The LODeXporter: Flexible Generation of Linked Open Data Triples from NLP Frameworks for Automatic Knowledge Base Construction

René Witte and Bahar Sateli
Semantic Software Lab
Concordia University, Montréal, QC, Canada
http://www.semanticsoftware.info

Abstract

We present LODeXporter, a novel approach for exporting Natural Language Processing (NLP) results to a graph-based knowledge base, following Linked Open Data (LOD) principles. The rules for transforming NLP entities into Resource Description Framework (RDF) triples are described in a custom mapping language, which is defined in RDF Schema (RDFS) itself, providing a separation of concerns between NLP pipeline engineering and knowledge base engineering. LODeXporter is available as an open source component for the GATE *(General Architecture for Text Engineering)* framework.

Keywords: Linked Open Data, Automatic Knowledge Base Construction, Semantic Web

1. Introduction

Knowledge bases in standardized graph-based formats have become increasingly important in modern applications, such as personal intelligent assistants. Due to the large amount of formalized knowledge required, automatic knowledge base (KB) construction is a crucial step that typically spans multiple information sources (such as structured databases, sensors, and documents). A significant amount of knowledge resides in natural language text and can be automatically extracted through existing or custom-made Natural Language Processing (NLP) components and pipelines. Yet, so far no generic solution existed for exporting the results of an NLP pipeline into a knowledge base.

Our contribution is a novel technique for converting the results of an NLP pipeline into an LOD-compliant KB, based on the Resource Description Format (RDF) (Cyganiak et al., 2014) and RDF Schema (RDFS) (Brickley and Guha, 2014) W3C standards. The concrete mapping of NLP results to entities in the knowledge base is defined through a mapping language, which is itself written in RDFS. This provides high flexibility, as the same NLP pipeline can drive the generation of triples in different KBs with different vocabularies. It also relieves the NLP engineer from dealing with the technical details of generating correct Uniform Resource Identifiers (URIs), thereby providing an agile solution for bringing NLP results onto the Linked Open Data (LOD) cloud (Heath and Bizer, 2011).

LODeXporter has been implemented based on the *General Architecture for Text Engineering* (GATE) framework (Cunningham et al., 2013) and is available as open source (LGPLv3) on GitHub.[1]

2. Foundations

In this section, we provide brief background information on the foundations of our approach.

Linked Data. Linked Open Data (LOD) (Heath and Bizer, 2011; Wood et al., 2014) is the practice of using semantic web technologies, like RDF, to publish structured, open datasets that can be interlinked and shared automatically between machines. All data within the LOD cloud have unique URIs. When dereferenced, the URIs provide useful, machine-readable information about that entity, as well as links to other related entities on the web of data.

Automatic Knowledge Base Construction. As semantic knowledge bases, triplestores are especially well suited for connecting previously isolated knowledge sources (so-called "information silos"). Towards this end, various solutions have been developed for converting existing sources to a knowledge base, e.g., RDB2RDF for creating RDF data from a relational database.[2] Natural language documents are an abundant source of rich semantic knowledge. One of the best known LOD datasets, *DBpedia*,[3] is a large knowledge base that is automatically generated from the *Wikipedia* (Bizer et al., 2009), with several billions of in-bound links from other open datasets. Similarly, custom solutions exist that convert the results of a specific NLP pipeline into a pre-defined knowledge base.

However, so far there existed no generic solution for populating a knowledge base with the results from a text mining pipeline. While it is always possible to develop custom export strategies, this is not a practicable solution in a modern agile data science workflow, where it often becomes necessary to experiment with different knowledge bases and representation vocabularies; typical examples are shared tasks, where the expected format of the knowledge base is prescribed and existing NLP pipelines need to be adapted to a challenge-specific format.

3. Design of LODeXporter

We now describe our design decisions behind LODeXporter in detail.

[1] LODeXporter on GitHub, https://github.com/SemanticSoftwareLab/TextMining-LODeXporter

[2] Relational Databases to RDF (RDB2RDF), https://www.w3.org/2001/sw/wiki/RDB2RDF

[3] DBpedia, http://wiki.dbpedia.org/

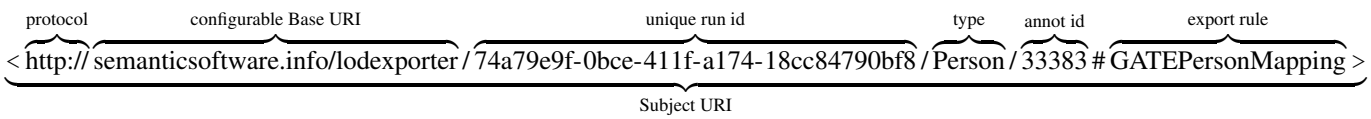


Figure 1: Anatomy of a LODeXporter-generated Subject URI

3.1. Requirements

Our high-level vision is the design of a new NLP component that can be added to any existing NLP pipeline, thereby providing functionality to export the text analysis results in linked data format. We derived a number of detailed requirements from this vision:

Scalability (R1). For the population of large knowledge bases, it must be possible to scale out the generation of triples, i.e., running pipelines on multiple (cloud) instances and storing the resulting triples in a networked triplestore. The time required to export the annotations as triples must scale linearly with the size of the documents and the number of triples to be exported.

Separation of Concerns (R2). Apart from adding a new component to an analysis pipeline, no further changes must be required on the NLP side. Thus, we separate the work of a language engineer, developing the NLP pipeline (e.g., to find domain-specific entities) from the work of a knowledge base engineer, who defines the structure of a concrete KB.

Configurability (R3). The export must be dynamically configurable, so that the same NLP pipeline can drive the generation of different triples in the same, or multiple different, knowledge bases. In particular, the vocabularies used in the mapping process must be easily changeable, to support experiments with different knowledge bases and different application scenarios.

LOD Best Practices (R4). The solution must conform to the relevant W3C standards on Linked (Open) Data. This includes the recommended format for the generated triples as well as the support of standard protocols for communicating with triplestores in a product-independent fashion.

3.2. Mapping NLP Annotations to Triples

A core idea of our approach is that the concrete mapping, from NLP results to knowledge base, is not part of the NLP pipeline itself: Rather, it is externalized in form of a set of *mapping rules* that are encoded in the knowledge base itself. In other words, only by dynamically connecting an NLP pipeline to a knowledge base is the concrete mapping effected: The same pipeline can so be used to export different results, using different mapping rules, to multiple knowledge bases. This is a key feature to enable agile data science workflows, where experiments with different formats can now be easily and transparently set up.

The mapping rules themselves are defined in form of RDF triples as well, based on our mapping vocabulary. We provide mappings for common NLP result types, in particular annotations, annotation features, and annotation relations. Additionally, a number of meta-features about the NLP pipeline itself can be defined for storage in a KB.

URI Generation. One of the core features of LODeXporter is the generation of URIs for LOD triples from text **(R4)**. In designing the URI generation scheme, we had to strike a balance between *(a)* conforming to LOD best practices; *(b)* taking into account the NLP source of the URIs; and their *(c)* usability, in particular for querying knowledge bases that mix NLP-generated knowledge with other sources. Figure 1 shows an example for a subject URI that was generated for a single *Person* instance in a document. Conforming to LOD best practices, URIs are HTTP-resolvable. A user-definable run-time parameter specifies the *base URI* (e.g., for a public SPARQL endpoint), which should always be a domain 'owned' by the user. This is followed by a *unique run id*, which ensures that each new run of LODeXporter on the same text generates new URIs. Generally, exposing implementation details in URIs is discouraged (Wood et al., 2014). However, re-running NLP pipelines (and therefore re-generating triples) is an extremely common occurrence in language engineering – for example, after improving a component, a machine learning model, or experimenting with different parameters. Hence, we decided that the triples from different runs of an NLP pipeline must be able to peacefully co-exist in a KB and be easily distinguishable, which is achieved with the generation of this *run id*. Next, the semantic (annotation) *type*, as detected by the NLP pipeline, is encoded (here "Person"), followed by its unique *annotation id* within a document. Finally, the mapping *export rule* name is added, as it is possible to export the same annotation (with the same type) multiple times, using different export rules (e.g., with different vocabularies).

3.3. Mapping Language

The central idea of our approach to automatic KB construction from NLP results is to *externalize* the knowledge about the export process, so that the NLP pipeline can remain unchanged (except for adding the LODeXporter component). This provides the required separation of concerns **(R2)** and makes it possible to easily reuse existing NLP pipelines for different knowledge bases. The export rules – how NLP results are mapped to triples – are themselves described in form of RDF triples using our new *mapping language*. Thus, the configuration how a specific knowledge base has to be populated with triples can be read from the same knowledge base, which provides for an introspective NLP export **(R3)**.

In detail, our mapping language provides constructs for mapping entities (subjects), features (properties and objects), as well as meta-information about the export (e.g., text offsets of entities or the name of the pipeline that generated an entity). A concrete mapping configuration is a set of rules (see Figure 2c for an example). These rules are read by LODeXporter, typically at the end of a pipeline, and triples

are then generated according to these rules. Hence, the exact same pipeline can be used to generate different LOD triples, e.g., when facts need to be expressed with different LOD vocabularies for different applications.

Mapping Entities. For each entity that needs to be exported, a corresponding mapping rule specifies the NLP type and its output RDF type. For example, a Person entity detected in a text can be mapped to a KB using the FOAF vocabulary.[4] To export all instances of a GATE annotation of type Person, detected in a document, to foaf:Person in a KB, this rule (in RDF/XML):[5]

```
<rdf:Description rdf:about="GATEPersonMapping">
  <rdf:type rdf:resource="map:Mapping"/>
  <map:baseURI rdf:resource=
      "http://semanticsoftware.info/lodexporter/"/>
  <map:type rdf:resource="foaf:Person"/>
  <map:GATEtype>Person</map:GATEtype>
</rdf:Description>
```

results in the generation of a triple with (1) a subject URI, as shown in Figure 1; (2) a property defining the rdf:type of that triple, and (3) the object URI with the defined type:

```
<SubjectURI>
    <http://www.w3.org/1999/02/22−rdf−syntax−ns#type>
    <http://xmlns.com/foaf/0.1/Person>
```

This mapping approach makes our solution extremely flexible: For example, to change the generation of triples to use the Person Core Vocabulary,[6] instead of FOAF, only a change of the mapping rule is required, which is read at export-time. Thus, the designer of a KB is free to experiment with multiple different knowledge representation approaches, without requiring any reconfiguration on the NLP side.

Mapping Features. Most entities are further described in an NLP process with additional features, e.g., the detected *gender* for a name. We can map any existing feature of an entity (one subject URI) to different properties and objects, with the same configuration approach. For example, to map a feature *gender* for a Person, we would define an additional mapping rule to transform it into foaf:gender:

```
<rdf:Description rdf:about="GATEFeatureMapping">
  <rdf:type rdf:resource="map:Mapping"/>
  <map:type rdf:resource="foaf:gender"/>
  <map:GATEfeature>gender</map:GATEfeature>
</rdf:Description>
```

This rule is then included in the first rule above with an additional triple:

```
<rdf:Description rdf:about="GATEPersonMapping">
  ...
  <map:hasMapping rdf:resource="GATEFeatureMapping"/>
</rdf:Description>
```

resulting in another output triple with the same subject URI:

```
<SubjectURI> <http://xmlns.com/foaf/0.1/gender> "male"
```

Additionally, we provide for the export of several meta-information of an annotation, such as the start- and end-offsets of the entity in a document, as well as the underlying string representation (surface form).

[4] Friend-of-a-Friend (FOAF), http://xmlns.com/foaf/spec/

[5] The prefix `map:` refers to our mapping language.

[6] Person Core Vocabulary, https://www.w3.org/ns/person

Mapping Relations. Additionally, we can export some pre-defined relations between entities into triples. In particular, we can map a *contains* relationship (based on text offsets) to any LOD vocabulary. An example is shown in Figure 2c, where we define that any *RhetoricalEntity* containing a *DBpediaNE* results in an additional triple with the two subject triples, linked by pubo:containsNE (see Figure 2b).

Finally, we provide for exporting a number of meta-information of an export process as relations, including: the name of the NLP pipeline that generated the annotations, creation timestamp, document name, corpus name, among others. These are important when the traceability of triples in a KB to their provenance is a concern.

4. Implementation

LODeXporter has been implemented in Java as an NLP component for the GATE *(General Architecture for Text Engineering)* framework (Cunningham et al., 2013). GATE is a mature, open source, component-based framework, which integrates with a number of well-known NLP libraries, such as Stanford CoreNLP, LingPipe, OpenNLP, and UIMA. This ensures LODeXporter can be used with a large variety of existing NLP applications.

The RDF(S) triples generated by LODeXporter follow the W3C standards and are not tied to any specific implementation. Internally, LODeXporter relies on the Apache Jena[7] libraries for generating the RDF graph. Results can be exported in two different modes **(R1)**.

Direct KB Export Mode: The first option is to directly connect the NLP framework with a triplestore. Running the NLP pipeline with LODeXporter will then directly populate the KB with the triples generated from a text. Here, we currently support the Apache TDB storage layer.[8]

File-Based Export Mode: The second option is to export the triples of a document into a file, based on the W3C standard N-Quads format.[9] These triples can then be imported into any triplestore. A typical application scenario in a distributed (e.g., cluster or cloud-based) environment is to upload the generated triples to a triplestore via the W3C SPARQL 1.1 Graph Store HTTP Protocol.[10] The files generated by LODeXporter can be used directly with an HTTP POST operation; we tested this extensively with the Apache Jena Fuseki SPARQL server[11] and its included *s-post* script.

The choice between the two export formats depends on the intended application scenario: In cases where a knowledge base is created in batch-mode, based on an existing corpus, the first solution avoids web protocol overhead, but currently

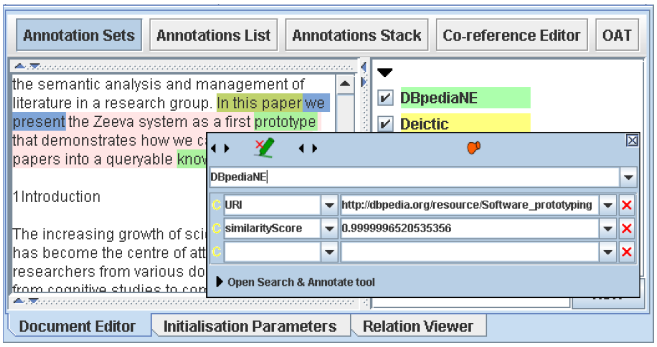
[7] Apache Jena, https://jena.apache.org/

[8] Apache TDB, https://jena.apache.org/documentation/tdb/index.html

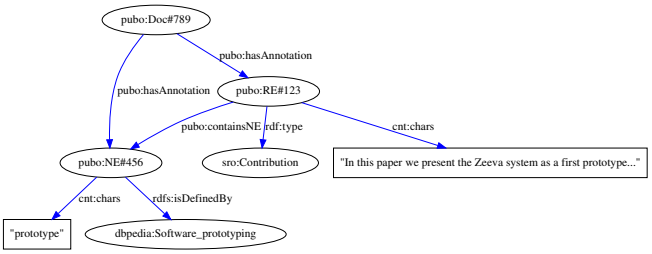
[9] W3C N-Quads, http://www.w3.org/TR/n-quads/

[10] SPARQL 1.1 Graph Store HTTP Protocol, http://www.w3.org/TR/sparql11-http-rdf-update/

[11] Apache Jena Fuseki, https://jena.apache.org/documentation/fuseki2/



(a) NLP annotations in GATE Developer



(b) LODeXporter output

```
@prefix map: <http://lod.semanticsoftware.info/mapping/mapping#> .
@prefix pubo: <http://lod.semanticsoftware.info/pubo/pubo#> .
@prefix rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf−schema#> .
@prefix cnt: <http://www.w3.org/2011/content#> .
@prefix sro: <http://salt.semanticauthoring.org/ontologies/sro#> .
@prefix ex: <http://example.com/> .

### Annotation Mapping ###
ex:GATERhetoricalEntity a map:Mapping ;
        map:type        sro:RhetoricalElement ;
        map:GATEtype    "RhetoricalEntity" ;
        map:hasMapping  ex:GATEContentMapping .

ex:GATEDBpediaNE a map:Mapping ;
        map:type        pubo:LinkedNamedEntity ;
        map:GATEtype    "DBpediaNE" ;
        map:hasMapping  ex:GATEContentMapping ;
        map:hasMapping  ex:GATELODRefFeatureMapping .

### Feature Mapping ###
ex:GATEContentMapping a map:Mapping ;
        map:type        cnt:chars ;
        map:GATEattribute "content" .
ex:LODRefFeatureMapping a map:Mapping ;
        map:type        rdfs:isDefinedBy ;
        map:GATEfeature "URI" .

### Relation Mapping ###
ex:RE_NE_RelationMapping a map:Mapping ;
        map:type        pubo:containsNE ;
        map:domain      ex:GATERhetoricalEntity ;
        map:range       ex:GATEDBpediaNE ;
        map:GATEattribute "contains"
```

(c) LODeXporter mapping rules example

Figure 2: Example RDF graph (b) generated by LODeXporter from NLP annotations (a) based on the mapping rules shown on the right (c) for a paper in a Semantic Publishing application

only works with TDB-based triple stores and is limited to parallelization within a single JVM (using the "GATE Cloud Parallelizer"[12] tool). The second approach is suitable for continuously updating a knowledge base, e.g., based on a stream of incoming documents, and can be easily distributed in a cloud infrastructure. It is also not limited to a specific knowledge base implementation, but requires an accessible SPARQL over HTTP interface.

5. Application

Figure 2 shows an example from a real-world application scenario in *Semantic Publishing* (Berners-Lee and Hendler, 2001), which aims at enriching scientific publications with machine-readable information in order to explicitly mark up experiments, data, and rhetorical elements in their raw text. While a number of RDFS and OWL ontologies are now available, like SALT (Groza et al., 2007) and the Semantic Publishing And Referencing (SPAR) ontology family (Peroni et al., 2012), a serious bottleneck to their adoption is the vast amount of existing publications, which would be too time-consuming to manually annotate.

Hence, NLP techniques play a crucial role in enabling Semantic Publishing workflows and applications. In (Sateli and Witte, 2015), we show how scientific literature can be transformed into a queryable knowledge base, through an NLP pipeline that detects rhetorical entities and domain concepts in their full-text (Figure 2). The input to our pipeline was a set of 136 open access computer science articles. We processed the full-text of each article to extract various rhetorical entities (e.g., claim or contribution sentences) and further linked each domain concept in the document to its corresponding resource on the LOD cloud. We used the LODeXporter component in our text mining pipeline to transform the detected entities (added to the document in form of GATE annotations) to RDF triples, based on the mapping rules shown in Figure 2c. The complete knowledge base contains 1,086,051 triples.[13]

A few example triples from the populated knowledge base are presented in Figure 3a, describing a contribution sentence found in a document, as well as a concept (named entity) mentioned in its text. Here, we can see the triple generated for the document and the PUBO vocabularies used to attach the triples describing the rhetorical and named entities to the document. Based on this schema, the knowledge base can then be queried, e.g., to find all documents with their contribution sentences and mentioned topics by using a SPARQL query, like the one shown in Figure 3b. The query shown here retrieves all documents that have an annotation of type `sro:Contribution` (sentence), as well as the named entities that are contained within the sentence boundary, returning *(a)* their surface form (as they appeared in the document), and *(*b) their corresponding resource from the DBpedia ontology. An example entry from the result set is illustrated in Figure 3c. Here, the sentence contains the term "ASR", which was grounded to the entry for 'Automatic Speech Recognition' in the DBpedia ontology, demonstrat-

[12] The GATE Cloud Paralleliser (GCP), https://gate.ac.uk/gcp/

[13] Available in N-Quads format at http://www.semanticsoftware.info/semantic-scientific-literature-peerj-2015-supplements

```
<http://example.com/Corpora/PeerJ-CompSci/cs-8.xml>
 <http://lod.semanticsoftware.info/pubo/pubo#hasAnnotation>
 <http://semanticsoftware.info/lodexporter/9182bfec-a88f-4b61-8bde-0bd6a670449b/RhetoricalEntity/97528#map:GATERhetoricalEntity>,
 <http://semanticsoftware.info/lodexporter/9182bfec-a88f-4b61-8bde-0bd6a670449b/DBpediaNE/114836#map:GATEDBpediaNE> .

<http://semanticsoftware.info/lodexporter/9182bfec-a88f-4b61-8bde-0bd6a670449b/RhetoricalEntity/97528#map:GATERhetoricalEntity>
 a <http://salt.semanticauthoring.org/ontologies/sro#RhetoricalElement> ,
   <http://salt.semanticauthoring.org/ontologies/sro#Contribution> ;
 <http://lod.semanticsoftware.info/pubo/pubo#containsNE>
   <http://semanticsoftware.info/lodexporter/9182bfec-a88f-4b61-8bde-0bd6a670449b/DBpediaNE/114836#map:GATEDBpediaNE> ;
 <http://www.w3.org/2011/content#chars>
  "We evaluated the system performance using this ASR implementation."^^<http://www.w3.org/2001/XMLSchema#string> .

<http://semanticsoftware.info/lodexporter/9182bfec-a88f-4b61-8bde-0bd6a670449b/DBpediaNE/114836#map:GATEDBpediaNE>
 a <http://lod.semanticsoftware.info/pubo/pubo#LinkedNamedEntity> ;
 <http://www.w3.org/2000/01/rdf-schema#isDefinedBy> <http://dbpedia.org/resource/Speech_recognition> ;
 <http://www.w3.org/2011/content#chars>"ASR"^^<http://www.w3.org/2001/XMLSchema#string> .
```

(a) Examples triples in N-Quads format from the knowledge base

```
PREFIX pubo: <http://lod.semanticsoftware.info/pubo/pubo#>
PREFIX sro: <http://salt.semanticauthoring.org/ontologies/sro#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX cnt: <http://www.w3.org/2011/content#>

SELECT DISTINCT ?document ?sentence ?topic ?uri WHERE {
      ?document pubo:hasAnnotation ?rhetoricalEntity .
      ? rhetoricalEntity  cnt:chars  ?sentence .
      ? rhetoricalEntity  rdf:type  sro:Contribution  .
      ? rhetoricalEntity  pubo:containsNE ?namedEntity.
      ?namedEntity rdfs:isDefinedBy ?uri .
      ?namedEntity cnt:chars ?topic .
} ORDER BY ?paper
```

(b) A SPARQL query to find contribution sentences and their domain concepts

Document	Sentence	Topic	URI
cs-8.xml	"We evaluated the system performance using this ASR implementation."	"ASR"	dbpedia:Speech_recognition

(c) One example result from the knowledge base

Figure 3: Example triples (a) generated by LODeXporter in a semantic publishing pipeline, the corresponding SPARQL query (b) on the knowledge base, and the resulting table (c) on the bottom.

ing the power of connecting the NLP results in a knowledge base with external open datasets.

6. Related Work

We previously presented *OwlExporter* (Witte et al., 2010) at LREC 2010, an NLP component for ontology population from text. In contrast with LODeXporter, OwlExporter is focused on the population of Web Ontology Language (OWL) ontologies,[14] with Description Logic (DL) reasoning as their primary application. In some respect, LODeXporter is our completely re-designed approach for knowledge export from NLP pipelines, based on the experience we gained from building numerous knowledge-intensive applications. In particular, with LODeXporter we focus on scenarios that were less common when we first designed OwlExporter, more than 15 years ago: Large-scale knowledge bases, generated in distributed cloud-based environments, linked data datasets and applications (based on the LOD initiative started in 2007), shared open vocabularies, as well as the proliferation of AI applications, such as intelligent assistants, where NLP techniques form just one of numerous knowledge sources and algorithms. While OwlExporter relied on special NLP annotations driving the ontology population, with LODeXporter we completely externalized the mapping rules, so that the same NLP pipeline can now generate different triples, based on different vocabularies, when connected to various knowledge bases.

NLP2RDF[15] is another approach for converting NLP tool output to RDF (Hellmann et al., 2013). However, the goals of NLP2RDF are largely orthogonal to the ones from LODeXporter: NLP2RDF focuses on the representation of *NLP data* (such as, words, sentences, strings), with the primary goal of providing an exchange format between different NLP tools. All knowledge is represented through its own NIF (NLP Interchange Format) vocabulary.[16] In contrast, LODeXporter focuses on the representation of *domain data* (e.g., biological entities, company intelligence, financial data), which is represented in domain-specific vocabularies. The resulting knowledge bases are primarily meant for creating LOD datasets for publication and/or building intelligent applications on top of them. Hence, although LODeXporter and NLP2RDF both generate RDF from NLP frameworks, they are two quite distinct solutions for different application use cases.

[14] Web Ontology Language (OWL), https://www.w3.org/OWL/

[15] NLP2RDF, see https://site.nlp2rdf.org/

[16] NIF 2.0 Core Ontology, see http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core

7. Conclusion

We presented LODeXporter, a novel approach for the generation of linked open data (LOD) compliant triples from text analysis pipelines. LODeXporter provides for a separation of concerns, as the NLP pipeline (developed by a language engineer) is strictly separated from the export of the NLP results into RDF(S) triples (designed by a knowledge engineer). We achieve this separation through a new mapping language, which is defined itself in RDFS, and defines the rules for converting NLP annotations and features into *(subject, property, object)* triples. This approach supports agile data science workflows (Sateli and Witte, 2016), where existing NLP pipelines can be easily connected with different web vocabularies, for example, to facilitate submissions to shared tasks within competitions.

LODeXporter enables NLP framework users to easily generate knowledge bases in an LOD-compliant format, which can then be shared on the linked open data (LOD) cloud. It also facilitates the development of complex AI applications, such as intelligent assistants, which typically rely on numerous knowledge sources, including text analysis results.

8. Acknowledgements

This work was partially funded by a Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant (DG).

9. Bibliographical References

Berners-Lee, T. and Hendler, J. (2001). Publishing on the semantic web. *Nature*, 410(6832):1023–1024.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. (2009). DBpedia—A crystallization point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165.

Brickley, D. and Guha, R. (2014). RDF Schema 1.1. W3C Recommendation, https://www.w3.org/TR/rdf-schema/.

Cunningham, H., Tablan, V., Roberts, A., and Bontcheva, K. (2013). Getting more out of biomedical documents with GATE's full lifecycle open source text analytics. *PLoS computational biology*, 9(2):e1002854.

Cyganiak, R., Wood, D., and Lanthaler, M. (2014). RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, https://www.w3.org/TR/rdf11-concepts/.

Groza, T., Handschuh, S., Möller, K., and Decker, S. (2007). SALT – Semantically Annotated LaTeX for Scientific Publications. In *The Semantic Web: Research and Applications*, LNCS, pages 518–532. Springer.

Heath, T. and Bizer, C. (2011). *Linked Data: Evolving the Web into a Global Data Space*. Synthesis lectures on the semantic web: theory and technology. Morgan & Claypool Publishers.

Hellmann, S., Lehmann, J., Auer, S., and Brümmer, M. (2013). Integrating NLP Using Linked Data. In *Proceedings of the 12th International Semantic Web Conference - Part II*, ISWC '13, pages 98–113. Springer-Verlag New York, Inc.

Peroni, S., Shotton, D., and Vitali, F. (2012). Scholarly publishing and linked data: describing roles, statuses, temporal and contextual extents. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 9–16. ACM.

Sateli, B. and Witte, R. (2015). Semantic representation of scientific literature: bringing claims, contributions and named entities onto the Linked Open Data cloud. *PeerJ Computer Science*, 1(e37), 12/2015.

Sateli, B. and Witte, R. (2016). From Papers to Triples: An Open Source Workflow for Semantic Publishing Experiments. In *Semantics, Analytics, Visualisation: Enhancing Scholarly Data (SAVE-SD 2016)*, Montréal, QC, Canada, 04/2016. Springer.

Witte, R., Khamis, N., and Rilling, J. (2010). Flexible Ontology Population from Text: The OwlExporter. In *International Conference on Language Resources and Evaluation (LREC)*, pages 3845–3850, Valletta, Malta, May 19–21. ELRA.

Wood, D., Zaidman, M., Ruth, L., and Hausenblas, M. (2014). *Linked Data: Structured data on the Web*. Manning Publications Co.