Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 7212–7221
Marseille, 11–16 May 2020

Fintan - Flexible, Integrated Transformation and Annotation eNgineering

Christian Fäth, Christian Chiarcos, Björn Ebbrecht, Maxim Ionov
Applied Computational Linguistics Lab (ACoLi)
Goethe University Frankfurt, Germany
{faeth, chiarcos, ebbrecht, ionov}@informatik.uni-frankfurt.de

Abstract

We introduce the Flexible and Integrated Transformation and Annotation eNgeneering (Fintan) platform for converting heterogeneous linguistic resources to RDF. With its modular architecture, workflow management and visualization features, Fintan facilitates the development of complex transformation pipelines by integrating generic RDF converters and augmenting them with extended graph processing capabilities:

Existing converters can be easily deployed to the system by means of an ontological data structure which renders their properties and the dependencies between transformation steps. Development of subsequent graph transformation steps for resource transformation, annotation engineering or entity linking is further facilitated by a novel visual rendering of SPARQL queries. A graphical workflow manager allows to easily manage the converter modules and combine them to new transformation pipelines.

Employing the stream-based graph processing approach first implemented with CoNLL-RDF, we address common challenges and scalability issues when transforming resources and showcase the performance of Fintan by means of a purely graph-based transformation of the Universal Morphology data to RDF.

Keywords: data formats, interoperability, RDF, graphs, linguistic annotation, tab-separated values, universal morphology, CoNLL-RDF

1. Motivation

In recent years, the focus in the development of Natural Language Processing (NLP) applications has constantly shifted from algorithmic, rule-based strategies towards more data-driven approaches. This includes not only statistical methods but also neural networks and deep learning. However, all these methods heavily rely on the amount of available language resources and their data structures. The EU-funded Prêt-à-LLOD project[1] aims at tackling the crucial challenges of discovering, linking and transforming language resources and making them available as interoperable Linked Data (LD) using well-established RDF-based[2] formats like OntoLex-Lemon (Cimiano et al., 2016), CoNLL-RDF[3] (Chiarcos and Fäth, 2017; Chiarcos and Schenk, 2018) or the NLP Interchance Format (Hellmann et al., 2012, NIF).

While resource discovery, maintenance and licensing is covered by other parts of the project, in this paper we focus on the Flexible and Integrated Transformation and Annotation eNgineering (Fintan) platform which provides a workflow-driven, modular approach to graph-transformation while still enabling the usage of well-established resource-specific converters.

2. Challenges in Linguistic Linked Data

2.1. Heterogeneity of Linguistic Resources

Since language resources on the web are highly heterogeneous and tailored towards specific use cases, they most often cannot easily be adapted to other NLP applications beyond their intended scope. Corpora may often be shared in specific CoNLL dialects, (Standoff-)XML, TEI[4] or proprietary, combined formats (e.g. the Sketch Engine vertical format)[5]. The same amount of diversity applies to dictionaries.

Addressing the issues of heterogeneity in linguistic data, an increasing amount of effort is being invested in the establishment of interoperable and extendable standards and data formats for publishing and distributing linguistic resources as linked data. The fruit of these efforts is the Linguistic Linked Open Data (LLOD) cloud [6].

In recent years, an ever increasing amount of corpora and dictionaries is becoming available as CoNLL-RDF, NIF or OntoLex-Lemon. Knowledge bases and terminological repositories have also seen an increasing level of adaption to linked data formalisms. The Ontologies of Linguistic Annotation (Chiarcos and Sukhareva, 2015, OLiA) include OWL[7] renderings of various linguistic annotation schemes which are connected through a common reference model. Furthermore, the DBpedia (Auer et al., 2007) derived from Wikipedia is widely used for Named Entity Recognition and Entity Linking applications, e.g. for "wikification" in Abstract Meaning Representations (Pan et al., 2015, AMR). However, while ontologies and terminologies are most often manually written and maintained, dictionaries and corpora are mostly transformed by isolated converters considering format- or even resource-specific peculiarities. The

[1] https://www.pret-a-llod.eu

[2] The Resource Description Framework is the W3C standard for rendering data in the Semantic Web (McBride, 2004).

[3] CoNLL-RDF is an isomorphic rendering of the CoNLL format family in RDF. CoNLL formats are task-specific TSV representations used in CoNLL shared tasks which have become a de-facto standard for NLP tools developed in their context, cf. http:/www.signll.org/conll

[4] Guidelines for Text Encoding by the Text Encoding Initiative, cf. https://tei-c.org/

[5] https://www.sketchengine.eu/my_keywords/vertical/

[6] http://linguistic-lod.org/

[7] The Web Ontology Language (OWL) is a W3C recommendation for rendering ontologies in the Semantic Web (McGuinness et al., 2004)

output of these converters may therefore include unique design decisions and is confined to the modeling constraints at the time of their writing, thus undermining interoperability.[8] In the end, the number of existing converters (not only including LD) with varying scope, completeness or reusability may well exceed the number of data formats they convert.

While graph based formalisms in general do have lots of advantages and surveys (Nastase et al., 2015) are conducted on the possibilities of rendering lingusitic data as graphs, most of the graph based approaches which are actually implemented in the context of NLP are limited to knowledge graphs, eg. for semantic parsing (Yih et al., 2015).

For most cases however, from the perspective of the NLP and Machine Learning communities, all these RDF or graph representations just display one more format to be dealt with before relevant data can be used in respective pipelines. Therefore, each data source tends to be handled by a task-specific "python wrapper"[9] and then injected to the actual NLP workflow.

2.2. Graph Mining and Transformation

One of the primary reasons for this sentiment towards linked data formalisms (among other efforts of standardization) may be lying within the structural nature of RDF. Since LD-resources represent a decentrally organized, interconnected set of graphs, they are usually published either on public endpoints or most commonly as RDF/XML, N3, or Turtle dumps[10].

In most cases, the endpoints (if they even exist) impose very harsh constraints on the available subset of SPARQL query commands (Buil Aranda et al., 2013) in order to maintain stability and avoid denial-of-service attacks. For the same reasons, federated queries, one of the main selling points of RDF during the time of its creation, are facing similar restrictions or are completely deactivated.

These issues when actually trying to query existing resources, most often only leave the option of downloading the full dumps and working offline. However, RDF still being a graph, these serializations are not necessarily well-structured. Bits of information on a specific entity may be scattered across gigabytes of data.

The only truly viable option in this case would be to set up a local graph database and extract the necessary pieces of information from there. The most obvious drawback of this approach is that it is hard to integrate into a full pipeline. It not only requires the use of database engines and respective APIs, but also may consume vast amounts of memory and processing power, especially when it comes to open source solutions. There had been some efforts for GPU-accelerated SPARQL by Blazegraph[11] but the implementation has never made it to the open source version.

2.3. CoNLL-RDF

In recent years, with the implementation of CoNLL-RDF (Chiarcos and Fäth, 2017), we have been tackling these constraints on resource consumption and pipeline interoperability, at least for corpora. The specifications of CoNLL-RDF as a lightweight, isomorphic rendering of tabular CoNLL formats also introduced a fully stream-based processing paradigm for linked data. Naturally, with CoNLL corpora being split into sentences and tokens in sequential order, it is fairly simple to read and transform them on-the-fly. The CoNLL-RDF canonical format imitates this structure as an ordered Turtle (Beckett et al., 2014) serialization (see below).

```
:s3_0 nif:nextSentence :s4_0 .
:s4_0 a nif:Sentence .
:s4_1 a nif:Word; conll:WORD 'I';   conll:HEAD :s4_2; ...
:s4_2 a nif:Word; conll:WORD 'see'; conll:HEAD :s4_0; ...
:s4_3 a nif:Word; conll:WORD 'you'; conll:HEAD :s4_2; ...
:s4_4 a nif:Word; conll:WORD '.';   conll:HEAD :s4_0; ...
```

This not only enables NLP pipelines to directly read the data as a text stream, but also allows sequential graph transformation using SPARQL. In the most recent version, multiple threads each read and transform one specific sentence at the same time. The fully sequential output stream may either provide RDF or TSV data to subsequent pipelines.

3. The Fintan Platform

Keeping in mind the aforementioned typical problems when working with language resources, we began to develop the Flexible and Integrated Transformation and Annotation eNgineering (Fintan) platform. The goals of Fintan can be summed up as follows:

- Convert to and from important LD standards and formats.
- Support as many existing formats as possible (both LD and non-LD)
- Allow integration with complex NLP pipelines
- Enable stream-based graph transformation for non-linear, non-TSV resources, wherever possible by maintaining the advantages in performance and scalability.

3.1. Fintan Architecture

To address these goals, the Fintan platform is designed to be highly modular and configurable (cf. Fig. 1). At its core, the graph processing module allows to create complex graph transformation pipelines using SPARQL update scripts and RDF resources. An integrated development frontend alleviates the creation and management of complex hierarchies of SPARQL updates. In order to gain a large coverage of input formats, Fintan allows existing RDF converters to be integrated as additional processing modules. The processing modules, graph transformation scripts

[8] For instance, many OntoLex-Lemon dictionaries available today still use the old monnet-lemon data model.

[9] A figurative notion, borrowed from what Mark Johnson stated in the ACL-IJCLNP 2012 keynote on the future of computational linguistics: "Standard data formats (...) I'm not sure these are important: if someone can use a parser, they can probably also write a Python wrapper" (Johnson, 2012, slide 8)

[10] For an overview of RDF serializations, cf. http://www.w3.org/standards/techs/rdf

[11] The original press note was released in 2015: https://blazegraph.com/press/gpu-launch-2015-12-15

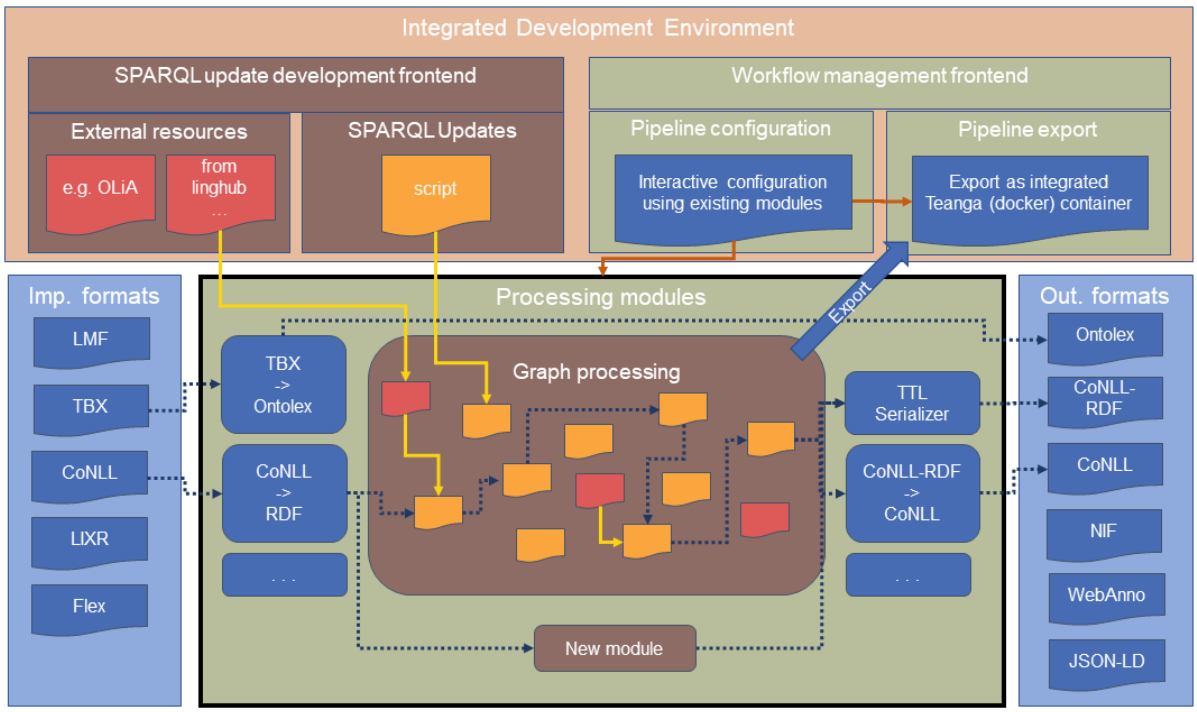

Figure 1: Fintan architecture

and their dependencies are described using the Fintan ontology. A workflow management frontend allows the development of complex converter pipelines. Fully integrated pipelines may be exported as Docker[12] containers.

3.2. Fintan Ontology

The Fintan ontology (Fig. 2) is designed to host information about processing modules and transformation scripts. Each module is described as an instance of a `Transformer`. `Loader` instances may read any type of input data and output RDF data as a stream. `Splitter` instances consume (mostly RDF) data, reorganize the serialization and output a segmented RDF stream. Loaders are encouraged to also have the characteristics of a Splitter and should thus be instances of both classes (e.g. CoNLL2RDF reads CoNLL data and produces a CoNLL-RDF stream split by sentences). `Update` modules refer to sets of SPARQL updates which perform specific transformation tasks. They are intended to consume data produced by Splitters and apply updates to each segment of data independently. Finally, `Writer` modules consume a stream of RDF data and create a serialized output file.
Several object properties define the requirements and assets of Transformer instances. This includes processed formats, required URIs or named graphs containing the data the Transformer needs to work with. Dependencies between Transformer instances within a pipeline are defined by the `enables` and `requires` properties.

3.3. Preparing Data for Graph Transformation

In Fintan we differentiate three types of data which may be used within a transformation pipeline. Their characteristics are listed below:

[12] `https://www.docker.com/`

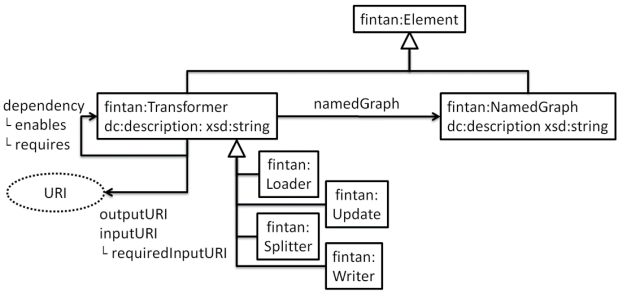

Figure 2: Fintan ontology

- *Processable input data* may be provided in any format for which a compatible `Loader` and `Splitter` combination is available. The data is streamed through `Transformer` modules and constantly rewritten.
- *External data* may also be provided in any format for which a compatible `Loader` is available. The main difference is, that this data is not consumed and transformed in the process. Instead it will be permanently stored read-only and made available for querying during the transformation process.
- *Scripts* are SPARQL queries and updates used to define specific transformation steps. They can be provided to the platform as either `Splitter` or `Update` modules to be used across various transformation workflows. Compatibility with specific pipelines can be inferred by the properties in the Fintan ontology.

This data-driven approach bears multiple advantages. When transforming larger datasets, streaming the input data reduces the bottleneck of both processing complexity and

memory consumption within the database engine. In addition, external repositories can still be "side-loaded" to a local triple store. They can be addressed as named graphs within an update script that is currently performed on the input data. Since they are read-only, no transaction penalties apply to these resources, allowing simultaneous access by multiple Transformer modules and threads.
For cases requiring lightweight datasets (esp. terminological repositories or ontologies used for inferences or transforming annotations), these can be stored in-memory. We furthermore support TDB for large scale datasets[13]. Optionally, federated queries can be used for directly extracting data from SPARQL endpoints which may be the preferred way of accessing very large datasets (e.g. DBPedia).

3.4. Parallelized Stream Processing

The core component of Fintan is a graph processing module which reads RDF input data as a stream, transforms it on multiple parallel threads and writes the resulting graph on an output stream. With its basic implementation originating from CoNLL-RDF, it takes advantage of the sequential sentence structure inherent to corpora. Since in CoNLL each sentence can be seen as an self-contained whole for most cases (excluding cross-sentence relations like coherence or coreference, which can be treated differently in Fintan), it is possible to reduce the processing complexity of SPARQL updates by executing them on each sentence independently.

3.4.1. Segmentation of Heterogeneous Data

Surely, this method cannot be as easily applied to other types of data (e.g. dictionaries). Therefore, dedicated Splitter modules are necessary to enable the same benefits. For dealing with unordered serializations of extensive RDF datasets, which in some cases may only be partly segmented due to a large amount of interconnection between distinct segments, Splitter modules can be designed as SPARQL `CONSTRUCT` or `DESCRIBE` queries for extracting subgraphs from the dataset.
In the context of OntoLex-Lemon, a feasible approach at segmentation would be to first extract the URIs of all instances of `ontolex:LexicalEntry` and to then iterate over them with the following `CONSTRUCT` statement:

```
PREFIX ontolex: <http://www.w3.org/ns/lemon/ontolex#>
PREFIX lexinfo: <http://www.lexinfo.net/.../lexinfo#>

CONSTRUCT {
    <?entryURI>  a             ontolex:LexicalEntry ;
                 ?lexForm      ?Form ;
                 ?sense        ?LexicalSense ;
                 lexinfo:partOfSpeech ?pos .
} WHERE {
    <?entryURI>  a             ontolex:LexicalEntry ;
                 ?lexForm      ?Form ;
                 ?sense        ?LexicalSense ;
                 lexinfo:partOfSpeech ?pos .

    GRAPH <http://www.w3.org/ns/lemon/ontolex> {
      ?lexForm rdfs:subPropertyOf* ontolex:lexicalForm .
      ?sense   rdfs:subPropertyOf* ontolex:sense .
    }
}
```

[13]Since Fintan is grounded in CoNLL-RDF our baseline implementation employs the Apache Jena API for graph processing. This includes TDB and TDB2 for hard disk assisted storage. For further information cf. `https://jena.apache.org/`

Within SPARQL, explicit URIs are denoted by angle brackets and variables start with question marks. In the context of Fintan, a string within angle brackts, which also starts with a question mark, is treated as a wildcard and is replaced by explicit URIs during runtime (in our case, the entry URI extracted in the first step). For including subproperties and subclasses for forms and senses, the ontolex core model needs to be loaded as *external data* (cf. Sect. 3.3.)
As a result, a set of subgraphs, each containing only a single `LexicalEntry` together with all its `Forms` and `LexicalSenses` is extracted and can now be used for parallelized graph transformation. Since it only contains a fraction of the data within a typical dictionary, subsequent transformation steps can be performed with a significantly reduced overhead regarding memory and processing power.

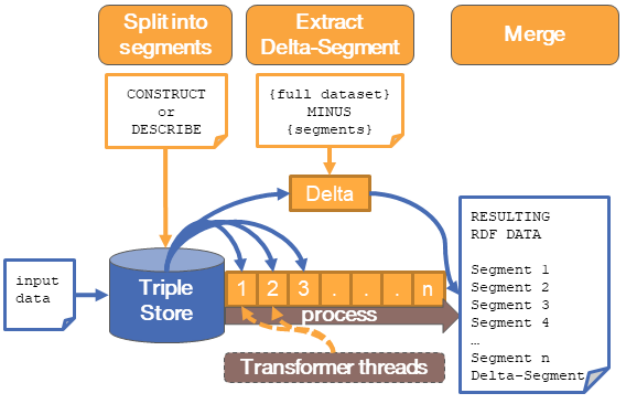

Figure 3: Fintan data segementation

However, even after all resulting subgraphs have been processed and merged again, a significant amount of triples will be lost. In our simplified example, this would even include the mandatory Literals addressed by the property `ontolex:writtenRep`. For this reason, a Fintan splitter always keeps track of the triples which have not yet been extracted. These will be added to the end of the resulting serialization as an unprocessed delta segment. Fig. 3 depicts the full workflow for data segmentation within Fintan.

3.4.2. Graph Transformation

During the actual graph transformation process a set of SPARQL updates is performed on one thread per segment. Since data segments are independent from each other, no write locks can occur and multiple segments can be transformed at the same time, further speeding up the process.
Similar to the CONSTRUCT queries used for the segmentation process, external models can be permanently preloaded and addressed from all threads during runtime. Since they are stored read-only, no write penalty applies.
The following example query depicts a simple transformation step which employs OLiA to infer LexInfo[14] (Cimiano et al., 2011) part of speech tags in any converted dictionary which still contains literals from its original tagset:

[14]LexInfo is the recommended vocabulary for describing linguistic annotations in the context of OntoLex-Lemon, cf. `https://www.lexinfo.net/`

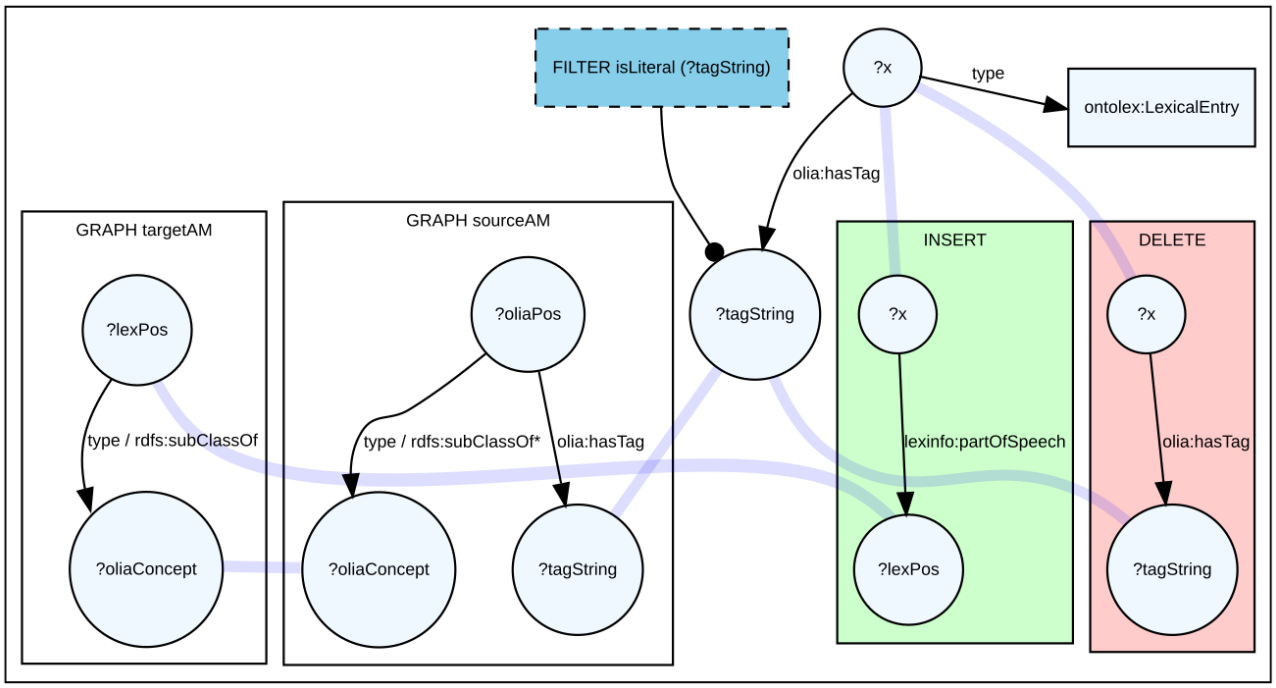

Figure 4: Visualization of a SPARQL update for annotation transformation

```
PREFIX ontolex: <http://www.w3.org/ns/lemon/ontolex#>
PREFIX lexinfo: <http://www.lexinfo.net/.../lexinfo#>
PREFIX olia: <http://purl.org/olia/system.owl#>

DELETE {
     ?x <?posProperty> ?tagString .
} INSERT {
     ?x lexinfo:partOfSpeech ?lexPos .
} WHERE {
     ?x a ontolex:LexicalEntry .
     ?x <?posProperty> ?tagString .
     FILTER(isLiteral(?tagString)) .
     # find OLiA concept
     GRAPH <?linkingModel> {
       ?oliaPos olia:hasTag ?tagString .
       ?oliaPos a/rdfs:subClassOf* ?oliaConcept .
     }
     # find closest match in lexinfo.
     GRAPH <?lexinfoLinkingModel> {
       ?lexPos a/rdfs:subClassOf ?oliaConcept .
     }
}
```

Three parameters have to be provided at runtime: the name of the property including the original tag string and two OLiA linking models (one for LexInfo and one for the source tagset) need to be loaded as external data.

For processing sequential data such as corpora, the lookahead and lookback capabilities introduced with CoNLL-RDF are available in Fintan as well. With this functionality, a specified number of preceding or subsequent data segments are made available read-only. This allows to establish cross-sentence relations such as coreference.

3.5. Development of Transformation Steps

In order to further alleviate creating transformation steps and reusing them across pipelines, the Fintan platform also integrates a visualization tool for SPARQL queries based on a prototype developed by Ebbrecht (2019). Based on Graphviz (Ellson et al., 2003) and its inherent DOT format, the visualization is created in a multi-step conversion. First, a SPARQL statement is parsed and verified with Apache Jena. Components of the statement such as subgraphs and triples are isolated and transformed to a visual representation to be included into a DOT file which is subsequently rendered as an SVG.

The visualization adheres to common design principles for visualizing RDF data by rectangular and circular nodes connected by labeled directed arrow shaped edges (as can be seen with ontology-visualization, EasyRdf Converter or gruff)[15]. Property names and property paths are directly attached to the edges. In our case, variable nodes are rendered as circles while explicit nodes are displayed as rectangles (literals are enclosed in quotes). This only slightly deviates from the most common choice where only literals are placed in rectangles. This decision was made in order to visually account for the importance of variables within SPARQL. Projected variables (placed in a `SELCET` statement) are marked with double edged borders. In addition, all subgraphs addressed within a SPARQL statement are rendered as isolated rectangular boxes. Nodes which occur within multiple subgraphs are connected by broad, semi-transparent lines. Displaying identical nodes separately in each subgraph makes it easier to conceive the subgraph as a whole while the shallow, rounded connectors depict a visually distinct layer of interconnection.

Filters are not parsed or visualized any further, but rather placed in dashed rectangle nodes as explicit statements. Subgraphs within Filters are unsupported at the moment and will also be displayed as text. However, the Filter node is attached to all variables addressed within. This makes it easy to see which parts of the graph are affected by it.

We currently support SPARQL queries with `SELECT` and `CONSTRUCT`, as well as updates with `INSERT` and

[15] A short, subjective overview is given on the W3C websites: `https://www.w3.org/2018/09/rdf-data-viz/`

DELETE. Figure 4 shows the visualization for the SPARQL update for transforming part of speech tags to LexInfo as described in section 3.4.2..

3.6. Workflow Management

For the workflow management we build on the user interface developed for Teanga (Ziad et al., 2018) which is also part of the Prêt-à-LLOD project. By rendering the transformer modules described in the Fintan ontology as nodes with their respective constraints on input and output data, the workflow manager will enable users to visually create complex transformation pipelines and assess their compatibility. Figure 5 shows an exemplary pipeline which was used for the case study described in section 4.

4. Case Study: Universal Morphology

In order to test the capabilities of our stream-based graph transformation we conducted a small case study which builds on an earlier effort to transform the Universal Morphology (Sylak-Glassman et al., 2015, UniMorph) to OntoLex-Lemon, publicly available as part of the LLODifier[16] toolset.

UniMorph describes inflected words of a given language together with their lexical meaning (as lemma) and a set of morphological features from the UniMorph annotation schema. The data in TSV format, together with licence information is available on the project's github page[17]. The following example shows an excerpt of the inflectional forms of the Albanian lemma *akrep*:

```
akrep akrepin N;ACC;SG;DEF
akrep akrepi N;NOM;SG;DEF
akrep akrepit N;ABL;SG;DEF
akrep akrepit N;DAT;SG;DEF
akrep akrepit N;GEN;SG;DEF
```

Because of the simple TSV layout of the UniMorph data, in our transformation pipeline we employed the CoNLL-RDF library to first convert the TSV data to the corpus-based CoNLL-RDF format and then performed all subsequent steps using the graph transformation module.

As depicted in Figure 5, two subsequent steps were performed: The replacement of corpus specific data structures inherent to CoNLL-RDF by an OntoLex-Lemon vocabulary could be acheived without accessing additional resources. Afterwards, employing OLiA's UniMorph annotation model by loading it as an external resource, the morphological features could be rendered as concepts linked to the OLiA reference model.

The resulting data renders each inflectional form of a word as `ontolex:LexicalEntry` together with a canonical form. The lexical forms encapsulate all possible lemma and feature combinations which can be represented by this lexical entry. The example below shows the converted ontolex representation for the Albanian inflectional form *akrepit*:

```
@prefix ontolex: <https://www.w3.org/ns/lemon/ontolex#> .
@prefix olia: <http://purl.org/olia/unimorph.owl#> .
@prefix : <https://github.com/unimorph/sqi/> .

:akrepit
    a ontolex:LexicalEntry ;
    ontolex:canonicalForm
        [ ontolex:writtenRep "akrepit" ] ;
    ontolex:lexicalForm
        :s1_1296 , :s1_1298 , :s1_1297 .

:s1_1296  a ontolex:Form ;
        ontolex:writtenRep  "akrep" ;
        olia:hasFeature olia:DEF ,
                                olia:SG ,
                                olia:N ,
                                olia:ABL .
```

In order to test the perfomance of the stream based graph transformation, the pipeline was executed using three distinct variants:

1. *en-bloc:* the whole dataset treated as one "sentence", thus applying the updates on the full dataset (similar to updating it on a Fuseki Server).

2. *serialized:* each lexical entry treated as one "sentence". Updates performed on a single thread.

3. *parallelized:* each lexical entry treated as one "sentence". Updates performed on four threads (one thread per processor core).

The en-bloc variant is the most basic way to convert UniMorph to CoNLL-RDF since UniMorph, not being a corpus, does not contain sentence boarders and thus no data segmentation is available. This leads to a significant scalability problem: larger wordlists would have to be loaded to memory in full and transformation would have to be executed on the whole dataset. In order to make this approach scalable, a triple store (e.g. TDB) would be needed as backend. Since the Albanian dataset is not very large, it still fit into Apache Jena's in-memory datasets.

The serialized method, in contrast, splits the whole dataset into single entries for each of which the updates are performed independently. In terms of the Fintan ontology it acts as a combined Loader and Splitter. This eliminates the scalability issues but still introduces a significant processing overhead, especially since in the original CoNLL-RDF implementation, external datasets would have to be loaded for each entry independently.

The parallelized method is the one implemented for Fintan. It also splits the data into single entries, but precaches the external dataset, so it only needs to be loaded once (reducing its processing overhead to approximately the same level as the en-bloc variant). Additionally, the processing of single entries is distributed across available threads.

	en-bloc	serialized	parallelized
Elapsed time	6m24s	12m34s	3m00s
OLiA loads	1	33483	1

Table 1: Results of the transformation approaches

[16]The LLODifier tools are available at `https://github.com/acoli-repo/LLODifier/`

[17]`https://unimorph.github.io/`

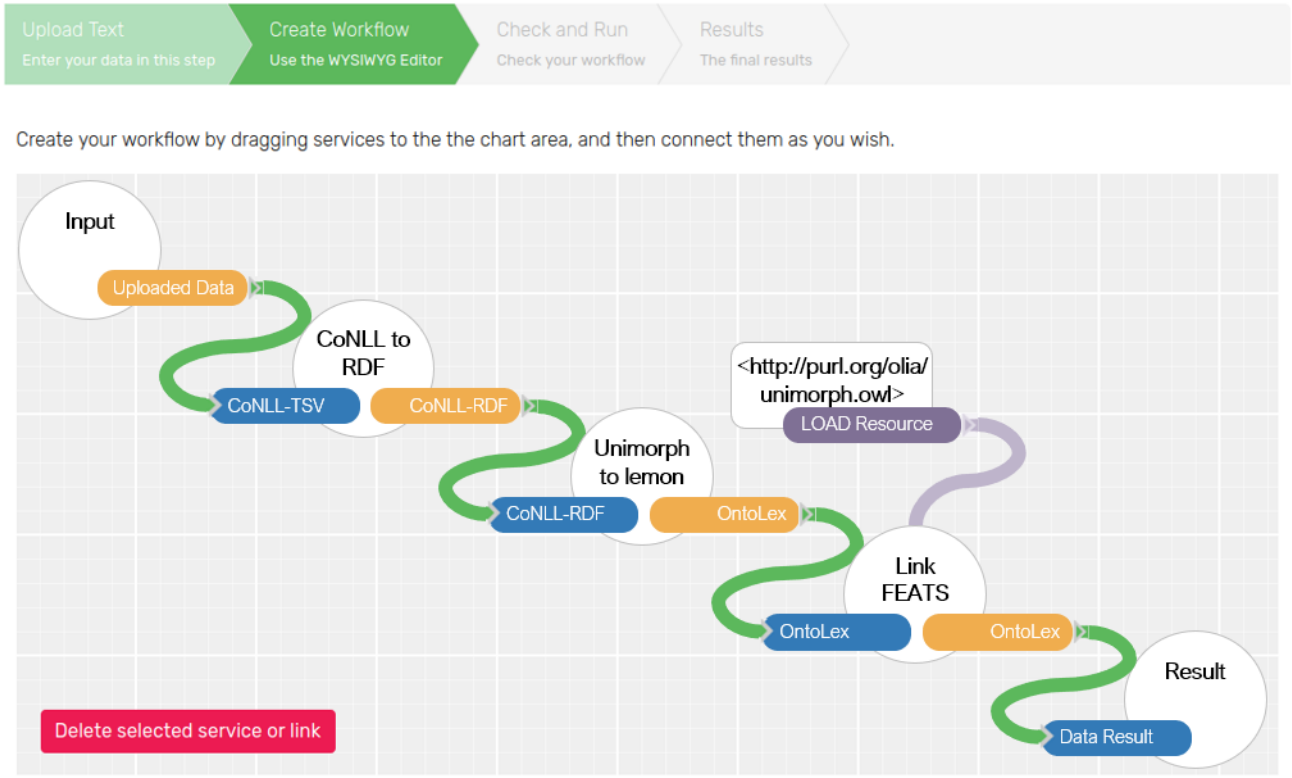

Figure 5: Fintan / Teanga workflow management

Table 1 shows the results of all three approaches[18]. While the en-bloc variant remains a fast option, it is limited by a memory overhead. The serialized approach is scalable, yet, given its old implementation, significantly slower. The Fintan parallelized approach with precached resources shows the potential of stream-based graph transformation. Keeping in mind, that the en-bloc approach executed the updates on a single Apache Jena dataset, the parallelized approach outperforms Jena's internal database engine. In our test case we achieve more than twice the performance by distributing the processing load across four threads.

5. Related Work

While numerous converters and data management suites exist, some also with graphical user interfaces, Fintan as a platform is fairly unique in scope. For RDF conversion of generic data into RDF, two prominent examples are the CSV2RDF[19] and R2RML (Das et al., 2012) suites which focus on tabular formats resp. relational databases as input formats. However, they produce highly generic output and do not employ workflow managers for integrated post processing. One of the key advantages of the Fintan approach is reusability of transformation steps and easy assessment of their interdependencies, allowing to post-process generic output into standard ontological formats. However, these tools could well be integrated as Fintan Loader components in future projects.

Regarding transformation of existing RDF resources, OpenRefine[20] offers a web-based user interface for editing and interlinking datasets and making them compliant to existing ontologies. It is however tailored towards manual assessment and semi-automated restructuring of single datasets. Fintan instead focuses on configuring integrated transformation steps which can be used in automated transformation pipelines. Furthermore, since OpenRefine relies on triple stores in the background, it can also be quite resource-heavy for large-scale datasets.

Database engines and triple stores (e.g. Neo4j[21], Blazegraph[22], AllegroGraph[23]) by themselves also pose the option of comfortably transforming resources, specifically by using SPARQL or similar query languages. Regarding usability, AllegroGraph offers a graphical interface for assessing and creating SPARQL queries. However, it is proprietary software and imposes severe restrictions on its free version. Blazegraph development on the other hand, as already mentioned in section 2.2., had focused on higher processing performance using GPU acceleration, but has never released the feature in the open source version. Apart from that, databases are server applications which need to be hosted on sufficiently powerful machines. Fintan instead focuses on distributing load across threads and working around memory limits by segmenting resources during transformation. It thus enables the creation of more portable pipelines which can run on weaker machines.

[18] The pipelines were executed on a 3.79 GHz i5 quadcore with 16 GB of memory

[19] https://www.w3.org/TR/csv2rdf/

[20] https://openrefine.org

[21] https://neo4j.com/

[22] https://blazegraph.com/

[23] https://allegrograph.com/

An alternative approach to further improve SPARQL processing speed is the restructuring of the queries and updates themselves. Adamou et al. (2019) recently released a recommendation for designing queries achieving promising results which possibly could also be incorporated by Fintan (e.g. as a preprocessing step or by optionally highlighting recommendations in the SPARQL visualization.)

Among existing proprietary workflow management engines, an open source alternative, LODflow (Rautenberg et al., 2015), provides tools for creating complex workflows including (manual) assessment and publication of Linked Data resources. It is, however, not specifically tailored towards creating automated transformation pipelines and has not seen further development since 2015. The Teanga implementation with its focus on integrating generic services into workflows thus posed a more feasible basis.

6. Summary and Outlook

We introduced Fintan, a platform that addresses the challenges of resource heterogeneity and interoperability from a processing or transformation perspective (rather than a standardization perspective). We provide a generic framework that allows to transform any linguistic annotation format (or data model for language resources) into any other kind of annotation (data model), by standardized means of graph transformation.

With this approach, we do not depend on any specific pivot format or data model for transformations, but merely require that a mapping to an RDF graph is possible for the formats to be transformed. Responsibility for performing those transformations is with the user of the system (the developer of a transformation workflow), but Fintan aims to provide accessible and (re-)usable technologies and facilitates the creation of such workflows by providing reusable loader, transformer and writer components, visualizations of SPARQL update operations and transformation workflow management. The stream-based graph transformation approach promises to also tackle scalability issues which are common with processing extensive LD resources and also prove a prominent hindrance for NLP applications.

While Fintan does not require a standard data model for representing the language resources it is applied to (beyond being mappable to a graph), adopting such models (where they exist) does improve the re-usability of Fintan workflows. For the specific case of representing lexical resources in RDF, such a model is available with OntoLex-Lemon, and we successfully performed the transformation and integration of a large number of lexical resources on that basis (Chiarcos et al., 2020).

As for linguistic annotations, a comparable standard does not exist, but different community standards co-exist and/or compete with each other, see Ide et al. (2017) for annotations in general and Cimiano et al. (2020, p.61-122 and 197-212) for RDF-based data models in particular. Important vocabularies include Web Annotation (Sanderson et al., 2017), the NLP Interchange Format (Hellmann et al., 2012, NIF), the NLP Annotation Format (Fokkens et al., 2014, NAF), the LAPPS Interchange Format (Verhagen et al., 2015, LIF), and the Linguistic Annotation Format (Ide and Suderman, 2014, LAF). In the context of the Cost Action CA 18209 'Nexus Linguarum. European network for Web-centred linguistic data science' (2019-2023),[24] and the W3C Community Group 'Linked Data for Language Technology' (LD4LT),[25] we are actively engaged in the development of a consensus vocabulary that harmonizes these conflicting specifications. Started only in late 2019, we expect this process to continue for a number of years before definite results will be achieved. Until then, Fintan will continue to support a broad range of RDF-based data models for representing linguistic annotations (and other forms of language resources).

Once a Fintan workflow for a particular type of resources has been designed, we plan to publish it as a stand-alone conversion component, wrapped into a Docker container and ready to be integrated in other systems, e.g., Teanga or the ELG. This container will comprise the Fintan backend, a particular configuration (a workflow description), and, optionally, language resources needed for a particular transformation task (e.g., lexical resources for a workflow that includes enrichment with lexical features), but can be used as a blackbox within more complex NLP workflows. With this approach, the Fintan frontend becomes a development IDE for workflow configuration.

This functionality, however, is not yet supported, but a strategic goal in the Pret-a-LLOD project. At the time of writing, the components of Fintan (backend, workflow management, SPARQL visualization) are functional and can be found in our GitHub repository.[26] As the integration of these components with each other and with external loader and writer components, we currently rely on application-specific JSON configuration files. More generic specifications for component metadata are currently developed in coordination with Teanga development, and will be partially based on the Fintan ontology. However, we plan to align our specifications also with those of the European Language Grid (ELG),[27] and thus anticipate a longer consolidation process and several cycles of revision until we arrive at stable specifications.

We also focus on collaboration both within and beyond the scope of the Prêt-à-LLOD project. Since Fintan is an open platform and tailored towards integrating existing converters, we always welcome additions. Based on earlier implementations (Gracia et al., 2017), an ongoing collaboration with the University of Zaragoza and Semalytix aims at creating a transformation pipeline for the Apertium bilingual dictionaries to be used for industry applications. We furthermore collaborate with the University of Bielefeld to establish a common ground between their novel Terme-à-LLOD platform and Fintan. Since Terme-à-LLOD[28] is currently limited to TBX[29] input in order to render, interlink and host it as OntoLex-Lemon, Fintan could incorporate its TBX to RDF conversion component. In return, Fintan promises to enable more heterogeneous input formats for Terme-à-LLOD.

[24] https://www.cost.eu/actions/CA18209/
[25] https://www.w3.org/community/ld4lt
[26] https://github.com/Pret-a-LLOD/Fintan
[27] https://www.european-language-grid.eu/
[28] https://github.com/ag-sc/terme-a-llod
[29] ISO 30042:2019, cf. https://www.tbxinfo.net/

7. Acknowledgements

The research described in this paper has been conducted in the context of the the Horizon 2020 Research and Innovation Action 'Prêt-à-LLOD', Grant Agreement number 825182.

8. Bibliographical References

Adamou, A., Allocca, C., d'Aquin, M., and Motta, E. (2019). SPARQL Query Recommendation by Example: Assessing the Impact of Structural Analysis on Star-Shaped Queries. In Maria Eskevich, et al., editors, *2nd Conference on Language, Data and Knowledge (LDK 2019)*, volume 70 of *OpenAccess Series in Informatics (OASIcs)*, pages 1:1–1:8, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11 - 15, 2007*, volume 4825, pages 722–735, Berlin. Springer.

Beckett, D., Berners-Lee, T., Prud'hommeaux, E., and Carothers, G. (2014). RDF 1.1 Turtle. W3C Recommendation. `https://www.w3.org/TR/turtle/`.

Buil Aranda, C., Corby, O., Das, S., Feigenbaum, L., Gearon, P., Glimm, B., Harris, S., Hawke, S., Herman, I., Humfrey, N., Michaelis, N., Ogbuji, C., Perry, M., Passant, A., Polleres, A., Prud'hommeaux, E., Seaborne, A., and Williams, G. (2013). SPARQL 1.1 Overview. W3C Recommendation. `https://www.w3.org/TR/sparql11-overview`.

Chiarcos, C. and Fäth, C. (2017). CoNLL-RDF: Linked Corpora Done in an NLP-Friendly Way. In *Language, Data, and Knowledge - First International Conference, LDK 2017, Galway, Ireland, June 19-20, 2017, Proceedings*, pages 74–88.

Chiarcos, C. and Schenk, N. (2018). The ACoLi CoNLL Libraries: Beyond Tab-Separated Values. In Nicoletta Calzolari, et al., editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 7-12, 2018. European Language Resources Association (ELRA).

Chiarcos, C. and Sukhareva, M. (2015). OLiA - Ontologies of Linguistic Annotation. *Semantic Web Journal*, 518:379–386.

Chiarcos, C., Fäth, C., and Ionov, M. (2020). The ACoLi dictionary graph. In Nicoletta Calzolari, et al., editors, *Proceedings of the Twelfth International Conference on Language Resources and Evaluation (LREC 2020)*, Marseille, France, May 11-16, 2020. European Language Resources Association (ELRA).

Cimiano, P., Buitelaar, P., McCrae, J., and Sintek, M. (2011). LexInfo: A declarative model for the lexicon-ontology interface. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(1):29–51.

Cimiano, P., McCrae, J., and Buitelaar, P. (2016). Lexicon Model for Ontologies. W3C Community Report. `https://www.w3.org/2016/05/ontolex/`.

Cimiano, P., Chiarcos, C., McCrae, J., and Gracia, J. (2020). *Linguistic Linked Data. Representation, Generation and Applications*. Springer, Cham.

Das, S., Sundara, S., and Cyganiak, R. (2012). R2RML: RDB to RDF Mapping Language. W3C Recommendation. `https://www.w3.org/TR/r2rml`.

Ebbrecht, B. (2019). SparqViz. Entwicklung einer benutzerfreundlichen Visualisierung für SPARQL-Abfragen. B.Sc. thesis. Goethe-University Frankfurt, Germany.

Ellson, J., Gansner, E. R., Koutsofios, E., North, S. C., and Woodhull, G. (2003). Graphviz and Dynagraph - static and dynamic graph drawing tools. In *Graph Drawing Software*, pages 127–148. Springer.

Fokkens, A., Soroa, A., Beloki, Z., Ockeloen, N., Rigau, G., van Hage, W. R., and Vossen, P. (2014). NAF and GAF: Linking linguistic annotations. In *Proceedings 10th Joint ISO-ACL SIGSEM Workshop on Interoperable Semantic Annotation*, pages 9–16.

Gracia, J., Villegas, M., Gomez-Perez, A., and Bel, N. (2017). The apertium bilingual dictionaries on the web of data. *Semantic Web*, 9:1–10, 01.

Hellmann, S., Lehmann, J., Auer, S., and Nitzschke, M. (2012). NIF Combinator: Combining NLP tool output. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 446–449. Springer.

Ide, N. and Suderman, K. (2014). The Linguistic Annotation Framework: a standard for annotation interchange and merging. *Language Resources and Evaluation*, 48(3):395–418.

Ide, N., Chiarcos, C., Stede, M., and Cassidy, S. (2017). Designing annotation schemes: From model to representation. In Nancy Ide et al., editors, *Handbook of Linguistic Annotation*, Text, Speech, and Language Technology. Springer.

Johnson, M. (2012). Computational Linguistics. Where do we go from here? Invited talk at the 50th Annual Meeting of the Association of Computational Linguistics (ACL-IJCNLP 2012), Jeju, Korea, (accessed 2016/07/13) `http://web.science.mq.edu.au/~mjohnson/papers/Johnson12next50.pdf`.

McBride, B. (2004). The resource description framework (RDF) and its vocabulary description language RDFS. *Handbook on Ontologies*, pages 51–66.

McGuinness, D., Van Harmelen, F., et al. (2004). OWL Web Ontology Language overview. W3C recommendation. `http://www.w3.org/TR/owl-features`.

Nastase, V., Mihalcea, R., and Radev, D. R. (2015). A survey of graphs in natural language processing. *Natural Language Engineering*, 21(5):665?698.

Pan, X., Cassidy, T., Hermjakob, U., Ji, H., and Knight, K. (2015). Unsupervised Entity Linking with Abstract Meaning Representation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1130–1139, Denver, Col-

orado, May–June. Association for Computational Linguistics.

Rautenberg, S., Ermilov, I., Marx, E., Auer, S., and Ngomo, A.-C. N. (2015). LODFlow: A Workflow Management System for Linked Data Processing. In *Proceedings of the 11th International Conference on Semantic Systems*, SEMANTICS '15, page 137–144, New York, NY, USA. Association for Computing Machinery.

Sanderson, R., Ciccarese, P., and Young, B. (2017). Web Annotation Data Model. Technical report, W3C Recommendation, February.

Sylak-Glassman, J., Kirov, C., Yarowsky, D., and Que, R. (2015). A language-independent feature schema for inflectional morphology. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 674–680, Beijing, China, July. Association for Computational Linguistics.

Verhagen, M., Suderman, K., Wang, D., Ide, N., Shi, C., Wright, J., and Pustejovsky, J. (2015). The LAPPS Interchange Format. In *International Workshop on Worldwide Language Service Infrastructure*, pages 33–47. Springer.

Yih, W.-t., Chang, M.-W., He, X., and Gao, J. (2015). Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1321–1331, 01.

Ziad, H., McCrae, J. P., and Buitelaar, P. (2018). Teanga: A Linked Data based platform for Natural Language Processing. In Nicoletta Calzolari, et al., editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 7-12, 2018. European Language Resources Association (ELRA).