corpus-tools.org: An Interoperable Generic Software Tool Set for Multi-layer Linguistic Corpora

Stephan Druskat[1,2], **Volker Gast**[1], **Thomas Krause**[2], **Florian Zipser**[2]
[1]Friedrich Schiller University, Dept. of English and American Studies, Ernst-Abbe-Platz 8, D-07743 Jena
[2]Humboldt-Universität zu Berlin, Dept. of German Studies and Linguistics, Unter den Linden 6, D-10099 Berlin
{stephan.druskat, volker.gast}@uni-jena.de, krauseto@hu-berlin.de, f.zipser@gmx.de

Abstract

This paper introduces an open source, interoperable generic software tool set catering for the entire workflow of creation, migration, annotation, query and analysis of multi-layer linguistic corpora. It consists of four components: Salt, a graph-based meta model and API for linguistic data, the common data model for the rest of the tool set; Pepper, a conversion tool and platform for linguistic data that can be used to convert many different linguistic formats into each other; Atomic, an extensible, platform-independent multi-layer desktop annotation software for linguistic corpora; ANNIS, a search and visualization architecture for multi-layer linguistic corpora with many different visualizations and a powerful native query language. The set was designed to solve the following issues in a multi-layer corpus workflow: Lossless data transition between tools through a common data model generic enough to allow for a potentially unlimited number of different types of annotation, conversion capabilities for different linguistic formats to cater for the processing of data from different sources and/or with existing annotations, a high level of extensibility to enhance the sustainability of the whole tool set, analysis capabilities encompassing corpus and annotation query alongside multi-faceted visualizations of all annotation layers.

Keywords: corpus tools, multi-layer corpora, interoperability

1. Introduction

This paper introduces an interoperable generic software tool set[1] which caters for the entire workflow of creation, migration, annotation, query and analysis of multi-layer linguistic corpora. The tool set consists of four major components:

1. Salt (Zipser and Romary, 2010) (Humboldt-Universität zu Berlin, 2016b), a meta model and API for linguistic data, working on a generic graph structure and functioning as a common data model for the other tools;
2. Pepper (Zipser et al., 2011) (Humboldt-Universität zu Berlin, 2016a), a conversion tool and platform for linguistic data that can be used to convert a large number of different linguistic formats into each other;
3. Atomic (Druskat et al., 2014) (Friedrich Schiller University Jena, 2015), an extensible, platform-independent multi-layer desktop annotation software for linguistic corpora;
4. ANNIS (Krause and Zeldes, 2014) (Humboldt-Universität zu Berlin, 2015), a search and visualization architecture for multi-layer linguistic corpora with many different visualizations and a powerful native query language.

All components are open source under the permissive Apache License, Version 2.0[2].
We define the corpus-tools.org tools as a software *set* rather than a *stack* or a *toolchain*, as they are designed for close integration, but can be used independently of each other and must be differentiated by their target functions: Salt and Pepper are primarily infrastructure tools, i.e., they provide infrastructural foundations for computer-based corpus research; Atomic and ANNIS are primarily workflow tools, i.e., they contribute to the actual processing of corpora.
The above is reflected in the architecture of corpus-tools.org (Figure 1): Salt serves as a common data model for Atomic, ANNIS and Pepper. The main function of Pepper is that of a "communication bus" between different corpus tools, both infrastructurally and within a concrete workflow. The tools subset of Pepper, Atomic and ANNIS forms a complete corpus workflow toolchain in itself, which is based on Salt.

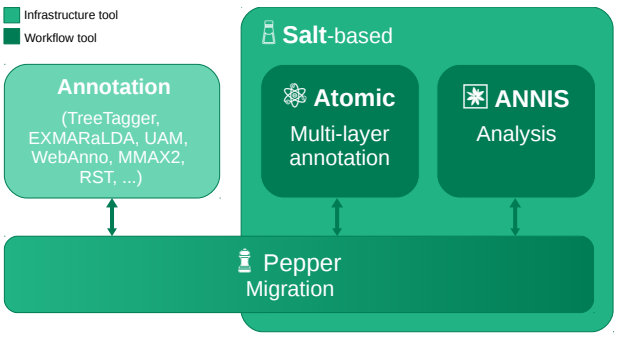

Figure 1: Overview of the corpus-tools.org architecture, with functional grouping.

Note that although all tools in the set can be used independently, their interoperability by design lets the user benefit most from corpus-tools.org when using all tools together. Additionally, all tools have been built to be able to integrate existing solutions: Pepper together with Salt facilitates not only the integration of existing corpus tools with corpus-

[1]`http://corpus-tools.org`, (corpus-tools.org Development Team, 2015).
[2]`www.apache.org/licenses/LICENSE-2.0.html`.

tools.org software, it also assists collaboration among different third-party tools; both Atomic and ANNIS have been built to be extensible via plugins. Thus, tried and tested annotation editor types or NLP components can be added to Atomic, established visualization types to ANNIS.
Let us take a step back now and look at the necessity of introducing yet more tools to the world of multi-layer corpora: While there already exist a number of corpus tools[3] – some of them equipped to deal with multiple layers of annotation, or different annotation types –, not all of them interoperate well with other software, due to a lack of common standards mostly on the part of data models and formats.
More generally, some key workflow problems have not been addressed comprehensively as of now:

1. To assure general interoperability, the tools in a corpus creation and analysis workflow should be able to transfer data losslessly between each other, ideally via a common data model.

2. Such a data model should avoid constraints on the types of annotation it can process. It must be sufficiently generic to be able to process diverse types of annotations, thus allowing for its utilization in a maximum number of use cases.

3. As corpora and annotations exist in many different formats, migration/conversion capabilities are a key requirement for tools when creating and annotating corpora, especially when they come from different sources and/or have existing annotations.

4. User-facing tools should be extensible, as new formats will be introduced in the future, and new research questions will require specific annotation types, which in turn will require specific tooling (e.g., annotation editors, visualizations, NLP components).

5. Comprehensive analysis of multi-layer annotated corpora – which complements any corpus workflow – requires an analysis tool that is able to both query the corpus and its annotations, and present the user with a multi-faceted view on her data, incorporating the aspects of all annotation layers.

With the corpus-tools.org tool set we try to address these challenges. In the following, we will present the single parts of the tool set in more detail, loosely following a hypothetical workflow.

2. A common generic data model

As mentioned above, the tools in a corpus creation and analysis workflow should ideally share a common data model, moreover one that is sufficiently generic to maximize the tool set's use cases, and thus its sustainability. The corpus-tools.org tool set's common data model is Salt (Zipser and Romary, 2010).
Salt is a generic, graph-based meta model for linguistic data, implemented as an open source Java API for storing, manipulating and representing data. Due to its high level of abstraction, it remains independent of linguistic theories, annotation schemes and tagsets. Salt is text-based, but also allows for the modeling of audio and video corpora. Its core structure is a graph, which allows for almost any conceivable annotation type, as long as it fits a graph structure[4]. This structure also helps to keep the model simple, with only a small set of model elements, given by $G = (V, E, L, label_a, \dots label_b)$ with

- V being a set of nodes with $v = (label_c, \dots label_d) \in V$
- E being a set of directed edges with $e = (v_1 \in V, v_2 \in V, label_e, \dots label_f) \in E$
- L being a set of layers with $l = (V' \subset V, E' \subset E, L' \subset L, label_g, \dots label_h) \in L$
- and a set of labels $label_a, \dots label_b$ the graph is labeled with. A label l is given by a triple $l = (ns, n, v)$ with a namespace ns, a name n and a value v. A namespace is optional and the combination of ns and n form a unique identifier for a label.

To illustrate the model of Salt, imagine a text which is syntactically annotated as given in Figure 2. In the Salt model,

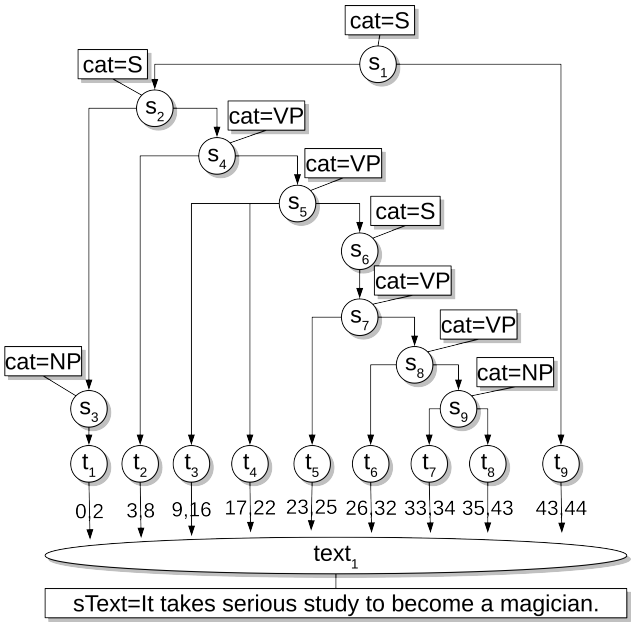

Figure 2: A syntactically annotated sentence modeled in Salt.

nodes and edges are placeholders for an abstract structure. They are initially free of semantics, but are subsequently

[3]For example EXMARaLDA (Schmidt, 2004), WebAnno (Yimam et al., 2013), ELAN (Brugman and Russel, 2004), @nnotate (Plaehn, 1998), Synpathy (`http://www.mpi.nl/tools/synpathy.html`), MMAX2 (Müller and Strube, 2006), Arborator (`http://arborator.ilpga.fr/`), or Praat (Boersma and Weenink, 2013) .

[4]We have yet to encounter an annotation model which cannot be mapped onto a graph structure.

specified by adding labels. Reading the example bottom up, we start with the base node $text_1$ representing the primary text. Its label contains a representation of the text. The nodes $t_1 \dots t_9$ represent a tokenization of the primary text. Each token node is related to the text node, and the relation between them determines the text sequence that the token overlaps, via an offset. The remaining nodes $s_1 \dots s_9$ in combination with the relations between them define the syntax tree. Each node in the syntax tree has a label $cat = X$, which adds semantics[5] to that node, such as representing an NP, a VP, etc.

In a multi-layer corpus, the number of nodes and edges is prone to become very large. Layers alleviate this issue by representing a grouping mechanism for bundling nodes and edges into smaller subgraphs. For instance, all nodes and edges belonging to morphological annotation, syntax annotation, information structure annotation, etc., can be bundled in seperate layers.

The Salt meta model differentiates between corpora and documents. A document contains one or more primary texts, and corresponding annotations. A corpus is an aggregation of documents and/or other corpora, i.e., subcorpora. The split of an entire corpus into non-overlapping documents – partitions – allows for processing each document independently. Salt is a main memory data structure, and its scalability is determined by the availability of main memory and the size of the largest document in a given corpus which will be required to reside in memory. Documents can be partitioned along logical definitions, such as volumes, chapters, sections, etc., which will be spanned by discrete annotation graphs, or according to technical definitions. Syntactic annotations, for example, will usually affect only one sentence. Therefore, each sentence can be a discrete partition. Thus we can build small documents along sentence boundaries. At the same time, unneeded documents can be serialized to disk and de-serialized for processing on demand. Hence, an intelligent distribution of document sizes can increase the potential size of corpora that can be represented in Salt models.

3. Creating/migrating corpus resources for annotation

Corpora and annotations exist in a multitude of different formats. In order to prepare them for further annotation, it is necessary to convert – and sometimes merge – them into a format the annotation software can process. This can be done via Pepper (Zipser et al., 2011), a platform-independent, modular framework for converting and processing linguistic data.

Pepper utilises an intermediate-model approach, with Salt as its intermediate model: Instead of implementing a direct conversion solution from format X to format Y, X is mapped onto Salt, manipulated where necessary, and subsequently mapped onto format Y. Thus, the number of mappings to convert *n* into *m* formats is reduced to $2n$, as compared to $n^2 - n$ mappings for direct conversion routes. Pepper supplies three types of modules: importers, manipulators, and exporters, an unrestricted number of which can be combined into one single conversion workflow.

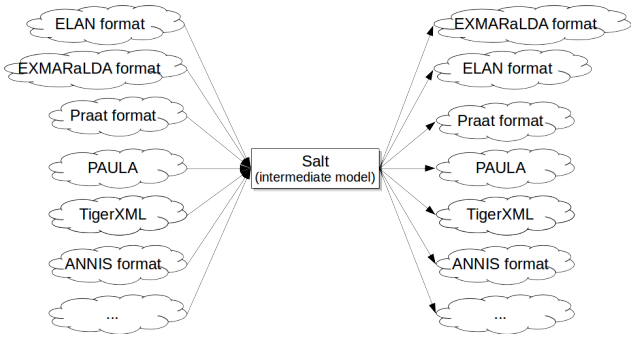

Figure 3: Intermediate model approach to convert linguistic formats.

The loose coupling of modules and the partitioning of data into documents in Salt enable the processing of data in isolation from preceding or following modules in the workflow. Let c be a corpus consisting of the documents $d_1, \dots, d_n$, let w be a workflow consisting of the modules $w = (m_1, m_o)$. For all $1 \geq i, j \geq n$ with $(i \neq j)$ and $1 \geq k \geq o$: The process $m_k(d_i)$, $m_{k+1}(d_i)$, $m_k(d_j)$, $m_{k+1}(d_j)$ is equal to the process $m_k(d_j)$, $m_k(d_i)$, $m_{k+1}(d_i)$, $m_{k+1}(d_j)$. Thus we can randomize the order of documents. Furthermore, documents can be processed in parallel, as long as each document passes each module in the workflow in the correct order. Therefore, Pepper has implemented multithreading in order to greatly reduce conversion times. The size of a corpus which can be processed with Pepper is not limited, as long as each single document fits into main memory.

The possibility to plug any manipulator module between an importer and an exporter allows for manipulation, enhancement or reduction of data in every possible sense. It is possible, for instance, to merge data. Often a research question demands several annotation types on the same corpus. Unfortunately, many tools and formats were created for just a single kind of annotation, for instance the TigerXML format (Lezius et al., 2002), which was created exclusively for constituents, or the MMAX2 format (Müller and Strube, 2006), which was created exclusively for coreferences. However, in order to build multi-layer corpora we need to combine different kinds of annotation. Due to the fact that Salt is a graph-based model, the task of merging different Salt models is reducible to a graph merging task based on an identical primary text or primary texts. With the process based on the primary text, it is possible in a first step to merge the tokens. Since Salt allows for multiple tokenizations, even alternative tokenizations can be processed. Additionally, higher-level structures such as spans or hierarchies can be compared with each other and merged (Zipser et al., 2014). Furthermore, the isolation of modules in the workflow allows for a combination of each set of importers with the merging step.

In addition to the merging module cited above, there already are Pepper modules for a large number of different linguistic formats[6], as well as a module for extracting meta-

[5] "Semantics" not in its strict linguistic meaning.

[6] E.g., EXMARaLDA, Tiger-XML, MMAX2, RST, TCF, Tree-

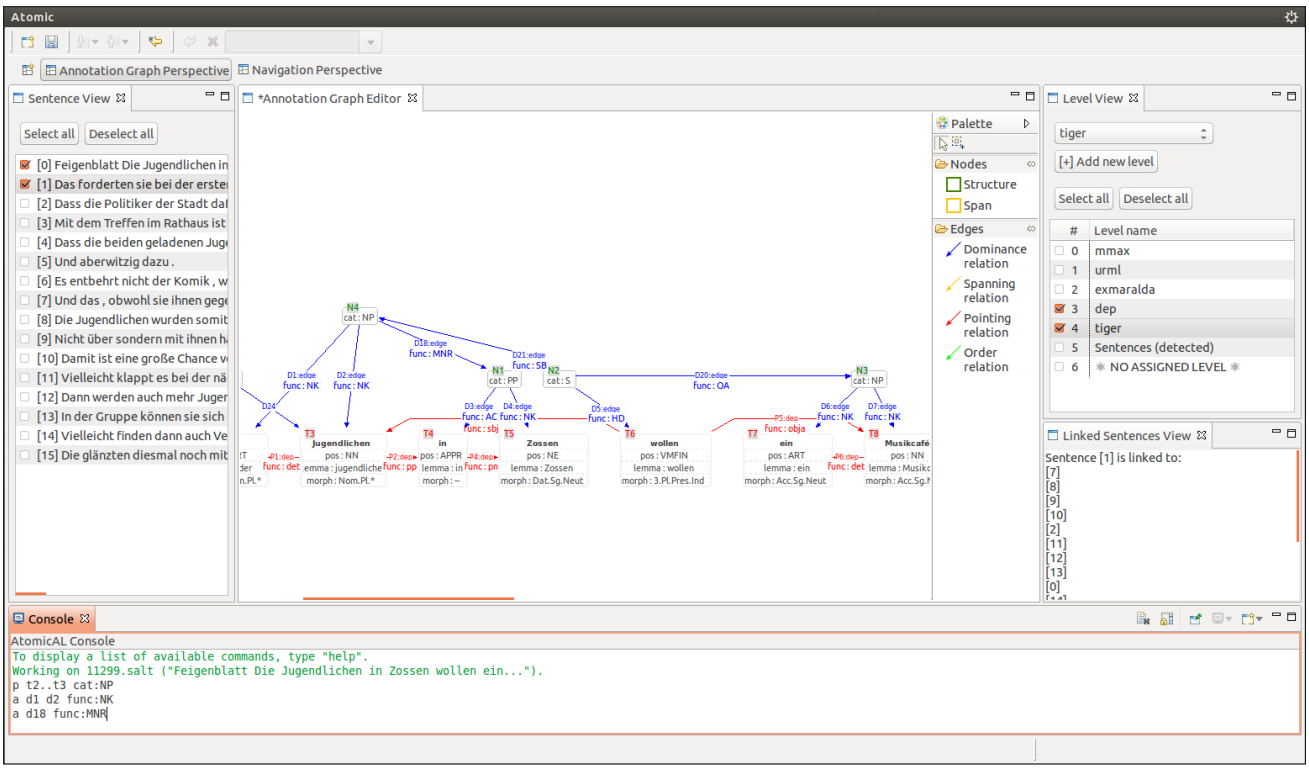

Figure 4: A screenshot of Atomic, showing a section of the pcc2 corpus (Stede and Neumann, 2014), with two annotation layers visible in the editor, and a number of AtomicAL commands in the console.

data, structural and annotation-related information from existing corpora[7] (Voigt et al., 2016). Due to its plugin-based architecture, newly-developed modules can easily be added to Pepper at any time.

Pepper comes in two flavours: as an interactive standalone command line tool and as an API library, which can be integrated in other software.

In an hypothetical workflow utilizing the corpus-tools.org tool set, Pepper is most likely to come in in two places: Firstly as the primary step, preparing corpora for annotation by merging and converting them into a format consumable by the annotation tool, and secondly in between or after annotation steps, preparing the resources for analysis or publication.

Pepper can process several documents in parallel, and only uses the main memory needed to hold the currently processed documents in the conversion workflow. For example, converting the syntactically annoted TIGER2 corpus (Brants et al., 2004), which has 888,578 tokens, 1,262,014 nodes and 1,971 documents, from SaltXML to the ANNIS format takes about one minute on a modern notebook with a 2.20GHz Intel i7-4770HQ CPU, using all four processor cores. In general, the time a conversion takes depends on the import and export modules used, and whether several import formats have to be merged, or if there are costly manipulators in the pipeline.

Tagger format, TEI (subset), ANNIS format, PAULA and many more.

[7]For an example of the output this module creates, see `https://corpora.uni-hamburg.de/sfb632/a5hausanews/`.

4. Annotation

Most of the above-mentioned general challenges for multilayer corpus tools can also be applied more specifically to annotation software: It should avoid putting constraints on the types of annotation it can process, it needs to be able to deal with the multitude of existing linguistic formats, and it should be extensible for new annotation processes, i.e. new annotation types, editors, data views, etc. Atomic (Druskat et al. (2014), cf. Figure 4), a platform-independent software for the desktop, addresses these issues through its architecture.

One of Atomic's assets is that it hooks, as it were, into the Pepper process at its intermediate stage – effectively as an extended quasi manipulator module – by working directly on Salt models. It includes the Pepper library and provides import and export GUI wizards for the different Pepper modules. Thus, Atomic inherits not only Salt's specific advantages – i.e., it is not limited to specific annotation types, layers, or tagsets – it is also highly compatible with a large number of linguistic formats via Pepper. Apart from that, it is also possible to initially create and pre-process, rather than import, corpora with Atomic (cf. below).

Atomic is built on top of the Eclipse Rich Client Platform (McAffer et al., 2010) – a mature, standardized plugin framework written in Java –, which makes it easily extensible. Atomic is basically a set of plugins, adding to the set of plugins making up the Eclipse RCP. Therefore, new functionality can be implemented as yet another plugin and added to Atomic. For this, the latter provides a number of

extension points[8] and interfaces for different common components: editors for different annotation types, data views, NLP components, model workflow steps, and annotation language dialects (see below). To facilitate the creation of corpora with Atomic, for example, the software does provide some basic pre-processing tools – a tokenizer and a partitioning tool –, but more importantly also extension points for further, custom pre-processing steps. Any corpus processing step, e.g., parsers, taggers, etc., can thus be implemented as an Eclipse plugin and added to Atomic dynamically via the respective extension point. Thus, Atomic is in principle an annotation *platform* rather than simply an annotation tool.

Another advantage of Atomic being implemented as an Eclipse RCP is that it benefits from the large number of existing compatible software and resources within the Eclipse ecosystem.[9] Third-party Eclipse RCP plugins, which can be added to Atomic on the fly, exist for a variety of applications: Editors for XML, TEX, R, and many others; version control system interfaces for Git, SVN and others (using a version control system enables collaborative annotation as well as providing change management for corpora); distributed real-time collaboration; tools for semantic web applications; and many more. Additionally, due to the size and commercial importance of Eclipse, its ecosystem, and the Rich Client Platform, software developers proficient in Eclipse-based development – and thus able to extend Atomic, e.g., in the context of service contracts – are readily available, which increases Atomic's sustainability.

As a workflow tool, Atomic, in its current iteration, provides the following features.

- A graphical editor for Salt annotation graphs. The editor provides a graphical representation of the Salt subgraph for selected partitions of a corpus document. The graph can be manipulated via the keyboard, e.g., by selecting and editing elements, and through mouse-driven tools available from a palette similar to those included in established graphics or desktop publishing applications.
- A synchronized, integrated command-line interface for use of the annotation language AtomicAL[10] for rapid creation and annotation of model elements. AtomicAL uses one-letter commands for common annotation tasks on the Salt graph, which are reflected synchronously in the graphical editor. Commands can take a number of arguments and flags, specifying the model elements, annotation layer, annotations, etc., to work on. AtomicAL is designed to be extensible, so that different "dialects" can be implemented for different annotation types and editors.
- A customizable workbench-like GUI providing core application functionality (project management, preferences, update management for Atomic and the integrated Pepper and Salt libraries, help and documentation) and task-specific view combinations, so-called "perspectives". For example, the annotation graph editor is embedded in a perspective which displays it together with relevant views, such as a view for displaying and selecting the corpus partitions to be edited; a view for displaying, creating, editing and selecting annotation layers; a view for displaying partitions with links to the current selection; the AtomicAL Console.

Further extensions are being developed or planned (cf. 8.). Alternating with ANNIS (cf. 5.), Atomic would be at the heart of an hypothetical workflow using our tool set.

As Atomic uses Salt as its data model, any corpus size restrictions for Salt are applicable for Atomic as well. Hence, the available main memory must be able to hold the largest document in the corpus. However, as Atomic supports customized partitioning of corpus documents, and the annotation graph editor can work on a given custom partition while the others remain serialized, the potential size of corpora Atomic can process depends merely on the strategy of document size distribution over that corpus.

5. Query and analysis

Comprehensive analysis of multi-layer annotated corpora requires a tool which provides extensive search functionality as well as appropriate visual representation of all annotation layers. With ANNIS (Krause and Zeldes, 2014) we provide a browser-based search and visualization architecture for complex multi-layer corpora.

ANNIS also makes use of Salt as a data model: Salt powers different visualizations and acts as an interchange format between front- and backend.

ANNIS furthermore provides the native query language AQL for complex search queries as well as different visualizations for corpus data, such as KWIC views, dependecy trees, grids, coreference and RST views, aligned A/V data, and many more.[11]

ANNIS can be extended with new visualizations for additional types of annotations via new plugins. Additionally, existing visualizations – such as the HTML visualizer – are highly configurable.

In an hypothetical workflow, ANNIS would most likely come into play in one of the later steps, being used to query, visualize and analyze multi-layer annotated corpora. It could, however, be utilized earlier as well, for example to identify annotation candidates in a corpus. Additionally, in interplay with Atomic, it also facilitates iterated annotation, i.e., repeated iterations of annotation tasks in Atomic and their subsequent evaluation in ANNIS, the results of which

[8] Extension points are basically contracts – usually a combination of a definition in XML markup, and Java interfaces – that extensions must conform to. New Atomic plugins that want to connect to a specific extension point must implement the specified contract. For an overview of Eclipse's extension point mechanism see Clayberg and Rubel (2009, 637–660).

[9] The best known Eclipse RCP-based product alone, an IDE for Java, has hundreds of thousands of users from IT and other disciplines, cf. `http://www.eclipse.org/downloads/`.

[10] AtomicAL is based on a data manipulation language for annotation graphs developed for use in the GraphAnno annotation tool (Gast et al., 2015), but exhibits a different syntax and different commands.

[11] `corpus-tools.org/annis/visualizations.html` gives an overview of available visualizations.

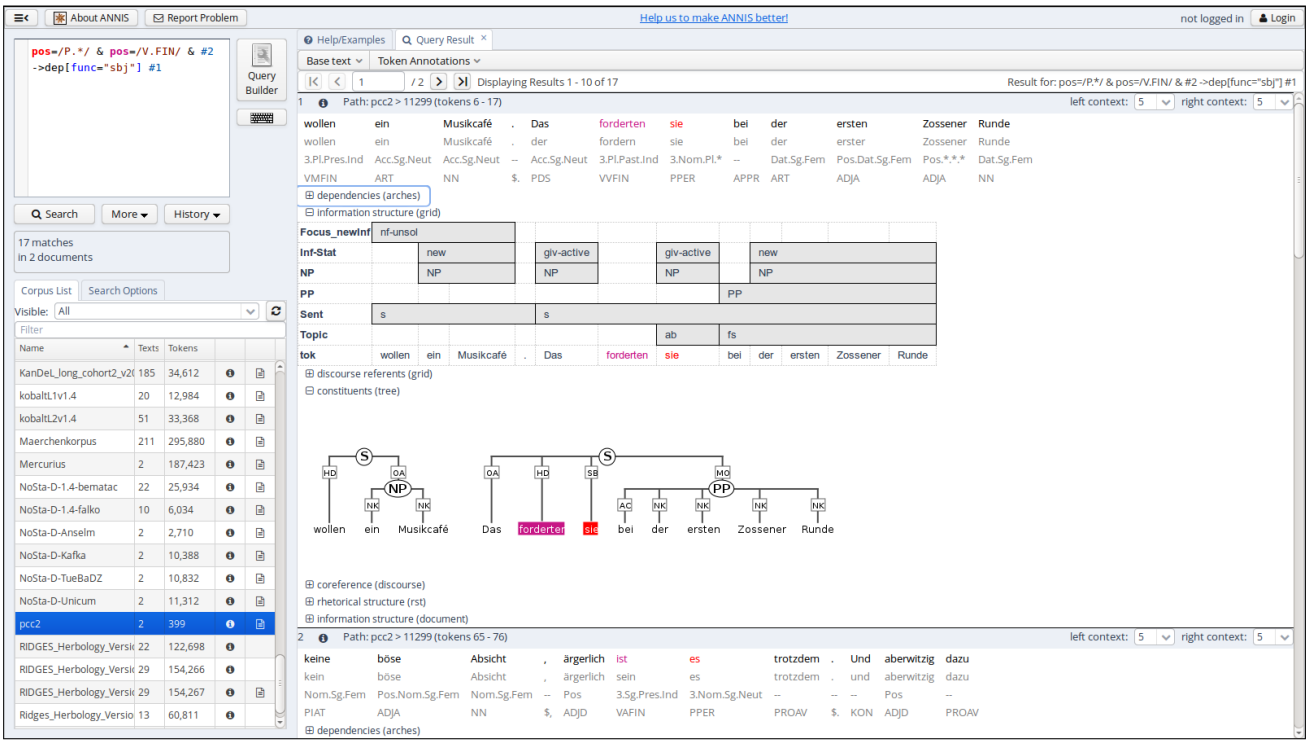

Figure 5: A screenshot of ANNIS, showing an AQL query over the pcc2 corpus, and example visualizations of the result.

entail another annotation iteration in Atomic, etc. And finally, ANNIS can also be used as a repository of sorts for a release of annotated corpora.

ANNIS is optimized to handle corpora with a large number of parallel annotation layers. Thus, the maximum size of corpora it can process mainly depends on the number of nodes and edges in a corpus, not on token figures. However, since ANNIS uses the PostgreSQL relational database[12] it is key to have a powerful server with sufficient main memory when querying corpora containing more than 1m nodes.

6. Development and user communities

While the centres of development of the corpus-tools.org tool set are the Deptartment of English and American Studies at the Friedrich Schiller University in Jena and the Department of German Studies and Linguistics at Humboldt-Universität zu Berlin, a host of linguists and developers from around the world have made and continue to make valuable contributions to corpus-tools.org tools,[13] rendering our tool set a true community effort.

Tools from the corpus-tools.org tool set are used by projects from different research fields, such as spoken language corpora[14], historical corpora[15], chat corpora[16], newspaper corpora[17], and many more.

7. Conclusion

The corpus-tools.org software tool set we present here facilitates a complete workflow for multi-layer corpora, from creation and annotation to analysis and release. At the same time it provides answers to some important challenges such a workflow poses, (a) through its internal interoperability by relying on a common data model which is also sufficiently generic to not limit the types of annotation it can model; (b) through its compatibility with a large number of linguistic formats by means of its integrated conversion framework; (c) through the extensibility of its components which contributes to its sustainability and versatility; (d) through its ability to integrate with third party tools.

[12] Cf. `http://www.postgresql.org/`.

[13] See, for example, the list of ANNIS contributors at `github.com/korpling/ANNIS/graphs/contributors`.

[14] E.g., the Berlin Map Task Corpus (`https://www.linguistik.hu-berlin.de/en/institut-en/professuren-en/korpuslinguistik/research/bematac`), or the KiezDeutsch-Korpus (`http://www.kiezdeutschkorpus.de/en/`).

[15] Cf. for example Coptic Scriptorium (`http://www.carrieschroeder.com/scriptorium/`), Ridges (Register in Diachronic German Science, `http://korpling.german.hu-berlin.de/ridges/index_en.html`), Perseus Latin and Ancient Greek Treebank (`http://annis.perseus.tufts.edu/`).

[16] Cf. sms4science (`http://www.sms4science.ch/index.html`), or What's up, Switzerland? (`http://www.whatsup-switzerland.ch/index.php/en/`). A comprehensive list of projects that use, for example, ANNIS can be found at `http://corpus-tools.org/annis/cooperations.html`.

[17] Cf. pcc2 (Stede and Neumann, 2014).

8. Outlook

The development of all corpus-tools.org tools is ongoing, i.e., they are constantly being extended, optimized, and iteratively released. The below paragraphs give an insight into some development ideas and goals for the future.

Salt is a mature tool, and currently available in version 3.0, last released in February 2016. Similarly, Pepper is available in stable release version 3.0, last released in early 2016. New Pepper modules are constantly developed, and enhance Pepper's ability to integrate further tools and formats. Ideas for new Pepper modules are abundant. For example: In corpus linguistics, the comparison of corpora is a recurrent topic of discussion. For instance, after several annotation iterations one may want to find the differences between versions of a corpus. Similarly, when multiple annotators have worked on the same corpus one may want to discover the differences between their respective annotation work. Due to Salt's graph-based nature these questions could be broken down into a graph-matching problem, in a similar fashion as with merging. As of now, Salt includes an API for graph comparison, but in order to inform annotators about changes in a well-structured manner, an easily accessible representation of differences of the above-mentioned natures is necessary. Such a visualization could be implemented as a Pepper module which would in turn offer the possibility to compare corpora across different formats.

Atomic is available in preview version 0.2.1, and is currently developed towards stable release version 1.0, due in 2016. After this, development efforts will focus on extending its core functionalities, thereby providing exemplary cases for further development, including by third parties. The exemplary cases are planned to include, for example, further NLP components to be made available for corpus building, such as parsers and taggers; support for annotation schemes and tagsets, binding them to editing functionalities in graphical editors as well as AtomicAL; API for additional editor types, and default implementations, such as a tabular editor for span-based annotation, a graph- and text-based editor for coreference annotation, and a dedicated editor for syntax trees; exchangable backends: Currently, Atomic supports serialization to SaltXML as well as export into different linguistic formats provided by Pepper. However, it should be possible for Atomic to persist Salt models in other ways, for example by storing them in a database or in the cloud, thus simplifying remote collaboration along the way; search functionality, and (semi-)automatic annotation of result sets; graphical and possibly TEX export of annotation graphs and trees.

ANNIS, currently available in stable version 3.3 released in September 2015, will continue to use a preview-release model where new features are added to frequent preview release versions, while more extensively tested regular versions are released about twice a year. It is planned to have a better integration of search results from ANNIS in the corpus-tools.org tool set by allowing the export to different formats, including SaltXML. It should also be made easier to update only parts of an already imported corpus, e.g. when annotation errors were corrected in Atomic. Instead of converting and re-importing the complete corpus in ANNIS, the user should be able to import the changed document only. Another substantial, and more long-term, development effort is the enhancement of ANNIS' scalability by developing a new, graph database-based backend.

9. Acknowledgements

We thank three anonymous reviewers for their valuable comments. Financial support from the German Science Foundation (DFG, grant GA-1288/5-1) is gratefully acknowledged.

10. Bibliographical References

Boersma, P. and Weenink, D. (2013). Praat: doing phonetics by computer [computer program]. http://www.praat.org.

Brants, S., Dipper, S., Eisenberg, P., Hansen, S., König, E., Lezius, W., Rohrer, C., Smith, G., and Uszkoreit, H. (2004). TIGER: Linguistic Interpretation of a German Corpus. *Journal of Language and Computation*, (2):597–620.

Brugman, H. and Russel, A. (2004). Annotating multimedia/multi-modal resources with ELAN. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*. Lisbon, Portugal.

Clayberg, E. and Rubel, D. (2009). *Eclipse Plug-ins*. Addison-Wesley Professional, Boston, Mass., 3rd edition.

Druskat, S., Bierkandt, L., Gast, V., Rzymski, C., and Zipser, F. (2014). Atomic: an open-source software platform for multi-level corpus annotation. In Josef Ruppert et al., editors, *Proceedings of the 12th Konferenz zur Verarbeitung natürlicher Sprache (KONVENS 2014)*, pages 228–234.

Gast, V., Bierkandt, L., and Rzymski, C. (2015). Annotating modals with GraphAnno, a configurable lightweight tool for multi-level annotation. In M. Nissim et al., editors, *Proceedings of the Workshop on Models for Modality Annotation*, pages 19–28, Stroudsburg, PA. Association for Computational Linguistics (ACL).

Krause, T. and Zeldes, A. (2014). ANNIS3: A new architecture for generic corpus query and visualization. *Digital Scholarship in the Humanities.*

Lezius, W., Biesinger, H., and Gerstenberger, C. (2002). TIGER-XML Quick Reference Guide. Technical report, IMS, University of Stuttgart.

McAffer, J., Lemieux, J.-M., and Aniszczyk, C. (2010). *Eclipse Rich Client Platform*. Addison-Wesley, Boston, 2nd edition.

Müller, C. and Strube, M. (2006). Multi-level annotation of linguistic data with MMAX2. In Sabine Braun, et al., editors, *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*, pages 197–214. Peter Lang, Frankfurt a.M., Germany.

Plaehn, O., (1998). *Annotate Programm-Dokumentation (NEGRA project report)*. Universität des Saarlandes, Saarbrücken.

Schmidt, T. (2004). Transcribing and annotating spoken language with EXMARaLDA. In *Proceedings of the LREC-Workshop on XML based richly annotated corpora*. Lisbon, Portugal.

Stede, M. and Neumann, A. (2014). Potsdam Commentary Corpus 2.0: Annotation for discourse research. In *Proceedings of the 9th International Language Resources and Evaluation Conference (LREC 2014)*. Reykjavik, Iceland.

Voigt, V., Zipser, F., and Odebrecht, C. (2016). SaltInfo-Module - the x-ray to your corpus. Poster presented at 38. Jahrestagung der Deutschen Gesellschaft für Sprachwissenschaft, 25 February, Konstanz University, Konstanz, Germany.

Yimam, S. M., Gurevych, I., Eckart de Castilho, R., and Biemann, C. (2013). WebAnno: A flexible, web-based and visually supported system for distributed annotations. System demonstration presented at ACL 2013, 5 August, Sofia, Bulgaria.

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

Zipser, F., Zeldes, A., Ritz, J., Romary, L., and Leser, U. (2011). Pepper: Handling a multiverse of formats. Poster presented at 33. Jahrestagung der Deutschen Gesellschaft für Sprachwissenschaft, 24 February, Göttingen University, Göttingen, Germany.

Zipser, F., Frank, M., and Schmolling, J. (2014). Merging data, the essence of creation of multi-layer corpora. Poster presented at 36. Jahrestagung der Deutschen Gesellschaft für Sprachwissenschaft, 6 March, Marburg University, Marburg, Germany.

11. Language Resource References

corpus-tools.org Development Team. (2015). *corpus-tools.org tool set for multi-layer corpora*. Humboldt-Universität zu Berlin and Friedrich Schiller University Jena, 1.0, ISLRN pending.

Friedrich Schiller University Jena. (2015). *Atomic*. LinkType, Tools for crosslinguistic multi-level annotation, 0.2.1, ISLRN pending.

Humboldt-Universität zu Berlin. (2015). *ANNIS*. Sonderforschungsbereich 632 “Information structure”, 3.3, ISLRN pending.

Humboldt-Universität zu Berlin. (2016a). *Pepper*. Sonderforschungsbereich 632 “Information structure”, 3.0, ISLRN pending.

Humboldt-Universität zu Berlin. (2016b). *Salt*. Sonderforschungsbereich 632 “Information structure”, 3.0, ISLRN pending.