UIMA2LOD: Integrating UIMA Text Annotations into the Linked Open Data Cloud

Claudia Bretschneider[1,2(✉)], Heiner Oberkampf[2], and Sonja Zillner[2,3]

[1] Center for Information and Language Processing, University Munich, Munich, Germany
claudia.bretschneider.ext@siemens.com
[2] Siemens AG, Corporate Technology, Munich, Germany
[3] School of International Business and Entrepreneurship, Steinbeis University, Berlin, Germany

Abstract. The LOD cloud is becoming the de-facto standard for sharing and connecting pieces of data, information and knowledge on the Web. As of today, means for the seamless integration of structured data into the LOD cloud are available. However, algorithms for integrating information enclosed in unstructured text sources are missing. In order to foster the (re)use of the high percentage of unstructured text, automatic means for the integration of their content are needed. We address this issue by proposing an approach for conceptual representation of textual annotations which distinguishes linguistic from semantic annotations and their integration. Additionally, we implement a generic UIMA pipeline that automatically creates a LOD graph from texts that (1) implements the proposed conceptual representation, (2) extracts semantically classified entities, (3) links to existing LOD datasets and (4) generates RDF graphs from the extracted information. We show the application and benefits of the approach in a case study on a medical corpus.

1 Semantic Web and Unstructured Data Sources

The LOD (Linked Open Data) cloud is a resource that uses Semantic Web technology to gather and interconnect all kinds of useful publicly available web information from any domain. In particular, the LOD cloud is a very valuable knowledge resource for reuse in any kind of data-based applications, such as analytics, search, etc. However, as of today, the integration of text content as triples into the LOD clouds often relies on human interaction. The integration of data from previously already structured sources can be realized by means of schema transformation, but the content enclosed in texts cannot be automatically integrated into the LOD cloud, thus, remains concealed in string objects.

Therefore, in this work we propose an approach that automatically extracts semantically classified entities and relations between them from unstructured text and subsequently creates a LOD graph (as illustrated in Figure 1). This resulting graph contains the triplified representation of information enclosed in the processed text, which is already linked to existing LOD resources. Compared

P. Klinov and D. Mouromtsev (Eds.): KESW 2015, CCIS 518, pp. 16–31, 2015.
DOI: 10.1007/978-3-319-24543-0_2

to so far chosen representations of text in LOD datasets as string objects, which conceal the text content, we are now able to reference the structured text content by URIs and include it into the LOD cloud.

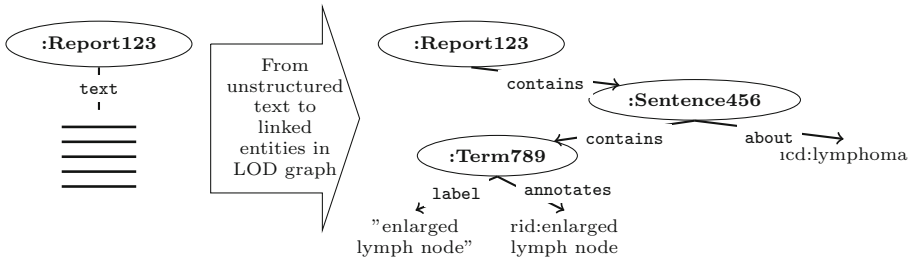

Fig. 1. Simplified illustration of target transformation from unstructured text to LOD graph of linked entities

There are several scenarios that can benefit from the resulting extended LOD cloud: For instance, a mechanism that allows the seamless integration of unstructured content into the LOD cloud drives its growth and enriches this knowledge resource. Furthermore, by enhancing information extraction algorithms with domain knowledge from the LOD cloud, a more holistic semantic understanding, and thus, interpretation of text documents becomes possible.

A first attempt to satisfy the request for structured representation of unstructured text is done by text analytics frameworks such as UIMA, whose usage became a de-facto standard and which already deliver modules for the triplification of their proprietary annotation structures. However, in most cases, their approaches only deliver incomplete solutions that fail to integrate the resulting annotations in the LOD cloud. For UIMA, the currently existing solution operates with data loss and needs intensive refactoring for correct and full creation of RDF graphs, hence, is currently not valuable for the integration task.

Therefore, the contribution of this paper is threefold: First, in Section 2 we introduce our approach for conceptual representation based on three dimensions of the textual annotations. Second, in Section 3 we describe which components with corresponding features have to be implemented in a UIMA pipeline to extract the relevant entities and relations and to create the LOD graph. Third, we evaluate the benefits of this approach in a study on extraction of a LOD dataset from medical texts (see Section 4), which is finally integrated into a semantic model for further data analytics.

2 Conceptual Representation of Text Annotations

2.1 Distinction of Three Dimensions of Annotation Types

We propose a general approach for the creation of new LOD datasets from unstructured text sources. This includes both a pipeline to fulfill the

requirements posed to the creation of RDF triples and an approach to represent the annotations created as metadata from the textual content. The conceptual representation acts as common vocabulary and agreed standard. Analyzing the results from text analytics pipelines, we identify three dimensions of annotations that are able to cover the variety:

1. *Linguistic Annotations* that reflect the basic linguistic units in the texts
2. *Semantic Annotations* that represent useful information of the target domain
3. *Structural Annotations* that interconnect the linguistic and the semantic world of the text annotations

Following the concept of *separation of concerns*, we identify the linguistic and semantic annotations as being semantically independent, but within a textual context they have a structural relation.

2.2 Representation of Linguistic Annotations Employing the NIF Ontology

For representation of the linguistic annotations and their relations, we draw on the NLP Interchange Format (NIF) ontology, which was created as core component in the context of the NIF project [4], [11] We only reuse a subset of classes and properties from the NIF ontology to represent the linguistic units in texts (see Figure 2), which is relevant for representing the semantic annotations. The

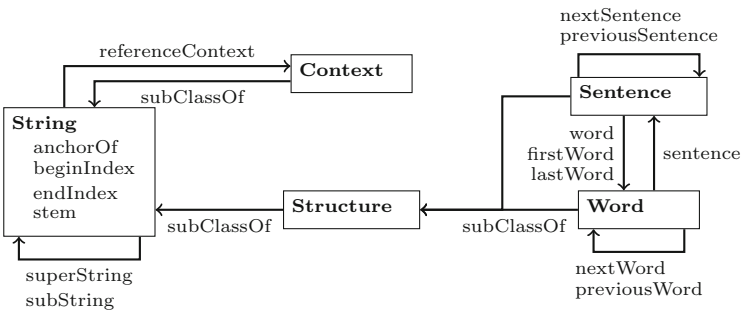

Fig. 2. (Partial) NIF ontology for representation of linguistic annotations.

main class of the model is `nif:String`, which is "the class of all words over the alphabet of Unicode characters" [4]. We model textual reports as `nif:Context`, which hold all other linguistic units via `nif:referenceContext`. `nif:Sentence` and `nif:Word` are used to model the sentence structure and its containing tokens. We also use the `nif:Word` class to model subtokens of compound terms.

We use the NIF ontology, since it satisfies our requirement for distinct representation of the linguistic information in a given text. Other models such as lemon[1] already leverage Semantic Web concepts, but rather target to model the

[1] http://lemon-model.net/lemon#

linguistic representation of lexicons or dictionaries from LOD resources than the content in (domain-specific) texts.

2.3 Representation of Semantic Annotations

We describe the annotations resulting from the semantic analysis of the text (Section 3) using the term *Semantic Annotations*. We subsume two types of annotations under this term:

1. In the original sense of the term, we include *annotations that are built based on ontology vocabulary*. The details of this step are described within the named entity recognition pipeline step (Section 3.2.3). There is no dedicated representation model for ontology-based semantic annotations necessary; we employ the concepts included in the Open Annotation (OA) data model. (Details are outlined in Section 2.4)
2. We also use the term Semantic Annotation to reference *annotations that are specific to the use case* tackled. An example of an application-specific annotation is a measurement annotation (shown in Figure 3 in triplified format), which is a metadatum annotated to any sequence of strings that is semantically interpreted as measurement.

```
@prefix ex: <http://example.org/stuff/1.0/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema/> .

<http://example.org/stuff/1.0/MeasurementAnnotation124112>
              ex:begin    20;   ex:end   26;
              ex:text     "1.5 cm"^^xsd:string;
              ex:measures 1.5;
              ex:unit     "cm"^^xsd:string .
```

Fig. 3. Example RDF resource for application-specific annotation type `MeasurementAnnotation`

2.4 Representation of Structural Annotations Employing the Open Annotation (OA) Ontology

We claim, the interconnection of linguistic and semantic annotations has to be explicitly modeled to show how the semantics is mapped to the linguistic entities and to support the backtracking of the sources of semantic information in the text. We employ the Open Annotation (OA) ontology [3] for modeling this structural relationship. The basic elements *body*, *target* and *annotation* of the ontology enable a generic content annotation approach.

We use the *target* to represent the linguistic entity of the text. The *body* is the part to represent the semantic annotations. The *annotation* is the part that connects the body and target, thus, holds the semantics of which linguistic part in the text is annotated by which semantic concept. For illustrating the semantic annotation of linguistic elements in text, we identify two (OA) annotation

types: On the one hand, we model the case that a single target is annotated as *simple annotation* model as illustrated in Figure 4. On the other hand, we also model annotations that associate a sequence of multiple consecutive tokens (a *phrase*) with a *multi-target annotation*. As illustrated in Figure 5, we use the composite element of the OA ontology to model the multitude of tokens, because the order of the tokens is not of importance. As seen in the example, no matter whether the text sequence is `axillary lymph node` or `lymph node axillary`, it is annotated using the same body (`radlex:RID1517`).

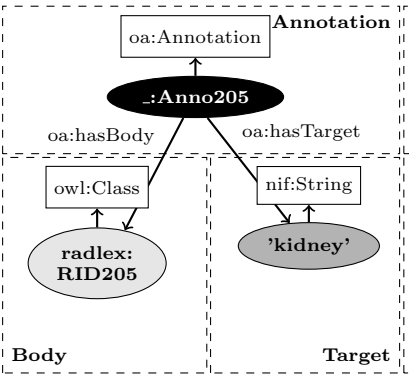

Fig. 4. Simple Annotation Model based on OA ontology. [2]

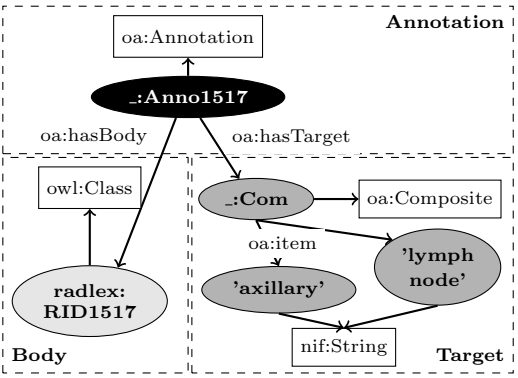

Fig. 5. Multi-target Annotation Model based on OA ontology. Multiplicity of targets is illustrated with `oa:Composite` element. [2]

3 UIMA-Based Implementation for Automated Creation of LOD Datasets from Unstructured Data Sources

Besides the high-level conceptual modeling of the target LOD dataset, we also present an end-to-end UIMA pipeline that includes all kinds of modules required to (1) extract recognized information pieces and their relations, (2) support our conceptual representation of text annotations and (3) create a serialization in RDF format. At the same time, the pipeline fulfills all requirements posed to new LOD datasets. The pipeline is configurable, so that texts of any domain can be used as input and the pipeline outputs a RDF graph that is at the same time already a LOD dataset that can be published as a part of the LOD cloud.

3.1 The UIMA Framework and Fundamental Components

For implementation of an end-to-end pipeline we build on the Unstructured Information Management Architecture (UIMA) framework, which handles unstructured resources, such as text, and facilitates their annotation and

[2] Undefined edges denote `rdf:type` relations. For reasons of simplicity, the ellipses representing linguistic resources are referenced by their labels.

extraction of structured information. The core components of the framework are *analysis engines* (or *annotators*) that extract defined entity types from the input texts as annotations. The framework includes an internal UIMA annotation index, the *Common Analysis Structure (CAS)*, that stores the annotation instances created. In the UIMA framework, the structure of annotations is defined in *type systems*, where each type of annotation gets assigned a list of features, whose values can be specified using three different data types:

1. Features with primitive data types (such as numeric and alphanumeric), for which an `owl:DataProperty` is created during triplification.
2. Features with complex data types that reference other annotation types, thus enable the relation between annotation instances, and for which an `owl:ObjectProperty` is created during triplification.
3. Features with arrays of primitive or complex data types, which can hold multiple instances (which are not created as `owl:FunctionalProperty` unlike the other feature types).

Within the UIMA framework, the architectural component that is responsible for integration of resulting annotations to external resources is defined as *consumer*. It extracts the annotation information from the CAS and persists selected information to resources such as search engine indexes, relational databases, or (as in our case) triple stores.

3.2 Description of the UIMA Pipeline's Modules

In order to fulfill the requirements towards correct conceptual and structural representation of the resulting RDF graph, our pipeline requires five functional processing steps: (1) *Linguistic Preprocessing* (2) *Information Extraction* (3) *Named Entity Recognition* (4) *Open Annotation (OA) Creation* and (5) *Triplification*. The correlating functionalities are implemented in one or more UIMA annotators. The steps of the pipeline are designed in a way so that the resulting RDF graph will be already a valid LOD dataset, which can be integrated into the cloud without further postprocessing. Thus, all pipeline's modules fulfill the four requirements imposed by the fundamental Linked Data principles formulated by Tim Berners-Lee[2] (marked in boldface):

1. **Use URIs as names for things**
2. **Use HTTP URIs so that people can look up those names**
   During the *Triplification* step each annotation instance gets assigned a unique ID represented as HTTP URI, so that the first two requirements are fulfilled.
3. **When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)**
   For extraction of useful information we included a number of steps (*Linguistic Preprocessing*, *Information Extraction*, *Open Annotation (OA) Creation*), which at the same time support the envisioned conceptual representation. Again here, for the correct structural representation of the resulting triples, the *Triplification* step is implemented to use the defined standards.

4. **Include links to other URIs, so that they can discover more things.** Finally, to enhance the LOD cloud with additional information entities that are also interconnected with existing datasets, we included the *Named Entity Recognition* step.

3.2.1 Linguistic Preprocessing

As first step in the pipeline, we conduct the basic linguistic analyses, such as sentence splitting and tokenization. In addition, we conduct a compound splitting step (to instantiate the object properties `nif:subString` and `nif:superString`) and add three annotators in order to resolve the object properties representing the linguistic relations (e.g. `nif:word` and `nif:sentence`). We describe details on the sentence splitting and semantic compound splitting in [7]. The underlying UIMA type system adapts the subset of NIF concepts as shown in Figure 2 and the annotation types are organized analogous to the `rdfs:subClassOf` hierarchy of the NIF ontology. We map the ontology classes to UIMA annotation types to have the implementation independent from the ontology, and thus, are able to support the adaptation of *existing* UIMA annotators for the respective task – not only annotators designed specifically for this task.

3.2.2 Information Extraction

The information extraction step targets the annotation of *domain-specific semantic annotations*. The information pieces resulting from this step are descibed by Berners-Lee as useful information. However, what useful information is and how it is modeled depends on the context of the application or the domain addressed. Therefore, the domain respectively the use case defines the annotators necessary. Depending on the type of information to be extracted, the implemented annotators make use of the multitude of NLP algorithms to identify and annotate the respective information in the text.

One example annotator implemented for the case study is an annotator that recognizes strings interpreted as measurements (`MeasurementAnnotator`) with the attributes illustrated in Figure 3. The underlying algorithm is based on precompiled regular expressions or list of entities (for the measurement units) and are also able to recognize other medical semantic types, such as diseases (based on typical suffixes), dates (regular expressions), abbreviations and negations (lists).

3.2.3 Named Entity Recognition (NER)

The second type of semantic annotations – *ontology-based semantic annotations* – are created during this processing step. Hence, this step satisfies two purposes: First, it serves the need for identification of domain-specific semantically classified concepts in the text and subsequently interconnect the knowledge in the ontology to the textual information. Second, this step satisfies the requirement to link to other, existing LOD datasets, since the resulting LOD dataset needs to link to existing entities (LOD principle #4). Thus, this implies that at least one of the employed ontologies for NER has to be published as resource in the LOD cloud.

The realization of ontology-based semantic annotations is based on the UIMA Concept Mapper[3]. To recognize ontology concepts it employs a previously created XML dictionary that contains the vocabulary to match and the metadata to annotate. For our case study, we transform the RadLex vocabulary into a dictionary that is Concept Mapper-compatible. Each ontology concept is transformed into a UIMA token (example shown in Figure 6) with its URI and preferred name as attributes. Additionally, each synonym and non-English variants are added as UIMA variants, which are used during the annotation process. The Concept Mapper creates annotations by mapping the stemmed UIMA variants from the XML dictionary to the text's tokens. If matches are found, an annotation is created that encloses the URI of the matching ontology concept. The Concept Mapper creates annotations for phrases if each of the tokens in a dictionary phrase can be mapped to a token within a sentence. The URI is the only information necessary to be annotated, since its semantics is already defined in the source ontology.

```
<token RID="RID1301" URI="http://www.owl-ontologies.com/Ontology1392225293.owl#RID1301"
          pn="lung" semanticClass="anatomical">
          <variant base="lung"/>
          <variant base="Lunge"/>
          <variant base="pulmo"/>
</token>
```

Fig. 6. Sample entry from UIMA Concept Mapper-compatible dictionary

3.2.4 Creating Open Annotations (OA)

The final step to reach the targeted conceptual representation model of the annotations is this transformation step to create the OA annotations shown in Figures 4 and 5. Simple annotations are created if single tokens in a sentence are annotated. Multi-target annotations are created if a semantic annotation needs to reference multiple linguistic entities for full semantic coverage. Within the created OA annotations, the respective linguistic and semantic annotations are just referenced by their URIs, since they already have been created in the proceeding steps.

3.2.5 Triplification

The main objective of this step is the correct representation of the annotations created in the steps before. Analyzing the data structure-wise, how existing modules triplify text annotations and where their shortcomings lie, we found five aspects to consider when creating RDF triples from text annotations. We conducted the analysis with strong influence from the existing UIMA RDF consumer[4] from the UIMA framework. As a result from this analysis, we developed

[3] https://uima.apache.org/downloads/sandbox/ConceptMapperAnnotatorUserGuide/ConceptMapperAnnotatorUserGuide.html

[4] UIMA CAS2RDF consumer http://uima.apache.org/downloads/sandbox/RDF_CC/RDFCASConsumerUserGuide.html

a module to transform the CAS into a RDF graph, which is intended to work for any kind of annotation types. It builds on the existing RDF consumer, but addresses the following limitations. This is the most important component for translating the structural requirements for LOD datasets into technical implementation.

1. **Declarative modeling of data properties (primitive data type features).** In the graph that the existing RDF consumer produces, each annotation feature that is triplified to `owl:DataProperty` is assigned a resource with two literals representing the feature name and feature value, so that each feature needs three triples for being represented. For us, this representation is not intuitive and not easy to process subsequently. Since this is more convenient and intuitive, we prefer a representation where the feature value corresponds to the literal's lexical form and the feature name corresponds to the edge label, which also avoids unnecessarily large numbers of triples.
2. **Typed literals instead of plain piterals.** The problem with triples, whose objects are defined as plain literals, is that they cannot be interpreted with their correct data type but rather have per default a string type assigned. This hinders the automated analysis of the data. As each feature is defined with a distinct data type in the UIMA type system, it is easy to access this information and reuse it for triplification of the annotations. Therefore, we define each literal as typed literal, which has a lexical form and an additional data type URI.
3. **Resolution of ambiguities with unique IDs.** In the RDF graph, each resource is identified by its URI. However, if the ID is not unique, because the calculation of the URI is not correct, this ambiguity leads to wrong representation of the annotations created. As a resolution, we calculate a hashcode for each text annotation that combines all available (primitive) feature names and values of each annotation instance in order to resolve this ambiguity and deliver a unique numeric identifier. This hashcode is combined with the annotation type name and an application-defined HTTP path to the final URI of a resource (example shown in Figure 3). A (desired) side effect of this calculation is that there is only a single instance of an annotation created, which subsequently can be referenced multiple times.
4. **Triplification of object properties (references between annotations).** The serialization of `owl:ObjectProperty` from features with complex data types requires a more sophisticated handling. The existing consumer does not resolve the objects attached to object properties, but rather triplifies the referenced annotation to its full string representation. This representation loses the link between the annotation instances. To prevent this loss, we implement a mechanism that considers the complex data type of the feature, triplifies the object property (if not already done) and maintains the reference.
5. **Triplification of non-functional properties (multi-value annotation features).** Per default all annotation features are defined as functional properties. If a feature allows multiple instances (for which the cardinal-

ity restriction of functional properties do not apply), each value needs to be triplified separately. This applies for feature values that are triplified to `owl:DataProperties` and `owl:ObjectProperties`, respectively. The existing consumer loses this information, since it represents the whole set using the string `'FSArray'`.

If the aspects just mentioned are not considered, this causes the effects of data loss, which leads to misinterpretation of the annotation values. Therefore, our implementation of a UIMA2LOD UIMA consumer builds upon the existing consumer functionality, but resolves the listed issues and now integrates more information – and even shows a more intuitive and leaner representation. At the same time, the resulting graph already represents a valid LOD dataset that can be published as part of the LOD cloud without any further postprocessing.

4 Case Study on Integrating RDF Annotations into the Model for Clinical Information (MCI)

4.1 General Goal and How the Approach Fits

Using our approach on representing and triplifying textual annotations, we want to show how structured information can be extracted from an example text and how it can be subsequently transformed into a LOD dataset. Also, we want to demonstrate how the graph can be used in Semantic Web applications for reasoning on the information enclosed in the texts. Therefore, we use a medical corpus and show how the pipeline operates and which output it produces. In a further step, this output is integrated in a semantic Model for Clinical Information (MCI) that integrates clinical information in a patient-centric way.

4.2 Resources Used in the Case Study

Corpus of Medical Reports. One of the core resources applied in our case study is a medical corpus containing 2,713 German radiology reports that describe the health status of lymphoma patients. The texts are provided by our clinical partner, the University Hospital Erlangen.

RadLex Ontology. To satisfy the requirements for creating links to existing LOD datasets and since the its vocabulary matches the vocabulary of the corpus, we employ the RadLex ontology [9]. RadLex is a medical ontology for the radiology domain and aims to unify the domain's vocabulary for the purpose of organization, indexing and retrieval of radiology resources, such as textual radiology reports. The current version of Radlex 3.12 (11/2014) contains 74,875 terms. RadLex is also published as LOD dataset `bioportal-rid`.

Model for Clinical Information (MCI). The MCI [8] provides the basis for data integration and knowledge exploration . It defines the most important structural concepts (classes and relations) for the representation of clinical data. This is, in the first place structured data about the patient like diagnoses and findings, as well as provided examinations, procedures and therapies. The MCI is based on selected upper- and midlevel ontologies from the OBO library and reuses established schemas like the Dublin Core. It is used in combination with large reference terminologies such as RadLex or the Foundational Model of Anatomy (FMA) to represent clinical findings from unstructured data resources.

4.3 Creation of Text Annotations and the RDF Graph

For illustrating the outcome of the process and how it can be used for further processing, we use the sentence `Axillary lymph node with diameter 1.5 cm.`, which is representative for the corpus given. Following the process steps and iterating the UIMA pipeline, several annotations are created (see Figure 7).

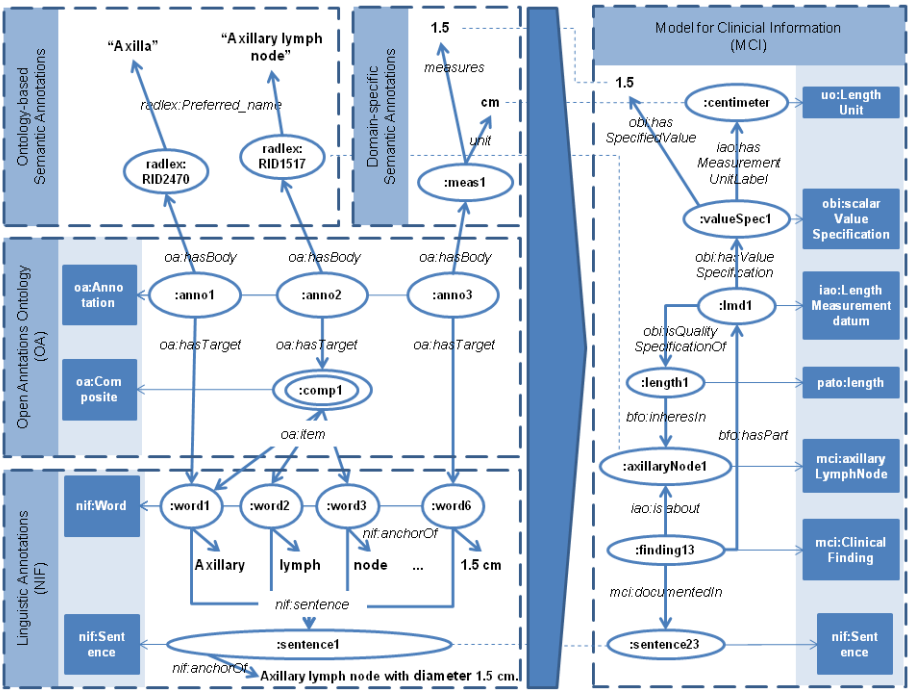

Fig. 7. Example illustrating a subset of annotations created from the sentence `Axillary lymph node with diameter 1.5 cm.` (left-hand side) and their transformation into the clinical model MCI (right-hand side).[5]

[5] Undefined edges denote `rdf:type` relations.

The linguistic annotators produce one sentence annotation, six token annotations and the respective relations defined in NIF schema. As domain-specific annotation type a measurement annotation with measure `1.5` and unit `cm` is identified. As results from the RadLex-based NER step, two possible anatomical entities are annotated in the sentence: `axilla` and `axillary lymph node`. Finally, the structural relations between the linguistic and the semantic annotations are generated as OA annotations. After being processed by the pipeline the semantic text content triplified to RDF and can be used for further analysis.

4.4 Transformation of Text Annotation RDF Graph into MCI

In general, the annotations from radiology reports represent clinical information, such as the observations made and the findings discovered. However, in the context of medical information, the textual information from examination reports are just one single piece to the holistic information describing the current and past health status of a patient. For clinical decision making this data however needs to be linked and interpreted to gain a holistic view on the patient data. In order to gain such a holistic representation, we employ the MCI. The process to transform the structured annotation data to the schema and semantics of MCI requires several steps:

Transformation of RDF Graph into the MCI Schema. Since the annotations are already represented in RDF format, we can simply use SPARQL queries to facilitate the required schema transformation. Once established, the transformation of the RDF graph to the MCI schema can be conducted automatically.

Transformation and Normalization of Measurement Annotations. The provided piece of measurement information is directly transformed to a size finding. Therefore, an instance of a length measurement datum, which describes a length quality, is created. We map the unit (represented in the annotation as a string) to an entity of the Units Ontology (UO) and normalize the value to centimeters (if necessary) to obtain a representation as shown in Figure 7.

Disambiguation of Anatomical Entities. Finally the correct relation to the described anatomical entity has to be created. We use the disambiguation algorithm described in [7] to determine the correct anatomical entity from the annotated ones (`axilla` and `axillary lymph node`), which is the latter.

Inference. Now, the MCI is combined with RadLex to link findings about same or similar anatomical entities from consecutive examinations, e.g. for treatment evaluation. Additionally, formalized medical knowledge about normal size specifications is used to infer that the finding represents an *abnormally enlarged* axillary lymph node, since normal size lymph nodes typically measure up to 1 cm. In a further diagnostic process this finding can be interpreted in the context of diseases: An enlarged lymph node is an abnormality caused by the immune system and related to specific types of cancer, such as lymphoma.

Finally, we point out that this information can be only inferred, because two information resources are combined: text and LOD knowledge. Since this clinical information is only reported as text, it requires our textual annotation and transformation process to link the enclosed information with clinical LOD knowledge.

4.5 Evaluation

4.5.1 Quantitative Comparison of UIMA2LOD Consumer

For a quantitative evaluation of the strength of the proposed approach, we compare the RDF graphs resulting from the triplification substep using a UIMA pipeline that integrates either the CAS2RDF consumer or our new triplification component UIMA2LOD but with the same annotator components (c.f. Table 1).

Table 1. Qualitative comparison of the RDF graph from annotation pipelines integrating either the CAS2RDF or the UIMA2LOD consumer

	CAS2RDF	**UIMA2LOD**
# NIF Annotations	1,506,029	
# Semantic Annotations	416,251	
# OA Annotations	486,425	
# Triples	144,215,917	47,700,368
# Triples with wrong serialization of		
Non-Functional Properties	681,745	–
Object Properties	8,302,645	–
Runtime of Annotation Pipeline	24h for 180 docs	9 min for all 2713 docs

indent Given the medical corpus with 2,713 texts as input, 2,408,705 text annotations are created. The graph resulting from the UIMA2LOD consumer only needs a third of the triples compared to the graph resulting from the CAS2RDF consumer. The difference is due to the simplified representation of annotation features with name and values. However, a more detailed analysis of the delta is not possible because the numbers are blurred by the wrong serialization of non-functional and object properties by the CAS2RDF consumer. This irregular triplification leads to data loss which can not be corrected by post processing either. Also, the existing process is highly resource intense. It leads to an extrapolated time of 2 weeks for the serialization of the whole corpus. (We stopped the process after 24h with 180 reports triplified.) The reason is the attempt to serialize object properties (and all of their subsequent object properties) as string representations. Because of this limitation, we only recommend the usage of the CAS2RDF module for creating RDF representation of un-linked text annotations. Whereas, the usage of the UIMA2LOD component, which addresses those limitations, is recommended for highly linked graphs, which are intented to be

integrated into the LOD cloud as datasets. Finally, only our pipeline is able to automatically create a valid LOD dataset from unstructured text, in a reasonable time. The CAS2RDF consumer does not create a valid LOD dataset, because it fails to comply to two core requirements: First, it does not create URIs as unique IDs, but rather IDs in an own format. Second, it fails to create links to existing datasets, thus does not add to the creation of a larger linked data cloud.

4.5.2 Comparison with Other Text2Semantic Web Systems

Only in the recent years the question of how to integrate results from NLP pipelines and the Semantic Web has become of importance. However, numerous information extraction pipelines for unstructured text can be found that take the first step to transform their outcomes into RDF triples. While taking a further step, they also succeed in extracting entities from the text that are part of LOD resources, such as MeSH for biomedical texts [10], plant information [5] or general information from DBpedia [1]. Thus, while their primary goal in terms of information extraction is the identification of *existing* entities, they focus on supporting the discovery of so far undiscovered relations between these entities.

Another field of research is the population of existing ontologies. Kawamura et al. [5] extract newly explored plant information from text. The resulting RDF representation is a requirement for their integration. Just recently, Augenstein et al. [1] propose a domain-independent approach for information that transforms the results into RDF without predefined schema. The main limitation from these existing systems is their lack in the ability to support the creation of new LOD datasets, either because of the missing link of the RDF graphs to existing LOD datasets or because they fail to generate new information from the given textual resource, while they just recognize existing LOD instances.

4.6 Future Work

We plan to fully integrate the UIMA pipeline into the Semantic Web context by automating the initial step of transforming a semantic model into the UIMA type system, which we currently conduct manually. First such attempts have been made by Lui [6] and Verspoor [12]. They also show how the resulting graph can be used within reasoning applications. Further, we regard the applied medical corpus as medium-size. We aim to extract information from corpora of Big Data size (which is text from about 10,000 patient from examinations over up to 30 years time). Additionally, we will evaluate the individual steps of the pipeline, in particular the quality of the information extraction and NER steps.

5 Conclusion

In this work we present an automatic approach for creating LOD datasets from entities and relations extracted from unstructured texts, whose content has been concealed so far. On the one hand, we propose an approach to conceptual representation that separates linguistic, semantic and integrating annotation. On the

other hand, we introduce a (reusable) UIMA pipeline, that extracts and serializes the structured annotation information from the input text. Compared to existing annotation systems our pipeline is advantageous, because we are able to create new LOD datasets from unstructured data sources in an automated manner. Our pipeline creates a full representation of text annotations without data loss and reuses URIs from existing LOD resources for seamless integration. The resulting RDF graph can be published as-is as a LOD dataset, since it fulfills all necessary requirements imposed. This results in the availability of semantic information from so far unexplored data sources for a subsequent data analysis. At the same time, also existing LOD datasets can benefit from this integration effort, since it is also applicable for unstructured text enclosed. In a case study we show the advantages of our approach: We decrease the number of triples necessary to a third, while preventing data loss from wrong serialization of object and non-functional properties. The subsequent data analysis reveals so far undiscovered knowledge, because the content in unstructured texts can now be integrated with LOD resources.

Acknowledgments. This research has been supported by the KDI project funded by the German Federal Ministry of Economics and Technology under grant number 01MT14001 and by the EU FP7 Diachron project (GA 601043).

References

1. Augenstein, I., Padó, S., Rudolph, S.: LODifier: generating linked data from unstructured text. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 210–224. Springer, Heidelberg (2012). http://dblp.uni-trier.de/db/conf/esws/eswc2012.html#AugensteinPR12
2. Berners-Lee, T.: Linked Data - Design Issues, July 2006. http://www.w3.org/DesignIssues/LinkedData.html
3. Ciccarese, P., Ocana, M., Garcia-Castro, L.J., Das, S., Clark, T.: An open annotation ontology for science on web 3.0. J. Biomedical Semantics **2**(S–2), S4 (2011). http://dblp.uni-trier.de/db/journals/biomedsem/biomedsem2S.html#CiccareseOGDC11
4. Hellmann, S., Lehmann, J., Auer, S., Brümmer, M.: Integrating NLP using linked data. In: Alani, H., et al. (eds.) ISWC 2013, Part II. LNCS, vol. 8219, pp. 98–113. Springer, Heidelberg (2013). http://svn.aksw.org/papers/2013/ISWC_NIF/public.pdf
5. Kawamura, T., Ohsuga, A.: Toward an ecosystem of LOD in the field: LOD content generation and its consuming service. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part II. LNCS, vol. 7650, pp. 98–113. Springer, Heidelberg (2012). http://dblp.uni-trier.de/db/conf/semweb/iswc2012-2.html#KawamuraO12
6. Liu, H., Wu, S.T.I., Tao, C., Chute, C.G.: Modeling UIMA type system using web ontology language - towards interoperability among UIMA-based NLP tools. In: Proceedings of Workshop on Managing Interoperability and compleXity in Health Systems (MIX-HS), pp. 31–36 (2012). http://dblp.uni-trier.de/db/conf/cikm/mixhs2012.html#LiuWTC12

7. Oberkampf, H., Bretschneider, C., Zillner, S., Bauer, B., Hammon, M.: Knowledge-based extraction of measurement-entity relations from german radiology reports. In: IEEE International Conference on Healthcare Informatics (ICHI) (2013)
8. Oberkampf, H., Zillner, S., Bauer, B., Hammon, M.: An OGMS-based model for clinical information (MCI). In: Proceedings of International Conference on Biomedical Ontology, pp. 97–100 (2013). http://www2.unb.ca/csas/data/ws/icbo2013/papers/ec/icbo2013_submission_56.pdf
9. Radiological Society of North America: Radlex (2012). http://rsna.org/RadLex.aspx
10. Ramakrishnan, C., Kochut, K.J., Sheth, A.P.: A framework for schema-driven relationship discovery from unstructured text. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 583–596. Springer, Heidelberg (2006). http://dx.doi.org/10.1007/11926078_42
11. Rizzo, G., Troncy, R., Hellmann, S., Brümmer, M.: In: Workshop on Linked Data on the Web (LDOW), Lyon, France
12. Verspoor, K., Baumgartner Jr., W., Roeder, C., Hunter, L.: Abstracting the types away from a UIMA type system. From Form to Meaning: Processing Texts Automatically, 249–256 (2009)