

KnowNet

Idee für eine Software
Piratenpartei Schweiz

KnowNet, eine Ontologie und Suchmaschine für Dokumente und Statements

Lukas Zurschmiede
Bachelor of Science in Information technology

Lommis, 21. Juni 2012

Lukas Zurschmiede
Pirateparty Switzerland

Inhaltsverzeichnis

| | | |
|----------|-----------------------------|----------|
| 1 | Einleitung | 1 |
| 1.1 | Die Idee <i>KnowNet</i> | 1 |
| 1.2 | Beschreibung | 1 |
| 1.3 | Struktur und Aufbau | 2 |
| 1.3.1 | Technologien | 3 |
| 2 | Aufbau der Ontologie | 4 |
| 2.1 | Klassen | 4 |
| 2.1.1 | Klasse: Word | 7 |
| 2.1.2 | Klasse: Translation | 7 |
| 2.1.3 | Klasse: Synonym | 8 |
| 2.1.4 | Klasse: Sentence | 8 |
| 2.1.5 | Klasse: Document | 9 |
| 2.1.6 | Klasse: Paper | 9 |
| 2.1.7 | Klasse: Statement | 10 |
| 2.1.8 | Klasse: Journal | 10 |
| 2.1.9 | Klasse: Person | 11 |
| 2.1.10 | Klasse: Profession | 11 |
| 2.1.11 | Klasse: Organization | 12 |
| 2.1.12 | Klasse: Association | 12 |
| 2.2 | Relationen | 13 |
| 2.2.1 | Relation: refers-to | 16 |
| 2.2.2 | Relation: has-sentence | 17 |
| 2.2.3 | Relation: sentence-of | 17 |
| 2.2.4 | Relation: has-word | 18 |
| 2.2.5 | Relation: word-of | 18 |
| 2.2.6 | Relation: contains-word | 19 |
| 2.2.7 | Relation: has-writer | 20 |
| 2.2.8 | Relation: writer-of | 20 |
| 2.2.9 | Relation: has-subregion | 21 |
| 2.2.10 | Relation: subregion-of | 21 |
| 2.2.11 | Relation: has-workers | 22 |
| 2.2.12 | Relation: has-employee | 22 |
| 2.2.13 | Relation: employee-of | 23 |
| 2.2.14 | Relation: working-as | 24 |
| 2.2.15 | Relation: has-member | 25 |

| | | |
|----------|--------------------------------------|-----------|
| 2.2.16 | Relation: member-of | 25 |
| 2.2.17 | Relation: has-document | 26 |
| 2.2.18 | Relation: document-of | 27 |
| 2.2.19 | Relation: is-about | 28 |
| 2.2.20 | Relation: mentioned-in | 29 |
| 2.2.21 | Relation: means-same | 29 |
| 2.2.22 | Relation: is-same | 30 |
| 2.3 | Datenwerte | 30 |
| 2.3.1 | Datenwert: firstName | 31 |
| 2.3.2 | Datenwert: lastName | 32 |
| 2.3.3 | Datenwert: shortName | 32 |
| 2.3.4 | Datenwert: speaks | 32 |
| 2.3.5 | Datenwert: lang | 33 |
| 2.3.6 | Datenwert: word | 33 |
| 3 | Implementation | 34 |
| 3.1 | Architektur | 34 |
| 3.2 | Server | 34 |
| 3.3 | Client: Datenerfassung | 34 |
| 3.4 | Client: Suche und Abfragen | 34 |
| | Abkürzungsverzeichnis | A |
| | Abbildungsverzeichnis | A |
| | Tabellenverzeichnis | A |
| | Codeverzeichnis | B |
| | Literaturverzeichnis | H |

1 Einleitung

1.1 Die Idee *KnowNet*

KnowNet - Knowledge Network - ist primär eine Ontologie, welche Dokumente von und über eine politische Partei sammelt und entsprechend aufbereitet anbietet. Die Idee dahinter ist, dass man durch eine einfache, eventuell neuartige Suchoberfläche, Aussagen und Meinungen aber auch Dokumente von einer Partei, respektive über eine Partei finden kann. Eine Partei kann Unterparteien (Sektionen) beinhalten, welche somit bei einem lokalen Thema die Ontologie abfragen können, ob zu diesem Thema schon ein Statement oder Dokument vorhanden ist oder ob eine andere Sektion schon etwas darüber gesagt hat, aber auch, ob es eventuell andere Organisationen wie Medienhäuser gibt, welche der Partei eine Aussage diesbezüglich unterstellt haben.

Die Ontologie soll schlussendlich eine einfache Möglichkeit bieten, Dinge über eine komplexe Struktur zu erfahren um daraus Schlussfolgerungen und Thesen auf zu stellen, jedoch auch um heraus zu finden, ob denn schon eine der Sektionen eine Meinung oder Aussage über ein gewisses Thema gemacht hat.

1.2 Beschreibung

Die Ontologie kann durch folgende Aussagen beschrieben werden:

- Die Ontologie besteht hauptsächlich aus Dokumenten, Personen, Organisationen und Vereinen
- Ein Dokument kann ein (Positions-)Papier, ein Statement oder ein Journalistisches Werk sein
- Ein Verein kann Untervereine haben - und somit auch Übergeordnete
- Eine Person kann einem oder mehreren Vereinen angehören
- Ist eine Person Mitglied eines Vereines, darf sie deren Meinung öffentlich vertreten
- Eine Person kann einen oder mehrere Berufe ausüben

- Eine Person kann in einer oder mehreren Organisationen arbeiten
- Ein Dokument das einem Verein zugeordnet ist, beschreibt deren Ansichten
- Ein Dokument das einer Organisation zugeordnet ist, beschreibt die Ansicht dieser eines Vereins
- Ein Dokument ist immer von einem oder mehreren Personen geschrieben
- Ein Dokument besteht aus Sätzen
- Ein Satz besteht aus Wörtern
- Ein Wort kann in verschiedenen Sprachen vorkommen (Übersetzungen)
- Ein Wort kann verschiedene Bedeutungen haben (Synonyme)
- Eine Person wird durch den name, Vornamen und den Nickname identifiziert
- Eine Person muss mindestens eine Sprache sprechen, kann aber auch mehrere
- Ein Wort ist immer genau einer Sprache zugeordnet
- Ein Dokument ist immer genau einer Sprache zugeordnet
- Ein Word ist eine Abfolge von Zeichen

1.3 Struktur und Aufbau

Es bietet sich an, *KnowNet* durch eine Ontologie ab zu bilden. Ontologien die auf dem OWL-Standard[1] basieren, bieten die Möglichkeit, mittels einer in den Grundzügen einfachen Sprache eine Wissensdatenbank komplex verknüpft zu speichern und managen. OWL ist eine Erweiterung von RDF/S und ist, wie RDF auch, durch die zwei Syntaxen *Turtle* respektive *N3* und *XML* definiert. Obwohl *XML* im Vergleich zu *Turtle* einiges mehr "drum herum" hat, ist es dennoch einfacher zu erstellen und auch zu pflegen als *Turtle*. Auch kann die Ontologie in dieser Syntax durch andere Programme einfacher gelesen werden und ist auch einfacher wartbar, da Syntaxfehler durch gute Editoren schneller erkannt werden respektive das ganze in einem Browser betrachtet werden kann.

OWL ist zur Zeit in Version 1 und in Version 2 definiert. Der OWL 2 Standard[2] ist komplett Rückwärtskompatibel zu OWL 1, enthält jedoch Erweiterungen, welche in der Ursprungsversion schlicht gefehlt haben. Einige der Neuerungen sind rein syntaktischer Natur, andere sind Funktionen und Features, welche bislang gefehlt haben. Die Ontologie für *KnowNet* wird noch mit OWL 1 implementiert, da nicht klar zu sehen ist, ob die verwendeten Libraries (siehe Technologien auf Seite 3) auch OWL 2 verstehen. Dies muss

in einem kurzen Test geprüft werden und anschliessend gegebenenfalls die Ontologie auch OWL 2 gewandelt werden, da die dortige Syntax ein wenig leserlicher ist.

1.3.1 Technologien

Es werden folgende Technologien und Standards verwendet:

Ontologie OWL 1, Subsprache OWL DL (Description Logic), als Ontologiebeschreibungssprache

Abfrage SPARQL[4] zur Abfrage

Programmierung C/C++ mit Qt4[6]

Eventuell HTML 5, Javascript als Client oder zur Abfrage mittels REST

Libraries Redland RDF Libraries[5]

Datenbank PostgreSQL und MySQL Serverseitig, SQLite zum Entwickeln und eventuell auf dem Client wenn notwendig

Die Liste ist offen und wird entsprechend angepasst wenn bei der Entwicklung und/oder Planung gemerkt wird, dass andere Dinge und Libraries eingesetzt oder nicht verwendet werden.

2 Aufbau der Ontologie

2.1 Klassen

Folgende Klassen und Spezialisierungen werden für die Ontologie gebraucht. In den nachfolgenden Kapiteln werden diese noch genauer beschrieben. Die Verlinkungen zwischen den einzelnen Klassen, also die ObjectProperties, werden im Kapitel ?? auf Seite ?? beschrieben. In den jeweiligen Detailbeschreibungen der Klassen wird jeweils darauf eingegangen und darauf verwiesen. Die Datenfelder, also die DataProperties, werden in den Detailbeschreibungen der Klassen beschrieben.

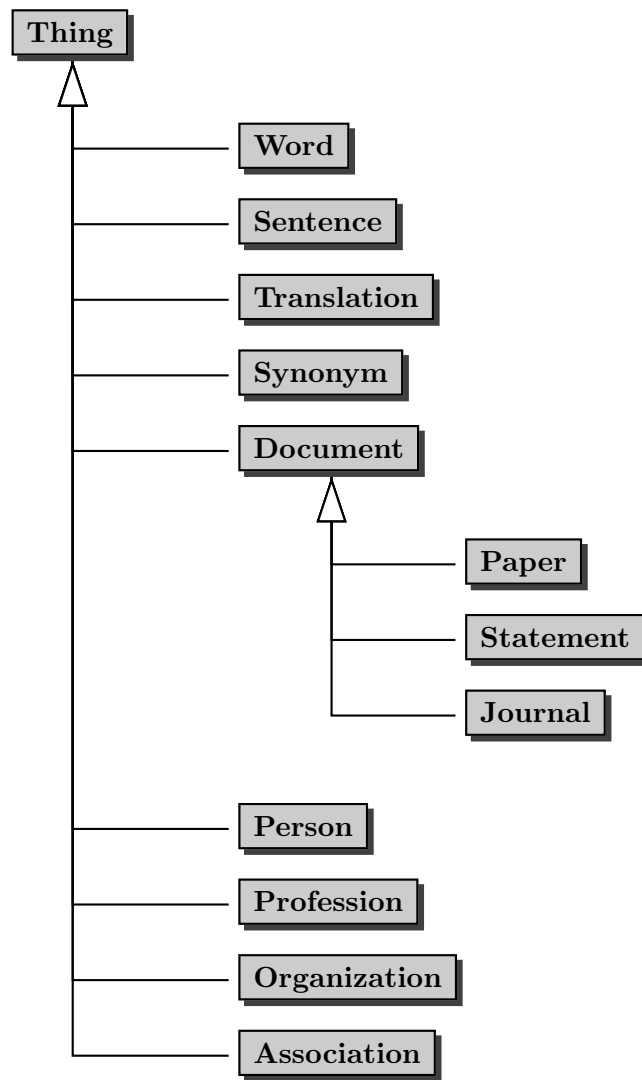


Abbildung 2.1: Die OWL-Klassen im Überblick

| Class | Beschreibung | SubClass of | Details |
|---------------------|--|-------------|-----------------------------------|
| Word | Beschreibt ein einzelnes Wort in einer Sprache, es kann auch aus mehreren wörtern zusammengesetzt sein; Anhand einer ID kann ein Link zu einer anderen Sprache oder einemSynonym hergestellt werden; | Thing | Klasse: Word auf Seite 7 |
| Translation | Gruppirt die gleichen Wörter aus verschiedenen Sprachen | Thing | Klasse: Translation auf Seite 7 |
| Synonym | Gruppirt unterschiedliche Wörter der gleichen Sprache mit der gleichen Bedeutung | Thing | Klasse: Synonym auf Seite 8 |
| Sentence | Ein Satz besteht aus verschiedenen Wörtern | Thing | Klasse: Sentence auf Seite 8 |
| Document | Ein Dokument ist eine Sammlung von Sätzen inkl. einigen weiteren Attributen | Thing | Klasse: Document auf Seite 9 |
| Paper | Ein Positionspapier oder eine andere Dokumentation einer Partei | Document | Klasse: Paper auf Seite 9 |
| Statement | Eine Aussage, welche ein Mitglied einer Partei gemacht hat und welche so vertreten wird | Document | Klasse: Statement auf Seite 10 |
| Journal | Ein Zeitungsbericht | Document | Klasse: Journal auf Seite 10 |
| Person | Eine Person im Allgemeinen | Thing | Klasse: Person auf Seite 11 |
| Profession | Beschreibt einen Beruf, welcher eine Person ausüben kann | Person | Klasse: Profession auf Seite 11 |
| Organization | Eine Organisation/Medienhaus welches Zeitungen etc. herstellt, Arbeitgeber eines Journalisten | Thing | Klasse: Organization auf Seite 12 |
| Association | Eine (politische) Partei, welche Mitglieder und Angestellte hat | Thing | Klasse: Association auf Seite 12 |

6
Tabelle 2.1: Kurze Beschreibung der Klassen in der Ontologie

2.1.1 Klasse: Word

Die Klasse *Word* wird verwendet, um ein einzelnes Wort, welches innerhalb eines Dokumentes gefunden werden soll, zu definieren. Bei zusammengesetzten Wörtern kann mit der Relation *contains-word* (siehe Relation: contains-word Seite 19) angegeben werden, welche einzelnen Wörter in diesem Wort vorkommen. Durch das Datenfeld *language* wird die Sprache des jeweiligen Wortes in iso639-1 (zwei Zeichen) definiert. Um Übersetzungen miteinander zu verknüpfen, wird die Klasse *Translation* (siehe Klasse: Translation Seite 7) verwendet. Durch die Relation *wordtype* wird die Art des Wortes definiert, also ob es sich um ein Nomen, ein Adjektiv oder ein Verb handelt.

Code

```
1: <owl:Class rdf:ID="Word">
2: </owl:Class>
```

Codeblock 2.1: Die Klasse *Word* beschreibt ein einzelnes Wort

2.1.2 Klasse: Translation

Die Klasse *Translation* ist eine Vereinigung aller Wörter verschiedener Sprachen mit der selben Bedeutung. Dies geschieht durch die Angabe *owl:minCardinality*, welche aussagt, dass eine Übersetzung mindestens zwei Wörter in der Range des Properties *is-same* aufweisen muss. Durch dieses Konstrukt können neue Wörter einfach und schnell eingepflegt und als Übersetzung definiert werden ohne die Wörter selber zu manipulieren, sowie bei einer Abfrage in der Liste gesucht werden.

Code

```
1: <owl:Class rdf:ID="Translation">
2:   <rdfs:subClassOf>
3:     <owl:Restriction>
4:       <owl:onProperty rdf:resource="#is-same" />
5:       <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">2</
        owl:minCardinality>
6:     </owl:Restriction>
7:   </rdfs:subClassOf>
8: </owl:Class>
```

Codeblock 2.2: Die Klasse *Translation* beinhaltet alle Übersetzungen eines Wortes

2.1.3 Klasse: *Synonym*

Die Klasse *Synonym* dient zur Verkettung von Wörtern mit der gleichen Bedeutung. Dies geschieht durch die Angabe `owl:minCardinality`, welche aussagt, dass ein Synonym mindestens zwei Wörter in der Range des Properties *means-same* aufweisen muss. Durch dieses Konstrukt können neue Wörter einfach und schnell eingepflegt und als Synonyme definiert werden ohne die Wörter selber zu manipulieren, sowie bei einer Abfrage in der Liste gesucht werden.

Code

```
1: <owl:Class rdf:ID="Synonym">
2:   <rdfs:subClassOf>
3:     <owl:Restriction>
4:       <owl:onProperty rdf:resource="#means-same" />
5:       <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">2</
        owl:minCardinality>
6:     </owl:Restriction>
7:   </rdfs:subClassOf>
8: </owl:Class>
```

Codeblock 2.3: Die Klasse *Synonym* beinhaltet alle Wörter die etwas ähnliches bedeuten

2.1.4 Klasse: *Sentence*

Ein Satz, welcher durch die Klasse *Sentence* definiert wird, ist eine Vereinigung von Wörtern zu einer Gruppe von solchen. Die Reihenfolge der Wörter ist hierbei egal, es geht nur um das Zusammenspiel und den Zusammenhang einzelner Wörter. Dieses Konstrukt wird bei der Instanz im Property *has-word* (siehe Relation: *has-word* Seite 18) durch eine anonyme Klasse mit `owl:unionOf` definiert.

Code

```
1: <owl:Class rdf:ID="Sentence">
2:   <rdfs:subClassOf>
3:     <owl:Restriction>
4:       <owl:onProperty rdf:resource="#has-word" />
5:       <owl:someValuesFrom ref:resource="#Word" />
6:     </owl:Restriction>
7:   </rdfs:subClassOf>
8: </owl:Class>
```

Codeblock 2.4: Die Klasse *Sentence* fügt einzelne Wörter zu einem Satz zusammen

2.1.5 Klasse: Document

Die Klasse *Document* ist, ähnlich wie *Sentence*, eine Kombination von verschiedenen Sätzen. Die Reihenfolge dieser spielt keine Rolle, es geht rein um den Logischen Zusammenhang der einzelnen Sätze und somit der verschiedenen Wörter.

Anmerkung: Eventuell macht es hier Sinn, ein Dokument noch durch Kapitel zu strukturieren. Dies kann durch die Relation "ein Dokument aus Dokumenten" gemacht werden.

Code

```
1: <owl:Class rdf:ID="Document">
2:   <rdfs:subClassOf>
3:     <owl:Restriction>
4:       <owl:onProperty rdf:resource="#has-sentence" />
5:       <owl:someValuesFrom ref:resource="#Sentence" />
6:     </owl:Restriction>
7:   </rdfs:subClassOf>
8: </owl:Class>
```

Codeblock 2.5: Die Klasse *Document* fügt einzelne Sätze zu einem Dokument zusammen

2.1.6 Klasse: Paper

Die Klasse *Paper* ist eine Unterklasse von *Document* und definiert ein Dokument, welches im Zusammenhang zu einem Verein, also der Klasse *Association* steht. Durch diese Unterklasse kann besser nach Dokumenten gesucht werden, welche von einer bestimmten politischen Partei oder einer Sektion von dieser, verfasst wurden. Ein Dokument, welches

durch diese Klasse implementiert ist, beschreibt in den meisten Fällen ein Positionspapier oder ähnliches.

Code

```
1: <owl:Class rdf:ID="Paper">
2:   <rdfs:subClassOf rdf:resource="#Document" />
3: </owl:Class>
```

Codeblock 2.6: Die Klasse *Paper* ist ein Dokument einer *Association*, also eines
partei/Gesellschaft

2.1.7 Klasse: Statement

Die Klasse *Statement* ist eine Unterklasse von *Document* und wird verwendet, um eine Aussage/Statement einer Person zu definieren. Bei einem Statement muss bei der Suche unterschieden werden, ob dieses von einer Person aus der Klasse *Association* stammt, oder von einer beliebigen anderen Person. Ein Statement von einer beliebigen Person kann genutzt werden, um sich ein Bild von Aussen zu machen, während eines von einer Person aus *Association* genutzt werden kann, um sich ein Bild von innen zu machen.

Durch das Property *from-text* (siehe ?? auf Seite ??) kann ein Statement einem beliebigen anderen *Document* zugeordnet werden.

Code

```
1: <owl:Class rdf:ID="Statement">
2:   <rdfs:subClassOf rdf:resource="#Document" />
3: </owl:Class>
```

Codeblock 2.7: Ein *Statement* beschreibt eine Aussage einer Person

2.1.8 Klasse: Journal

Ein *Journal* ist eine Unterklasse von *Document* und definiert einen Text, welcher von einem *Writer* oder *Blogger* geschrieben und Veröffentlicht worden ist. Er stellt nicht eine Meinung der *Association* dar, sondern diejenige eines Aussenstehenden. Ist der Author ebenfalls Mitglied der Klasse *Member*, kann der Text als persönliche Meinung und somit indirekt als Parteimeinung gedeutet werden.

Code

```
1: <owl:Class rdf:ID="Journal">
2:   <rdfs:subClassOf rdf:resource="#Document" />
3: </owl:Class>
```

Codeblock 2.8: Ein *Journal* ist ein Bericht über eine *Association*, welcher von einer *Organization* herausgegeben wurde

2.1.9 Klasse: Person

Die Klasse *Person* wird verwendet, um irgend eine Person zu definieren. Dies kann ein Autor, ein Parteimitglied oder eine andere beliebige Person sein.

Code

```
1: <owl:Class rdf:ID="Person">
2: </owl:Class>
```

Codeblock 2.9: Die Klasse *Person* definiert alle Personen

2.1.10 Klasse: Profession

Die Klasse *Profession* wird verwendet, um einer *Person* einen Beruf/Anstellung zuzuweisen. Dadurch kann man gegebenenfalls gewissen Berufsgruppen gewissen Meinungen zuweisen.

Die Zuweisung erfolgt durch die Relation *working-as* (siehe Relation: *working-as* auf Seite 24).

Code

```
1: <owl:Class rdf:ID="Profession">
2:   <rdfs:subClassOf>
3:     <owl:Restriction>
4:       <owl:onProperty rdf:resource="#has-workers" />
5:       <owl:someValuesFrom ref:resource="#Person" />
6:     </owl:Restriction>
7:   </rdfs:subClassOf>
8: </owl:Class>
```

Codeblock 2.10: Die Klasse *Profession* beschreibt jegliche Berufe

2.1.11 Klasse: Organization

Die Klasse *Organization* definiert den Arbeitgeber von einer *Person*. Durch diese Zuweisung können Aussagen und Meinungen, welche eine Zeitung von der Partei hat, eruiert werden. Eine *Organization* kann mehrere Angestellte haben, welche nicht zwingendermassen nur bei der einen Organisation angestellt sind. Idealerweise besitzt die *Person* ebenfalls das Attribut *working-as* mit einem Verweis auf einen Beruf.

Eine Instanz kann nie gleichzeitig eine *Organization* und eine *Association* sein, diese zwei Klassen schliessen sich gegenseitig aus.

Code

```
1: <owl:Class rdf:ID="Organization">
2:   <rdfs:subClassOf>
3:     <owl:Restriction>
4:       <owl:onProperty rdf:resource="#has-employee" />
5:       <owl:someValuesFrom ref:resource="#Person" />
6:     </owl:Restriction>
7:   </rdfs:subClassOf>
8:   <owl:disjointWith rdf:resource="#Association" />
9: </owl:Class>
```

Codeblock 2.11: Die Klasse *Organization* definiert einen Arbeitgeber, meistens ein Medienhaus

2.1.12 Klasse: Association

Eine *Association* definiert eine Politische Partei oder andere Gesellschaft, über welche man sich durch diese Ontologie ein Bild verschaffen können soll.

Eine Instanz kann nie gleichzeitig eine *Organization* und eine *Association* sein, diese zwei Klassen schliessen sich gegenseitig aus.

Code

```
1: <owl:Class rdf:ID="Association">
2:   <rdfs:subClassOf>
3:     <owl:Restriction>
4:       <owl:onProperty rdf:resource="#has-member" />
5:       <owl:someValuesFrom ref:resource="#Member" />
6:     </owl:Restriction>
7:   </rdfs:subClassOf>
8:   <owl:disjointWith rdf:resource="#Organization" />
9: </owl:Class>
```

Codeblock 2.12: Eine *Association* ist eine politische Partei oder eine andere Gesellschaft

2.2 Relationen

Die Relationen, resp. Properties, beschreiben die Abhängigkeiten wie auch die Verknüpfungen unter den Klassen. Relationen, welche klassenspezifisch sind, also direkt gebraucht werden um eine Klasse zu beschreiben, sind als anonyme Unterklassen bereits in diesen definiert worden. Hier werden nur diejenigen Relationen beschrieben, welche verwendet werden zur direkten dynamischen und individuellen Verknüpfung verschiedener Instanzen.

| Relation | Beschreibung | Domain | Range | Details |
|----------------------|--|--------------------|--------------------|--------------------------------------|
| refers-to | Beziehung zwischen einzelnen Dokumenten | <i>Document</i> | <i>Document</i> | Relation: refers-to auf Seite 16 |
| has-sentence | Gibt an, aus welchen Sätzen ein Dokument besteht | <i>Document</i> | <i>Sentence</i> | Relation: has-sentence auf Seite 17 |
| sentence-of | Gibt an, in welchem Dokument dieser Satz vorhanden ist | <i>Sentence</i> | <i>Document</i> | Relation: sentence-of auf Seite 17 |
| has-word | Gibt an, aus welchen Wörtern ein Satz besteht | <i>Sentence</i> | <i>Word</i> | Relation: has-word auf Seite 18 |
| word-of | Gibt an, in welchem Satz ein Wort vorhanden ist | <i>Word</i> | <i>Sentence</i> | Relation: word-of auf Seite 18 |
| contains-word | Gibt bei einem zusammengesetzten Wort alle einzelnen Wörter an | <i>Word</i> | <i>Word</i> | Relation: contains-word auf Seite 19 |
| has-writer | Definiert einen oder mehrere Autoren eines Dokuments | <i>Document</i> | <i>Person</i> | Relation: has-writer auf Seite 20 |
| writer-of | Definiert die Dokumente, an welchen eine Person mitgearbeitet hat | <i>Person</i> | <i>Document</i> | Relation: writer-of auf Seite 20 |
| has-subregion | Definiert die Sub-Associations der aktuellen | <i>Association</i> | <i>Association</i> | Relation: has-subregion auf Seite 21 |
| subregion-of | Definiert, dass diese Association eine Sub-Association der angegebenen ist | <i>Association</i> | <i>Association</i> | Relation: subregion-of auf Seite 21 |
| has-workers | Gibt alle Personen eines Berufes an | <i>Profession</i> | <i>Person</i> | Relation: has-workers auf Seite 22 |

Tabelle 2.2: Kurze Beschreibung der Objekt-Relationen in der Ontologie

| Relation | Beschreibung | Domain | Range | Details |
|---------------------|---|----------------------------------|----------------------------------|-------------------------------------|
| has-employee | Gibt alle Employees der Organization an | <i>Organization, Association</i> | <i>Employee</i> | Relation: has-employee auf Seite 22 |
| employee-of | Gibt an, bei welchen Organizations der Employee angestellt ist | <i>Employee</i> | <i>Organization, Association</i> | Relation: employee-of auf Seite 23 |
| has-member | Definiert alle Mitglieder einer Association | <i>Association</i> | <i>Member</i> | Relation: has-member auf Seite 25 |
| member-of | Definiert in welcher Association der Member Mitglied ist | <i>Member</i> | <i>Association</i> | Relation: member-of auf Seite 25 |
| has-document | gibt an, welche Association oder Organization ein dokument verfasst hat | <i>Association, Organization</i> | <i>Document</i> | Relation: has-document auf Seite 26 |
| document-of | gibt an, zu welcher Association oder Organization ein Dokument gehört | <i>Document</i> | <i>Association, Organization</i> | Relation: document-of auf Seite 27 |
| is-about | Definiert, dass das aktuelle Dokument etwas über die angegebene Association aussagt | <i>Document</i> | <i>Association</i> | Relation: is-about auf Seite 28 |
| mentioned-in | Gibt an, in welchem Dokument auf die Association referenziert wird | <i>Association</i> | <i>Document</i> | Relation: mentioned-in auf Seite 29 |
| means-same | Definiert, welche Wörter das gleiche meinen, also Synonyme sind | <i>Synonym</i> | <i>Word</i> | Relation: means-same auf Seite 29 |
| is-same | Gibt an, welche Wörter die gleiche Bedeutung haben in unterschiedlichen Sprachen | <i>Translation</i> | <i>Word</i> | Relation: is-same auf Seite 30 |

Tabelle 2.3: Fortsetzung: Kurze Beschreibung der Objekt-Relationen in der Ontologie

Mögliche Eigenschaften

Relationen können folgende Charakteristiken aufweisen:

Funktional Eine funktionale Relation P impliziert:

wenn $P(u, v)$ und $P(u, w)$ dann $v == w$

Invers Funktional Eine inverse funktionale Relation P impliziert:

wenn $P(v, u)$ und $P(w, u)$ dann $v == w$

Transitiv Wenn eine Relation P *transitiv* definiert ist für u, v, w :

wenn $P(u, v)$ und $P(v, w)$ impliziert $P(u, w)$

Symmetrisch Wenn eine Relation P symmetrisch definiert ist:

wenn $P(u, v)$ dann $P(v, u)$

InverseOf Wenn bei der Relation $P1$ eine Inverse Relation $P2$ definiert ist:

wenn $P1(u, v)$ dann $P2(v, u)$

Wichtig: Die Eigenschaften *Reflexiv*, *Irreflexiv* und *Asymmetrisch* aus der OWL-2 Definition werden nicht verwendet.

2.2.1 Relation: refers-to

Die Relation *refers-to* beschreibt die Beziehungen zwischen den einzelnen Dokumenten. Wird zum Beispiel in einem Blog, was ja als *Document* interpretiert ist, auf einen Artikel aus einer Zeitung verwiesen, so kann dies durch diese Relation definiert werden. Die Relation ist sowohl *symmetrisch* als auch *transitiv* und das inverse von sich selbst.

Eigenschaften

- Transitiv
- Symmetrisch
- inverseOf *refers-to*

Code

```
1: <owl:ObjectProperty rdf:ID="refers-to">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <rdf:type rdf:resource="&owl;SymmetricProperty" />
4:   <owl:inverseOf rdf:resource="#refers-to" />
5:   <rdfs:domain rdf:resource="#Document" />
6:   <rdfs:range rdf:resource="#Document" />
7: </owl:ObjectProperty>
```

Codeblock 2.13: Die Relation *refers-to* beschreibt die Abhängigkeiten unter Dokumenten

2.2.2 Relation: has-sentence

Die Relation *has-sentence* gibt an, welche *Sentence* Instanzen in einem *Document* vorkommen.

Eigenschaften

- inverseOf *sentence-of*

Code

```
1: <owl:ObjectProperty rdf:ID="has-sentence">
2:   <owl:inverseOf rdf:resource="#sentence-of" />
3:   <rdfs:domain rdf:resource="#Document" />
4:   <rdfs:range rdf:resource="#Sentence" />
5: </owl:ObjectProperty>
```

Codeblock 2.14: Die Relation *has-sentence* gibt an, welches Dokument aus welchen Sätzen besteht

2.2.3 Relation: sentence-of

Die Relation *sentence-of* gibt an, in welchem *Document* ein *Sentence* vorkommt.

Eigenschaften

- Transitiv
- inverseOf *has-sentence*

Code

```
1: <owl:ObjectProperty rdf:ID="sentence-of">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <owl:inverseOf rdf:resource="#has-sentence" />
4:   <rdfs:domain rdf:resource="#Sentence" />
5:   <rdfs:range rdf:resource="#Document" />
6: </owl:ObjectProperty>
```

Codeblock 2.15: Die Relation *sentenceof* gibt an, in welchem Dokument ein Satz vorkommt

2.2.4 Relation: has-word

Die Relation *has-word* gibt an, welche *Word* Instanz in der *Sentence* Instanz verwendet wird. Die Reihenfolge der Wörter in den Sätzen spielt keine Rolle, es kommt nur auf den Sinn der Wörter an.

Eigenschaften

- Transitiv
- inverseOf *word-of*

Code

```
1: <owl:ObjectProperty rdf:ID="has-word">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <owl:inverseOf rdf:resource="#word-of" />
4:   <rdfs:domain rdf:resource="#Sentence" />
5:   <rdfs:range rdf:resource="#Word" />
6: </owl:ObjectProperty>
```

Codeblock 2.16: Die Relation *has-word* gibt an, welches Wort in einem Satz vorkommt

2.2.5 Relation: word-of

Die Relation *word-of* gibt an, in welchen *Sentence* Instanzen ein *Word* verwendet wird. Die Reihenfolge der Wörtern in den Sätzen spielt keine Rolle, es kommt nur auf den Sinn der Wörter an.

Eigenschaften

- Transitiv

- inverseOf *has-word*

Code

```

1: <owl:ObjectProperty rdf:ID="word-of">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <owl:inverseOf rdf:resource="#has-word" />
4:   <rdfs:domain rdf:resource="#Word" />
5:   <rdfs:range rdf:resource="#Sentence" />
6: </owl:ObjectProperty>

```

Codeblock 2.17: Die Relation *word-of* gibt an, in welchem Satz das Wort vorkommt

2.2.6 Relation: contains-word

Die Relation *contains-word* gibt bei einem zusammengesetzten Wort alle *Word*-Instanzen an, welche in diesem Wort verwendet werden. So kann zum Beispiel definiert werden, dass das Wort „Hausboot“ aus den zwei Wörtern „Haus“ und „Boot“ besteht, selber aber auch ein Wort ist.

Eigenschaften

- Transitiv

Code

```

1: <owl:ObjectProperty rdf:ID="contains-word">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <rdfs:domain rdf:resource="#Word" />
4:   <rdfs:range>
5:     <owl:Class>
6:       <owl:someOf rdf:parseType="Collection">
7:         <owl:Class rdf:about="#Word" />
8:       </owl:someOf>
9:     </owl:Class>
10:   </rdfs:range>
11: </owl:ObjectProperty>

```

Codeblock 2.18: Die Relation *contains-word* gibt an, aus welchen Wörtern das Wort besteht

2.2.7 Relation: has-writer

Die Relation *has-writer* definiert alle *Person* Instanzen, welche an dem aktuellen *Document* gearbeitet haben.

Eigenschaften

- Transitiv
- inverseOf *writer-of*

Code

```
1: <owl:ObjectProperty rdf:ID="has-writer">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <owl:inverseOf rdf:resource="#writer-of" />
4:   <rdfs:domain rdf:resource="#Document" />
5:   <rdfs:range rdf:resource="#Person" />
6: </owl:ObjectProperty>
```

Codeblock 2.19: Die Relation *has-writer* gibt an, welche *Person* an dem *Document* geschrieben hat

2.2.8 Relation: writer-of

Die Relation *writer-of* definiert die Verlinkung einer *Person* mit einem *Document*. Daraus kann gelesen werden, welche Person welches Dokument erstellt, respektive daran mitgearbeitet hat.

Eigenschaften

- Transitiv
- inverseOf *has-writer*

Code

```
1: <owl:ObjectProperty rdf:ID="writer-of">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <owl:inverseOf rdf:resource="#has-writer" />
4:   <rdfs:domain rdf:resource="#Person" />
5:   <rdfs:range rdf:resource="#Document" />
6: </owl:ObjectProperty>
```

Codeblock 2.20: Die Relation *writer-of* gibt an, welche *Person* an welchem *Document* gearbeitet hat

2.2.9 Relation: has-subregion

Die Relation *has-subregion* definiert alle *Association* Instanzen, welche unterhalb der aktiven angelegt sind. Instanzen von *Association* können in einer Baumstruktur angeordnet werden. Diese Property definiert die Verlinkung von Oben nach Unten.

Eigenschaften

- Transitiv
- Inverse Funktional
- inverseOf *subregion-of*

Code

```
1: <owl:ObjectProperty rdf:ID="has-subregion">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <rdf:type rdf:resource="&owl;InverseFunctionalProperty" />
4:   <owl:inverseOf rdf:resource="#subregion-of" />
5:   <rdfs:domain rdf:resource="#Association" />
6:   <rdfs:range rdf:resource="#Association" />
7: </owl:ObjectProperty>
```

Codeblock 2.21: Die Relation *has-subregion* definiert, welche Untersektionen eine *Association* hat

2.2.10 Relation: subregion-of

Die Relation *subregion-of* definiert die Eltern-Instanz der aktiven *Association* Instanz. Instanzen von *Association* können in einer Baumstruktur angeordnet werden. Dieses Property definiert die Verlinkung von einer Unten nach Oben.

Eigenschaften

- Transitiv
- inverseOf *has-subregion*

Code

```
1: <owl:ObjectProperty rdf:ID="subregion-of">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <owl:inverseOf rdf:resource="#has-subregion" />
4:   <rdfs:domain rdf:resource="#Association" />
5:   <rdfs:range rdf:resource="#Association" />
6: </owl:ObjectProperty>
```

Codeblock 2.22: Die Relation *subregion-of* definiert die übergeordnete *Association*

2.2.11 Relation: has-workers

Die Relation *has-workers* definiert alle *Person*-Instanzen, welche die gegebenen *Profession* ausführen oder beherrschen, also den Beruf einer Person. Eine *Person* kann bei mehreren *Profession*-Instanzen zugewiesen sein.

Eigenschaften

- keine

Code

```
1: <owl:ObjectProperty rdf:ID="has-workers">
2:   <rdfs:domain rdf:resource="#Profession" />
3:   <rdfs:range rdf:resource="#Person" />
4: </owl:ObjectProperty>
```

Codeblock 2.23: Die Relation *has-workers* gibt an, welche *Person* welche *Profession* beherrscht oder ausübt

2.2.12 Relation: has-employee

Die Relation *has-employee* definiert alle *Person* Instanzen, welche bei der gegebenen *Organization* oder *Assocaition* arbeiten. Eine *Person* kann bei mehreren *Organization* angestellt sein.

Eigenschaften

- Transitiv
- inverseOf *employee-of*

Code

```
1: <owl:ObjectProperty rdf:ID="has-employee">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <owl:inverseOf rdf:resource="#employee-of" />
4:   <rdfs:domain>
5:     <owl:Class>
6:       <owl:unionOf rdf:parseType="Collection">
7:         <owl:Class rdf:about="#Organization" />
8:         <owl:Class rdf:about="#Association" />
9:       </owl:unionOf>
10:    </owl:Class>
11:  </rdfs:domain>
12:  <rdfs:range rdf:resource="#Person" />
13: </owl:ObjectProperty>
```

Codeblock 2.24: Die Relation *has-employee* gibt an, welche *Person* bei der *Organization* oder *Association* angestellt sind

2.2.13 Relation: employee-of

Die Relation *employee-of* weist einer *Person* einen Arbeitgeber zu. Eine *Person*, welche eine solche Zuweisung aufweist, sollte entsprechend auch über die Relation *working-as* verfügen, um so zu definieren, welchen Beruf diese Person ausübt.

Der Arbeitgeber kann entweder eine *Organization* oder aber eine *Association* sein. Eine *Person* kann mehrere Arbeitgeber aufweisen.

Eigenschaften

- Transitiv
- inverseOf *has-employee*

Code

```
1: <owl:ObjectProperty rdf:ID="employee-of">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <owl:inverseOf rdf:resource="#has-employee" />
4:   <rdfs:domain rdf:resource="#Person" />
5:   <rdfs:range>
6:     <owl:Class>
7:       <owl:unionOf rdf:parseType="Collection">
8:         <owl:Class rdf:about="#Organization" />
9:         <owl:Class rdf:about="#Association" />
10:      </owl:unionOf>
11:    </owl:Class>
12:  </rdfs:range>
13: </owl:ObjectProperty>
```

Codeblock 2.25: Die Relation *employee-of* gibt an, bei welcher *Organization* oder *Association* die Person angestellt ist

2.2.14 Relation: working-as

Die Relation *working-as* weist einer *Person* einen oder mehrere Berufe zu. Durch dieses Property kann beispielsweise geprüft werden, ob gewisse Meinungen oder Richtungen nur in gewissen Berufsgattungen vorherrschen, oder ob dies die breitere Masse betrifft.

Eigenschaften

- Transitiv
- inverseOf *working-as*

Code

```
1: <owl:ObjectProperty rdf:ID="working-as">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <owl:inverseOf rdf:resource="#working-as" />
4:   <rdfs:domain rdf:resource="#Person" />
5:   <rdfs:range rdf:resource="#Profession" />
6: </owl:ObjectProperty>
```

Codeblock 2.26: Die Relation *working-as* gibt an, welchen Beruf eine Person ausübt

2.2.15 Relation: has-member

Die Relation *has-member* enthält alle *Person* Instanzen, welche Mitglied bei der gegebenen *Association* sind.

Eigenschaften

- Transitiv
- inverseOf *member-of*

Code

```
1: <owl:ObjectProperty rdf:ID="has-member">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <owl:inverseOf rdf:resource="#member-of" />
4:   <rdfs:domain rdf:resource="#Association" />
5:   <rdfs:range rdf:resource="#Person" />
6: </owl:ObjectProperty>
```

Codeblock 2.27: Die Relation *has-member* enthält alle Mitglieder der *Association*

2.2.16 Relation: member-of

Die Relation *member-of* weist eine *Person* einer *Association* zu. Dadurch wird definiert, dass diese Person Mitglied der Partei, resp. Gesellschaft ist und deren Meinung offiziell vertreten darf. Eine *Person* kann mehreren *Association* zugewiesen werden.

Eigenschaften

- Transitiv
- inverseOf *has-member*

Code

```
1: <owl:ObjectProperty rdf:ID="member-of">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <owl:inverseOf rdf:resource="#has-member" />
4:   <rdfs:domain rdf:resource="#Person" />
5:   <rdfs:range rdf:resource="#Association" />
6: </owl:ObjectProperty>
```

Codeblock 2.28: Die Relation *member-of* gibt alle *Association* an, bei welcher die Person Mitglied ist

2.2.17 Relation: has-document

Die Relation *has-document* gibt an, welche *Organization* oder *Association* welches *Document* veröffentlicht hat. Das Property beinhaltet in der Domain also entweder eine Instanz einer *Organization* oder einer *Association* und in der Range ein *Document*. Da in der Range unterschiedliche Typen vorkommen können, müssen diese in der Klassendefinition (siehe Klasse: *Organization* und Klasse: *Association*) explizit als Unterschiedliche Klassen definiert werden.

Eigenschaften

- Transitiv
- inverseOf *document-of*

Code

```
1: <owl:ObjectProperty rdf:ID="has-document">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <owl:inverseOf rdf:resource="#document-of" />
4:   <rdfs:domain>
5:     <owl:Class>
6:       <owl:unionOf rdf:parseType="Collection">
7:         <owl:Class rdf:about="#Organization" />
8:         <owl:Class rdf:about="#Association" />
9:       </owl:unionOf>
10:    </owl:Class>
11:  </rdfs:domain>
12:  <rdfs:range rdf:resource="#Document" />
13: </owl:ObjectProperty>
```

Codeblock 2.29: Die Relation *has-document* verknüpft eine *Organization* mit einem *Document*

2.2.18 Relation: document-of

Die Relation *document-of* gibt an, welche *Organization* oder *Association* der "Eigentümer" eines Dokumentes ist. Es wird also definiert, wer ein Dokument veröffentlicht hat. Das Property beinhaltet demnach in der Domain eine Instanz eines *Document* und in der Range entweder eine *Organization* oder eine *Association*. Da in der Range unterschiedliche Typen vorkommen können, müssen diese in der Klassendefinition (siehe Klasse: *Organization* und Klasse: *Association*) explizit als Unterschiedliche Klassen definiert werden.

Eigenschaften

- Transitiv
- inverseOf *has-document*

Code

```
1: <owl:ObjectProperty rdf:ID="document-of">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <owl:inverseOf rdf:resource="#has-document" />
4:   <rdfs:domain rdf:resource="#Document" />
5:   <rdfs:range>
6:     <owl:Class>
7:       <owl:unionOf rdf:parseType="Collection">
8:         <owl:Class rdf:about="#Organization" />
9:         <owl:Class rdf:about="#Association" />
10:      </owl:unionOf>
11:    </owl:Class>
12:  </rdfs:range>
13: </owl:ObjectProperty>
```

Codeblock 2.30: Die Relation *document-of* gibt an, welche *Organization* oder *Association* das *Document* veröffentlicht hat

2.2.19 Relation: is-about

Die Relation *is-about* beschreibt, welche *Association* Instanzen im gegebenen *Document* erwähnt wird. Das Property enthält also eine Instanz eines *Document* in der domain und eine Liste von *Association* Instanzen in der Range.

Eigenschaften

- Transitiv
- inverseOf *mentioned-in*

Code

```
1: <owl:ObjectProperty rdf:ID="is-about">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <owl:inverseOf rdf:resource="#mentioned-in" />
4:   <rdfs:domain rdf:resource="#Document" />
5:   <rdfs:range rdf:resource="#Association" />
6: </owl:ObjectProperty>
```

Codeblock 2.31: Die Relation *is-about* gibt an, über welche *Association* ein Dokument ist

2.2.20 Relation: mentioned-in

Die Relation *mentioned-in* gibt an, in welchen *Document* Instanzen die *Association* erwähnt wird. Das Property enthält also eine Instanz einer *Association* in der Domain und eine Liste von *Document* Instanzen in der Range. Je nach Dokument-Instanz, also der Unterklasse, kann ein internes oder externes Meinungsbild geschaffen werden. Zu beachten ist auch noch das Property *member-of* wie auch *writer-of*, welche die Autoren definieren und deren Parteizugehörigkeit.

Eigenschaften

- Transitiv
- inverseOf *is-about*

Code

```
1: <owl:ObjectProperty rdf:ID="mentioned-in">
2:   <rdf:type rdf:resource="#owl:TransitiveProperty" />
3:   <owl:inverseOf rdf:resource="#is-about" />
4:   <rdfs:domain rdf:resource="#Association" />
5:   <rdfs:range rdf:resource="#Document" />
6: </owl:ObjectProperty>
```

Codeblock 2.32: Die Relation *mentioned-in* gibt alle *Document* an, in welcher eine *Association* erwähnt wird

2.2.21 Relation: means-same

Die Relation *means-same* wird auf einem *Synonym* verwendet und definiert alle *Word*-Instanzen, welche die gleiche Bedeutung haben.

Eigenschaften

- Transitiv

Code

```
1: <owl:ObjectProperty rdf:ID="means-same">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <rdfs:domain rdf:resource="#Synonym" />
4:   <rdfs:range rdf:resource="#Word" />
5: </owl:ObjectProperty>
```

Codeblock 2.33: Die Relation *means-same* definiert auf *Synonym*-Instanzen alle Wörter

2.2.22 Relation: is-same

Die Relation *is-same* wird auf einer *Translation* verwendet und definiert alle *Word*-Instanzen, welche in verschiedenen Sprachen die gleiche Aussage haben.

Eigenschaften

- Transitiv

Code

```
1: <owl:ObjectProperty rdf:ID="is-same">
2:   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3:   <rdfs:domain rdf:resource="#Translation" />
4:   <rdfs:range rdf:resource="#Word" />
5: </owl:ObjectProperty>
```

Codeblock 2.34: Die Relation *is-same* definiert auf *Translation*-Instanzen alle Wörter

2.3 Datenwerte

Datenwerte, auch *Datentypen-Properties* genannt, werden verwendet um den einzelnen Klassen Eigenschaften zuzuweisen, respektive die einzelnen Instanzen mit "Daten" zu bestücken. Die Datentypen werden sowohl bei den *Relationen* wie auch bei den Klassen-internen Properties verwendet und müssen jeweils bei der Instanzierung einer Instanz angegeben werden.

Standardmässig sind alle Datentypen verfügbar, welche im XML-Schema[3] verfügbar sind. Diese können in den Datentypen-Relationen als Variabler oder vordefiniertes Wert verwendet werden, oder als Liste aus Werten (Enumeration). Eine Datentyp-Relation

besitzt, wie die Objekt-Relation, immer eine *domain* und eine *range*, sodass ein solches an eine Klasse oder Objekt-Relation gebunden werden kann.

| Relation | Beschreibung | Domain | Range | Details |
|------------------|---------------------------------------|-----------------------|---------------------|-----------------------------------|
| firstName | Vorname / Rufname der Person | <i>Person</i> | <i>xsd:string</i> | Datenwert: firstName auf Seite 31 |
| lastName | Nachname / Familienname der Person | <i>Person</i> | <i>xsd:string</i> | Datenwert: lastName auf Seite 32 |
| shortName | Nickname der Person, wenn vorhanden | <i>Person</i> | <i>xsd:string</i> | ?? auf Seite ?? |
| speaks | Sprache(n), welche die Person spricht | <i>Person</i> | <i>xsd:language</i> | Datenwert: speaks auf Seite 32 |
| lang | Sprache des Wortes / des Dokuments | <i>Word, Document</i> | <i>xsd:language</i> | Datenwert: lang auf Seite 33 |
| word | Das Wort | <i>Word</i> | <i>xsd:string</i> | Datenwert: word auf Seite 33 |

Tabelle 2.4: Kurze Beschreibung der Datentypen-Properties in der Ontologie

2.3.1 Datenwert: **firstName**

Der Datenwert *firstName* beinhaltet den Vornamen/Rufnamen der Person und ist vom Typ *xsd:string*. Der Vorname unterliegt keinen Restriktionen.

Code

```

1: <owl:DatatypeProperty rdf:ID="firstName">
2:   <rdfs:domain rdf:resource="#Person"/>
3:   <rdfs:range rdf:resource="&xsd:string"/>
4: </owl:DatatypeProperty>

```

Codeblock 2.35: Der Datenwert *firstName* gibt den Vornamen einer Person an.

2.3.2 Datenwert: *lastName*

Der Datenwert *lastName* beinhaltet den Nachnamen/Familiennamen der Person und ist vom Typ *xsd:string*. Der Nachname unterliegt keinen Restriktionen.

Code

```
1: <owl:DatatypeProperty rdf:ID="lastName">
2:   <rdfs:domain rdf:resource="#Person" />
3:   <rdfs:range rdf:resource="&xsd:string" />
4: </owl:DatatypeProperty>
```

Codeblock 2.36: Der Datenwert *lastName* gibt den Nachnamen einer Person an.

2.3.3 Datenwert: *shortName*

Der Datenwert *shortName* beinhaltet den Nicknamen der Person und ist vom Typ *xsd:string*. Der Nickname unterliegt keinen Restriktionen.

Code

```
1: <owl:DatatypeProperty rdf:ID="shortName">
2:   <rdfs:domain rdf:resource="#Person" />
3:   <rdfs:range rdf:resource="&xsd:string" />
4: </owl:DatatypeProperty>
```

Codeblock 2.37: Der Datenwert *shortName* gibt den Nicknamen einer Person an.

2.3.4 Datenwert: *speaks*

Der Datenwert *speaks* beinhaltet die Sprache der Person und ist vom Typ *xsd:language*, welches eine Anwendung des RFC-1766 ist, also die zweistellige Angabe einer Sprache.

Code

```
1: <owl:DatatypeProperty rdf:ID="speaks">
2:   <rdfs:domain rdf:resource="#Person" />
3:   <rdfs:range rdf:resource="&xsd:language" />
4: </owl:DatatypeProperty>
```

Codeblock 2.38: Der Datenwert *speaks* gibt die Sprache einer Person an.

2.3.5 Datenwert: lang

Der Datenwert *lang* beinhaltet die Sprache von Wörtern und Dokumenten und ist vom Typ *xsd:language*, welches eine Anwendung des RFC-1766 ist, also die zweistellige Angabe einer Sprache.

Code

```
1: <owl:DatatypeProperty rdf:ID="lang">
2:   <rdfs:domain>
3:     <owl:Class>
4:       <owl:unionOf rdf:parseType="Collection">
5:         <owl:Class rdf:about="#Word" />
6:         <owl:Class rdf:about="#Document" />
7:       </owl:unionOf>
8:     </owl:Class>
9:   </rdfs:domain>
10:  <rdfs:range rdf:resource="&xsd;language" />
11: </owl:DatatypeProperty>
```

Codeblock 2.39: Der Datenwert *lang* gibt die Sprache eines Wortes oder Dokumentes an.

2.3.6 Datenwert: word

Der Datenwert *word* beinhaltet das Wort der *Word*-Instanzen an sich und ist vom Typ *xsd:string*. Das Wort unterliegt keinen Restriktionen.

Code

```
1: <owl:DatatypeProperty rdf:ID="word">
2:   <rdfs:domain rdf:resource="#Word" />
3:   <rdfs:range rdf:resource="&xsd:string" />
4: </owl:DatatypeProperty>
```

Codeblock 2.40: Der Datenwert *word* beinhaltet das Word einer *Word*-Instanz.

3 Implementation

3.1 Architektur

3.2 Server

3.3 Client: Datenerfassung

3.4 Client: Suche und Abfragen

Abbildungsverzeichnis

| | |
|--|---|
| 2.1 Die OWL-Klassen im Überblick | 5 |
|--|---|

Tabellenverzeichnis

| | | |
|-----|---|----|
| 2.1 | Kurze Beschreibung der Klassen in der Ontologie | 6 |
| 2.2 | Kurze Beschreibung der Objekt-Relationen in der Ontologie | 14 |
| 2.3 | Fortsetzung: Kurze Beschreibung der Objekt-Relationen in der Ontologie | 15 |
| 2.4 | Kurze Beschreibung der Datentypen-Properties in der Ontologie | 31 |

Codeblock-Verzeichnis

| | | |
|------|---|----|
| 2.1 | Die Klasse <i>Word</i> beschreibt ein einzelnes Wort | 7 |
| 2.2 | Die Klasse <i>Translation</i> beinhaltet alle Übersetzungen eines Wortes | 7 |
| 2.3 | Die Klasse <i>Synonym</i> beinhaltet alle Wörter die etwas ähnliches bedeuten . | 8 |
| 2.4 | Die Klasse <i>Sentence</i> fügt einzelne Wörter zu einem Satz zusammen | 9 |
| 2.5 | Die Klasse <i>Document</i> fügt einzelne Sätze zu einem Dokument zusammen . | 9 |
| 2.6 | Die Klasse <i>Paper</i> ist ein Dokument einer <i>Association</i> , also eines partei/- Gesellschaft | 10 |
| 2.7 | Ein <i>Statement</i> beschreibt eine Aussage einer Person | 10 |
| 2.8 | Ein <i>Journal</i> ist ein Bericht über eine <i>Association</i> , welcher von einer <i>Or-</i> <i>ganization</i> herausgegeben wurde | 11 |
| 2.9 | Die Klasse <i>Person</i> definiert alle Personen | 11 |
| 2.10 | Die Klasse <i>Caption</i> beschreibt jegliche Berufe | 11 |
| 2.11 | Die Klasse <i>Organization</i> definiert einen Arbeitgeber, meistens ein Me- dienhaus | 12 |
| 2.12 | Eine <i>Association</i> ist eine politische Partei oder eine andere Gesellschaft . | 13 |
| 2.13 | Die Relation <i>refers-to</i> beschreibt die Abhängigkeiten unter Dokumenten . | 17 |
| 2.14 | Die Relation <i>has-sentence</i> gibt an, welches Dokument aus welchen Sätzen besteht | 17 |
| 2.15 | Die Relation <i>sentenceof</i> gibt an, in welchem Dokument ein Satz vorkommt | 18 |
| 2.16 | Die Relation <i>has-word</i> gibt an, welches Wort in einem Satz vorkommt . . | 18 |
| 2.17 | Die Relation <i>word-of</i> gibt an, in welchem Satz das Wort vorkommt | 19 |
| 2.18 | Die Relation <i>contains-word</i> gibt an, aus welchen Wörtern das Wort besteht | 19 |
| 2.19 | Die Relation <i>has-writer</i> gibt an, welche <i>Person</i> an dem <i>Document</i> ge- schrieben hat | 20 |
| 2.20 | Die Relation <i>writer-of</i> gibt an, welche <i>Person</i> an welchem <i>Document</i> ge- arbeitet hat | 21 |
| 2.21 | Die Relation <i>has-subregion</i> definiert, welche Untersektionen eine <i>Associa-</i> <i>tion</i> hat | 21 |
| 2.22 | Die Relation <i>subregion-of</i> definiert die übergeordnete <i>Association</i> | 22 |

| | | |
|------|--|----|
| 2.23 | Die Relation <i>has-workers</i> gibt an, welche <i>Person</i> welche <i>Profession</i> beherrscht oder ausübt | 22 |
| 2.24 | Die Relation <i>has-employee</i> gibt an, welche <i>Person</i> bei der <i>Organization</i> oder <i>Association</i> angestellt sind | 23 |
| 2.25 | Die Relation <i>employee-of</i> gibt an, bei welcher <i>Organization</i> oder <i>Association</i> die Person angestellt ist | 24 |
| 2.26 | Die Relation <i>working-as</i> gibt an, welchen Beruf eine Person ausübt | 24 |
| 2.27 | Die Relation <i>has-member</i> enthält alle Mitglieder der <i>Association</i> | 25 |
| 2.28 | Die Relation <i>member-of</i> gibt alle <i>Association</i> an, bei welcher die Person Mitglied ist | 26 |
| 2.29 | Die Relation <i>has-document</i> verknüpft eine <i>Organization</i> mit einem <i>Document</i> | 27 |
| 2.30 | Die Relation <i>document-of</i> gibt an, welche <i>Organization</i> oder <i>Association</i> das <i>Document</i> veröffentlicht hat | 28 |
| 2.31 | Die Relation <i>is-about</i> gibt an, über welche <i>Association</i> ein Dokument ist . | 28 |
| 2.32 | Die Relation <i>mentioned-in</i> gibt alle <i>Document</i> an, in welcher eine <i>Association</i> erwähnt wird | 29 |
| 2.33 | Die Relation <i>means-same</i> definiert auf <i>Synonym</i> -Instanzen alle Wörter . . | 30 |
| 2.34 | Die Relation <i>is-same</i> definiert auf <i>Translation</i> -Instanzen alle Wörter . . . | 30 |
| 2.35 | Der Datenwert <i>firstName</i> gibt den Vornamen einer Person an. | 31 |
| 2.36 | Der Datenwert <i>lastName</i> gibt den Nachnamen einer Person an. | 32 |
| 2.37 | Der Datenwert <i>shortName</i> gibt den Nicknamen einer Person an. | 32 |
| 2.38 | Der Datenwert <i>speaks</i> gibt die Sprache einer Person an. | 32 |
| 2.39 | Der Datenwert <i>lang</i> gibt die Sprache eines Wortes oder Dokumentes an. . | 33 |
| 2.40 | Der Datenwert <i>word</i> beinhaltet das Word einer <i>Word</i> -Instanz. | 33 |

Index

- Assocaition, 22
- Association, C, D, 9–15, 21–29
- Asymmetrisch, 16
- Blogger, 10
- Caption, C, 11
- contains-word, 7, 19
- contsins-word, C, 19
- Datentypen-Properties, 30
- Datenwerte, 30
- Document, C, D, 9, 10, 14–17, 20, 21, 26–29, 31
- document-of, D, 26–28
- domain, 31
- Employee, 15
- employee-of, D, 23, 24
- firstName, D, 31
- from-text, 10
- has-document, D, 26, 27
- has-employee, D, 22, 23
- has-member, D, 25
- has-sentence, C, 17
- has-subregion, C, 21, 22
- has-word, C, 8, 18, 19
- has-workers, D, 22
- has-writer, C, 20
- Irreflexiv, 16
- is-about, D, 28, 29
- is-same, D, 7, 30
- Journal, C, 10, 11
- KnowNet, 1, 2
- lang, D, 33
- language, 7
- lastName, D, 32
- means-same, D, 8, 29, 30
- Member, 10, 15
- member-of, D, 25, 26, 29
- mentioned-in, D, 28, 29
- N3, 2
- Organization, C, D, 11, 12, 15, 22–24, 26–28
- OWL 2 Web Ontology Language Primer, F
- OWL Web Ontology Language Reference, F
- Paper, C, 9, 10
- Person, C, D, 11, 12, 14, 20–25, 31
- Profession, D, 11, 14, 22
- Qt-Cross-Platform application and UI framework, F

- range, 31
- Redland RDF Libraries, F
- refers-to, C, 16, 17
- Reflexiv, 16
- Relationen, 30

- Sentence, C, 8, 9, 14, 17, 18
- sentence-of, 17
- sentenceof, C, 18
- shortName, D, 32
- SPARQL Query Language for RDF, F
- speaks, D, 32
- Statement, C, 10
- subregion-of, C, 21, 22
- symmetrisch, 16
- Synonym, C, D, 8, 15, 29, 30

- transitiv, 16
- Translation, C, D, 7, 15, 30
- Turtle, 2

- Word, C, D, 7, 14, 15, 18, 19, 29–31, 33
- word, D, 33
- word-of, C, 18, 19
- wordtype, 7
- working-as, D, 11, 12, 23, 24
- Writer, 10
- writer-of, C, 20, 21, 29

- XML, 2
- XML Schema Part 2: Datatypes Second
Edition, F
- xsd:language, 31–33
- xsd:string, 31–33

Literaturverzeichnis

- [1] W3C, *OWL Web Ontology Language Reference*, 10. Februar 2004, <http://www.w3.org/TR/owl-ref/>
- [2] W3C, *OWL 2 Web Ontology Language Primer*, 27. Oktober 2009, <http://www.w3.org/TR/owl2-primer/>
- [3] W3C, *XML Schema Part 2: Datatypes Second Edition*, 28. Oktober 2004, <http://www.w3.org/TR/xmlschema-2/>
- [4] W3C, *SPARQL Query Language for RDF*, 15. Januar 2008, <http://www.w3.org/TR/rdf-sparql-query/>
- [5] Dave Beckett, Redland, *Redland RDF Libraries*, 2012, <http://librdf.org/>
- [6] Nokia, OpenSource, *Qt-Cross-Platform application and UI framework*, 2012, <http://qt.nokia.com/>