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 Die Idee *KnowNet*

KnowNet - Knowledge Network - ist primär eine Ontologie, welche Dokumente von und über ein 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 rekursiv Untersektionen beinhalten, welche somit bei einem lokalen Thema die Ontologie abfragen können, ob zu diesem Thema schon ein Statement oder Dokument vorhanden ist, welche andere Sektion schon etwas darüber gesagt hat, aber auch, ob es eventuell andere Organisatinen wie Medienhäuser gibt, welche eine Aussage diesbezüglich der Partei 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 Sektione eine Meinung oder Uassage über ein gewisses Thema gemacht hat.

1.2 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 an, mittels einer in den Grundzügen einfachen Sprache eine Wissensdatenbank komplex verknüpft zu speichern und darauf Abfragen aus zu führen. OWL ist eine Erweiterung von RDF/S und ist wie dieses 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 denn *Turtle*. Auch kann die Ontologie in dieser Syntak durch andere Programme einfacher gelesen werden und ist auch einfacher wartbar, da Syntaxfehler durch gute Editoren schneller erkennt werden.

OWL ist zur Zeit in Version 1 und in Version 2 definiert. Der OWL 2 Standard[2] ist komplett Rückwärtskompatibel zu OWL 1, enthällt jedoch Erweiterungen, welche in

der Ursprungsversion schlicht gefehlt haben. Einige der Neuerungen sind schlicht syntaktischer Natur, andere sind funktionen und Features welche bislang gefehlt haben. Die Ontologie für *KnowNet* wird noch mit OWL 1 implementiert, wenn jedoch die verwendeten Libraries (siehe Technologien auf Seite 2) kann diese auch nach OWL 2 konvertiert, respektive deren Neuerungen verwendet werden.

1.2.1 Technologien

Es werden Folgende Technologien und Standards verwendet:

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

Abfrage SPARQL[3] zur Abfrage

Programmierung C/C++ mit Qt4[5]

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

Libraries Redland RDF Libraries[4]

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.

- Thing
  - Word
  - Sentence
  - Translation
  - Synonym
  - Document
    - Paper
    - Statement
    - Journal
  - Person
  - Profession
  - Organization
  - Association

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 einenmSynonym hergestellt werden;	Thing	Klasse: Word auf Seite 5
Translation	Gruppiert die gleichen Wörter aus verschiedenen Sprachen	Thing	Klasse: Translation auf Seite 5
Synonym	Gruppiert unterschiedliche Wörter der gleichen Sprache mit der gleichen Bedeutung	Thing	Klasse: Synonym auf Seite 5
Sentence	Ein Satz besteht aus verschiedenen Wörtern	Thing	Klasse: Sentence auf Seite 6
Document	Ein Dokument ist eine Sammlung von Sätzen inkl. einigen weiteren Attributen	Thing	Klasse: Document auf Seite 6
Paper	Ein Positionspapier oder eine andere Dokumentation einer Partei	Document	Klasse: Paper auf Seite 7
Statement	Eine Aussage, welche ein Mitglied einer Partei gemacht hat und welche so vertreten wird	Document	Klasse: Statement auf Seite 7
Journal	Ein Zeitungsbericht	Document	Klasse: Journal auf Seite 8
Person	Eine Person im Allgemeinen	Thing	Klasse: Person auf Seite 8
Profession	Beschreibt einen Beruf, welcher eine Person ausüben kann	Person	Klasse: Profession auf Seite 9
Organization	Eine Organisation/Medienhaus welches Zeitungen etc. herstellt, Arbeitgeber eines Journalisten	Thing	Klasse: Organization auf Seite 9
Association	Eine (politische) Partei, welche Mitglieder und Angestellte hat	Thing	Klasse: Association auf Seite 10

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. Einzelne Wörter können mit der Relation *wordcombo* (siehe **??** Seite **??**) zu Wortkombinationen zusammengesetzt werden. 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 5) 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 das Konstrukt `owl:oneOf` und der Angabe aller Individuen. Durch dieses Konstrukt können neue Wörter einfach und schnell eingepflegt und übersetzt werden und bei einer Abfrage kann jeweils eines aus der Liste gesucht werden.

Code

```
1: <owl:Class rdf:ID="#Translation">
2:   <owl:oneOf rdf:parseType="Collection">
3:     <Word rdf:about="#Word_Instance_1" />
4:     <Word rdf:about="#Word_Instance_2" />
5:     ...
6:   </owl:oneOf>
7: </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 das Konstrukt `owl:oneOf` und der Angabe aller Individuen. Durch

dieses Konstrukt können neue Wörter einfach und schnell eingepflegt und als Synonyme definiert werden, sowie bei einer Abfrage in der Liste gesucht werden.

Code

```
<owl:Class rdf:ID="#Synonym">
  <owl:oneOf rdf:parseType="Collection">
    <Word rdf:about="#Word_Instance_1" />
    <Word rdf:about="#Word_Instance_2" />
    ...
  </owl:oneOf>
</owl:Class>
```

Codeblock 2.3: Die Klasse *Synonym* beinhaltet alle Wörter dir 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 14) durch eine anonyme Klasse mit `owl:unionOf` definiert.

Code

```
<owl:Class rdf:ID="#Sentence">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#has-word" />
      <owl:someValuesFrom ref:resource="#Word" />
    </owl:Restriction>
  </rdfs:subClassOf>
</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

```
<owl:Class rdf:ID="#Document">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#has-sentence" />
      <owl:someValuesFrom ref:resource="#Sentence" />
    </owl:Restriction>
  </rdfs:subClassOf>
</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

```
<owl:Class rdf:ID="#Paper">
  <rdfs:subClassOf rdf:resurce="#Document" />
</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 19).

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 *Caption* 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 enem Verweis auf einen Beruf.

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

Code

```
<owl:Class rdf:ID="#Organization">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#has-employee" />
      <owl:someValuesFrom ref:resource="#Person" />
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#Association" />
</owl:Class>
```

Codeblock 2.11: Die Klasse *Organization* definiert einen Arbeitgeben, 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

```
<owl:Class rdf:ID="#Association">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#has-member" />
      <owl:someValuesFrom ref:resource="#Member" />
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#Organization" />
</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 einzenlnen Dokumenten	*Document*	*Document*	Relation: refers-to auf Seite 13
has-sentence	Gibt an, aus welchen Sätzen ein Dokument besteht	*Document*	*Sentence*	Relation: has-sentence auf Seite 13
sentence-of	Gibt an, in welchem Dokuemnt dieser Satz vorhanden ist	*Sentence*	*Document*	Relation: sentence-of auf Seite 14
has-word	Gibt an, aus welchen Wörtern ein Satz besteht	*Sentence*	*Word*	Relation: has-word auf Seite 14
word-of	Gibt an, in welchem Satz ein Wort vorhanden ist	*Word*	*Sentence*	Relation: word-of auf Seite 15
combined-in	Gibt an, dass ein Wort mit einem anderen verknüpft werden kann	*Word*	*Word*	**??** auf Seite **??**
has-writer	Definiert einen oder mehrere Autoren eines Dokuments	*Document*	*Person*	Relation: has-writer auf Seite 15
writer-of	Definiert die Dokumente, an welchen eine Person mitgearbeitet hat	*Person*	*Document*	Relation: writer-of auf Seite 16
has-subregion	Definiert die Sub-Associations der aktuellen	*Association*	*Association*	Relation: has-subregion auf Seite 16
subregion-of	Definiert, dass diese Association eine Sub-Association der angegebenen ist	*Association*	*Association*	Relation: subregion-of auf Seite 17

Tabelle 2.2: Kurze Beschreibung der Relationen in der Ontologie

Relation	Beschreibung	Domain	Range	Details
has-employee	Gibt alle Employees der Organization an	*Organization*, *Association*	*Employee*	Relation: has-employee auf Seite 17
employee-of	Gibt an, bei welchen Organizations der Employee angestellt ist	*Employee*	*Organization*, *Association*	Relation: employee-of auf Seite 18
has-member	Definiert alle Mitglieder einer Association	*Association*	*Member*	Relation: has-member auf Seite 19
member-of	Definiert in welcher Association der Member Mitglied ist	*Member*	*Association*	Relation: member-of auf Seite 20
has-document	gibt an, welche Associaton oder Organization ein dokument verfasst hat	*Association*, *Organization*	*Document*	Relation: has-document auf Seite 20
document-of	gibt an, zu welcher Association oder Organization ein Dokument gehört	*Document*	*Association*, *Organization*	Relation: document-of auf Seite 21
is-about	Definiert, dass das aktuelle Dokument etwas über die angegebene Assoiciation aussagt	*Document*	*Association*	Relation: is-about auf Seite 22
mentioned-in	Gibt an, in welchem Dokument auf die Association referenziert wird	*Association*	*Document*	Relation: mentioned-in auf Seite 22

Tabelle 2.3: Kurze Beschreibung der 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* interpertiert ist, auf einen Artikel uas 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

```
<owl:ObjectProperty rdf:ID="#has-sentence">
  <rdfs:domain rdf:resource="#Document" />
  <rdfs:range rdf:resource="#Sentence" />
</owl:ObjectProperty>
```

Codeblock 2.14: Die Relation *has-sentence* gibt an, welches Dokuemnt 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

```
<owl:ObjectProperty rdf:ID="#sentence-of">
  <rdfs:domain rdf:resource="#Sentence" />
  <rdfs:range rdf:resource="#Document" />
</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

```
<owl:ObjectProperty rdf:ID="#has-word">
  <rdfs:domain rdf:resource="#Sentence" />
  <rdfs:range rdf:resource="#Word" />
</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

```
<owl:ObjectProperty rdf:ID="#word-of">
  <rdfs:domain rdf:resource="#Word" />
  <rdfs:range rdf:resource="#Sentence" />
</owl:ObjectProperty>
```

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

2.2.6 Relation: has-writer

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

Eigenschaften

- Transitiv
- inverseOf *writer-of*

Code

```
<owl:ObjectProperty rdf:ID="#has-writer">
  <rdfs:domain rdf:resource="#Document" />
  <rdfs:range rdf:resource="#Person" />
</owl:ObjectProperty>
```

Codeblock 2.18: Die Relation *has-writer* gibt an, welche *Person* an dem *Document* geschrieben hat

2.2.7 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

```
<owl:ObjectProperty rdf:ID="#writer-of">
  <rdfs:domain rdf:resource="#Person" />
  <rdfs:range rdf:resource="#Document" />
</owl:ObjectProperty>
```

Codeblock 2.19: Die Relation *writer-of* gibt an, welche *Person* an welchem *Document* gearbeitet hat

2.2.8 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:   <rdfs:domain rdf:resource="#Association" />
3:   <rdfs:range rdf:resource="#Association" />
4: </owl:ObjectProperty>
```

Codeblock 2.20: Die Relation *has-subregion* definiert, welche Untersektionen eine *Association* hat

2.2.9 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:   <rdfs:domain rdf:resource="#Association" />
3:   <rdfs:range rdf:resource="#Associaton" />
4: </owl:ObjectProperty>
```

Codeblock 2.21: Die Relation *subregion-of* definiert die übergeordnete *Association*

2.2.10 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

```
<owl:ObjectProperty rdf:ID="#has-employee">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Organization" />
        <owl:Class rdf:about="#Association" />
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="#Person" />
</owl:ObjectProperty>
```

Codeblock 2.22: Die Relation *has-employee* gibt an, welche *Person* bei der *Organization* oder *Association* angestellt sind

2.2.11 Relation: employee-of

Die Relation *employee-of* weisst einer *Person* einen Arbeitgeber zu. Eine *Person*, welche eine solche Zuweisung aufweist, sollte entsprechend auch Ã¼ber 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

```
<owl:ObjectProperty rdf:ID="#employee-of">
  <rdfs:domain rdf:resource="#Person" />
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Organization" />
        <owl:Class rdf:about="#Association" />
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
```

Codeblock 2.23: Die Relation *employee-of* gibt an, bei welcher *Organization* oder *Association* die Person angestellt ist

2.2.12 Relation: working-as

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

Eigenschaften

- Transitiv
- inverseOf *working-as*

Code

```
<owl:ObjectProperty rdf:ID="#working-as">
  <rdfs:domain rdf:resource="#Person" />
  <rdfs:range rdf:resource="#Profession" />
</owl:ObjectProperty>
```

Codeblock 2.24: Die Relation *working-as* gibt an, welchen Beruf eine Person ausübt

2.2.13 Relation: has-member

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

Eigenschaften

- Transitiv
- inverseOf *member-of*

Code

```
<owl:ObjectProperty rdf:ID="#has-member">
  <rdfs:domain rdf:resource="#Association" />
  <rdfs:range rdf:resource="#Person" />
</owl:ObjectProperty>
```

Codeblock 2.25: Die Relation *has-member* enthällt alle Mitglieder der *Association*

2.2.14 Relation: member-of

Die Relation *member-of* weisst 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

```
<owl:ObjectProperty rdf:ID="#member-of">
  <rdfs:domain rdf:resource="#Person" />
  <rdfs:range rdf:resource="#Association" />
</owl:ObjectProperty>
```

Codeblock 2.26: Die Relation *member-of* gibt alle *Association* an, bei welcher die Person Mitglied ist

2.2.15 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:    <rdfs:domain>
3:      <owl:Class>
4:        <owl:unionOf rdf:parseType="Collection">
5:          <owl:Class rdf:about="#Organization" />
6:          <owl:Class rdf:about="#Association" />
7:        </owl:unionOf>
8:      </owl:Class>
9:    </rdfs:domain>
10:   <rdfs:range rdf:resource="#Document" />
11: </owl:ObjectProperty>
```

Codeblock 2.27: Die Relation *has-document* verknüpft eine *Organization* mit einem *Document*

2.2.16 Relation: document-of

Die Relation *document-of* gibt an, welche *Organization* oder *Association* der "Eigentümer" eines Dokumentes ist. Es wird also definiert, wer ein Dokuemnt 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

```
<owl:ObjectProperty rdf:ID="#document-of">
  <rdfs:domain rdf:resource="#Document" />
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Organization" />
        <owl:Class rdf:about="#Association" />
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
```

Codeblock 2.28: Die Relation *document-of* gibt an, welche *Organization* oder *Association* das *Document* veröffentlicht hat

2.2.17 Relation: is-about

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

Eigenschaften

- Transitiv
- inverseOf *mentioned-in*

Code

```
<owl:ObjectProperty rdf:ID="#is-about">
  <rdfs:domain rdf:resource="#Document" />
  <rdfs:range rdf:resource="#Association" />
</owl:ObjectProperty>
```

Codeblock 2.29: Die Relation *is-about* gibt an, über welche *Association* ein Dokument ist

2.2.18 Relation: mentioned-in

Die Relation *mentioned-in* gibt an, in welchen *Document* Instanzen die *Association* erwähnt wird. Das Property enthällt 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:   <rdfs:domain rdf:resource="#Association" />
3:   <rdfs:range rdf:resource="#Document" />
4: </owl:ObjectProperty>
```

Codeblock 2.30: Die Relation *mentioned-in* gibt alle *Document* an, in welcher eine *Association* erwähnt wird

3 Implementation

3.1 Architektur

3.2 Server

3.3 Client: Datenerfassung

3.4 Client: Suche und Abfragen

Abbildungsverzeichnis

Tabellenverzeichnis

Codeblock-Verzeichnis

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, *SPARQL Query Language for RDF*, 15. Januar 2008, http://www.w3.org/TR/rdf-sparql-query/

[4] Dave Beckett, Redland, *Redland RDF Libraries*, 2012, http://librdf.org/

[5] Nokia, OpenSource, *Qt-Cross-Platform application and UI framework*, 2012, http://qt.nokia.com/