Semi-Automatic Knowledge Acquisition Through CODA

Manuel Fiorelli, Riccardo Gambella, Maria Teresa Pazienza, Armando Stellato, Andrea Turbati

ART Research Group, Dept. of Enterprise Engineering (DII),
University of Rome, Tor Vergata
Via del Politecnico, 1, 00133 Rome, Italy
{fiorelli, pazienza, stellato, turbati}@info.uniroma2.it; gambella.riccardo@gmail.com

Abstract. In this paper, we illustrate the benefits deriving from the adoption of CODA (Computer-aided Ontology Development Architecture) for the semi-automatic acquisition of knowledge from unstructured information. Based on UIMA for the orchestration of analytics, CODA promotes the reuse of independently developed information extractors, while providing dedicated capabilities for projecting their output as RDF triples conforming to a user provided vocabulary. CODA introduces a clear workflow for the coordination of concurrently working teams through the incremental definition of a limited number of shared interfaces. In the proposed semi-automatic knowledge acquisition process, humans can validate the automatically produced triples, or refine them to increase their relevance to a specific domain model. An experimental user interface tries to raise efficiency and effectiveness of human involvement. For instance, candidate refinements are provided based on metadata about the triples to be refined, and the already assessed knowledge in the target semantic repository.

Keywords: Human-Computer Interaction, Ontology Engineering, Ontology Population, Text Analytics, UIMA

1. Introduction

Efficient Information Management and Information Gathering are becoming extremely important to derive value from the large amount of available information. While the uptake of Linked Data [1] promoted uniform standards for the publication of information as interlinked datasets, the Web still consists mainly of unstructured content.

Dealing with this heterogeneous content asks for coordinated capabilities of several dedicated tools. In fact, development of knowledge acquisition systems today largely requires non-trivial integration effort and the development of ad hoc solutions for tasks, which could be better defined and channeled into an organic approach.

Platforms such as GATE [2] and UIMA [3] provide standard support for content analytics, while they completely demand to developers tasks concerning data transformation and publication. There have been a few attempts at completing information extraction architectures with facilities for the generation of RDF

(Resource Description Framework) [4] data, e.g. the RDF UIMA CAS Consumer[1] and Apache Stanbol[2]. The former is a UIMA component consuming the analysis metadata to generate RDF statements, while the latter focuses on semantic content management. These projects share the same approach, which consists in a vocabulary-agnostic serialization of metadata inferred by analytics. However, Stanbol provides a component, called Refactor Engine[3], which can refactor the extracted triples, possibly to target a user chosen vocabulary.

In our opinion, these approaches lack an overall perspective on the task, as well as a proposal of a gluing architecture supporting the triplification process as a whole.

In this paper we discuss the benefits of adopting CODA (Computer-aided Ontology Development Architecture), an architecture based on standard technologies (such as UIMA, OSGi[4], and OBR[5]) and data models (e.g. RDF, OWL [5] and SKOS [6]), which defines a comprehensive process for the production of semantic data from unstructured content. This process includes content analysis, generation of RDF out of extracted information, and involvement of humans for validation and refinement of the produced triples.

2. CODA Architecture

CODA (Fig. 1) extends UIMA with specific capabilities for populating RDF datasets and evolving ontology vocabularies with information mined from unstructured content.

Section 2.1 provides an overview of UIMA, while section 2.2 introduces the main components of CODA and details their role in the overall architecture. Finally, section 2.3 describes our approach for a user interface supporting validation and enrichment of the automatically extracted triples.

2.1. UIMA: Unstructured Information Management Architecture

In this section, we introduce some basic concepts about UIMA, in order to show how CODA can work jointly with this architecture.

UIMA is an architecture (OASIS Standard[6] in 2009) and an associated framework[7] supporting development and orchestration of analysis components – called Analysis Engines (AEs) – for the extraction of information from unstructured content of any media type.

UIMA adopts a data-driven approach for component integration, in which Analysis Engines represent and share their results by using a data structure called CAS (*Common Analysis Structure* [7]).

---

[1] http://uima.apache.org/downloads/sandbox/RDF_CC/RDFCASConsumerUserGuide.html
[2] http://stanbol.apache.org/
[3] https://stanbol.apache.org/docs/trunk/components/rules/refactor.html
[4] http://www.osgi.org/Specifications/HomePage
[5] http://felix.apache.org/site/apache-felix-osgi-bundle-repository.html
[6] http://docs.oasis-open.org/uima/v1.0/uima-v1.0.html
[7] http://uima.apache.org

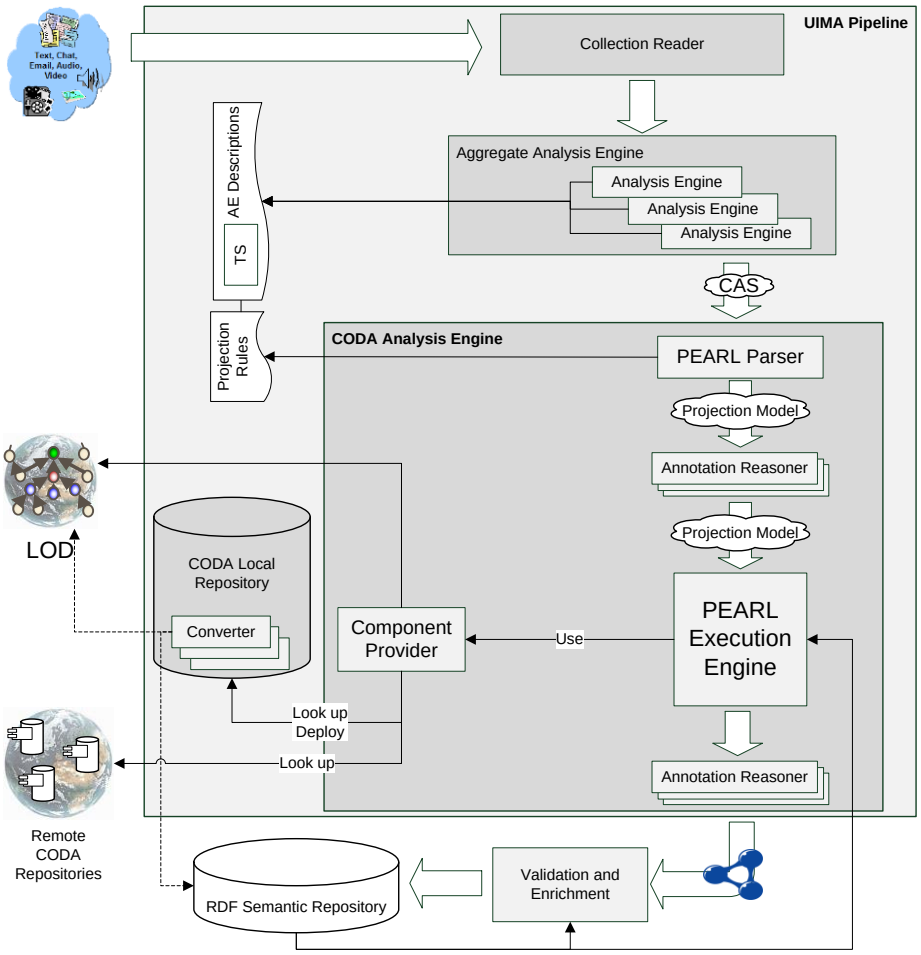

Fig. 1 Overall system architecture

A CAS contains both the information to be analyzed (called Subject of Analysis, or *Sofa*) and the extracted metadata, represented as typed feature structures [8] constrained by a model called *Type System* (TS) in the UIMA terminology. A specific kind of metadata (*annotations*) refers to specific sofa segments, and assigns explicit semantics to them. The CAS also contains an index over the metadata, which guarantees fast access to them, and allows iterating over the annotations, usually according to the order in which they appear in the sofa.

CODA fits the UIMA architecture by providing a concrete Analysis Engine that projects UIMA annotations onto RDF knowledge bases with a user-defined vocabulary.

2.2. CODA Analysis Engine

The *CODA Analysis Engine* represents the real innovative contribution provided by CODA that proves relevant to applications dealing with knowledge acquisition tasks.

While UIMA manages the unstructured content analytics, CODA provides components and their orchestration for projecting the resulting CAS onto an RDF Semantic Repository.

This projection is a complex process including the following activities:

- Selection of relevant information from within the CAS;
- Construction of RDF nodes out of UIMA metadata;
- Instantiation of user defined graph patterns with the prepared RDF nodes.

To support the user in the specification of this process, CODA provides a rule-based pattern matching and transformation language, called PEARL (ProjEction of Annotations Rule Language) [9].

The *PEARL Execution Engine* orchestrates the triplification process, and executes PEARL rules against the CASes populated by UIMA AEs. These specifications may include annotations, providing various kinds of metadata, which may guide extensions of the execution engine, or, if required, be propagated to the output triples. User-supplied *annotation reasoners* may automatically annotate the PEARL rules, or customize the mechanism for annotating the results.

PEARL syntax covers UIMA metadata selection and RDF graph construction patterns. Conversely, PEARL externalizes the construction of RDF nodes to external components, called *converters*, since this is a varied task, possibly depending on domain/application specific procedures, or on external services (such as OKKAM [10]).

PEARL specifications refer to converters indirectly in the form of URIs that identify the desired behavior (a *contract*) rather than concrete implementations. During the triplification process, the *Component Provider* follows a two-step procedure for resolving these references into suitable converters. By first, the provider lookups a match in well-known repositories (starting from the *CODA Local Repository*). If no candidate is found, the Component Provider activates a procedure to discover additional repositories in the Web. Indeed, complying with the principle of self-descriptiveness [11] and grounding in the Web, contract URIs dereference to RDF descriptions which include a list of authoritative repositories of known implementations.

This architecture enables mostly autonomous configuration of CODA systems, which is valuable when reusing PEARL documents written by third parties, as in the open and distributed scenario described in [12]. In fact, the reuse of PEARL specifications benefits from the lazy resolution of contracts into converters, as this approach allows choosing (among functionally equivalent ones) the implementation best fitting the non-functional requirements of a specific user, including consumption of computational resources, performance with respect to the tasks and even licensing terms.

2.3. CODA Human Interface

Human experts may revise the triples produced by *CODA Analysis Engine*, in order to raise their quality. An experimental user interface, called *CODA-Vis*, hides low-level issues concerning the RDF syntax, and supports more effective interaction modalities with domain experts. CODA-Vis (see later Fig. 3) benefits from the interaction with a Knowledge Management and Acquisition framework for RDF [13] for what concerns the UI and the interaction with the target semantic repository.

The validation aspect is limited to the binary judgment about the acceptability of individual triples. User support consists of additional features:

- Exploration of the source data which originated the suggestion, and of the semantic context of the elements in the triple;
- Alternative ways of representing the acquired data (e.g. list of triples, focus on common subjects, or groups of subject/predicate)

However, users may actively enter into the knowledge acquisition process by enhancing the automatically acquired results, in order to augment their preciseness, while strengthening their binding to the specific domain. For instance, the RDF triple "`art:Armando rdf:type ex:Person`" is obtained by the projection of a result from a UIMA Named Entity Recognizer into the generic class representing persons. However, a further evolution of the ontology for the Academic domain contains further specializations such as professor, researcher and student. The results would be more relevant, if the individual were assigned to the most specific class. In order to enable this kind of triple modification and improvement, the system has to organize and present the relations between the extracted information and the underlying RDF dataset, thus supporting the user in more effective decisions. In the given scenario, the system might pop up a class tree rooted at the class representing persons, thus providing the user with a focused slice of the underlying ontology, which should bring the user the right specialization.

3. Running Example

In this section, we step through the main phases of the triplification process by using a sample application. In section 3.1, we illustrate the use of PEARL for specifying the extraction and transformation process of information contained in UIMA annotations. For this example, we assume the existence of a UIMA Analysis Engine able to recognize a `Person` with its working `Organization` (if present). This means that the resulting CAS may contain several annotations, each one holding the identifiers of the recognized entities (the person and possibly the organization). In section 3.2, we show how CODA-Vis enables humans not just to accept or reject the suggested triples, but also to refine them with the help of a dedicated context sensitive user interface.

3.1. Metadata Matching and Projection

PEARL is a pattern-matching and transformation language for the triplification of UIMA metadata. It allows concise yet powerful specifications of the transformation,

```
prefix ontologies: <http://art.uniroma2.it/ontologies#>
annotations = {
    @Duplicate
    @Target(Subject, Object)
    Annotation IsClass.
  }
rule it.uniroma2.art.Annotator id:PEARLRule{
  nodes = {
    person uri personId
    organization uri organizationId
  }
  graph = {
    @IsClass(Object).
    $person a ontologies:Person .
    OPTIONAL { $person ontologies:organization $organization . } .
  }
}
```

Fig. 2 A PEARL rule example

while allowing the externalization (see section 2.2) of highly application and/or domain specific procedures.

Since its initial specification [9], PEARL has evolved towards greater expressiveness and support for more complex workflows. In this paper, we will emphasize the features introduced for supporting human validation and enrichment (see the previous section and section 3.2).

Fig. 2 provides an example of a PEARL document consisting of one rule, which drives the projection process in the running example.

A rule definition begins with the keyword *rule*, followed by the matched UIMA type name. The rule applies when an annotation from the index matches the declared UIMA type: the rule execution flows through a series of steps, specified within the rest of the rule definition.

The *nodes* section initializes *placeholders* with RDF nodes, which are constructed out of information that is extracted from the matched annotation using a *feature path* (a sequence of features inside a feature structure). For instance, in the rule in Fig. 2, the value of feature `personId` is transformed into a URI resource that is assigned to the placeholder `person`. Default conversion heuristics are applied, and are inferred based on the specified node type: in this case, as the target is a URI, the feature value is first normalized to match the restrictions on the URI syntax, and then prefixed with the namespace of the target semantic repository.

The *graph* section defines a graph pattern[8], which contains mentions (starting with the symbol "`$`") of placeholders originating from the *nodes* section. During a rule

[8] http://www.w3.org/TR/rdf-sparql-query/#GraphPattern

execution, this graph pattern is instantiated into RDF triples, by substituting the nodes they hold for the placeholders occurring in the pattern. For example, the triple pattern "`$person a ontologies:Person`" states that CODA will suggest a triple, asserting that the value inside the placeholder `person` is an instance of the class `ontologies:Person`. Since the other triple pattern is inside an *OPTIONAL graph pattern*, it is possible that CODA will not produce any triple, if the annotation triggering the rule has no value for the feature `organizationId` (so no value is stored inside the placeholder `organization`).

PEARL allows providing metadata about the elements of a graph pattern and the rules themselves, by attaching annotations to them. CODA provides a few meta-annotation types, for the further characterization of new annotation types, defined in the *annotations* section. *Target* indicates the elements, which an annotation type applies to: e.g. a triple as a whole, the subject, the predicate or the object. *Duplicate* controls whether annotations of a give type will be propagated to the results that originated from the annotated elements. In fact, with the exception of the aforementioned meta-annotation types, PEARL is almost agnostic with respect to the meaning of the annotations. In fact, annotations within a PEARL specification stand primarily as hints to extensions of the execution engine, which may somehow react to these annotations. Additionally, if an annotation is marked with *Duplicate*, it is subject to a uniform propagation process. Therefore, the output of the PEARL execution engine is a set of triples, possibly annotated with metadata helping their further exploitation.

While remaining consistent with this linear process, it is possible to plug reasoners that apply simple pattern matching techniques, to annotate either a PEARL document, or directly the output triples. These reasoners may handle common and simple cases, while manual annotation remains an option for solving the uncommon and hard ones.

In the example in Fig. 2, we first define the annotation *IsClass*, which is applicable only to the subject and the object of a triple. Then, we annotate the object of the first triple pattern with *IsClass.* As the annotation has been marked as *Duplicate*, CODA will propagate the annotation to the triples instantiated out of the triple pattern.

3.2. Human validation and enrichment

The annotated triples produced by the CODA Analysis Engine might not meet the quality requirements for being committed to the target semantic repository as they are. Therefore, we developed an experimental user interface for supporting the involvement of humans in the validation and enrichment of the acquired data. The validation involves a binary judgment, whether to accept or reject the proposed triples, while the enrichment consists in the specialization of the suggested triples to better suit the specific domain model.

Continuing the example before, it could be useful to specialize the class of each recognized person with respect to finer-grain distinctions made by the target vocabulary. To support this use case, the interface provides pluggable enhancers for different enrichment scenarios. In Fig. 3, for instance, a human validator activates the enhancer to assign an instance of the class `Person` resulting from a knowledge acquisition process to a more specific subclass found in the target ontology. To achieve this result the user should select a suggested RDF resource produced by

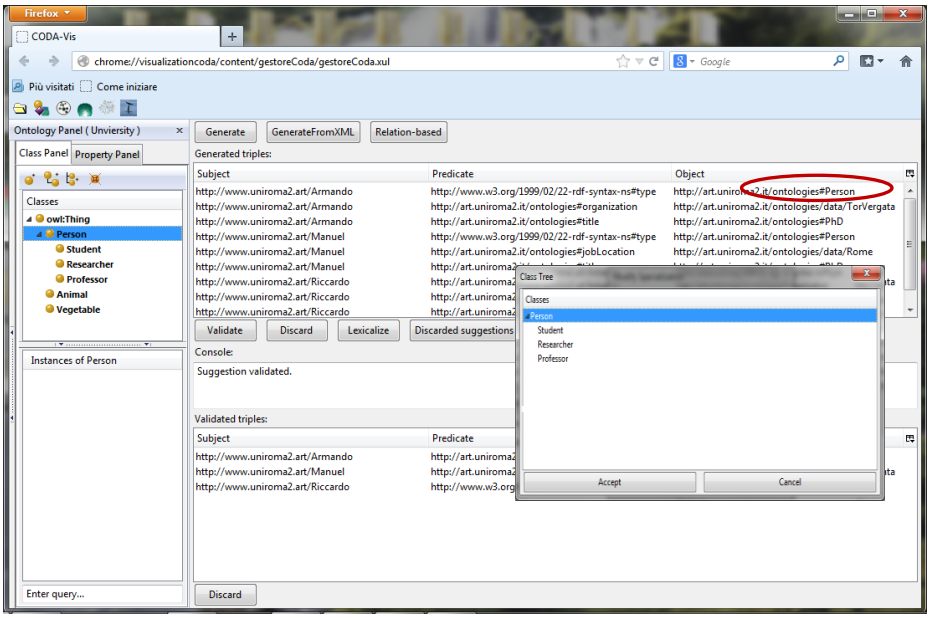

Fig. 3 Class Specialization

CODA. The enhancer suggests relevant modifications by dynamically constructing a class tree consisting of all the subclasses of the class `Person`. This specific behavior is offered since the selected resource has been annotated with *IsClass*. Further annotation types are associated to other behaviors, e.g. refining a property, refining a SKOS concept, or refining a street address by mashing-up popular online map applications. An extension mechanism in the UI assures that new enhancers can be associated to new annotation types. The selection of the relevant behaviors amounts to suitably annotating the triples, without any constraint on the mechanism to adopt. The primary means for producing these annotations is the duplication of the annotations in the PEARL transformation rules. However, these annotations do not need to be hard-coded in the rules; in fact, dedicated reasoners may automatically annotate the rules, usually by simple pattern matching techniques.

4. A Use Case: AgroIE

AGROVOC [14][9] is a thesaurus developed and maintained by The Food and Agriculture Organization (FAO)[10] of the United Nations (UN). It contains more than 30000 concepts in up to 22 languages covering topics of interest for the FAO. Recently the vocabulary has been migrated to SKOS [15] and published as Linked

[9] http://aims.fao.org/standards/agrovoc/about
[10] http://www.fao.org

Data. In compliance with the fourth Linked Data rule, reliable mappings [16] to other popular datasets have been produced by means of a semi-automatic process.

AGROVOC contains, among others, a number of concepts regarding plants, insects and, to a minor extent, pesticides. However, recent measurements have highlighted a low-level coverage of semantic relations among them, therefore motivating a massive introduction of new relation instances.

In the context of a collaboration between our research group and FAO, we have developed a semi-automatic platform, called *AgroIE,* for the augmentation of AGROVOC, based on the CODA framework.

We worked on web pages from Wikipedia and other specialized sites that tend to use a formal and repetitive language. By experimentation on about a hundred documents from the available sources, we identified a small set of high precision extraction patterns for the desired relations (e.g. an insect is a pest of a plant), in order to avoid an unsustainable load on the human validators.

AgroIE uses a UIMA pipeline for the linguistic analysis and lexico-syntactic pattern matching. We combined third-party analysis engines from DKPro (wrapping Stanford Core NLP) to ad-hoc engines for the recognition of the relevant entities (insects, plants and pesticides) and the relationship among them.

The clear separation between content extraction and triplification issues and the adoption of common interfaces to share has enabled parallel development and thus increased productivity. In our setting, two MsC students have been developing – in the context of their thesis work – the UIMA Information Extraction engine, a PhD with expertise in CODA has written the PEARL document once the UIMA type system had been defined, while a fourth developer familiar with linked data has written the two *converters/identity resolvers*.

5. Conclusions

In the context of a specific application, we confirmed that CODA benefits the development of knowledge acquisition systems in several ways. The use of UIMA for content analysis increases the chances of reusing independently developed solutions, thus possibly raising performance and reducing the development effort. Furthermore, the clear process definition underpinning CODA supports concurrent development on different aspects, while reducing the synchronization needs to the (possibly incremental) definition of shared interfaces. Finally, an experimental user interface completes the envisioned *Computer-aided* Ontology Development Architecture, towards more effective and productive interactions with humans during the validation and enhancement activities.

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. International Journal on Semantic Web and Information Systems (IJSWIS), Special Issue on Linked Data 5(3), 1-22 (2009)
2. Cunningham, H.: GATE, a General Architecture for Text Engineering. Computers

and the Humanities 36, 223-254 (2002) 10.1023/A:1014348124664.

3. Ferrucci, D., Lally, A.: Uima: an architectural approach to unstructured information processing in the corporate research environment. Nat. Lang. Eng. 10(3-4), 327-348 (2004)
4. Manola, F., Miller, E.: RDF Primer. In: World Wide Web Consortium (W3C). (Accessed February 10, 2004) Available at: http://www.w3.org/TR/rdf-primer/
5. W3C: OWL 2 Web Ontology Language. In: World Wide Web Consortium (W3C). (Accessed October 27, 2009) Available at: http://www.w3.org/TR/2009/REC-owl2-overview-20091027/
6. Isaac, A., Summers, E.: SKOS Simple Knowledge Organization System Primer. In: World Wide Web Consortium (W3C). (Accessed August 18, 2009) Available at: http://www.w3.org/TR/skos-primer/
7. Götz, T., Suhre, O.: Design and implementation of the UIMA common analysis system. IBM System Journal 43(3), 476-489 (July 2004)
8. Carpenter, B.: The Logic of Typed Feature Structures. Cambridge Tracts in Theoretical Computer Science (hardback) edn. 32. Cambridge University Press (1992)
9. Pazienza, M., Stellato, A., Turbati, A.: PEARL: ProjEction of Annotations Rule Language, a Language for Projecting (UIMA) Annotations over RDF Knowledge Bases. In : International Conference on Language Resources and Evaluation (LREC 2012), Instanbul, Turkey (2012)
10. Bouquet, P., Stoermer, H., Bazzanella, B.: An Entity Naming System for the Semantic Web. In : In Proceedings of the 5th European Semantic Web Conference (ESWC 2008) (2008)
11. Mendelsohn, N.: The Self-Describing Web. In: World Wide Web Consortium (W3C) Technical Architecture Group (TAG). (Accessed February 7, 2009) Available at: http://www.w3.org/2001/tag/doc/selfDescribingDocuments.html
12. Diosteanu, A., Turbati, A., Stellato, A.: SODA: A Service Oriented Data Acquisition Framework. In Pazienza, M., Stellato, A., eds. : Semi-Automatic Ontology Development: Processes and Resources. IGI Global (2012) 48-77
13. Pazienza, M., Scarpato, N., Stellato, A., Turbati, A.: Semantic Turkey: A Browser-Integrated Environment for Knowledge Acquisition and Management. Semantic Web Journal 3(3), 279-292 (2012)
14. Caracciolo, C., Stellato, A., Morshed, A., Johannsen, G., Rajbhandari, S., Jaques, Y., Keizer, J.: The AGROVOC Linked Dataset. Semantic Web Journal 4(3), 341–348 (2013)
15. Caracciolo, C., Stellato, A., Rajbahndari, S., Morshed, A., Johannsen, G., Keizer, J., Jacques, Y.: Thesaurus Maintenance, Alignment and Publication as Linked Data. International Journal of Metadata, Semantics and Ontologies (IJMSO) 7(1), 65-75 (August 2012)
16. Morshed, A., Caracciolo, C., Johannsen, G., Keizer, J.: Thesaurus Alignment for Linked Data Publishing. In : International Conference on Dublin Core and Metadata Applications, pp.37-46 (2011)