A Metadata-Based Generic Matching Framework for Web Ontologies

Malgorzata Mochol, Elena Paslaru Bontas
AG Netzbasierte Informationssysteme
mochol@inf.fu-berlin.de
paslaru@inf.fu-berlin.de

June 30, 2005

Technical Report B-05-08

Abstract

Current algorithms can not to be used optimally in automatic and semi-automatic ontology matching tasks as those envisioned by the Semantic Web community, mainly because of the inherent dependency between particular algorithms and ontology properties such as size, representation language or underlying graph structure, and because of performance and scalability limitations. In order to cope with the first problem we designed a generic matching framework which exploits the valuable ideas embedded in current matching approaches, but in the same time accounts for their limitations—for specific input ontologies it optimizes the matching results by automatically eliminating unsuitable candidate matching methods.

Contents

1 Introduction

Despite the young nature of the ontology engineering area ontologies have already found a wide acceptance in various research areas beyond the boundaries of the Semantic Web community. In the last decades an ever-growing number of ontologies have been developed and deployed in numerous computer science fields such as knowledge management, information retrieval, personalization, multimedia, software engineering or Web Services. In parallel to the dissemination of ontologies across these research communities a plethora of tools and methodologies to build, maintain and manage ontologies have emerged.

Due to its open design, a fully developed Semantic Web will contain numerous, distributed and ubiquitously available ontologies. Users will be able to choose among them to allow a mediated access to Web information, or to integrate or transform them to application-specific, customized models. Further on, the next generation of Web Services will apply ontologies to describe service capabilities and to mediate inter-process communication. A fundamental requirement for the realization of this vision are proved and tested ontology matching algorithms, which are able to deal with the heterogeneity of current ontological sources available on the Web.[1] The importance of ontology matching is emphasized by its implication in most of the phases of an ontology management process, be that ontology merging, mapping or evaluation. Though containing valuable ideas and techniques current matching approaches are tailored to certain *types of ontologies* and lack exhaustive testing in real world scenarios. Therefore they cannot *optimally* contribute to the realization of the envisioned Semantic Web—as described in the next section.

In this paper we propose a generic matching framework which copes with some of these limitations by being aware of the link between matching algorithms and the ontologies (or the types of ontologies) they have been originally designed for (or successfully applied to). Our approach allows a more flexible, automatically triggered usage of various matching algorithms, depending on their suitability to particular phases of the ontology management process. Due to its generic and automatic character the approach can be applied in a service-oriented context, in order to enable the discovery and operation of appropriate matching services required to deal with specific, (previously unknown) ontologies. The proposed framework uses semantical descriptions of both single matching algorithms and Web ontologies, which are then related by means of rules to optimize matching results.

The remaining of the paper is organized as follows: we give an overview of recent related work in Section 2, aligning our work to existing combined matching frameworks. Section 3 elaborates the basic idea of our approach, whose main components are described in Sections 4 to 6. We close with a discussion of the results and an outline of future work in Section 7.

[1] By heterogeneity we mean not only various representation languages, but also different maturity and granularity levels, different views upon the modeled domains etc.

2 Related Work

Matching conceptual structures, be that database schemes, XML schemes, conceptual graphs or more recently Semantic Web ontologies is a discipline with a long tradition, which plays a significant role in various areas of computer science such as data integration, data warehouses, agent communication and Web Service composition. The importance of the matching issue is reflected by the high number of matching algorithms, which have been proposed and applied in particular application settings in the last decades[1, 2, 3, 4, 5, 6, 7, 8]. Comprehensive studies, surveys and classifications on this topic are given for example in [9, 10, 11].

Usually one distinguishes between individual algorithms (e.g. FCA-MERGE[6] or S-Match[3])—applying only a single method of matching items e.g. linguistic or taxonomical matchers—and combinations of the former ones, which intend to overcome their limitations by proposing hybrid and composite solutions. A hybrid approach (e.g.Cupid[1]) follows a black box paradigm, in which various individual matchers are melt together to a new algorithm, while the so-called composite matchers allow an increased user interaction (e.g. GLUE[2],COMA[8], CMC[12]).

Despite of the relatively large number of promising approaches their limitations w.r.t. certain ontology characteristics have been often emphasized in recent literature[10, 3, 7, 1]:

- some approaches assume a common or, at least to large extent, overlapping universe of discourse[13],
- they can not be applied across various domains with the same effect (for example Cupid[3]),
- they require certain representation (or translation to the suitable format) or natural languages (e.g. the COMA approach[8]),
- they perform well on relatively small inputs with at most hundreds of concepts and have not been tested or do not scale for real world applications processing complex schemes,
- they do not perform well on inputs with heterogeneous (graph) structures (e.g. Cupid[3]) or are restricted to tree-based concept models (SimilarityFlooding (SF)[7], S-Match[3]),
- the results are based on a one-to-one mapping between taxonomies (such as in GLUE[14]),
- they need some manual pre-processing (like in GLUE, COMA[11])

In comparison to the mentioned solutions our approach does not intend to propose a new matching technique in any of the categories presented so far[9], but focuses on developing a novel *matching strategy*. It aims at applying *existing*

matching algorithms depending on the characteristics of the ontological inputs. We believe that joining different matching techniques in composite or hybrid algorithms offers solely partial solutions to this heterogeneity issue. Within these combined frameworks, the quality of the matching results is *implicitly* dependent on the features of the input ontologies, with the result that these framework can not be optimally used in an open, distributed environment such as the Web. A Web-enabled matching framework requires a maximally flexible selection and composition of available matching services in order to be able to cope with the whole range of ontologies across the network.

The best example in this category is the COMA framework[8]. COMA supports different applications and scheme types (like XML and relational schemes) and provides an extensible library of matching algorithms, a component for combining the results obtained, and extensive functionality for the evaluation of matching effectiveness. One of the weakness points in COMA comes however from the fact that the suggested combined methods may prove to be inadequate for complex situations. Since each base matcher performs differently in different conditions, simple, *pre-defined* composition methods are incapable of capturing such performance variation[12]. In such cases users are required to manually customize the matching workflow, which means that the framework can not be directly employed in open, service-oriented environments which should work without human intervention. Our approach is different in this respect, since it describes explicitly the properties of each of the matching services available in a repository and relates this information by means of rules with the "ideal" input ontology characteristics.

3 A Generic Matching Framework

The framework consists of four components, as depicted in Figure 1:

- the matching repository which contains reusable matching components and metadata describing their properties (see Section 4)
- the ontology repository which manages the matching inputs described by ontology metadata (see Section 5)
- the rule repository linking ontology and matching properties (see Section 6 for a description of the generic rules employed)
- the matching engine which is responsible for the decision making process on which algorithms are applicable for a specific set of inputs and for the execution of the workflow.

Metadata, which is stored declaratively in the repositories, offers an ontological description of some of the most significant properties of the corresponding items (i.e. matching algorithms and their inputs). This formal description allows the matching engine to automatically compare the metadata of the inputs with the constraints of the available algorithms and, by means of generic rules, exclude

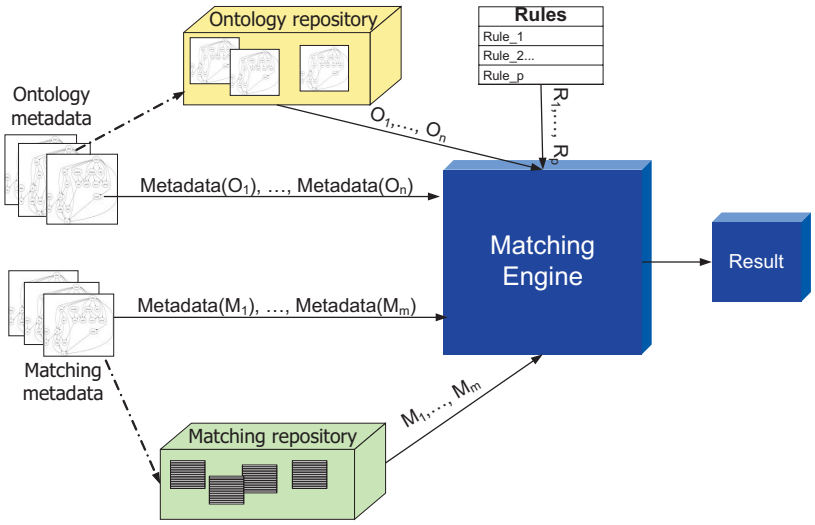

Figure 1: High-Level Architecture of the Matching Framework

unappropriate candidate algorithms, which can not deal with certain properties of the ontologies to be matched, or do not correspond to specific quality of service parameters (e.g. performance or accuracy of the matching results).

The metadata model used to describe matchers is derived from empirical case studies[15, 16] and from surveying recent relevant research literature, which offer various classification schemes for existing approaches and their combinations[9, 10, 11]. Further on, ontologies are described using the information model described in[17], which is a contextual model for Semantic Web Resources incorporating intrinsic and extrinsic ontology properties, some of which are recognized to influence the quality of the matchers. In the following we concentrate on the description of the two metadata models.

4 Matching Metadata

Matching metadata describe single ontology matchers of a matching repository. The model used for classifying the matching algorithms strongly relies on [9] (see Fig. 2).

This classification distinguishes between individual matchers which compute a mapping based on a single matching criterion and combining matchers which use multiple individual matchers[18]:

Individual matchers can work on instance data (*instance/contents-based matchers*) or consider only structure information, be that relationship types, data types and schema structures (*schema-only based matchers*). Both algorithms can be applied on individual schema elements such as attributes

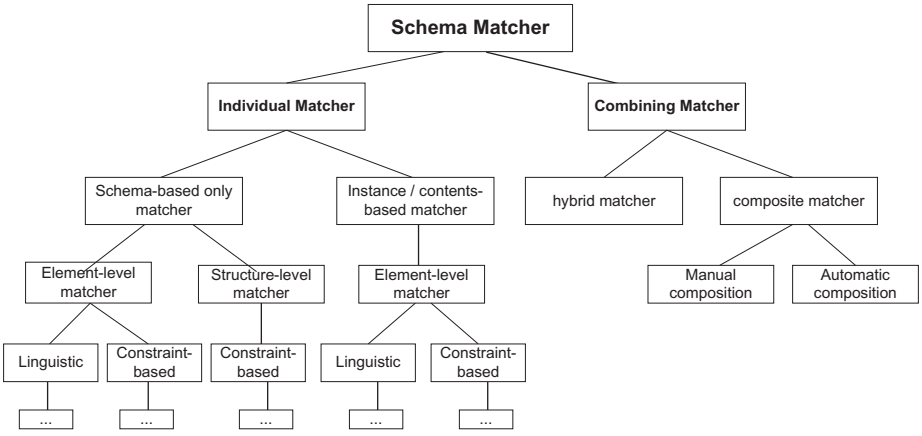

Figure 2: Classification of schema matching approaches[9]

or concept labels (*element-level matchers*). In addition, schema-only based approaches can deal with combinations of these schema elements such as complex schema structures, thus computing mappings by analyzing subgraphs (*structure-level matcher*). A single element-level matcher uses linguistic, as well as constraint-based techniques, while a schema-only based matcher is considered to use only the latter(see Fig. 3).

Combining matchers are divided into two categories. Composite matchers combine the different results of independently executed matchers whereupon the order of the execution of the individual matchers can be assign manually (*manual composition*) or (semi-)automatically (*automatic composition*). In contrast, a *hybrid matcher* does not allow such manual intervention (see Fig. 4).

Beside the aforementioned classification, the metadata model distinguishes among different *matching results* (mappings, value) and includes the following *matching characteristics*:

- input type: instances or schemas, eventually numerical values;
- cardinality which specifies whether a matcher compares one or more elements of one schema with one or more elements of another schema; we differentiate between global cardinality (w.r.t different mapping elements) and local cardinality (w.r.t individual mapping elements);
- matching level: *atomic level*, e.g. attributes in an XML schema and *higher (non-atomic) level* e.g. XML elements,
- completeness: a *full match* considers all elements of the two schemes, in contrast to a *partially match*.

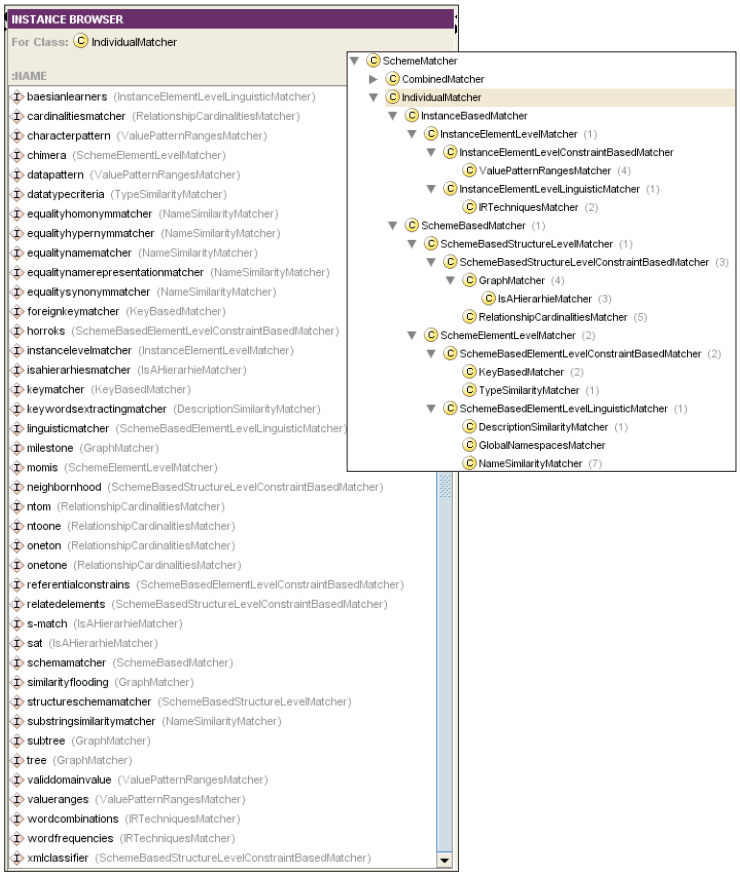

Figure 3: Classifications of individual matchers with some instances (single matcher approaches)

Further on properties of the ontologies to be matched such as type formality level, domain type, representation languages, supported natural language, supported used primitives etc. are defined in the ontology metadata model (see Section 5) and are referenced in the matching metadata model to refine the description of the matching inputs.

The matcher classification supplemented by the aforementioned features was conceptualized in form of an ontology and implemented in OWL. Matcher types are defined as OWL classes within a hierarchical structure with "sub-class" relationships between them. By means of OWL constraints we specified the characteristics of each type of matching algorithm (for example that the input of an instance-based matcher can not be a scheme without instance data). This ontology together with the context ontology—described in the next section— and the rules build the knowledge background of the matching engine, contributing to the automatic exclusion of unsuitable matchers.

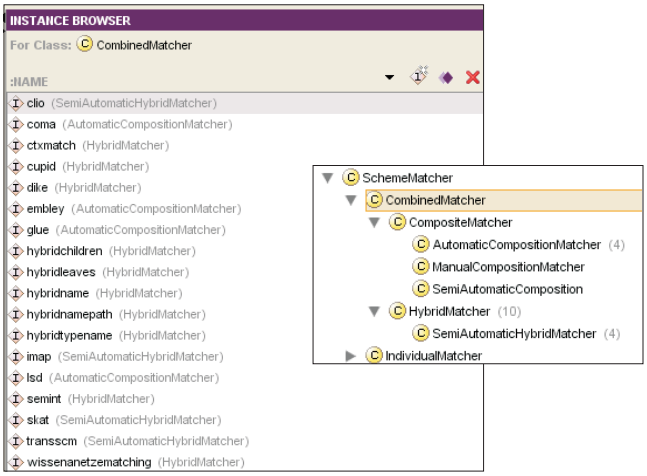

Figure 4: Classification of combined matchers with some instances (combined matcher approaches)

5 Ontology Metadata

The information model used to describe matching inputs i.e. ontologies contains—according to Stamper's semiotic framework[19]—three categories of features: syntactic, semantic and pragmatic:[2]

Syntactic features offer quantitative and qualitative information about the ontology and its underlying (graph) topology. Examples of syntactical features are the number of concepts and properties for each class, the depth of an inheritance tree, the number of incoming properties, the number of concept instances, the average path length, the number of connected components. Since ontologies are published in an open network like the Semantic Web, it is also important to consider the links a particular ontology has to other networked information sources[20]. Finally, there is qualitative, representation language-dependent information like the representation language itself, the number of syntax constructs used and syntactic correctness (validity).

Semantic features are related to the formal semantics of the representation language and the meaning of the ontology content:

- consistency (as measured by a reasoner),
- correctness (i.e.whether the asserted information is true),

[2]The rationales behind and the method applied to generate this information model are beyond the scope of this paper and are described for example in [17].

- readability (i.e. the non-ambiguous interpretation of the meaning of the concept names w.r.t. a lexicon, the usage of human-readable concept names),
- level of formality (e.g. highly informal, semi-informal, semi-formal, rigorously formal[21]),
- type of model (upper-level, domain ontology, thesaurus etc.[22, 23]),
- ontology domain (e.g. medicine),
- representation paradigm (i.e. the class of representation languages w.r.t. its expressivity such as a specific Description Logic),
- natural language (e.g. English).

Heuristic and pragmatic features refer to authoring and historical data about an ontology, for example when, by whom and to which purpose it was developed, whether multiple versions are available, or about the engineering process the ontology originally resulted from.

The features mentioned above are part of the so-called "ontology context"[17], an information model which is used to describe ontologies in various phases of their life cycle. The model itself is formalized by means of OWL ontologies (see Fig. 5). [3]

Syntactic features are conceptualized as DatatypeProperties with an integer or a string range for the numerical and the qualitative information respectively. Semantic features are modeled as OWL classes or individuals. For interoperability purposes every instance of OntologyDomain references a topic in the Open Directory taxonomy[4], which was translated to OWL for this purpose. The class DomainType is used to define the generality levels of a conceptualization as in [22, 23]. By means of the class RepresentationParadigm and its individuals (e.g. a Description Logic like OWL DL) one can define which ontological primitives are supported in the representation language of the matching input (e.g. supports existential constraints) and which are actually used in the respective model (e.g. the ontology is written in OWL DL, but it uses solely classes and sub-class relationships). General-purpose FormalityLevels are defined as in [21, 23]. The properties of different kinds of ontologies (such as thesauri, taxonomies or Semantic Web ontologies) can be declared by means of OWL constraints. For example a taxonomy is formalized using a representation paradigm supporting concepts and is-a relationships and has usually the FormalityLevel "semi-formal".

The pragmatic category contains classes such as OntologyTask, OntologyRole, OntologyApplication, IndustrialSector, OntologyAuthor, EngineeringMethodology. For the categorization of ontology applications we use a modified version of the ACM classification,[5] while the industrial sectors relate to the

[3] http://nbi.inf.fu-berlin.de/research/swpatho/context/context.owl
[4] http://www.dmoz.org
[5] http://www.acm.org/class/1998/

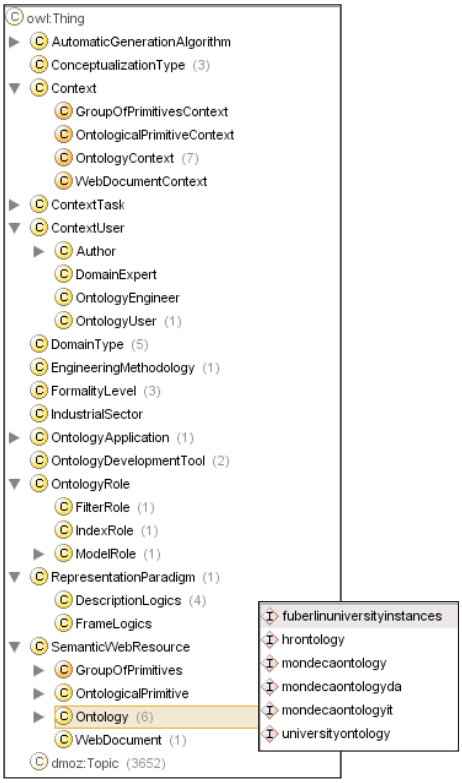

Figure 5: Metadata for ontologies

NAICS taxonomy.[6] We analyzed the state of the art of ontology-based information systems according to recent surveys on this topic[24, 25] in order to establish the range of purposes ontologies are currently used for and the roles ontologies play within these tasks (specifying subclasses of OntologyTask and OntologyRole, respectively).

Matching algorithms can not be applied with the same success expectations independently of all the dimensions of the mentioned ontology information model. In particular, we identified the following ontology features as being relevant for matching tasks:

- Quantitative syntactic features such as number of specific ontological primitives influence the matching execution performance and the quality of the structured-based matchers, which usually perform better on simple graph structures.

- Semantic features such as domain, natural language, type of domain and representation language restrict the number of applicable matching algo-

[6] http://www.census.gov/epcd/www/naics.html; North American Industry Classification System

rithms, which are sometimes domain-specific or accept solely certain types of schemes.

These dependencies are formalized in terms of rules in the next section.

6 Generic Rules

For a given pair of ontologies to be matched, the matching engine (Figure 1) has to decide which matching algorithms should be applied to obtain the desired outputs. The engine is aware of background information describing the available matching services and the properties of the input ontologies. However, in order to automatically infer which algorithms suit to concrete inputs, it needs explicit knowledge about the dependencies between these algorithms and the structures they operate on. We formalize this knowledge in terms of *generic dependency rules*—statements that determine which elements (in this case which matchers) are to be used or excluded (see Fig. 6).

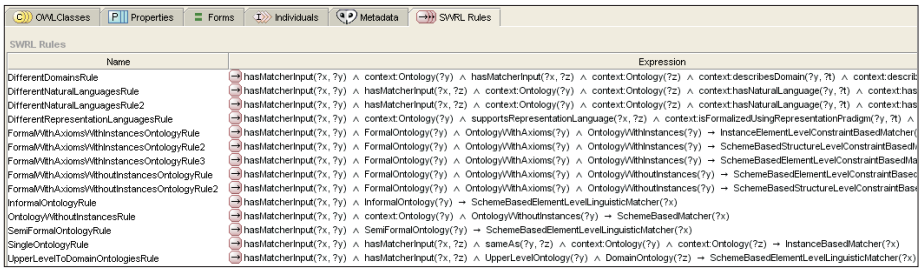

Figure 6: Some rules defined with SRWL (extract from Protege)

Generic matching rules are, for example:

- Apply only instance matchers for a single ontology
- Apply only matchers which are able to deal with the representation language of the inputs
- Use only linguistic matchers for informal and semi-formal ontologies
- Use structure-based matchers for ontologies with different natural languages
- Use constraints-based matchers only for formal ontologies and only if ontologies contain axioms
- Match upper-level to domain ontologies using linguistic matchings
- Match only ontologies in similar domains
- Apply only scheme matchers if no instance data is available

- Do not apply linguistic matchers for ontologies with incompatible concept names

The rules are the result of analyzing recent publications in this research discipline and were confirmed empirically within the projects "KnowledgeNets" and "A Semantic Web for Pathology" [15, 16, 26], which required ontology matching techniques to merge and integrate existing ontologies to the corresponding target ontologies used to build different Semantic Web applications. Further rules, especially relating the syntactic features of ontologies with specific performance and accuracy parameters, are subject of current work.

For implementation purposes the matching rules were implemented in SWRL[27], a rule language for the Semantic Web, which allows us to formalize them in terms of the concepts defined in the two metadata models[18]. However, the usage of the rules in decision making processes requires a reasoning engine which is able to operate on OWL ontologies and SWRL rules—an issue which is still subject of active research in the Semantic Web community. Details about this subject and about the rules implementation are available as technical report[18].

7 Conclusion and Future Work

In this paper we have presented the high-level architecture of a metadata-based generic matching framework for Web ontologies. We have described the main components of the framework: the matching metadata describing the properties of matchings, the ontology repository described by metadata, the rule repository linking ontology and matching properties and the matching engine which uses a reasoner to choose the appropriate matcher for given ontologies. As a next step we are extending the rule base with information concerning the syntactic dimension of ontology inputs. Another important topic which we do not address in the paper is the matching execution, which requires an automatic composition of the candidate matching services to achieve the desired results. We plan to investigate available approaches in the area of Web Service composition in order to apply them to this issue.

Acknowledgements This work is a result of the cooperation within the Semantic Web PhD-Network Berlin-Brandenburg and has been partially supported by the projects KnowledgeWeb - Network of Excellence, "A Semantic Web for Pathology" funded by the DFG (German Research Foundation) and "Knowledge Nets", which is part of the InterVal- Berlin Research Centre for the Internet Economy, funded by the German Ministry of Research BMBF.

8 Appendix

The appendix includes three ontology examples described using the ontology metadata, three matching examples from the matching metadata ontology, and

three rules with reasoning results.[7]

8.1 Ontology description - Examples

1. **Danish University Ontology**

```
<Ontology rdf:ID="mondecaontologyda">
 <hasNaturalLanguage rdf:datatype="...#string">da</hasNaturalLanguage>
 <usesSource rdf:resource="#mondecaontology"/>
 <hasStatus rdf:datatype="...#string">stable</hasStatus>
 <hasCreationDate rdf:datatype="...#date">2004-07-06</hasCreationDate>
 <hasURI rdf:datatype="...#anyURI">http://www.mondeca.com/owl/moses/dan.owl</hasURI>
 <imports rdf:resource="#mondecaontology"/>
 <isCompatibleWith rdf:resource="#mondecaontologyit"/>
 <hasDomainType rdf:resource="#domain"/>
 <supportsOntologicalPrimitive rdf:resource="#concept"/>
 <isFormalizedUsingRepresentationPradigm rdf:resource="#owldl"/>
 <supportsOntologicalPrimitive rdf:resource="#subclassof"/>
 <isCreatedUsingTool>
  <OntologyDevelopmentTool rdf:ID="protege"/>
 </isCreatedUsingTool>
 <isCompatibleWith rdf:resource="#mondecaontology"/>
 <hasConceptualizationMethod rdf:resource="#manualconceptualization"/>
 <rdfs:comment rdf:datatype= "...#string">Danish University Ontology</rdfs:comment>
 <usedFor rdf:resource="#integration"/>
  <describesDomain rdf:resource="...context/dmoz_instances.owl#University"/>
  <hasFormalityLevel rdf:resource="#formal"/>
  <createdBy>
   <OntologyAuthor rdf:ID="mondecaontologydaauthor">
    <hasName rdf:datatype="...#string">Lina Henriksen</hasName>
    <creates rdf:resource="#mondecaontologyda"/>
    <belongsToContext>
     <OntologyContext rdf:ID="mondecaontologydacontext1">
      <hasTarget rdf:resource="#mondecaontologyda"/>
      <hasTask rdf:resource="#integration"/>
     </OntologyContext>
    </belongsToContext>
   </OntologyAuthor>
  </createdBy>
 <belongsToContext rdf:resource="#mondecaontologydacontext1"/>
</Ontology>
```

[7] For a better readability we replaced: http://www.w3.org/2001/XMLSchema by "...", http://nbi.inf.fu-berlin.de/research/swpatho/context/ by "...context/", http://projects.mi.fu-berlin.de/semweb/ns/ by "...ns/", and http://www.w3.org/1999/02/22-rdf-syntax-ns by "...rdf-syntax-ns".

2. **Italian University Ontology**

```
<Ontology rdf:ID="mondecaontologyit">
 <hasDomainType rdf:resource="#domain"/>
 <isFormalizedUsingRepresentationPradigm rdf:resource="#owldl"/>
 <supportsOntologicalPrimitive>
  <UniversalRestriction rdf:ID="owlallvaluesfrom">
   <isSupportedBy rdf:resource="#owldl"/>
   <isSupportedBy rdf:resource="#mondecaontologyit"/>
  </UniversalRestriction>
 </supportsOntologicalPrimitive>
 <createdBy>
  <OntologyAuthor rdf:ID="authormondecaontologyit">
   <belongsToContext>
    <OntologyContext rdf:ID="mondecaontologyitcontext1">
     <hasTarget rdf:resource="#mondecaontologyit"/>
     <hasTask rdf:resource="#integration"/>
    </OntologyContext>
   </belongsToContext>
   <hasName rdf:datatype="...#string">Bernard Vatant</hasName>
   <creates rdf:resource="#mondecaontologyit"/>
   (...)
  </OntologyAuthor>
 </createdBy>
 <describesDomain rdf:resource="...context/dmoz_instances.owl#University"/>
 <hasFormalityLevel rdf:resource="#formal"/>
 <belongsToContext rdf:resource="#mondecaontologyitcontext1"/>
 <hasNaturalLanguage rdf:datatype="...#string">it</hasNaturalLanguage>
 <supportsOntologicalPrimitive rdf:resource="#concept"/>
 <hasNaturalLanguage rdf:datatype="...#string">en</hasNaturalLanguage>
 <usedFor rdf:resource="#integration"/>
 <supportsOntologicalPrimitive rdf:resource="#subclassof"/>
 <hasURI rdf:datatype="...#anyURI">http://www.mondeca.com/owl/moses/ita.owl</hasURI>
</Ontology>
```

3. **Computer Science Faculty at the Free University Berlin**

```
<Ontology rdf:ID="fuberlinuniversityinstances">
 <belongsToContext rdf:resource="#fuberlinuniversitycontext1"/>
 <usedInApplication>
  <ContentManagementSystem rdf:ID="fuberlin"/>
 </usedInApplication>
 <imports>
  <SemanticWebOntology rdf:ID="fuberlinontologyschema">
   <hasStatus rdf:datatype="...#string">final</hasStatus>
   <describesDomain rdf:resource="...context/dmoz_instances.owl#University"/>
```

```
  <hasURI rdf:datatype="...#anyURI">...ns/schema</hasURI>
 </SemanticWebOntology>
</imports>
<usedFor rdf:resource="#integration"/>
<hasFormalityLevel rdf:resource="#semiformal"/>
<supportsOntologicalPrimitive>
 <Concept rdf:ID="concept">
  <isSupportedBy>
   (...)
  </isSupportedBy>
  <isSupportedBy rdf:resource="#mondecaontologyda"/>
  <isSupportedBy rdf:resource="#mondecaontologyit"/>
 </Concept>
</supportsOntologicalPrimitive>
<isCreatedUsingTool>
 <OntologyDevelopmentTool rdf:ID="oiled"/>
</isCreatedUsingTool>
<isFormalizedUsingRepresentationPradigm rdf:resource="#daml"/>
<hasConceptualizationMethod rdf:resource="#manualconceptualization"/>
<createdBy rdf:resource="#fuberlinontologyauthor"/>
<hasCreationDate rdf:datatype="...#date">2003-07-17</hasCreationDate>
<hasNaturalLanguage rdf:datatype="...#string">de</hasNaturalLanguage>
<imports>
 (...)
</imports>
<describesDomain rdf:resource="...context/dmoz_instances.owl#University"/>
<createdBy rdf:resource="#fuberlinontologyauthor2"/>
<hasURI rdf:datatype="...#anyURI">...ns/inf/institut</hasURI>
<supportsOntologicalPrimitive rdf:resource="#subclassof"/>
<belongsToContext rdf:resource="#fuberlinuniversitycontext2"/>
</Ontology>
```

8.2 Matching description - Examples

1. **Description of the LSD (Composite matcher)**

```
<AutomaticCompositionMatcher rdf:ID="lsd">
 <usesMatcher>
  <NameSimilarityMatcher rdf:ID="equalitysynonymmatcher">
   <hasMatcherInput rdf:resource="#InputScheme"/>
  </NameSimilarityMatcher>
 </usesMatcher>
 <hasMatcherInput rdf:resource="#InputScheme"/>
 <supportsCardinality>
  <GlobalCardinality rdf:ID="global1to1"/>
 </supportsCardinality>
```

```
 <usesAuxiliaryInformation>
  <AuxiliaryInformation rdf:ID="listofvaliddomainvalues"/>
 </usesAuxiliaryInformation>
 <usesMatcher>
  <InstanceElementLevelLinguisticMatcher rdf:ID="baesianlearners">
   <hasMatcherInput rdf:resource="#InputInstances"/>
  </InstanceElementLevelLinguisticMatcher>
 </usesMatcher>
 <supportsRepresentationLanguage>
  <context:RepresentationParadigm rdf:ID="xml"/>
 </supportsRepresentationLanguage>
 <supportsCardinality>
  <LocalCardinality rdf:ID="local1to1"/>
 </supportsCardinality>
 <usesMatcher>
  <ValuePatternRangesMatcher rdf:ID="validdomainvalue">
   <hasMatcherInput rdf:resource="#InputInstances"/>
  </ValuePatternRangesMatcher>
 </usesMatcher>
 <owl:sameAs>
  <AutomaticCompositionMatcher rdf:ID="glue">
   <owl:sameAs rdf:resource="#lsd"/>
   <hasMatcherInput rdf:resource="#InputInstances"/>
   <hasMatcherInput rdf:resource="#InputScheme"/>
  </AutomaticCompositionMatcher>
 </owl:sameAs>
 <hasMatcherInput rdf:resource="#InputInstances"/>
 <usedForContextTask>
  <context:IntegrationTask rdf:ID="dataintegration"/>
 </usedForContextTask>
 <usesAuxiliaryInformation>
  <AuxiliaryInformation rdf:ID="trainingmatches"/>
 </usesAuxiliaryInformation>
 <usesMatcher>
  <NameSimilarityMatcher rdf:ID="equalitynamematcher">
   <hasMatcherInput rdf:resource="#InputScheme"/>
  </NameSimilarityMatcher>
 </usesMatcher>
 <usesMatcher>
  <SchemeBasedStructureLevelConstraintBasedMatcher rdf:ID="xmlclassifier">
   <supportsRepresentationLanguage rdf:resource="#xml"/>
   <hasMatcherInput rdf:resource="#InputScheme"/>
  </SchemeBasedStructureLevelConstraintBasedMatcher>
 </usesMatcher>
</AutomaticCompositionMatcher>
```

2. **Description of the Cupid (Hybrid matcher)**

```
<HybridMatcher rdf:ID="cupid">
 <usesAuxiliaryInformation rdf:resource="#useracronyms"/>
 <usedForContextTask>
  <TranslationTask rdf:ID="translation"/>
 </usedForContextTask>
 <usesMatcher rdf:resource="#abbreviationmatcher"/>
 <usesMatcher>
  <SchemeBasedElementLevelConstraintBasedMatcher rdf:ID="referentialconstrains">
   <hasMatcherInput rdf:resource="#InputScheme"/>
  </SchemeBasedElementLevelConstraintBasedMatcher>
 </usesMatcher>
 <usesMatcher>
  <NameSimilarityMatcher rdf:ID="substringsimilaritymatcher">
   <hasMatcherInput rdf:resource="#InputScheme"/>
  </NameSimilarityMatcher>
 </usesMatcher>
 <usesMatcher rdf:resource="#equalityhypernymmatcher"/>
 <usesMatcher rdf:resource="#equalityhomonymmatcher"/>
 <usesAuxiliaryInformation>
  <UserInputInformation rdf:ID="usersynonyms"/>
 </usesAuxiliaryInformation>
 <usesMatcher rdf:resource="#equalitysynonymmatcher"/>
 <usesMatcher rdf:resource="#equalitynamematcher"/>
 <hasMatcherInput rdf:resource="#InputScheme"/>
 <usesMatcher>
  <GraphMatcher rdf:ID="subtree">
   <hasMatcherInput rdf:resource="#InputScheme"/>
  </GraphMatcher>
 </usesMatcher>
 <supportsRepresentationLanguage>
  <context:RepresentationParadigm rdf:ID="relationaldb"/>
 </supportsRepresentationLanguage>
 <supportsCardinality rdf:resource="#local1to1"/>
 <usesAuxiliaryInformation>
  <AuxiliaryInformation rdf:ID="thesauri"/>
 </usesAuxiliaryInformation>
 <supportsRepresentationLanguage rdf:resource="#xml"/>
 <usesMatcher>
  <TypeSimilarityMatcher rdf:ID="datatypecriteria">
   <hasMatcherInput rdf:resource="#InputScheme"/>
  </TypeSimilarityMatcher>
 </usesMatcher>
 <usesAuxiliaryInformation>
  <UserInputInformation rdf:ID="userabbreviations"/>
```

```
 </usesAuxiliaryInformation>
 <usesMatcher>
  <SchemeBasedElementLevelLinguisticMatcher rdf:ID="linguisticmatcher">
   <hasMatcherInput rdf:resource="#InputScheme"/>
  </SchemeBasedElementLevelLinguisticMatcher>
 </usesMatcher>
 <supportsCardinality rdf:resource="#global1to1"/>
 <usesAuxiliaryInformation>
  <AuxiliaryInformation rdf:ID="glossaries"/>
 </usesAuxiliaryInformation>
</HybridMatcher>
```

3. **Description of the Similarity Flooding (Graph matcher)**

```
<GraphMatcher rdf:ID="similarityflooding">
 <supportsRepresentationLanguage rdf:resource="...context/context.owl#rdfs"/>
 <rdfs:comment rdf:datatype="...#string">SimilarityFlooding (SF)</rdfs:comment>
 <supportsRepresentationLanguage rdf:resource="#xml"/>
 <supportsRepresentationLanguage>
  <context:RepresentationParadigm rdf:ID="sqlddl"/>
 </supportsRepresentationLanguage>
 <supportsCardinality rdf:resource="#local1to1"/>
 <hasMatcherInput rdf:resource="#InputScheme"/>
</GraphMatcher>
```

8.3 Rules - Examples

1. **Use only linguistic matchers for semi-formal ontologies**

```
<swrl:Imp rdf:ID="SemiFormalOntologyRule">
 <swrl:body>
  <swrl:AtomList>
   <rdf:rest>
    <swrl:AtomList>
     <rdf:first>
      <swrl:DatavaluedPropertyAtom>
       <swrl:argument1 rdf:resource="#y"/>
       <swrl:argument2 rdf:resource="...context/context.owl#semiformal"/>
       <swrl:propertyPredicate
       rdf:resource="...context/context.owl#hasFormalityLevel"/>
      </swrl:DatavaluedPropertyAtom>
     </rdf:first>
     <rdf:rest>
      <swrl:AtomList>
       <rdf:first>
        <swrl:ClassAtom>
```

```
          <swrl:argument1 rdf:resource="#y"/>
          <swrl:classPredicate rdf:resource="#w"/>
         </swrl:ClassAtom>
        </rdf:first>
        <rdf:rest rdf:resource="...rdf-syntax-ns#nil"/>
       </swrl:AtomList>
      </rdf:rest>
     </swrl:AtomList>
    </rdf:rest>
    <rdf:first>
     <swrl:IndividualPropertyAtom>
      <swrl:argument1 rdf:resource="#x"/>
      <swrl:propertyPredicate rdf:resource="#hasMatcherInput"/>
      <swrl:argument2 rdf:resource="#w"/>
     </swrl:IndividualPropertyAtom>
    </rdf:first>
   </swrl:AtomList>
  </swrl:body>
  <swrl:head>
   <swrl:AtomList>
    <rdf:first>
     <swrl:ClassAtom>
      <swrl:argument1 rdf:resource="#x"/>
      <swrl:classPredicate rdf:resource="#SchemeBasedElementLevelLinguisticMatcher"/>
     </swrl:ClassAtom>
    </rdf:first>
    <rdf:rest rdf:resource="...rdf-syntax-ns#nil"/>
   </swrl:AtomList>
  </swrl:head>
 </swrl:Imp>
```

Query results:

x
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#equalityhomonymmatcher
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#equalitysynonymmatcher
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#equalitynamematcher
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#abbreviationmatcher
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#equalityhypernymmatcher
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#keywordsextractingmatcher
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#substringsimilaritymatcher
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#linguisticmatcher
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#equalitynamerepresentationmatcher

9 results found in 6 ms.

Figure 7: Results of rule: “Use only linguistic matchers for semi-formal ontologies”

2. **Use structure-based matchers for ontologies with different natural languages**

```
<swrl:Imp rdf:ID="DifferentNaturalLanguagesRule">
 <swrl:body>
  <swrl:AtomList>
   <rdf:first>
    <swrl:IndividualPropertyAtom>
     <swrl:propertyPredicate rdf:resource="#hasMatcherInput"/>
     <swrl:argument1 rdf:resource="#x"/>
     <swrl:argument2 rdf:resource="#p"/>
    </swrl:IndividualPropertyAtom>
   </rdf:first>
   <rdf:rest>
    <swrl:AtomList>
     <rdf:rest>
      <swrl:AtomList>
       <rdf:first>
        <swrl:ClassAtom>
         <swrl:classPredicate rdf:resource="#p"/>
         <swrl:argument1 rdf:resource="#y"/>
        </swrl:ClassAtom>
       </rdf:first>
       <rdf:rest>
        <swrl:AtomList>
         <rdf:first>
          <swrl:ClassAtom>
           <swrl:classPredicate rdf:resource="#q"/>
           <swrl:argument1 rdf:resource="#z"/>
          </swrl:ClassAtom>
         </rdf:first>
         <rdf:rest>
          <swrl:AtomList>
           <rdf:rest>
            <swrl:AtomList>
             <rdf:rest>
              <swrl:AtomList>
               <rdf:first>
                <swrl:DifferentIndividualsAtom>
                 <swrl:argument1 rdf:resource="#u"/>
                 <swrl:argument2 rdf:resource="#t"/>
                </swrl:DifferentIndividualsAtom>
               </rdf:first>
               <rdf:rest rdf:resource="...rdf-syntax-ns#nil"/>
              </swrl:AtomList>
             </rdf:rest>
```

```
            <rdf:first>
             <swrl:DatavaluedPropertyAtom>
              <swrl:argument1 rdf:resource="#z"/>
              <swrl:argument2 rdf:resource="#t"/>
              <swrl:propertyPredicate
              rdf:resource="...context/context.owl#hasNaturalLanguage"/>
             </swrl:DatavaluedPropertyAtom>
            </rdf:first>
           </swrl:AtomList>
          </rdf:rest>
          <rdf:first>
           <swrl:DatavaluedPropertyAtom>
            <swrl:argument1 rdf:resource="#y"/>
            <swrl:argument2 rdf:resource="#u"/>
            <swrl:propertyPredicate
            rdf:resource="...context/context.owl#hasNaturalLanguage"/>
           </swrl:DatavaluedPropertyAtom>
          </rdf:first>
         </swrl:AtomList>
        </rdf:rest>
       </swrl:AtomList>
      </rdf:rest>
     </swrl:AtomList>
    </rdf:rest>
    <rdf:first>
     <swrl:IndividualPropertyAtom>
      <swrl:argument2 rdf:resource="#q"/>
      <swrl:argument1 rdf:resource="#x"/>
      <swrl:propertyPredicate rdf:resource="#hasMatcherInput"/>
     </swrl:IndividualPropertyAtom>
    </rdf:first>
   </swrl:AtomList>
  </rdf:rest>
 </swrl:AtomList>
</swrl:body>
<swrl:head>
 <swrl:AtomList>
  <rdf:rest rdf:resource="...rdf-syntax-ns#nil"/>
  <rdf:first>
   <swrl:ClassAtom>
    <swrl:classPredicate rdf:resource="#SchemeBasedStructureLevelMatcher"/>
    <swrl:argument1 rdf:resource="#x"/>
   </swrl:ClassAtom>
  </rdf:first>
 </swrl:AtomList>
</swrl:head>
```

```
</swrl:Imp>
```

Query results:

x
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#xmlclassifier
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#cardinalitiesmatcher
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#sahierarhiesmatcher
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#tree
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#relatedelements
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#subtree
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#sat
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#ntom
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#structureschemamatcher
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#milestone
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#ntoone
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#similarityflooding
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#s-match
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#neighbornhood
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#onetone
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#oneton

16 results found in 5 ms.

Figure 8: Results of rule:"Use only linguistic matchers for semi-formal ontologies"

3. **Use constraints-based matchers only for formal ontologies without instances and only if ontologies contain axioms**

```
<swrl:Imp rdf:ID="FormalWithAxiomsWithoutInstancesOntologyRule">
 <swrl:head>
  <swrl:AtomList>
   <rdf:first>
    <swrl:ClassAtom>
     <swrl:argument1 rdf:resource="#x"/>
      <swrl:classPredicate
      rdf:resource="#SchemeBasedElementLevelConstraintBasedMatcher"/>
     </swrl:ClassAtom>
   </rdf:first>
   <rdf:rest rdf:resource="...rdf-syntax-ns#nil"/>
  </swrl:AtomList>
 </swrl:head>
 <swrl:body>
  <swrl:AtomList>
   <rdf:first>
    <swrl:IndividualPropertyAtom>
```

```
  <swrl:propertyPredicate rdf:resource="#hasMatcherInput"/>
  <swrl:argument2 rdf:resource="#w"/>
  <swrl:argument1 rdf:resource="#x"/>
 </swrl:IndividualPropertyAtom>
</rdf:first>
<rdf:rest>
 <swrl:AtomList>
  <rdf:first>
   <swrl:ClassAtom>
    <swrl:classPredicate rdf:resource="#w"/>
    <swrl:argument1 rdf:resource="#y"/>
    </swrl:ClassAtom>
   </rdf:first>
   <rdf:rest>
    <swrl:AtomList>
     <rdf:first>
      <swrl:IndividualPropertyAtom>
       <swrl:argument2 rdf:resource="#r"/>
       <swrl:propertyPredicate
       rdf:resource="...context/context.owl#supportsOntologicalPrimitive"/>
       <swrl:argument1 rdf:resource="#y"/>
      </swrl:IndividualPropertyAtom>
     </rdf:first>
     <rdf:rest>
      <swrl:AtomList>
       <rdf:first>
        <swrl:ClassAtom>
         <swrl:classPredicate rdf:resource="...context/context.owl#Restriction"/>
         <swrl:argument1 rdf:resource="#r"/>
        </swrl:ClassAtom>
       </rdf:first>
       <rdf:rest>
        <swrl:AtomList>
         <rdf:first>
          <swrl:ClassAtom>
           <swrl:classPredicate rdf:resource="#OntologyWithoutInstances"/>
           <swrl:argument1 rdf:resource="#y"/>
          </swrl:ClassAtom>
         </rdf:first>
         <rdf:rest>
          <swrl:AtomList>
          <rdf:first>
           <swrl:DatavaluedPropertyAtom>
            <swrl:argument1 rdf:resource="#y"/>
            <swrl:argument2 rdf:resource="...context/context.owl#formal"/>
            <swrl:propertyPredicate
```

```
                rdf:resource="...context/context.owl#hasFormalityLevel"/>
              </swrl:DatavaluedPropertyAtom>
             </rdf:first>
            <rdf:rest rdf:resource="...rdf-syntax-ns#nil"/>
           </swrl:AtomList>
          </rdf:rest>
         </swrl:AtomList>
        </rdf:rest>
       </swrl:AtomList>
      </rdf:rest>
     </swrl:AtomList>
    </rdf:rest>
   </swrl:AtomList>
  </rdf:rest>
 </swrl:AtomList>
</swrl:body>
</swrl:Imp>
```

Query results:

x
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#validdomainvalue
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#datapattern
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#characterpattern
http://www.inf.fu-berlin.de/inst/ag-nbi/research/wissensnetze/matching/matching.owl#valueranges

4 results found in 5 ms.

Figure 9: Results of rule: "Use only linguistic matchers for semi-formal ontologies"

References

[1] Jayant Madhavan, Philip A. Bernstein, E.R.: Generic Schema Matching with Cupid. In: Proc. of the 27th VLDB Conference. (2001)

[2] Doan, A.; Madhavan, J.D., P.; Halevy, A.: Ontology matching: A machine learning approach. Handbook on Ontologies (2004) 385–516

[3] Giuchiglia, F., Shvaiko, P.: Semantic matching. Knowledge Web Review Journal (2004) 265–280

[4] Poole, J., Campbell, J.: A novel algorithm for matching conceptual and related graphs. Conceptual Structures: Applications, Implementation and Theory (1995) 293–307

[5] McGuinness, D.; Fikes, R.R., J.; Wilder, S.: The Chimaera ontology environment. In: Proc. of the 17th National Conference on Artificial Intelligence (AAAI-2000). (2000) 1123–1124

[6] Stumme, G.; Alexander, M.: FCA-MERGE: Bottom-up merging of ontologies. In: Proc. of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001). (2001) 225–230

[7] Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In: Proc. of the 18th International Conference on Data Engineering ICDE02. (2002)

[8] Do, H.H., Rahm, E.: COMA—a system for flexible combination of schema matching approaches. In: Proc. of the 28th VLDB Conference. (2002)

[9] Erhard Rham, P.A.B.: A survey of approaches to automatic schema matching. Journal of Very Large Data Bases (2001)

[10] Shvaiko, P.: Iterative schema-besed semantic matching. Technical Report DIT-04-020, University of Trento, http://eprints.biblio.unitn.it/archive/00000550/01/020.pdf (2004)

[11] Hong-Hai Do, S.M., Rahm, E.: Comparison of Schema Matching Evaluations. In: Proc. of GI-Workshop "Web and Databases". (2002)

[12] Tu, K., Yu, Y.: CMC: Combining Multiple Schema-Matching Strategies based on Credibility Prediction. In: To appear: Proc. of 10th International Conference on Database Systems for Advanced Applications (DASFAA 2005). (2005)

[13] Cohen, W.W.: Integration of heterogeneous databases without common domains using queries based on textual similarity. In: Proc. of the ACM SIGMOD98 Conference. (1998) 201–212

[14] Doan, A.; Domingos, P., Halevy, A.: Reconciling schemas of disparate data sources: A machine learning approach. In: Proc. of the ACM SIGMOD01 Conference. (2001)

[15] Mochol, M., Oldakowski, R., Heese, R.: Ontology based Recruitment Process. In: Proc. of the GI2004 Conference, Ulm, Germany. (2004)

[16] Bizer, C., Heese, R., Mochol, M., Oldakowski, R., Tolksdorf, R., Eckstein, R.: The impact of semantic web technologies on job recruitment processes. In: 7th Internationale Tagung Wirtschaftsinformatik (WI05). (2005)

[17] Paslaru Bontas, E.: Using Context Information to Improve Ontology Reuse. In: Proc. of the Doctoral Consortium at the CAISE'05. (2005)

[18] Mochol, M., Paslaru Bontas, E.: A metadata-based generic matching framework for web ontologies. Technical Report TR-B-05-03, FU Berlin, ... (2005)

[19] Stamper, R.: The Semiotic Framework for Information Systems Research. Inf. Systems Research: Contemporary Approaches and Emergent Traditions (1991)

[20] Newman, M.: The structure and function of complex networks. SIAM Review (2003) 167–256

[21] Uschold, M., Grninger, M.: Ontologies: Principles, methods and applications. Knowledge Engineering Review (1996)

[22] Guarino, N.: Formal Ontology and Information Systems. In: Proc. of the International Conference of Formal Ontology in Information Systems (FOIS'98). (1998)

[23] Wand, Y., Weber, R.: Information Systems and Conceptual Modelling: A Research Agenda. Information Systems Research (2002)

[24] SWAD European Project: Semantic Web applications - analysis and selection (Deliverable Project SWAD IST-2001-34732) (2001)

[25] KnowledgeWeb European Project: Typology of ontology-based processing tasks (Deliverable D1.1.3 KnoweldgeWeb FP6-507482) (2004)

[26] Paslaru Bontas, E., Mochol, M., Tolksdorf, R.: Case Studies on Ontology Reuse. In: Proc. of the IKNOW05 International Conference on Knowledge Management. (2005)

[27] Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: Swrl: A semantic web rule language combining owl and ruleml. http://www.w3.org/Submission/SWRL/ (2004)