Proceedings of the 7th Workshop on Linked Data in Linguistics (LDL-2020), pages 36–44
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

Annohub – Annotation Metadata for Linked Data Applications

Frank Abromeit, Christian Fäth, Luis Glaser

Applied Computational Linguistics Lab, Goethe University Frankfurt, Germany
{abromeit,faeth}@em.uni-frankfurt.de, lglaser@informatik.uni-frankfurt.de

Abstract

We introduce a new dataset for the Linguistic Linked Open Data (LLOD) cloud that will provide metadata about annotation and language information harvested from annotated language resources like corpora freely available on the internet. To our knowledge annotation metadata is not provided by any metadata provider, e.g. linghub, datahub or CLARIN so far. On the other hand, language metadata that is found on such portals is rarely provided in machine-readable form, especially as Linked Data. In this paper, we describe the harvesting process, content and structure of the new dataset and its application in the Lin|gu|is|tik portal, a research platform for linguists. Aside from that, we introduce tools for the conversion of XML encoded language resources to the CoNLL format. The generated RDF data as well as the XML-converter application are made public under an open license.

Keywords: LLOD, OLiA, CoNLL, Tiger-XML

1. Motivation

Over the past decade, an ever growing amount of linguistic resources has become available on the web under an open license. The Linguistic Linked Open Data (LLOD) cloud[1] currently encompasses not only annotated corpora and dictionaries, but also terminological repositories, ontologies and knowledge bases. However, despite the efforts of improving interoperability and interconnection of resources by using semantic web vocabularies and technologies, many resources are still heterogeneously annotated and intransparently labeled with regard to their compatibility. This problem is by no means limited to LLOD but applies to machine readable language resources in general. Metadata repositories like META-SHARE[2] , CLARIN centers[3] or DataHub[4] lack information about applicable annotation schemes. As for language metadata, language encoding standards vary across different metadata providers[5], and in addition metadata is not always provided as linked data.

With *Annohub* (annotation hub) we tackle this deficit by creating a collection of languages and annotation schemes used in existing language resources, and thus aim to augment existing metadata repositories. Annohub therefore utilizes classification schemes supported by and linked to the thesaurus of the Bibliography of Linguistic Literature (BLL).[6] This encompasses the Ontologies of Linguistic Annotation (OLiA) (Chiarcos and Sukhreva, 2015) and its respective Linking Models for compatibility with a large amount of linguistic annotation schemes (Dimitrova et al., 2016), and also Glottolog (Nordhoff and Hammarström, 2011) and lexvo (de Melo, 2015) as supported language identifiers (Dimitrova et al., 2018).

In previous work (Abromeit and Chiarcos, 2019) we focused on the analysis of language resources available in tab-separated column formats, a de-facto standard for annotated corpora as popularized as part of the CoNLL Shared Tasks. In the paper we describe the extension of our analysis to text resources encoded in RDF and XML formats thereby introducing tools that can be used for transforming XML language resources into the CoNLL format.

Finally, we discuss how the harvested metadata is integrated into the Lin|gu|is|tik portal (Chiarcos et al., 2016), a research platform for linguists funded by the DFG project *Fachinformationsdienst Linguistik* (FID) and hosted at the University Library Frankfurt. A special focus is put on our continued efforts on mapping BLL language identifiers to Glottolog and lexvo. Both, the Annohub RDF dump and the BLL linking models are available at `https://www.linguistik.de/de/lod/`. The XML-CoNLL converter application can be found at `https://github.com/acoli-repo/xml2conll`.

2. Finding annotated language resources

Our premier source of metadata for existing language resources is the linghub RDF dump[7] that contains over 200,000 linguistic resources. Additionally we query various CLARIN centers[8] via the OAI protocol[9], but also manually collect metadata from providers such as `http://opus.nlpl.eu/` and others. All harvested resource metadata is stored in a database which is used to keep track of already processed resources. Of course, duplicate entries are a problem especially when a resource is available at different locations. We did not tackle this problem in detail yet. It is planned, however, to subsequently integrate new metadata entries, that we discovered, into the linghub portal.

2.1. Document classification

On the basis of the collected metadata, we identify language resources which could contain annotated text, such as corpora, lexica and also linguistic metadata as found in

[1] The LLOD cloud (`http://linguistic-lod.org`) is an integral part of the general Linked Open Data cloud under `https://www.lod-cloud.net`

[2] `http://www.meta-share.eu` and `http://www.meta-net.eu`

[3] `https://www.clarin.eu/`

[4] `https://datahub.io`

[5] e.g. different ISO639 encodings, lexvo URLs or even plain text language descriptors

[6] `https://data.linguistik.de/bll/index.html`

[7] `http://linghub.org/download`

[8] `https://centres.clarin.eu/restxml/`

[9] `https://www.openarchives.org/OAI/openarchivesprotocol.html`

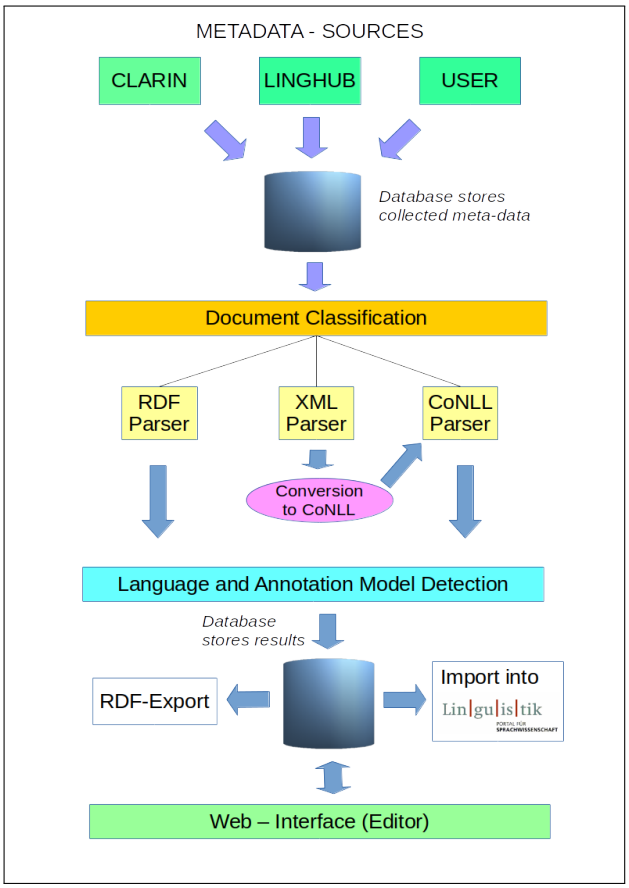

Figure 1: Annohub architecture

ontologies. For the identification of relevant files we rely on standard techniques for content type detection like (html/mime) type, file extensions and also employ the Apache Tika content analysis toolkit[10]. A considerable amount of linguistic resources is provided in archive formats[11]. Archives can include large numbers of files and we observed archives with up to 100,000 files. Since processing all files in such cases would take too long, we sample at maximum 100 files (depending on the file type) for the subsequent analysis. Currently, we support RDF, CoNLL[12] [13], CoNLL-RDF (Chiarcos and Fäth, 2017) and XML type documents with certain limitations (see chapter 6.1.)

2.2. Document processing

Linghub resources are automatically processed by first querying the linghub RDF dump via SPARQL and then feeding RDF, CONLL, XML and archive resource types into the Annohub NLP pipeline (see fig. 1). Since the automatic processing of language resources listed in CLARIN centers is not implemented yet, we currently only use the CLARIN resource metadata (e.g. author, title, etc.), that we harvest via the OAI protocol to augment manually found language resources. These can be processed by means of the Annohub web application with the respective *download URL* or by csv file import. Additionally, processing can be triggered from the command-line interface.

3. Extraction of annotation information

We are mainly dealing with annotations from the field of syntax and morphology. Word annotations (e.g. part-of-speech tags) usually take the form of string tokens, which have been assigned to a word either automatically by a tool, or manually by a researcher. Alternatively, ontology classes (e.g. `http://www.lexinfo.net/ontology/2.0/lexinfo#adverb`) typically found in RDF corpora are used to annotate words. Feature annotations (e.g. *number*), as found in CoNLL data [14], involve a feature name and a feature value (e.g. Number=Plur). Finally, dependency information about the syntactic relation between words[15] in a sentence is relevant to us. In order to extract annotations from texts, specific parsers for CoNLL, XML and RDF data have been developed.

3.1. CoNLL annotation extraction

A CoNLL file has dedicated columns for e.g. part-of-speech, morphological features and dependency information. However, since CoNLL is a format family with distinct dialects mostly originating from specific CoNLL shared tasks, the number of columns and also the ordering of columns in a CoNLL file is not standardized. Therefore, first the columns storing POS-tag, feature and dependency information have to be determined. For each such column the set of occurring tags is fed into our model detection component (see fig. 1).

3.2. XML annotation extraction

In a preprocessing step we convert XML files to the CoNLL format (see chapter 6.). Thus, the extraction process is the same as for CoNLL files.

3.3. RDF annotation extraction

Extracting linguistic annotations from RDF language resources is a more complex task since such resources often contain multiple types of annotations at the same time, for example for syntax, semantics and pragmatics. Another problem is that, although RDF vocabularies like Lexinfo[16], NIF[17] or OntoLex-Lemon[18] exist which have been specifically designed to model syntax and morphology, researchers sometimes use their own proprietary RDF vocabularies.
One way to implement the extraction process would be to get the object values for selected RDF predicates typically

[10] `https://tika.apache.org/`
[11] gzip,zip,tar,rar to name a few
[12] `https://universaldependencies.org/format.html`
[13] CoNLL-U, CoNLL-X and other TSV formats
[14] `https://universaldependencies.org/u/feat/index.html`
[15] `https://universaldependencies.org/u/dep/index.html`
[16] `https://www.lexinfo.net/ontology/2.0/lexinfo.owl`
[17] `https://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core/nif-core.html`
[18] `https://www.w3.org/2016/05/ontolex`

used for annotating (e.g. `http://www.lexinfo.net/ontology/2.0/lexinfo#partOfSpeech`) via SPARQL queries[19] from an RDF file. However, in this approach the list of RDF predicates known to the algorithm would limit the results. To avoid this limitation we approach the extraction problem in a more generic way by sampling object values from the set of *all* RDF predicates that occur in an RDF file, (see algorithm 1) and comparing them with the list of annotations defined in the OLiA[20] ontologies (Chiarcos and Sukhreva, 2015). The OLiA ontologies provide a formalized, machine-readable view on linguistic annotations for more than 75 different language varieties. They cover morphology, morphosyntax, phrase structure syntax, dependency syntax, aspects of semantics, and recent extensions to discourse, information structure and anaphora, all of these are linked with an overarching reference terminology module. Furthermore OLiA includes several multi-lingual or cross-linguistically applicable annotation models such as the Universal Dependencies (77 languages), EAGLES (11 European languages) and Multext-East (16 Eastern European and Near Eastern languages).

Algorithm 1 *RDF annotation detection*

1: **Input** : The set of all different RDF predicates in a file
2: **For** each RDF predicate ***p*** :
3: Take a sample of *n* object values *o* from triples (s p o)
4: **If** *o* is of type *literal* :
5: Try to match *o* against the set of known annotation tags (in OLiA linking models)
6: **Else** :
7: Try to match *o* against the set of known annotation classes (in OLiA annotation models)
8: **Repeat** the algorithm with the set of predicates with values that could be matched to tags or classes in OLiA as *input* and omit in the second pass step 3 in order to retrieve all annotation matchings.

4. Annohub dataset

The RDF data model for Annohub is depicted in fig. 2. For modeling language resources, we utilize commonly used RDF vocabularies including DCAT[21], Dublin Core [22], DCMI Metadata Terms[23] and PROV[24]. Each language resource is modeled as a DCAT dataset. It includes general resource information like title, author, etc. that we harvest automatically from the metadata providers linghub[25] and CLARIN[26]. The metadata for language resources that were collected manually was taken from the websites of the resource providers. Each annotated text file that comes with

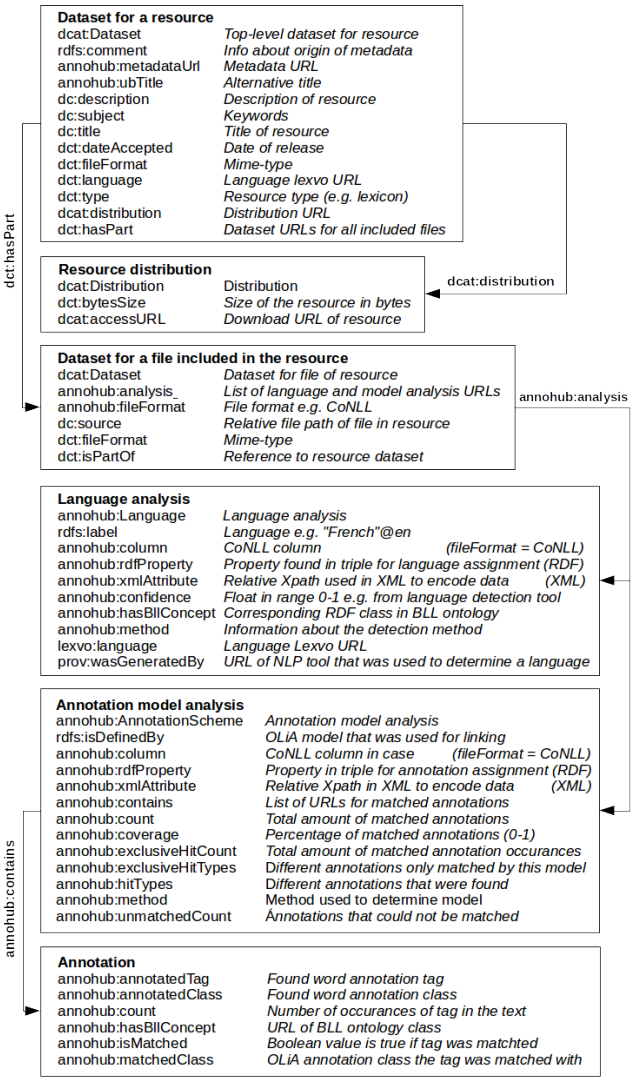

Figure 2: Annohub RDF data model

a resource is represented as a separate dataset. It contains general information like file type, file size, etc. The analysis results of each file comprise the detected languages and identified annotations from the field of syntax and morphology. By application of the Ontologies of Linguistic Annotations (OLiA) it is also possible to detect the annotation schemes (tag sets) that were used.

4.1. Annotation model analysis

The annotation model analysis is similar for all processed document types. It is exemplarily described here for CoNLL files.

Annotation model analysis for CoNLL files A CoNLL file has dedicated columns for part-of-speech, morphological features and dependency information. Generally, for each of these, a specific annotation model is used. By definition, a CoNLL column can contain annotations only from a single annotation model. A CoNLL file dataset therefore may link to several *annohub:AnnotationScheme* instances, but only to one for each column.

[19] `https://www.w3.org/TR/sparql11-overview/`
[20] `http://purl.org/olia`
[21] `https://www.w3.org/TR/vocab-dcat/`
[22] `http://purl.org/dc/elements/1.1/`
[23] `http://purl.org/dc/terms/`
[24] `http://www.w3.org/ns/prov#`
[25] `http://linghub.org/`
[26] `https://centres.clarin.eu/restxml/`

CoNLL model analysis results In the property *annohub:column* the considered column of a file is marked. The information of what annotation scheme (e.g. Universal-Dependencies-part-of-speech[27]) has been detected in a column is included with the *annohub:isDefinedBy* property. Its value is a URL that points to an OLiA linking model[28] (e.g. `http://purl.org/olia/ud-pos-link.rdf`). This provides a mapping of all identified tags to annotation classes in the OLiA ontology.
Part-of-speech annotations are highly ambiguous across different annotation models. For example *DT* is equally used in 6 different annotation models[29] to declare a word as *Determiner*. The algorithm used to determine a best fitting annotation scheme is fully described in (Abromeit and Chiarcos, 2019) chapter 3 (*Automated detection of annotation models)*. It is also used for RDF document types. The results of this algorithm comprise several statistical parameters that can be used to assess the quality of the model analysis results which are provided in the *annohub:annotationScheme* class, and are described as follows. This statistical data is also available in the Annohub editor application and is helpful for the verification of the detected annotation schemes. A detailed description of the editor application is also presented in the above mentioned publication.

- *annohub:exclusiveHitTypes* is the number of different tags that could be matched only by that model
- *annohub:hitTypes* is the number of different tags that could be matched
- *annohub:unmatchedTags* is the number of unmatched tags
- *annohub:coverage* is the value of matched/unmatched tags which is a float value (0-1)
- *annohub:count* is the total number of occurrences of matched tags
- *annohub:exclusiveHitCount* is the number of occurrences of matched tags (but only for *exclusiveHitTypes*)

Furthermore the property *annohub:method* stores information on how the model assignment was achieved. It is *ANNOMODEL* by default and *SELECTION* if it was changed manually by means of the Annohub editor.
All properties but *annohub:column* also apply to the model analysis for **RDF** and **XML** files. For these file types instead *annohub:rdfProperty* and *annohub:xmlAttribute* are used to specify the location where an annotation was found. Detailed information about each matched tag (or annotation URL, as found in RDF files) and its counterpart in OLiA is stored in instances of *annohub:Annotation* by using *annohub:annotatedTag*, *annohub:annotatedClass* or *annohub:matchedClass*. Finally *annohub:hasBllConcept* provides a link to a class in the BLL ontology that is related to a word's annotation (see chapter 5.).

4.2. Language analysis

The language analysis is different for each of the examined data formats.

4.2.1. Language analysis for RDF files

In RDF files there are several sources for language information. First language tagged literals can provide an ISO-639-1 (2-letter) code. Secondly language information can be encoded with specific RDF predicates where the language info is a URL (e.g. `http://lexvo.org/id/iso639-3/nld`) or a literal value that contains a ISO language code. We employ standard RDF predicates that are widely used to encode language information taken from the Dublin Core, OntoLex-Lemon and lexvo ontologies[30] and discard cases where the language info is encoded as plain text (e.g. *English, Old (ca. 450-1100))*. The extracted language information is finally normalized to a ISO639-3 representation.

RDF language analysis results All detected languages are encoded as lexvo-URL[31] with the property *lexvo:language*. Additionally the full (english) language name taken from the ISO639-3 code table [32] is provided in *rdfs:label*. In the property *annohub:confidence* a probability for a detected language is stored. Because the language info in RDF files is explicitly given as URL or ISO code its value is 1.0. The property *annohub:method* stores where the language info originates from. It's a string constant which is *LANGTAG* for tagged literals and *LANGPROP* for languages that were encoded with a RDF property. The value of *annohub:rdfProperty* is simply *p* in the RDF triple (s p o) where o stores language information. That property can also be used to determine the type of a resource (e.g. `http://www.w3.org/ns/lemon/lime#language`) indicates a document of type lexicon). Finally *annohub:hasBllConcept* links to a class in the BLL ontology[33] that relates to a language (see also chapter 5.).

4.2.2. Language analysis for CoNLL files

In previous work (Abromeit and Chiarcos, 2019) we performed a quantitative analysis of the language detection procedure for the CoNLL format. For the language detection we use the *Optimaize*[34] n-gram based language classification tool. As input we choose a fixed number of *k*[35]

[27] `https://universaldependencies.org/u/pos/index.html`

[28] as documented at `http://purl.org/olia`

[29] e.g. `http://purl.org/olia/brown.owl#DT` as well as for the GENIA, QTag, Mamba and Penn Treebank annotation schemes

[30] 11 properties taken from the Dublin Core, OntoLex-Lemon and lexvo ontologies, e.g. `http://purl.org/dc/terms/language`, `http://www.w3.org/ns/lemon/lime#language`, `http://lexvo.org/ontology#iso639P3PCode`)

[31] `http://lexvo.org/ontology`

[32] `https://iso639-3.sil.org/code_tables/download_tables`

[33] `https://data.linguistik.de`

[34] `https://github.com/optimaize/language-detector`

[35] We choose k=15 for scalability reasons. Increasing k might improve results but has not been tested

sample sentences from the WORD (LEMMA) column in a CoNLL file. Then we run the language classification on each sentence and select the language as winner *Lw* which was detected for the majority of sentences. The probability *Lp* for a language is then

$$\frac{\#sample\ sentences\ where\ Lw\ was\ detected}{\#sample\ sentences}$$

In situations where no majority winner was found one of the best languages is randomly selected. For a discussion on misclassification errors we refer to the mentioned evaluation paper above.

CoNLL language analysis results The detected WORD (LEMMA) columns of a CoNLL file are stored with the *annhub:column* property. Languages are encoded in the same manner as for RDF files where the language probability *Lp* is saved in *annohub:confidence* as a float value in the range [0-1] and *annohub:method* describes how the language was determined. Its value is either *LANGPROFILE* in case the *Optimaize* tool provided the result or *SELECTION* if it was assigned manually by means of the Annohub editor. In order to allow other language detection tools the detection tool info is provided with *prov:wasGeneratedBy* which is in this case the URL of the *Optimaize* language-detector library. Finally *annohub:hasBllConcept* stores a URL which is a class from the BLL ontology[36] that is related to a language (see chapter 5. for more details).

4.2.3. Language analysis for XML files

Since XML files are converted to CoNLL files before processing the only difference to the CoNLL language analysis is in the *annohub:xmlAttribute* property. It stores a relative XPath that describes the location of the word/lemma information in an XML document.

5. Integration of Annohub and the Lin|gu|is|tik portal

After harvesting linguistic resources of heterogeneous sources and formats and extracting their annotations and language information, an additional linking step is performed to make them not only available through Annohub but also searchable and browsable within the Lin|gu|is|tik portal. This process is relying on ontological links between BLL index terms and other repositories for linguistic classifications, i.e. OLiA (Chiarcos and Sukhreva, 2015), Glottolog (Nordhoff and Hammarström, 2011) and lexvo (de Melo, 2015). In this chapter we give a brief overview on how these links are established.

5.1. Linking of linguistic annotations

In earlier efforts of the DFG-funded series of projects (Chiarcos et al., 2016) we had created a linking of the BLL-Thesaurus with OLiA for linguistic terminology, mostly on the field of syntax and morphosyntax. For this purpose the BLL thesaurus had been transformed to RDF in a two-step process. While the original thesaurus hierarchy is automatically exported in SKOS format on a regular basis, the BLL ontology is manually assessed and updated in order to improve usability and interoperability in the context of LLOD. Within the ontology, BLL index terms are rendered as OWL classes and then linked to OLiA classes through subclass relations.

5.2. Linking of language identifiers

Recently, the scope of BLL ontology has been extended to include the index terms for language identifiers and a linking to both Glottolog and lexvo. Our last release only included the language identifiers subsumed under the ontological class `Indo-European language identifiers` of which approximately 60% could be linked to at least one LLOD repository (Dimitrova et al., 2018). For the current version we also include terminology from non-Indo-European languages.

Family	both	lexvo	Glottolog	none	Total
Indo-Eur.	236	104	7	298	645
Other	1329	416	32	187	1964
Total	1565	520	39	485	2609

Table 1: Linking of Language Identifiers

Table 1 shows the current status of the linking which is, however, an ongoing effort and thus subject to change. While the status of the language identifiers in the Indo-European family is consistent with the last release, the non-Indo-European branch adds almost 2000 new classes of which approximately 90% are linked to at least Glottolog or lexvo. The higher coverage of links within this branch of the thesaurus stems from the BLL's focus on European languages. While the terminology on the Indo-European branch is more granularly organized and comprises a comprehensive set of dialects, language families and historical terms, the non-Indo-European branch mostly contains well-established identifiers for actual languages which have a one-to-one correspondence in other repositories.
Both the revised version of the BLL ontology and the language linking are available under `https://www.linguistik.de/de/lod/`.

6. Support for XML language resources

The XML formalism has been widely used for the representation of linguistic resources. This is partly due to the fact that NLP tools use this format for input and output. Also, popular formats like TEI[37] are built on it. Today, a substantial part of available corpora data is still only available in XML formats[38] although new Linked Open Data formats have been introduced. We have built tools that can be used to convert corpus data from XML to the CoNLL[39] format that is widely used in the community. Furthermore, for the

[36] `https://data.linguistik.de`

[37] `https://github.com/TEIC/TEI`

[38] Examples are `http://opus.nlpl.eu/` and `https://spraakbanken.gu.se/`

[39] `https://universaldependencies.org/format.html`

CoNLL format, tools for the conversion into the RDF format are available[40].

6.1. Supported XML formats for CoNLL conversion

Stand-off annotation XML formats have annotations stored in separate files from the text data (e.g. ATLAS (Bird and Liberman, 2001), PAULA XML (Dipper, 2005) or GrAF (Ide and Suderman, 2007). We do not support these formats at the moment because this would involve a reconstruction of the data. Moreover we focus on XML resources that include annotations and the raw text data together within a single file like Tiger-XML (W. Lezius and Gerstenberger, 2002) does. In Tiger XML[41] a graph element contains a list of terminal nodes that represent the words of a sentence. Each terminal element has word information like reference_id, part-of-speech, morphological features, word and lemma. However, there is no defined standard with respect to the naming of XML elements and attributes. In practice, a class of XML documents can be identified that shares the encoding formalism of Tiger-XML. For parsing the XML several issues have to taken into account :

1. XML documents can include optional elements that store text structure information like chapter, paragraph, chunk
2. All names of XML elements and attributes are user-defined
3. All attributes of a terminal element (e.g. lemma, POS, etc.) are optional

Instead of writing a specific XML parser for each document class we developed a template-based method that uses a description of a documents XML structure. This description (*template*) is then used by our XML parser application to extract the annotated content.

6.2. Template based XML-CoNLL conversion

With the template-based conversion we are able to perform a lossless transformation from the XML to the CoNLL format. It can be applied, if the following requirements are met:

1. The XML represents the concept of a sentence
2. The XML represents the concept of a word/token
3. All word level annotations are represented as descendants of the word node
4. The name of the annotations are given as attribute names / the annotations are not reified.

Intuitively, one writes a set of rules (template) in a JSON file that is used as input together with an XML file to generate CoNLL from it. A template provides a mapping from the data found at a (relative) XPath in the XML

```
<?xml version="1.0" encoding="utf-8"?>
<document>

<CHAPTER ID="1"><P id="1">
<s id="1">
 <w tree="NC" lem="aprobación" id="w1.1">Aprobación</w>
 <w tree="PDEL" lem="del" id="w1.2">del</w>
 <w tree="NC" lem="acta" id="w1.3">Acta</w>
 <w tree="PREP" lem="de" id="w1.4">de</w>
 <w tree="ART" lem="el" id="w1.5">la</w>
 <w tree="NC" lem="sesión" id="w1.6">sesión</w>
 <w tree="ADJ" lem="anterior" id="w1.7">anterior</w>
</s></P><SPEAKER ID="1" NAME="La Presidenta"><P id="2">
<s id="2">
 <w tree="ART" lem="el" id="w2.1">El</w>
 <w tree="NC" lem="acta" id="w2.2">Acta</w>
 <w tree="PREP" lem="de" id="w2.3">de</w>
...
```

Figure 3: Example XML file

document to a CoNLL column where this data should be placed. In addition to this column mapping, a template specifies the word nodes and the sentence boundaries in the XML format. This has two consequences: firstly, the resulting CoNLL will have a valid separation of sentences with a newline. Secondly, the sentence boundaries allow the XML to be streamed sequentially. Thus arbitrary XML file sizes can be read, because a XML-DOM tree is only created for each sentence and not the entire document.

Specification of a template A template definition contains the following information:

1. `id`: An arbitrary identifier for the template. It is used for the template matching algorithm
2. `sentencePath`: The name of the XML nodes that contain a single sentence as their subtree
3. `wordPath`: XPath expression that will return a list of nodes. Each column row will represent data relative to a single node in that list
4. `columnPaths`: XPath expressions relative to a single node contained in the list specified in 3. Each of these XPath expressions is assigned to a specific column in the resulting CoNLL
5. `description`: Provide a description of this template, e.g. what corpus family this template was tailored to. Useful for debugging. (optional)
6. `featurePaths`: Identical to 4 in structure. Resulting values will be conflated into a single column with the `key1=value1|key2=value2` style CoNLL-U uses for morphosyntactic features. (optional)

Template matching algorithm In our processing workflow (fig.1) documents which have been classified as XML documents are fed into our processing pipeline. However at this point it is not clear if an XML document contains any useful linguistic content. In order to rule out useless content we check the document's XML structure against a set of given XML-conversion template definitions. Algorithm 2 computes a score that is then used to decide which XML

[40] https://github.com/acoli-repo/conll-rdf

[41] http://www.coli.uni-saarland.de/projects/salsa/salto/doc/html/node55.html

```
0       w1.1        aprobación        Aprobación
1       w1.2        del        del
2       w1.3        acta        Acta
3       w1.4        de        de
4       w1.5        el        la
5       w1.6        sesión        sesión
6       w1.7        anterior        anterior

0       w2.1        el        El
1       w2.2        acta        Acta
2       w2.3        de        de
...
```

Figure 4: Resulting CoNLL for XML depicted in fig. 3

```
{
  "id" : "8",
  "sentencePath" : "s",
  "wordPath" : "//w",
  "columnPaths" : {
    "id" : "@id",
        "lem" : "@lem",
        "token" : "text()"
  }
}
```

Figure 5: Template for XML in fig. 3 to produce CoNLL in fig. 4

documents will be passed to the next processing stage (Language and Annotation Model detection). We define the *best fitting template* for a given XML file as the template with the highest recall score.

Algorithm 2 *Template XML-CoNLL conversion*

1: **For** each sample sentence **s**:
2: Create the DOM tree **d** for **s**
3: **For** each conversion template **t**:
4: Compare all attribute names found in **d** with the attribute names specified in the `columnPaths` field of **t**
5: Calculate recall, precision and accuracy for **t**
6: Sort all templates **t** in descending order by recall, precision, accuracy
7: Output the top-most template as best matching template

Limitations The rule-based approach requires human input and is limited to the set of available templates. So it cannot identify documents with an unknown XML structure. It is highly accurate and the best solution if a language resource needs to be converted that has many XML documents that all share the same structure. It also can produce acceptable results if the best matching template is not a perfect match. Finally, we only support XML dialects that do not reify their annotations: E.g. encoding the first lemma in fig. 3 as `<w annotation="lem" value="aprobación">` would break the algorithm.

6.3. Generic XML-CoNLL conversion

In the following, we describe a algorithm that solves some of the problems of the template approach. It is well suited to a scenario were a stream of XML documents enters our analysis pipeline where

- the structure of XML documents is unknown
- it can be assumed that many XML documents do not contain useful content
- the amount of documents is much larger than for the template-based approach

Generic matching algorithm The primary goal of algorithm 3 is to find the element in a XML document that contains the most annotation information. As a first step a list of all available XML nodes which are unique up to a list index is computed [42]. In the following a example computation of the algorithm is described for the XML source in fig.3. The results are depicted in table 2. A row in table 2 stores all relative XPath of one unique XML node. In the first column the relative XPath, in the second column the value (attribute or text value) and in the third column the score (matching annotations) are depicted. From the table it can be seen that only the values of `document/CHAPTER/P/s/w//@tree` could be matched with known annotations. Although only the attribute `document/CHAPTER/P/s/w//@tree` from elements `document/CHAPTER/P/s//w` could be matched we conclude that also the other attributes like `document/CHAPTER/P/s/w//@lemma` contain *word*, *lemma* or *dependency* information.[43] Hence we extract all attribute and text values from XML elements `document/CHAPTER/P/s//w`. These are then used to build a CoNLL file from it. Since we only use a small portion of the XML document for the detection process the method can be used to quickly rule out XML documents that do not contain any annotation information at all. We have found that taking 10 sample values (see algorithm 3 step 2) is sufficient for this purpose.

Algorithm 3 *Generic XML-CoNLL conversion*

1: Generate the list of unique XML nodes
2: Take **n** sample values[44] for each of the nodes in 1.
3: Filter nodes that have only numeric values
4: Compute a *score* which is simply the sum of all different values of one XML node that could be matched against the set of known annotations
5: The XML node with the highest score provides the most annotation information. Extract all attribute and text values from XML elements that are represented by a relative XPath for that node

[42] A list of XML child nodes that have the same parent node is represented by a relative XPath

[43] In general, attribute names don't have to be meaningful (like here : *lemma*)

[44] XML attribute or text value

Relative XPath for xml nodes	Found values	Score
document		0
document/CHAPTER//@id	1,2	0
document/CHAPTER/P//@id	1,2	0
document/CHAPTER/P/s//@id	1,2	0
document/CHAPTER/P/s/w//@tree	**ART, NC**	2
document/CHAPTER/P/s/w//@lem	el,sesión	0
document/CHAPTER/P/s/w//@id	1,2	0
document/CHAPTER/P/s/w//text()	la,sesión	0
document/CHAPTER/SPEAKER//@id	1	0
document/CHAPTER/SPEAKER//@name	La Presidenta	0

Table 2: Example computation for alg. 3

7. Results

So far 2317 files from 1508 resources have been processed. Of these were 1572 RDF, 263 CoNLL and 482 XML files. Table 3 below shows in how many files at least one language or annotation model could be detected.

File type	Processed	Model found	Language found
RDF	1572	393 (25%)	503 (32%)
CoNLL	263	263 (100%)	254 (97%)
XML	482	350 (73%)	375 (78%)
Total	2317	1006 (43%)	1132 (49%)

Table 3: Results by file type

Obviously, for the CoNLL format the percentage of files that yielded results is nearly 100%. This can be explained by the fact that CoNLL files include usable data by default. For the few cases were no language could be detected a parse error might be the cause for the error, or the language info was missing in the CoNLL data. A large portion of the processed XML files (~75%) could be converted to the CoNLL format. Because most XML files were manually selected from language resource providers like `http://opus.nlpl.eu/` and `https://spraakbanken.gu.se`, this result is not unexpected and the error rate (~25%) can be explained with other XML files that were included in archive resources (e.g. zip, tar) together with the real data. On the other hand RDF resources were automatically collected and did not reveal as much usable data as we expected. In fact the RDF format is commonly used for semantic web data, perhaps even more than for LLOD data. In total, 22 different annotation models and 3855 different languages could be identified. In addition to that, we detected RDF namespaces for vocabularies of linguistic interest such as OLiA[45], GOLD[46], NIF[47], OntoLex-Lemon[48], UBY[49] and LexInfo[50]. However, these are not included in the Annohub RDF dump for now.

The results for files listed in table 3 are finally reviewed by a linguist with the Annohub editor application. It is used to correct wrong results and to select the resources that are published in the Annohub RDF dataset. Currently the dataset contains 609 different language resources that include 915 files. In table 4 the distribution for different resource types is shown.

	Corpus	Lexicon	Ontology	Total
Resources	391	214	4	609

Table 4: Language resource classification

For future releases we are planning to extend this classification to other resource types like wordnets, etc.

	linghub.org	CLARIN	USER	Total
Resources	128	77	404	609

Table 5: Overview of metadata providers

Table 5 shows the metadata providers for resources included in Annohub. At the date of writing, the *linghub* [51] portal is updated with new resources that have been collected by members of the Prêt-à-LLOD [52] working group. We are planning to include the results for these as well as more results for language resources listed on CLARIN in future releases.

8. Summary

We introduced a new LLOD dataset which provides annotation model and language information for publicly available language resources. By means of SPARQL queries this data can be linked with other existing LLOD resources, as we have shown by the use case of the Lin|gu|is|tik portal. Certainly, the availability of annotation metadata will enable other new Linked Data applications. Finally, the provided tools for the conversion of XML data to the CoNLL format also contribute to the LLOD cloud since other converters[53] for a later conversion to the CoNLL-RDF (Chiarcos and Fäth, 2017) format exist.

9. Acknowledgements

The research described in this paper was conducted in the context of the Specialized Information Service Linguistics (FID), funded by German Research Foundation (DFG/LIS, 2017-2019). The contributions of the second author were conducted with additional support from the Horizon 2020 Research and Innovation Action 'Prêt-à-LLOD', Grant Agreement number 825182. The authors would like to thank Christian Chiarcos for providing expert advice throughout the project. We would also like to thank Thorsten Fritze and Yunus Söyleyici for technical support and Vanya Dimitrova for helpful comments.

10. Bibliographical References

Abromeit, F. and Chiarcos, C. (2019). Automatic Detection of Language and Annotation Model Information in CoNLL Corpora. In Maria Eskevich, et al., editors, *2nd Conference on Language, Data and Knowledge (LDK*

[45] `http://purl.org/olia/olia.owl`
[46] `http://purl.org/linguistics/gold`
[47] `http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#`
[48] `http://www.w3.org/ns/lemon/ontolex#`
[49] `http://purl.org/olia/ubyCat.owl`
[50] `http://www.lexinfo.net/ontology/2.0/lexinfo`
[51] `http://linghub.org`
[52] `https://www.pret-a-llod.eu/`
[53] `https://github.com/acoli-repo/conll-rdf`

2019), volume 70 of *OpenAccess Series in Informatics (OASIcs)*, pages 23:1–23:9, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

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

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 Sukhreva, M. (2015). OLIA - Ontologies of Linguistic Annotation. *Semantic Web Journal*, 518:379–386.

Chiarcos, C., Fäth, C., Renner-Westermann, H., Abromeit, F., and Dimitrova, V. (2016). Lin|gu|is|tik: Building the Linguist's Pathway to Bibliographies, Libraries, Language Resources and Linked Open Data. In Nicoletta Calzolari (Conference Chair), et al., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 4463–4471, Portorož, Slovenia, May. European Language Resources Association (ELRA).

de Melo, G. (2015). Lexvo.org: Language-Related Information for the Linguistic Linked Data Cloud. *Semantic Web Journal*, 6:393–400.

Dimitrova, V., Fäth, C., Chiarcos, C., Renner-Westermann, H., and Abromeit, F. (2016). Building an Ontological Model of the BLL Thesaurus: First Steps Towards and Interface with the LLOD Cloud. In John P. McCrae, et al., editors, *Proceedings of the 5th Workshop on Linked Data in Linguistics: Managing, Building and Using Linked Language Resources (LDL-2016)*, pages 50–58, Portorož, Slovenia, May. European Language Resources Association (ELRA).

Dimitrova, V., Fäth, C., Chiarcos, C., Renner-Westermann, H., and Abromeit, F. (2018). Interoperability of Language-related Information: Mapping the BLL Thesaurus to Lexvo and Glottolog. In Nicoletta Calzolari (Conference chair), 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).

Dipper, S. (2005). Xml-based stand-off representation and exploitation of multi-level linguistic annotation. In *Berliner XML Tage 2005, Humboldt-Universität zu Berlin, 12. bis 14. September 2005*, pages 39–50.

Ide, N. and Suderman, K. (2007). GrAF: A graph-based format for linguistic annotations. In *Proceedings of the Linguistic Annotation Workshop*, pages 1–8, Prague, Czech Republic, June. Association for Computational Linguistics.

Nordhoff, S. and Hammarström, H. (2011). Glottolog/Langdoc: Defining dialects, languages, and language families as collections of resources. In Tomi Kauppinen, et al., editors, *Proceedings of the First International Workshop on Linked Science 2011*, pages 53–58, Bonn, Germany.

W. Lezius, H. B. and Gerstenberger, C. (2002). *Tiger-xml quick reference guide*. Technical report, IMS,University of Stuttgart.