Atomic: an open-source software platform for multi-level corpus annotation

Stephan Druskat[*†], **Lennart Bierkandt**[*%], **Volker Gast**[*†], **Christoph Rzymski**[*†], **Florian Zipser**[**‡]
[*]Friedrich Schiller University Jena, Dept. of English and American Studies, Jena/Germany
[**]Humboldt University of Berlin, Dept. of German Studies and Linguistics, Berlin/Germany
[†]{firstname.lastname}@uni-jena.de
[%]post@lennartbierkandt.de
[‡]zipseflo@hu-berlin.de

Abstract

This paper[1] presents Atomic, an open-source[2] platform-independent desktop application for multi-level corpus annotation. Atomic aims at providing the linguistic community with a user-friendly annotation tool and sustainable platform through its focus on extensibility, a generic data model, and compatibility with existing linguistic formats. It is implemented on top of the Eclipse Rich Client Platform, a pluggable Java-based framework for creating client applications. Atomic - as a set of plug-ins for this framework - integrates with the platform and allows other researchers to develop and integrate further extensions to the software as needed. The generic graph-based meta model Salt serves as Atomic's domain model and allows for unlimited annotation levels and types. Salt is also used as an intermediate model in the Pepper framework for conversion of linguistic data, which is fully integrated into Atomic, making the latter compatible with a wide range of linguistic formats. Atomic provides tools for both less experienced and expert annotators: graphical, mouse-driven editors and a command-line data manipulation language for rapid annotation.

1 Introduction

Over the last years, a number of tools for the annotation and analysis of corpora have been developed in linguistics. Many of these tools have been created in the context of research projects and designed according to the specific requirements of their research questions. Some tools have not been further developed after the end of the project, which often precludes their installation and use today.

Some of the available annotation tools, such as MMAX2 (Müller and Strube, 2006), @nnotate (Plaehn, 1998) and EXMARaLDA (Schmidt, 2004), have been designed to work on specific annotation types and/or levels (e.g., token-based or span-based; coreference annotations, constituent structures, grid annotations). However, the analysis of some linguistic phenomena greatly profits from having access to data with annotations on more than one or a restricted set of levels. For instance, an empirical, corpus-based analysis of scope relations and polarity sensitivity cannot be conducted without richly annotated corpora, with token-based annotation levels such as part-of-speech and lemma, and additionally syntactic and sentence-semantic annotations. And analyses of learner corpora, for example, benefit from an annotation level with target hypotheses, and a level that records discrepancies between target hypothesis and learner production.[3] Similarly, in order to analyze the information structure of utterances, annotations on the levels of syntax, phonology and morphology are essential, as

[1] This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). Page numbers and proceedings footer are added by the organizers. License details: http://creativecommons.org/licenses/by/4.0/.

[2] Atomic is open source under the Apache License 2.0.

[3] For an overview see Lüdeling and Hirschmann (forthcoming).

information structure is "interweaved with various [. . .] linguistic levels" (Dipper et al., 2007).[4] And finally, typological questions such as those that have been the topic of the research project "Towards a corpus-based typology of clause linkage"[5] – in the context of which Atomic has been developed – require corpora that have fine-grained annotations, at various levels.[6] Consequentially, any software designed for unrestricted multi-level corpus annotation should meet the following requirements.

(1) The data model has to be generic and able to accommodate various types of annotation based on the requirements of different annotation schemes. (2) The architecture needs to exhibit a high level of compatibility, as the corpora available for a specific research question may come in different formats. (3) The software must be extensible, since new types of annotation will have to be accommodated, and some may require new functionalities. (4) As the software should be usable by a wide variety of annotators – e.g., experienced researchers as well as students and specialists of various languages including field workers – it needs to provide support and accessibility for users with different levels of experience in corpus annotation.

There has been a trend in corpus linguistics towards the creation of multi-level corpora for a number of years now, which has of course had an effect on tool development as well. Therefore, there already exist some annotation tools which can handle annotations on more than one level, e.g., TrED (Pajas and Štěpánek, 2004) and MATE (Dybkjær et al., 1999). TrED has been developed mainly for the annotation of tree-like structures and therefore handles only limited types of annotations. These do not fully cover some of the required levels for, e.g., typological research questions. MATE, a web-based tools platform, is specifically aimed at multi-level annotation of spoken dialogue corpora in XML formats. Atomic's architecture avoids these limitations by using a generic graph-based model capable of handling potentially unlimited annotation types in corpora of spoken and written texts (cf. 2.2). WebAnno (Yimam et al., 2013) is another multi-level annotation tool currently under development, designed with a focus on distributed annotations. It has a web-based architecture partly based on brat.[7] In contrast to this, Atomic is built as a rich client for the desktop based on the Eclipse Rich Client Platform (RCP, McAffer et al. (2010)), which offers a number of advantages over the implementation as a web application, most importantly ease of extensibility, platform-independence, data security and stability, and network-independence.

The Eclipse RCP comes with a mature, standardised plug-in framework,[8] something which very few web frameworks are able to provide. And while it is a non-trivial task in itself to develop a truly browser-independent web application, the Eclipse RCP provides platform-independence out-of-the-box, as software developed against it run on the Java Virtual Machine.[9] Desktop applications also inherently offer a higher grade of data security than web-based tools, as sensitive data – such as personal data present in the corpus or its metadata – does not have to leave the user's computer. Additionally, access to the corpus data itself will be more stable when it is stored locally rather than remotely, as desktop applications are immune to server failures and the unavailability of server administrators. And finally, a desktop tool such as Atomic, which is self-contained in terms of business logic and data sources, can be used without an internet connection, which is important not only to field workers but also to anyone who wants to work in a place with low or no connectivity, e.g. on public transport.[10]

In terms of interoperability and sustainability, Atomic has been specifically designed to complement the existing interoperable software ecosystem of ANNIS (Zeldes et al., 2009), the search and visualisation tool for multilayer

[4] See also Lüdeling et al. (forthcoming).

[5] Cf. http://linktype.iaa.uni-jena.de.

[6] Annotation levels should minimally include morphology, syntax and information structure, ideally also semantics (sentence semantics and reference) and phonology (including prosody).

[7] Cf. http://brat.nlplab.org/.

[8] Eclipse Equinox, cf. 2.1.

[9] Ibid.

[10] Nevertheless it is of course possible to extend Atomic to use remote logic and/or data sources, such as databases.

corpora, Pepper (Zipser et al., 2011), the converter framework for corpus formats, and LAUDATIO (Krause et al., 2014), the long-term archive for linguistic data. Thus, it seamlessly integrates the compilation and annotation of resources with their analysis and long-term accessibility.

2 Architecture

In order to fulfill the above-mentioned requirements, Atomic's architecture is particularly concerned with extensibility, a generic data model, and a feature set which focuses on accessibility.

2.1 Extensibility

It should be possible for other research groups to build extensions for Atomic for their specific needs (e.g., new editors, access to remote data sources) and integrate them into the platform. This would increase the sustainability of both the platform and the extensions.

We have therefore decided to develop Atomic on top of the Eclipse RCP, an open-source Java application platform which operates on sets of plugins. The RCP itself is also a set of plugins running on a module runtime for the widely distributed Java Virtual Machine (JVM).[11] Hence applications developed on top of it run on any system for which a JVM is available.[12] It enables the implementation of Atomic as a set of plugins and their integration into the application platform, which in turn makes it possible for Atomic to interact with, and benefit from, the vast number of plugins available in the Eclipse ecosystem. These include version control system interfaces (for, e.g., git and Subversion), plugins for distributed real-time collaboration, an R development environment, a TEX editor, and many more. By adding one of the version control system interfaces to Atomic, for example, the corpus data itself, the annotations and all metadata can be versioned in atomic detail, which can be utilised for collaborative corpus annotation.[13]

Eclipse is tried and tested technology which has originally been developed by IBM in 2001 and is now under the aegis of the Eclipse Foundation. Due to its long existence and high impact it is supported by a very large community.[14] Eclipse is used by a wide spectrum of disciplines, mostly from IT and the sciences.[15] These parameters make Eclipse a highly sustainable technology, more so than any single research project can hope to achieve.

Atomic's extensibility is further enhanced through its use of the Salt data model, whose feature set (cf. 2.2) in combination with the capabilities of the Eclipse RCP allow for the creation of very diverse extensions for Atomic, such as for the annotation of historic text, or of speech data. Salt has successfully been used, for example, as an intermediate model for the data used in the RIDGES[16] project, which was possible because the model allows for different text segmentations over the same tokens in the context of the representation of different transcription and annotation levels. And as Salt supports audio and video data sources in addition to textual sources, it has also been used as an intermediate model for the dialogue data of the BeMaTaC[17] project.

2.2 Data model

Atomic's domain model is Salt (Zipser and Romary, 2010), a graph-based metamodel for linguistic data. Salt's generic nature and general lack of semantics makes it independent of specific linguistic analyses, tagsets, annotation schemes and theories, and its graph-based nature allows for the modeling of nearly all conceivable kinds of linguistic structures as nodes and edges. Tokens, spans, hierarchies, and primary texts are all represented as nodes. There can be an unlimited number of edges between nodes, which permits the creation of very diverse types of structures, such as co-reference chains, dependencies, constituent trees, and simpler morphological token annota-

[11] Eclipse Equinox is an implementation of the OSGi specification for a dynamic component model for the JVM.

[12] This includes all major operating systems including Windows, Mac OS X, and Linux.

[13] In Atomic's current iteration, this would be achieved by versioning a corpus document's SaltXML files, cf. 2.2.

[14] Eclipse has over 200 open-source projects and "millions of users" under its umbrella (Eclipse Foundation, 2011).

[15] The Eclipse science community is organized in the Science Working Group at the Eclipse Foundation, cf. http://science.eclipse.org.

[16] Register in Diachronic German Science, cf. http://hdl.handle.net/11022/0000-0000-24EC-E.

[17] Berlin Map Task Corpus, http://u.hu-berlin.de/bematac.

tions. An annotation in Salt is represented as an attribute-value pair with an additional optional namespace tag, and is therefore not restricted to specific tagsets. Salt has originally been designed as a main memory model, but could also be mapped to graph-based persistence technologies such as Neo4j.[18] Salt provides a Java API which is open source under the Apache License 2.0. It was designed using the Eclipse Modeling Framework (EMF, Steinberg et al. (2009)). Models can be persisted via an XMI serialisation of the EMF model as SaltXML, a stand-off format which bundles annotations in one file per document. SaltXML is used as Atomic's default persistence format.

A graph-based domain model such as Salt can be mapped to a graphical representation model of annotation graphs for Atomic relatively easily, as this is supported by the underlying technology: Solutions for the creation of editors and visualizations of EMF-based domain models are available in the Eclipse ecosystem. Projects like the Eclipse Graphical Modeling Framework (GMF) and the Eclipse Graphical Editing Framework (GEF (Rubel et al., 2011), used for Atomic's annotation graph editor, cf. 2.3) have been specifically designed for this purpose.

2.3 Usability and features

User-friendliness starts at the acquisition and installation of software: Atomic is provided as a single zip archive file on the Atomic website,[19] and is available for Linux, Mac OS X, and Windows. No installation as such is necessary, simply extracting the archive to a directory of choice suffices. Unlike other tools – including locally installed web applications –, Atomic is self-contained inasmuch as no further dependencies such as databases, server backends, etc. have to be installed, and the Eclipse RCP plugins are included in the distributed zip file.

At its heart, Atomic consists of a workspace-driven navigator; a document editor for overview, basic annotation and segmentation of corpus documents; a graphical editor for mouse-and-keyboard-based annotation; a command-line shell for rapid annotation with the native annotation language AtomicAL (Figure 1). Additionally, the current version of Atomic includes a dedicated editor for co-reference annotations.

The navigation view provides an interface to the user's workspaces as well as the usual project management features.

The document editor is a simple, text-based overview of a corpus document's primary text. It can be used for token-based annotation, segmentation of a corpus document into processable units, and navigation of a document. The editor is the initial entry point for annotation work. Subsequently, the user is forwarded to the annotation graph editor for further, more granular annotation of the selected corpus segment, span, or sentence.

The annotation graph editor – which offers the user editing facilities based on a graphical representation of the complete, if relatively abstract, annotation graph – is implemented on top of the Eclipse Graphical Editing Framework, and provides intuitive, mouse-based annotation with support for a set of hotkeys for advanced users. The use of well-established graphical user interface metaphors in the editor, such as the tools palette, make it easy for less experienced users to build sophisticated annotation graphs quickly. More experienced annotators can resort to a command-line shell driven by the Atomic Annotation Language (AtomicAL), a data manipulation language for annotation graphs originally developed for use in the GraphAnno annotation tool.[20] AtomicAL enables rapid annotation, as it works with one-char commands (e.g., *a* for "**a**nnotate this element", or *p* for "group these elements under a new **p**arent"), followed by optional flags (e.g., for changing the annotation level), a list of target elements, and a list of annotations.[21] Additionally, annotation options can be restricted by annotating against freely configurable tagsets, defined by the user via project-specific preferences.

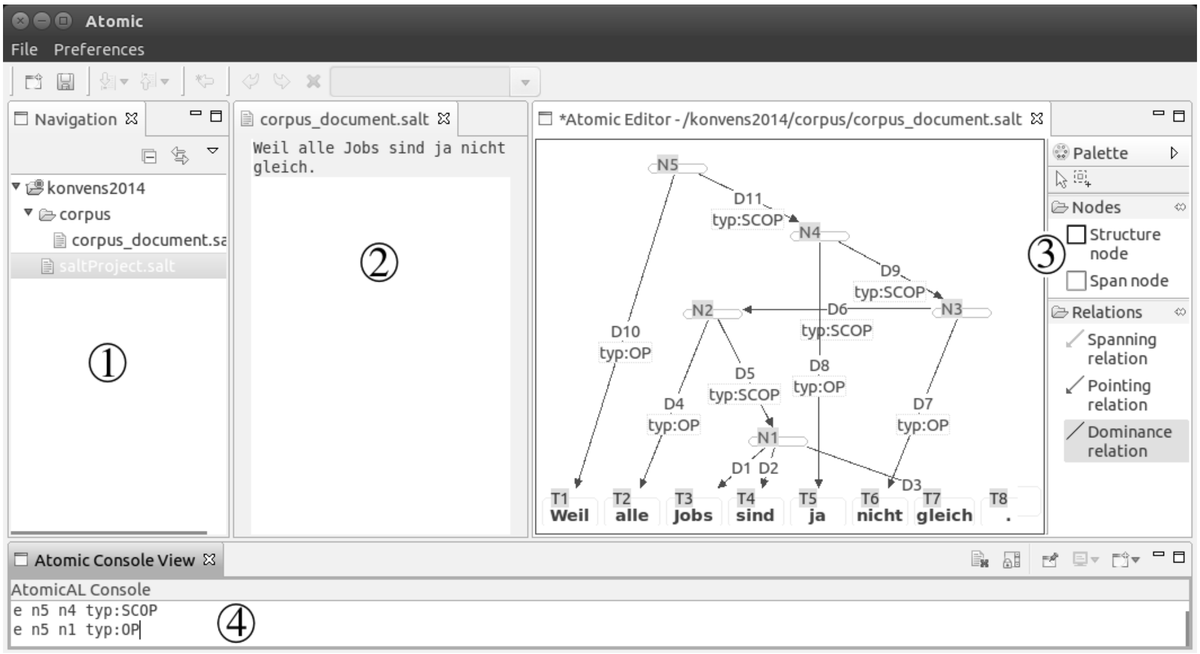
While the annotation graph editor is an all-purpose editor which allows for annotation on arbitrary levels, it may be imperfectly suited for specific annotation tasks. It is therefore desir-

[18] Cf. http://www.neo4j.org/.

[19] http://linktype.iaa.uni-jena.de/atomic.

[20] URL: http://linktype.iaa.uni-jena.de/?nav=graph-anno.

[21] E.g., commands like "On the syntax level, create a new syntactical structure, assign it the category *VP*, and create dominance relations from it to tokens T1, T2, T3, T4, and T5" can be expressed as `p -ls t1..t5 cat:VP`.



```
Weil alle Jobs sind ja nicht
gleich.
```

```
e n5 n4 typ:SCOP
e n5 n1 typ:OP
```

Figure 1: Atomic's application window in a typical configuration: ① Navigation view, ② document editor, ③ graphical annotation editor, ④ AtomicAL command-line shell.

able to include further editors in Atomic which specialise in such tasks, e.g., annotating on only one specific annotation layer or a set of layers. Some tools for example, which are not actively developed anymore, have fulfilled these tasks to the user's satisfaction, who in turn will be well-accustomed to them.[22] It therefore suggests itself to re-create the functionality of these tools in an extension to Atomic, and the above-mentioned co-reference editor is such an attempt.

2.4 Compatibility

For a newly developed multi-level annotation tool, compatibility with existing tools will have a major impact on its half-life and sustainability, and is therefore an indispensible requirement, as corpora should be easily transferrable between the existing tools and the new one. A corpus originally created with, e.g., EXMARaLDA should be importable into Atomic for further annotation, as should be a corpus which has been pre-annotated in an NLP pipeline such as WebLicht (Hinrichs et al., 2010), in order to correct or enhance the annotations. Furthermore, if the annotators are comfortable with using a certain tool for certain annotations, they should still be able to use Atomic for those annotation levels that their favourite tool does not support. As mentioned above, Atomic includes the Pepper framework to tackle this problem area. It is a universal format conversion framework which follows an intermediate model approach, with the Salt meta-model as the intermediate model. To convert data from format X to format Y, X is mapped to Salt and subsequently mapped to Y. The detour via Salt reduces the number of mappings that need to be implemented from $n^2 - n$ mappings (for direct mapping) to $2n$.

Pepper, like Atomic, is plugin-based and comes with a lot of modules realizing such mappings, for instance for EXMARaLDA, TigerXML, tiger2, PAULA, MMAX, Penn Treebank, TreeTagger's XML format, CoNLL, RST, the ANNIS format, and many more.[23] Since Pepper and Atomic share the same data model – Salt –, it has been easy to

[22] Examples include the MMAX2 (Müller and Strube, 2006) co-reference editor, and @nnotate (Plaehn, 1998) for syntax annotations.

[23] This also includes modules for processing TCF (cf. http://korpling.github.io/pepperModules-TCFModules/), an XML format developed within the WebLicht architecture and used by WebAnno as interchange format. The Pepper TCF modules, therefore, provide Atomic with compatibility to WebAnno, as data processed in the latter can be imported into Atomic.

integrate it into Atomic, which in turn provides import and export wizards for all the existing formats. Thus, Atomic is equipped with support for all of the formats for which a mapping exists in Pepper, making it compatible with a wide variety of existing linguistic annotation and search tools. Support for further data formats can be achieved by developing Pepper import and/or export modules for the desired format.

3 Outlook

Following the initial release of Atomic, and some feedback and optimization iterations, we plan to enhance Atomic with additional editors for specific annotation types. Additionally, Atomic should integrate NLP pipelines like UIMA (Ferrucci and Lally, 2004), WebLicht, etc., to provide semi-automatic workflows from within the tool. We also plan to embed support for distributed collaboration via one of the above-mentioned existing plugins.

4 Conclusion

Software for multi-level corpus annotation is subject to a number of requirements. It should operate on a generic data model to allow for potentially unlimited types of annotations, be easily extensible so that new types of annotations and the tooling required for them can be added to it by third parties, and be compatible to other software and data formats in order to make it usable for enriching pre-annotated corpora with additional annotation levels. Additionally, it should be accessible to users with different levels of experience in corpus annotation, as some annotations may be provided not only by corpus linguists, but also by less experienced annotators.

In this paper we have introduced Atomic, an open-source desktop application based on the Eclipse Rich Client Platform which aims at fulfilling the above-mentioned requirements. It does so by operating on the generic graph-based data model Salt, whose lack of semantics allows for potentially unlimited types of annotations, which in Salt are modeled as nodes and edges. Atomic's architecture allows for ease of extensibility through its use of the Eclipse RCP plugin framework, and by incorporating the converter framework Pepper provides compatibility with a wide range of corpus and annotation formats. The tooling making up the core of Atomic is capable of accommodating a user base with potentially diverse levels of experience in corpus annotation: Easily accessible tools such as the annotation graph editor with its tools palette and point-click-and-type workflow are made available for less experienced annotators, while expert annotators can resort to a command-line interface powered by the native data manipulation language AtomicAL.

Atomic complements an ecosystem of software for corpus linguistics, affiliated through a shared data model and a conversion framework based on it respectively. This, together with its high degree of extensibility, makes Atomic a potentially very sustainable tool.

Despite its ready-for-use set of features for multi-level corpus annotation, Atomic is not complete, finalised software. It rather intends to be a platform for corpus annotation tooling upon which the community can build customized solutions for specific research questions as well as feature-complete tools for more general annotation tasks.

References

Dipper, Stefanie, Götze, Michael, and Skopeteas, Stavros (eds.). 2007. *Information Structure in Cross-Linguistic Corpora: Annotation Guidelines for Phonology, Morphology, Syntax, Semantics, and Information Structure*. Interdisciplinary Studies on Information Structure 7, special issue.

Dybkjær, Laila; Møller, Morten Baun; Bernsen, Niels Ole; Carletta, Jean; Isard, Amy; Klein, Marion; McKelvie, David and Mengel, Andreas. 1999. *The MATE Workbench*. In: Proceedings of ACL-1999, demo session, University of Maryland, June 1999, 12-13.

Eclipse Foundation. 2011. *The open-source Developer Report. 2011 Eclipse Community Survey*. URL: `http://www.eclipse.org/org/community_survey/Eclipse_Survey_2011_Report.pdf`

Ferrucci, David and Lally, Adam. 2004. *UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment*. Cambridge University Press, Cambridge.

Hinrichs, Marie; Zastrow, Thomas and Hinrichs, Erhard W. 2010. *WebLicht: Web-based LRT Services in a Distributed eScience Infrastructure*. LREC, European Language Resources Association.

Krause, Thomas; Lüdeling, Anke; Odebrecht, Carolin; Romary, Laurent; Schirmbacher, Peter; Zielke, Dennis. 2014. *LAUDATIO-Repository: Accessing a heterogeneous field of linguistic corpora with the help of an open access repository*. Digital Humanities 2014 Conference. Poster Session. July 2014, Lausanne.

Lüdeling, Anke and Hirschmann, Hagen. forthcoming. *Error Annotation*. In: Granger, Sylviane; Gilquin, Gaetanelle and Meunier, Fanny (eds.). The Cambridge Handbook of Learner Corpus Research. Cambridge University Press, Cambridge.

Lüdeling, Anke; Ritz, Julia; Stede, Manfred and Zeldes, Amir. forthcoming. *Corpus Linguistics*. In: Fery, Caroline and Ishihara, Shinishiro (eds.), OUP Handbook of Information Structure, Oxford University Press, Oxford.

McAffer, Jeff; Lemieux, Jean-Michel and Aniszczyk, Chris. 2010. *Eclipse Rich Client Platform*. 2nd ed. Addison-Wesley, Boston.

Müller, Christoph and Strube, Michael. 2006. *Multi-level annotation of linguistic data with MMAX2*. In: Braun, Sabine; Kohn, Kurt and Mukherjee, Joybrato (eds.). Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods. Peter Lang, Frankfurt/Main, Germany.

Pajas, Petr and Štěpánek, Jan. 2004. *Recent Advances in a Feature-Rich Framework for Treebank Annotation*. In: Proceedings of the 22nd International Conference on Computational Linguistics. Manchester, 673-680.

Plaehn, Oliver. 1998. *Annotate Programm-Dokumentation (NEGRA Project Report)*. Universität des Saarlandes, Saarbrücken, Germany.

Rubel, Dan; Wren, Jaime and Clayberg, Eric. 2011. *The Eclipse Graphical Editing Framework (GEF)*. Addison-Wesley, Boston.

Schmidt, Thomas. 2004. *Transcribing and annotating spoken language with EXMARaLDA*. In: Proceedings of the LREC-Workshop on XML-based richly annotated corpora, Lisbon 2004, ELRA, Paris.

Steinberg, David; Budinsky, Frank; Paternostro, Marcelo and Merks, Ed. 2009. *EMF: Eclipse Modeling Framework 2.0*. Addison-Wesley, Boston.

Yimam, Seid Muhie; Gurevych, Iryna; Eckart de Castilho, Richard; and Biemann Chris. 2013. *WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations*. In: Proceedings of ACL-2013, demo session, Sofia, Bulgaria.

Zeldes, Amir; Ritz, Julia; Lüdeling, Anke and Chiarcos, Christian. 2009. *ANNIS: A Search Tool for Multi-Layer Annotated Corpora*. In: Proceedings of Corpus Linguistics 2009, July 20-23, Liverpool, UK.

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

Zipser, Florian; Zeldes, Amir; Ritz, Julia; Romary, Laurent; Leser, Ulf. 2011. *Pepper: Handling a multiverse of formats*. In: 33. Jahrestagung der Deutschen Gesellschaft für Sprachwissenschaft, Göttingen, February 2011.