# **Ontology Creation**

* Relation Extraction is extracting relations between entities and arguments. After extraction all relations, these axioms can be represented into the ontology using different standardized languages XML, RDF or OWL.
* RDFs are less powerful than OWL but allows schema Relation like class hierarchies, domains and ranges for predicate to be expressed in RDF.
* RDF is holding a linked collection of (entity, attribute, value) which supports in term o0f storage and query applications.
* RDF specification permits an attribute (or property, or predicate=an edge joining 2 nodes in graph terms) to be a subject node as well.
* For instance, in order to specify domain and range of the attribute as rdf statements;

(Picasso, paints, “Guernica”)

Paints is a predicate or edge is complemented by two schema relations defining the predicate.

(Paints, domain, Painter)

(Paints, range, Painting)

Where Paint is a node.

* RDF is a directed, labeled graph data format for representing information in the Web. This specification defines the syntax and semantics of **the SPARQL query** language for RDF. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs.

# **Ontology Evaluation Catalogue**

## **Accuracy:**

A higher accuracy comes from correct definitions and descriptions of classes, properties, and individuals. There are different methods to evaluate accuracy of the ontology.

|  |  |
| --- | --- |
| **Methods** | **Description** |
| Hash vs Slash URI | * The basic question is on the difference between using http://example.org/ontology#joe and <http://example.org/ontology/joe> in order to refer to a non-information resource. * The former type of URI is called a hash URI (since the local part is separated by the hash character # from the namespace), the latter type a slash URI (since the local part is separated by the slash character / from the namespace). * When resolving a hash URI, only the namespace is resolved. All hash URIs with   the same namespace thus resolve to the same resource. This has the advantage that the ontology can be downloaded in one pass, but it also has the disadvantage that the file can become very big.  Therefore, terminological ontologies and ontologies with a closed, rarely changing, and rather **small set of individuals** (e.g. a list of all countries) would use **hash URIs**, whereas open domains with often **changing individuals** often **use slash URIs** |
| **Querying for anti-patterns:** To detect so called anti-patterns is at least as important as detecting patterns in  ontologies. Anti-patterns are strong indicators for problems in an ontology. | (Searching for Anti-Patterns)  SPARQL queries over the ontology graph can be used to discover potentially problematic patterns. For example results to the following queries have been found to be almost always problems.  Detecting the anti-pattern of subsuming nothing:  select ?a where {  ?a rdfs:subClassOf owl:Nothing .  }  Detecting the anti-pattern of skewed partitions:  select distinct ?A ?B1 ?B2 ?C1 where {  ?B1 rdfs:subClassOf ?A .  ?B2 rdfs:subClassOf ?A .  ?C1 rdfs:subClassOf ?B1 .  ?C1 owl:disjointWith ?B2 .  } |
| Class and relation ratio | Measure (M29) in (Gangemi et al., 2005) is called the "Class / relation ratio",  suggesting that it returns the ratio between classes and relations (or properties). The  exact de\_nition of the measure is: "nG∈S/nR∈S  where nG∈S is the cardinality of the set of  classes represent[ed] by nodes in g, and nR∈S is the cardinality of the set of relations  represented by arcs in g" (Gangemi et al., 2005). |
| Checking competency questions against results | * Formalize your competency question as a SPARQL query. Write down the expected answer as a SPARQL query result * Compare the actual and the expected results. Note that the order of results is often undefined. |
| Checking competency questions with constraints | Formalize your competency question for ontology O as a SPARQL CONSTRUCT  query that formulates the result in RDF as ontology R. Merge R with O and  a possibly empty ontology containing further constraints C. Check the merged ontology for inconsistencies. |
| Affirming derived knowledge |  |
| Expressive competency check | An ontology O can be accompanied by a highly axiomatized version of the ontology,  C. The merged ontology of O[C has to be consistent, otherwise the inconsistencies  point out to errors in O. |
| Inconsistency checking with rules | * Translate the ontology to be evaluated and possible constraint ontologies to a logic program. This translation does not have to be complete. Formalize further constraints as rules or integrity constraints. * Concatenate the translated ontologies and the further constraints or integrity constraints. Run the resulting program. If it raises any integrity constraints, then the evaluated ontology contains errors. |

## **Adaptability**

|  |  |
| --- | --- |
| **Methods** |  |
| URI declarations and punning |  |
| Blank Nodes |  |
| Querying for anti-patterns |  |
| Stability |  |
| Maximum depth of the taxonomy |  |
| A\_rming derived knowledge |  |
| Expressive consistency checks |  |
| Inconsistency checking with rules |  |

## **Clarity**

|  |  |
| --- | --- |
| **Methods** |  |
| Linked Data |  |
| Linked Data |  |
| Hash vs slash |  |
| Opaqueness of URIs |  |
| URI declarations  and punning |  |
| Typed literals and datatypes |  |
| Language tags |  |
| Labels and comments |  |
| Analysis and Examples |  |
| Class or Relation ratio |  |

## **Completeness**

|  |  |
| --- | --- |
| **Methods** |  |
| Hash vs Slash |  |
| URI declarations and punning |  |
| Typed literals and datatypes |  |
| Labels and comments |  |
| Blank nodes |  |
| XML validation |  |
| Structural metrics in practice |  |
| Stability |  |
| Language completeness |  |
| Maximum depth of the taxonomy |  |
| Formalized competency questions |  |

## **Computational efficiency**

|  |  |
| --- | --- |
| **Methods** |  |
| URI declarations and punning |  |
| Typed literals and datatypes |  |
| Structural metrics in  practice |  |
| Language completeness |  |

## **Conciseness**

|  |  |
| --- | --- |
| **Methods** |  |
| URI reuse |  |
| Blank nodes |  |
| Stability |  |
| Maximum depth of the taxonomy |  |
| Class / relation ratio |  |
| Formalized competency questions |  |

## **Consistency**

|  |  |
| --- | --- |
| **Methods** |  |
| Hash vs slash |  |
| Opaqueness of URIs |  |
| URI  reuse |  |
| Labels and comments |  |
| Structural  metrics in practice |  |
| Querying for anti-patterns |  |
| Analysis and Examples |  |
| Language completeness |  |
| Affrming derived knowledge |  |
| Expressive  consistency checks |  |
| Consistency checking with rules |  |

## **Organizational fitness**

|  |  |
| --- | --- |
| **Methods** |  |
| Linked data |  |
| Linked data |  |
| Hash vs Slash |  |
| Opaqueness of URIs |  |
| URI reuse |  |
| Lamguage tags |  |
| Labels and comments |  |
| XML validation |  |
| Formalized competency questions |  |