.5) - Formen ermöglicht, die Größe und Beschriftung der Knoten lässt sich über SPARQL-Variablen steuern.
Eine Einschränkung hierbei ist jedoch die Größe: Wird diese erhöht, kann es dazu führen, dass die Pfeilspitzen der Links nicht dargestellt werden, da diese unterhalb der Knoten liegen und überdeckt werden. Leider konnte trotz mehrstündiger

Suche keine Lösung für dieses Problem gefunden werden. Prinzipiell ist dieses Problem jedoch lösbar, evtl. muss die Codebasis von VisualRDF überarbeitet oder ersetzt werden. Auf diesen Punkt wird bei den offenen Fragen (6.2) eingegangen.

4.4 Zeitersparnis gegenüber textlicher Auswertung

Dieser Punkt ist sicherlich der subjektivste und kritischste Punkt der Anforderungen.
Eine grafische Darstellung kann sehr viel Zeit sparen und Zusammenhänge - übersichtlicher machen. Im Graph des genannten Beispiels (Abb. 4) wird der zentrale Knoten (1. Hierarchiebene) als rotes Quadrat, die direkt verbundenen Knoten (2. Hierarchieebene) als orangene Quadrate und die damit verbundenen Knoten (3. Hierarchiebene) als gelbe Kreise dargestellt. So kann sehr schnell eine Art „Heat-Map“ erstellt und direkt und indirekt verbundene Knoten visuell sortiert werden.
Trotzdem sieht man hier bereits Schwächen: Zum Teil ist es schwierig, die vielen Links auseinanderzuhalten. Die Anordnung der Knoten stellt nicht unbedingt eine Nähe zum zentralen Knoten dar (die gelben Knoten der dritten Ebene sind viel näher am roten Knoten als die orangenen, direkt verbundenen Knoten). Dies hängt z.T. damit zusammen, dass die gelben Knoten mit weiteren Knoten verbunden sind und so zwischen diesen (und somit eher in der Mitte des Graphen) platziert werden.
Ein weiteres Problem stellt der Umgang mit großen Datenmengen dar. Die Knoten können zur Zeit aus Performancegründen nicht bewegt werden. Das kann schnell dazu führen, dass Beschriftungen nicht lesbar sind, da sich Beschriftungen überlagern können.
Zusammenfassend kann man sagen, dass Abfragen mit einer geringen Knotenzahl zu guten Ergebnissen führen. Hierbei kann der Nutzen der visuellen Darstellung beliebiger, evtl. sehr komplexer SPARQL-Abfragen voll ausgenutzt werden und es entstehen große Zeitersparnisse bei der Interpretation. Aufgrund technischer Grenzen des Prototyps wird die Darstellung bei vielen Knoten jedoch schnell unübersichtlich. Diese sollten in aufbauenden Arbeiten beseitigt werden. Unter dem Punkt „offene Fragen“ (vgl. 6.2) sind einige Verbesserungsmöglichkeiten aufgeführt.

4.5 Performanz

Prinzipiell haben drei Faktoren Einfluss auf die Performanz des Prototyps: Zum einen die benötigte Zeit, die SPARQL-Ergebnisse vom Endpoint zu erhalten, die serverseitige Performanz des Prototyps selbst und die Anzeige als Graph im Clienten.
Während der erste und der zweite Faktor bei den eingesetzten Servern in der Testumgebung (SPARQL Endpoint des SFB Triple Store und ein Mittwald Managed vServer MAX) eine akzeptable Performanz auch bei komplexen Abfragen

lieferten, war insbesondere die clientseitige Darstellung per D3 zu Beginn der Entwicklung ein Flaschenhals. Durch die Deaktivierung der Drag und Drop-Funktionalität konnte jedoch auch hier eine akzeptable Performance von einer Renderzeit von insgesamt 7,5 Sekunden auf dem Test-Rechner (Intel Core i5-3570 CPU @3.40GHz, 8GB RAM, Windows 7 x64 Professional, Firefox 30) erreicht werden.

Das Beispielquery war hierbei:
Node-Query:

```
PREFIX surgi:<http://surgipedia.sfb125.de/wiki/Special:URIResolver/>
SELECT (?firstSource) as ?nodetitle
   (?firstSource) as ?nodeuri
?type1 as ?type
  ("1" as ?priority)
  ("rect" as ?shape)
  ("#FF631A" as ?color)
  ("5" as ?size)
WHERE {
     ?firstSource rdf:type ?type1 .
     FILTER (?type1= <http://www.w3.org/2002/07/owl#Ontology>)
     ?firstSource <http://www.w3.org/2002/07/owl#imports> ?anotherSource.
     ?anotherSource rdf:type ?type1 .
     FILTER(?anotherSource != <http://semantic-mediawiki.org/swivt/1.0>)
}
```

Link-Query:

```
PREFIX surgi:<http://surgipedia.sfb125.de/wiki/Special:URIResolver/>

SELECT ((?firstSource)) as ?source (?firstTarget) as ?target
(?firstLink) as ?linkuri (?firstLink) as ?linktitle {
     {
        ?firstSource ?firstLink ?firstTarget .
        ?firstSource rdf:type ?type .
        ?firstTarget rdf:type ?type .
        FILTER (?type = <http://www.w3.org/2002/07/owl#Ontology>)
     }
}
```

Die Ausführung Desselben ergab ein Ergebnis von 1348 Knoten und 1416 möglichen Links (es werden nur die angezeigt, die zwischen den selektierten Knoten verbinden). Abbildung 5 stellt das Ergebnis dar.

4.6 Open-Source

Sowohl alle eingesetzten Komponenten als auch der Prototyp selbst sind Open-Source, somit ist diese Anforderung erfüllt.

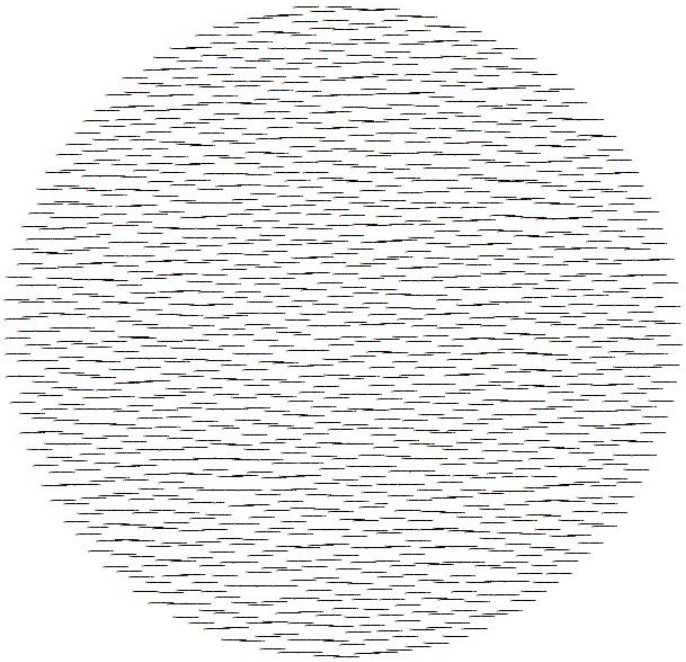


Abb. 5. Graph des Performance-Beispiels mit 1348 Knoten

4.7 Zusammenfassung

Die grundlegenden technischen Anforderungen konnten erfüllt werden, der Prototyp zeigt das Potential dieses Ansatzes. Bei dem für die praktische Nutzung entscheidenden Punkt - der Zeitersparnis durch eine übersichtliche Darstellung und schnelle Interpretation der erstellten Graphen - ist jedoch noch Verbesserungsbedarf: Inbesondere bei Graphen mit vielen Knoten und Verbindungen sind noch übersichtlichere Darstellungsmöglichkeiten nötig. Auf diesen Punkt und die aufbauenden Arbeiten an diesem Prototyp wird unter 6.2 weitergehend eingegangen.

5 Benachbarte Arbeiten

Im Folgenden soll die Arbeit mit drei benachbarten Projekten verglichen werden:

5.1 VisualRDF

Da VisualRDF als Ausgangspunkt des Prototyps dient und gleichzeitig ein ähnliches Ziel - die übersichtliche Darstellung von semantischen Daten - hat, beginnt der Vergleich mit diesem Projekt.
Ein zentraler Unterschied zwischen VisualRDF und dem Wissensbasis-Visualisierer ist, dass VisualRDF keine SPARQL- Abfragen unterstützt sondern gelieferte RDF-Dateien auswertet und visualisert. Ein Problem hierbei ist eine mangelnde Knoten-Filterfunktion, die dazu führt, dass immer sämtliche Knoten sowie Relationen, die als RDF übergeben werden, angezeigt werden. Dies wird schnell unübersichtlich. Der Wissensbasis-Visualisierer unterstützt hingegen keine direkte Auswertung von RDF-Dateien, sondern funktioniert über eine Schnittstelle zu einem SPARQL-Endpoint. Die Filterung der Knoten und Kanten erfolgt über beliebig anpassbare SPARQL-Queries.

5.2 WikiStalker

Wikistalker ist ein Portal, welches die „Visualisierung semantischer Relevanz zwischen verlinkten Wikipedia Artikeln“ [vgl. Sep] wiedergibt. Dabei kann ein beliebiger Wikipedia-Artikel in das Suchen-Feld eingegeben werden und alle verknüpften Artikel werden dargestellt und nach semantischer Relevanz gewichtet. Die semantische Relevanz wird dabei das Data Mining Tool Wikipedia Miner ([vgl. Dav]) berechnet.
Hierbei handelt es sich um eine andere Zielsetzung als beim Wissensbasis--Visualisierer, jedoch ist der Aspekt der Darstellungsform (gewichtet nach Relevanz) und die Möglichkeit, Unterpunkte dynamisch zu öffnen und damit weitere semantische Abfragen zu starten, eine interessante Option, die Komplexität zu filtern. Es ist denkbar, sich von dieser dynamischen Anzeige und Gewichtung für die weitere Verbesserung der Darstellung komplexer Graphen des Wissensbasis-Visualisierer inspirieren zu lassen.

5.3 Flexvis

Flexvis hat das Ziel, semantische Daten webbasiert graphisch übersichtlich darzustellen. Es bietet umfangreiche Funktionen, wie z.B. verschiedene Layouts, Filterung, Navigation, Interaktion (z.B. Drag- und Drop), Suche und Export. Ziel ist es hierbei, eine webbasierte Alternative für die bisher weiterverbreiteten lokal installierte Software im Bereich der semantischen Datenvisualisierung zu schaffen ([vgl. FCS10]).
Insgesamt stellt Flexvis ein sehr mächtiges, umfangreiches Toolkit dar. Es versteht sich als Werkzeugkasten, der durch sogenannte Extensions genutzt werden kann. Diese Extensions werden für bestimmte Zwecke entwickelt, z.B. die BioPortal-Extension, welche ermöglicht, biomedizinische Ontologien und Ressourcen übersichtlich darzustellen und aufzufinden ([vgl. FCS10]).
Im Vergleich zu Flexvis ist der Wissensbasis-Visualisierer deutlich schlanker und verfügt über ein viel geringeres Funktionsspektrum. Darüber hinaus basiert er im Gegensatz zu Flexvis auf HTML5/JavaScript und benötigt kein Java-Plugin. Prinzipiell wäre es denkbar, den Wissensbasis-Visualisierer als Extension für Flexvis zu programmieren. Zum jetzigen Zeitpunkt erscheint dies jedoch als ein vergleichsweise großer Aufwand für das unter der Zielsetzung (vgl. 1.1) beschriebene Problem, das sich auch ohne diese mächtige Bibliothek und die damit einhergehende Komplexität schlank und komplett auf HTML5/JS-Basis lösen lässt. Sollten jedoch weitergehende Funktionalitäten, die in den Wissensbasis-Visualisierer integriert werden könnten, einen großen Aufwand darstellen, wäre die Nutzung des Flexvis Toolkits eine interessante Option.

6 Fazit

6.1 Zusammenfassung der Ergebnisse

Die grafische Darstellungsweise semantischer Daten hat ein großes Potential, die Datenmenge für Menschen schnell verständlich und begreifbar zu machen.
Insbesondere der Filterung der semantischen Daten kommt dabei ein großer Wert zu: Dieser konnte mit Hilfe von zwei separaten SPARQL-Queries - einer für Knoten, einer für Kanten - Rechnung getragen werden. So lassen sich die Knoten und Kanten selektieren, die für eine Fragestellung relevant sind, alle anderen werden direkt ausgefiltert.
Auch die unterschiedliche grafische Darstellung der verschiedenen Knoten, die ebenfalls über SPARQL steuerbar ist, bietet großes Potential. Es wurde ein flexibler Prototyp geschaffen, dessen prinzipielle Anwendungsmöglichkeiten vielzählig sind.
Trotzdem kommt es bei Abfragen mit vielen Knoten und Kanten zur Zeit schnell zu unübersichtlichen Graphen. Hier wäre eine dynamischere Visualisierung, z.B. durch Echtzeit-Filterung oder Verschiebung einzelner Knoten wichtig.
Dementsprechend kann die bisherige Lösung als Prototyp gesehen werden, der das Potential aufzeigt, jedoch noch nicht voll Nutzbar ist. Dazu ist eine aufbauende Arbeit am Wissensbasis-Visualisierer notwendig, wie unter dem Punkt „Offene Fragen“ (vgl. 6.2) ausgeführt wird.

6.2 Offene Fragen

Im folgenden Abschnitt werden einige offene Fragen und Verbesserungspotentiale der Anwendung angesprochen, die in aufbauenden Arbeiten berücksichtigt werden können.

6.2.1 Anklickbarkeit Links Bisher lassen sich nur Knoten anklicken. Wünschenswert wären anklickbare Kanten.

6.2.2 Pfeilspitzen bei Größenveränderungen Wie bereits unter 4.3 beschrieben, verschwinden momentan die Pfeilspitzen der Kanten unter den Knoten, wenn diese stark vergrößert werden. Das Problem konnte trotz mehrstündiger Recherche nicht gelöst werden. Vermutlich befindet es sich in der Datei js/main.js, da diese den Code zur Visualisierung (ursprünglich von VisualRDF) enthält. Werden weitere Änderungen in Richtung Filterung und Benutzerfreundlichkeit unternommen, wäre zu prüfen, ob man die Komponenten von VisualRDF entfernt und neu entwickelt. Im Zuge dessen könnte dieses Problem mitgelöst werden.

6.2.3 Bewegen der Knoten Um übereinanderliegende Beschriftungen und Kanten klar zu unterscheiden, kann ein Bewegen einzelner Knoten nützlich sein. Ursprünglich unterstützt VisualRDF Drag- und Drop der einzelnen Knoten. Dies hat jedoch bereits bei recht geringen Knotenzahlen (ca. 20-30) zu deutlichen Performanzeinbußen geführt, weshalb diese Funktion entfernt wurde.
Sollte es nicht möglich sein, diese Funktion performant auch bei großen Knotenanzahlen zu realisieren, wäre denkbar, einzelne Knoten oder Kanten bei Mouseover hervorzuheben und somit besser lesbar / erkennbar zu machen.

6.2.4 Echtzeitfilterung und -suche Obwohl SPARQL bereits eine Art Filterung der Daten darstellt, kann eine Abfrage eine so große Zahl an Knoten zurückgeben, dass der Graph recht unübersichtlich wird. Hierbei empfiehlt sich eine ausgereifte Echtzeit-Filterung, mit welcher Knoten(gruppen) oder Kanten(gruppen) ein- und ausgeblendet werden können. Auch eine Suchfunktion zur Fokussierung einzelner Knoten wäre hilfreich.
Denkbar wäre, die zur Verfügung stehenden Filter ebenfalls durch eine Benutzer-Eingabe konfigurierbar zu gestalten. So könnten benannte Filter einen deutlichen Zugewinn an Usability und Übersichtlichkeit, gerade bei umfangreichen Graphen, bieten.

Literaturverzeichnis

[Alv14] Alvaro Graves, 24.01.2014. https://github.com/alangrafu/visualRDF, besucht: 26.01.2014.

[Dav] David Milne: *Wikipedia Miner*. http://wdm.cs.waikato.ac.nz/, besucht: 19.07.2014.

[FCS10] Sean M. Falconer, Chris Callendar und Margaret Anne Storey: *A Visualization Service for the Semantic Web*, Band 6317 der Reihe *LNCS sublibrary. SL 7, Artificial intelligence*. Springer, Berlin, 2010, ISBN 978-3-642-16438-5.

[Git14] GitHub, 22.01.2014. https://github.com/mbostock/d3, besucht: 26.01.2014.

[Jua13] Juan Sequeda: *Introduction to: Triplestores*, 31.01.2013. http://semanticweb.com/introduction-to-triplestores_b34996, besucht: 19.07.2014.

[Sep] Raschin Sepans: *Wikistalker*. http://www.visualizing.org/visualizations/wikistalker, besucht: 19.07.2014.

[SFB] SFB/TRR 125 - Cognition-Guided Surgery: *SFB/Transregio 125 Cognition-Guided Surgery - Wissens- und modellbasierte Chirurgie*. http://www.cognitionguidedsurgery.de/, besucht: 19.07.2014.

[Sur14] Surgipedia: *Surgipedia - Main Page*, 07.03.2014. http://surgipedia.sfb125.de/wiki/Main_Page, besucht: 19.07.2014.

[W3C13] W3C, 22.03.2013. http://www.w3.org/RDF/, besucht: 26.01.2014.