TEXTANNOTATOR: A flexible framework for semantic annotations

Giuseppe Abrami
Goethe University Frankfurt
abrami@em.uni-frankfurt.de

Alexander Mehler
Goethe University Frankfurt
mehler@em.uni-frankfurt.de

Andy Lücking
Goethe University Frankfurt
luecking@em.uni-frankfurt.de

Elias Rieb
Goethe University Frankfurt
s0294036@stud.uni-frankfurt.de

Philipp Helfrich
Goethe University Frankfurt
helfrich@cs.uni-frankfurt.de

Abstract

Modern annotation tools should meet at least the following general requirements: they can handle diverse data and annotation levels within one tool, and they support the annotation process with automatic (pre-)processing outcomes as much as possible. We developed a framework that meets these general requirements and that enables versatile and browser-based annotations of texts, namely TEXTANNOTATOR. It combines NLP methods of pre-processing with methods of flexible post-processing. In fact, machine learning (ML) requires a lot of training and test data, but is usually far from achieving perfect results. Producing high-level annotations for ML and post-correcting its results are therefore necessary. This is the purpose of TEXTANNOTATOR, which is entirely implemented in ExtJS and provides a range of interactive visualizations of annotations. In addition, it allows for an flexible integration of knowledge resources. The paper describes TEXTANNOTATOR's architecture together with three use cases: annotating temporal structures, argument structures and named entity linking.

1 Introduction

Among other things, annotation tools must be able to take into account the diversity of data and its multi-level annotation, be easy to use and enable the convertibility of data formats and files (Dipper et al., 2004, p. 55 f.). In view of the current achievements of automatic annotations as a result of *Natural Language Processing* (NLP), one can add to this list of requirements the integration of NLP-based pre- and post-processing to support and facilitate human annotation. To avoid using multiple tools when addressing different annotation tasks and to combine efficient pre-processing with flexible post-annotation we developed TEXTANNOTATOR as a framework that enables versatile and browser-based annotations of texts while using the pre-processing methods of TEXTIMAGER (Hemati et al., 2016). In this paper, we compare TEXTANNOTATOR with related tools in Sec. 2, describe its architecture in Sec. 3 and its components in Sec. 4. To this end, we consider three annotation objects: temporal structure, argumentation structure and named entity linking. Finally, in Sec. 5 we conclude and in Sec. 6 we give an outlook on upcoming development steps. In order to prepare the subsequent sections, we start by outlining TEXTANNOTATOR's main features.

As a matter of fact, machine learning-based NLP tools require a lot of training and test data, but are usually far from achieving perfect results (Pinto et al., 2016; Gleim et al., 2019). Since tools propagate their errors to subsequent processing steps, TEXTANNOTATOR provides an interface for editing such errors. To this end, TEXTANNOTATOR addresses the following requirements:

a) **Customizable annotation scheme:** TEXTANNOTATOR's annotation scheme is implemented by means of native UIMA TYPE SYSTEM DESCRIPTORS (Götz and Suhre, 2004). In this way, its annotation scheme can be flexibly extended or modified, for example, to reflect changing requirements in the course of an annotation project. Such an annotation scheme has already been created by Helfrich et al. (2018) to map, for example, data structures on the level of hypergraphs.

b) **Resource management:** For managing different corpora, TEXTANNOTATOR uses the so-called *ResourceManager* (Gleim et al., 2012) which processes text files of various formats (HTML, TEI, XMI, etc.).

c) **User management:** For managing annotation projects, TEXTANNOTATOR uses the *AuthorityManager* (Gleim et al., 2012) which allows for specifying user- and group-related access permissions.

d) **Pre-processing:** In order to profit from automatic pre-processing of text corpora, TEXTANNOTATOR uses the range of tools of TEXTIMAGER (Hemati et al., 2016). This includes *inter alia* the following annotation tasks: tokenization, sentence splitting, POS tagging, lemmatization, morphological tagging, named entity recognition (NER), time recognition, sentiment analysis, disambiguation, and topic labeling. These tasks are performed for a wide range of languages using several NLP pipelines.

e) **Annotation interface:** TEXTANNOTATOR's interface offers means for selecting, annotating and visualizing information objects (text spans and their annotations). It is developed as a browser-based system using the ExtJS[1] framework. The interface has already been evaluated in Helfrich et al. (2018) by example of annotating rhetorical structures according to RST (Mann and Thompson, 1988) – see also Sec. 4.6.

f) **Storage solutions:** Managing different storage and export formats is essential for processing annotations in NLP. By means of UIMA *Type System Descriptors*, TEXTANNOTATOR uses a standard for representing such annotations (Biemann et al., 2017). As a result, annotations can be exported or saved with the help of UIMA's exchange format *XMI*. However, a purely document-based approach leads to problems with regard to redundancy management and the lack of DB-related query options: The reason being that for each document, information units have to be annotated separately and duplicated over individual annotation documents. This in turn leads to an increase in document size, which delays any queries on these documents. In order to prevent this, TEXTANNOTATOR's management of UIMA documents is based on the UIMA DATABASE INTERFACE (Abrami and Mehler, 2018). The latter provides an interface for generic database support that are not offered for UIMA documents natively, so that annotation objects can be centrally stored and referred to from individual documents.

g) **Knowledge bases:** Enriching texts with information from various knowledge source such as Wikidata (`https://www.wikidata.org/wiki/Wikidata:Main_Page`) or WordNet (`https://wordnet.princeton.edu/`, Fellbaum (1998)) is essential for many NLP tasks (e.g., NER or topic labeling (Uslu et al., 2019)). TEXTANNOTATOR allows for annotating texts with such information. This includes a wide range of knowledge resources that provide a Web API as Wikipedia, Wikidata, Wiktionary and GeoNames.

h) **Distributed and augmented annotation:** In order to enable distributed annotations in the sense that various annotators have (simultaneous) access to the same annotation files and single annotators have access to their annotation files independent from their location, the use of browser-based annotation tools has become a *de facto* standard (see also the tools covered in Sec. 2, as well as in Biemann et al., 2017). At the same time, more and more advanced devices such as smartphones, tablets, or virtual reality (VR) glasses not only support mobile and ubiquitous access, but also

[1] `https://www.sencha.com/products/extjs/`

three-dimensional and augmented reality (AR) visualizations. Anticipating future applications which increase operability due to the immersive impact of VR and AR techniques, the backend of TEXTANNOTATOR is designed to connect to platforms like VANNOTATOR (Spiekermann et al., 2018) or RESOURCES2CITY (Kett et al., 2018). This allows access to the same data in different virtual ways. Note that VANNOTATOR also meets the requirement of outdoor annotation (cf. Wither et al., 2009). Both tools, VANNOTATOR and RESOURCES2CITY, were developed to visualize and annotate multimodal information units in virtual environments, in the case of VANNOTATOR this also includes extended, three-dimensional environments.

i) **Simultaneous collaboration:** With regard to cooperative annotation tasks, the processes involved need to be coordinated. For instance, possibly complex annotation tasks have to be carried out by several annotators. Furthermore, joint annotation gives rise to a displaying challenge: annotations from other annotators as well as automatic annotations from pre-processing steps needs to be displayed. Moreover, each annotation must be fingerprinted to determine who authored it (e.g., a human annotator or a pre-processor). Since TEXTANNOTATOR's backend communicates via WebSockets, annotations can be performed in real time and platform-independently, and visualized to all participants.

Note that some of these features still undergo further extensions: the knowledge bases from g) are continuously augmented by additional ontologies, distributed and augmented annotation from h) is adapted to more tools of TEXTANNOTATOR's backend. All these further developments of TEXTANNOTATOR are available via the website `www.textannotator.texttechnologylab.org`. For each annotation component described in this paper in Sec. 4, completely annotated sample documents are available.

As explained in the following section, there exists already a number of different annotation tools for different annotation tasks. However, it is usually time-consuming or difficult to change annotation tools within a running project in order to continue working on the same document (e.g. due to a change or extension of the annotation focus). To facilitate this, we are pursuing the idea of making UIMA documents interchangeable within our uniform annotation system, so that TEXTANNOTATOR can load and process already annotated documents. In addition, any annotation results can be exported and reused by our UIMA-based tool to manage a specific project.

2 Related Work

There exists a wide range of web-based annotation tools for many annotation tasks. Annotation tools like GATE (Cunningham et al., 2013), which are not primarily web-based, are not considered here in detail (see, e.g., Wilcock, 2017 for a respective overview). In any event, because of not being based on the UIMA framework, GATE falls short of accounting for features a), partially d), and g)–i) of Sec. 1. In contrast to this, BRAT (Stenetorp et al., 2012), which has been the source of the WEBANNO annotation framework (de Castilho et al., 2014), is based on the UIMA architecture. However, BRAT and WEBANNO are specialized tools in the sense that their annotation schemes and graphical interfaces are intrinsically related to a certain class of annotation tasks. They aim at mapping continuous text spans to labels and linking them as instances of certain relations (e.g. co-reference). That is, in order to add, for instance, annotations of rhetorical discourse structure one has to change to another, presumably likewise specialized, tool. The difference between TEXTANNOTATOR and these tools is its strict implementation as a web-based solution and its modular architecture (Sec. 4) for integrating modules for special annotation tasks. That is, TEXTANNOTATOR follows a more modular approach so that extensions can be developed within the TEXTANNOTATOR framework.

At the same time, the flexible maintenance of user rights and the ability to integrate web-based knowledge resources such as Wikidata, WordNet or Wiktionary are desirable features of current annotation tools. There are tools that focus on utilizing knowledge resources in the latter sense: For instance,

BABELFY (Moro et al., 2014) integrates different knowledge databases, but does not allow for post-processing of its annotations. With INCEPTION (Klie et al., 2018) the authors provide a tool for interactive and semantic annotation of texts supported by knowledge databases, providing an active learning unit supporting the annotators.

TEXTANNOTATOR is developed in such a way that it adheres to state-of-art technologies and meets real world annotation requirements (see Sec. 1), but at the same time tries to avoid architectural or conceptual limitations as known from many years of annotation experience (Gries and Berez, 2017).

3 Architecture

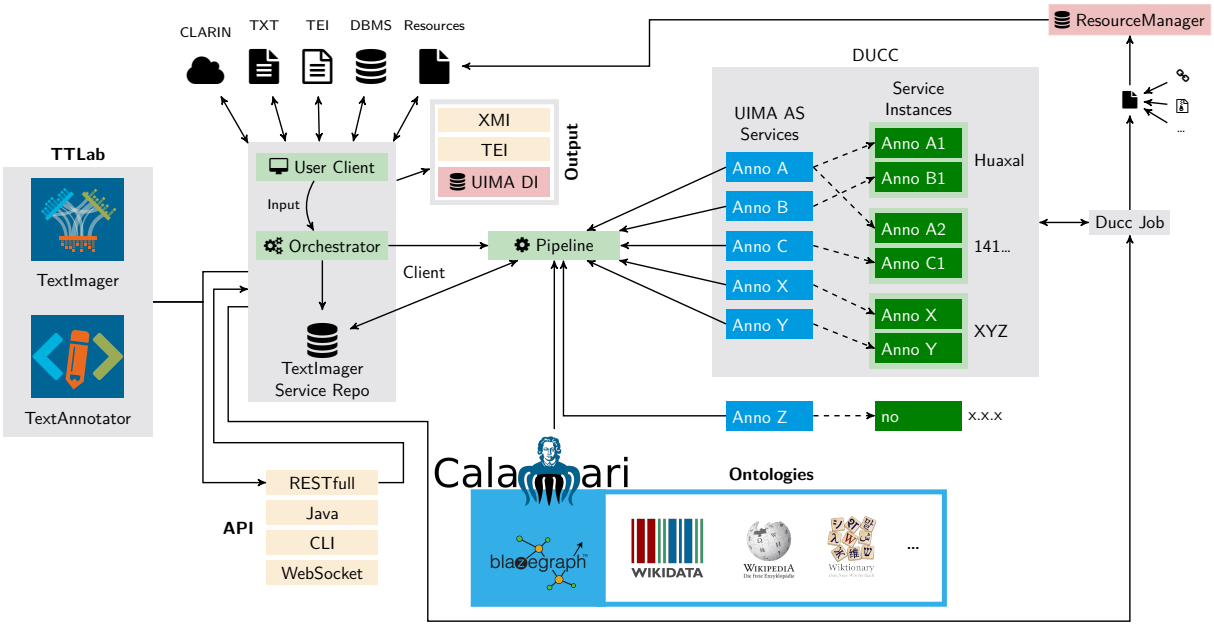

Figure 1: The system architecture with all system components used by TEXTANNOTATOR.

The architecture of TEXTANNOTATOR is primarily based on the consistent use of the UIMA framework, which supports current requirements of annotation processes (Wilcock, 2017). This means that TEXTANNOTATOR allows the pre-processing and analysis of plain text and of text corresponding to the UIMA format. Generally speaking, we distinguish two types of architectures: the architecture of underlying system components and the architecture of TEXTANNOTATOR itself. Both are described in the following sections.

3.1 System Components

The distributed system, the basis of TEXTANNOTATOR, is illustrated in Fig. 1. On the left-hand side, the two independent but interlinked main tools are shown, namely, TEXTIMAGER and TEXTANNOTATOR. Both of them access the TEXTIMAGER SERVICE REPOSITORY using a browser interface implemented in ExtJS via a RESTful webservice. In addition, the API of TEXTIMAGER can be used independently via RESTful. Every request is taken on by the so-called ORCHESTRATOR, which assembles the underlying UIMA pipelines and orders them in time (synchronous/asynchronous; hence its name). As a result of such a pre-processing pipeline, a UIMA document is created, which can be exported in different formats (XMI, TEI) as shown in Fig. 1, "Output". Beyond that, the results can be stored by means of the UIMA DATABASE INTERFACE. However, TEXTANNOTATOR is not limited to UIMA, since TEXTIMAGER can process and convert different formats, implementing the convertibility of data formats, a requirement identified in Sec. 1.

Finally, different NLP resources can be transferred from the local system to UIMA pipelines using the web interface or the RESOURCEMANAGER (top right of Fig. 1). The gray box displays the *Apache DUCC*[2] architecture, a Linux cluster controller that allows for scaling any UIMA pipeline to high data throughput and for low latency real-time applications running on distributed systems in multiple threads. This means that large corpora can be processed much faster with DUCC than with conventional methods. This approach allows a flexible, synchronous and asynchronous pre-processing of texts, realizing the corresponding feature d) of Sec. 1.

The lower part of Fig. 1 shows the CALAMARI system, our currently developed database solution based on Blazegraph[3] for integrating various knowledge databases such as Wikidata or Wiktionary. CALAMARI aims at flexibly handling knowledge resources to meet Requirement g) of Sec. 1.

3.2 TEXTANNOTATOR

TEXTANNOTATOR consists of two main components: a component for selecting input documents together with pipelines from TEXTIMAGER to be used for pre-processing, and a component that includes all annotation modules currently provided by TEXTANNOTATOR. The former component implements three ways of selecting input documents: one can insert a plain text, load a document with the help of RESOURCEMANAGER or insert a URL to extract the corresponding web content. The second component is responsible for the annotation and visualization of input documents; both components are further described in Sec. 4. TEXTANNOTATOR's frontend is built with the help of ExtJS[4]. It makes use of a combination of the *Model View Controller* pattern and a loose data binding provided by the framework. Through a strict modularization, as forced by this pattern, new annotation and visualization components can be easily integrated into TEXTANNOTATOR. Each annotation tool is implemented in its own panel, which inherits from an underlying base panel. The latter is controlled by a controller and serves as an abstraction layer to handle events and actions that are common to all annotation tools. This allows TEXTANNOTATOR for using different visualization libraries and annotation tools while handling them on an equal footing. To this end, the annotation panels use the visualization library d3.js. New modules can be derived from given panels extending the aforementioned base panel and automatically provide an SVG after rendering. Because annotation requirements increase with the capabilities of web browsers, additional classes for new panel types such as Canvas[5], OpenGL, Unity3D[6] etc. can and will be included in future versions of TEXTANNOTATOR. All annotation panels can perform annotations of NLP documents with graphical assistance, where changes are directly realized in the underlying XMI representation of the document. These changes are also stored in the underlying UIMA database (Abrami and Mehler, 2018).

All annotations are defined as a *UIMA TypeSystemDescriptor*, which enables their porting. For all annotation components for which there was currently no *UIMA TypeSystemDescriptor available*, such descriptors were created accordingly. This concerns the annotation components TIMEANNOTATOR, ARGUMENTANNOTATOR and KNOWLEDGEBASELINKER.

In addition, a separate annotation panel for each document is created and displayed within a split window. This allows different documents to be compared with each other. Furthermore, TEXTANNOTATOR basically allows the distributed, simultaneous annotation of the same text by different users. These annotations from different users are stored in shared or separate logical user views. During post-processing, the different user annotations can be compared with each other (e.g. regarding agreement). Within the TEXTANNOTATOR this feature does not yet exist and is planned for future developments, however. This feature mainly addresses requirement i) and allows multiple users to simultaneously work on a document and mutually view changes in real time.

[2] https://uima.apache.org/doc-uimaducc-whatitam.html
[3] https://www.blazegraph.com/
[4] https://www.sencha.com/products/extjs/
[5] https://www.w3.org/TR/2dcontext/
[6] https://unity3d.com

Figure 2: Visualization of a *split window* using QUICKANNOTATOR to visualize two documents. The quick annotations can be pre-selected in the *OptionPanel* and afterwards annotated in the text. The different named entities and common nouns are distinguished by coloured highlighting. The texts shown are extracted from the *BIOfid* project on biodiversity (`https://www.biofid.de/en/`).

4 Components

TEXTANNOTATOR currently contains six annotation components: QUICKANNOTATOR, PROPOSITIONANNOTATOR, TREEANNOTATOR, TIMEANNOTATOR, ARGUMENTANNOTATOR and KNOWLEDGEBASELINKER. In order to shorten the time required to get used to the different annotation views and to create a clear user interface, each of these annotation modules uses uniform navigation and selection mechanisms as shown in Fig. 4.

As this paper is limited in size, only four components can be presented; TREEANNOTATOR has already been described in detail in (Helfrich et al., 2018). We skip PROPOSITIONANNOTATOR, that allows for the annotation of predicate-argument structures.

4.1 QUICKANNOTATOR

The so called QUICKANNOTATOR enables the fast annotation of lexical semantic structures. These currently include the annotation of named entities, common nouns and taxa as well as the annotation of sentence boundaries. This can be done by pre-selecting the types defined in the *OptionPanel* or by directly selecting the tokens. In the first case the selected token is annotated directly, in the latter case all annotation possibilities are displayed to perform the annotation. Named entities can be assigned to kinds, which are distinguished by color on display – cf. Fig. 2. Note that named entities can be annotated recursively. In this case, a "hierarchical merge" is applied and visualized by means of blue borders, as shown in Fig. 3. Furthermore, QUICKANNOTATOR can be used to create comments for the text units, which are visualized by note symbols on each token.

The primary focus of QUICKANNOTATOR is its capacity to perform annotations rapidly and easily. For more complex annotations, the targeted annotation views are designed.

4.2 TREEANNOTATOR

TREEANNOTATOR (Helfrich et al., 2018) is a graphical tool for annotating tree-like structures, in particular structures that jointly map dependency relations and inclusion hierarchies as used by RST (Mann and Thompson, 1988). Its inclusion expands the area of text data structures covered by TEXTANNOTATOR by a number of functions that go beyond sequence labeling and linking, as currently addressed

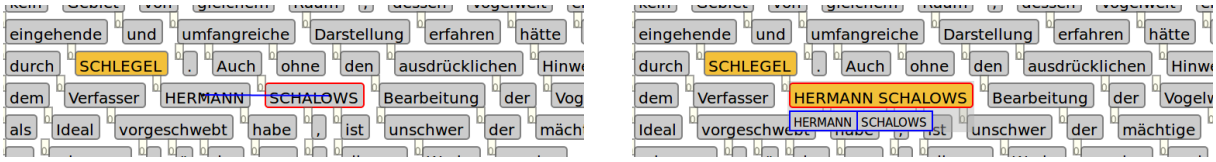

Figure 3: Creating multitokens: the left side shows two tokens manifesting the two parts of a compound proper name. The tokens are not individually annotated as *person* but combined to a multitoken which is then annotated as *person* (right). The right side also shows the combined tokens; the multitoken can be separated again at any time.

by tools such as BRAT or WEBANNO. For more details on this annotation module see Helfrich et al. (2018).

4.3 ARGUMENTANNOTATOR

ARGUMENTANNOTATOR allows the annotation of complex argumentation structures of texts, following the approach of Freeman (1991) which provides different *composition schemata*. As shown in Fig. 4, the individual arguments from the TEXTPANEL can be selected in the annotation window in order to transform them into an argumentation structure using the *schemata* from the OPTIONPANEL. The combination of these schemes is illustrated in Stede (2007, p. 113f), where the argument structure of the underlying sample text is explicated.

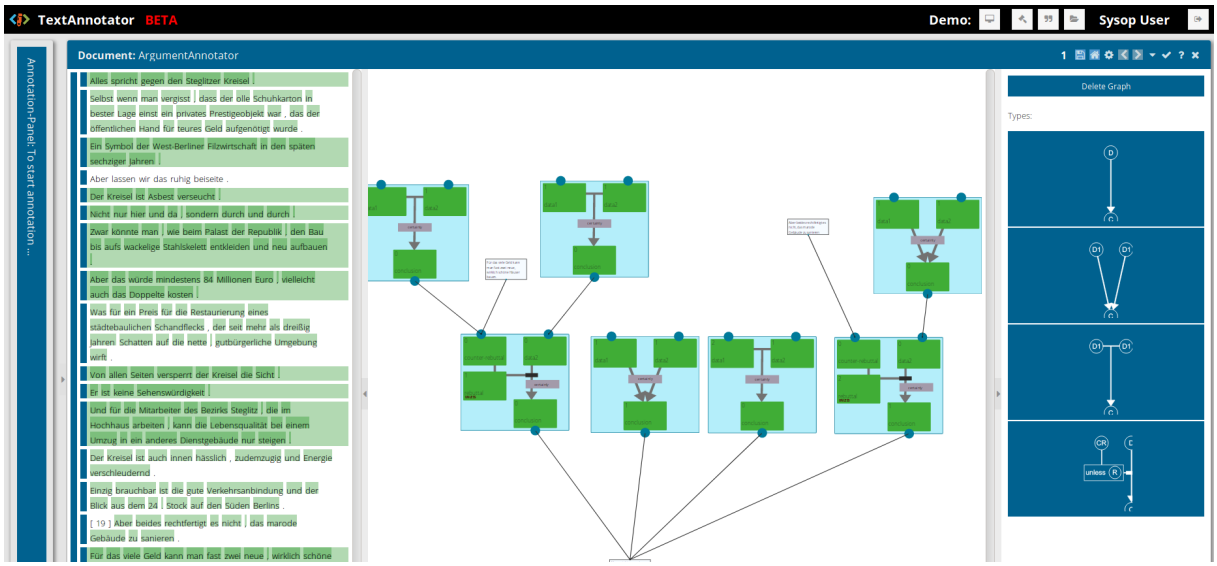

Figure 4: Annotation of argument structures. The central area (*AnnotationPanel*) of each annotation view contains the visualization of the annotations. The left area (TEXTPANEL) visualizes the raw text and highlights its paragraph, sentence and token level in order to facilitate the selection of the text segments to be annotated. The right area (OPTIONPANEL) shows the argument structures according to Freeman (1991) for annotation.

4.4 TIMEANNOTATOR

TIMEANNOTATOR (see Fig. 5) enables the annotation of basic temporal structures in texts. The underlying annotation scheme used is that of Mani and Pustejovsky (2004), according to which temporal structure is expressed in terms of the relative sequence of the events described in a text, where this relative sequence is represented as a tree. In TIMEANNOTATOR, this approach is complemented by that of

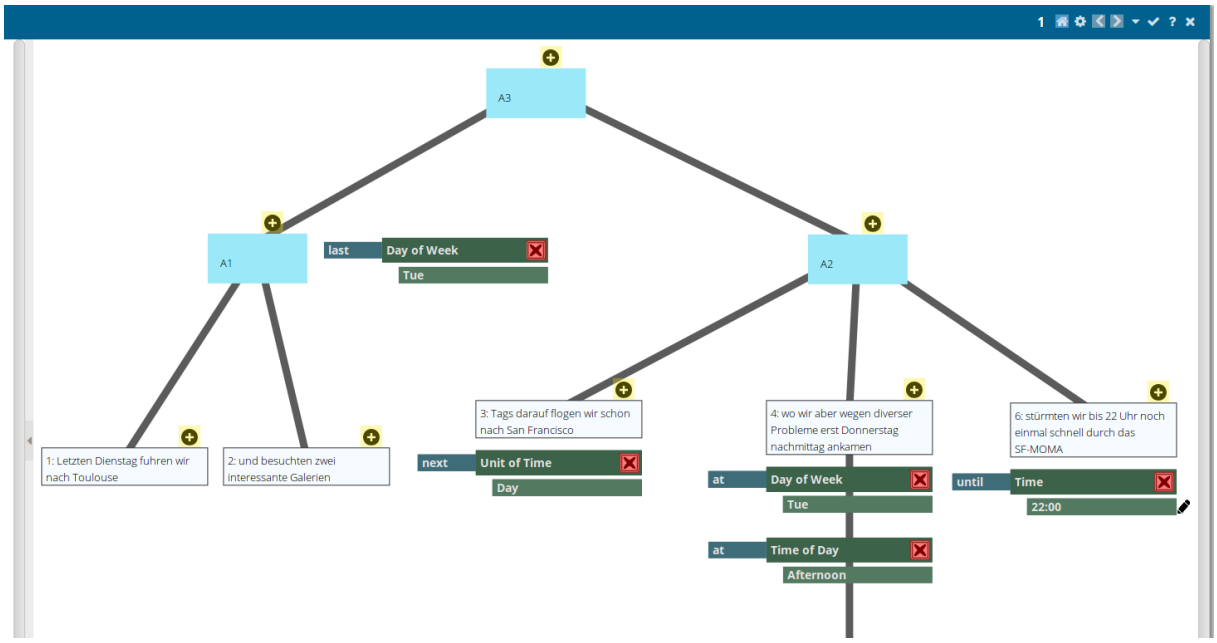

Figure 5: Annotation of temporal structures by means of TIMEANNOTATOR. Selected text segments (white) are grouped hierarchically (blue) and can be connected to temporal instances (green). Note that the *TextPanel* is collapsed in this screenshot.

Stede (2007), which also captures single points in time. Thus, each node in a time tree can receive temporal annotations in the form of concrete time stamps, time spans or relational expressions (cf. Table 1). This assignment takes place in two stages where a relative time specification is followed by a temporal instance. Subsequently, values can be selected (e.g. *Day Of Week: Mon, Thu, ...*) or entered freely (*Time, Other*). Moreover, temporal instances are not limited to text segments, but also grouped nodes can be annotated accordingly.

4.5 KNOWLEDGEBASELINKER

TEXTANNOTATOR provides an annotation panel for *Named Entity* annotation, namely KNOWLEDGEBASELINKER (KBL). The visualization of the KBL panel was inspired by BABELFY[7]; in comparison to BABELFY, however, KBL supports the integration and linking of any number of knowledge databases.

Designed to expand on existing tools, TEXTANNOTATOR's KBL approach combines already implemented NLP tools from TEXTIMAGER with an easy to use graphical interface. Each token is automatically represented by an annotation box, which holds references and quick overviews for each linked ontology, including images, hyperlinks and short descriptions (see Fig. 6). Currently, Wikidata, Wikipedia, Wiktionary, Geonames[8], the German National Library[9], GermaNet and Babelfy (Moro et al., 2014) are accessible and can quickly be searched within TEXTANNOTATOR. For a more fine-grained NE categorization, we developed the TTLAB NAMED ENTITY TYPE system, which distinguishes 15 different NE types and 90 subtypes (see Nagel (2008); Debus (2012); Kamianets (2000); Brendler (2004); Vasil'eva (2011), Wiktionary, Urban Dictionary[10] and the ICOS List of Key Onomastic Terms[11]). In addition, relations between Wikidata elements (direct connections based on Wikidata properties) are visualized as directed edges at the bottom of the display, allowing users to quickly detect relations between annotated objects, where the type of relation is depicted as an edge label (see Fig. 6).

[7] http://babelfy.org
[8] http://www.geonames.org/
[9] http://www.dnb.de
[10] www.urbandictionary.com, accessed February 28, 2019
[11] https://tinyurl.com/y4ubpzdc, accessed February 28, 2019

Relation		Temporal instance
at		Day of Week
next		Time of Day
previous		Month
before		Day
after	×	Year
during		Time
first		Date
last		Unit of Time
		Other

Table 1: Selection options for temporal annotation in TIMEANNOTATOR: possible combinations of time relations and temporal instances.

KBL makes it possible to quickly annotate text streams by mapping proper names and common nouns to elements of ontological resources, thereby generating data that can be used to train taggers to automatically perform these tasks. This corresponds exactly to a current application scenario of KBL in the BIOfid project.

4.6 Usability

Since TEXTANNOTATOR allows for various kinds of annotations, comparing its usability with other, usually more specialized annotation tools has to focus on specific annotation tasks which define the intersection between the tools to be evaluated. So far, the usability of TEXTANNOTATOR has been tested in a comparative evaluation study regarding the annotation of rhetorical structures using TREEANNOTATOR, which is reported in (Helfrich et al., 2018). In sum, RST annotation based on TREEANNOTATOR requires less clicks, proceeds faster, and produces less errors than the corresponding annotation based on the RSTTOOL (O'Donnell, 1997, 2000). This evaluation shows that the user interface of TEXTANNOTATOR – at least in the example of TREEANNOTATOR – does justice to one of the most important features from the user's point of view, namely ease of use (Dipper et al., 2004, cf. also Sec. 1).

5 Conclusion

In this paper, we presented TEXTANNOTATOR as a novel text annotation tool that addresses different linguistic annotation tasks within the same framework. Given its user-friendliness, state-of-the-art technology, linkage of multilevel annotations, collaborative usability and connection to external knowledge resource, among others, we feel save to claim that TEXTANNOTATOR is the most advanced annotation tool currently around. However, even at this stage there are still many ideas for further improvements, some of which will be concludingly highlighted.

6 Future Work

TEXTANNOTATOR will be further developed regarding various aspects: first of all, the feature to select and annotate discontinuous text segments will be extended. Further, CALAMARI will be expanded through integrating more knowledge resources, such as domain-specific ontologies, for example, from the field of biodiversity. For all (new) visualizations provided by TEXTANNOTATOR, a LaTeX (TiKZ) export will be provided to obtain customizable LaTeX source files as well as high-quality vector graphics, thereby following the example of TREEANNOTATOR. Furthermore, a generic active learning component (e.g. Fang et al., 2017) will be implemented in TEXTANNOTATOR that supports the annotators. In addition, possibilities for collaborative annotation will be expanded. An important issue in this respect is the

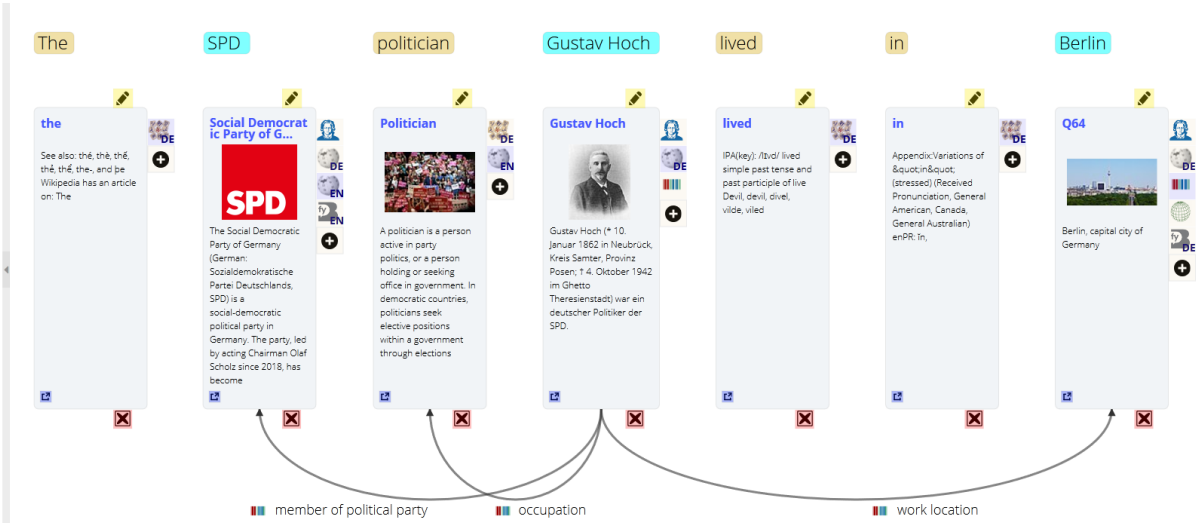

Figure 6: KNOWLEDGEBASELINKER: The selected text "The SPD politician Gustav Hoch lived in Berlin." is graphically displayed in the *AnnotationPanel*. For each recognized token from pre-processing, a query is made in the knowledge databases (in the future this will be done by CALAMARI). Under each token, the notes from the knowledge databases are visualized in groups (gray box), which can be added, edited or removed. Additionally, all tokens, which are connected by a relation in Wikidata, are visualized with a directed edge and the corresponding relation name is displayed.

easy inclusion of annotation schemes: currently, new annotation schemes can only be applied by creating new *UIMA TypeSystemDescriptor*s, which requires a restart of the underlying database. To avoid this and to ensure greater flexibility, it should be possible within TEXTANNOTATOR to utilize schemes based on single or groups of users without having to define them as individual *UIMA TypeSystemDescriptor*s. Finally, once TEXTANNOTATOR has left its current construction phase, it will be published via GitHub.

Acknowledgement

The support by the *German Research Foundation* (DFG) through the BIOfid project (`https://www.biofid.de/en/`) and by the *Federal Ministry of Education and Research* (BMBF) via the project CEDIFOR (https://www.cedifor.de/) are both gratefully acknowledged.

References

Abrami, G. and A. Mehler (2018). A UIMA Database Interface for Managing NLP-related Text Annotations. In *Proc. of LREC 2018*, Miyazaki, Japan.

Biemann, C., K. Bontcheva, R. E. de Castilho, I. Gurevych, and S. M. Yimam (2017). Collaborative web-based tools for multi-layer text annotation. In *Handbook of Linguistic Annotation*, pp. 229–256. Springer.

Brendler, S. (2004). Klassifikation der Namen. In A. Brendler and S. Brendler (Eds.), *Namenarten und ihre Erforschung. Ein Lehrbuch für das Studium der Onomastik*, Chapter 2, pp. 69–92. Hamburg: Baar.

Cunningham, H., V. Tablan, A. Roberts, and K. Bontcheva (2013, 02). Getting more out of biomedical documents with gate's full lifecycle open source text analytics. *PLOS Computational Biology 9*(2), 1–16.

de Castilho, R. E., C. Biemann, I. Gurevych, and S. M. Yimam (2014). WebAnno: a flexible, web-based annotation tool for CLARIN. In *Proc. of CAC'14*, Utrecht, Netherlands.

Debus, F. (2012). *Namenkunde und Namengeschichte. Eine Einführung*. Grundlagen der Germanistik. Berlin: Erich Schmidt Verlag.

Dipper, S., M. Götze, and M. Stede (2004). Simple annotation tools for complex annotation tasks: an evaluation. In *Proceedings of the LREC Workshop on XML-based Richly Annotated Corpora*, LREC 2004, pp. 54–62.

Fang, M., Y. Li, and T. Cohn (2017). Learning how to active learn: A deep reinforcement learning approach. In *Proc. of the EMNLP*, pp. 595–605. Association for Computational Linguistics.

Fellbaum, C. (Ed.) (1998). *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press.

Freeman, J. B. (1991). *Dialectics and the Macrostructure of Arguments, A Theory of Argument Structure*. Number 10 in Studies of Argumentation in Pragmatics and Discourse Analysis. Berlin: de Gruyter Mouton.

Gleim, R., S. Eger, A. Mehler, T. Uslu, W. Hemati, A. Lücking, A. Henlein, S. Kahlsdorf, and A. Hoenen (2019). Practitioner's view: A comparison and a survey of lemmatization and morphological tagging in German and Latin. *Journal of Language Modeling*. Forthcoming.

Gleim, R., A. Mehler, and A. Ernst (2012). SOA implementation of the eHumanities Desktop. In *Service-oriented Architectures (SOAs) for the Humanities: Solutions and Impacts. Proc. of the Joint CLARIN-D/DARIAH Workshop at Digital Humanities Conference 2012*, pp. 24–29.

Götz, T. and O. Suhre (2004). Design and implementation of the UIMA Common Analysis System. *IBM Systems Journal 43*(3), 476–489.

Gries, S. T. and A. L. Berez (2017). Linguistic annotation in/for corpus linguistics. In N. Ide and J. Pustejovsky (Eds.), *Handbook of Linguistic Annotation*, pp. 379–409. Dordrecht: Springer Netherlands.

Helfrich, P., E. Rieb, G. Abrami, A. Lücking, and A. Mehler (2018). TreeAnnotator: Versatile Visual Annotation of Hierarchical Text Relations. In *Proc. of LREC 2018*, Miyazaki, Japan.

Hemati, W., T. Uslu, and A. Mehler (2016). TextImager: a Distributed UIMA-based System for NLP. In *Proc. of COLING 2016 System Demonstrations*. Federated Conference on Computer Science and Information Systems.

Kamianets, W. (2000). Zur Einteilung der deutschen Eigennamen. *Grazer Linguistische Studien 54*, 41–58.

Kett, A., G. Abrami, A. Mehler, and C. Spiekermann (2018). Resources2City Explorer: A system for generating interactive walkable virtual cities out of file systems. In *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings*, UIST '18 Adjunct, pp. 123–125.

Klie, J.-C., M. Bugert, B. Boullosa, R. E. de Castilho, and I. Gurevych (2018, Juni). The inception platform: Machine-assisted and knowledge-oriented interactive annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pp. 5–9. Association for Computational Linguistics.

Mani, I. and J. Pustejovsky (2004). Temporal discourse models for narrative structure. In *Proceedings of the 2004 ACL Workshop on Discourse Annotation*, DiscAnnotation '04, Stroudsburg, PA, USA, pp. 57–64. Association for Computational Linguistics.

Mann, W. and S. Thompson (1988). Rhethorical structure theory: Towards a functional theory of text organization. *Text 8*(3), 243–281.

Moro, A., A. Raganato, and R. Navigli (2014). Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics (TACL) 2*, 231–244.

Nagel, S. (2008). *Lokale Grammatiken zur Beschreibung von lokativen Sätzen und ihre Anwendung im Information Retrieval*. Ph. D. thesis, Ludwig-Maximilians-Universität München.

O'Donnell, M. (1997). RST-Tool: An RST analysis tool. In *Proc. of ENLG 1997*.

O'Donnell, M. (2000). RSTTool 2.4 – A markup tool for rhetorical structure theory. In *INLG'2000 Proceedings of the First International Conference on Natural Language Generation*, pp. 253–256.

Pinto, A., H. G. Oliveira, and A. O. Alves (2016). Comparing the Performance of Different NLP Toolkits in Formal and Social Media Text. In M. Mernik, J. P. Leal, and H. G. Oliveira (Eds.), *5th Symposium on Languages, Applications and Technologies (SLATE'16)*, Volume 51 of *OpenAccess Series in Informatics (OASIcs)*, Dagstuhl, Germany, pp. 3:1–3:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Spiekermann, C., G. Abrami, and A. Mehler (2018, may). VAnnotatoR: a Gesture-driven Annotation Framework for Linguistic and Multimodal Annotation. In J. Pustejovsky and I. van der Sluis (Eds.), *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Stede, M. (2007, 01). *Korpusgestützte Textanalyse : Grundzüge der Ebenen-orientierten Textlinguistik*. Narr Studienbücher.

Stenetorp, P., S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii (2012). BRAT: a web-based tool for NLP-assisted text annotation. In *Proc. ofEACL 2012*, Avignon, France, pp. 102–107.

Uslu, T., A. Mehler, and D. Baumartz (2019). Computing Classifier-based Embeddings with the Help of text2ddc. In *Proceedings of the 20th International Conference on Computational Linguistics and Intelligent Text Processing, (CICLing 2019)*, CICLing 2019.

Vasil'eva, N. (2011). Die Terminologie der Onomastik, ihre Koordinierung und lexikographische Darstellung. In K. Hengst and D. Kremer (Eds.), *Namenkundliche Informationen*, Volume 99/100, pp. 31–45. Leipzig: Leipziger Universitätsverlag.

Wilcock, G. (2017). The evolution of text annotation frameworks. In N. Ide and J. Pustejovsky (Eds.), *Handbook of Linguistic Annotation*, pp. 193–207. Dordrecht: Springer Netherlands.

Wither, J., S. DiVerdi, and T. Höllerer (2009). Annotation in outdoor augmented reality. *Computers & Graphics 33*(6), 679–689.