Verwendung von Unschärfe in semantischen Modellen

Seminararbeit
Technische Universität Dresden
Juni 2012

Jonas Schulz

Betreuer: Dipl.-Medieninf. Martin Voigt
Hochschullehrer: Prof. Dr.-Ing. Klaus Meißner

Fakultät Informatik
Institut für Software- und Multimediatechnik
Seniorprofessur für Multimediatechnik

Technische Universität Dresden
Fakultät Informatik

Aufgabenstellung für eine Seminararbeit

Name, Vorname des Studenten: Schulz, Jonas
Immatrikulationsnummer: 3573438

Thema: **Verwendung von Unschärfe in semantischen Modellen**

Zielstellung:

Ein Ziel des Topic/S-Projekts ist die semantische Modellierung von (Nachrichten-)Themen, die aus verschiedensten Nachrichtenkanälen im Newsroom einer Redaktion eintreffen. Dabei werden die Informationen aus verschiedenen Kanälen automatisch gesammelt, bereinigt, annotiert und in eine gemeinsame Datenbasis eingepflegt. Hierzu werden semantische Technologien wie RDFS zur Modellierung und SPARQL zur Abfrage der semantischen Daten eingesetzt. Diese W3C-Standards beruhen auf dem Paradigma, das binäre „100%-Relationen“ zwischen Ressourcen etabliert werden. Problematisch ist dies jedoch, wenn manche Relationen nicht mit hundertprozentiger Sicherheit definiert werden können. Ein Beispiel ist u. a. die in einem Text erkannte Entität „Jaguar“, die mit gewisser Wahrscheinlichkeit dem Konzept „Automobilmarke“, „Großkatze“ oder „Betriebssystem“ zugeordnet werden kann. Ein ähnliches Problem existiert bei der Modellierung der Glaubwürdigkeit von Informationen, die man zumeist nur unscharf spezifizieren kann.

Ziel dieser Arbeit ist die Untersuchung des Related Work hinsichtlich der Modellierung und Abfrage semantischer Informationen. Die gewonnenen Erkenntnisse sollen Basis für Folgearbeiten im Rahmen des Topic/S-Projektes sein. Hierzu ist zunächst ein Beispielszenario zu entwickeln, das die Probleme und Ziele bei der Modellierung semantischer Medieninformationen im Detail aufzeigt. Auf dessen Basis sind anschließend Kriterien für die Untersuchung des Related Work zu definieren. Für diese sind u.a. folgende Fragestellungen relevant: Modelliert man Unschärfe qualitative oder quantitativ? Welche Möglichkeiten bieten existierende Sprachmittel, z. B. die Definition von n-ary Relations? Welche Quasi-Standards, z. B. Fuzzy OWL oder N-Quads, kann man nutzen und werden diese von aktuellen Frameworks unterstützt?

Betreuer:	Dipl.-Medieninf. Martin Voigt
Verantwortlicher Hochschullehrer:	Prof. Dr.-Ing. Klaus Meißner
Institut:	Software- und Multimediatechnik

Erklärung

Hiermit erkläre ich, Jonas Schulz, die vorliegende Seminararbeit zum Thema

Verwendung von Unschärfe in semantischen Modellen

selbstständig und ausschließlich unter Verwendung sowie korrekter Zitierung der im Literaturverzeichnis angegebenen Quellen und Hilfsmittel verfasst zu haben.

Dresden, 08. Juni 2012

Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

In diesem einleitenden Kapitel wird die Motivation und grundlegende Problemstellung für diese Seminararbeit aufgezeigt und die Arbeit im Kontext des *Topic/S* Projektes[1] der fink & partner Media Services GmbH in Zusammenarbeit mit der Professur Multimediatechnik der Technischen Universität Dresden eingeordnet. Anschließend werden die Ziele verdeutlicht, die mit dieser Arbeit erreicht werden sollen und der Aufbau dieser Seminararbeit erläutert.

1.1 Motivation und Problemstellung

Beschreibt man Daten semantisch, so trifft man meist eindeutige Aussagen über Ressourcen, welche man modellieren möchte. Das *World Wide Web Consortium* (W3C) hat z. B. mit dem *Ressource Description Framework* (RDF) und der *Web Ontology Language* (OWL) Technologien geschaffen, die diese Modellierung ermöglichen und sogar für Maschinen lesbar machen. Nicht selten modelliert man aber Sachverhalte aus der realen Welt, deren Zusammenhänge nicht eindeutig bestimmt sind, was bei der Abbildung auf eine eindeutige Logik Probleme bereitet [Hol09]. So ist die Aussage »der Ball ist rund« für einen Football, welcher eine ovale Form hat, nicht eindeutig. Speziell in Gebieten, bei denen mit Medienobjekten gearbeitet wird, finden sich viele Anwendungsfälle, in denen diese *Unsicherheit* (Uncertainty) bzw. *Unschärfe* (Fuzziness) auftritt und modelliert werden muss. Das Topic/S Projekt, in dessen Rahmen sich diese Arbeit eingliedert, beschäftigt sich mit der automatisierten semantischen Beschreibung von journalistischen Medienobjekten und der Möglichkeit diese abzufragen. Im Mittelpunkt steht hier bei der *NewsRoom*, eine Komponente, in dem von journalistischen Quellen, wie Agenturen oder Blogs, Medienobjekte eingehen. Der NewsRoom gibt dabei unter anderem einen Überblick über alle derzeitig verfügbaren Artikel. Diese Artikel werden analysiert und deren Metadaten in einer Ontologie eingeordnet. Neben zeitlichen Aspekten und Metadaten, wie Autor und Quelle, wird ein Artikel je nach Inhalt einem Thema und Schlagwörtern zugeordnet. Schlagwörter helfen dabei den Artikel einer oder mehrerer Kategorien zuzuordnen. Bei all diesen Vorgängen treten Probleme der Unschärfe auf, wobei die Daten mit Gewichtungen beschrieben werden müssen Ein Beispiel wäre ein Artikel über Angela Merkel, bei dem unter anderem die Schlagwörter »Angela Merkel«,»CDU« und »Bundeskanzler« herausgezogen wurden. Man kann erkennen, dass der Begriff »Bundeskanzler« eine stärkere Gewichtung zur Kategorie Politik hat, als »Angela Merkel«, deren Begriff aber auch die Kategorie »wichtige Persönlichkeiten/Prominente« abdeckt. Eine weitere Problematik ist das erkennen von künftigen interessanten Themen, dem so genannten *Hot Topic*, welches eine Ungewissheit aufweist. Das Problem ist Parameter zu bestimmen und einen Algorithmus

[1] weitere Informationen und Aktuelles unter `http://topic-s.de/`

zu entwickeln, der zuverlässig ähnliche Artikel gruppiert, die ein in naher Zukunft wichtiges Thema behandeln und dass diese Daten in geeigneter Form für den Benutzer bereitgestellt werden. Neben der Modellierung zeigen sich aber auch Probleme beim Stellen von Anfragen an die Ontologie. Beschreibt man beispielsweise Kategorien von Zeitungsartikeln, so würde ein Artikel über ein Sportevent, an dem ein hoher Politiker zuschaute die Kategorien »Sport« und »Politik« abdecken. Durch die eindeutigen Aussagen geschieht das nun aber zu gleichen Teilen. Eine Maschine kann nicht, wie ein Mensch, den Kontext erkennen, ob der Sport oder die Politik hier einen höheren Stellenwert hat. Probleme entstehen bei einer Anfrage eines Politik-begeisterten, Sport-desinteressierten Benutzers, dem nun auch alle Sportartikel präsentiert werden, in denen die Politik nur zu einem Bruchteil eine Rolle spielt. Der Benutzer müsste nun selbst nochmal das Ergebnis nach wirklich relevanten Artikeln durchsuchen. Einen großen Nutzen hat unscharfe semantische Modellierung daher bei Anwendungsfällen, wo entschieden werden muss, ab welchem Grad von relevanten Informationen Daten behalten und weiterverarbeitet werden sollen und welche Daten außen vor gelassen bzw. verdrängt werden können. Auch können Gewichtungen eine große Hilfe bei der Beschreibung der Daten sein, um Prioritäten oder Rankings festlegen zu können. Dadurch wird die Fülle an möglichen relevanten Daten nochmals für den Benutzer übersichtlich strukturiert.
In vielen semantischen Modellen wurden die eben aufgezeigten Probleme der unscharfen Modellierung nicht von vornherein betrachtet, weswegen dies auch schon seit Jahren ein Forschungsschwerpunkt ist. Dabei sind Erweiterungen von Technologien entstanden, die z. B. auf Basis von *Fuzzy-Logik* arbeiten und diese Modellierung ermöglichen. In Bezug auf das Topic/S Projekt besteht das Problem, wie diese Daten überhaupt semantisch unscharf beschrieben werden können, einschließlich der Frage, welche Voraussetzungen dafür nötig sind. Außerdem muss eine Methode gefunden werden, dass die Medienobjekte zuverlässig automatisiert gewichtet werden.

1.2 Ziele der Arbeit

Für die Umsetzung der Aufgabe müssen derzeitige Standards und Technologien in Bezug auf Unschärfe und Gewichtung untersucht werden. Zusätzlich muss untersucht werden, welche Frameworks unscharfe Modellierung unterstützen und inwiefern sie für die Implementierung von Nutzen sind. Gleichzeitig sollen Projekte und Arbeiten untersucht werden, die bei der Bestimmung von Relevanz, Gewichtung, Ranking etc. helfen. Diese Vorbereitungen dienen schlussendlich dazu in einer weiterführenden Bachelorarbeit ein Konzept für das Topic/S Projekt auszuarbeiten und zu implementieren, wobei der entwickelte Prototyp eingehende Mediendaten semantisch gewichtet annotieren und in eine Datenbasis einpflegen, sowie abfragen können soll. In Bezug, auf die in 1.1 dargestellte Problematik muss außerdem untersucht werden, wie man Gewichtungen von Schlagworten und Kategorien automatisiert berechnen kann, wobei deren *Relatedness*, also der Zusammenhang der Begriffe, in einer vorhandenen Datenbasis berücksichtigt werden muss. Dadurch soll es möglich sein zu erkennen, wie stark Artikel miteinander verwandt sind, um Vorschläge für ähnliche Artikel zu erhalten. Bei der Problematik des Hot Topic sollen verschiedene Parameter bestimmt werden, die eine zuverlässige Erkennung möglich machen.

1.3 Aufbau der Arbeit

In Kapitel 2 werden zunächst grundlegende Begriffe diskutiert und voneinander abgegrenzt, sowie Grundlagen für unscharfe Modellierung vorgestellt. Anschließend wird an Hand eines Szenarios eine Anforderungsanalyse in Bezug auf diese Arbeit durchgeführt. Das 3. Kapitel wird den aktuellen Stand der Technik in Bezug auf unscharfe Modellierung, bestehende Frameworks und Implementierungen vorstellen und voneinander abgrenzen. Am Ende dieses Kapitels steht eine Zusammenfassung, in der hilfreiche Technologien und Forschungen für die weitere Arbeit hervorgehoben werden sollen. Kapitel 4 behandelt das Konzept, welches auf den bisherigen Kapiteln aufbauen wird und in Kapitel 5 als prototypische Umsetzung dokumentiert wird. Den Abschluss der Arbeit wird eine Zusammenfassung der Forschungsarbeit und einen Ausblick über weitere Arbeiten in Kapitel 6 geben.

2 Grundlagen

Dieses Kapitel gibt einen Überblick über die Wichtigsten Begriffe und Grundlagen, deren Verständnis für die vorliegende Seminararbeit benötigt wird. Dafür werden in 2.1 allgemeine Begriffe erläutert, in 2.2 hingegen grundlegende semantische Technologien vorgestellt. Am Ende des Kapitels wird anhand eines Beispielszenarios eine Anforderungsanalyse vorgenommen unter deren Aspekte der Stand der Technik in Kapitel 3 auf Verwendbarkeit für die Problemstellung aus 1.1 geprüft wird.

2.1 Grundbegriffe

2.1.1 Unschärfe und Unsicherheit

Die wohl wichtigsten Begriffe in dieser Arbeit sind die der Unschärfe (engl. fuzziness) und Unsicherheit (engl. uncertainty) im Kontext des Semantic Web. Jedoch gibt es laut [Str11] einen entscheidenden Unterschied zwischen dieses Begriffen. Diesen Unterschied zu erkennen hilft vor allem dem Verständnis bei der semantischen Modellierung, wenn man weiß, ob man nun Unschärfe oder Unsicherheit abbildet. Allgemein steht dabei die Uneindeutigkeit von Aussagen im Mittelpunkt, die nicht exakt *wahr* oder *falsch* sind bzw. dadurch nur unzutreffend beschrieben werden. Diese Arten von Aussagen sind ebenfalls ein grundlegendes Problem des Semantic Web, wie es auch schon von [Ger05] beschrieben wurde.
Unschärfe beschreibt dabei einen bestimmten *Grad*, den eine Aussage in Bezug auf ein eindeutiges Konzept hat. Das Intervall ist dabei meist [0,1], wobei 1 ein eindeutiges Zutreffen von *Wahr* und 0 ein Zutreffen von *Falsch* beschreibt. Ein Beispiel wäre die Aussage, dass ein bestimmter Journalartikel über Angela Merkel bei den Olympischen Spielen zu einem Grad 0,3 in den Bereich Politik und 0,7 in den Bereich des Sportes fällt bzw. abdeckt. Diese Information kann einerseits eine subjektive Einschätzung, andererseits aber auch aus vorherigen Gewichtungen, die im Zusammenhang mit dem Artikel stehen berechnet worden sein.
Unsicherheit hingegen kann nicht auf Basis von bisherigen Werten berechnet werden. Es wird hierbei ein Wert angegeben, der meist die obere Grenze einer Wahrscheinlichkeit repräsentiert zu der eine Aussage zutreffen *kann*. Der Grund dafür ist fehlendes Wissen. Im Gegensatz zur Unschärfe, wo man Vagheit ausdrückt,wird hier eine Wahrscheinlichkeitsaussage über eine Möglichkeit angegeben. Ein Beispiel hierfür wäre die Wahrscheinlichkeit von bis zu 70 % , mit der ein Thema in den Nachrichten für die Masse interessant werden wird. Das Thema kann nun interessant werden, muss aber nicht. Man kann sich jedoch sicher sein, dass es zu mindestens 30 % nicht eintreffen wird.

2.1.2 Fuzzylogik

Die Fuzzylogik, auch unscharfe Theorie genannt, ist die mathematische Grundlage für viele Frameworks und Sprachen, die sich mit der unscharfen semantischen Modellierung beschäftigen. Es ist eine Verallgemeinerung der zweiwertigen Logik und bildet Wahrheitswerte auf das Intervall [0,1] ab. Grundlage für die Fuzzylogik selbst ist die Fuzzy-Set-Theorie von Lotfi Zadeh, die 1965 in [Zad65] vorgestellt wurde. Dabei werden Objekten Werte aus dem Intervall zugeordnet, wodurch sie in eine bestimmte Menge fallen, dem so genannten Fuzzy-Set. Bei der Fuzzylogik werden die Intervallwerte, auch Grad der Wahrheit genannt, Aussagen zugeschrieben [Str11]. Hierbei werden Konjunktion, Disjunktion und Negation der Booleschen Logik für den Fall der Unschärfe erweitert und Regeln für das Schlussfolgern definiert. Weitere Informationen zum Thema Fuzzy-Sets, Fuzzylogik und deren Ableitungsregeln können in [Zad65], [Zad88] und [Str11] nachgelesen werden.

2.1.3 Ontologie

In der Informatik wird die Definition des Begriffes sehr von Tom Gruber in [Gru09] geprägt. Hierbei ist eine Ontologie eine Menge von Begriffen, die Attribute besitzen und in Beziehung zueinander stehen. Dies ermöglicht es, Wissensstrukturen, seien es reale oder imaginäre Anwendungsbereiche, zu modellieren. Ein wesentlicher Bestandteil ist die Definition von Regeln, die beschreiben, wie die Begrifflichkeiten in Beziehung zueinander stehen können. Mithilfe dieser Regeln ist es möglich, logisch Schlussfolgerungen über Inkonsistenzen oder Widersprüche innerhalb der Ontologie zu finden. Es ist ein wesentlicher Bestandteil des Semantic Web und im Gebiet der künstlichen Intelligenz. Es gibt diverse Editoren, um Ontologien beispielsweise mit den Sprachen RDF Schema und OWL zu formulieren. Die wohl bekanntesten, frei verfügbaren Beispiele dafür sind Protégé[1] und TopBraid Composer[2].

2.2 Grundlegende Semantische Technologien

2.2.1 RDF

Als der W3C 1999 den ersten Standard zum Resource Description Framework (RDF) veröffentlichte, war es ein Modell um Metadaten von Webseiten, wie Autoren und Lizenzbedingungen, zu beschreiben. Dieses Modell erfuhr allerdings einen ideologischen Wandel, als es für die allgemeine Beschreibung semantischer Daten genutzt werden sollte [LBH01]. Das Grundmodell, welches auf gerichteten Graphen basiert, die Beziehungen von so genannten Ressourcen untereinander darstellen, blieb dabei unverändert. Diese Ressourcen werden mit eineindeutigen Uniform Resource Identifier (URI) beschrieben [KCM04], sodass auch beim Zusammenfügen mehrerer Graphen die Daten konsistent bleiben. Ressourcen werden hierbei als Subjekte bezeichnet, die über Prädikate, auch Property Ressource genannt, mit Objekten verbunden sind. Objekte können ebenfalls Ressourcen oder aber Literale sein. Diese Tripel bilden in ihrer Menge den RDF Graphen. Das gesamte Framework basiert dabei auf

[1] http://protege.stanford.edu/
[2] http://www.topquadrant.com/products/TB_Composer.html

XML als Auszeichnungssprache, es ist aber auch möglich kompakte, für Menschen lesbare, Sprachen, wie N3 oder Turtle zum Serialisieren zu nutzen [HKR10].

2.2.2 RDF Schema

RDF Schema kann als eine Erweiterung von RDF angesehen werden, wobei ein Vokabular bereitgestellt wird, mit dem man nun die Möglichkeit hat, Klassen und deren Beziehungen zueinander in einer Domäne zu definieren [Hit+08]. Demnach ist es möglich, mit RDF Schema Ontologien zu beschreiben, wobei dies aufgrund der geringen Komplexität aber auch nur für leichtgewichtige Ontologien nutzbar ist. Neben der Instanziierung von Entitäten aus diesen Klassen ist es auch möglich Vererbungen von Klassen und Propertys (vgl. Prädikat) zu modellieren. Ebenso ist es möglich für die Ressourcen, die durch eine Property verbunden werden, Typenbeschränkungen festzulegen. Das vollständige Vokabular von RDF Schema kann unter [BGM04] eingesehen werden.

2.2.3 OWL

Die Web Ontology Language (OWL), die eine Erweiterung von DAML+OIL, RDFS ähnlichen Beschreibungssprachen darstellt, ist ebenfalls eine W3C Recommendation und Sprachfamilie zum Formulieren von Ontologien. Das Vokabular ist viel umfangreicher als bei RDF Schema, weswegen es möglich ist viel komplexere und mächtigere Ontologien zu beschreiben. Es gibt 3 Sprachversionen von OWL: OWL Lite, OWL DL und OWL Full [MH04]. OWL Full bietet dabei den gesamten Sprachumfang, um prädikatenlogische Ausdrücke höheren Grades abzubilden, ist aber nicht immer entscheidbar und daher nicht zur Benutzung empfohlen, weswegen es auch von nur wenigen Tools unterstützt wird. OWL DL ist eine Description Language und liefert eine Untermenge von Funktionen, die OWL Full bietet, um Entscheidbarkeit zu gewährleisten. OWL Lite bietet die kleinste Menge an Sprachkonstrukten, kann aber auch schon für komplexere Ontologien, als RDF Schema, eingesetzt werden. Aufgrund der zu Grunde liegenden Prädikatenlogik ist es möglich implizites Wissen zu erschließen.
Als Erweiterung wurde mittlerweile OWL 2 als Recommendation vorgestellt [Gro09]. OWL 2 ist dabei abwärtskompatibel und erweitert die erste Version um Datentypen, Arten von Properties (asymmetrisch, reflexiv und disjunkt), sowie mehr Annotationsmöglichkeiten.

2.2.4 SPARQL

Die SPARQL Protocol and RDF Query Language (SPARQL) ist eine Recommendation des W3C um RDF Graphen anzufragen [PS08]. Sie ist stark an SQL angelehnt und bietet unterschiedliche Formen der Abfrage. Neben SELECT-Anweisungen, mit denen eine Untermenge an Tripeln aus dem Graph zurückgegeben werden gibt es ASK, um einen booleschen Wahrheitswert und DESCRIBE und CONSTRUCT, um einen Graph als Antwort geliefert zu bekommen. Neben der Konjunktion, Disjunktion, Negation und optionalen Anweisungen gibt es auch die Möglichkeit Filter und Gruppier- bzw. Sortierfunktionen in einer Anfrage zu verwenden. Es wird derzeit an

der Erweiterung SPARQL 1.1 gearbeitet, die weitere Funktionalitäten, wie das Bearbeiten von Graphen, Sub-Queries und Aggregatfunktionen bereitstellen soll [HS12].

2.3 Referenzszenario

In Anlehnung an die Motivation soll das folgende Szenario beispielhaft aufzeigen, welche Anforderungen an das Topic/S-System gestellt werden müssen und welche Themen im Related Work betrachtet werden müssen. Es wird konkret davon ausgegangen, dass sowohl eine Nutzersuche durch einen Journalisten, der Artikel zur Weiterverarbeitung sucht, als auch das Eintreffen eines Artikels in den NewsRoom simultan erfolgen. Der eingehende Artikel behandelt »Angela Merkels« Wirken in der »CDU« und Politik in der Position des »Bundeskanzlers« in »Berlin«. Genau so ein Thema wird gleichzeitig vom Nutzer gesucht.

1. **Artikelsuche durch Nutzer einleiten**
   Der Nutzer initiiert eine Suche über die Programmoberfläche, indem er Schlagwörter und/oder gewünschte Kategorien selbst bestimmt bzw. aus einer Liste auswählt. Er kann dabei bestimmen wie stark ein Artikel eine Kategorie mindesten oder höchstens abdecken soll.

2. **Anfrage an das System**
   Die Anfrage wird an das System geleitet. Diese soll möglichst mit SPARQL verarbeitet werden können. Es wird eine Liste von Tripeln aus der Medienontologie zurückgegeben.

3. **Darstellen der Ergebnisse**
   Die Ergebnisse der Anfrage werden dem Nutzer listenartig dargestellt. Die angezeigten Informationen sind dabei der Titel des Artikels, eine Reihe von Schlagwörtern, die Kategorien, in die es fällt und ein prozentualer Wert, wie sehr die das Ergebnis auf die Suche zutrifft. Dieser Wert ist abhängig von den eingestellten Parametern, wie Kategorie, Schlagwort, Quelle des Artikels, Zeitlicher Aspekt etc. .

4. **Eintreffen eines neuen Artikels**
   Während der Suche trifft ein neuer Artikel in der zentralen Sammelstelle des Systems, dem NewsRoom, ein. Der Artikel hat eine Quelle und weitere Metadaten, welche in die Medienontologie übernommen werden. Metadaten, wie Erstellungsdatum, Autor und Publisher werden dabei schon im IPTC Metadatenformat mitgeliefert und übernommen.

5. **Textanalyse des Artikels**
   Schlagwörter werden durch Textanalyse und anschließende Named Entity Recognition (NER) bzw. Named Entity Disambiguation (NED) als Entitäten herausgezogen und typisiert. Dabei werden Begriffe auch durch externe Services, wie einer Thesaurus-Ontologie auf Synonyme geprüft, um Dopplungen auszuschließen.

6. **Kategorisierung der Begriffe**
   Die extrahierten Entitäten werden nun einer Liste von möglichen Kategorien

oder Themengebieten zugeordnet. Dabei kann ein Begriff in mehrere Kategorien fallen. Ob bzw. worein ein Begriff in eine der vorgesehenen Themengebiete fällt muss durch Anfragen einer Thesaurus-artigen Ontologie geschehen. Je nach Verteilung auf mehrere Kategorien wird der Einfluss des Begriffes dafür berechnet.

7. **Gewichtung des Artikels in Kategorien**
   Auf Basis der erhaltenen Gewichtungen in mögliche Kategorien der Schlagwörter wird nun die Gewichtung für den Artikel in diese Kategorien durch Verrechnen aller relevanten Schlagwörter bestimmt. Diese Werte werden zur Medienontologie hinzugefügt.

8. **Aktualisierung der Suchergebnisse beim Nutzer**
   Ist der Artikel in der Medienontologie eingetragen, so wird er dem Benutzer zur Verfügung gestellt, soweit er für die Suche relevant ist.

9. **HotTopics-Anzeige**
   Immer im Blickfeld ist die Anzeige über mögliche kommende Themen, die beim Eintreffen von Artikeln in den NewsRoom aktualisiert wird. Es werden dabei Untermengen von Schlagwörtern und die Gewichtungen der Kategorien der Artikel untereinander verglichen und bei mehreren Übereinstimmung eine Aufnahme in die HotTopics Liste initiiert, wobei die übereinstimmende Menge der Schlagwörter, die Anzahl der ähnlichen Artikel und einem gewichteten Wert für die Relevanz dieses Themas im Kontext von kommenden Themen sichtbar sind.

2.4 Anforderungsanalyse

Aus diesem Szenario ergeben sich folgende Anforderungen für diese Arbeit:

- **Funktionale Anforderungen**

  1. **automatisierte Verarbeitung**
     Die Verarbeitung der Metadaten und Gewichtung der Artikel müssen vollkommen ohne menschliches Einwirken geschehen und dabei zuverlässig arbeiten. Ein manuelles Einwirken sollte nur beim Bestimmen der Parameter für eine Suchanfrage auftreten. Optional kann allerdings eine Art Administrationsfunktion eingebaut werden, um beispielsweise fälschlich erkannte Schlagworte oder Kategoriezuweisungen zu entfernen oder zu korrigieren.

  2. **Gewichtung**
     Die Gewichtung wird bei der Beschreibung, beispielsweise wie stark ein Artikel eine Kategorie abdeckt, verwendet. Ein weiterer Anwendungsfall sind Gewichtungen von Entitäten auf Basis ihrer Verwandtschaft zueinander. Diese Berechnung muss wiederum automatisch und auf Basis bisheriger Daten, sowie mit Hilfe von Services durchgeführt werden.

  3. **Services und Datenbasen nutzen**
     Um die Verarbeitung der Schlagworte bzw. Entitäten in Zusammenhang

mit Thesauri zu gewährleisten, müssen bestehende Services bzw. Ontologien genutzt werden, die solch einen Dienst anbieten. DBPedia und Wikipedia wären hier eine gute Grundlage.

4. **Medienontologie**
   Es muss eine bestehende Medienontologie so erweitert werden, dass Gewichtungen modelliert werden können, jedoch die bisherige Funktionalität nicht dabei eingeschränkt wird. Daher muss besonders auf die Verwendung von semantischen Technologien geachtet werden, die vom W3C empfohlen werden und kompatibel zur bisherigen Ontologie sind.
5. **Suche von Artikeln**
   Es müssen Parameter bzw. mögliche Einschränkungen für die Suche von Artikeln bezüglich der Unschärfe festgelegt werden. Die UI der Suchfunktion wird dabei entsprechend angepasst, sodass alle ausgewählten Parameter von den Benutzer verständlich und einfach verwendet werden können.
6. **Hot Topic**
   Auf Basis von Ähnlichkeiten zwischen Artikeln und daraus resultierenden thematischen Gruppierungen sollen Gruppen von Artikeln als Hot Topic, also dem kommenden Trendthema, aufgelistet werden. Diese Anzeige soll jederzeit für den Benutzer sichtbar sein und sich automatisch entweder bei Änderung der Inhalte oder in einem festen Zeitintervall aktualisieren.
7. **Reasoning**
   Auf Grund von fehlenden Standards wird nicht angenommen, dass ein Reasoning von unscharfen Ausdrücken unterstützt wird, jedoch darf dadurch die Funktionalität für nicht annotierte Teile der Ontologie nicht eingeschränkt werden. Das heißt, dass trotz der Erweiterung der Ontologie das Reasonign eines Reasoner auf Basis von Standards möglich sein muss.

- **Nichtfunktionale Anforderungen**

1. **Einfachheit**
   Die genutzte Medienontologie und die dafür verarbeitende Technologie soll so einfach wie möglich gehalten werden, um eine bessere Wartbarkeit des Systems, als auch eine bessere Performance sicherzustellen. Hier bietet sich besonders RDF Schema an, da hier ausreichend Sprachmittel bei vergleichsweise wenig Komplexität zu OWL zur Verfügung stehen.
2. **Standardkonformität**
   Es sollte vor allem darauf geachtete werden, dass bestehende Standards ausgeschöpft und auch genutzt werden. Für das Topic/S Projekt betrifft das vor allem die Standards und Empfehlungen des W3C, da diese schon weit verbreitet und anerkannt sind.
3. **Leicht verständliche UI**
   Die Oberfläche muss für den Benutzer intuitiv zu bedienen sein. Das heißt, dass keinerlei Vorkenntnisse über semantische Technologien, vor allem bei der Suche mit SPARQL, vorhanden sein müssen. Zu keinem

Zeitpunkt wird daher eine Eingabe von syntaktisch korrekten Anfragen verlangt, aber auch nicht die Möglichkeit gegeben die Anfrage zu sehen oder zu manipulieren, um Fehler durch den Benutzer auszuschließen und ihn nicht mit irrelevanten Informationen zu überfordern.

4. **Performanz**
   Die entwickelten Funktionalitäten sollen die Performanz nicht negativ beeinflussen. So ist vor allem bei der Suche wichtig einen geeigneten Algorithmus zu finden, der Stabilität und die Leistung der Software nicht zu stark beeinträchtigen.

3 Related Work - Stand der Forschung und Technik

In diesem Kapitel sollen Ansätze vorgestellt werden, wie Gewichtungen auf Basis von Standards und Erweiterungen von Standards realisiert werden können. Im Anschluss werden zwei Arbeiten vorgestellt, die in Bezug auf Relevanzbestimmung Einfluss auf die Entwicklung des Konzepts haben können. Am Ende soll eine Zusammenfassung nochmals die nützlichsten Technologien hervorgehoben werden.

3.1 Modellierungsansätze von Unschärfe mit bestehenden Standards

In diesem Abschnitt werden zunächst Ansätze vorgestellt, die bestehende Semantic Web Standards des W3C ausschöpfen, um Werte zu annotieren. Da für unscharfe bzw. gewichtete Anwendungsdomänen noch kein spezieller Standard definiert wurde muss auf allgemeine Wege zur Annotation von Aussagen zurückgegriffen werden, um diese entsprechend für den Fall der Gewichtungen zu nutzen.

3.1.1 N-äre Relationen

Mithilfe semantischer Technologien des W3C ist es möglich binäre Relationen darzustellen und zu modellieren. Jedoch ist dies in manchen Fällen nicht ausreichend oder bedeutet mehr Schreibaufwand und auch eine hohe Redundanz der Informationen, weswegen man sich Gedanken über die Darstellung von n-ären Relationen gemacht hat [NR06]. Möchte man zum Beispiel einen Artikel in mehrere Kategorien einordnen, so ist dies mit binären Relationen noch ohne Probleme realisierbar. Allerdings treten die ersten Schwierigkeiten auf, sobald man jeder Kategorie ein bestimmtes Gewicht zuteilen möchte, das auch im Kontext zu dieser Kategorie bestehen bleibt. Das Problem ist die feste Zuordnung von der Gewichtung zu der passenden Kategorie für den Artikel. Hier greifen n-äre Relationen ein, bei denen man mit Hilfe von Blank Nodes oder auch Hilfsknoten Aussagen modellieren kann, bei denen mehr Ressourcen den Kontext bilden, als dies bei binären Relationen der Fall ist. Die Formulierung von n-ären Relationen durch Blank Nodes ist syntaxunabhängig und wird schon vom RDF Standard an unterstützt.
N-äre Relationen sind allerdings nicht mit Reifikation zu verwechseln. Bei der Reifikation werden Aussagen über Aussagen (Statements) getroffen, was hier nicht der Fall ist. Folgende Anwendungsfälle für n-äre Relationen wurden vom W3C vorgestellt:

- Anwendungsfall 1: Relationen, die mit zusätzlichen Attributen beschrieben werden.

- Anwendungsfall 2: Relationen, denen verschiedene Ansichten zugeteilt werden können.
- Anwendungsfall 3: Relationen mit Rollenverteilungen, wobei die Relation als Subjekt und die Beteiligte als Objekte verstanden werden.
- Anwendungsfall 4: Modellierung von Listen mit einer festen Reihenfolge.

An dieser Stelle soll ein Minimalbeispiel eingeführt werden, auf das den Rest des Kapitels verwiesen wird. Gegeben sei eine Relation zwischen einem Artikel und der Kategorie »Sport«, die in Abbildung 3.1 in Turtle Syntax dargestellt ist.

```
:Artikel_1  :inCategory  :Sport .
```

Abbildung 3.1: Das Minimalbeispiel für dieses Kapitel

Nun soll dieser Relation eine Gewichtung gegeben werden, so dass die Aussage »$Artikel_1$ fällt zu einem Grad von 0.3 in die Kategorie Politik.« entsteht.
Für die Modellierung des Beispiels trifft hier Anwendungsfall 1 zu, da einer Relation zwischen Artikel und den Kategorien ein Gewicht im Kontext beigefügt wird. Anwendungsfall 3 kann für die Arbeit in Bezug auf die Typisierung von Schlagworten eines Artikels auch von Relevanz sein, soll hier aber vorerst nicht betrachtet werden. In Abbildung 3.2 ist das erweiterte Beispiel für n-äre Relationen dargestellt. Eine SPARQL-Abfrage aller Artikel mit ihren Gewichten pro Kategorie ist in Abbildung 3.3 zu sehen.

```
:Artikel_1     a                :Artikel ;
               :hasCategories   _:Art_1_Cat .

:_Art_1_Cat    a                :Category_Relation ;
               :Category        :Sport;
               :hasDegree       "0.3"^^xsd:float .
```

Abbildung 3.2: Erweitertes Minimalbeispiel mit n-ärer Relation

```
SELECT ?subject ?category ?degree
WHERE {
?subject  a              :Artikel .
?subject  :hasCategories ?bnode .
?bnode    :Category      ?category .
?bnode    :hasDegree     ?degree  .

}
```

Abbildung 3.3: SPARQL-Abfrage für n-äre Relationen

Der Einsatz von n-ären Relationen bietet sich besonders für leichtgewichtige Ontologien an, die einfache Zusammenhänge repräsentieren, die nicht mit binären Relationen darstellbar sind. Ein Problem könnte die Verschachtelung von BlankNodes bzw. Hilfsknoten geben, wenn mehrere Kategorien zu einem Artikel hinzugefügt werden. Dies könnte jedoch durch Namenskonventionen je nach Anwendungsfall so gelöst werden, dass die Übersichtlichkeit nicht verloren geht.

3.1.2 Reifikation

Eine weitere Möglichkeit mit Hilfe von bestehenden Standards zusätzliche Informationen über Tripel bzw. Aussagen zu modellieren ist die Reifikation. Nach [MM04] bietet sich diese Art der Beschreibung an, wenn man anmerken möchte, wer der Autor einer Aussage ist, wann sie getroffen wurde oder woher sie ursprünglich erstellt wurde. Ein weiterer Anwendungsfall wäre die Beschreibung von Statements mit Gewichtungen, wie sie im Rahmen dieser Arbeit untersucht wird. [Lop+09] beschreibt Reifikation sogar als derzeit einzige Methode, um zuverlässig Aussagen über Aussagen mit bestehenden Standard zu beschreiben. RDF stellt dafür ein extra Vokabular bereit [HM04].
Es soll wieder das Minimalbeispiel 3.1 betrachtet werden. Nun wird über dieses Tripel wieder die Aussage getroffen, dass $Artikel_1$ eine Gewichtung von 0.3 für der Kategorie Sport bekommt. Dabei wird eine URI Referenz oder ein Blank Node vom Typ rdf:Statement voran gestellt. Subjekt, Prädikat und Objekt des zu beschreibenden Tripels werden dem Statement zugeordnet, was als indirekte Referenz für die Reifikation zu diesem Tripel gesehen werden kann. In [HM04] wird gesagt, dass es in RDF nicht möglich erkennen zu lassen, welches Tripel durch welche Reifikation beschrieben wird. Dies muss extern gelöst werden, wie beispielsweise mit einer rdf:ID in RDF/XML. In Turtle mit einem Blank Node vorangestellt kann die Aussage wie in Abbildung 3.4 beschrieben werden. Eine passende SPARQL-Abfrage ist in Abbildung 3.5 dargestellt.

```
:Artikel_1  ex:inCategory  :Sport .
_:stmt1     rdf:type       rdf:Statement.
_:stmt1     rdf:subject    ex:Artikel_1.
_:stmt1     rdf:predicate  ex:inCategory.
_:stmt1     rdf:object     :Sport.
_:stmt1     :hasDegree     "0.3"^^xsd:float.
```

Abbildung 3.4: Erweitertes Minimalbeispiel mit Reifikation

Auch wenn Reifikation mit eigenem Vokabular unterstützt wird und damit jegliche Meta-Aussagen formuliert werden können hat das Reifizieren von Aussagen auch einige Nachteile. So entsteht ein erhöhter Schreibaufwand, da das Tripel meist in zweifacher Ausführung niedergeschrieben werden muss. Gleichzeitig kann durch diese Redundanz die gesamte Persistenz der Ontologie verloren gehen. Daher ist diese Technologie mit großem Nutzen verbunden, kann jedoch bei zu wenig Vorsicht Probleme bereiten.

```
SELECT ?subject ?category ?degree
WHERE {
    ?subject :inCategory    ?category .
?stmt    rdf:subject    ?subject .
?stmt    rdf:predicate  :inCategory .
?stmt    rdf:object     ?category .
?stmt    :hasDegree     ?degree .

}
```

Abbildung 3.5: SPARQL-Abfrage für Reifikation

3.1.3 OWL Annotations

Schon in OWL DL war es möglich in einer Ontologie zusätzliche Informationen durch Annotationen zu modellieren, die nicht zur Semantik der Ontologie dazugehören [Bec+04]. Darunter zählen Kommentare, für Menschen lesbare Namen und auch Beschreibungen von Klassen, Properties, Individuen oder der Ontologie selbst [Bec+04]. Mit OWL 2 kam nun noch die Möglichkeit hinzu zusätzliche Informationen über Axiome zu treffen [GW]. Hierbei hat jede Annotation eine Property und dazugehörenden Wert, der ein Literal, ein Individuum oder eine IRI sein kann. Dadurch wird es möglich auch Aussagen um Informationen, wie beispielsweise Gewichtungen, zu ergänzen. Nachfolgend sind zwei Lösungsansätze zu sehen, die unser Beispiel mit Annotationen modellieren. Wie in Abbildung 3.6 zu sehen ist, wird eine

```
:hasDegree     a           owl:AnnotationProperty.

:inCategory    a           owl:ObjectProperty;
               rdfs:domain :Artikel;
               rdfs:range  :Kategorie;
               :hasDegree  "0.3"^^xsd:float .

:Artikel_1     a           :Artikel;
               inCategory :Sport .
```

Abbildung 3.6: Erweitertes Minimalbeispiel mit einer annotierten Property

`AnnotationProperty hasDegree` definiert, um die Property `inCategory` mit einer Gewichtung zu beschreiben. Der Nachteil dieser Methode wird schnell klar, da die Property diesen Wert nur einmal annehmen kann. Man müsste für jeden neuen Artikel eine neue Property zu einer Kategorie erstellen, die ein Gewicht bekommen soll. Ein anderer Ansatz, wie in Abbildung 3.7 zu sehen, wäre ein Axiom zu annotieren. Diese Art der Annotation ist einem Beispiel [GW] nachempfunden worden. Es wird dabei eine Instanz eines Axioms angelegt, welches das zuvor beschriebene Tripel mit zusätzlichen Informationen beschreibt. Auffällig ist hierbei die Ähnlichkeit zur

```
:hasDegree      a               owl:AnnotationProperty.

:inCategory     a               owl:ObjectProperty;
                rdfs:domain     :Artikel;
                rdfs:range      :Kategorie.

:Artikel_1      a               :Artikel;
                inCategory>     :Sport .

:Category1      rdf:type        owl:Axiom ;
                owl:subject     :Artikel_1 ;
                owl:predicate   :inCategory ;
                owl:object      :Sport ;
                :hasDegree      "0.3"^^xsd:float .
```

Abbildung 3.7: Erweitertes Minimalbeispiel mit einem annotierten Axiom

Reifikation in RDFS. Dementsprechend ähnlich würde auch eine SPARQL-Abfrage für den zweiten Fall wie in Abbildung 3.8 aussehen.

```
SELECT ?subject ?category ?degree
WHERE {
?subject a             :Artikel .
?subject :inCategory   ?category .
?axiom   owl:subject   ?subject .
?axiom   :hasDegree    ?degree  .

}
```

Abbildung 3.8: SPARQL-Abfrage für annotierte Axiome

3.1.4 Fuzzy OWL 2 Ontology

Klassische Ontologiesprachen sind, trotz der häufigen Notwendigkeit, nicht dafür zugeschnitten semantisch Unschärfe zu modellieren. Ein Beispiel dafür ist die Web Ontology Language (OWL). Diese basiert, wie einige andere Ontologiesprachen, auf Beschreibungslogiken, auch Description Logic genannt [BS10]. Diese Beschreibungslogiken sind eine entscheidbare Untermenge der Prädikatenlogik erster Stufe und ermöglichen das Schlussfolgern in diesen Ontologien. Da die Standard-Reasoner keine Fuzzylogik unterstützen, wurden Beschreibungslogiken erweitert und entsprechende Reasoner, wie FUZZYDL[1], DELOREAN[2] oder FIRE[3] entwickelt, die jedoch auf

[1] http://nemis.isti.cnr.it/~straccia/software/fuzzyDL/fuzzyDL.html
[2] http://webdiis.unizar.es/~fbobillo/delorean.php
[3] http://www.image.ece.ntua.gr/~nsimou/FiRE/

verschiedenen Beschreibungslogiken basieren, da kein einheitlicher Standard für den Fall der Unschärfe existiert. Bobillo stellt in [BS09] mit FuzzyOwl2Ontology eine Ontologie für OWL 2 vor, mit der es möglich ist unscharfe OWL 2 Aussagen zu modellieren ohne den Standard zu erweitern. OWL 2 basiert dabei auf der Beschreibungslogik SROIQ(D), welche im Vergleich zu der Beschreibungslogik SHOIN(D) in OWL DL zusätzliche Aussagen über Rollen unterstützt. Eine ausführliche Beschreibung der Änderungen kann in [KR10] nachgelesen werden.
Die Ontologie besteht aus 168 Klassen, 28 Object Properties und 11 Datatype Properties. Mit Hilfe dieser Klassen ist es möglich Unschärfe in OWL 2 zu modellieren, jedoch ist dadurch noch kein Schlussfolgern möglich. Dies muss durch die oben erwähnten Fuzzy DL-Reasoner geschehen, wobei ein Parser für fuzzyDL und DeLorean entwickelt wurde. Mit diesem Parser ist es möglich die Ontologie in die komplizierten Syntaxen der beiden Reasoner zu übersetzen. Trotz dieses entwickelten Parsers zur Unterstützung von verschiedenen Reasonern und einem Plugin für Protégé birgt FuzzyOwl2Ontology einige Probleme. Aufgrund der Anzahl von Klassen und Möglichkeiten bedeutet es einen großen Aufwand und vor allem Einarbeitung um selbst nur einfache unscharfe Aussagen zu modellieren, wobei die Ontologie beim Hinzufügen von Instanzen und deren Beziehungen exponentiell wächst.
Dieses Problem wird in [BS11] diskutiert, wobei auch ein komplett neuer Ansatz vorgestellt wird. Die hier vorgestellte Ontologie benutzt ausschließlich in 3.1.3 vorgestellte Annotationen in OWL. Dabei werden weit über einfache Annotationen hinausgehende Funktionen beschrieben, die durch einen geeigneten Reasoner ausgewertet werden können. Grundlegend wird hier innerhalb von Annotationen ein `<fuzzyOWL2>`-Tag benutzt. Der Vorteil ist, dass Standard-Reasoner Annotationen ignorieren und somit auch dieses Tag nicht fehlinterpretieren. Es können Datentypen definiert werden, die verschiedene Funktionen der Fuzzylogik repräsentieren, Fuzzy Modifizierer verwendet, sowie Fuzzy Rollen und Axiome beschrieben werden. Dadurch wird ein großer Teil der Bestandteile von Fuzzylogik und Fuzzy-Sets abgedeckt. Auch hier ist ein Plugin für Protégé verfügbar, welches beim Erstellen von unscharfen Ontologien assistieren soll. Aufgrund einer eigenen Benutzeroberfläche innerhalb Protégés entsteht aber auch hier ein Aufwand bei der Einarbeitung. Ein Vorteil nach [BS11] jedoch ist, dass Ontologien zunächst mit allen Tools, die den Standard unterstützen gebaut und anschließend manuell oder mit dem Plugin annotiert werden können. In Abbildung 3.9 ist eine mögliche Repräsentation für eine Annotation eines Axiomes in OWL/XML-Syntax, analog zu dem Beispiel in 3.1.3. Abfragen können über den fuzzyDL-Reasoner an die Ontologie gestellt werden. Jedoch benutzt dieser eine eigene Syntax, weswegen an dieser Stelle auf eine Beispiel-Abfrage verzichtet wird. Auf Grund der mäßigen Unterstützung für Abfragen und dem hohen Einarbeitungsaufwand ist dieses Plugin nur zu empfehlen, sollte man wirklich komplexere Aussagen der Fuzzylogik modellieren wollen. Für einfache Gewichtungen, die nicht unbedingt für unscharfes logisches Schlussfolgern vorgesehen sind steht der Nutzen mit dem Aufwand in einem schlechten Verhältnis.

```
<ObjectPropertyAssertion >
    <ObjectProperty IRI="#inCategory"/>
    <NamedIndividual IRI="#Artikel_1"/>
    <NamedIndividual IRI="#Sport"/>
    <Annotation >
        <AnnotationProperty IRI="#fuzzyLabel"/>
        <Literal datatypeIRI =’&rdf;PlainLiteral">
            <fuzzyOwl2 fuzzyType =" axiom">
                <Degree value ="0.3" />
            </fuzzyOwl2 >
        </Literal >
    </Annotation >
</ClassAssertion >
```

Abbildung 3.9: Erweitertes Minimalbeispiel in der vorgestellten OWL 2 Ontology

3.2 Modellierungsansätze von Unschärfe durch Standarderweiterungen

Auf Grund der Notwendigkeit nicht nur zusätzliche Informationen zu modellieren, sondern dies effizient und in einer bestimmten Anwendungsdomäne zu tun, wurden verschiedene Standarderweiterungen gebaut. Hierbei werden nun ein paar Repräsentative in Bezug auf gewichtete Annotationen vorgestellt.

3.2.1 N-Quads

Ein Ansatz, welcher als Standarderweiterung einzuordnen ist, sind N-Quads. Diese erweitert N-Triple um ein Kontextattribut. N-Triple ist ein vom W3C entwickeltes Format um RDF Graphen in reinem Text darzustellen [BB01]. Dies dient vor allem der Lesbarkeit für Benutzer und als Repräsentationsformat für Ergebnisse von Suchanfragen. Dabei steht eine Zeile der Ausgabe für ein Tripel des zurückgegebenen Graphen. Die Form von N-Triple ist folgende:

```
<Subjekt>  <Prädikat>  <Objekt>
```

Die Notwendigkeit für einen zusätzlichen Kontextwert kam auf, um eine zusätzliche Information über die Herkunft von Aussagen zu erhalten. Dies interessiert meist beim Austausch von Datensätzen zwischen verschiedenen Quellen. Laut [CHH09] fügen heutige RDF-Repository ebenfalls eine Kontextkomponente an Tripel an, um über die Herkunft zu informieren. Dies wird vor allem bei Datensätzen eingesetzt, die aus verschiedenen Named Graphs[4] bestehen. Named Graphs sind Graphen aus verschiedenen Quellen, die gemeinsam in einem Dokument oder Repository liegen und dabei eine Beschreibung der Herkunft als Zusatzinformation besitzen. Neben

[4] http://www.w3.org/2004/03/trix/

URIs, die in den meisten Fällen im Kontextwert stehen, können auch Zeit und Ortsinformationen als Kontext gespeichert werden. Die Form eines N-Quads ist wie folgt:

```
<Subjekt>  <Prädikat>  <Objekt>  <Kontext>
```

Mit dem NxParser[5] ist es möglich nicht nur mit N-Triple und N-Quads, sondern jeglicher Art von Tupel zu arbeiten bzw. in das N-Triple Format umzuwandeln, um damit weiter zu arbeiten. In Bezug auf unser Szenario könnte man nun die Kontextkomponente nutzen, um Gewichte zuzuordnen. Ein Beispiel dafür ist in Abbildung 3.10 dargestellt. Wie man sieht ist die Notation für Kontexte zu einem Triple mit

```
<:Artikel_1>  <:inCategory>  <:Sport>  <"0.3">.
```

Abbildung 3.10: Beispielhafte Quadrupel mit Gewichten

einem N-Quad sehr übersichtlich und schnell geschrieben. Der ursprüngliche Verwendungszweck von N-Quads ist jedoch das Hinzufügen einer URI bzw. IRI der Quelle oder ähnliche Informationen auf eine große Menge von Tripeln, um die Herkunft eines Tripels zu beschreiben. Dabei stellt sich die Frage, wie nützlich diese Technologie in der Praxis ist. Für das Beschreiben von vielen Tripeln mit dem selben Kontextwert, wie bei vielen Named Graphs in einem Repository, bietet das einen großen Komfort, da jedes Tripel nicht als Statement definiert und reifiziert werden muss. Im Anwendungsfall von Unschärfe jedoch tragen Tripel selten gleiche Gewichtungen, sodass nie ein großer Datensatz mit einem Mal bearbeitet werden muss. Ein weiteres Problem ist die Unterscheidung von verschiedenen Arten von Kontextinformationen, wie Gewichtungen und Zeitinformationen für ein Tripel.

3.2.2 Annotated RDFS/aRDFS

Wie schon eingangs erwähnt, wurde bisweilen zwar noch kein Standard für die Modellierung von Unschärfe und Unsicherheit vom W3C entwickelt, jedoch hat sich eine W3C Incubator Group von 2007 bis 2008 ausgiebig mit dem Problem beschäftigt und dabei auch Use Cases für »Uncertain Reasoning« in [Las+08] ausgearbeitet und veröffentlicht. Auf Basis dieser Ausarbeitung wurde das Verlangen einen einheitlichen Standard zu haben noch größer, sodass etwas später mit [Lop+09] ein Paper veröffentlicht wurde, in dem direkte Vorschläge für die Umsetzung solch eines Standards vorgestellt wurden. [Lop+09] sagt, dass Reifikation (siehe 3.1.2) als einzige Standardmöglichkeit zwar gut genutzt werden kann, um Statements mit Kontext bzw. zusätzlichen Informationen wie Zeitkomponenten, Vagheit, Unschärfe und Herkunft zu beschreiben, diese Arten von Kontextwerten aber auf Grund fehlender Schlussregeln semantisch nicht in RDFS und OWL verarbeitet werden können. Als andere Möglichkeit wird das Nutzen von Quadrupeln, wie bei Named Graphs erwähnt, welche aber ebenfalls nicht standardisiert sind.
Umberto Straccia stellt in [Str+10] ein Framework vor um einhergehende Probleme, wie Abwärtskompatibilität, Schlussfolgern und Annotieren von neu gewonnenen Informationen zu bewältigen. auf Basis von Quadrupeln und Reifikation (beide

[5] http://code.google.com/p/nxparser/

Operator	Beschreibung	Berechnung mit Beispiel
$\wedge$	keine explizite Definition	
$\vee$	mehrere Gewichtungen des selben Statements	$a \vee b = max(a, b)$ $(\tau : 0.3) \vee (\tau : 0.8) = max(0.3, 0.8) = 0.8$
$\oplus$	Disjunktion	$a \oplus b = a + b - a \cdot b$ $(\tau : 0.3) \oplus (\tau : 0.8) = 0.3 + 0.8 - 0.3 \cdot 0.8 = 0.86$
$\otimes$	Konjunktion	$a \otimes b = a \cdot b$ $0.3 \otimes 0.8 = 0.3 \cdot 0.8 = 0.24$

Tabelle 3.1: einige Operatoren für unscharfes Schlussfolgern

Darstellungsformen werden unterstützt) werden Annotationen modelliert. Zusätzlich werden Regeln für das Schlussfolgern von Annotationen vorgestellt. Die allgemeine Form für ein Tripel in dem Framework ist folgende:

```
<Subjekt, Prädikat, Objekt> : Annotationswert
```

Hierbei bekommt ein RDF Tripel τ einen Annotationswert υ zugeordnet. Somit wird jedes Tripel, egal ob notwendig oder nicht, mit einem Wert annotiert. Neben Gewichtungen im Intervall von [0, 1] können diese Annotationswerte auch temporale Informationen, z. B. ein Zeitintervall, sein. Für das Beispiel eines Artikels, der zu 0.3 in die Kategorie »Sport« fällt, sehe die Notation in der allgemeinen Form demnach wie in Abbildung 3.11 aus. Eine Repräsentation des Quadrupels in Tripeln ist in Abbildung 3.12 dargestellt, wobei die Darstellung stark an Reifikation ohne die Spezifizierung des eigentlichen Tripels erinnert.

```
<Artikel_1, inCategory, Sport> : 0.3
```

Abbildung 3.11: Erweitertes Beispiel in der allgemeinen Form in aRDF

```
_:b1  rdf:type      rdf:Statement .
_:b1  rdf:subject   :Artikel_1 .
_:b1  rdf:predicate :inCategory .
_:b1  rdf:object    :Sport .
_:b1  :hasDegree    "0.3" .
```

Abbildung 3.12: Erweitertes Beispiel in Tripel-Schreibweise in aRDF

Straccia beschreibt weiterhin in [Str+10] eine Menge formaler Regeln für verschiedene Anwendungsbereiche, weswegen in Tabelle 3.1 nur Operatoren in Bezug auf Gewichtungen vorgestellt werden. Diese Operatoren werden von Straccia in [Str09] genutzt, um Regeln zum Schlussfolgern in RDFS festzulegen. Dabei wird eine Untermenge des Vokabulars von RDFS mit ρdf = {subclass, subproperty, type, domain,

range} gewählt. Für dieses Vokabular Regeln, wie Typisierung, Vererbung und Generalisierung entwickelt, um Schlussfolgern mit unscharf beschriebenen Aussagen zu ermöglichen.

Das Framework[6] wurde bisher nur prototypisch in Prolog umgesetzt. In Anbetracht, dass es bisher keinen festgelegten Standard gibt, erfüllt das Framework aber grundlegende Ansprüche, um Unschärfe zu modellieren und sogar zu Schlussfolgern, wobei eine einfache Syntax und Vokabular von bestehenden Standards genutzt wird. Neben dem unscharfen Beschreiben von semantischen Daten, wurde dabei auch eine Abfragesprache namens AnQL entwickelt, die im Kapitel 3.2.3 behandelt wird.

3.2.3 AnQL

Ergänzend zu dem in 3.2.2 vorgestellten Framework wurde eine Abfragesprache entwickelt, die in diesem Kapitel separat beschrieben wird, da sie sich unabhängig von aRDFS nutzen lässt. Die Annotation Query Language, kurz AnQL, ist in Anlehnung an aRDF entwickelt worden, um annotierte Daten in RDF Graphen auch abfragen zu können [Lop+10].
Die Abfragesprache baut dabei auf dem SPARQL-Standard auf und erweitert diesen gleichzeitig mit weiteren Funktionen, die auch in SPARQL 1.1 vorgesehen sind, wie beispielsweise Aggregatfunktionen. Der syntaktische Aufbau einer Abfrage ist dabei analog zu einer gewöhnlichen SPARQL-Abfrage. Es gibt einen Datensatz, der abgefragt wird und ein Abfrage-Pattern, welches die Abfrage spezifiziert, wobei eine Menge von Variablen genutzt werden um die Ergebnismenge zu definieren. Im Falle von AnQL ist besteht diese Menge aus Annotations-Variablen.
Wie bei aRDFS werden auch bei AnQL mehrere Anwendungsdomänen unterschieden, wie z. B. Unschärfe mit Werten im Intervall von 0 bis 1 oder temporale Informationen als Einzelwert bzw. Zeitintervall. Dabei werden Graph-Muster wie `OPTIONAL`, `FILTER`, `AND` oder `UNION` unterstützt. Im Vergleich zum derzeitigen SPARQL Standard werden aber weitere Funktionen geliefert, die derzeit für SPARQL 1.1 entwickelt werden. Nach [Lop+10] beinhaltet dies Assignments, also das Definieren von Ausdrücken als neue Variablen, Sub-`SELECT` Pattern und Aggregatfunktionen, wie `SUM`, `AVG`, `MAX`, `MIN`, `COUNT`, $\vee$, $\wedge$ und $\otimes$. Die Benutzung der Funktion ist allerdings je nach Anwendungsdomäne unterschiedlich. So können z. B. `SUM` und `AVG` nur bei numerischen Werten eingesetzt werden.
Nachfolgend ist in Abbildung 3.13 eine Abfrage, dass alle Artikel liefert, deren Anteil an Sport kleiner als 0.5 ist, womit unser Minimalbeispiel in der Ergebnismenge enthalten wäre. AnQL ist durch die starke Anlehnung an SPARQL leicht zu verste-

```
SELECT ?subject ?degree
WHERE {
?subject :inCategory :Sport :?degree
FILTER (?degree <= 0.5)
}
```

Abbildung 3.13: Beispielhafte AnQL Abfrage

[6] http://anql.deri.org/index.html

hen und geht derzeit sogar über den SPARQL-Standard hinaus. In [Lop+10] werden jedoch auch Probleme vorgestellt, die in weiterer Arbeit gelöst werden müssen. So gilt es eine Lösung zu finden um annotierte und nicht annotierte Tripel gleichzeitig abfragen zu können. Weiterhin stellt sich die Frage, wie verschiedene Anwendungsdomänen behandelt werden sollen, wenn also gewichtete und temporale Informationen gleichzeitig abgefragt werden. Daher ist die Abfragesprache AnQL zur Zeit noch in Entwicklung. Jedoch gibt es eine prototypische Implementierung[7] der grundlegenden Funktionen.

3.3 Projekte und Ansätze für Relevanzbestimmung und -berechnung

Im folgenden werden zwei Arbeiten vorgestellt, die sich damit beschäftigen wie mit Entitäten bzw. Begriffen vorgegangen wird, um ihnen eine Gewichtung in einer Anwendung zuzuschreiben. Dabei werden zwei unterschiedliche Ansätze gezeigt, die auf das künftige Konzept dieser Arbeit Einfluss haben können und sollen.

3.3.1 Twitcident

Mit Twitcident[8] wird in [Abe+12] ein Framework vorgestellt, mit dessen Hilfe Social Web Streams analysiert werden, um Informationen zu Unfällen und Katastrophen zu filtern und für Benutzer durchsuchbar zu machen. Das System wird dabei in zwei Hauptbestandteile eingeteilt: Das automatische Erstellen von Profilen zu Ereignissen und diesbezügliche Filterung von Social Media Streams, sowie eine Facettensuche und Echtzeitanalyse der erhaltenen Daten. Social Media Stremas die dabei betrachtet werden sind Twitter, Twitpic und Twitvid[9]. Die Architektur ist in Abbildung 3.14 dargestellt.

Das Bereitstellen der nötigen Informationen für den User läuft automatisiert in folgenden Schritten ab und bildet dabei den ersten Teil der Architektur:

1. Ereignisse werden durch Notfallsysteme erkannt und an das System übermittelt.

2. Für das neue Ereignis wird ein Profil angelegt, welches durch Ansammlungen von Social Media und Anreicherung semantischer Informationen stetig verfeinert wird. Im Mittelpunkt stehen hierbei Eigenschaft-Werte-Paare, welche die Charakteristik eines Ereignisses in seinem Profil prägen. Diese Eigenschaften können Orte, Personen, Schlüsselwörter und Art des Ereignis sein. Diese Paare werden dabei gewichtet. Je öfter ein solches Paar in Nachrichten erwähnt wird, desto höher fällt sein Gewicht aus und gewinnt damit an Wichtigkeit für das Ereignis und sein Profil.

3. Auf Basis des Profils werden Bilder von Twitpic, Videos von Twitvid und Nachrichten von Twitter gesammelt und analysiert.

[7] http://anql.deri.org/anql.html
[8] http://twitcident.com
[9] http://twitter.com, http://twitpic.com, http://twitvid.com

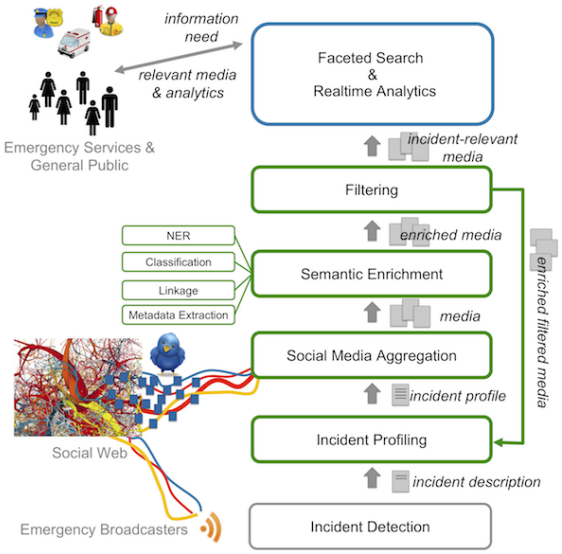

Abbildung 3.14: Die Architektur von twitcident. 1) Grün umrahmt laufen die automatisierten Prozesse zum Erstellen des Ereignisprofiles und der Filterung ab; 2) Blau umrahmt ist die vom User angestoßene und gesteuerte Facettensuche und Echtzeitanalyse

4. Mit der Semantischen Anreicherung, welches ein Schlüsselkonzept darstellt, werden Daten für das spätere semantische Filtern und die Facettensuche gewonnen. Dabei werden durch Named Entity Recognition mit Hilfe von beispielsweise DBPedia Spotlight[10] und Zemanta[11] Entitäten, wie Personen, Orte etc. aus Nachrichten extrahiert. Diese werden auf Konzepte abgebildet, die in der Menge der Eigenschaft-Werte-Paare ergänzt werden. Zusätzlich werden Verlinkungen in Twitter-Nachrichten verfolgt und die Web-Ressource ebenfalls auf Entitäten untersucht. Um die Vertraulichkeit der Nachrichten für User bei der Suche sichtbar zu machen werden dahingehend auch Metadaten, wie Zahl der Follower oder Hintergrundinformationen zum Autor ermittelt.

5. Twitter-Nachrichten werden mit den am höchsten gewichteten Eigenschaft-Werte-Paaren des Profils eines Ereignisses verglichen und dementsprechend nach Relevanz gefiltert.

Mit diesen automatisierten Schritten werden dem Benutzer nun Daten bereitgestellt, die er durchsuchen kann. Dabei werden die Eigenschaft-Werte-Paare als Facetten verwendet, die zu einem Ereignis gelistet und vom Benutzer ausgewählt werden können. Nun werden ihm nur Nachrichten präsentiert, die diese Facette erfüllen. Zusätzlich können Echtzeitanalysen über die Ausbreitung des Ereignisses im zeitlichen und geographischen Kontext angezeigt werden.
Für die Relevanzbestimmung von Twitter-Nachrichten wurden dabei in [Tao+12] Untersuchungen vorgestellt, welche Parameter einer Twitter-Nachricht Indikatoren für die Relevanz einer Nachricht sein können. Untersucht wurden dabei der Einfluss von Hashtags, Verlinkungen, Entitäten etc. . Dabei wurde festgestellt, dass vor al-

[10] http://dbpedia.org/spotlight
[11] http://zemanta.com

lem eine hohe Zahl von Entitäten und deren Vielfalt bzw. Diversität innerhalb einer Nachricht Indikatoren für eine zum gesuchten Thema relevante Nachricht sein können.
In Bezug auf diese Arbeit können die Gewichtung bzw. die Parameter für solch eine Gewichtung von Entitäten und Relevanz interessant sein. So können die zuletzt genannten Indikatoren zu einer Relevanzbestimmung von Artikeln in einer Suchanfrage beitragen. Außerdem sollten für aus Texten extrahierte Entitäten ebenfalls die Relevanz betrachtet werden, um deren Wichtigkeit im Kontext eines Artikels zu erschließen.

3.3.2 Named Entity Disambiguation durch semantischen Verwandtschaftsgrad

Zum Erschließen von Begrifssbedeutungen aus Texten wird in [Gen+10] eine Methode vorgestellt wie dies mit einem Graph-basierten Modells auf Daten von Wikipedia und daraus resultierenden Berechnungen von Verwandtschaftsgraden zwischen Entitäten erreicht werden kann. Hierbei wird auf einem Graph aus Begriffen eine Zufallsbewegung, auch *Random Walk* genannt, ausgeführt, um eine Matrix aufzustellen, aus denen so genannte *Semantic Relatedness Scores* berechnet werden. Dabei soll der Kontext nicht aus der zu einer Entität Nähe befindlichen Wörtern, sondern dem Zusammenhang aller im Text befindlichen Entitäten ermittelt werden. Im Wesentlichen laufen folgende Schritte ab:

1. Es werden so genannte *Named Entity Surfaces* durch eine Named Entity Recognition aus einem Text extrahiert. Diese Surfaces (im Folgenden Entität genannt) bilden die Grundlage für den entstehenden Graph.

2. Mit Hilfe der Entitäten werden Suchanfragen an Wikipedia gestellt, um alle Bedeutungen dieser Entitäten zu ermitteln. Liefert Wikipedia eine Ergebnisseite zurück, so hat die Entität nur eine Bedeutung. Wird eine Seite mit Begriffserklärungen zurückgegeben, so existieren mehrere Bedeutungen für die angefragte Entität.

3. Die erhaltenen Bedeutungen werden mit den Entitäten in einen Graph übernommen. Dazu wird der Link der jeweiligen Bedeutung verfolgt und ein Merkmalsraum für jede Entität aufgebaut, der ebenfalls in den Graphen eingefügt wird. Merkmale sind dabei Begriffe aus Titelbeschreibungen, im Text meist genutzte Wörtern, Kategorien und Begriffe aus dem Titel eines weiterführenden Links.

4. Auf dem entstandenen Graphen aus Entitäten, Bedeutungen und Merkmalsräumen wird eine Zufallsbewegung ausgeführt, um eine Wahrscheinlichkeitsverteilung von den Entitäten zueinander zu erhalten. Je länger es dauert bis eine Entität von einer anderen aus erreicht wird, desto unwahrscheinlicher ist ein Zusammenhang zwischen diesen Entitäten.

5. Die Kanten werden gewichtet und daraus eine *Relatedness Matrix R* aufgestellt, die beschreibt wie stark die Knoten miteinander in Verbindung stehen.

6. In der Arbeit werden nun verschiedene Funktionen vorgestellt, mit denen relevante Begriffe und damit der Kontext bestimmt wird. Die bei der Evaluation beste Methode summiert alle Relatedness-Werte von einer Entität zu jedem Begriff im Merkmalsraum anderer Entitäten.

Obwohl hier eine Methode zur Named Entity Disambiguation vorgestellt wurde, die im Rahmen dieser Arbeit nicht weiter betrachtet werden soll, bietet die Idee des Graph-basierten Modells Potential, um Beziehungen zwischen Entitäten zu bestimmen und dadurch Gewichtungen von Artikeln in den jeweiligen Kategorien bzw. Themen zu berechnen. Der Vorteil liegt darin, dass keine Ontologie mit Entitäten beschrieben werden muss und auch auf kein bisheriger Datenbestand notwendig ist um eine automatisierte Gewichtung zu erreichen.

3.4 Zusammenfassung

Bis heute ist kein Standard entwickelt worden, der sich mit der Modellierung von Unschärfe in semantischen Modellen beschäftigt, was sich die nächsten Jahre wohl auch nicht ändern wird. Jedoch, wie wir in 3.1 gesehen haben, gibt es gut funktionierende, auf Standards basierende Ansätze für die Modellierung von Unschärfe, die zwar keine fuzzy-logischen Regeln zum Schlussfolgern benutzen, aber für die Annotation von Gewichtungen vollkommen ausreichen. Die Reifikation bietet mit ihrem mitgelieferten Vokabular in RDF eine einfache Notation, allerdings sind die Daten redundant, wodurch es zu Problemen mit der Persistenz der Daten kommen kann. Annotationen (von Axiomen) in OWL sind ähnlich zu Reifikation, wobei diese aber ganz von Reasonern ignoriert werden und in ihrer teilweisen komplexen Notation Probleme bereiten können. Auch mit Standards nutzbare Erweiterungen, wie in 3.1.4 beschrieben haben für Anwendungen, in denen Unschärfe im Sinne der Fuzzy-Logik nur schwach eingesetzt wird, keinen größeren Nutzen, da viele Funktionen geboten werden, die nur in komplexen unscharfen Anwendungen lohnenswerten Einsatz finden. Positiv ist hierbei aber die Plugin-Unterstützung und die Möglichkeit des Reasonings mit dem externen Reasoner. Die Nutzung von n-ären Relationen hingegen ist auf mehrere Anwendungsfälle anwendbar und einfach umzusetzen und sehr empfehlenswert für weniger komplexe Fälle, die davon profitieren.
Bei Standarderweiterungen ist besonders aRDF hervorzuheben. Durch die Berücksichtigung von mehreren Domänen und Schlussfolgerungsregeln werden Funktionen bereitgestellt, die mit derzeitigen Standards nicht möglich sind, wohl aber in künftigen Standards berücksichtigt werden. Mit AnQL wird außerdem eine SPARQL-erweiternde Anfragesprache zur Verfügung gestellt, die leider noch nicht komplett entwickelt ist. N-Quads hingegen bieten zwar eine einfache Notation und einen entsprechenden Parser, sind aber für die Anwendungsdomäne von Unschärfe und Gewichtungen nicht wirklich von Nutzen.
Mit Hilfe des Twitcident-Systems und einem Graph-basierten Modell zur Extraktion von Entitäten und deren Relevanzbestimmung mit einem Random Walk wurden hilfreiche Erkenntnisse für die spätere Arbeit, insbesondere für die Berechnung von Relevanz, Gewichtungen und Rankings, gewonnen.

Literaturverzeichnis

[Abe+12] Fabian Abel, Claudia Hauff, GeertJan Houben u. a. *Semantics + Filtering + Search = Twitcident Exploring Information in Social Web Streams*. 2012 (siehe Seite 21).

[BB01] Dave Beckett und Art Barstow. *N-Triples*. http://www.w3.org/2001/sw/RDFCore/ntriples/. 2001 (siehe Seite 17).

[Bec+04] Sean Bechhofer, Frank van Harmelen, Jim Hendler u. a. *OWL Web Ontology Language: Reference*. http://www.w3.org/TR/owl-ref/#Annotations. 2004 (siehe Seite 14).

[BGM04] Dan Brickley, R.V. Guha und Brian McBride. *RDF Vocabulary Description Language 1.0: RDF Schema*. http://www.w3.org/TR/rdf-schema/. 2004 (siehe Seite 6).

[BS09] Fernando Bobillo und Umberto Straccia. *An OWL Ontology for Fuzzy OWL 2*. 2009 (siehe Seite 16).

[BS10] Fernando Bobillo und Umberto Straccia. *Representing Fuzzy Ontologies in OWL 2*. 2010 (siehe Seite 15).

[BS11] Fernando Bobillo und Umberto Straccia. *Fuzzy ontology representation using OWL 2*. http://gaia.isti.cnr.it/~straccia/software/FuzzyOWL/IJAR11.pdf. 2011 (siehe Seite 16).

[CHH09] Richard Cyganiak, Andreas Harth und Aidan Hogan. *N-Quads: Extending N-Triples with Context*. http://sw.deri.org/2008/07/n-quads/. 2009 (siehe Seite 17).

[Gen+10] Anna Lisa Gentile, Ziqi Zhang, Lei Xia u. a. *Semantic Relatedness approach for Named Entity Disambiguation*. 2010 (siehe Seite 23).

[Ger05] Dr. Thomas Gerick. *Die Zukunft des Semantic Web*. http://www.community-of-knowledge.de/beitrag/die-zukunft-des-semantic-web/. 2005 (siehe Seite 4).

[Gro09] W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview*. http://www.w3.org/TR/owl2-overview/. 2009 (siehe Seite 6).

[Gru09] Tom Gruber. »Ontology«. In: *Encyclopedia of Database Systems*. Springer-Verlag, 2009 (siehe Seite 5).

[GW] Christine Golbreich und Evan K. Wallace. *OWL 2 Web Ontology Language: New Features and Rationale*. http://www.w3.org/TR/2009/WD-owl2-primer-20090421/#Annotating_Axioms_and_Entities (siehe Seite 14).

[Hit+08] Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph u. a. *Semantic Web - Grundlagen*. Springer Verlag Berlin Heidelberg, 2008 (siehe Seite 6).

[HKR10] Pascal Hitzler, Markus Krötzsch und Sebastian Rudolph. *Foundations of Sementic Web Technologies*. Chapman und Hall/CRC, 2010 (siehe Seite 6).

[HM04] Patrick Hayes und Brian McBride. *RDF Semantics*. http://www.w3.org/TR/rdf-mt/. 2004 (siehe Seite 13).

[Hol09] Markus Holi. »Crisp, Fuzzy, and Probablistic Faceted Semantic Search«. Dissertation. School of Science und Technology, Aalto University, 2009 (siehe Seite 1).

[HS12] Steve Harris und Andy Seaborne. *SPARQL 1.1 Query Language*. http://www.w3.org/TR/sparql11-query/. 2012 (siehe Seite 7).

[KCM04] Graham Klyne, Jeremy J. Carroll und Brian McBride. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. http://www.w3.org/TR/rdf-concepts/. 2004 (siehe Seite 5).

[KR10] Markus Krötzsch und Sebastian Rudolph. *Webbasierte Informationssysteme - OWL 2*. http://dbis.informatik.uni-freiburg.de/content/courses/WS1011/Spezialvorlesung/WebbasierteInformationssysteme/folien/Vorlesung-OWL2.pdf. 2010 (siehe Seite 16).

[Las+08] Kenneth J. Laskey, Kathryn B. Laskey, Paulo C. G. Costa u. a. *Uncertainty Reasoning for the World Wide Web*. http://www.w3.org/2005/Incubator/urw3/XGR-urw3-20080331/. 2008 (siehe Seite 18).

[LBH01] Ora Lassila, Tim Berners-Lee und James Hendler. *The Semantic Web*. 2001 (siehe Seite 5).

[Lop+09] Nuno Lopes, Antoine Zimmermann, Aidan Hogan u. a. *RDF Needs Annotations*. http://www.w3.org/2009/12/rdf-ws/papers/ws09/. 2009 (siehe Seiten 13, 18).

[Lop+10] Nuno Lopes, Axel Polleres, Umberto Straccia u. a. *AnQL: SPARQLing Up Annotated RDFS*. 2010 (siehe Seiten 20, 21).

[MH04] Deborah L. McGuinness und Frank van Harmelen. *OWL Web Ontology Language Overview*. http://www.w3.org/TR/owl-features/. 2004 (siehe Seite 6).

[MM04] Frank Manola und Eric Miller. *RDF Primer*. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/. 2004 (siehe Seite 13).

[NR06] Natasha Noy und Alan Rector. *Defining N-ary Relations on the Semantic Web*. http://www.w3.org/TR/swbp-n-aryRelations/. 2006 (siehe Seite 11).

[PS08] Eric Prud'hommeaux und Andy Seaborne. *SPARQL Query Language for RDF*. http://www.w3.org/TR/rdf-sparql-query/. 2008 (siehe Seite 6).

[Str09] Umberto Straccia. *A Minimal Deductive System for General Fuzzy RDF*. http://gaia.isti.cnr.it/~straccia/download/papers/RR09/RR09.pdf/. 2009 (siehe Seite 19).

[Str+10] Umberto Straccia, Nuno Lopes, Gergely Lukácsy u. a. *A General Framework for Representing and Reasoning with Annotated Semantic Web Data*. http://gaia.isti.cnr.it/~straccia/download/papers/AAAI10/AAAI10.pdf/. 2010 (siehe Seiten 18, 19).

[Str11] Umberto Straccia. *Fuzzy Logic, Annotation Domains and Semantic Web Languages*. 2011 (siehe Seiten 4, 5).

[Tao+12] Ke Tao, Fabian Abel, Claudia Hauff u. a. *What makes a tweet relevant for a topic?* 2012 (siehe Seite 22).

[Zad65] Lofti A. Zadeh. *Fuzzy Sets*. 1965 (siehe Seite 5).

[Zad88] Lofti A. Zadeh. *Fuzzy Logic*. 1988 (siehe Seite 5).