Proceedings of the Linked Data in Linguistics Workshop @ LREC2022, pages 61–68
Marseille, 24 June 2022

Spicy Salmon: Converting between 50+ Annotation Formats with Fintan, Pepper, Salt and Powla

Christian Fäth, Christian Chiarcos
Applied Computational Linguistics (ACoLi)
Goethe University Frankfurt, Germany
{faeth|chiarcos}@em.uni-frankfurt.de

Abstract

Heterogeneity of formats, models and annotations has always been a primary hindrance for exploiting the ever increasing amount of existing linguistic resources for real world applications in and beyond NLP. Fintan - the Flexible INtegrated Transformation and Annotation eNgineering platform introduced in 2020 is designed to rapidly convert, combine and manipulate language resources both in and outside the Semantic Web by transforming it into segmented RDF representations which can be processed in parallel on a multithreaded environment and integrating it with ontologies and taxonomies. Fintan has recently been extended with a set of additional modules increasing the amount of supported non-RDF formats and the interoperability with existing non-JAVA conversion tools, and parts of this work are demonstrated in this paper. In particular, we focus on a novel recipe for resource transformation in which Fintan works in tandem with the Pepper toolset to allow computational linguists to transform their data between over 50 linguistic corpus formats with a graphical workflow manager.

Keywords: Interoperability, Transformation, Annotation, Corpora, Ontologies, Linguistic Linked Open Data

1. Entree

With the continued rise of corpus technologies in language sciences and lexicography that we have seen in the last decades, the number and diversity of linguistic annotations has been growing at an exponential rate – and this trend still continues. At the same time, the increasing maturity of language technology and machine learning and their spread to novel domains calls for ever increasing amounts of homogeneous training and evaluation data, so that a core challenge of applied NLP is, in fact, not so much to come up with innovative algorithms, but to secure the availability and consistency of the data needed for applying state-of-the-art technology.

Indeed, the heterogeneity of formats, models and annotations has always been a primary hindrance for exploiting the ever increasing amount of existing linguistic resources for real world applications in and beyond NLP. The Linguistic Linked Open Data (LLOD) community has a decade-spanning history of creating community standards for homogeneous data publication and interlinking. Relying on Semantic Web technology, Fintan - the Flexible INtegrated Transformation and Annotation eNgineering platform (Fäth et al., 2020) has been designed as a tool to rapidly convert, combine and manipulate heterogeneous language resources in a generic, sustainable and scalable way. Its transformation and export capabilities are tailored towards, but not limited to commonly used LLOD vocabularies such as CoNLL-RDF (Chiarcos and Fäth, 2017) or Ontolex Lemon (Cimiano et al., 2016) and thus alleviate generation of LLOD datasets and their integration into NLP workflows.

Fintan has recently been extended with a set of additional modules increasing the amount of supported non-RDF formats and the interoperability with existing non-JAVA conversion tools. In this paper, parts of this work are demonstrated. In particular, we focus on a novel recipe for resource transformation in which Fintan works in tandem with the Pepper (Zipser and Romary, 2010) toolset to allow computational linguists to transform their data between over 50 linguistic corpus formats with a graphical workflow manager.

2. The Fish: Fintan

Although Fintan as a tool doesn't have anything fishy about it, the acronym is actually coined as a metaphor to its generic, variable design. Fintan mac Bóchra in Irish folklore was a shape-shifting sage who survived a great flood in the shape of a salmon (Macalister, 1941), which is also reflected in the logo of the Fintan platform (cf. Fig. 1).

2.1. Software design

Fintan is designed to adapt to the flood of data and formats computational linguists are confronted with and also enable users to integrate their data with a wealth of resources from the Semantic Web. This is also reflected by the internal architecture which heavily relies on Semantic Web Standards such as SPARQL (Buil Aranda et al., 2013) and encourages users to take advantage of graph-based transformation capabilities while adhering to the following principles:

- Fintan is *generic* in that it builds on the transformation of linguistic annotations, lexical data structures, etc. into labeled directed multi-graphs and back. In particular, this can represent every type of linguistic annotation (Bird and Liberman, 2001).

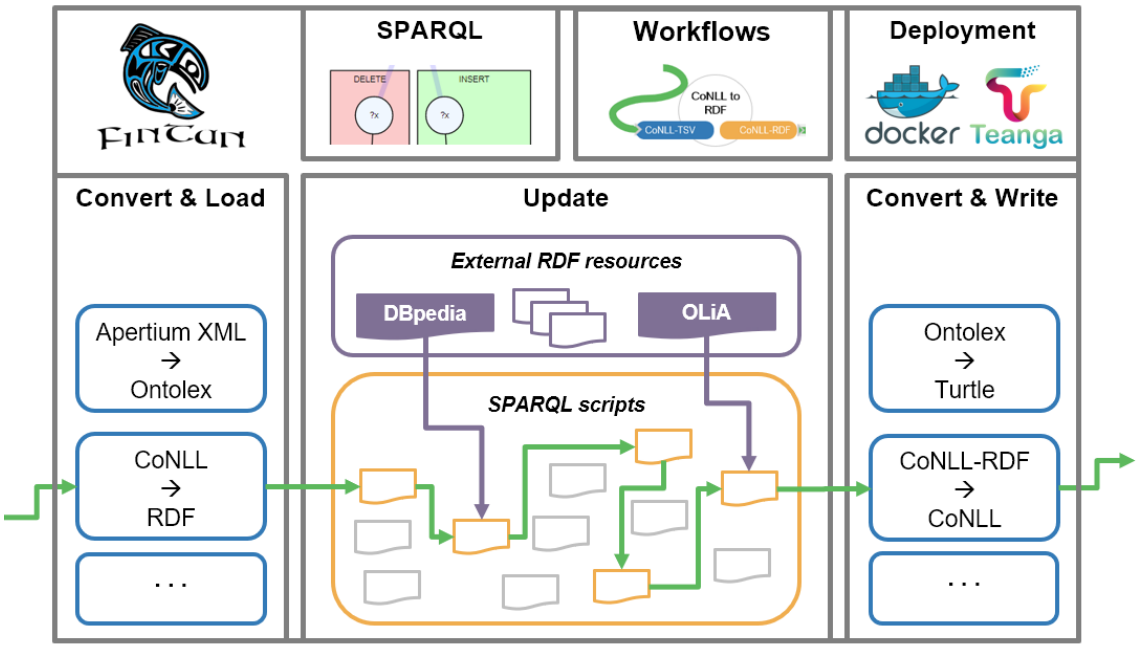

Figure 1: The Fintan plattform

- Fintan is *sustainable* in that it builds on web standards (RDF) and standardized, declarative transformations (SPARQL) for representing and manipulating these graphs. In particular, the SPARQL scripts can be run fully independently from the current Fintan code base, but against any SPARQL end point or with any programming language for which a SPARQL library is available.

- Fintan is designed to be *scalable*: It provides parallelized stream processing. Fintan hereby takes advantage of the inherently localized structure of most linguistic data which can often be divided into self-contained segments like entries in dictionaries or sentences, tokens etc. in corpora. By splitting data in such a way, we can process multiple segments at the same time while also minimizing memory consumption during query execution resulting in increased scalability, stability and faster execution.

Figure 1 provides an overview of the general software architecture and its *modular* structure. Within Fintan, data is transformed and streamed between components each providing specific processing capabilities:

- `Loader` components prepare data for segmented processing.

- `Updaters` apply SPARQL scripts on data segments, optionally also relying on external LLOD resources such as OLiA (Chiarcos and Sukhreva, 2015) for transforming linguistic annotations or DBpedia (Mendes et al., 2012) for entity linking.

- `Writer` components producing RDF serializations such as Turtle or exporting tabular formats like CoNLL.

Additional transformer components may also take other script languages to process various types of data, e.g. XSL for converting XML data as has been applied in a complex workflow enabling the Apertium bilingual dictionaries to be used for cross-lingual transfer learning in the pharmaceutical domain (Gracia et al., 2020). The CoNLL-RDF (Chiarcos and Fäth, 2017) library, Fintan's spiritual predecessor, is now also a native part of the toolchain allowing Fintan to directly execute any existing CoNLL-RDF pipeline.
By treating transformation scripts and pipeline configurations as data fed into standardized transformer components, we also open possibilities to increase *reusability*. Some scripts (like annotation transformation) may be applicable in multiple workflows for several types of resources including lexical and corpus data alike, albeit this is highly dependent on how users structure their pipeline configurations.
To alleviate this design process, Fintan also features a stand-alone graphical workflow manager which renders existing components as processing nodes which can be connected by edges reflecting data streams (cf. Fig. 4). Streams are hereby distinguished between unsegmented text streams and streams of pre-loaded RDF segments.

2.2. OpenAPI support

Pipelines created with the workflow manager can directly be inserted and run in the Fintan JAVA backend on a shell environment. However, they can also be

exported as dockerized[1] web services to be integrated into decentralized complex workflows. To achieve this we built a Python server which can expose the Fintan backend as an OpenAPI[2] compliant web service and provides functionality to upload scripts and data and to run pipeline configurations. While this server can be run stand alone, the Workflow manager can also create a makefile to directly build an integrated Docker container containing all relevant data and a specific pipeline configuration. The make script includes all relevant code and resources for automated deployment.
However, OpenAPI support is not just limited to deployment of workflows. Instead, we recently integrated API functions which allow external web services to be run as part of Fintan pipelines. This API is mostly based on the Swagger Code Generator[3] and has originally been created specifically for the use case described in this paper, but has been slightly redesigned to host generic services by exposing most configuration options (i.e. request methods etc.) as parameters in the Fintan JSON configuration. The generic wrapper component is however structurally limited to single requests per transformation as more complex operations would require specific treatment (possible wait operations, poll for success etc.) which are highly service specific. Such peculiarities must be addressed by implementing service specific wrapper components using the Fintan API[4].
This not only allows to address existing web services but also provides a means to wrap other toolsets which are structurally incompatible or not natively available in JAVA directly within Fintan.

3. The Spices: Salt and Pepper

Language resource interoperability is, indeed, a problem that has received a lot of attention over the years, and as far as linguistic annotations are concerned, the state of the art in this regard is represented by the Linguistic Annotation Framework (Ide and Suderman, 2014, LAF). LAF defines an abstract data model (a generic labelled directed acyclic multi-graph over streams of primary data) and an XML syntax (GrAF), and it is designed to be able to represent any linguistic annotation.
In terms of processing tools, however, very few technology seems to be around that actually implements LAF/GrAF directly.[5] Instead, real-world tools use a

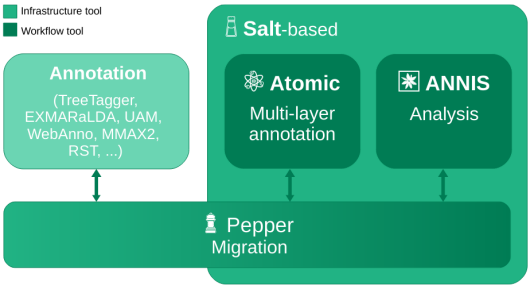

Figure 2: Interdependencies in the Salt/ANNIS universe of tools, taken from Druskat et al. (2016).

number of derivative formats (usually with less generic, but otherwise equivalent features), a notable example being the JSON-LD based LAPPS Interchange Format (Verhagen et al., 2015, LIF) developed by the creators of LAF. One such application is the converter suite Salt'n Pepper, to some extent overlapping in spirit to Fintan, but firmly integrated in its own, independent technological ecosystem

3.1. ANNIS and Salt ecosystem

ANNIS is a corpus management system specifically designed for dealing with multi-layer corpora (Dipper et al., 2004). With this goal in mind, its developers adopted early drafts of the LAF standard (Ide and Romary, 2004) as the underlying data model and developed a large set of converters from and to various established corpus formats.
ANNIS provides search and visualization capabilities for of multi-layer corpora, it supports annotations of different types (spans, trees with and without labelled edges, dependencies, arbitrary pointing relations), internally represented as a directed acyclic multi-graph. ANNIS comes with convenient visualizations and with different backend implementations as in-memory database (ANNIS1), relational database (ANNIS2, ANNIS3) and an experimental graph backend (Krause et al., 2016, GraphANNIS). The primary input format to ANNIS is PAULA XML (Sect. 4.1), and the underlying PAULA Object Model represents the basis for a small universe of interconnected tools, specifications and resources (Chiarcos et al., 2008; Druskat et al., 2016), see Fig. 2 for an architectural overview. In addition to ANNIS, these include

- PAULA (XML format and abstract data model)
- AQL (ANNIS Query Language)
- Salt (Java API and theory-neutral meta model)
- Pepper (converter suite)
- Atomix/Hexatomic (annotation tool)
- Laudatio (corpus repository)

[1] https://www.docker.com/
[2] https://www.openapis.org/
[3] https://swagger.io/tools/swagger-codegen/
[4] A detailed description on how to create and integrate custom components into Fintan is available here: https://github.com/acoli-repo/fintan-doc/blob/master/3c-build-custom-components.md
[5] There are a number of third-party converters, but the only tool natively building on LAF/GrAF seems to be the ANC-Tool (Suderman and Ide, 2006).

ANNIS operates a powerful, tagset-independent and theory-neutral meta model, and with its reference implementation in the Salt API, it allows for storing, manipulating, and representing nearly all types of linguistic data.

3.2. Pepper plattform

The Pepper (Zipser and Romary, 2010) plattform and its internal theory-neutral Salt meta model compose a framework which enables means of direct conversion between at least 20 formats for annotated corpora including EXMARaLDA, Tiger XML, MMAX2, RST, TCF, TreeTagger format, TEI (subset), PAULA and more[6]. Pepper's general architecture is partly reminiscent of Fintan in that it is based on Java and Maven and divides its processing steps into *Importers* (corresponding to Fintan's Loaders), *Manipulators* (corresponding to Fintan's Transformers and Updaters) and *Exporters* (corresponding to Fintan's Writers). However, the implementation and design principles differ in many regards:

- Pepper uses the Salt model as an internal abstraction layer for the processed data. This introduces compatibility between modules but also narrows the aim towards corpora and may result in a loss of unsupported pieces of information stored in the original data. Fintan on the other hand is completely format-independent.
- Pepper focuses on fully designed converters as modules. Fintan instead emphasises on atomic reusable operations, e.g. by allowing users to directly load and manipulate transformation scripts for components.
- Since Pepper is focused on corpora, it cannot easily replicate Fintan's ability to side-load ontologies or external RDF repositories for Annotation Engineering and resource enrichment tasks.

As a corpus transformation tool, Pepper, nevertheless, is a valuable addition to Fintan's portfolio and extends its coverage of corpus formats. Because of the structural similarities, we were first considering a direct integration as a native Java library, however there were some drawbacks to consider:

- Pepper exclusively uses file I/O and is not natively streamable without major refactoring or caching.
- Pepper uses the OSGi framework while Fintan operates on native Java and Apache Jena, thus introducing additional complexity and risks when trying to directly map Pepper modules as Fintan components.
- Pepper needs a lot of additional module data (bloating a possible direct integration into Fintan's backend)

For these reasons, we decided to treat Pepper as a stand-alone converter module and wrapped it into a dockerized OpenAPI service which is specifically designed to generate POWLA-RDF (cf. Sect. 4.2) data from any corpus format supported by a Pepper Importer. This service can be accessed within Fintan workflows using the OpenAPI transformer component.

4. The Scullions: PAULA and POWLA

PAULA is an abstract data model for the ANNIS query language AQL, underlying the Salt API and the PAULA XML format, but also POWLA, an OWL2/DL data model for linguistic annotation on the web (Chiarcos, 2012), and in terms of its expressivity, it is equivalent (but slightly older than) the ISO-standardized Linguistic Annotation Framework (LAF).

4.1. PAULA

PAULA's underlying data model, much like RDF, is represented by labeled directed acyclic (multi)graphs (DAGs) and thus contains various types of *nodes*, *labels* and *edges*:

- *Nodes* are distinguished between *terminals* (tokens or spans of characters in the source data), *markables* (flat, positional annotations referring to spans of terminals) and *structs* (functioning as structural parents to other nodes in a tree).
- *Edges* can thus be *dominance relations* (parent to child in structs) or simple *pointing relation* (directed, but without hierarchical implications).
- *Labels* can be attached to nodes and edges alike representing linguistic annotations.

4.2. POWLA

The POWLA vocabulary shown in Figure 3 is an OWL2/DL implementation of the PAULA Object Model and preserves similar data structures for linguistic annotations, as an example, syntactic tree structures are rendered in POWLA by means of:

- `powla:Node` for tokens and phrasal nodes,
- `powla:hasParent` for hierarchical relations between nodes,
- `powla:next` for sequential relations between nodes, and
- `powla:Relation` (with `powla:hasSource` and `powla:hasTarget`) for labelled edges.

PAULA thus has a high level of compatibility with existing Fintan workflows, as its OWL2/DL implementation POWLA is used in conjunction with CoNLL-RDF

[6] A full list of "known" modules is provided by the developers: `https://corpus-tools.org/pepper/knownModules.html`

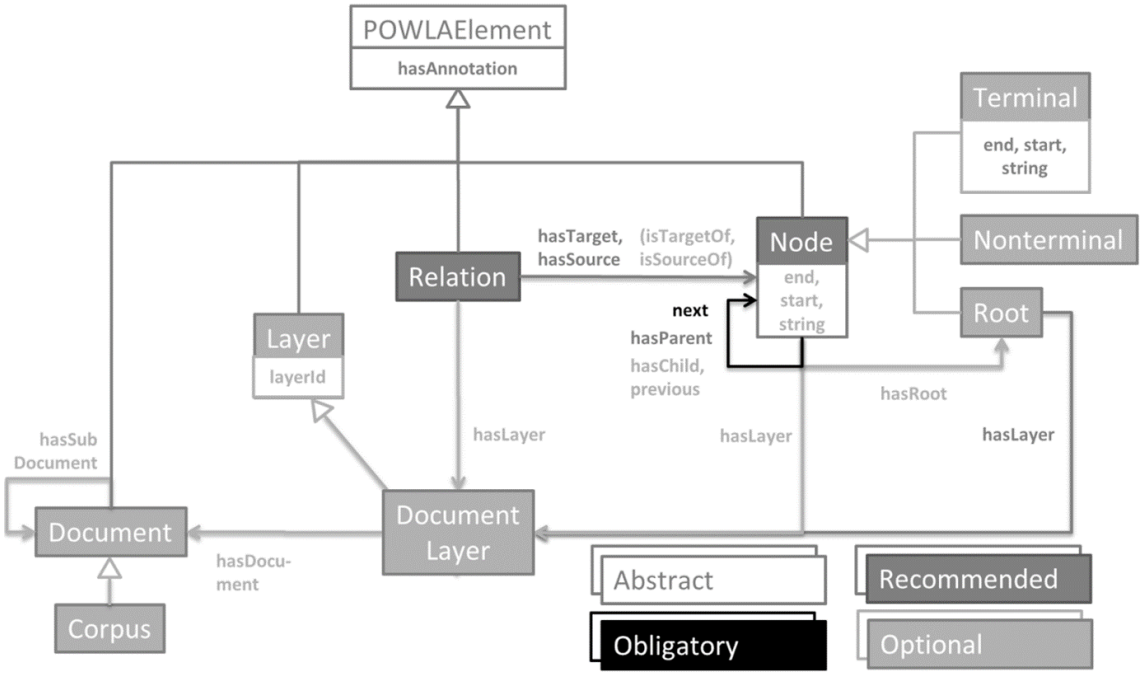

Figure 3: The POWLA vocabulary

(Chiarcos et al., 2021) to model linguistic data structures that exceed beyond labels of or pointers between individual words (Chiarcos and Glaser, 2020). PAULA and POWLA thus form a natural technological bridge between Pepper and Fintan, and here, they are being used to connect both technologies.

5. Main Course: Converting 50 Formats

Now that we have all ingredients and helpers ready, we can start concocting our workflow.

5.1. Preparations and Workflow

Since POWLA is a supported format in the CoNLL-RDF toolset (Chiarcos and Glaser, 2020) it can be converted to CoNLL-RDF by a set of SPARQL updates:

- Token level annotations in the form of `conll:COL` for typical columns such as `WORD`, `POS` etc. are derived from PAULA labels on markables and terminals.
- Dependencies are derived from dominance relations.
- Since POWLA does not necessarily annotate sentence boundaries, CoNLL-RDF's `nif:Sentence` nodes also need to be derived from the hierarchical data structure in order to produce a fully delimited CoNLL corpus. In case this fails or produces inconsistent output (e.g. if the corpus is annotated with a lot of cross-sentence relations), we can optionally split sentences purely based on punctuation in a post processing step

The sample configuration in Figure 4 shows how to convert data from the original PAULA XML format via POWLA into segmented CoNLL-RDF for further processing within Fintan. As a first step, we use the Pepper API service to transform PAULA to POWLA, then we use the RDF Splitter to induce the CoNLL-RDF data structure and split it by sentences into segmented graphs, which can be simultaneously processed with the RDF Updater for further customization. The splitting process uses Fintan's `ITERATE_CONSTRUCT` method by first selecting all sentence nodes in order with an iterator query:

```
SELECT ?s
WHERE {
  ?s a nif:Sentence
  BIND(xsd:integer(
    REPLACE(STR(?s),'[^0-9]','')
  ) AS ?snr)
} order by asc(?snr)
```

Subsequently, for each sentence `?s`, we execute a construct statement in which the wildcard `<?s>` is replaced by the specific sentence identifier:

```
CONSTRUCT {
  <?s> ?sp ?so .
  ?w ?wp ?wo .
} WHERE {
  <?s> ?sp ?so .
  ?w conll:HEAD+ <?s> .
  ?w ?wp ?wo .
}
```

The RDF Writer and CoNLL-RDF Formatter can then

output structured CoNLL or the CoNLL-RDF canonical format.

5.2. Adjusting the Recipe

The workflow depicted in Figure 4 is prepared to introduce additional processing steps. Since the resulting CoNLL-RDF data is already split into sentences, it is possible to directly execute updates on the segments transforming the existing annotations to commonly used schemes such as Universal Dependencies e.g. by using OLiA. OLiA at this point supports over 50 annotation schemes in its stable branch and features partial support for various additional models or reference catalogues including ISOcat and GOLD. In a similar manner, dictionaries could be side-loaded to infer foreign language lemmatization. Depending on the input data, even a complete recombination and restructuring of corpora is possible as we demonstrated by engineering a gold corpus for Role and Reference Grammar (Chiarcos and Fäth, 2019).

For native CoNLL output the CoNLL-RDF Formatter also alleviates structural customization, such as column reordering to directly feed data into subsequent NLP tools. With the CoNLL-RDF Ontology and CoNLL Transform (Chiarcos et al., 2021) we even introduced a means to automatically derive transformation pipelines from one CoNLL dialect to another which we aim to use as a blueprint for other formats as well.

In addition to producing CoNLL-RDF and CoNLL, also, other conventional corpus formats can be produced from POWLA. This includes bracketing formats as commonly used in treebanks such as the Penn Treebank (Marcus et al., 1993). XML-augmented TSV formats are SketchEngine (Kilgarriff et al., 2014) and the Corpus Workbench (Evert and Hardie, 2011) as also supported by the Fintan/CoNLL-RDF tool chain, but at the moment primarily as input formats.

6. One for the road

We have not just been cooking this up. Taking the combined capabilities of Pepper, Fintan and their configuration options into consideration, we are capable to cross-transform and recombine over 50 formats with a multitude of annotation schemes also taking advantage of parallelized stream processing. At the moment, this includes any CSV format (via Fintan's Tarql wrapper), 24 TSV formats (different CoNLL formats, Universal Morphology format, Sketch Engine/Corpus Workbench formats, OMW TSV format via CoNLL-RDF), 28 common corpus formats (via Pepper), all XML formats (with format-specific XSLT scripts for individual formats, e.g. for the Apertium dictionaries), the TBX format and numerous serializations of RDF data (RDF/XML, Turtle, JSON-LD, etc.). In addition to supporting different types of input data, Fintan also supports side-loading SKOS taxonomies, OWL ontologies, RDF and RDFS knowledge graphs as well as any custom XML or C/TSV resource when preprocessed into RDF graphs. As output formats we primarily support RDF serializations and customizable TSV formats. Since Fintan is an open platform, the number of supported formats can always be extended by building custom transformer components.

This combined support for taxonomies and both dictionary and corpus data inside a complex workflow manager which not only allows recombining existing converter components but alleviates full customization of transformation scripts is a somewhat unique approach to the data transformation challenge. Instead of dishing a buffet we provide ingredients and recipes in a prepare-your-own-pipeline package. Surely this is not a one-click solution and workflow accuracy is tied to the transformation components used. Specifically the preprocessing of data into POWLA using the Pepper framework heavily depends on individual Pepper modules and how lossless they render data in the internal SALT model. However, the resulting data can always be optimized within Fintan by additional resources or transformation steps in order to meet the requirements for specific use cases. Such preconfigured workflows can then be exported as stand-alone dockerized web services and made available in a sustainable way on Docker Hub[7] or as part of the European Language Grid[8] (ELG), which could also establish an interface to create additional processing nodes in large-scale infrastructure efforts such as the Switchboard or WebLicht platforms from CLARIN (de Jong et al., 2020).

We would like to emphasize that the integration of Pepper into Fintan not only increases the number of input and output formats supported by the platform. More importantly, it means that existing Fintan and CoNLL-RDF workflows can now be complemented with support for doing manual annotation (from HexAtomic, via Pepper), linguist-friendly means of querying and visualization (from ANNIS, via Pepper). And from the perspective of the Pepper/ANNIS universe, the addition of Fintan means that more advanced means of automated annotation and annotation engineering now become available that the native Java implementation of Salt'n'Pepper did not provide. Finally, via Fintan, Salt'n'Pepper can be connected with general-purpose NLP workflow management systems such as Teanga[9], linking it to the more algorithmic side of corpus linguistics with a possibility to integrate external webservices and Docker containers in annotation pipelines.

7. Acknowledgements

The research described in this paper has been partially conducted in the context of the BMBF Early Career Research Group 'Linked Open Dictionaries (LiODi)', and partially in the context of the Horizon 2020 Research and Innovation Action 'Pret-a-LLOD', Grant Agreement number 825182.

[7] https://hub.docker.com/

[8] https://www.european-language-grid.eu

[9] https://github.com/Pret-a-LLOD/teanga

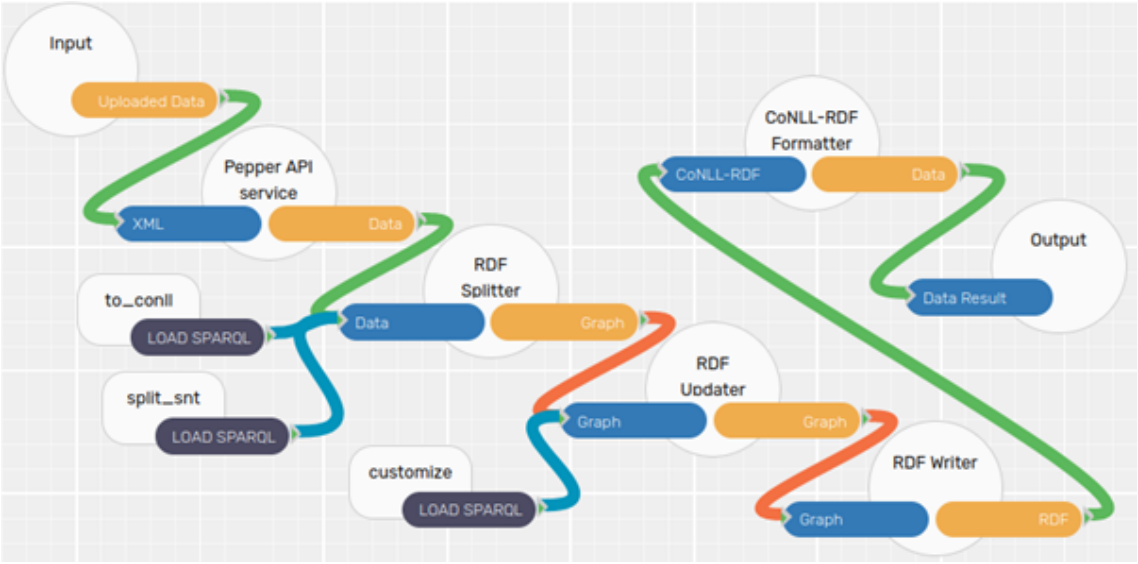

Figure 4: Fintan workflow converting PAULA to CoNLL-RDF

8. Bibliographical References

Bird, S. and Liberman, M. (2001). A formal framework for linguistic annotation. *Speech communication*, 33(1-2):23–60.

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. `https://www.w3.org/TR/sparql11-overview`.

Chiarcos, C. and Fäth, C. (2017). CoNLL-RDF: Linked corpora done in an NLP-friendly way. In *International Conference on Language, Data and Knowledge*, pages 74–88. Springer.

Chiarcos, C. and Fäth, C. (2019). Graph-based annotation engineering: towards a gold corpus for role and reference grammar. In *2nd Conference on Language, Data and Knowledge (LDK 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Chiarcos, C. and Glaser, L. (2020). A tree extension for CoNLL-RDF. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 7161–7169.

Chiarcos, C. and Sukhreva, M. (2015). OLiA – ontologies of linguistic annotation. *Semantic Web Journal*, 518:379–386.

Chiarcos, C., Dipper, S., Götze, M., Leser, U., Lüdeling, A., Ritz, J., and Stede, M. (2008). A Flexible Framework for Integrating Annotations from Different Tools and Tag Sets. *TAL (Traitement automatique des langues)*, 49(2):217–246.

Chiarcos, C., Ionov, M., Glaser, L., and Fäth, C. (2021). An ontology for CoNLL-RDF: Formal data structures for TSV formats in language technology. In *3rd Conference on Language, Data and Knowledge (LDK 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Chiarcos, C. (2012). POWLA: Modeling linguistic corpora in OWL/DL. In *9th Extended Semantic Web Conference (ESWC-2012)*, pages 225–239, Heraklion, Crete, May.

Cimiano, P., McCrae, J., and Buitelaar, P. (2016). Lexicon Model for Ontologies. Technical report, W3C Community Report, 10 May 2016.

de Jong, F., Maegaard, B., Fišer, D., van Uytvanck, D., and Witt, A. (2020). Interoperability in an infrastructure enabling multidisciplinary research: The case of CLARIN. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 3406–3413, Marseille, France, May. European Language Resources Association.

Dipper, S., otze, M. G., Stede, M., and Wegst, T. (2004). ANNIS: A linguistic database for exploring information structure. In *Interdisciplinary Studies on Information Structure*, ISIS Working papers of the SFB 632 (1), pages 245–279. Universitätsverlag Potsdam.

Druskat, S., Gast, V., Krause, T., and Zipser, F. (2016). corpus-tools. org: An interoperable generic software tool set for multi-layer linguistic corpora. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4492–4499.

Evert, S. and Hardie, A. (2011). Twenty-1st century Corpus Workbench: Updating a query architecture for the new millennium. In *Proc. of the Corpus Linguistics 2011 conference*, Birmingham. University of Birmingham.

Fäth, C., Chiarcos, C., Ebbrecht, B., and Ionov, M. (2020). Fintan - Flexible, Integrated Transformation and Annotation eNgineering. In *Seventh conference*

on International Language Resources and Evaluation, LREC 2020.

Gracia, J., Fäth, C., Hartung, M., Ionov, M., Bosque-Gil, J., Veríssimo, S., Chiarcos, C., and Orlikowski, M. (2020). Leveraging linguistic linked data for cross-lingual model transfer in the pharmaceutical domain. In *International Semantic Web Conference*, pages 499–514. Springer.

Ide, N. and Romary, L. (2004). A registry of standard data categories for linguistic annotation. In *Proc. the 4th International Conference on Language Resources and Evaluation (LREC-2004)*, pages 135–139, Lisbon, Portugal.

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.

Kilgarriff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychlý, P., and Suchomel, V. (2014). The sketch engine: ten years on. *Lexicography*, 1(1):7–36, Jul.

Krause, T., Leser, U., and Lüdeling, A. (2016). graphannis: A fast query engine for deeply annotated linguistic corpora. *J. Lang. Technol. Comput. Linguistics*, 31(1):1–25.

R. A. S. Macalister, editor. (1941). *Lebor Gabála Érenn: Book of the Taking of Ireland*, volume 2 and 3. Irish Texts Society, Dublin, Ireland.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19:313–330.

Mendes, P., Jakob, M., and Bizer, C. (2012). DBpedia for NLP: A multilingual cross-domain knowledge base. In *8th international Conference on Language Resources and Evaluation (LREC-2012)*, Istanbul, Turkey, May.

Suderman, K. and Ide, N. (2006). Layering and merging linguistic annotations. In *Proceedings of the 5th Workshop on NLP and XML (NLPXML-2006): Multi-Dimensional Markup in Natural Language Processing*.

Verhagen, M., Suderman, K., Wang, D., Ide, N., Shi, C., Wright, J., and Pustejovsky, J. (2015). The lapps interchange format. In *Proc. of the Int. Workshop on Worldwide Language Service Infrastructure*, pages 33–47. Springer.

Zipser, F. and Romary, L. (2010). A model oriented approach to the mapping of annotation formats using standards. In *Proceedings of the Workshop on Language Resource and Language Technology Standards, LREC 2010*, Malta.