An Ontology for CoNLL-RDF: Formal Data Structures for TSV Formats in Language Technology

Christian Chiarcos
Applied Computational Linguistics Lab, Goethe University Frankfurt, Germany

Maxim Ionov
Applied Computational Linguistics Lab, Goethe University Frankfurt, Germany

Luis Glaser
Applied Computational Linguistics Lab, Goethe University Frankfurt, Germany

Christian Fäth
Applied Computational Linguistics Lab, Goethe University Frankfurt, Germany

Abstract

In language technology and language sciences, tab-separated values (TSV) represent a frequently used formalism to represent linguistically annotated natural language, often addressed as "CoNLL formats". A large number of such formats do exist, but although they share a number of common features, they are not interoperable, as different pieces of information are encoded differently in these dialects.

CoNLL-RDF refers to a programming library and the associated data model that has been introduced to facilitate processing and transforming such TSV formats in a serialization-independent way. CoNLL-RDF represents CoNLL data, by means of RDF graphs and SPARQL update operations, but so far, without machine-readable semantics, with annotation properties created dynamically on the basis of a user-defined mapping from columns to labels. Current applications of CoNLL-RDF include linking between corpora and dictionaries [28] and knowledge graphs [36], syntactic parsing of historical languages [12, 11], the consolidation of syntactic and semantic annotations [8], a bridge between RDF corpora and a traditional corpus query language [24], and language contact studies [6].

We describe a novel extension of CoNLL-RDF, introducing a formal data model, formalized as an ontology. The ontology is a basis for linking RDF corpora with other Semantic Web resources, but more importantly, its application for transformation between different TSV formats is a major step for providing interoperability between CoNLL formats.

2012 ACM Subject Classification Information systems → Graph-based database models; Computing methodologies → Language resources; Computing methodologies → Knowledge representation and reasoning

Keywords and phrases language technology, data models, CoNLL-RDF, ontology

Digital Object Identifier 10.4230/OASIcs.LDK.2021.20

Supplementary Material *Dataset (Ontology)*: `https://doi.org/10.5281/zenodo.4361476`

Funding This work was funded by the project "Prêt-à-LLOD" within the European Union's Horizon 2020 research and innovation programme under grant agreement no. 825182, as well as the project "Linked Open Dictionaries" (LiODi), funded within the eHumanities program of the German Ministry of Education and Science (BMBF, 2015-2021).

1 Motivation: Incompatible TSV formats

The automated analysis of natural language requires different, and often, complex steps of processing, traditionally organized in a pipeline architecture. Depending on the specific goals, this does include designated modules for standard tasks such as sentence splitting, tokenization, part-of-speech labelling, lemmatization, morphological analysis, named entity

recognition, word sense disambiguation, entity linking, chunking, syntactic parsing, semantic parsing, coreference analysis, but also extend to more research-intense challenges such as discourse parsing, zero anaphora resolution or implicit semantic role labelling. For each of these processing steps, numerous implementations and data sets to train your own classifiers upon are available, normally, the formats they use only support information that is relevant to their specific annotation task. They do, however, usually follow common conventions, as both data sets and reference implementations often originate from long-standing series of shared tasks, and for the family of formats under consideration here, these are also shared with other applications in corpus linguistics and digital lexicography.

Tab-separated values (TSV) are a frequently used formalism to represent linguistically annotated natural language, e.g., in the long-standing series of Shared Tasks of the Conference of Natural Language Learning (CoNLL), recent initiatives on the creation of corpora and tools with cross-linguistically applicable ("universal") annotations [31, Universal Dependencies, UD], [27, UniMorph], [2, Universal Propositions], or in computational lexicography and corpus linguistics [13, Corpus Workbench], [26, Sketch Engine]. Many such "CoNLL" formats exist, but although they share a number of common features (e.g., one word per line, empty line to mark sentence breaks, comments after #), they are not interoperable with each other, as different pieces of information are represented differently in different dialects, e.g., placed in different columns or spread over multiple columns in one format, but consolidated into one in another.

CoNLL-RDF [7] is a set of tools introduced to facilitate processing and transforming CoNLL and other TSV formats in a serialization-independent way: On the basis of a user-provided mapping from columns to labels (properties), sentence by sentence (blocks of annotations separated by empty lines), tab-separated data is transformed to RDF graphs in accordance with the CoNLL-RDF data model.[1] Annotations can then be manipulated using SPARQL Update operations and serialized in TSV, RDF or XML formats. Unlike CSV2RDF [20], R2RML [15], and related general-purpose technology for mapping tabular data to RDF, CoNLL-RDF provides *linguistic* data structures: The CoNLL-RDF data model uses the NLP Interchange Format [21, NIF] to encode sentences, words and sequential relations between these, and extends it with properties for the annotation of words, syntactic dependencies and semantic roles. First introduced in 2017, this technology is now being used in a number of projects in NLP [1], knowledge engineering [19], linguistics [29] and Digital Humanities [22].

We describe a novel extension of CoNLL-RDF technology, introducing a formal data model. So far, CoNLL-RDF used a shallow approach to semantics, where annotation properties were created dynamically on the basis of a user-defined mapping from columns to labels (property names), without any machine-readable semantics: CoNLL-RDF representations were data-driven and unrestricted, so that the same information could be found under different properties, etc. It is, however, essential to provide machine-readable semantics for these properties as individual CoNLL dialects record this information differently.

The CoNLL-RDF ontology provides machine-readable semantics for an inventory of CoNLL properties (and classes) for a growing collection of about two dozen CoNLL and related formats currently used in language technology. In addition, a mapping between CoNLL properties and columns provides a formal, and machine-readable definition of the

[1] Even though sentence by sentence transformation creates a computational overhead, it prevents from having memory issues while processing large amounts of text. A detailed discussion of this design decision can be found in the original paper.

respective formats. Using this information, we provide a mapping between different TSV formats. A user is only required to specify input and output formats (say, CoNLL-U and CoNLL-X). Using the column mappings defined in the CoNLL-RDF ontology, we derive a transformation workflow that retrieves information from source columns, (optionally) transforms it and allocates them to the corresponding columns in the target format. This workflow is then executed using the Flexible Integrated Annotation Engineering (Fintan) platform [18].

The CoNLL-RDF ontology, introduced with this paper, adds machine-readable semantics for existing datasets encoded as CoNLL-RDF and provides the basis for linking RDF corpora with other Semantic Web resources. With the ontology, the relations between 24 TSV formats have be made explicit in a machine-readable way, and it now becomes possible to (a) automatically transform one TSV format into another, resp. (b) to assert/infer that a particular format cannot be automatically transformed into another. Aside from the ontology, we introduce CoNLL-Transform, a converter that uses the CoNLL-RDF ontology to bootstrap automated conversion routines.

In the context of transformation, the CoNLL-RDF ontology serves two main purposes: It provides a mapping from columns to properties, and it defines standard identifiers (URIs) for these properties. Even though this aspect is beyond the scope of the current paper, this is the basis for develop transformers that are capable to perform more complex operations, e.g., to derive CoNLL-2004 chunking information from a CoNLL-U dependency parse.

2 Background: CoNLL-RDF

Natural language processing (NLP) and knowledge graphs are two critical areas in the developmentof language technologies. Building bridges between the two bears potential to enable progress in both. CoNLL-RDF has been designed to serve as such a technological bridge, enabling researchers to easily go back-and-forth between popular one-word-per-line TSV formats used in language technology, and SPARQL and Semantic Web technologies used in knowledge engineering. CoNLL-RDF refers to a library that allows parsing each sentence from a CoNLL-TSV data stream (together with its context) into a separate RDF graph, to manipulate and to enrich it with SPARQL and reasoning technologies, and to serialize the result back to Turtle,[2] to (any) TSV format or to a number of other common formats used in language technology.

The CoNLL-RDF library is part of the Flexible Integrated Annotation Engineering (Fintan) platform [18], but also distributed individually. It is available as open source (Apache license 2.0) from our Github repository.[3]

2.1 One-word-per-line TSV formats in language technology ("CoNLL")

One-word-per-line formats, especially tab-separated value (TSV) formats have been a popular choice in a variety of applications for more than three decades. The fields of use include digital lexicography (SketchEngine [26]), corpus linguistics (Corpus Workbench/CWB [16]), natural language processing (TreeTagger [35]), and as an exchange format in a variety of different corpora projects. These formats enjoy continued and rising popularity because TSV allows for flexible encoding of any kind of word-level annotations, they provide an ideal

[2] That is, a canonical TTL representation that emulates the structure of CoNLL/TSV.
[3] `https://github.com/acoli-repo/conll-rdf`

middle ground between being machine-processable and human-readable, and they can be easily extended by creating additional columns with new annotations. As a result, TSV formats have become a de-facto standard in exchanging NLP data.

The listing below is an example from the 2005 Shared Task of the SIGNLL Conference on Computational Natural Language Learning (CoNLL-05):

```
#  WORDS          NE   POS  PARTIAL_SYNT       PARSE
   The             *   DT    (NP*   (S*        (S (NP *
   spacecraft      *   NN       *)    *               *)
   faces           *   VBZ   (VP*)    *           (VP *
   a               *   DT    (NP*     *           (NP *
   ...
```

Here, the first column contains the word, the second column contains named entity annotation, the third contains part-of-speech information. The following columns contain different forms of syntax annotation: The `PARTIAL_SYNT` column has two subcolumns, where the first subcolumn contains nominal and verbal chunks, and the second subcolumn contains sentence chunks. The `PARSE` column contains a full parse in accordance with the Penn Treebank [30].

Subsequently, the use of TSV formats to exchange linguistic data has since extended its spread beyond the CoNLL Shared Task and inspired novel corpora formats, e.g. the CoNLL-U format by Universal Dependencies has been created independent of the conference; however adheres and extends standards already motivated by the CoNLL Shared Task. In this paper we follow this convention by referring to all one-word-per-line TSV formats as CoNLL-TSV.

2.2 Words and sentences

CoNLL-RDF can transform any CoNLL-TSV dialect into a CoNLL-RDF representation, apply SPARQL updates to the transformed sentences and re-serialize the representations into RDF or TSV as defined by the user.

The CoNLL-RDF vocabulary builds on a minimal fragment of the NIF data model: Each word (row) is represented as a `nif:Word`, and connected to the following word of the same sentence by the `nif:nextWord` property. Each sentence (sequence of rows not interrupted by an empty line) is represented as a `nif:Sentence`, and connected to the following sentence by the `nif:nextSentence` property. Words can be organized in a dependency tree (using the `conll:HEAD` property), and for words that do not have a parent in the dependency annotation (incl. formats where no `HEAD` column is given), are linked by `conll:HEAD` to the respective `nif:Sentence`.

For representing annotations, the CoNLL-RDF toolchain uses column labels provided by the user in order to associate each column with a novel property in the `conll` namespace. CoNLL-TSV can be transformed into CoNLL-RDF using ad hoc labels, but these are not backed by a formal ontology. In general, these column labels were simply treated as such; with the sole exception of semantic role annotations in the form of `PRED_ARGs` and the `HEAD` column. Both of these carry special semantics and are handled specially during conversion.

In real-world applications, e.g., the creation of novel forms of syntactic-semantic annotation [9] or experimental forms of syntactic parsing [12], where specialized data structures are required, a more constrained view is taken, and consistent labels should be used throughout the project – but so far, CoNLL-RDF provides no way to facilitate interoperability and interpretability of column labels across different annotation projects.

2.3 Tree extension

One-word-per-line formats originally provide no base vocabulary for representing annotations that span beyond more than one token, and different extensions have been developed throughout the CoNLL shared tasks. These include the IOB(ES) annotation for non-recursive spans introduced with the CoNLL-00 Shared Task [34], the bracket notation of the Penn Treebank [30], and the application of XML, resp., SGML markup *between* the word-level annotations (as used by TreeTagger, the Sketch Engine and the Corpus Workbench).

We illustrate tree structures with the bracket notation from the sixth column (`PARSE`) of the CoNLL-05 example given above: The original CoNLL-RDF implementation represented these structures as plain string literals, without analyzing their internal structure, i.e., as `conll:PARSE "(S ( NP *"`, etc. Processing such data with SPARQL is possible but cumbersome, as the strings need to be decoded before their content can be analysed. In essence, a user would need to write a CFG parser in SPARQL – and this is possible, but slow. Chiarcos and Glaser [10] thus extended the CoNLL-RDF library with routines for the native parsing and serialization of such structures. In order to avoid the introduction of ad hoc data structures into the CoNLL-RDF data model, the internal representation of phrases is grounded in the POWLA vocabulary [4].

POWLA provides an OWL2/DL formalization of the Linguistic Annotation Framework (LAF) as described by [23, 25]. LAF provides generic data structures for representing *any* kind of linguistic annotation. In particular, this includes a separation of positions and spans in the primary data (represented as anchors and regions, comparable to the target element of Web Annotation, or to instances of `nif:String`) and annotations (represented as nodes and edges/relations, comparable to annotations, resp., their bodies in Web Annotation; NIF, instead, recommends to model annotations as `nif:String` objects).[4] POWLA does not provide a formalization of anchors and regions, but builds on other vocabularies for this purpose, most notably NIF and Web Annotation [14]. Accordingly, the CoNLL-RDF tree extension uses POWLA data structures primarily to represent data structures above the level of the token (word), and for these, POWLA allows to define an annotation graph independent of the sentence and word structure imposed by CoNLL.

In CoNLL-RDF, only a minimal fragment of the POWLA vocabulary is used, partially with slightly more constrained definitions than in POWLA originally:

powla:Node Within CoNLL-RDF, every `nif:Word` is a `powla:Node`. Other nodes in CoNLL-RDF are recursively defined as a `nif:Word` or any grouping of `powla:Node`s.

powla:hasParent property pointing from an element to the phrase (or other aggregate node) that contains it.

powla:next To facilitate navigation in POWLA graphs, adjacent `powla:Node`s that share the same parent (and only these) should be connected by the `powla:next` property. A `powla:Node` may have multiple `powla:next` properties relative to different parent nodes, e.g., if multiple levels of syntax annotation are provided as in the three levels of syntax annotation of CoNLL-05.

powla:Relation for representing labelled (annotated) edges, POWLA allows to reify relations between a source (`powla:hasSource`) and a target node (`powla:hasTarget`). POWLA relations are not automatically generated during the process of parsing TSV formats, but can be created and used by subsequent SPARQL Update operations.

[4] Note that NIF 2.0 also introduced a `nif:Annotation` object, but `nif:Phrase`, etc., are still defined as `nif:String` subclasses.

When parsing the bracketing notation, the original column name is maintained as a datatype property, but applied to the POWLA node rather than the `nif:Word`. The value of this property is the label of the corresponding phrase. For the CHUNK column (column 4) in the example, three POWLA nodes are created:

- a node containing *The spacecraft*, with `conll:CHUNK 'NP'`
- a node containing *faces*, with `conll:CHUNK 'VP'`
- a node containing *a ...*, with `conll:CHUNK 'NP'`

SENTENCE (column 5) and PARSE (column 6) are processed analoguously.

3 CoNLL-RDF ontology

We designed the CoNLL-RDF ontology to capture the semantic and structural dependencies of annotations in TSV formats and tree extensions, with a focus on properties, as the basic structures of annotation are defined in external vocabularies, i.e., NIF (words and sentences) and POWLA (other units of annotation).

The ontology consists of two components: (1) classes, properties and axioms used to define formats (Fig. 1), and (2) the machine-readable description of existing CoNLL and related formats. The namespace prefix is `conll:`.

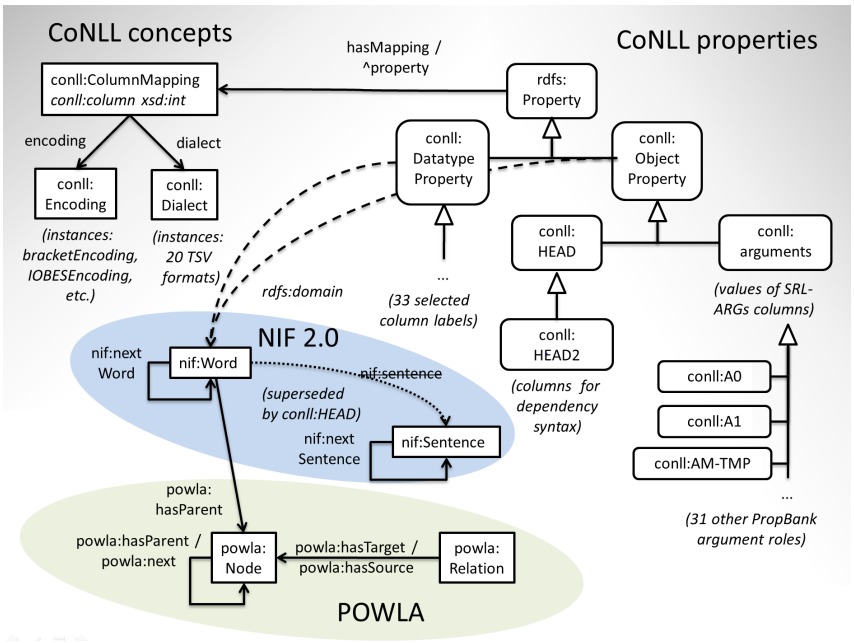

Figure 1 CoNLL-RDF Ontology: Classes and properties of CoNLL and external vocabularies.

As for CoNLL properties, the ontology provides a catalog of 33 datatype properties, with human-readable descriptions and labels as used in previous literature, and organized in an inheritance structure. The column label WORD, used in CoNLL Shared Tasks until 2005, does, for example, roughly correspond to the column label FORM, but the latter is a generalization, so that `conll:WORD` is a `:subPropertyOf conll:FORM`.

As for object properties, their creation is triggered by conventions in the CoNLL-RDF tool chain: HEAD contains the ID (or sentence position) of the syntactic head, and `conll:HEAD` refers to the head *or* to the sentence (if no head does exist). The column label *PRED_ARGS*

is used for semantic role annotations, where every predicate in a sentence triggers the creation of a subsequent argument column that annotates all words for their respective semantic role relative to this particular predicate. Here, the CoNLL properties are not generated from user-provided labels, but from the labels used in the annotation, e.g., `conll:A0` for the agent argument of a transitive verb. The CoNLL ontology thus contains the full inventory of PropBank roles.

We provide `conll:DatatypeProperty` and `conll:ObjectProperty` as subproperties of `rdf:Property` – not because we want to replicate OWL semantics, but because we restrict their domain to `nif:Word`s and `powla:Node`s.

As for classes, the CoNLL ontology does not introduce data categories, but concepts for metadata only (the mapping from CoNLL properties to different formats). It does, however, refine NIF and POWLA concepts with a more constrained definition than in their original vocabularies. As an example, a `nif:Word` within CoNLL-RDF must be an instance of `powla:Node`.

Each CoNLL-TSV and related TSV format is represented by an individual of the `conll:Dialect` type. A minimal dialect definition consists of a name (`rdfs:label`) and a link to documentation (`rdfs:isDefinedBy`).

A dialect *may* be used in one or more `conll:ColumnMapping`s. A column mapping links a CoNLL property (`conll:property`) with a particular column position (`conll:column`) in a particular format (`conll:dialect`). Any CoNLL property can be related to multiple mappings. Each relation then describes a mapping for a specific property in a specific dialect. This allows to represent data independent of the exact dialect. Instantiations of both property types will be represented by a column in the TSV file or by a `conll:column` property in CoNLL-RDF. In different TSV dialects, these will sit in different columns, considering their index. In addition, the column mapping can define the encoding strategy of POWLA nodes by means of the `conll:encoding` property. Note that the same property can be encoded in different ways, as shown for the bracket notation above and the IOBES encoding below.

With the classes and properties introduced above, we are able now to model CoNLL data in CoNLL-RDF, independent of the dialect in which it was originally encoded. As part of the ontology, we provide formal data structures and properties for 22 CoNLL-TSV and related TSV dialects. This includes all CoNLL Shared Task TSV formats until 2018, CoNLL-U, UniMorph, several PropBank formats, the formats of the Open Multilingual WordNet initiative, SketchEngine, Corpus WorkBench, TreeTagger, and the format of a series of Shared Tasks on Translation Inference Across Dictionaries (TIAD). We illustrate this below for the CoNLL-00 format:

```
The             DT    B-NP
spacecraft      NN    I-NP
faces           VBZ   B-VP
a               DT    B-NP
...
```

The CoNLL-00 columns are WORD, POS and CHUNK, for a Shared Task on chunking (shallow syntax). Note that the CoNLL-RDF tree extension renders the content of the CHUNK column *exactly* in the same way as the CHUNK information from the CoNLL-05 format described in Section 2.1. In the CoNLL-RDF ontology, we provide the full description of the CoNLL-00 format:

```
:CoNLL-00 a :Dialect;
rdfs:label "CoNLL-00 format";
rdfs:isDefinedBy <https://www.clips.uantwerpen.be/conll2000/chunking/>.

# first column (for CoNLL-00): WORD
:WORD rdfs:subPropertyOf :FORM; rdfs:label "WORD";
  rdfs:comment "Word form in an annotated text ..."@en;
  :hasMapping [ a :ColumnMapping; :column "1"^^xsd:int ; :dialect :CoNLL-00 ] ;
  :hasMapping [ a :ColumnMapping; :column "4"^^xsd:int ; :dialect :CoNLL-11 ] . # etc
```

```
# second column: POS
:POS rdfs:subPropertyOf :DatatypeProperty; rdfs:label "POS", "POSTAG", "TAG";
rdfs:comment "Fine-grained part-of-speech tag ..."@en;
:hasMapping [ a :ColumnMapping; :column "2"^^xsd:int; :dialect :CoNLL-00 ] ;
:hasMapping [ a :ColumnMapping; :column "3"^^xsd:int; :dialect :CoNLL-05 ] . # etc

# third column: CHUNK
:CHUNK rdfs:subPropertyOf :DatatypeProperty ; rdfs:label "CHUNK";
rdfs:comment "The chunk tags contain the name of the chunk type,
              for example I-NP ..."@en;
:hasMapping [ a :ColumnMapping; :column "3"^^xsd:int; :dialect :CoNLL-00 ] ;
:hasMapping [ a :ColumnMapping; :column "4"^^xsd:int; :dialect :CoNLL-05 ] . # etc
```

The listing only provides a partial view of the column mappings beyond CoNLL-00, illustrated for one example per CoNLL property. Also, these are slightly simplified, as they do not specify the actual encoding in CoNLL.

Although different `conll:Dialects` may share a `conll:COLUMN` or even the `conll:ColumnMapping`, the textual representation in CoNLL might differ, e.g. phrase structure might be encoded in IOBES or in a bracket notation. This information is encoded by an instance of `conll:Encoding`; `conll:iobesEncoding` resp. `conll:bracketEncoding` in this case.

As an example, CoNLL-05 does contain the same (plus other) annotations as CoNLL-00, but the chunk information (first `PARTIAL_SYNT` column, column 4) uses the bracketing notation of the Penn Treebank (equivalent with the CHUNK information from our CoNLL-00 sample).

The (abbreviated) entry of the property `conll:CHUNK` is presented below.

```
:CHUNK rdfs:subPropertyOf :DatatypeProperty ; # ...
    :hasMapping [
     :encoding :iobEncoding;
     a :ColumnMapping;
     :column "3"^^xsd:int;
        :dialect :CoNLL-00, :CoNLL-01, :CoNLL-03, :CoNLL-04];
    :hasMapping [
     :encoding :bracketEncoding;
     a :ColumnMapping;
     :column "4"^^xsd:int; :dialect :CoNLL-05 ].
```

The `conll:DatatypeProperty conll:CHUNK` sits in the third column in the CoNLL-00, -01, -03 and -04 dialects. In CoNLL-05, the `conll:CHUNK` property moves to the fourth column, the position is encoded using the `conll:column` property. The formats also differ in their encoding: The `conll:CHUNK` column in CoNLL-00, -01, -03 and -04 was encoded using an IOB-schema.[5] In CoNLL-05 however, the `conll:CHUNK` column was not only moved but encoded using a PTB-style annotation, marked with the `conll:encoding` property.

4 CoNLL-Transform: Ontology-based transformation

The Flexible Integrated Annotation Engineering (Fintan) platform is a recently introduced additional abstraction layer on the existing CoNLL-RDF library [18]. While CoNLL-RDF focuses on the transformation of CoNLL corpora, Fintan broadens the scope towards integrating support for other data formats, such as OntoLex-Lemon for lexica by allowing to easily integrate and run existing converters in complex pipelines. It furthermore adds a graphical workflow manager to build, assess and run these pipelines.

[5] Simplified IOBES encoding, using B- (begin of a single-token or multi-token sequence), I- (middle or end of a multi-token sequence), O (no annotation).

The Fintan API distinguishes different types of transformation modules for which CoNLL-RDF provides designated implementations:

- *Loader* modules may consume any type of input data and must write back RDF data. The CoNLL-RDF `CoNLLStreamExtractor` serves as a Loader module which transforms CoNLL into CoNLL-RDF.
- *Update* modules transform a stream of segmented RDF data on multiple threads. Each Update module is defined by the resources and graphs it requires and an iteration over SPARQL update scripts. CoNLL-RDF provides a `CoNLLRDFUpdater` class that implements the Update function.
- *Writer* modules create serializations of the transformed data. In CoNLL-RDF, the `CoNLLRDFFormatter` functions as a writer module that yields RDF serializations (Turtle, canonical CoNLL-RDF), TSV output and other representations. In the context of Fintan, other existing transformation tools may be mapped as Loaders or Writers.

4.1 Transforming CoNLL dialects

We provide CoNLL-Transform as a command-line tool to generate Fintan transformation workflows directly from the CoNLL-RDF ontology. The code and its integration with the Fintan infrastructure will be published under the Apache 2.0 license along with the publication of this paper.

CoNLL-Transform takes three parameters, source format (e.g., CoNLL-00), target format (e.g., CoNLL-05) and the CoNLL-RDF ontology (or a replacement that provides alternative column mappings, etc.). If the format identifiers match the (local names of) `conll:Dialects` in the ontology, we retrieve the corresponding column mappings (and the encoding specification) to derive the corresponding Fintan Loader and Writer configurations, either as a JSON configuration file, or in the form of a shell script. The combination of a particular Loader and a particular Writer already allows the reordering of columns, but moreover, also to switch from one way of phrase-level encoding (say, IOBES) to another (say, bracket notation) – if specified in the ontology.

4.2 Mapping strategies

Most CoNLL-TSV formats will not provide fully equivalent content. CoNLL-Transform also produces a protocol that lists target format properties (columns) not found in the source (which will be replaced by the empty annotation _), as well as source format properties (columns) not expressible in the target format (which will be omitted). In addition to repositioning columns, changing their encoding, and identifying mismatches between formats, CoNLL-Transform uses the subsumption hierarchy of the CoNLL-RDF ontology to derive heuristic mappings of column names.

We employ the following ranking of mapping strategies:

maintain **if** a CoNLL property from the target format occurs in the source format, maintain it; **otherwise:**

generalize **if** a CoNLL property from the source format (say, `conll:WORD` in CoNLL-00) is a subproperty of a CoNLL property from the target format (say, `conll:FORM` in CoNLL-U), copy the object of the source property to the target property and produce a warning; **otherwise**

specialize **if** a CoNLL property from the source format (say, conll:POS in CoNLL-00) is a superproperty of a CoNLL property from the target format (say, conll:XPOS in CoNLL-U), and the target format property does not already exist, copy the object of the source property to the target property and produce a warning; **otherwise**

skip if no other mapping applies, set target property to empty (`_`)

These mappings are provided as a series of SPARQL Update operations and executed by Fintan before the Writer component is called. Note that this mapping is heuristic. In particular the **specialize** step does introduce a certain amount of errors. Along with `conll:XPOS` (original POS annotation of a Universal Dependencies file), CoNLL-U features another subproperty of `conll:POS`, namely `conll:UPOS`. The property `conll:UPOS` is, however, more constrained than `conll:POS`, and restricted to POS annotation in accordance with the Universal Dependencies tagset. While mapping `conll:POS` to `conll:XPOS` will be correct in most cases, our mapping heuristic will also produce an analoguous mapping from `conll:POS` to `conll:UPOS` which will be incorrect in many cases.

In the future development, we also plan to provide an inventory of SPARQL update scripts for transformations between selected pairs of near-equivalent CoNLL properties. As an example, CoNLL-04 uses two columns to represent predicates in semantic role annotation: PRED_LEMMA contains the lemma of the predicate (e.g., `say`), PRED_FRAMESET contains the sense number (for that particular lemma, e.g., `1`). In CoNLL-08, this information is provided in the PRED column, but concatenated with a separator symbol (`say.01`). Once a SPARQL Update with such a concatenation is provided, it should be applied with preference over the `generalize` and `specialize` mappings.

4.3 Encoding

Aside from the mapping, a key challenge of transforming between different CoNLL-TSV dialects is the encoding of annotations spanning multiple words. We adopt the recent tree extension of CoNLL-RDF: Every span is represented as a `powla:Node`, and linked with the words (or nodes) it contains by means of `powla:hasParent`. These nodes then receive the corresponding annotation.

With the ontology, we can now define which CoNLL dialect uses which encoding strategy, and decode the correct spans and labels from the input data. Likewise, we can use the definition of the output format to trigger a serialization according to another encoding, and thus translate between the IOBES and bracket notations in the listings given above. Note that, however, this is not a transformation of the RDF graph, but only a decoding, resp. encoding instruction executed by Fintan Loader and Writer modules, respectively.

5 Related Community Standards

In this paper, we introduced the CoNLL-RDF ontology as a machine-readable formalization of the CoNLL-RDF data model, in order to facilitate interoperability and transformability between different TSV formats in language technology as well as between these and the knowledge representation / knowledge engineering stack. It is to be noted, however, that the CoNLL-RDF vocabulary does remain extensible, i.e., users can still provide ad hoc labels to generate `conll:` properties, and this feature is very much required. For *established* formats, however, the ontology provides instructions for decoding TSV annotation and for encoding CoNLL-RDF graphs in a TSV format.

It is important, however, that the CoNLL-RDF ontology may also serve a role in the development of vocabularies for linguistic annotations on the web. The development of RDF-based data models for language technologies coincides with a growing trend in publishing language resources (lexicons, corpora, dictionaries, etc) as linked open data (LOD) on the Web [3]. In application to language resources, *linguistic linked open data* (LLOD) is concerned with LOD resources that are *linguistically relevant*, i.e., part of an application or a use case in language technology or the language sciences, cumulating in the emergence of the so-called LLOD cloud.[6] In comparison to conventional means of publishing language resources, LLOD allows for as higher independence from domain-specific data formats or vendor-specific APIs, as well as easier access and re-use of linguistic data by semantic-aware software agents [14].

Prominent vocabularies in the LLOD context include OntoLex-Lemon[7] as the main community standard for ontology lexicalization and representing lexical data in RDF. For representing linguistic annotations, however, no single consensus vocabulary has emerged so far, but instead, incompatible and competing specifications. Most notably, these include Web Annotation [33, WA], the NLP Interchange Format [21, NIF] and the LAPPS Interchange Format [37, LIF].

Web Annotation was originally developed for adding shallow annotations (primarily labels, glosses or entity links) to web resources, but lacking the necessary data structures to represent complex data structures as needed for linguistic annotation. It is possible to complement Web Annotation with linguistic data structures [38], but these are not covered by the Web Annotation specification nor do they seem to be used in current practice.

The NLP Interchange Format (NIF) has been developed to facilitate the implementation of NLP workflows on the basis of web technologies. NIF is a representative of a broader class of RDF-based vocabularies designed for this purpose, e.g., TELIX [32], NAF-RDF [17], etc., but taken here as an example because it is relatively widely used and not tied to a specific piece of software. RDF-based NLP data models such as NIF provide linguistic data structures for a number of specific applications (part-of-speech tagging, entity linking, parsing, etc.), but they are extended according to the requirements of these applications.

CoNLL-RDF is grounded in the NIF vocabulary, but extends it in two important ways: (i) It introduces its own IRI fragment schema, based on a segmentation in sentences and tokens, and thus allows to refer to empty elements. (ii) For the representation of syntax and other, advanced levels of representation CoNLL-RDF complements NIF with POWLA data structures and is thus capable to represent *every* kind of linguistic annotation (a claim we inherit from the Linguistic Annotation Framework that POWLA formalizes [5]).

6 Summary and Outlook

CoNLL-RDF thus complements both Web Annotation and NIF with a model firmly grounded in state-of-the-art NLP *research* and used in mature NLP *applications*, and thus, better prepared for future applications based on current-day research. Within this paper, we provide the first formal account of the necessary data structures, and the first formalization of individual CoNLL dialects and related formats.

At the same time, however, CoNLL-RDF is not merely a data model, but it comes with a number of tools to facilitate the creation, manipulation and evaluation of linguistic annotations, as well as interoperability with "classical" formats in NLP and the language

[6] https://linguistic-lod.org/llod-cloud
[7] https://www.w3.org/2016/05/ontolex/

resource community. It is important here to note that CoNLL-RDF does not aim to replace these with an RDF substitute within NLP, but instead, it formalizes existing TSV formats, can be serialized in TSV and thus be seamlessly integrated in existing NLP workflows – for applications that benefit from graph data structures as opposed to tables (e.g., dependency syntax, coreference or semantic roles), applications that build on the integration of information from multiple, distributed sources or that are beyond the expressivity of an existing TSV format. The main application, however, is to establish interoperability among TSV formats and between these and knowledge graph technologies, and this is where we see the potential of CoNLL-RDF.

Despite the wide range of potential applications, we see CoNLL-RDF not as a potential replacement for either NIF or Web Annotation. Instead, it aims to provide a technological bridge between NLP standards and the RDF/Linked Data world. In the context of RDF vocabularies, we expect CoNLL-RDF to serve (along with Web Annotation, the NLP Interchange Format and ISO TC37 standards) as a main input for the development of a harmonized vocabulary for linguistic annotations on the web that is currently under development within the W3C Community Group Linked Data for Language Technology (LD4LT).[8]

The CoNLL-RDF ontology is available as part of the CoNLL-RDF repository and published under the same license (Apache 2.0).[9] The ACoLi CoNLL libraries (that contain CoNLL-RDF and CoNLL-Transform, along with other modules) are also open-source published and available from `https://github.com/acoli-repo/conll`. The ontology has been published under `http://purl.org/acoli/conll#`.

References

1 Frank Abromeit and Christian Chiarcos. Automatic Detection of Language and Annotation Model Information in CoNLL Corpora. In Maria Eskevich, Gerard de Melo, Christian Fäth, John P. McCrae, Paul Buitelaar, Christian Chiarcos, Bettina Klimek, and Milan Dojchinovski, editors, *2nd Conference on Language, Data and Knowledge (LDK 2019)*, volume 70 of *Open-Access Series in Informatics (OASIcs)*, pages 23:1–23:9, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

2 Alan Akbik, Laura Chiticariu, Marina Danilevsky, Yunyao Li, Shivakumar Vaithyanathan, and Huaiyu Zhu. Generating High Quality Proposition Banks for Multilingual Semantic Role Labeling. In *Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 397–407, 2015.

3 Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data – The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

4 Christian Chiarcos. A Generic Formalism to Represent Linguistic Corpora in RDF and OWL/DL. In *Proc. of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 3205–3212. ELRA, 2012.

5 Christian Chiarcos. POWLA: Modeling Linguistic Corpora in OWL/DL. In Elena Simperl, Philipp Cimiano, Axel Polleres, Oscar Corcho, and Valentina Presutti, editors, *The Semantic Web: Research and Applications*, pages 225–239, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[8] `https://www.w3.org/community/ld4lt/`, `https://github.com/ld4lt/linguistic-annotation`

[9] `https://doi.org/10.5281/zenodo.4361476/`

6 Christian Chiarcos, Kathrin Donandt, Hasmik Sargsian, M Ionov, and J Wichers Schreur. Towards llod-based language contact studies. a case study in interoperability. In *Proc. of the 6th Workshop on Linked Data in Linguistics (LDL)*, 2018.

7 Christian Chiarcos and Christian Fäth. CoNLL-RDF: Linked Corpora Done in an NLP-Friendly Way. In Jorge Gracia, Francis Bond, John P. McCrae, Paul Buitelaar, Christian Chiarcos, and Sebastian Hellmann, editors, *Language, Data, and Knowledge*, pages 74–88, Cham, Switzerland, 2017. Springer.

8 Christian Chiarcos and Christian Fäth. 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, 2019.

9 Christian Chiarcos and Christian Fäth. Graph-Based Annotation Engineering: Towards a Gold Corpus for Role and Reference Grammar. In *2nd Conference on Language, Data and Knowledge (LDK-2019)*, pages 9:1–9:11. OpenAccess Series in Informatics, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, Germany, 2019.

10 Christian Chiarcos and Luis Glaser. A Tree Extension for CoNLL-RDF. In *Proc. of the Twelfth International Conference on Language Resources and Evaluation (LREC-2020)*, pages 7161–7169, Marseille, France, 2020. ELRA.

11 Christian Chiarcos, Ilya Khait, Émilie Pagé-Perron, Niko Schenk, Christian Fäth, Julius Steuer, William Mcgrath, Jinyan Wang, et al. Annotating a low-resource language with llod technology: Sumerian morphology and syntax. *Information*, 9(11):290, 2018.

12 Christian Chiarcos, Benjamin Kosmehl, Christian Fäth, and Maria Sukhareva. Analyzing Middle High German syntax with RDF and SPARQL. In *Proc. of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, pages 4525–4534, Miyazaki, Japan, 2018.

13 O. Christ. A modular and flexible architecture for an integrated corpus query system. In *Papers in Computational Lexicography (COMPLEX-1994)*, page 22–32, Budapest, Hungary, 1994.

14 Philipp Cimiano, Christian Chiarcos, John P. McCrae, and Jorge Gracia. *Linguistic Linked Data: Representation, Generation and Applications*. Springer International Publishing, Cham, 2020.

15 Souripriya Das, Seema Sundara, and Richard Cyganiak. R2RML: RDB to RDF Mapping Language. W3C Recommendation. `https://www.w3.org/TR/r2rml`, 2012.

16 Stefan Evert and Andrew Hardie. Twenty-first Century Corpus Workbench: Updating a Query Architecture for the New Millennium. In *Proc. of the Corpus Linguistics 2011 Conference*, pages 1–21, Birmingham, UK, 2011.

17 Antske Fokkens, Aitor Soroa, Zuhaitz Beloki, Niels Ockeloen, German Rigau, Willem Robert van Hage, and Piek Vossen. NAF and GAF: Linking Linguistic Annotations. In *Proc. of the Tenth Joint ISO-ACL SIGSEM Workshop on Interoperable Semantic Annotation*, pages 9–16, 2014.

18 Christian Fäth, Christian Chiarcos, Björn Ebbrecht, and Maxim Ionov. Fintan – Flexible, Integrated Transformation and annotation eNgineering. In *Proc. of the Twelfth International Conference on Language Resources and Evaluation (LREC-2020)*, pages 7212–7221, Marseille, France, 2020. ELRA.

19 A. Ghiran and R. A. Buchmann. Semantic Integration of Security Knowledge Sources. In *Twelfth International Conference on Research Challenges in Information Science (RCIS-2018)*, pages 1–9, 2018.

20 Noori Haider and Fokhray Hossain. CSV2RDF: Generating RDF Data from CSV File Using Semantic Web Technologies. *Journal of Theoretical and Applied Information Technology*, 96(20):6889–6902, 2018.

21 Sebastian Hellmann, Jens Lehmann, Sören Auer, and Martin Brümmer. Integrating NLP Using Linked Data. In Camille Salinesi, Moira C. Norrie, and Óscar Pastor, editors, *Advanced Information Systems Engineering*, volume 7908, pages 98–113. Springer Berlin Heidelberg, 2013.

22 Eero Antero Hyvönen, Petri Leskinen, Minna Tamper, and Jouni Antero Tuominen. Semantic National Biography of Finland. In Eetu Mäkelä, Mikko Tolonen, and Jouni Tuominen, editors, *Proc. of the DHN 2018*, CEUR Workshop Proceedings, pages 372–385, International, 2018. CEUR Workshop Proceedings.

23 N. Ide and L. Romary. International Standard for a Linguistic Annotation Framework. *Natural language engineering*, 10(3-4):211–225, 2004.

24 Maxim Ionov, Florian Stein, Sagar Sehgal, and Christian Chiarcos. cqp4rdf: Towards a suite for rdf-based corpus linguistics. In *European Semantic Web Conference*, pages 115–121. Springer, 2020.

25 ISO. Language Resource Management - Linguistic Annotation Framework (LAF). Standard, International Organization for Standardization, Geneva, 2012. Project leader: Nancy Ide.

26 Adam Kilgarriff, Vít Baisa, Jan Bušta, Miloš Jakubíček, Vojtěch Kovář, Jan Michelfeit, Pavel Rychlý, and Vít Suchomel. The Sketch Engine: Ten Years On. *Lexicography*, 1(1):7–36, 2014.

27 Christo Kirov, Ryan Cotterell, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sebastian J Mielke, Arya D McCarthy, Sandra Kübler, et al. UniMorph 2.0: Universal Morphology. In *Proc. of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, pages 1868–1873, 2018.

28 Francesco Mambrini and Marco Passarotti. Linked open treebanks. interlinking syntactically annotated corpora in the lila knowledge base of linguistic resources for latin. In *Proc. of the 18th International Workshop on Treebanks and Linguistic Theories (TLT, SyntaxFest 2019)*, pages 74–81, 2019.

29 Francesco Mambrini and Marco Passarotti. Linked Open Treebanks. Interlinking Syntactically Annotated Corpora in the LiLa Knowledge Base of Linguistic Resources for Latin. In *Proc. of TLT, SyntaxFest 2019*, pages 74–81, Paris, France, 2019. Association for Computational Linguistics.

30 Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

31 Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proc. of the Tenth International Conference on Language Resources and Evaluation (LREC-2016)*, pages 1659–1666, 2016.

32 Emilio Rubiera, Luis Polo, Diego Berrueta, and Adil El Ghali. TELIX: An RDF-based Model for Linguistic Annotation. In *Extended Semantic Web Conference*, pages 195–209. Springer, 2012.

33 Robert Sanderson, Paolo Ciccarese, and Benjamin Young. Web Annotation Data Model. Technical report, W3C Recommendation, 2017. URL: `https://www.w3.org/TR/annotation-model/`.

34 Erik F Sang and Sabine Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. *arXiv preprint*, 2000. `arXiv:cs/0009008`.

35 Helmut Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proc. of International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK, 1994.

36 Minna Tamper, Petri Leskinen, Kasper Apajalahti, and Eero Hyvönen. Using biographical texts as linked data for prosopographical research and applications. In *Euro-Mediterranean Conference*, pages 125–137. Springer, 2018.

37 Marc Verhagen, Keith Suderman, Di Wang, Nancy Ide, Chunqi Shi, Jonathan Wright, and James Pustejovsky. The LAPPS Interchange Format. In *International Workshop on Worldwide Language Service Infrastructure*, pages 33–47. Springer, 2015.

38 Karin Verspoor and Kevin Livingston. Towards Adaptation of Linguistic Annotations to Scholarly Annotation Formalisms on the Semantic Web. In *Proc. of the Sixth Linguistic Annotation Workshop*, pages 75–84, 2012.