

CHAPTER 6

FoLiA in Practice: The Infrastructure of a Linguistic Annotation Format

Maarten van Gompel^{a,c}, Ko van der Sloot^{a,c}, Martin Reynaert^{a,b,c}
and Antal van den Bosch^c

^aCentre for Language and Speech Technology, Radboud University, ^bTilburg Centre for Cognition and Communication, Tilburg University, ^cCentre for Language Studies, Radboud University

ABSTRACT

We present an overview of the software and data infrastructure for FoLiA, a **F**ormat for **L**inguistic **A**nnotation developed within the scope of the CLARIN-NL project and other projects. FoLiA aims to provide a single unified file format accommodating a wide variety of linguistic annotation types, preventing the proliferation of different formats for different annotation types. FoLiA is being developed in a bottom-up and practice-driven fashion. We have invested mainly in the creation of a rich infrastructure of tools that enable developers and end-users to work with the format. This work will present the current state of this infrastructure.

6.1 Introduction

CLARIN's aim is to deliver an infrastructure for researchers that work with language data and tools. This is impossible without agreeing on standards with regard to data formats. Standardisation is an important prerequisite for good interoperability between the many language tools that have emerged within and outside of the scope of the CLARIN project, and to ensure the various datasets released are usable in practice.

In the field, however, we often encounter an abundance of ad-hoc formats. We define ad-hoc formats to be data formats that are characterised by most of the following traits:

- They are only used once, often by one specific tool or for just one specific purpose;
- They are poorly formalised or not formalised at all, i.e. there is a lack of a formal schema and semantics;

How to cite this book chapter:

van Gompel, M, van der Sloot, K, Reynaert, M and van den Bosch, A. 2017. FoLiA in Practice: The Infrastructure of a Linguistic Annotation Format. In: Odijk, J and van Hessen, A. (eds.) *CLARIN in the Low Countries*, Pp. 71–81. London: Ubiquity Press. DOI: <https://doi.org/10.5334/bbi.6>. License: CC-BY 4.0

- They are poorly documented;
- They are often rigid and hard to extend.

The use of such ad-hoc formats can be considered the opposite of proper standardisation and is to be avoided in any large infrastructure project.

CLARIN adheres to the following principles when it comes to standardisation:

- Open standards are preferred over proprietary standards;
- Formats and protocols should be:
 - well documented
 - verifiable
 - proven (being used in practice);
- Text-based formats are (where possible) preferred over binary formats.

Fortunately, there are various initiatives for standardisation resulting in annotation formats that transcend the ad-hoc level, each with their own merit, and ours being of one of them. At the onset of CLARIN-NL, however, the Dutch and Flemish Natural Language Processing (NLP) community lacked such a proper standard with respect to linguistically annotated text, and ad-hoc formats were prevalent in the field. In the scope of CLARIN-NL project TTNWW (see chapter 7), the NWO project DutchSemCor, and the STEVIN project SoNaR, FoLiA (Format for Linguistic Annotation) was developed as a solution to accommodate the representational needs of these projects.

The aim of FoLiA is to provide a practical standard, following a generic paradigm, for the linguistic annotation of primarily written text. For this purpose, a wide variety of linguistic annotation types is supported.

In the current chapter, we intend to focus on the *practical* nature of the format, or rather, on the infrastructure that is built around the format, the software that supports it, and the ways in which it has been put to use in CLARIN and beyond. Section 6.3 will explain the philosophy behind FoLiA and its infrastructure.

Earlier work (van Gompel and Reynaert, 2013) addresses the motivation for the creation of FoLiA. In summary, FoLiA sprung from a limited corpus format used in the Dutch and Flemish NLP communities (Apperloo, 2006), at a time and place where a more comprehensive format was needed for various corpora and tools in development. Existing solutions often did not sufficiently meet the needs at the time, were not mature yet, or were simply not well known.

FoLiA currently represents one of various possible solutions. We claim that its merit is best decided on its practical usability with respect to the user's specific purpose. Focus on the practical dimension, i.e. the availability of hands-on tools and libraries, was in fact a key reason for the creation of yet another format. The tools, libraries and existing FoLiA-delivered corpora described in the current work are intended to help people assess whether it is an appropriate solution for their tasks.

The aforementioned work (van Gompel and Reynaert, 2013) presents a comparison with similar initiatives such as the D-Spin Text Corpus Format (TCF), PAULA XML, XCES, as well as with more abstract frameworks such as LAF (Linguistic Annotation Format) and comprehensive text-encoding formats such as TEI. In summary, the prior study observes that rather than a format, LAF (Ide and Romary, 2004) is an abstract framework which offers a greater level of abstraction and genericness than FoLiA, whereas FoLiA is more specific and aims at the practical level. This makes FoLiA more readily adoptable in software tools. In the comparison with TEI (Burnard and Bauman, 2007), it was observed that TEI is very extensive and specific when it comes to encoding text structure, but FoLiA is more specific when it comes to linguistic annotation types, for which TEI only offers more abstract solutions. TEI is very extensive and therefore fairly complex;

schemas may come in various flavours, as elements can be adapted by users in many ways. FoLiA offers one single specific solution instead, the format is a given, and the flexibility to customise is deliberately limited to the data categories or tagsets, in the form of set definitions. Initiatives such as TCF (Heid et al., 2010), PAULA XML (Zeldes et al., 2013), and also NAF (Fokkens et al., 2014) are more similar to FoLiA, as they are less abstract and provide practical usability in software. Differences come down to paradigm choices, sustainability, tool availability and documentation maturity, and especially to variation in coverage of available linguistic annotation types and text structure elements.

Full documentation of FoLiA is available elsewhere (van Gompel, 2014). It offers a reference guide to all elements and attributes that FoLiA defines. A brief summary of key features will be repeated in Section 6.2. Section 6.4 subsequently presents the currently available software infrastructure for FoLiA. Section 6.5 presents some corpora that have been delivered in FoLiA.

6.2 Overview

FoLiA is an XML-based format and defines specific XML elements for *structure annotation* (e.g. paragraphs, sentences, word tokens, lists, figures, etc.) and *linguistic annotation* (e.g. part-of-speech, dependency relations, syntax, named entities, etc.). FoLiA makes use of a combination of inline and stand-off annotation, making proper use of the hierarchical nature of XML and facilitating the job for parsers where possible. FoLiA does not define any linguistic categories; the format is fully language and tagset independent as tagsets are defined separately in *FoLiA Set Definitions* by users and never prescribed by FoLiA itself. These tagsets can in turn be related to data category registries. Validation can proceed on a shallow level, against a RelaxNG schema, as well as on a deep level which validates the used tagsets against the set definition files.

The sets are at the core of the FoLiA paradigm; annotation elements take a generic attribute named ‘class’. These *classes* pertain to a set and are defined by whatever set definition the user decides to use. The set definition defines all allowed classes and allows for links with data category registries for formal semantic closure.

Other generic attributes besides ‘class’ are attributes to denote the annotator of a particular annotation, the annotator type (human or machine), the confidence level of the annotation, the time of the annotation, and more.

FoLiA also allows for various types of *higher-order annotation*, such as the ability to include alternative annotations, as well as extensive support for corrections on annotations. Moreover, there is the possibility to link other modalities, such as imagery or audio fragments of speech, to structural elements. So, even though FoLiA is primarily a format to annotate text documents, speech transcripts are supported as well.

For metadata CLARIN-NL was committed to the CMDI standard (Broeder et al., 2011). Although FoLiA has simple native support for metadata, we see no sense in reinventing the wheel and FoLiA is ideally used in combination with an external metadata format such as CMDI whenever extensive metadata is desired. A reference to the metadata file can be made in the header of the FoLiA document.

The FoLiA paradigm laid out here is schematically illustrated in Figure 6.1 (van Gompel, 2014). A more in-depth treatise is beyond the scope of this current chapter.

6.3 Our philosophy

Recalling the CLARIN principle that a format should be proven and used in practice, FoLiA has been designed in a bottom-up manner taking especially this principle to heart. Our focus is to solve real problems people face in the field with regards to their linguistic representation needs, and to

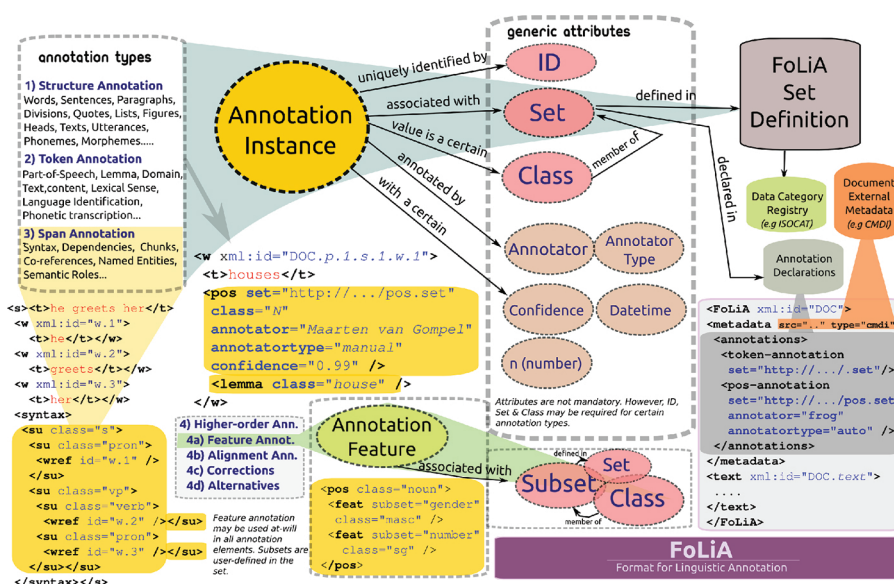


Figure 6.1: A schematic overview of the most important aspects of the FoLiA paradigm, including XML examples (van Gompel, 2014).

do so in a generic manner. The ambition is to deliver a single unified file format that can effectively handle a multitude of annotation needs in a generic way. The main motivation is to prevent the need to switch formats whenever an extra annotation type is introduced, and to prevent the scenario in which a plethora of different formats are used for different annotation types.

It is nevertheless always conceivable that a user's particular need is not yet covered by the latest version of FoLiA; in such cases we gladly hear from the user and expand FoLiA where necessary, in collaboration with the user. The development of FoLiA has already proceeded for several years in such a collaborative workflow, and various annotation types have been added in close contact with end-users both from within CLARIN and from beyond.

In our philosophy, the creation of a file format is useless if an infrastructure of tools to work with said format is not simultaneously created. This has therefore been our main focus over the years and will be the subject of the next section.

6.4 Software Infrastructure

When we speak of a FoLiA software infrastructure we refer to a published set of software, from whatever sources and for whatever architecture, that enable people to work with FoLiA. Such an infrastructure in simple terms encompasses anything that can either process or deliver the data in the format. We can subdivide it into the following components:

1. programming libraries;
2. tools for validation;
3. tools for conversion from and to other formats;
4. tools for visualisation;
5. tools for searching/querying;
6. editing tools and

7. special-purpose tools; i.e. specialised tools that use the format but are not necessarily focused on it. In the case of FoLiA, this includes Natural Language Processing or Information Retrieval tools that use the format as input and/or output.

The programming libraries and tools that are purely designed to visualise, manipulate, or convert the format in basic ways can be considered part of a *core layer* of the infrastructure, whereas the special-purpose tools can be considered to constitute an outer layer.

As FoLiA is an XML-based format, the rich and well-established XML infrastructure is open to its users as well. In fact, almost all FoLiA tools effectively rely on the existing software infrastructure available for XML.

It is possible to not use any of the FoLiA-specific tools and use the infrastructure offered by XML directly. For instance, one can use XPath to query a FoLiA document and XSL to transform it. To do so effectively, however, the user/developer needs to be more familiar with the intricacies of FoLiA than when using a tool from the FoLiA infrastructure that abstracts over this for the benefit of the user/developer.

Many of the tools of the core layer are available as command-line tools and are bundled in two software packages: there is a Python-based **FoLiA Tools** package¹ and a **FoLiA Utilities** package² consisting of tools written in C++. Both are built on the respective libraries. There is some overlap in tools, but each also offers distinct tools the other does not. It is therefore recommended to install both.

These packages, and all other tools pertaining to the FoLiA infrastructure which have been developed at Radboud University, are bundled in our **LaMachine** distribution.³ LaMachine greatly facilitates installation of this software and is a recommended starting point if you work with FoLiA. It is available as a Virtual Machine, a Docker package or a local compilation & installation script.

We subscribe strongly to the CLARIN principle that standards should be open and place a similar requirement on the infrastructure components we build.

6.4.1 Programming Libraries

At the heart of the FoLiA infrastructure are the *programming libraries* that enable developers to work with documents in the format in their software. We ourselves offer libraries for both Python and for C++.

Python is a widely popular high-level programming language in the academic world, and the NLP world in particular. The Python library for FoLiA enables developers to quickly integrate support for FoLiA in their scripts. The library is part of the larger **PyNLPI** library⁴ and is also available from the Python Package Index.⁵ It is extensively documented and comes with tutorials for users.

The Python library suffers from the performance drawback that any high-level interpreted language has. Whenever faster processing is required, or integration in high-performance tools is desired, **libfolia**,⁶ the FoLiA library for C++, offers a better solution. The library is modelled after the Python library, so both are similarly structured, employ a similar syntax and the respective authors try their best to keep the libraries in sync.

¹ <https://pypi.python.org/pypi/FoLiA-tools>

² <https://github.com/LanguageMachines/foliautils>

³ <https://proycon.github.io/LaMachine/>

⁴ <https://github.com/proycon/pynlpl>

⁵ <https://pypi.python.org/pypi/PyNLPI>

⁶ <https://github.com/languagemachines/libfolia>

A third popular language in the field is Java, but no Java-based FoLiA library is available yet to our knowledge. There are a number of Java-based tools in the FoLiA infrastructure that have nevertheless been developed without a common underlying FoLiA library.

6.4.2 Validation

We already touched upon the notion of shallow and deep validation. FoLiA's syntax is formalised in a RelaxNG schema, and shallow validation can therefore be done using any XML validator with support for RelaxNG.

The tools **foliavalidator** and **folialint**⁷ also perform shallow validation, and their usage is strongly recommended, or should even be considered mandatory, for anybody producing FoLiA documents. Moreover, the former tool can optionally perform deep validation as well, i.e. it can validate the used classes against the set definitions.

6.4.3 Conversion

The FoLiA tools and utilities collections contain tools for the conversion from and to various different other formats:

- Conversion to plaintext
- Conversion to HTML
- Conversion to simple columned data or to CSV
- Conversion from/to reStructuredText⁸
- Conversion from/to DCOI XML format (Apperloo, 2006)
- Conversion from the Alpino XML format (Bouma et al., 2000)
- Conversion from ALTO XML format⁹
- Conversion from hOCR HTML format (Breuel, 2007)
- Conversion from PAGE XML format¹⁰

Conversions may be limited by the source or target format. Conversion to FoLiA's predecessor DCOI XML, for instance, is only possible for the subset of elements that DCOI supports. Similarly, conversion to reStructuredText is limited to text, its structure and markup, and does not include linguistic annotations.

Besides the in-house developed FoLiA tools, third parties also make available converters from or to FoLiA. A notable case is **OpenConvert**,¹¹ developed by the former Institute for Dutch Lexicology (INL), now Institute for the Dutch Language (INT), which can convert from TEI, plaintext, ALTO, Microsoft Word, and HTML to FoLiA.

6.4.4 Visualisation

An XSL stylesheet is available to visualise FoLiA documents. It renders documents and unobtrusively pops up with annotation information when hovering over structural items such as words. A major advantage is that this form of visualisation can be conducted entirely client-side in nearly every web browser. The **folia2html** conversion tool also employs the same stylesheet.

⁷ Part of respectively **FoLiA Tools** and **FoLiA Utilities**

⁸ <http://docutils.sourceforge.net/rst.html>

⁹ <http://www.loc.gov/standards/alto/>

¹⁰ <http://www.primaresearch.org/tools>

¹¹ <https://github.com/INL/OpenConvert>

6.4.5 Searching

Tools for searching and querying FoLiA documents can be divided into two categories:

1. In-document search and
2. Document retrieval systems / corpus search tools.

At a low level, in-document search can be conducted with the command-line tool **foliaquery**, part of the FoLiA tools. This tool reads one or more FoLiA documents in memory (sequentially), executes a search query, and presents the matching results. This, however, is not a solution that scales to large numbers of documents as it takes a fair amount of time and memory to process a document.

Full document retrieval systems do not rely on such costly real-time processing of the FoLiA documents, but construct smart indices from the original documents and operate on these indices. The corpus retrieval engine **BlackLab**¹², based on Apache Lucene, and the front-end **WhiteLab** (Reynaert et al., 2014) (see chapter 19) are examples of this. WhiteLab was developed in the CLARIN-NL project OpenSoNaR¹³, and can operate on FoLiA documents, as does BlackLab. So far, these engines typically only supported a simpler subset of the annotation types supported by FoLiA, such as Part-of-Speech tags and lemmas. At the time of writing, there is collaboration, and some competition, between the various developers in the Netherlands to support span annotation types such as dependency relations, syntax and named entities. Another FoLiA-capable search and retrieval system called Multi-Tier Annotation Search (**MTAS**) has been promised by the Meertens Institute, and builds upon Solr and Lucene. It is being developed in the scope of the Nederlab project (Brugman et al., 2016) and the CLARIAH project. This system, however, is still in early stages of development and has not been released yet.

As FoLiA is a highly expressive format, the need arose for a query language tuned specifically to the idiosyncrasies of FoLiA. Although FoLiA can be perfectly searched with XPath, formulating a robust query is not always trivial and may require more in-depth knowledge of FoLiA. The *FoLiA Query Language* (FQL) was designed as a higher-level query language, covering all of FoLiA, to make querying FoLiA documents easier. FQL is implemented alongside the FoLiA Python library in **PyNLPI**. It is documented as part of the FoLiA documentation (van Gompel, 2014).

FQL is a new and expressive query language specifically attuned to the FoLiA paradigm. People in the field are likely more accustomed to the simpler and established query languages such as CQL, the *Corpus Query Language* (Christ, 1994), developed at the Corpora and Lexicons group, IMS, at the University of Stuttgart in the early 1990s. For this reason, **PyNLPI** includes a library that converts CQL to the more expressive but verbose FQL. The low-level query tool makes use of both these libraries. In the next section we will discuss FQL further and introduce higher-level tools in the FoLiA infrastructure that make use of it.

6.4.6 Editing

FQL has been designed in such a way that it is not just a language for passive querying, but a language that allows active manipulation of FoLiA documents. In other words, FQL is to FoLiA as SQL is to relational database tables. Therefore, the **foliaquery** command-line tool and the FQL library it relies on can be used not just to passively retrieve information, but also to actively edit documents.

¹² <https://github.com/INL/BlackLab>

¹³ <https://github.com/TiCCSoftware/WhiteLab>

A FoLiA document server ¹⁴ has been constructed as a back-end for the editing of FoLiA documents. It is implemented as a RESTful webservice, with a simple human-interface to manually enter queries, and takes care of on-demand loading and unloading of documents in memory and serialising them to disk. It maintains a browsable document repository, which features **git** version control support.

Neither the command-line tool nor the document server offers an interface adequate for human end-users to easily work with. To provide such an environment, we have been developing the **FoLiA Linguistic Annotation Tool (FLAT)**¹⁵. It is a modern web-application that offers an interface for the visualisation and editing of FoLiA documents. Under the hood, user-interface interactions are translated to FQL queries and communicated to the aforementioned FoLiA document server. The motivation for the creation of FLAT, as opposed to the adaption of existing web annotation environments, was the desire for a solution that seamlessly integrates with FoLiA and adopts the same paradigm. Different design choices implied it would be easier to build this from the ground up.

Although not yet supporting all of FoLiA at the current stage, **FLAT** has already been used successfully in several annotation projects with student assistants at Radboud University. Further development of FLAT is planned for the CLARIN-NL successor project CLARIAH, with the aim of providing a mature editing environment covering all of FoLiA. FLAT is intended to be deployed as a platform for crowd-sourcing annotation tasks in CLARIAH and other projects.¹⁶

6.4.7 *Special-purpose tools*

The previous sections discussed tools that can be considered part of the core layer. In this section we will discuss the outer layer of tools; these are tools that either take FoLiA as their input or deliver it as their output to perform a specific and specialised task, usually an NLP (annotation) task. It is a most essential layer to the infrastructure and consists of tools such as:

- **Ucto**¹⁷ – An advanced rule-based tokeniser and sentence-splitter for a variety of languages. Supports FoLiA input and output. Can be used to bootstrap plaintext to tokenised FoLiA (van Gompel et al., 2012).
- **Frog**¹⁸ – An NLP suite for Dutch, implementing tokenisation (through Ucto), Part-of-Speech tagging, Lemmatisation, Dependency Parsing, Named Entity Recognition, Shallow Parsing and Morphological Analysis. Supports FoLiA input and output.
- **CLAM**¹⁹ – Turns command-line NLP tools into RESTful webservices with an interface for human end-users. It integrates the FoLiA viewer to visualise FoLiA documents. (van Gompel, 2012)
- **TICCL**²⁰ – Text-Induced Corpus Clean-up (Reynaert, 2010). Supports FoLiA input and output. Used in the CLARIN-NL projects TICCLops²¹ and @PhilosTEI²² (Reynaert, 2014), see chapter 32.

¹⁴ <https://github.com/proycon/foliadocserve>

¹⁵ <https://github.com/proycon/flat>

¹⁶ The PARSEME project for example, <http://typo.uni-konstanz.de/parseme/>, has recently adopted FLAT for the annotation of Multi-Word Expressions

¹⁷ <https://languagemachines.github.io/ucto>

¹⁸ <https://languagemachines.github.io/frog>

¹⁹ <https://proycon.github.io/clam>

²⁰ <https://github.com/martinreynaert/TICCL>

²¹ Available in the CLARIN infrastructure at: <http://ticclops.clarin.inl.nl/ticclops/>

²² Available in the CLARIN infrastructure at: <http://ticclops.clarin.inl.nl/philostei/>

- **Cesax**²³ – A co-reference editor for syntactically annotated XML corpora. Supports FoLiA import and output through conversion.
- **T-Scan**²⁴ – A Dutch text analytics tool for readability prediction (Pander Maat et al., 2014).
- **Colibri Core**²⁵ – A tool for the computation of corpus statistics on n-grams and skipgrams in a quick and memory-efficient way. It can import FoLiA documents, which it subsequently compresses to an internal optimised binary format.
- **Gecco**²⁶ – Generic Environment for Context-Aware Correction of Orthography: A spelling correction engine fully based on FoLiA. Powers *Valkuil.net* and soon also *Fowlit.net*.
- **FoLiA-langcat**²⁷ – Performs language detection, built on TextCat²⁸. Part of the FoLiA utilities.
- **FoLiA-stats**²⁹ – Performs simple n-gram statistics on FoLiA documents. Part of the FoLiA utilities.
- **PaQu**³⁰ – A web application for the analysis of Dutch texts based on dependency parses computed using Alpino (Bouma et al., 2000). PaQu supports FoLiA input (see chapter 23).

6.5 Data Infrastructure

A format's usefulness is not just determined by the tools available, but also by the data sets delivered in the format. The following corpora are currently delivered in FoLiA:

- **Basilex** – The Basilex corpus collects Dutch written language by children, and contains about 11.5 million words. It includes lexical semantic sense annotation (Tellings et al., 2014).
- **DutchSemCor** – The DutchSemCor project delivered a Dutch corpus annotated with lexical semantic senses. Part of the annotation was manual, and a part was tagged automatically with a Word Sense Disambiguation system trained on the manual part. The corpus is based on SoNaR, as well as extra sources (Görög and Vossen, 2010).
- **VU-DNC** – A diachronic Dutch newspaper corpus (2 million tokens) with annotations of subjectivity. Provides a gold standard for OCR'd newspapers published in 1950. (Vis et al., 2012)
- **SoNaR-500** – The STEVIN project SoNaR delivered a 540 million word corpus of written Dutch (including Flemish) from numerous sources. The corpus is annotated with Part-of-Speech tags, lemmas, and named entities (Oostdijk et al., 2013).
- **Nederlab** – The Nederlab project attempts to collect all digitised texts relevant to the history of Dutch language, culture and heritage (circa 800 – present) in one user-friendly and tool-enriched open access web interface³¹ (Brugman et al., 2016).

In addition to corpora, the data part of the infrastructure also consists of a growing number of Set Definitions.³²

²³ <http://erwinkomen.ruhosting.nl/software/Cesax/>

²⁴ <https://github.com/proycon/tscan>

²⁵ <https://proycon.github.io/colibri-core>

²⁶ <https://github.com/proycon/gecco>

²⁷ <https://github.com/LanguageMachines/foliautils>

²⁸ <http://odur.let.rug.nl/~vannoord/TextCat/>

²⁹ Also part of the FoLiA Utilities

³⁰ <https://github.com/rug-compling/paqu>

³¹ <http://www.nederlab.nl/onderzoeksporaal/>

³² <https://github.com/proycon/folia/tree/master/setdefinitions>

6.6 Conclusion

In this chapter we have described the rich infrastructure that has been developed around the Format for Linguistic Annotation (FoLiA). We emphasised the need for a practical and proven format, in line with CLARIN's standardisation principles, and hence placed the focus for this chapter on the software and data infrastructure. A more extensive overview of FoLiA itself and of the motivation for its inception was presented in earlier work (van Gompel and Reynaert, 2013).

Continued efforts in the CLARIN-NL successor project CLARIAH ensure that the developments on the infrastructure surrounding FoLiA will continue in the foreseeable future. FoLiA XML is the pivot format in the project 'Philosophical Integrator of Computational and Corpus Libraries', or PICCL, (Reynaert et al., 2015) which is part of CLARIAH.

Acknowledgements

We gratefully acknowledge the funding provided by CLARIN-NL and its successor CLARIAH in a range of projects. Martin Reynaert further acknowledges being funded by NWO in the Nederlab project.

References

- Wilko Apperloo. 2006. XML basisformaat D-Coi: Voorstel XML formaat presentational markup. Technical report, Polderland Language and Speech Technology.
- Gosse Bouma, Gertjan van Noord, and Rob Malouf. 2000. Alpino: Wide-coverage Computational Analysis of Dutch. In Walter Daelemans, Khalil Sima'an, Jorn Veenstra, and Jakub Zavrel, editors, *CLIN*, volume 37 of *Language and Computers - Studies in Practical Linguistics*, pages 45–59. Rodopi.
- Thomas Breuel. 2007. The hOCR Microformat for OCR Workflow and Results. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, volume 2, pages 1063–1067. IEEE Computer Society.
- Daan Broeder, Oliver Schonefeld, Thorsten Trippel, Dieter Van Uytvanck, and Andreas Witt. 2011. A pragmatic approach to XML interoperability – the Component Metadata Infrastructure (CMDI). In *Balisage: The Markup Conference 2011*, volume 7.
- Hennie Brugman, Martin Reynaert, Noline van der Sijs, René van Stipriaan, Erik Tjong Kim Sang, and Antal van den Bosch. 2016. Nederlab: Towards a single portal and research environment for diachronic Dutch text corpora. In Nicoletta Calzolari et al., editor, *Proceedings of the Tenth International Language Resources and Evaluation Conference (LREC-2016)*, Portorož, Slovenia. ELRA.
- Lou Burnard and Syd Bauman, editors, 2007. *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. Text Encoding Initiative Consortium.
- Christ, Oliver (1994) A modular and flexible architecture for an integrated corpus query system. *Proceedings of COMPLEX'94: 3rd Conference on Computational Lexicography and Text Research*. Budapest, Hungary. pp. 23–32.
- Antske Fokkens, Aitor Soroa, Zuhaitz Beloki, German Rigan, Willem Robert van Hage, and Piek Vossen. 2014. NAF: The NLP annotation format. Technical report.
- Attila Görög and Piek Vossen. 2010. Computer assisted semantic annotation in the DutchSemCor project. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation, LREC-2010*, pages 1220–1226, Valletta, Malta.
- Ulrich Heid, Helmut Schmid, Kerstin Eckart, and Erhard Hinrichs. 2010. A Corpus Representation Format for Linguistic Web Services: The D-SPIN Text Corpus Format and its Relationship with

- ISO Standards. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Language Resources Association (ELRA).
- Nancy Ide and Laurent Romary. 2004. International standard for a linguistic annotation framework. *Natural Language Engineering*, 10(3-4):211–225.
- Nelleke Oostdijk, Martin Reynaert, Véronique Hoste, and Ineke Schuurman. 2013. The construction of a 500-million-word reference corpus of contemporary written Dutch. In *Essential Speech and Language Technology for Dutch: Results by the STEVIN-programme*, chapter 13. Springer Verlag.
- Henk Pander Maat, Rogier Kraf, Antal van den Bosch, Nick Dekker, Maarten van Gompel, Suzanne Kleijn, Ted Sanders, and Ko van der Sloot. 2014. T-Scan: a new tool for analyzing Dutch text. *Computational Linguistics in the Netherlands Journal*, 4.
- Martin Reynaert, Matje van de Camp, and Menno van Zaanen. 2014. OpenSoNaR: user-driven development of the SoNaR corpus interfaces. In *Proceedings of COLING 2014: System Demonstrations*, pages 124–128, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.
- Martin Reynaert, Maarten van Gompel, Ko van der Sloot, and Antal van den Bosch. 2015. PICCL: Philosophical Integrator of Computational and Corpus Libraries. In *Proceedings of CLARIN Annual Conference 2015 – Book of Abstracts*, pages 75–79, Wrocław, Poland. CLARIN ERIC.
- Martin Reynaert. 2010. Character confusion versus focus word-based correction of spelling and OCR variants in corpora. *International Journal on Document Analysis and Recognition*, 14: 173–187. DOI: 10.1007/s10032-010-0133-5.
- Martin Reynaert. 2014. Synergy of Nederlab and @PhilosTEI: diachronic and multilingual Text-Induced Corpus Clean-up. In *Proceedings of the Ninth International Language Resources and Evaluation Conference (LREC'14)*, Reykjavik, Iceland. ELRA.
- Agnes Tellings, Micha Hulsbosch, Anne Vermeer, and Antal van den Bosch. 2014. Basilex: an 11.5 million words corpus of Dutch texts written for children. *Computational Linguistics in the Netherlands Journal*, 4:191–208, 12/2014.
- Maarten van Gompel and Martin Reynaert. 2013. FoLiA: A practical XML Format for Linguistic Annotation - a descriptive and comparative study. *Computational Linguistics in the Netherlands Journal*, 3.
- Maarten van Gompel, Ko van der Sloot, and Antal van den Bosch. 2012. Ucto: Unicode Tokeniser. Reference Guide. Technical report, Tilburg Centre for Cognition and Communication, Tilburg University and Radboud Centre for Language Studies, Radboud University Nijmegen.
- Maarten van Gompel. 2012. CLAM: Computational Linguistics Application Mediator. Documentation. Technical report, Tilburg Centre for Cognition and Communication, Tilburg University and Radboud Centre for Language Studies, Radboud University Nijmegen.
- Maarten van Gompel. 2014. FoLiA: Format for Linguistic Annotation. Documentation. Technical report, Radboud University Nijmegen.
- Kirsten Vis, José Sanders, and Wilbert Spooren. 2012. Diachronic changes in subjectivity and stance – A corpus linguistic study of Dutch news texts. *Discourse, Context & Media*, 1(2–3): 95–102. The view from here, there and nowhere: discursive approaches to journalistic stance.
- Amir Zeldes, Florian Zipser, and Arne Neumann. 2013. PAULA XML Documentation. Rapport de recherche, Institut für Deutsche Sprache und Linguistik - IDSL, INRIA Saclay - Ile de France, Universität Potsdam.