UNIVERSITÄT LEIPZIG

Fakultät für Mathematik und Informatik

Institut für Informatik

Betriebliche Informationssysteme

Repräsentation und Visualisierung von persönlichen Gewichtungen in semantischen Daten

Bachelorarbeit

vorgelegt von Abicht, Konrad

Studiengang Informatik

Leipzig, August 2010

Inhaltsverzeichnis

1 EINLEITUNG 4

1.1 Übersicht der Bachelorarbeit 4

1.2 Motivation 4

2 EINFÜHRUNG IN DIE THEMATIK DER BACHELORARBEIT 5

2.1 Zur Geschichte 5

2.2 Rasanter Anstieg der Netzteilnehmer 6

2.3 Probleme beim Umgang mit Informationen 6

2.4 Empfehlungssysteme als Lösungsmöglichkeit 7

2.5 Von Empfehlungssystemen zu dieser Arbeit 8

2.6 Einführung in Semantic Web 9

2.7 Vorstellung sonstiger Technologien 10

3 ANFORDERUNGSANALYSE 11

3.1 Die Ausgangssituation 11

3.2 Grobspezifikationen 11

3.2.1 Erweiterbarkeit und Offenheit 11

3.2.2 Wiederverwendbarkeit 12

3.2.3 Datenkapselung 12

3.2.4 Visualisierung in Form eines Graphen 12

3.3 Getroffene Einschränkungen 12

3.4 USE - CASE - Diagramm 13

4 SPEZIFIKATION 14

4.1 Die Paradigmen der Gewichtung 14

4.1.1 Die Gewichtungsaussage 14

4.1.2 Mehrstufige Gewichtung 16

4.1.3 Gleichberechtigte Gewichtungsaussagen 16

4.1.4 Personalisierte Gewichtungsaussagen 16

4.2 UML- Klassendiagramm 17

4.2.1 Die Klasse Graph 18

4.2.2 Die Klasse Edge 19

4.2.3 Die Klasse Node 19

5 IMPLEMENTIERUNG UND EVALUATION 20

5.1 Vorstellung der Ontologie 20

5.1.1 Die OWL-Klasse WeightStatement 20

5.1.2 Die ObjectProperty subject 21

5.1.3 Die ObjectProperty predicate 21

5.1.4 Die ObjectProperty object 22

5.1.5 Die ObjectProperty weight 22

5.2 Die Benutzerschnittstelle 22

5.3 Fremde verwendete Bibliotheken 23

5.4 Die Gewichtungsalgorithmen 24
5.4.1 Demonstration der Vorgehensweise 24
5.5 Gewichtungsalgorithmus: Spaltenaufsummierung 25
5.5.1 Allgemeine Informationen 25
5.5.2 Eigenheiten des Algorithmus 25
5.5.3 Vorgehen als Pseudocode 26
5.6 Gewichtungsalgorithmus: Spaltenmaximum 27
5.6.1 Allgemeine Informationen 27
5.6.2 Eigenheiten des Algorithmus 27
5.6.3 Vorgehen als Pseudocode 28
5.7 Weitere Funktionalitäten 28
5.7.1 Einfügen von Elementen aus einem SPARQL – Ergebnis 29
5.7.2 Konvertierung der Graph-Elemente ist darstellbaren Javascript-Code 30
5.8 Vorbereitungen der Evaluation 31
5.8.1 Konfiguration der Testumgebung 31
5.8.2 Testdatenbestand 31
5.8.3 Verwendete SPARQL – Abfrage 32
5.8.4 Einblick in den Testdatenbestand 32
5.8.5 Übersicht der Elemente des Testdatenbestands 34
5.9 Evaluation der Gewichtungsalgorithmen 35
5.9.1 Ausgewählter Testdatenbestand 36
5.9.2 Die durchgeführten Berechnungen 37
5.9.3 Untere Grenzen bei Gewichtungen 39
5.9.4 Visualisierung der Gewichtungsberechnungen 39
5.9.5 Schlussfolgerungen 41
6 FAZIT UND AUSBLICK 42
6.1 Fazit 42
6.2 Ausblick 42
7 ABKÜRZUNGEN UND BEGRIFFSERKLÄRUNGEN 43
8 ABBILDUNGSVERZEICHNIS 44
9 TABELLENVERZEICHNIS 45
10 CODEVERZEICHNIS 46
11 LITERATURVERZEICHNIS UND WEBLINKS 47
12 QUELLCODE UND ONTOLOGIE 49
13 ERKLÄRUNG 50

1 Einleitung

1.1 Übersicht der Bachelorarbeit

In dieser Bachelorarbeit geht es darum herauszuarbeiten, wie die Gewichtung von Aussagen bei der Informationsinteraktion hilft. Mit Gewichtung ist in dem Zusammenhang die Zuordnung eines Wertes zu einer Aussage gemeint.

Der Hauptteil der Arbeit beschäftigt sich mit der Anforderungsanalyse, der Spezifikation und Implementierung einer Komponente, welche die Gewichtung von Aussagen in Graphen visualisieren kann. Aufgrund der zeitlichen Einschränkung und des begrenzten Seitenumfangs beschränkt sich diese Arbeit auf eine Einführung in die Gewichtungsproblematik sowie die Erstellung eines Prototypen und der damit verbundenen Visualisierung verschiedener Gewichtungsalgorithmen. Hierbei geht es weder um die Ermittlung des besten Algorithmus zur Gewichtung von Aussagen, noch um die Interaktion mit großen Datenmengen.

1.2 Motivation

Die Motivation zur Beschäftigung mit einer Gewichtung von Aussagen in Graphen und die Erstellung eines entsprechenden Prototypen rührt aus einem fremden Projekt, welches unter anderem diese Problematik aufgreift und weiter benutzt. Das Projekt trägt den Arbeitstitel *europa*^{N}. Darin soll die Gewichtung von Aussagen genutzt werden, um darauf aufbauend Benutzerschnittstellen zu implementieren. Die Form und der Umfang des fremden Projektes seien hier nur schemenhaft skizziert, da es sich zu dem Zeitpunkt der Erstellung der Bachelorarbeit erst in der Planungsphase befindet. Es sei aber erwähnt, dass geplant ist, gewisse Elemente, die hier aufgegriffen und analysiert werden, später dort wieder zu verwenden.

2 Einführung in die Thematik der Bachelorarbeit

In diesem Kapitel wird ein kurzer Querschnitt der Medien- und Internetlandschaft erstellt und gewisse Probleme, wie die Informationsflut, aufgegriffen und analysiert. Dabei soll ein Lösungsansatz herausgearbeitet werden, um mit den genannten Problemen umzugehen. Am Ende des Kapitels findet eine Einführung in die technische Umgebung statt, in der die später zu entwickelnde Komponente eingebunden werden soll. Weiterhin werden einige Begriffe näher erläutert.

2.1 Zur Geschichte

Die Geschichte über die Entwicklung des Internet wird in den Medien und der Literatur häufig mit einer Revolution gleichgesetzt. Man wage fast zu behaupten, es entspräche einer Umwälzung wie etwa der Übergang von der Agrar- in die Industriegesellschaft. Die Entwicklung fand jedoch in den ersten Zügen nur schleichend statt.

Am Anfang gab es nur wenige Universitäten und Großunternehmen, welche verschiedene Rechneranlagen in der Größe von Garagen betrieben. Mit der Zeit begannen, durch die Einführung der Mikroelektronik, die Ausmaße der Rechentechnik immer kleiner zu werden. Parallel zu dieser Entwicklung fingen die Nutzer der Rechneranlagen an sich untereinander zu vernetzen. Dabei ging es primär um Austausch von Forschungsergebnissen zwischen Universitäten oder zur Übertragung von geheimen Regierungsdaten. Mit der Zeit entwickelte sich langsam ein Netzwerk. Was heute den gesamten Globus sowohl unter- als auch überirdisch umspannt, war damals ein Flickenteppich, der noch riesige Löcher aufwies.

Es kam zu immer mehr Entwicklungen aus dem Militärsektor oder der Hochindustrie, die in den zivilen Sektor vordrangen. Für den sogenannten Mainstream gab es plötzlich Kassettenrekorder und CD-Player.

Bei all den technischen Innovationen fand im gleichen Atemzug auch eine immer stärkere Vernetzung statt. Das eigentliche Internet, was heutzutage oft durch einen Browser betrachtet oder für den Nachrichtenaustausch über E-Mail genutzt wird, begann seinen rasanten Aufstieg erst später durch die Einführung des grafikfähigen Browsers *Mosaic* im Jahre 1993 [18].

2.2 Rasanter Anstieg der Netzteilnehmer

Eine Online-Studie der ARD und ZDF aus dem Jahre 2000, welche sich mit der Entwicklung der Online-Medien in Deutschland befasst, ermittelte, dass im Jahr 1997 ca. 4.1 Millionen Bundesbürger aktive online waren [1]. Im Jahr 2000 waren es bereits 18.3 Millionen. Das entspricht fast einer Verfünffachung der Netzteilnehmer innerhalb von nur drei Jahren.

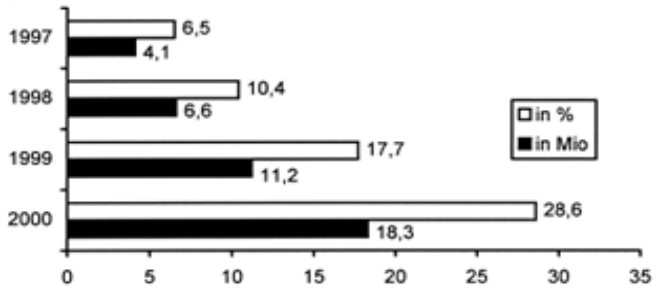

Abb. 1 – ARD-ZDF-Onlinestudie 2000: Entwicklung der Online-Nutzung in Deutschland

Eine weitere Studie der ARD und ZDF aus dem Jahr 2009, weist eine Online-Nutzung von 67,1 Millionen Bundesbürgern auf [2]. Innerhalb von ca. 10 Jahren hat sich die Anzahl der Netzteilnehmer in Deutschlands fast mehr als verdreifacht.

Diese Zahlen waren nur für Deutschland gemessen. In anderen Industrienationen fand eine ähnliche Entwicklung statt. Das Nachrichtenmagazin heise-online brachte am 21.07.2003 eine Meldung, basierend auf Statistiken des China Internet Network Information Center, dass im Jahr 2003, 68 Millionen Chinesen das Internet regelmäßig nutzen [3]. Eine andere Studie der CNNIC vom 15.03.2010 belegt, dass am 31.12.2009 diese Zahl bereits auf ca. 384 Millionen angestiegen war [4].

2.3 Probleme beim Umgang mit Informationen

Ein Missstand der im Zusammenhang mit einer ständig steigenden Informationsflut auftritt, ist die oft unzureichende Qualität von Informationen. Da die Hürden für das Bereitstellen von Daten jeglicher Art, ob Text, Musik oder Video immer weiter sinken, steigt die Menge der neu erscheinenden Daten täglich weiter an.

Heutzutage findet man im Netz fast alles. Wollte man früher Fachwissen zu einem Thema haben, so blieb oft nur der Gang in die nächstgelegene Bibliothek. Gut recherchierten Journalismus gab es häufig nur in Wissenschaftsmagazinen oder bei den Tageszeitungen.

Heutzutage wird einem die Suche durch Suchmaschinen wie Google Search [L1] oder Microsoft 's Bing [L2] erleichtert. Diese bieten über eine mächtige Stichwortsuche nicht nur die Möglichkeit Milliarden von Webseiten zu durchsuchen, sondern auch wissenschaftliche Artikel (über Google Scholar [L3]) oder Bücher (über Google Books [L4]).

Über diese Dienste findet man zwar vieles leichter und schneller, nur ist nicht immer klar, ob die gefundenen Informationen auch den Erwartungen entsprechen, die sich je nach Publikum in Korrektheit, Aktualität und Motiven unterscheiden können.

2.4 Empfehlungssysteme als Lösungsmöglichkeit

Um mit dem im vorherigen Abschnitt erwähnten Missstand umzugehen, gibt es verschiedene Möglichkeiten. Eine davon wäre die Nutzung eines Empfehlungssystems. Verschiedene Onlineplattformen wie Amazon [L5] oder Digg [L6] bieten ihren Nutzern unterschiedliche Bewertungssysteme für die bereitgestellten Produkte an.

Bei Amazon kann man bei der Betrachtung eines konkreten Produktes die Bewertung und Kommentare anderer Käufer sehen. Die Wertung erfolgt auf einer vordefinierten Skala von fünf möglichen zu vergebenden Sternen. Neben den bereitgestellten Daten des Herstellers bekommt der Betrachter damit auch Informationen über die Einsatzfähigkeit, das Nutzungsverhalten oder ähnliches zum dem Produkt.

Ein etwas anderes Bewertungssystem wird bei dem Onlinedienst Digg verwendet. Auf dessen Seite werden beliebige Internetseiten veröffentlicht. Dazu zählen Angebote wie zum Beispiel Nachrichten, Videos und Blog-Einträge. Benutzer können diese Internetseiten dann positiv oder negativ bewerten und zu ihnen weitere Daten erfassen, wie die zugehörige Kategorie. Die Seite, welche viele positive Bewertungen bekommen haben, stehen in den Ratings höher als Seiten die weniger erhalten haben. Alternativ kann man eine Internetseite auch *begraben*, was im Digg - Slang einer negative Bewertung entspricht.

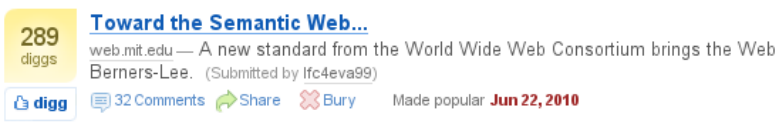

Abb. 2 – Eintrag auf Digg.com mit 289 positiven Bewertungen (diggs) [5]

Wie man an den beiden Beispielen erkennen kann, gibt es verschiedene Möglichkeiten bei der Bewertung von Informationen.

Es existiert verschiedene Alternativen zu Bewertungssystemen. Eine besteht darin, Leute, die man kennt, nach Vorschlägen und Tipps zu fragen und selbst auch neue interessante Dinge seinen Bekannten mitzuteilen. Nutzer tendieren dazu, ehrlichen Bewertungen oder anderen Nutzern, deren Benutzername auf eine natürliche Person hinweist, mehr Vertrauen zu schenken, als einer Bewertung, bei der der Betrachter sofort erkennt, dass sie von einer Firma geschrieben wurde, um das Produkt positiv darzustellen.

Empfehlungen enthalten nicht nur Daten über einem gewissen Gegenstand. Die Informationen, die eine wie auch immer geartete Empfehlung enthält, können auch Erfahrungen im Umgang mit dem Gegenstand oder alternative Nutzungsmöglichkeiten sein. Ob diese Informationen für denjenigen der sie erhält im gleichen Maße nützlich sind, wie für den Bereitsteller, kann nicht beurteilt werden. Aber sie bieten dem Interessenten eine differenzierte Sicht auf den Gegenstand und helfen womöglich bei der späteren Entscheidung. Empfehlungs- und Bewertungssysteme gibt es viele. Die in diesem Kapitel vorgestellten Beispiele sollten die Grundprinzipien dahinter vermitteln.

2.5 Von Empfehlungssystemen zu dieser Arbeit

Die vorgestellten Bewertungssysteme dienten zum Teil als Grundlage für die Implementierung der in dieser Arbeit behandelten Gewichtungsalgorithmen. Wie bereits erwähnt wurden bei der Umsetzung verschiedene Elemente adaptiert.

Zum einen die Bewertung auf einer abgegrenzten Skala, zum anderen werden die abgegebenen Bewertungen personalisiert. Da die Umsetzung in Form eines Prototypen für ein konkretes System vorliegt, sei darauf verwiesen, dass es gewisse Einschränkungen gibt. Diese sind neben der eingeschränkten Praxisfähigkeit, besonders der geringe Funktionsumfang und die Stabilität.

Bei der Entwicklung war das Ziel, eine Softwarekomponente zu erstellen, welche unterschiedliche Konzepte und Hypothesen umsetzt und wo die Möglichkeit besteht, die Berechnungen durch ein Visualisierungsmodul evaluieren zu können. Unter Beachtung dieser Beschränkungen wurde daher bei der Entwicklung der Komponente unter anderem auch darauf geachtet, dass sie später einfach erweiterbar und flexibel bei Änderungen ist.

2.6 Einführung in Semantic Web

In diesem Abschnitt wird auf eine detaillierte Vorstellung der einzelnen verwendeten Begriffe verzichtet. Dennoch gibt es einige Begriffe die im Vorfeld näher erläutert werden müssen, um ein Mindestverständnis zu gewährleisten. Es werden viele Elemente aus der *Semantic Web* – Welt verwendet. Darunter fallen Dinge, wie RDF, OWL, N3, SPARQL und Ontologie. Im Folgenden werden die einzelnen Begriffe kurz erläutert.

RDF

RDF ist eine Sprache zur Repräsentation von Informationen [6]. Die hier verwendete Form basiert auf XML [7]. Vereinfacht gesagt, werden die Daten bei RDF in Form von Tripel abgelegt. Diese bestehen aus einem Subjekt, Prädikat und Objekt.

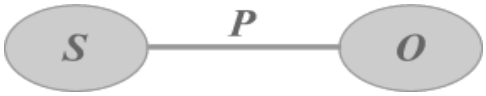

Abb. 3 – Grafische Darstellung eines Tripels aus Subjekt, Prädikat und Objekt

OWL

Ein weiterer Punkt ist die Sprache OWL. Sie bietet unter anderem die Möglichkeit, vorliegende Daten in RDF/XML-Form so aufzubereiten und formal zu beschreiben, dass auch eine Maschine ihre Bedeutung verarbeiten kann. [8]

```
[1.]   <owl:Thing rdf:ID="CentralCoastRegion"/>
[2.]   <owl:Thing rdf:about="#CentralCoastRegion">
[3.]         <rdf:type rdf:resource="#Region"/>
[4.]   </owl:Thing>
```

Code 1 – Kurzes OWL Beispiel

Code 1 ist ein kurzer Codeausschnitt in OWL aus dem W3C-Guide zur OWL – Spezifikation [9]. Die Form der Daten basiert wie bei RDF ebenfalls auf XML.

Notation 3

Gespeicherte Daten in RDF- als auch in OWL-Form können auch über die Sprache Notation3, kurz N3, dargestellt werden [10]. Sie speichert die Daten in einer Form ab, dass Menschen diese relativ einfach lesen und interpretieren können. Die später vorgestellte Ontologie wurde in N3-Code erfasst, um die Erstellung und spätere Erweiterung zu erleichtern.

Ontologie

Der Begriff Ontologie kommt aus der Philosophie und bedeutet aus dem griechischen übersetzt *„Lehre vom Sein“* [11]. Im Kontext des Semantic Web ist eine Ontologie maschinell interpretierbar und beschreibt Begrifflichkeiten, die auf einen Gegenstandsbereich bezogen sind. [Z1]

SPARQL

Als letzter Begriff aus dem Semantic Web Kontext sei SPARQL erwähnt. Das ist eine Abfragesprache für Graphen, deren Datenbestand in RDF vorliegt. Sie stellt Möglichkeiten zur Abfrage von semantischen Daten im RDF-Format bereit [12]. SPARQL - Abfragen werden in der Komponente verwendet, um gezielt Informationen über eine Gewichtung abzufragen.

2.7 Vorstellung sonstiger Technologien

Die bisher vorgestellten Technologien und Begriffe waren aus dem Semantic Web Kontext. Bei der Entwicklung der Komponente wurden zusätzlich noch PHP, HTML und Javascript verwendet. Das sind drei Websprachen, mit denen man verschiedene Teile einer Webpräsentation erstellen kann.

PHP kümmert sich um die Programmlogik und stellt Schnittstellen unter anderem zum Datenbestand her. Die Sprache HTML ist für die Struktur, CSS für das Aussehen einer Webseite verantwortlich. Javascript ist eine Sprache, welche clientseitig im Browser interpretiert wird und für die Interaktivität auf einer Webpräsentation verantwortlich ist. Sie stellt somit Methoden bereit, um auf Nutzeraktionen zu reagieren.

3 Anforderungsanalyse

In diesem Kapitel findet eine Anforderungsanalyse der entwickelten Komponente statt. Dabei wird zuerst auf die Ausgangssituation im Vorfeld der Entwickelung eingegangen. Eine Ist-Analyse liegt hier nicht vor, da es kein bestehendes System gibt, auf das man aufbauen könnte. Im Verlauf werden die Anforderungen in Form einer Grobspezifikation formuliert. Die im vorherigen Kapitel vorgestellten Bewertungssysteme und ihre Eigenschaften fließen dabei mit in die Analyse ein.

3.1 Die Ausgangssituation

Zum Zeitpunkt der Erstellung der Bachelorarbeit existierte keine Komponente für OntoWiki, welche gewichtete Aussagen in Graphen interpretieren und visualisieren kann. Weiterhin gab es verschiedene Systeme zur Darstellung von Graphen, unter anderem auf Basis von Javascript.

Der Hauptteil der Bachelorarbeit beschäftigt sich mit der Erstellung von Gewichtungsalgorithmen und die Präsentation der Berechnungsergebnisse. Um die durchgeführten Berechnungen evaluieren und demonstrieren zu können, gab es im Vorfeld verschiedene Möglichkeiten. Eine bestand darin Tabellen zu nutzen, anhand derer man das Vorgehen und die Zwischenergebnisse präsentieren konnte. Eine andere demonstrierte die Berechnungsergebnisse in Form einer Graphvisualisierung.

Am Ende wurde die letztere Möglichkeit ausgewählt, da man sich davon versprach, einerseits eine bessere Veranschaulichung zu ermöglichen und andererseits die zu entwickelnden Module das Potenzial haben, an anderen Stellen im OntoWiki weiterverwendet werden zu können.

3.2 Grobspezifikationen

Die im letzten Abschnitt erwähnten Gegebenheiten werden nun zu konkreten Anforderungen zusammengefasst und beschrieben.

3.2.1 Erweiterbarkeit und Offenheit

Die Komponente soll so ausgelegt sein, dass sie sich später mit begrenztem Aufwand erweitern lässt. Darunter zählt die einfache Implementierung von weiteren

Gewichtungsalgorithmen oder auch die Erweiterung des Moduls für die Visualisierung um weitere Funktionen. Weiterhin soll durch diese Anforderung die Integration in andere Systembereiche unterstützt werden.

3.2.2 Wiederverwendbarkeit

Die komplette Komponente oder nur einzelne Module aus ihr sollen auch in andere Bereiche des OntoWiki mit überschaubarem Aufwand integriert werden können. Um dies zu gewährleisten, soll die Komponente den Datenbestand selbst verwalten können, ohne das Fremdcode dafür vorgesehen werden muss. Zudem sollen offene Schnittstellen bereitgestellt werden, um die Kommunikation mit einem Fremdcode zu vereinfachen.

3.2.3 Datenkapselung

Die abgespeicherten Daten des Graphen, wie Knoten und Kanten, sollen vor der Außenwelt komplett abgekapselt werden. Der Zugriff auf diese soll nur über spezielle Schnittstellen bereitgestellt werden, um so eine kontrollierte Interaktion zu gewährleisten. Weiterhin soll eine saubere Datenkapselung die Portierung in andere Bereiche unterstützen.

3.2.4 Visualisierung in Form eines Graphen

Die durch die Gewichtungsalgorithmen durchgeführten Berechnungen sollen grafisch in Form eines Graphen präsentiert werden. Dabei sollen sowohl die einzelnen Knoten als auch ihre Kanten gut lesbar dargestellt werden. Die Gewichtungen sollen über eine Abstufung bei den Linienstärken gut erkennbar sein.

3.3 Getroffene Einschränkungen

Die entwickelte Komponente beschränkt sich in der vorliegenden Bachelorarbeit auf folgende Kriterien. Diese werden im Nachhinein kurz erläutert.

Zum einen muss bei jeder SPARQL - Abfrage, deren Ergebnismenge an die Datenstrukturen übergeben werden, im SELECT - Teil die Platzhalter `?s`, `?p` und `?o` enthalten sein. Diese werden später für die Zuordnung von Subjekt, Prädikat und Objekt genutzt und sind nicht dynamisch setz- oder erfassbar. Existieren diese nicht, so wird die Komponente nicht in der Lage sein, die Gewichtungen den Tripel zuzuordnen.

Zum anderen wird für die Nutzung der Komponente die entwickelte Ontologie benötigt. Sie stellt einerseits das benötigte Vokabular bereit, damit man die Gewichtungen interpretieren kann, andererseits liegen der Ontologie noch Testdaten bei. Diese liegen in Form von N3, RDF und OWL vor. Im kommenden Kapitel wird noch näher auf die Ontologie eingegangen.

3.4 USE - CASE - Diagramm

In diesem Abschnitt werden die funktionalen Anforderungen an das System über ein USE – CASE - Diagramm vorgestellt.

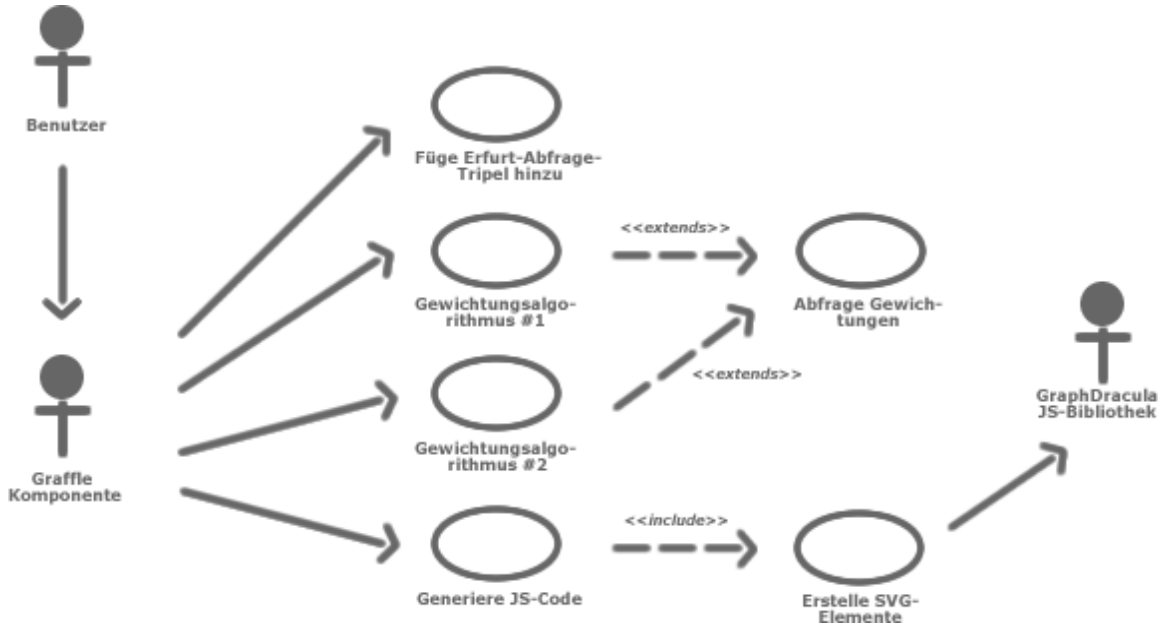

Abb. 4 – USE CASE zur Gewichtungsvisualisierung

Auf der Abbildung 4 ist der Anwendungsfall Gewichtungsvisualisierung dargestellt. Dabei beschränkt sich die Hauptfunktionalität auf wenige Funktionen. Es soll die Möglichkeit bestehen, abgefragte Tripel eines vorherigen SPARQL – Query 's in die geschlossene Datenstruktur aufzunehmen. Dazu müssen sie sich einer Transformation unterziehen. Anschließend sollen die abgespeicherten Daten über Gewichtungsalgorithmen mit Gewichtungen aufbereitet werden.

Der damit entstehende Datenbestand wird später in ausführbaren Javascript-Code umgewandelt, damit er über ein Visualisierungsmodul dargestellt werden kann.

4 Spezifikation

Dieses Kapitel geht auf die Spezifikation der zu entwickelnden Komponente ein. Im Vorfeld werden jedoch zuerst die verwendeten Methoden und Paradigmen vorgestellt und vereinzelt näher erläutert. Es folgt danach die Beschreibung der entwickelten Ontologie und deren Grundlage für die späteren Gewichtungsalgorithmen. Am Ende werden die Bestandteile der zu erstellenden Komponente vorgestellt und durch ein UML - Klassendiagramm visualisiert.

4.1 Die Paradigmen der Gewichtung

Der Kern der zu entwickelnden Komponente sind die Gewichtungsalgorithmen. Jeder Algorithmus muss sich an gewisse Paradigmen halten, kann aber ansonsten frei implementiert werden. Diese Paradigmen werden über die später noch vorgestellte Ontologie festgehalten.

4.1.1 Die Gewichtungsaussage

Eine Gewichtungsaussage ist eine Aussage über eine andere Aussage, wobei dieser noch eine Wertung zugeordnet wird. Diese Wertung kann einer Zahl auf einer begrenzten Skala entsprechen. Eine weitere Möglichkeit wäre, dass diese Wertung keine Zahl, sondern eine beliebige Zeichenkette darstellt. Diese Möglichkeit wird in dieser Arbeit aber nicht betrachtet, sondern nur die Wertung über eine Zahl.

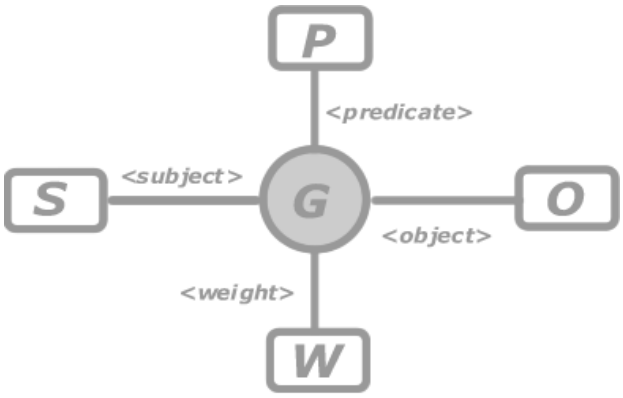

Abb. 5 – Grafische Darstellung einer Gewichtungsaussage

In Abbildung 5 ist schematisch eine Gewichtungsaussage dargestellt. Sie besitzt verschiedene Eigenschaften, darunter `<subject>`, `<predicate>` und `<object>`. Diese Eigenschaften enthalten die jeweiligen Elemente von bestehenden Aussagen. Im Folgenden wird dieser Sachverhalt an einem Beispiel verdeutlicht.

Sei folgendes Tripel gegeben:

Abb. 6 – Grafische Darstellung eines Beispieltripels

In dem, in Abbildung 6, dargestellten Tripel steht das Subjekt `<foo>` über das Prädikat `<ttt>` mit dem Objekt `<bar>` in Verbindung. Diese Aussage hat nun hypothetisch gesehen einen hohen Wert, daher wird die Eigenschaft `<weight>` auf den Wert `0,8` gesetzt, bei einer Skala von 0 bis 1. Eine Gewichtungsaussage, welche dieses abbildet, ist in Abbildung 7 dargestellt:

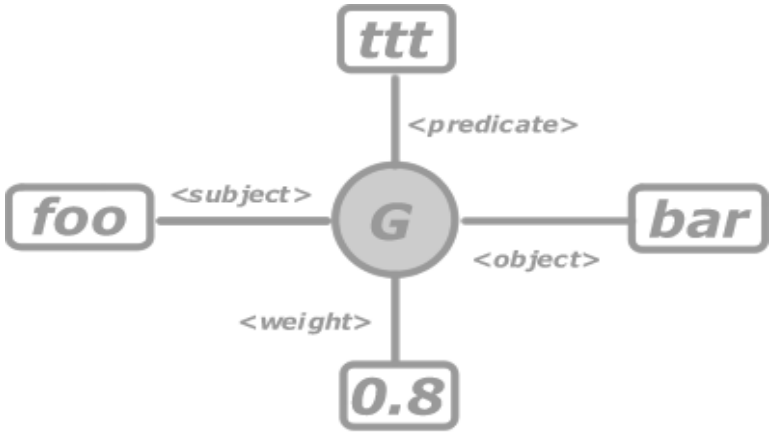

Abb. 7 – Gewichtungsaussage über Beispieltripel mit Wert 0,8

Es ist zu erwähnen, dass man nicht alle drei Eigenschaften `<subject>`, `<predicate>` und `<object>` setzen muss, um eine Gewichtungsaussage treffen zu können. Jedoch muss mindestens eine davon gesetzt sein.

4.1.2 Mehrstufige Gewichtung

Es findet eine Gewichtung in mehreren Stufen statt. Diese richten sich nach der Anzahl der gesetzten Eigenschaften in einem Tripel, wodurch es insgesamt drei Stufen gibt. Die Eigenschaft `<weight>` ist hingegen immer gesetzt. Die Möglichkeiten der Eigenschaftskombination aus `<subject>`, `<predicate>` und `<object>` seien in folgender Tabelle 1 illustriert:

	1. Stufe	**2. Stufe**	**3. Stufe**
Möglichkeiten	3	3	1
Gesetzte Elemente	S, P oder O	SP, PO oder SO	Nur SPO

Tab. 1 – Auflistung Gewichtungsstufen und Kombinationsmöglichkeiten

4.1.3 Gleichberechtigte Gewichtungsaussagen

Jede vorliegende Gewichtungsaussage wird gleich behandelt. Damit geht der Wert bei einer niedrigstufigen Gewichtung genauso stark in das Endergebnis ein, wie der einer höheren Stufe. Eine Alternative dazu wäre es, dass man jeder Stufe einen prozentualen Anteil an der Endgewichtung gibt und diesen Anteil erhöht, je höher die Stufe wird. Diese Alternative wird aber in dieser Arbeit nicht behandelt.

4.1.4 Personalisierte Gewichtungsaussagen

Weiterhin sei festgelegt, dass jede Gewichtung entweder für alle Benutzer gelten soll oder nur für einen Konkreten. Damit soll die Möglichkeit gegeben werden unterschiedlich mit Gewichtungen umzugehen, welche die gleiche Aussage betreffen, wobei es mehrere Werte von unterschiedlichen Personen gibt. Dieses Paradigma ähnelt dem personalisierten Kommentieren in einem Bewertungssystem, wie in Kapitel 2 bereits angesprochen. In der Komponente nicht vorgesehen, wäre es praktisch möglich, dass man sich nur Gewichtungen von konkreten Benutzern anzeigen lassen kann. Oder dass die Gewichtungen später gesondert dargestellt werden, z.B. durch stärkere Hervorhebung.

4.2 UML- Klassendiagramm

Es wird im Folgenden eine Übersicht über die erstellten und verwendeten Klassen gegeben. Dabei wurden die drei Klassen, welche die eigentliche Datenstruktur für die Speicherung von Graphdaten darstellen, farblich hervorgehoben.

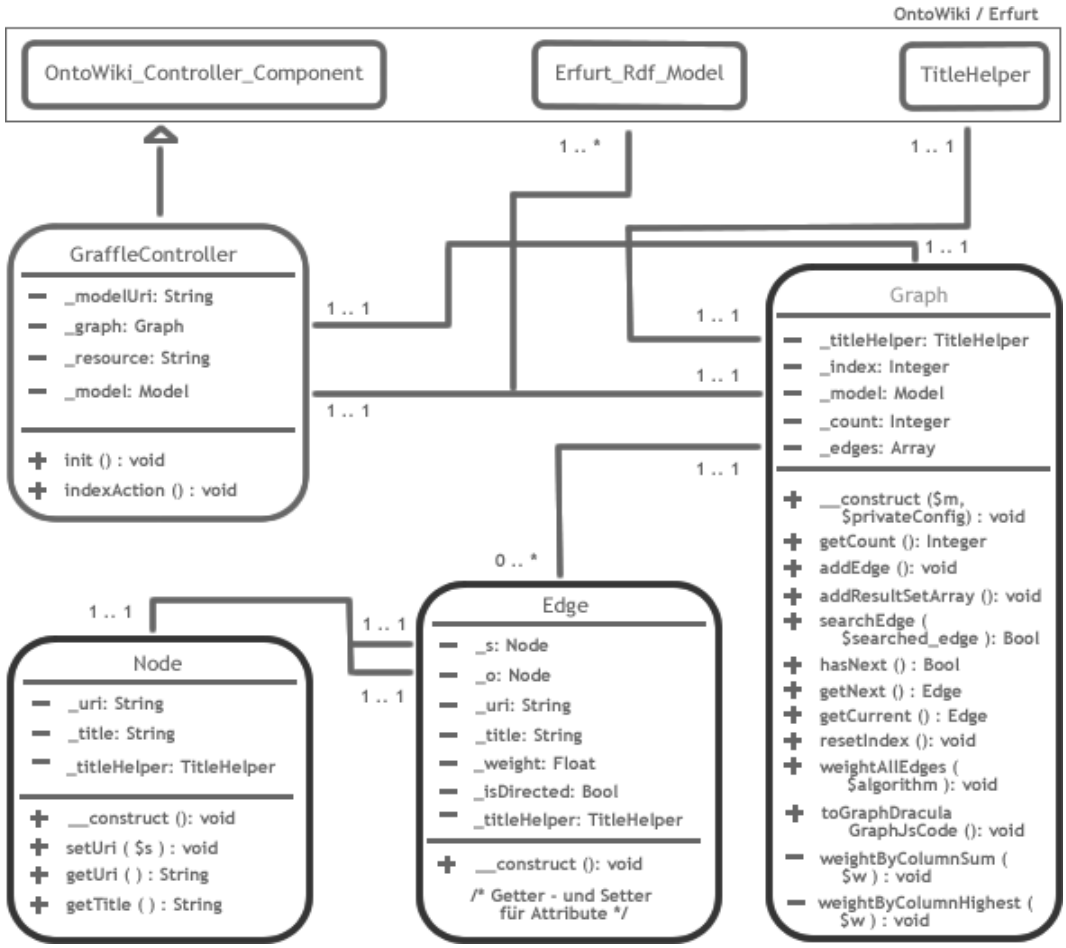

Abb. 8 – UML – Klassendiagramm der Komponente

Die oberen drei Klassen, `OntoWiki_Controller_Component`, `Erfurt_Rdf_Model` und `TitleHelper`, sind Klassen aus OntoWiki und der Erfurt-API. Die erstellte Komponente, hier dargestellt durch die Klasse `GraffleController`, ist von der Klasse `OntoWiki_Controller_Component` abgeleitet. Sie erbt damit die nötigen Eigenschaften und Methoden, um als Komponente in OntoWiki agieren zu können. In ihr werden die drei Klassen genutzt, welche die Hauptfunktionalität bereitstellen: `Graph`, `Edge` und `Node`. Im Folgenden werden die Klassen näher erläutert.

4.2.1 Die Klasse Graph

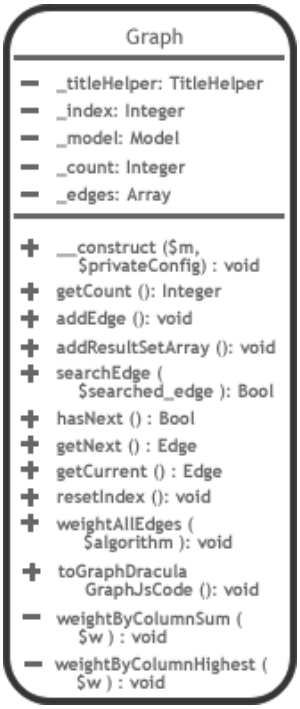

Abb. 9 – UML – Klassendiagramm der Klasse Graph

Diese Klasse repräsentiert einen Graphen. Sie stellt Funktionen bereit, um die Elemente des Graphen, die Kanten und Knoten, verwalten zu können. Die Graph-Klasse speichert eine Liste von Kanten (vom Typ `Edge`), welche wiederum die Knoten enthalten.

Dabei werden die Elemente vom Zugriff der Außenwelt abgeschirmt, nur zwei Schnittstellen stehen zur Verfügung. Die Funktionen `getNext` und `getCurrent`. Die Funktion `hasNext` ist eine Hilfsfunktion und kann zur Iteration über die Elemente der Graph-Klasse genutzt werden.

Über die Funktion `addResultSetArray` soll eine Schnittstelle zur Erfurt-API bereitgestellt werden. Sie bietet die Möglichkeit ein Erfurt-ResultSet, was als einfaches zweidimensionales Array vorliegt, einzulesen und die Elemente in eigene Kanten und Knoten umzuwandeln.

Neben den Kanten und Knoten stellt die Klasse noch zwei Gewichtungsalgorithmen bereit: `weightByColumnSum` und `weightByColumnHighest`. Diese können nicht direkt aufgerufen werden, sondern nur indirekt über die Funktion `weightAllEdges`. Sie soll die Vorarbeit des jeweiligen Gewichtungsalgorithmus koordinieren.

4.2.2 Die Klasse Edge

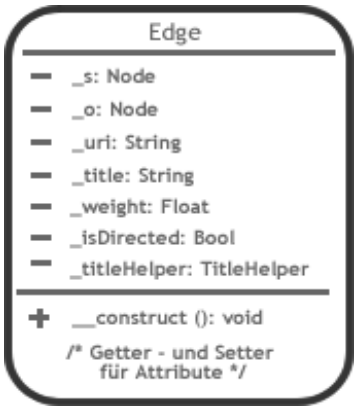

Abb. 10 – UML – Klassendiagramm der Klasse Edge

Diese Klasse steht für eine Kante zwischen zwei Knoten (Klasse `Node`). Sie stellt keine direkten Funktionalitäten bereit, sondern speichert nur Informationen über eine Kante und sowie die beiden Knoten, die sie verbindet. An Informationen wird sowohl die URI und der Titel der Kante abgelegt, sowie ob sie gerichtet ist oder nicht und die ihr zugewiesene Gewichtung.

4.2.3 Die Klasse Node

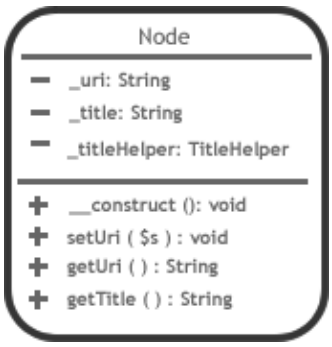

Abb. 11 – UML – Klassendiagramm der Klasse Node

Die Klasse `Node` repräsentiert einen Knoten in einem Graphen. Sie ist recht einfach gehalten und speichert die URI und den Titel eines Knotens. Man kann die URI direkt setzen, der Titel soll über das System ermittelt werden.

5 Implementierung und Evaluation

In diesem Kapitel geht es um die Vorstellung der entwickelten Ontologie und der Implementierung der Komponente. Es wird dabei nicht alles im vollen Umfang vorgestellt, sondern auf einige wenige Punkt konzentriert.

Im letzten Teil des Kapitels folgt die Evaluation der Gewichtungsalgorithmen. Dafür wird zuerst auf die Systemkonfiguration eingegangen und danach die einzelnen Algorithmen miteinander verglichen.

5.1 Vorstellung der Ontologie

Für die Entwicklung der Komponente war es essentiell eine Ontologie zu erstellen. Diese Ontologie ist klein gehalten und enthält eine Klasse und vier Objekteigenschaften.

5.1.1 Die OWL-Klasse WeightStatement

ObjectProperty	**Wert**
rdfs:label	„A Weight Statement“
skos:note	„This resource represents …“

Tab. 2 – Eigenschaften rdfs:label und skos:note von WeightStatement

Die beiden Property - Elemente in Tabelle 2 sind allgemeine Angaben über die Klasse. Sie stehen für den Titel und eine kurze Beschreibung.

ObjectProperty	**Wert**
a	owl:Restriction
owl:onProperty	weight
owl:cardinality	1 (xsd:int)
rdfs:range	xsd:decimal

Tab. 3 – subClassOf - Eigenschaft von WeightStatement

Diese Klasse besitzt eine `Property` namens `weight`, welche nur Gleitkomma-Werte annehmen kann und die Kardinalität ist auf genau eins beschränkt. In ihr wird der Wert einer Gewichtungsaussage abgelegt.

ObjectProperty	Wert
soic:has_owner	foaf:Person

Tab. 4 – Eigenschaft sioc:has_owner von WeightStatement

Wie bereits in den Anforderungen erwähnt, kann jede Gewichtungsaussage entweder einen Besitzer haben oder keinen. Besitzer sind, wenn sie denn gesetzt sind, damit automatisch auch eine Person im `foaf` – Kontext.

5.1.2 Die ObjectProperty subject

ObjectProperty	Wert
rdfs:domain	WeightStatement
rdfs:range	rdf:Resource
rdfs:label	„Subject mask“
skos:note	„Used for a subject in a statement (optional)“

Tab. 5 – Eigenschaften der ObjectProperty subject

Die `ObjectProperty subject` speichert das Subjekt einer Aussage ab. Dabei sei vorausgesetzt, dass das Subjekt eine RDF - Ressource ist. Jede Klasse welche diese `ObjectProperty` verwendet ist somit ein `WeightStatement`.

5.1.3 Die ObjectProperty predicate

ObjectProperty	Wert
rdfs:domain	WeightStatement
rdfs:range	rdf:resource
rdfs:label	„Predicate mask“
skos:note	„Used for a predicate in a statement (optional)“

Tab. 6 – Eigenschaften der ObjectProperty predicate

Die Beschreibung der `ObjectProperty predicate` ist äquivalent zu der von `ObjectProperty subject`.

5.1.4 Die ObjectProperty object

ObjectProperty	**Wert**
rdfs:domain	WeightStatement
rdfs:range	rdf:Resource
rdfs:label	„Object mask“
skos:note	„Used for an object in a statement (optional)“

Tab. 7 – Eigenschaften der ObjectProperty object

Die Beschreibung der `ObjectProperty object` ist äquivalent zu der von `Object-Property subject`.

5.1.5 Die ObjectProperty weight

ObjectProperty	**Wert**
rdfs:domain	WeightStatement
rdfs:range	rdf:Literal
rdfs:label	„Weight“
skos:note	„ A weight-value.“

Tab. 8 – Eigenschaften der ObjectProperty weight

Das `ObjectProperty weight` speichert die gesetzte Gewichtung ab. Dieser Wert ist vom Typ RDF–Literal. Jede Klasse welche diese `ObjectProperty` verwendet ist somit ein `WeightStatement`.

5.2 Die Benutzerschnittstelle

Die entwickelte Komponente ist wie bereits erwähnt auf das Grundlegende reduziert. Darunter zählen die Gewichtungsalgorithmen, das Modul zur Präsentation des Ergebnisgraphen und einige Hilfsfunktionen.

Die Oberfläche beschränkt sich hauptsächlich auf die Bereitstellung einer Eingabemaske für beliebige SPARQL – Kommandos und einen Bereich, auf dem der Ergebnisgraph visualisiert wird.

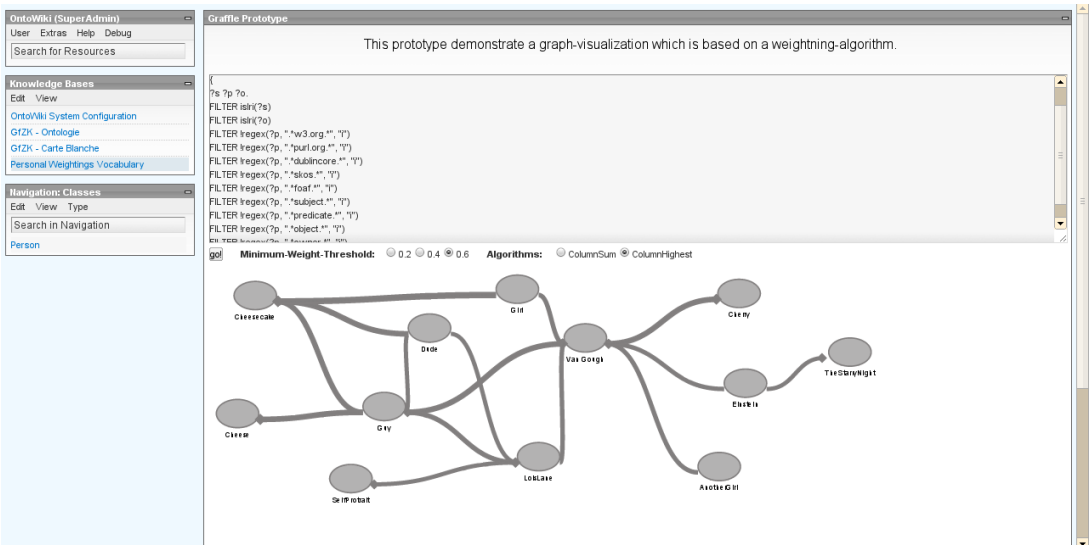

Abb. 12 – Screenshot der Benutzeroberfläche der Komponente

In Abbildung 12 sieht man die Eingabe-Shell und den Ergebnisgraphen. Der Ergebnisgraph wird nur durch Javascript und CSS dargestellt.

5.3 Fremde verwendete Bibliotheken

Die Komponente benutzt bei der Visualisierung des Ergebnisgraphen fremde Bibliotheken. Diese bieten einfache Möglichkeiten Knoten und die Verbindungen zwischen ihnen darstellen zu können. Es werden dafür die Bibliothek Raphaël [L7] und Dracula Graph Library [L8] genutzt.

Dracula Graph Library baut auf Raphaël auf. Diese wird um einige Hilfsfunktionen erweitert, um unter anderem die Erstellung von Kanten zwischen Knoten einfacher zu gestalten.

Die Klasse `Graph` besitzt die nötige Funktionalität, um entsprechenden Javascript-Code zu generieren, der anschließend an die Dracula Graph Library weitergeben wird, welche ihn dann visualisiert.

Eine Einbindung der beiden fremden Bibliotheken erfolgt über einen Eintrag im Kopf der HTML-Seite. Dies wird über Funktionen aus OntoWiki realisiert.

5.4 Die Gewichtungsalgorithmen

Im Folgenden werden zwei Gewichtungsalgorithmen skizziert und ihre Umsetzung in der Komponente erläutert. Diese Algorithmen stellen dabei den Hauptteil der eigentlichen Implementierungsarbeit. Zuerst wird eine Übersicht der Vorgehensweise gegeben und danach konkret die Implementierung analysiert.

Es wurde darauf Wert gelegt, dass man ein Verständnis von der Arbeitsweise der Algorithmen bekommt. Wie bereits erwähnt, ist es außer den vorgestellten Paradigmen bei jedem Gewichtungsalgorithmus freigestellt, wie er implementiert wird. Um das Verständnis für die beiden exemplarisch erstellten Algorithmen zu erhöhen, wird deren Gemeinsamkeit in der Berechnung einmal kurz vorgestellt.

Nochmal die Übersicht der Möglichkeiten bei der Kombination der Elemente Subjekt, Prädikat und Objekt bei der Stufe eins bis drei:

	1. Stufe	**2. Stufe**	**3. Stufe**
Möglichkeiten	3	3	1
Gesetzte Elemente	S, P oder O	SP, PO oder SO	nur SPO

Tab. 1 – Auflistung Gewichtungsstufen und Kombinationsmöglichkeiten

Beide Algorithmen fragen zu jedem Tripel, das gewichtet werden soll, die vorhandenen Gewichtungsaussagen ab. Bei jeder Gewichtungsaussage wird die Stufe gespeichert und bildlich gesehen der gesetzte Wert an die Stelle des jeweiligen Elementes, wie Subjekt, Prädikat und Objekt, gesetzt. Dieses Vorgehen wird zum besseren Verständnis an einem Beispiel demonstriert.

5.4.1 Demonstration der Vorgehensweise

Sei folgende Gewichtungsaussage gegeben:

	<subject>	**<predicate>**	**<object>**	**<weight>**
Gewichtungsaussage	<foo>	<bar>	-	0.5

Tab. 9 – Beispiel einer Gewichtungsaussage

In ihr sind das Subjekt und Prädikat gesetzt sowie eine Gewichtung von `0.5`. Das Vorgehen erfolgt dadurch, dass dem Subjekt und Prädikat jeweils der Wert `0.5` dieser Gewichtungsaussage zugeordnet wird. Das nicht gesetzte Element `<object>` bekommt einen neutralen Wert für die Berechnung, in dem Falle den Wert `0`.

<subject>	**<predicate>**	**<object>**
0.5	0.5	0

Tab. 10 – Umgewandelte Gewichtungsausage

In Darstellung in Tabelle 10 ist zu sehen, wie die Daten später als Datenstruktur abgelegt werden können. Das Ablegen erfolgt in Form eines Arrays.

5.5 Gewichtungsalgorithmus: Spaltenaufsummierung

5.5.1 Allgemeine Informationen

Bei diesem Gewichtungsalgorithmus wird über die spaltenweise Aufsummierung die Endgewichtung gebildet. Wie bereits im letzten Abschnitt vorgestellt, geht diesem Algorithmus einige Vorarbeit voraus. Diese besteht unter anderem darin, die vorhandenen Gewichtungsaussagen eines Tripel abzufragen und die Wertungen aufzubereiten. Nachdem alle Wertungen aufbereitet wurden, werden für jedes einzelne Tripel die Spalten für Subjekt, Prädikat und Objekt durchlaufen und die enthaltenen Werte aufsummiert und das arithmetische Mittel gebildet.

5.5.2 Eigenheiten des Algorithmus

Jede Gewichtungsaussage geht zu gleichen Teilen in die Endgewichtung ein. Zusätzlich wird am Ende noch das arithmetische Mittel gebildet. Damit wird eine recht unpräzise Gewichtung, also eine Gewichtung der ersten oder zweiten Stufe, genauso aussagekräftig wie eine der Stufe 3, welche sehr präzise ist. Das ist in solchen Situationen nützlich, in denen man allgemeineren Aussagen die gleiche Aussagekraft zusichern möchte, wie den sehr präzisen Aussagen.

5.5.3 Vorgehen als Pseudocode

Es folgt nun eine Pseudocode-Darstellung der Vorgehensweise des Algorithmus.

```
function weightByColumnSum ( Gewichtungsaussagen )

[1.] while Wenn nächste Kante gesetzt ist then
[2.]  foreach Durchlaufe alle Gewichtungsaussagen
[3.]     if Stufe 3 Aussage über aktuelle Kante gefunden
[4.]       then Setze Wert für Spalte s, p und o in item
[5.]         stage3 [] = item
[6.]     elseif Stufe 2 Aussage über aktuelle Kante gefunden
[7.]       then Setze Wert für Spalten sp, po oder so in item
[8.]         item [z] = 0 /* z = ungesetzte Spalte */
[9.]     elseif Stufe 1 Aussage über aktuelle Kante gefunden
[10.]       then Setze Wert für Spalte s, p oder o in item
[11.]         item [y] = item [z] = 0 /* y und z ungesetzt */
[12.] if stage3 hat Elemente then
[13.]   Einfügen aller s-Einträge zu s_weights, p-Einträge zu
[14.]   p_weights und o-Einträge zu o_weights
[15.] /* Das Gleiche für Elemente von stage2 und stage1 */
[16.] foreach Durchlaufe s_weights Werte und summiere sie auf
[17.] foreach Durchlaufe p_weights Werte und summiere sie auf
[18.] foreach Durchlaufe o_weights Werte und summiere sie auf
[19.] Berechne arithmetisches Mittel aller Summen
[20.] ENDGEWICHTUNG = Arithmetisches Mittel
```

Code 2 – Pseudocode für Algorithmus Spaltenaufsummierung

5.6 Gewichtungsalgorithmus: Spaltenmaximum

5.6.1 Allgemeine Informationen

Bei diesem Algorithmus wird das Maximum aller Werte einer konkreten Spalte für die Berechnung des Endgewichtes genutzt. Der Verlauf der Berechnung ähnelt dabei stark dem des vorherigen Gewichtungsalgorithmus der Spaltenaufsummierung.

5.6.2 Eigenheiten des Algorithmus

Bei diesem Algorithmus geht nur das jeweils höchste Gewicht der Spalten S, P und O aller Gewichtungsaussagen eines Tripels in das Endgewicht ein. Damit werden nur die höchsten gesetzten Werte betrachtet, unabhängig davon auf welcher Stufe die Gewichtungsaussage war. Auf diese Weise werden unpräzise Gewichtungsaussagen, solche der Stufe 1 und 2, wenn sie die höchsten Werte haben, noch viel stärker in die Berechnung des Endgewichtes einbezogen, als noch bei dem Algorithmus mit der Spaltenaufsummierung.

Dieser Algorithmus ist nützlich, wenn man einzelnen Aussagen einen höheren Stellenwert bemessen möchte, egal um welche Stufe es sich handelt. Auf diese Weise wird deren Aussagekraft nicht von anderen Gewichtungsaussagen relativiert.

Statt dem Nutzen des Maximums wäre auch das Nutzen des Minimums denkbar.

5.6.3 Vorgehen als Pseudocode

Es folgt nun eine Pseudocode-Darstellung der Vorgehensweise des Algorithmus.

```
function weightByColumnHighest ( Gewichtungsaussagen )

[1.] while  Wenn nächste Kante gesetzt ist then
[2.]  foreach Durchlaufe alle Gewichtungsaussagen
[3.]     if Stufe 3 Aussage über aktuelle Kante gefunden
[4.]       then Setze Wert für Spalte s, p und o in item
[5.]         stage3 [] = item
[6.]     elseif Stufe 2 Aussage über aktuelle Kante gefunden
[7.]       then Setze Wert für Spalten sp, po oder so in item
[8.]         item [z] = 0 /* z = ungesetzte Spalte */
[9.]     elseif Stufe 1 Aussage über aktuelle Kante gefunden
[10.]       then Setze Wert für Spalte s, p oder o in item
[11.]         item [y] = item [z] = 0 /* y und z ungesetzt */
[12.] if stage3 hat Elemente then
[13.]    Einfügen aller s-Einträge zu s_weights, p-Einträge zu
[14.]    p_weights und  aller o-Einträge zu o_weights
[15.]    /* Das Gleiche für stage2 und stage1 */
[16.] Ermittle Maximumwert aus s_weights, p_weights und o_weights
[17.] Berechne arithmetisches Mittel aller gesetzten Maximumwerte
[18.] ENDGEWICHTUNG = Arithmetisches Mittel
```

Code 3 – Pseudocode für Algorithmus Spaltenmaximum

5.7 Weitere Funktionalitäten

In diesem Abschnitt wird auf die anderen Funktionalitäten eingegangen. Sie spielen eine eher untergeordnete Rolle im Vergleich zu den Gewichtungsalgorithmen, sind aber dennoch essentiell bei der Interaktion mit den Daten.

5.7.1 Einfügen von Elementen aus einem SPARQL – Ergebnis

Die Funktion `addResultSetArray` bietet die Möglichkeit, aus einem vorher ausgeführten SPARQL - Query die Ergebnismenge in den Datenbestand der Graph-Instanz aufzunehmen:

```
function addResultSetArray ( Ergebnismenge )

[1.] if Ergebnismenge ist kein Array oder ist leer then Abbruch
[2.] foreach Durchlaufe Elemente in Ergebnismenge
[3.]    $subjekt = new Node /* Erstelle Subjekt-Knoten */
[4.]    $subjekt->setUri ( $aktuellesElement [ 's' ] )
[5.]    $objekt = new Node /* Erstelle Objekt-Knoten */
[6.]    $objekt->setUri ( $aktuellesElement [ 'o' ] )
[7.]    $kante = new Edge /* Erstelle Kante */
[8.]    $kante->setS ($subjekt) /* Füge der Kante $subjekt zu */
[9.]    $kante->setUri ( $aktuellesElement [ 'p' ] )
[10.]   $kante->setO ( $objekt )  /* Füge der Kante $objekt zu */
[11.]   $this->addEdge ( $kante )
```

Code 4 – Pseudocode für Funktion addResultSetArray

Im Code 4 sieht man, als Pseudocode dargestellt, die Funktionsweise. Die Funktion ist recht simpel aufgebaut, zuerst wird die Ergebnismenge auf Inhalt geprüft und danach wird der Inhalt durchlaufen. Dabei werden immer zwei Knoten und eine Kante angelegt, ob sie in der Abfrage gesetzt wurden oder nicht.

Die einzelnen Funktionen wie `setUri` prüfen eigenständig nochmal die Validität der Daten. Fehlerhafte Elemente werden einfach ignoriert und stattdessen wird das angelegte Objekt im Ausgangszustand belassen. Auf diese Weise erhält man immer eine Liste von Kanten mit zwei Elementen.

5.7.2 Konvertierung der Graph-Elemente ist darstellbaren Javascript-Code

Die Funktion `toGraphDraculaGraphJsCode` erzeugt aus der gespeicherten Menge an Kanten und Knoten äquivalenten Javascript-Code, welche an die entsprechenden Instanzen der *Dracula Graph Library* übergeben wird.

Der überwiegende Teil des Codes enthält einfache Kommandos zur Erzeugung der nötigen Instanzen für die Dracula Graph – Bibliothek. Daher wird sich im Nachhinein nur auf die Generierung des äquivalenten Javascript-Code konzentriert.

```
function toGraphDraculaGraphJsCode ( Ergebnismenge )

[1.] [ ... ]
[2.] foreach Generiere für jede Kante den folgenden JS-Code
[3.]    g.addEdge (
[4.]      <?php $kante->subjectTitle () ?>,
[5.]      <?php $kante->objectTitle () ?>,
[6.]      {
[7.]         directed: ''<?php $kante->directed () ?>'',
[8.]         fill: ''#D2D2D2| <?php $kante->getWeight()*10 ?>'',
[9.]         label: '' ''
[10.]     }
[11.]    );
[12.] [ ... ]
```

Code 5 – Pseudocode für Funktion toGraphDraculaGraphJsCode

Die Funktion erzeugt einen Javascript-Code-Block und speichert ihn als `string` ab. Nach Abschluss der Generierung wird dieser von der Funktion zurückgegeben. Die Komponente fügt diesen Rückgabewert dann per `echo`-Ausgabe in das Formular ein und erstellt damit den Ergebnisgraphen.

5.8 Vorbereitungen der Evaluation

Dieser Abschnitt behandelt die Evaluation der Algorithmen. Dabei geht es im Vordergrund darum, die unterschiedlichen Ergebnisgraphen kritisch zu betrachten und die Unterschiede zu analysieren. Wie im Vorfeld bereits erwähnt, gibt es keinen optimalen Gewichtungsalgorithmus. Die Situationen, in denen ein Algorithmus eingesetzt wird, bestimmen seine Merkmale. Die entwickelten und im Nachhinein vorgestellten Algorithmen werden daher auf ihre Einsatzfähigkeit und ihre Unterschiede untersucht.

5.8.1 Konfiguration der Testumgebung

Zur Konfiguration des Systems und der zu entwickelnden Komponente gibt es keine speziellen Anforderungen. Es wird eine funktionsfähige OntoWiki - Installation vorausgesetzt und eine lauffähige Version dieser Komponente. Zum Betrachten des Ergebnisgraphen wird ein zeitgemäßer Browser benötigt, um die SVG - Elemente darstellen zu können.

5.8.2 Testdatenbestand

Zu besseren Demonstration der Algorithmen wurde ein Datenbestand mit Testtripel angelegt. Die Tripel liegen in Notation3 – Code vor. Es besteht die Möglichkeit einen eigenen Datenbestand mit Gewichtungsaussagen aufzubereiten und dann abzufragen. Aber für den Einstieg wird empfohlen die Testtripel als Wissensbasis in das OntoWiki einzubinden und abzufragen. Die Menge der Testdatensätze beläuft sich auf wenige Elemente, um die Übersichtlichkeit zu waren.

5.8.3 Verwendete SPARQL – Abfrage

Die SPARQL – Abfrage um die Wissensbasis abzufragen sieht folgendermaßen aus:

```
[1.] SELECT ?s ?p ?o
[2.] WHERE
[3.] {
[4.]    ?s ?p ?o.
[5.]    FILTER isIri ( ?s ).
[6.]    FILTER isIri ( ?o ).
[7.]    FILTER !regex ( ?p, ".*w3.org.*", "i" ).
[8.]    FILTER !regex ( ?p, ".*purl.org.*", "i" ).
[9.]    FILTER !regex ( ?p, ".*dublincore.*", "i" ).
[10.]   FILTER !regex ( ?p, ".*skos.*", "i" ).
[11.]   FILTER !regex ( ?p, ".*foaf.*", "i" ).
[12.]   FILTER !regex ( ?p, ".*subject.*", "i" ).
[13.]   FILTER !regex ( ?p, ".*predicate.*", "i" ).
[14.]   FILTER !regex ( ?p, ".*object.*", "i" ).
[15.]   FILTER !regex ( ?p, ".*owner.*", "i" )
[16.] }
```

Code 6 –Verwendete SPARQL – Abfrage während der Evaluation

Es wird Subjekt, Prädikat und Objekt der Tripel abgefragt. Weiterhin werden verschiedene Filter auf das Prädikat gesetzt, um Tripel auszuschließen, welche Metadaten-Elemente beinhalten. Darunter fallen Tripel aus den Konventionssammlungen wie etwa Dublin Core metadata [13] oder foaf [14].

5.8.4 Einblick in den Testdatenbestand

Zur besseren Demonstration der Algorithmen wurde ein Datenbestand mit Testtripel angelegt. Die Tripel liegen in Notation3 – Code vor. Es besteht zwar die Möglichkeit einen eigenen Datenbestand mit Gewichtungsaussagen aufzubereiten und dann abzufragen.

Aber für den Einstieg wird empfohlen die vorliegenden Testtripel als Wissensbasis in OntoWiki einzubinden und abzufragen. Die Menge der Test - Datensätze beläuft sich auf ungefähr 20 Elemente, inklusive Verknüpfungen untereinander. Das hat den Hintergrund, dass mehr Elemente mit entsprechenden Verbindungen auf der Benutzeroberfläche nicht so übersichtlich darstellbar wären. Bei weniger Elemente wäre zwar die Darstellung einfacher, aber das wäre nicht sehr praxisnah und verdeutlicht in dieser Form die Arbeitsweise der jeweiligen Gewichtungsalgorithmen nicht so gut.

Im Folgenden wird konkret auf die einzelnen Daten eingegangen. Es liegen grob gesehen zwei Arten von Elementen vor. Zum einen ganz einfach gehaltene Elemente, die größtenteils nur einen Titel tragen, zum anderen Gewichtungsaussagen über diese Elemente und ihre Verknüpfungen untereinander. Bei der Erstellung des Datenbestandes wurde darauf Wert gelegt, dass die Elemente den Einstieg in die Thematik erleichtern und die unterschiedlichen Anwendungsfälle gut darstellen können.

Aus diesem Grund wurde bei den normalen Elementen oft nur ein Titel und wenige Verbindungen gesetzt. Mit Blick auf das Visualisierungsmodul sei erwähnenswert, dass es zur Zeit noch recht ungeeignet ist, größere Mengen von Elementen inklusive ihrer Verbindungen darzustellen. Damit ist die Anordnung der Elemente und ihrer Verbindungen gemeint, welche bei zunehmender Zahl unübersichtlicher werden. Um dem vorzubeugen wurde die Menge der Elemente und Verbindungen gering gehalten.

Die Titel der Elemente wurden frei gewählt und dienen lediglich der Veranschaulichung. Ebenso hätte man alle Elemente von A bis Z durchnummerieren können, was aber dem Verständnis nicht zuträglich gewesen wäre.

5.8.5 Übersicht der Elemente des Testdatenbestands

Element	***like***	***love***	***know***	***paint***	***consistsOf***
Dude	Cheesecake, Cheese	LoisLane	Guy, Girl	-	-
Guy	Cheesecake, VanGough, Cheese	LoisLane	Girl	-	-
LoisLane	SelfProtrait, VanGough	-	-	-	-
Girl	Cheesecake, VanGough	-	Guy	-	-
AnotherGirl	Cheesecake, VanGough, Cheese	-	-	-	-
VanGough	Cheesecake, Cherry, Cheese, Cake	-	-	SelfProtrait, Courtesan, TheNightCafe, TheStarryNight, TheSower	-
Einstein	Cheesecake, TheStarryNight, Cake, Cherry	-	VanGough	-	-
Superman	-	-	LoisLane	-	-
Cheesecake	-	-	-	-	Cheese, Cake
Cheese	-	-	-	-	-
Cake	-	-	-	-	-
SelfProtrait	-	-	-	-	-
Courtesan	-	-	-	-	-
TheNightCafe	-	-	-	-	-
TheStarryNight	-	-	-	-	-
TheSower	-	-	-	-	-

Tabelle 11 – Übersicht der Elemente des Testdatenbestands

In Tabelle 11 ist eine Liste aller Elemente und ihrer Verknüpfungen untereinander aufgelistet, ausgenommen die Gewichtungsaussagen. Diese Übersicht soll dem Verständnis dienen.

Gewichtungsaus-sage	***<subject>***	***<predicate>***	***<object>***	***<weight>***
ex1	LoisLane	love	Superman	0.9
ex18	LoisLane	like	SelfProtrait	0.6
ex6	Dude	know	Guy	0.6
ex7	Dude	know	Girl	0.4
ex8	Dude	like	Cheesecake	0.8
ex26	Dude	love	LoisLane	0.6
ex9	Guy	love	LoisLane	0.7
ex13	Guy	like	Cheesecake	0.2
ex14	Guy	like	VanGough	0.6
ex16	Guy	know	Girl	0.3
ex22	Guy	like	-	0.8
ex15	Girl	like	Cheesecake	0.9
ex25	Girl	know	Guy	0.5
ex21	Einstein	like	Cake	0.4
ex24	Einstein	like	Cherry	0.1
ex23	VanGough	like	Cherry	0.8
ex19	VanGough	-	-	0.5
ex20	Einstein	-	-	0.7
ex2	-	like	-	0.5
ex3	-	know	-	0.4
ex4	-	paint	-	0.15
ex5	-	consistsOf	-	0.1
ex10	-	-	VanGough	0.8
ex11	-	-	Einstein	0.5
ex12	-	-	Cheesecake	0.1
ex17	-	-	Superman	0.2

Tabelle 12 – Übersicht der Gewichtungsaussagen

In Tabelle 12 wird eine Übersicht über die einzelnen Gewichtungsaussagen gegeben. Dabei wurden diese nach dem Subjekt gruppiert und nach der Stufe absteigend sortiert.

5.9 Evaluation der Gewichtungsalgorithmen

Im Folgenden wird anhand eines Szenarios die Arbeitsweise und die Visualisierung der Algorithmen demonstriert. Dazu wird zuerst eine Übersicht der Testdaten gezeigt, danach werden für die beiden Gewichtungsalgorithmen Spaltenaufsummierung und

Spaltenmaximum jeweils die Berechnungen vorgeführt, mit der Einschränkung, dass alle Endgewichtungen größer `0.2` sein müssen.

Im Verlauf der Berechnungen wird ersichtlich, dass durch diese Einschränkung keine Tripel aussortiert werden. Später jedoch, nach einer Erhöhung der Einschränkung auf `0.4` bzw. `0.6`, fallen Tripel weg. Auf diesen Sachverhalt wird später noch einmal genauer eingegangen.

Im Anschluss den Berechnungen folgt eine Visualisierung der Testdaten und mit den berechneten Gewichtungen unter Setzung von drei Gewichtungsgrenzen. Diese Grenzen werden in drei Stufen erhöht und sollen damit eine Verfeinerung der Datenbestände durch Wegfall von Elementen und Verbindungen veranschaulichen.

5.9.1 Ausgewählter Testdatenbestand

	Gewichtungsaussage	***<subject>***	***<predicate>***	***<object>***	***<weight>***
Datensatz 1	**ex24**	Einstein	like	Cherry	0.1
	ex20	Einstein	-	-	0.7
	ex2	-	like	-	0.5
Datensatz 2	**ex13**	Guy	like	Cheesecake	0.7
	ex22	Guy	like	-	0.8
	ex2	-	like	-	0.5
	ex12	-	-	Cheesecake	0.1

Tabelle 13 – Ausgewählte Testdaten, Datensatz 1 und 2

Tabelle 13 ist eine Auflistung der verwendeten Testdaten. Diese sind aufgeteilt in zwei Datensätze. Jeder Datensatz ist einer konkrete Aussage zugeordnet.

Datensatz 1 `<Einstein>` `<like>` `<Cherry>`

Bei diesem Datensatz gibt es eine sehr präzise Stufe 3 Aussage (ex24), welche die Gewichtung auf `0.1` setzt. Die anderen beiden Stufe 1 Aussagen (ex20, ex2) sind sehr unspezifiziert. Der Benutzer wollte mit der Stufe 3 Aussage ausdrücken, dass Einstein keine Kirschen mag.

Datensatz 2	`<Guy>`	`<like>`	`<Cheesecake>`

Es liegt eine sehr präzise Stufe 3 Aussage (ex13) vor, welche aussagt, dass ein Typ Käsekuchen mag. Weiterhin gibt es eine Stufe 2 Aussage, welche alle Tripel mit `<Guy>` und `<like>` betrifft und diese mit `0.8` gewichtet. Eine ähnliche Aussage traf der Benutzer bei der Stufe 1 Aussage über das `<like>`, welcher er eine Gewichtung von `0.5` gab. Damit wäre die Gewichtung der Stufe 3 Aussage durch die anderen beiden bestätigt.

5.9.2 Die durchgeführten Berechnungen

Nun folgen durch die Anwendung des Algorithmus: `Spaltenaufsummierung` folgende Tabellen:

	Aufsummierung	***Arithmetisches Mittel***	***Zwischensumme***
<subject>	0.1 + 0.7	0.8 / 2	0.4
<predicate>	0.1 + 0.5	0.6 / 2	0.3
<object>	0.1	0.1	0.1

Tabelle 14 – Anwendung Spaltenaufsummierung auf Datensatz 1

	Aufsummierung	***Arithmetisches Mittel***	***Zwischensumme***
<subject>	0.7 + 0.8	1.5 / 2	0.75
<predicate>	0.7 + 0.8 + 0.5	2 / 3	0.67
<object>	0.7 + 0.1	0.8 / 2	0.4

Tabelle 15 – Anwendung Spaltenaufsummierung auf Datensatz 2

Nun eine Aufschlüsselung der Endgewichtungen für Datensatz 1 und 2:

	Arithmetisches Mittel	***Endgewichtung***
Datensatz 1	(0.4 + 0.3 + 0.1) / 3	0.2667
Datensatz 2	(0.75 + 0.67 + 0.4) / 3	0.6067

Tabelle 16 – Spaltenaufsummierung, Endgewichtungen Datensatz 1 und 2

Und das Gleiche nochmal für den Algorithmus: `Spaltenmaximum`:

	Spaltenmaxima
<subject>	0.7
<predicate>	0.5
<object>	0.1

Tabelle 17 – Anwendung Spaltenmaximum auf Datensatz 1

	Spaltenmaxima
<subject>	0.8
<predicate>	0.8
<object>	0.7

Tabelle 18 – Anwendung Spaltenmaximum auf Datensatz 2

Nun eine Aufschlüsselung der Endgewichtungen für Datensatz 1 und 2 sowie eine Gesamtübersicht aller berechneten Endgewichtungen:

	Arithmetisches Mittel	***Endgewichtung***
Datensatz 1	(0.7 + 0.5 + 0.1) / 3	0.4333
Datensatz 2	(0.8 + 0.8 + 0.7) / 3	0.7667

Tabelle 19– Spaltenmaximum, Endgewichtungen Datensatz 1 und 2

	Daten	***Endgewichtung***
Spaltenaufsummierung	Datensatz 1	0.2667
	Datensatz 2	0.6067
Spaltenmaxima	Datensatz 1	0.4333
	Datensatz 2	0.7667

Tabelle 20 - Gesamtübersicht der Endgewichtungen

Für diese Berechnungen wurde eine Einschränkung der Endgewichtungen von `0.2` gesetzt. Es liegen alle Endgewichtungen über dieser unteren Grenze, wodurch es keine Änderungen in der Visualisierung gibt.

5.9.3 Untere Grenzen bei Gewichtungen

Im folgenden eine Übersicht mit den unteren Grenzen bei der Endgewichtung:

Einschränkung	Algorithmus	Daten	*Endgewichtung*
0.2	Spaltenaufsummierung	Datensatz 1	0.2667
		Datensatz 2	0.6067
	Spaltenmaximum	Datensatz 1	0.4333
		Datensatz 2	0.7667
0.4	Spaltenaufsummierung	Datensatz 2	0.6067
	Spaltenmaximum	Datensatz 1	0.4333
		Datensatz 2	0.7667
0.6	Spaltenaufsummierung	Datensatz 2	0.6067
	Spaltenmaximum	Datensatz 2	0.7667

Tabelle 21 – Übersicht der unteren Grenzen bei Endgewichtungen

5.9.4 Visualisierung der Gewichtungsberechnungen

Es folgt nun eine Darstellung des Testdatenbestandes mit einer Hervorhebung der beiden Testdatensätze. Dabei wird eine Unterteilung nach den beiden Gewichtungsalgorithmen vorgenommen und diese wiederum nochmal unterteilt in die drei Einschränkungsgrenzen. Damit werden 6 Grafiken gezeigt und erläutert.

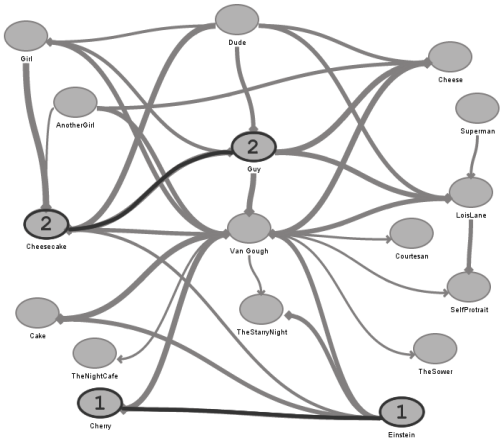

Abb. 13 – Spaltenmaximum, Grenze 0.2

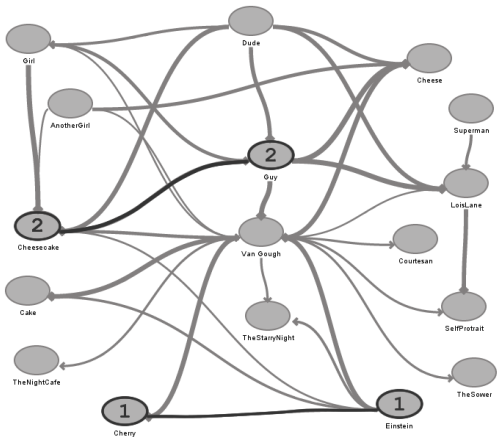

Abb. **14 –** Spaltenaufsummierung, Grenze 0.2

Auf der Abbildung 13 und 14 sieht man jeweils eine verkleinerte Version des Ergebnisgraphen der SPARQL-Abfrage aus Kapitel 5.8.3. Auf den Abbildungen wurden jeweils beide Datensätze mit der zugehörigen Nummer markiert und farblich hervorgehoben. Die verkleinerten Vorschaubilder verhindern womöglich eine genauere Betrachtung der einzelnen Elemente, ihrer Verbindungen und deren Linienstärken. Dabei ist zu erwähnen, dass hierbei die Graphen als Ganzes betrachtet werden.

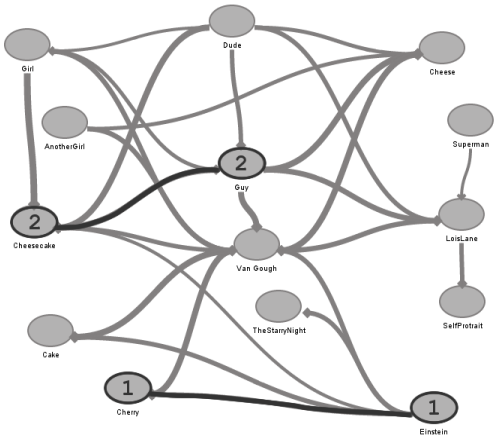

Bild 15 – Spaltenmaximum, Grenze 0.4

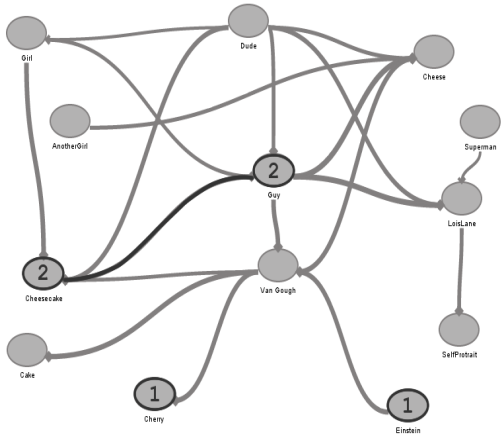

Bild 16 – Spaltenaufsummierung, Grenze 0.4

Auf den Bildern 15 und 16 sieht man reduzierte Ergebnisgraphen. Die Abfrage ist die Gleiche wie bei den beiden Graphen auf Bild 13 und 14, es wurde nur die Grenze für die zulässige Endgewichtung von `0.2` und `0.4` erhöht. Dadurch fallen einige Verbindungen weg und somit auch Elemente, welche keinerlei Verbindungen haben.

Datensatz 1 besaß eine Endgewichtung bei der Spaltenaufsummierung von `0.2667`, weshalb diese Verbindung nun nicht mehr vorhanden ist.

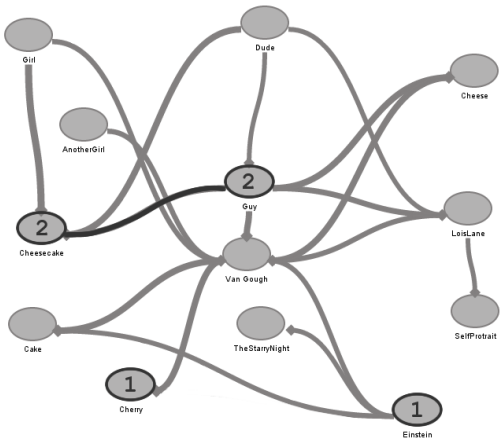

Bild 17 – Spaltenmaximum, Grenze 0.6

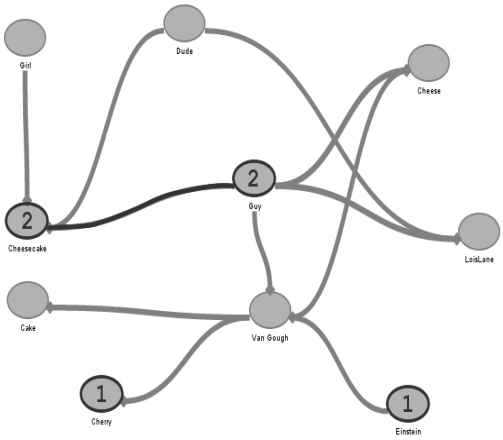

Bild 18 – Spaltenaufsummierung, Grenze 0.6

Die letzten beiden Bilder zeigen die Graphen nach einer Erhöhung der Grenze der Endgewichtung von `0.4` auf `0.6`. Auf dem linken Bild ist nun ebenfalls die Verbindung bei Datensatz 1 weggefallen.

5.9.5 Schlussfolgerungen

Die gesetzten Aussagen für Datensatz 1 und 2 wurden nur teilweise bestätigt. Bei Datensatz 1 wurde mit der präzisen Stufe 3 Aussage ausgesagt, dass Einstein keine Kirschen mag. Die Gewichtung lag bei einem Wert von `0.1`. Der Algorithmus der Spaltenaufsummierung hat dies größtenteils bestätigt, indem er eine Endgewichtung von `0.2667` berechnete. Der Algorithmus des Spaltenmaximum hat hingegen einen Wert von `0.4333` berechnet, welcher sehr stark von der Intention des Benutzers während der Gewichtung abweicht.

Datensatz 2 wurde von beiden Algorithmen ähnlich bewertet. Die präzise Stufe 3 Bewertung setzte eine Gewichtung von `0.7`. Der Algorithmus der Spaltenaufsummierung hat einen Endwert von `0.6067` berechnet, der andere `0.7667`. Damit lagen bei dieser Aussagekonstellation die berechneten Endgewichtungen im Rahmen der Erwartung.

Wie in den Abbildungen 13 bis 18 zu sehen, reduziert sich die Anzahl der Elemente bei Veränderung der Grenze für die Endgewichtungen teilweise erheblich. Das wäre zum Beispiel in für Explorationstools verwendbar. Man bietet dem Betrachter am Anfang nur wenige Daten und er kann durch Verringerung der Grenze die Anzahl der anzuzeigenden Daten eigenständig anpassen. Dies entspräche dann einer Top–Down-ähnlichen Herangehensweise bei der Untersuchung von Informationen, die in Form von Graphen abgelegt wurden.

6 Fazit und Ausblick

6.1 Fazit

Das Ziel dieser Bachelorarbeit war es, Gewichtungsalgorithmen zu implementieren und evaluieren. Dabei wurden Empfehlungssysteme und ihre Eigenschaften als Grundlage genommen. Im Laufe der Konzeption und Implementierung wurden verschiedene Dinge festgestellt. Zum Beispiel, dass es recht anspruchsvoll ist, effektive Gewichtungsalgorithmen für die jeweiligen Situationen zu finden. Dafür gab es verschiedene Gründe. Der Hauptgrund ist die sich, je nach Situation, unterscheidende Anforderungslage, auf die immer wieder eingegangen werden muss. Weitere Gründe sind der spätere Verwendungszweck und die Behandlung der einzelnen Gewichtungsaussagen.
Das Ziel der Bachelorarbeit wurde erreicht. Die Algorithmen sind verwendungsfähig und liefern nutzbare Ergebnisse. Weiterhin bietet das Visualisierungsmodul viel Potential für die weitere Verwendung in anderen Teilen des OntoWiki-Systems.
Die Umsetzungen, sowohl der Ontologie als auch die Implementierung der Komponente, weisen teilweise noch Defizite auf. So ist es bei dieser Implementierung nicht zu verhindern, dass es mehr Gewichtungsaussagen einer Stufe gibt, als es laut Paradigmen vorgesehen ist. Oder wie man mehrfach gesetzte Subjekt-, Prädikat- oder Objektwerte in einer Gewichtungsaussage behandelt.

6.2 Ausblick

Es ist im Rahmen des OntoWiki-Projektes geplant, die Komponente und besonders deren Visualisierungsmodul noch einmal komplett zu überarbeiten. An der Stelle, wo in ausführbaren Javascript-Code konvertiert wird, ist nun geplant, nicht direkt ausführbaren Code zur Darstellung zu generieren, sondern eine Objektstruktur in Javascript zu implementieren, welche es erlaubt, unter anderem Events an Elemente in der Objektstruktur zu hängen oder deren Aussehen zu verändern.
Ein anderer Punkt ist die Implementierung weiterer Gewichtungsalgorithmen. Wie bereits in der Einleitung erwähnt, wurden hier Dinge aufgegriffen und analysiert, die in einem anderen Projekt namens *europa*^{N} weiter benutzt werden sollen. Dazu gehören primär die Gewichtungsalgorithmen und die Visualisierungskomponente. Im Rahmen dessen werden effektivere Gewichtungsalgorithmen benötigt, welche zur Laufzeit unter anderem konfigurierbar sein müssen und auch bei großen Datenmengen gut skalieren.

7 Abkürzungen und Begriffserklärungen

Abkürzungen

Abkürzung	Titel
URI	Uniform Resource Identifier
RDF	Resource Description Framework
XML	Extensible Markup Language
OWL	Web Ontology Language
N3	Notation 3
SPARQL	SPARQL Protocol and RDF Query Language
UML	Unified Modeling Language
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
SVG	Scalable Vector Graphics
PHP	Hypertext Preprocessor

Begriffserklärungen

Begriff	Beschreibung
USE – CASE - Diagramm	Diagrammform aus der UML - Diagrammfamilie zur Demonstration des erwarteten Verhalten eines Systems [15]
Klassendiagramm	Diagrammform aus der UML - Diagrammfamilie zur Visualisierung von Klassen, ihrer Eigenschaften und Beziehungen untereinander [16]
Klasse (UML)	Ein komplexer Datentyp, welcher Eigenschaften und Funktionen besitzt
Klasse (OWL)	Ist ein Begriff, Konzept oder ein Individuum [17]
Blog - Eintrag	Ein Blog ist eine Art Online Tagebuch. Enthält oft Kommentierungen zu aktuellen Ereignissen, interessante Links zu anderen Webseiten oder Fotosammlungen.

8 Abbildungsverzeichnis

Nr.	Seite	Titel
1	6	ARD-ZDF-Onlinestudie 2000: Entwicklung der Online-Nutzung in Deutschland
2	7	Eintrag auf Digg.com mit 289 positiven Bewertungen (diggs)
3	9	Grafische Darstellung eines Tripels aus Subjekt, Prädikat und Objekt
4	13	USE CASE – Diagramm zur Gewichtungsvisualisierung
5	14	Grafische Darstellung einer Gewichtungsaussage
6	15	Grafische Darstellung eines Beispieltripels
7	15	Gewichtungsaussage über Beispieltripel mit Wert 0,8
8	17	UML – Klassendiagramm der Komponente
9	18	UML – Klassendiagramm der Klasse Graph
10	19	UML – Klassendiagramm der Klasse Edge
11	19	UML – Klassendiagramm der Klasse Node
12	23	Screenshot der Benutzeroberfläche der Komponente
13	39	Spaltenmaximum, Grenze 0.2
14	39	Spaltenaufsummierung, Grenze 0.2
15	40	Spaltenmaximum, Grenze 0.4
16	40	Spaltenaufsummierung, Grenze 0.4
17	41	Spaltenmaximum, Grenze 0.6
18	41	Spaltenaufsummierung, Grenze 0.6

9 Tabellenverzeichnis

Nr.	Seite	Titel
1	16, 24	Auflistung Gewichtungsstufen und Kombinationsmöglichkeiten
2	20	Eigenschaften rdfs:label und skos:note von WeightStatement
3	20	subClassOf - Eigenschaft von WeightStatement
4	21	Eigenschaft sioc:has_owner von WeightStatement
5	21	Eigenschaften der ObjectProperty subject
6	21	Eigenschaften der ObjectProperty predicate
7	22	Eigenschaften der ObjectProperty object
8	22	Eigenschaften der ObjectProperty weight
9	24	Beispiel einer Gewichtungsaussage
10	25	Umgewandelte Gewichtungsausage
11	34	Übersicht der Elemente des Testdatenbestands
12	35	Übersicht der Gewichtungsaussagen
13	36	Ausgewählte Testdaten, Datensatz 1 und 2
14	37	Anwendung Spaltenaufsummierung auf Datensatz 1
15	37	Anwendung Spaltenaufsummierung auf Datensatz 2
16	37	Spaltenaufsummierung, Endgewichtungen Datensatz 1 und 2
17	38	Anwendung Spaltenmaximum auf Datensatz 1
18	38	Anwendung Spaltenmaximum auf Datensatz 2
19	38	Spaltenmaximum, Endgewichtungen Datensatz 1 und 2
20	38	Gesamtübersicht der Endgewichtungen
21	39	Übersicht der unteren Grenzen bei Endgewichtungen

10 Codeverzeichnis

Nr.	Seite	Titel
1	9	Kurzes OWL Beispiel
2	26	Pseudocode für Algorithmus Spaltenaufsummierung
3	28	Pseudocode für Algorithmus Spaltenmaximum
4	29	Pseudocode für Funktion addResultSetArray
5	30	Pseudocode für Funktion toGraphDraculaGraphJsCode
6	32	Verwendete SPARQL – Abfrage während der Evaluation

11 Literaturverzeichnis und Weblinks

Nr.	Titel
[1]	ARD-ZDF-Online-Studie 2000: Gebrauchswert entscheidet über Internetnutzung (eingesehen am 18.08.2010): http://www.ard-zdf-onlinestudie.de/fileadmin/Online00/Online00_Nutzung.pdf
[2]	ARD/ZDF Onlinestudie 2009: Nachfrage nach Videos und Audios steigt weiter / Stand: 03.08.2009 (eingesehen am 18.08.2010): http://www.ard-zdf-onlinestudie.de/fileadmin/Online00/Online00_Nutzung.pdf
[3]	Heise-online Artikel vom 21.07.2003 (eingesehen am 18.08.2010): http://www.heise.de/newsticker/meldung/68-Millionen-Chinesen-sind-online-82525.html
[4]	Statistical Survey Report on Internet Development in China (January2010) / Seite 3 (eingesehen am 18.08.2010): http://www.cnnic.cn/uploadfiles/pdf/2010/3/15/142705.pdf
[5]	Bildschirmausschnitt der Suchmaske von digg.com vom 10.08.2010 nach Sucheingabe von „Semantic Web“
[6]	RDF / XML Syntax Specification / 10.02.2004 (eingesehen am 18.08.2010): http://www.w3.org/TR/REC-rdf-syntax/#section-Syntax-intro
[7]	Extensible Markup Language (XML) 1.0 (Fifth Edition) / 26.11.2008 (eingesehen am 18.08.2010): http://www.w3.org/TR/REC-xml/#sec-intro
[8]	OWL Web Ontology Language Overview / 10.02.2004 (eingesehen am 18.08.2010): http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1
[9]	OWL Web Ontology Language Guide / 10.02.2004 / Abschnitt 3.1.2. Individuals (eingesehen am 18.08.2010): http://www.w3.org/TR/2004/REC-owl-guide-20040210/#DefiningIndividuals
[10]	W3C Team Submission Notation3 (N3) (eingesehen am 18.08.2010): http://www.w3.org/TeamSubmission/n3/#intro
[11]	Vorlesung Semantic Web / Prof. Brewka und Dr. Auer / SS09 / Skript 7 - OWL - Syntax & Intuition / Folie 5 (eingesehen am 18.08.2010)
[12]	SPARQL Query Language for RDF (eingesehen am 18.08.2010): http://www.w3.org/TR/rdf-sparql-query/#introduction
[13]	Dublin Core metadata Basics (eingesehen am 18.08.2010): http://dublincore.org/metadata-basics/
[14]	The Friend of a Friend (FOAF) project (eingesehen am 18.08.2010): http://www.foaf-project.org/original-intro
[15]	USE-CASE-Diagramm (eingesehen am 18.08.2010): http://en.wikipedia.org/wiki/Use_case_diagram
[16]	Klassendiagramm (eingesehen am 18.08.2010): http://en.wikipedia.org/wiki/Class_diagram
[17]	Klassen im OWL-Kontext (eingesehen am 18.08.2010): http://en.wikipedia.org/wiki/Web_Ontology_Language#Classes
[18]	National Center for Supercomputing Applications: „In the beginning there was NCSA Mosaic....“: ftp://ftp.ncsa.uiuc.edu/Web/Mosaic/Windows/Archive/MosaicHistory.html

Zitate

Nr.	Titel
Z1	Vorlesung „Semantic Web“ / SS 09 / Prof. Brewka, Dr. Auer / Kapitel 7 „OWL - Syntax & Intuition “ / Folie 6 – Übersetzung von Gruber (1993)

Weblinks

Nr.	Titel
L1	Google Search: http://www.google.de
L2	Bing: http://www.bing.com
L3	Google Scholar: http://scholar.google.de/
L4	Google Books: http://books.google.de
L5	Amazon.com: http://www.amazon.com
L6	Digg: http://www.digg.com
L7	Raphaël / Version 1.4.7 : http://raphaeljs.com/
L8	Dracula Graph Library / Version 0.0.3 alpha : http://dracula.ameisenbar.de/

12 Quellcode und Ontologie

Der Quellcode für die Komponente und die erstellte Ontologie findet man im Repository von OntoWiki auf GoogleCode. Um die Daten herunterladen zu können benötigt man Mercurial, ein verteiltes Versionierungssystem. Die Installation und Inbetriebnahme werden im Wiki näher erläutert.

Quellcode

Die erstellte Komponente befindet sich unter folgender URL:
http://code.google.com/p/ontowiki/source/browse/?r=ac3c3c91140ec2416fda9ad0edec7f5504ae8ff7#hg/extensions/components/graffle

Verwendete Revision: *ac3c3c91140ec2416fda9ad0edec7f5504ae8ff7*
Verwendeter Branch: *Feature-QuickAdd*

Ontologie

Die erstellte Ontologie befindet sich unter folgender URL:
http://code.google.com/p/ontowiki/source/browse?repo=models#hg/PersonalWeightings

Verwendete Revision: *0c6e954c213f4370427a1f9f311680b9f6eec539*
Verwendeter Branch: *default*

13 Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Konrad Abicht Leipzig, August 2010