Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 891–900
Marseille, 11–16 May 2020

TEXTANNOTATOR: A UIMA based tool for the simultaneous and collaborative annotation of texts

Giuseppe Abrami, Manuel Stoeckel, Alexander Mehler
Goethe University Frankfurt
Robert-Mayer-Strasse 10
60325 Frankfurt am Main / Germany
{abrami, mehler}@em.uni-frankfurt.de, manuel.stoeckel@stud.uni-frankfurt.de

Abstract

The annotation of texts and other material in the field of digital humanities and Natural Language Processing (NLP) is a common task of research projects. At the same time, the annotation of corpora is certainly the most time- and cost-intensive component in research projects and often requires a high level of expertise according to the research interest. However, for the annotation of texts, a wide range of tools is available, both for automatic and manual annotation. Since the automatic pre-processing methods are not error-free and there is an increasing demand for the generation of training data, also with regard to machine learning, suitable annotation tools are required. This paper defines criteria of flexibility and efficiency of complex annotations for the assessment of existing annotation tools. To extend this list of tools, the paper describes TEXTANNOTATOR, a browser-based, multi-annotation system, which has been developed to perform platform-independent multimodal annotations and annotate complex textual structures. The paper illustrates the current state of development of TEXTANNOTATOR and demonstrates its ability to evaluate annotation quality (inter-annotator agreement) at runtime. In addition, it will be shown how annotations of different users can be performed simultaneously and collaboratively on the same document from different platforms using UIMA as the basis for annotation.

Keywords: UIMA, Annotation, Simultaneous and collaborative use, Inter-Annotator Agreement

1. Introduction

The annotation of texts and other material in the field of digital humanities and Natural Language Processing (NLP) is a common task of research projects. At the same time, the annotation of corpora is certainly the most time- and cost-intensive component in research projects and often requires a high level of expertise according to the research interest. Such annotations enrich the data with additional information and subsequently allow them to be processed according to the needs of the project at hand. For this, over the past years, many annotation tools have been developed for various tasks (for a recent process and feature oriented overview see (Finlayson and Erjavec, 2017); for an overview of multimodal tools see (Cassidy and Schmidt, 2017)). In addition to manual annotation tools, there is a proportionately larger number of automatic pre-processing tools. Despite the diversity of automated pre-processing applications (e.g. (Eckart de Castilho and Gurevych, 2014; Hemati et al., 2016)), a post-correction is often required because no *complete coverage* is achieved. For the efficient correction of errors in automatic pre-processing as well as for the generation of training data for machine learning and their efficient management, it is necessary to use a *suitable* annotation tool. This *suitability* of an annotation tool depends highly on the topic of the annotation task. In this context, it is rather unlikely to annotate, for example, spoken language, handwritten recordings, written text, videos and images with a single tool. However, if different tools are used, a schema language is required with which the annotations achieved with the individual tools can be linked even across modality boundaries.

Interchangeability can be achieved by using the *Unstructured Information Management Architecture* (Ferrucci et al., 2009, UIMA), which is widely implemented in different disciplines (Roeder et al., 2010; Wu et al., 2011). UIMA allows to define software modules for the analysis of unstructured data such as text documents where the results of the analysis are stored with reference to a defined schema description. Thereby a separation is made between two core components, the *Type System Descriptors* (*TSD*) and the *Common Analysis Structure* (CAS) (Götz and Suhre, 2004). CAS is the basic data structure in which the deposited data is analyzed and stored with the meta data (*TSD*).

In addition to interchangeability, there are several technical requirements for annotation tools that are necessary in order to perform flexible and complex annotation tasks:

(a) Multi-Authoring System: A multi-authoring system allows different users to perform annotations in shared documents independently of each other. This includes multiple permission levels to grant different users various views and access permissions to documents, repositories or projects. In any event, different users can annotate the same documents, but not simultaneously.

(b) Multi-Document System: A multi-document system is a tool that includes a repository management for structuring documents by topic-, project- or task-related criteria. The documents which are managed should be logically assigned access permissions for (groups of) users to ensure data security as well as data and document protection. In addition, the utilization of a database is necessary for efficient document processing.

(c) Multi-Annotation System: A multi-annotation system is given when it allows to perform different annotation tasks. In the NLP scope, this means, for example, to annotate named entities, tree structures (RST, time structures, etc.) and to perform disambiguation

using the same tool. Ideally, this means that users can create dynamic annotation schemas on their own.

(d) Multi-Platform System: A multi-platform system is a tool that provides an API that makes it possible to use it platform and software independent.

(e) Multi-Analysis System: A multi-analysis system enables the further processing of annotation data and advanced data analyses independent of the annotation itself. This implies the possibility of instantaneous checking the annotation quality, for example, with *inter-annotator agreement* (c.f. (Meyer et al., 2014, IAA)) methods. In addition, annotation projects based on annotations already performed can be structured accordingly. In this way, it is possible, for example, to annotate only texts for which the IAA value of a selected class is too low.

(f) Collaborative and simultaneous use: Dealing with complex annotation tasks requires that several users can simultaneously edit the same documents at the same time. Simultaneous use allows for a collaborative workflow and should significantly increase the efficiency of annotation. However, it should be possible to disable simultaneous use of the system for certain users or groups as required. To this end, a user and group management system is needed.

Looking at the literature, which is reviewed in Section 2, it is obvious that there is no annotation tool that fulfills all these requirements. Since it should be possible to react to changes in annotation projects project-specifically, flexibly, platform-independently and at short-term, the combination of the requirements in one tool is more than a worthwhile goal. Consequently, we are dealing with a technological gap in the field of annotation systems that is now to be closed with the help of TEXTANNOTATOR (Abrami et al., 2018; Abrami et al., 2019).

In this paper, the current state of development of TEXTANNOTATOR is presented, which is designed as a platform-independent, simultaneous and collaborative multi-annotation system for the multimodal annotation of texts. In the present context, multimodality means, beyond intra- and intertextual annotations, connecting texts and their segments with resources such as images, digitized 3D objects (from VR) or entries from knowledge graphs. The annotations managed within TEXTANNOTATOR are based on UIMA, which is now a *de facto* standard in the area of NLP. In order to preprocess new documents and convert them into the UIMA format, TEXTANNOTATOR uses TEXTIMAGER (Hemati et al., 2016), which provides various pre-processing pipelines for a variety of languages. This results in document-based stand-off annotations that enrich the input texts on the basis of established pipelines. Since the focus of TEXTANNOTATOR are is on NLP and because there are a multitude of automatic pre-processing routines (c.f. Eckart de Castilho and Gurevych (2014), Bethard et al. (2014), Hemati et al. (2016)) as well as a database solution for UIMA documents (Abrami and Mehler, 2018), UIMA builds the core environment for TEXTANNOTATOR to ensure interchangeability and interoperability. However, the existence of interchangeability does not necessarily guarantee the reuse of the annotation schemes defined in UIMA, as demonstrated in (Rak and Ananiadou, 2013). The authors show that many projects develop their own schemes for the execution of annotations, although such schemes are already available through other projects as *TSD*. Even if this results in a valid type system, this is not yet really usable, since a separate analysis engine must be developed for each type system. This is still an unsolved problem that requires considerable research which will be an open task for the further development of UIMA and TEXTANNOTATOR.

The paper is structured as follows: After the overview of existing annotation systems in Section 2, Section 3 gives a detailed description of the architecture of TEXTANNOTATOR, while Section 4 describes the method and concept behind text authoring and Section 5 describes the inter-annotator agreement functionality of TEXTANNOTATOR. In Section 6 TEXTANNOTATOR is evaluated, while Section 7 gives a preview of future development steps of TEXTANNOTATOR. Finally, a summary and conclusion is made in Section 8.

2. Related Work

Annotation tools are used in many ways; some have multimodal applications and enable different types of annotation. However, the number of existing annotation tools is very large, so that the following overview must make a selection by concentrating on text-based tools. In principle, there are many very excellent tools, many of which provide methods that are now part of the standard repertoire.

First we examine *MAE2* (Rim, 2016, Multi-document Annotation Environment), the successor to MAE, which is a Java-based annotation environment for manual annotation of natural language. *MAE2* does not include pre-processing methods and does not use external ones, nor is the annotation web-based or distributed. The tool is multi-document capable, but does not allow for collaborative use and is not a multi-platform system. Furthermore, no complex structures such as trees can be annotated with *MAE2*. However, *MAE2* supports the computation of inter-annotator agreement (IAA) by means of *DKPro* (Meyer et al., 2014). *MAE2* fulfills only partly a subset (i.e. **(b)** and **(e)**) of the above-mentioned requirements for a flexible annotation tool, which was, however, within the scope of the project.

The annotation tool *BRAT* (Stenetorp et al., 2012) focuses on the fast and intuitive annotation of texts in web browsers, supported by NLP pipelines. Using drag and drop, spans of texts can be visualized and annotated using flexible annotation schemes. *BRAT* uses a simple but sufficient user and document management system, which allows users to annotate the same documents simultaneously and collaboratively. Concluding, *BRAT* fulfills, to our knowledge, all requirements except **(d)** and **(e)** and partly **(c)**.

Another annotation tool which is based on *BRAT* is *WebAnno* (Eckart de Castilho et al., 2016). *WebAnno* supports simple document- and user management functionalities and uses *BRAT* for performing and visualizing annotations. In addition, graphically supported annotations can be performed and a function is available for comparing annotations (curator mode) and for calculating the IAA.

However, *WebAnno* does not support a remote API and the collaborative simultaneous annotation of documents is not possible. Thus, *WebAnno* fulfills the requirements **(a)**, **(c)**, **(e)**, and (partially) also **(b)**.

Argo (Rak et al., 2012), an annotation editor based on *U-Compare* (Kano et al., 2009), enables the automatic and manual annotation of texts based on UIMA. *Argo* offers a web-based interface and an API to connect custom services to execute task-related annotation operations. In addition, external automatic pre-processing methods can be used and documents and user access permissions can be managed; the latter in a hierarchical manner, since it allows for group-based curations.

Furthermore, users can collaborate by sharing results, documents, and workflows, but the collaboration is not simultaneous. Thus *Argo* fulfills the requirements **(a)**, **(b)**, **(d)** and partly also **(e)**. By and large, this is a very flexible tool, but it lacks simultaneous use and the ability to annotate more complex text structures in graphical mode.

Another browser-based annotation tool is *Anafora* (Chen and Styler, 2013). *Anafora* is designed to allow multiple users to annotate documents in the same way, but not simultaneously. The documents to be annotated are saved as XML files without any database support. Additionally, the annotation scheme is based on XML, the same format in which the annotations are stored. A human-readable and proprietary annotation schema is used, allowing easy understanding and manual porting, but preventing automatic and systematic migration due to missing standards. Thus, *Anafora* fails to meet or only partly meets most of the requirements enumerated in Section 1.

Tool	(a)	(b)	(c)	(d)	(e)	(f)
MAE2	⊖	●	⊖	⊖	⊕	⊖
Brat	⊕	⊕	●	⊖	⊖	⊕
WebAnno	●	●	⊕	⊖	⊕	⊖
Argo	⊕	⊕	●	⊕	●	⊖
Anafora	⊕	●	⊖	⊖	⊖	●
TEXTANNOTATOR	⊕	⊕	⊕	⊕	⊕	⊕
VANNOTATOR	⊕	⊕	⊕	⊕	⊕	⊕

Table 1: An overview of conformance to the requirements described in Section 1. Legend: Requirement fulfilled (⊕), requirement partially fulfilled (●), requirement not fulfilled (⊖). VANNOTATOR is a tool that uses the TEXTANNOTATOR to execute annotations. For this reason it meets the same requirements as TEXTANNOTATOR, but for the area of 3D virtual environments.

This short overview illustrates, as summarized in Table 1, that many existing and actively used annotation tools only partially fulfill the requirements for adequate and flexible annotation tools. To fill this gap the TEXTANNOTATOR was developed, which is described in the next section.

3. Architecture

The architecture of TEXTANNOTATOR's backend uses a REST- and WebSocket-based Java application which provides the functionality via *Java Spark*[1]. Java Spark, a lightweight framework for implementing web interfaces, allows TEXTANNOTATOR to exchange information via JSON, as well as XML, plaintext or files. Thereby the main communication occurs via the WebSocket, which provides a multitude of commands for the annotation. Via this WebSocket the annotations are executed by the respective client systems as for instance the TEXTANNOTATOR-Web-GUI[2] and the VANNOTATOR (Spiekermann et al., 2018). Regardless the use of WebSockets or REST, Figure 1 shows the components interacting with TEXTANNOTATOR to enable the requirements listed in Section 1.

Through the use of the so-called RESOURCEMANAGER, TEXTANNOTATOR gains access to already pre-processed or not yet pre-processed documents. The selection of available resources in RESOURCEMANAGER is ensured by including the so-called AUTHORITYMANAGER which is used for user authentication. Both tools are part of the *eHumanities Desktop* (Gleim et al., 2012) and operate in this context as standalone components. As TEXTANNOTATOR performs UIMA based annotations, it is necessary that texts are presented in UIMA format in order to be processed. Therefore, texts that already exist in UIMA format and are managed by RESOURCEMANAGER can be annotated directly, while texts that exist in raw form must first be converted to UIMA format. The conversion into the UIMA format and all NLP-related pre-processing operations are executed by TEXTIMAGER (Hemati et al., 2016). TEXTIMAGER provides various preprocessing pipelines for different languages to process documents and creates a UIMA document from them. TEXTIMAGER is also used to create annotations for the application of the tools implemented in TEXTANNOTATOR (e.g. Named Entity Recognition, sentence detection, knowledge base linking), which will be explained in Section 4. Finally, documents that are available in UIMA format, managed by RESOURCEMANAGER and for which AUTHORITYMANAGER has assigned access rights to a user can be annotated by this user. Using both tools, TEXTANNOTATOR enables the annotation of multiple documents based on a repository management system by simultaneously accessing a system for managing user and group access rights. As a result, TEXTANNOTATOR meets the requirements **(a)** and **(b)**.

UIMA supports a document-based approach to text annotation which, as explained in (Abrami et al., 2018), is a bottleneck for larger annotation projects. To avoid this bottleneck, RESOURCEMANAGER and therefore TEXTANNOTATOR use the *UIMA-Database Interface* (UIMA-DI) (Abrami et al., 2018) to store the texts and their annotations. The UIMA-DI represents a dynamic approach to the storage, retrieval and administration of UIMA documents using various database management systems. As a result, TEXTANNOTATOR also fulfills requirement **(b)**.

The latter functionalities and the communication via the WebSocket allow external applications to utilize the backend of TEXTANNOTATOR to perform project-specific annotations. External applications that are not part of TEXTANNOTATOR, such as the *StolperwegeApp* (Mehler et al.,

[1] http://sparkjava.com/

[2] http://www.textannotator.texttechnologylab.org

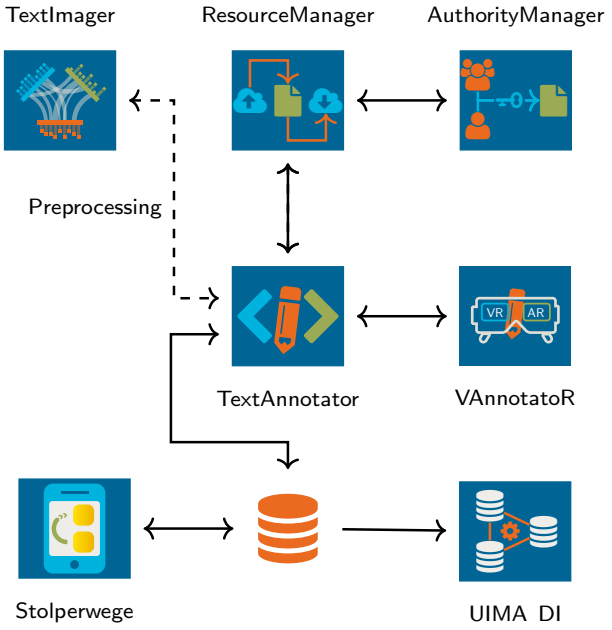

Figure 1: The architecture of TEXTANNOTATOR.

2017) or VANNOTATOR (Spiekermann et al., 2018), for instance, can operate via TEXTANNOTATOR's API or directly through the UIMA-DI. For this purpose, the usage of WebSockets is recommended in order to utilize the methods described in Section 4. In addition, VANNOTATOR implements an interface for virtual and augmented reality using *Unity3D* and 3D glasses as well as *ARCore* from Google to annotate multimodal elements by means of TEXTANNOTATOR. Through VANNOTATOR and tools currently under development, TEXTANNOTATOR is able to meet requirement **(d)**. In addition, the WebSocket allows the simultaneous and joint annotation of the same documents and annotation views. This addresses Requirement **(f)** and allows annotators to work with TEXTANNOTATOR platform independent (Requirement **(d)**) on complex annotation tasks by simultaneously supporting each other. Requirement **(f)** has to be considered differently depending on the project task and does not make sense in every situation. This becomes particularly clear when, for later analyses such as the calculation of an IAA value, a distinction has to be made between which annotator performed which annotation. How this is implemented in TEXTANNOTATOR is described in Section 4 and 5. This will finally relate to the fulfillment of Requirement **(e)** by TEXTANNOTATOR.

In addition to the backend of TEXTANNOTATOR, which is used for the distribution, storage and execution of annotations, the frontend of TEXTANNOTATOR provides the native interface to the user (Abrami et al., 2019). Implemented in *ExtJS*[3] – a framework for JavaScript – this interface gives access to TEXTANNOTATOR's backend and provides, implemented with *d3.js*[4], visualizations of annotations and even allows to annotate these visualizations. This is illustrated in the figures 3 and 4, where the visualisation of annotations can be added or modified.

[3] https://www.sencha.com/products/extjs/
[4] https://d3js.org/

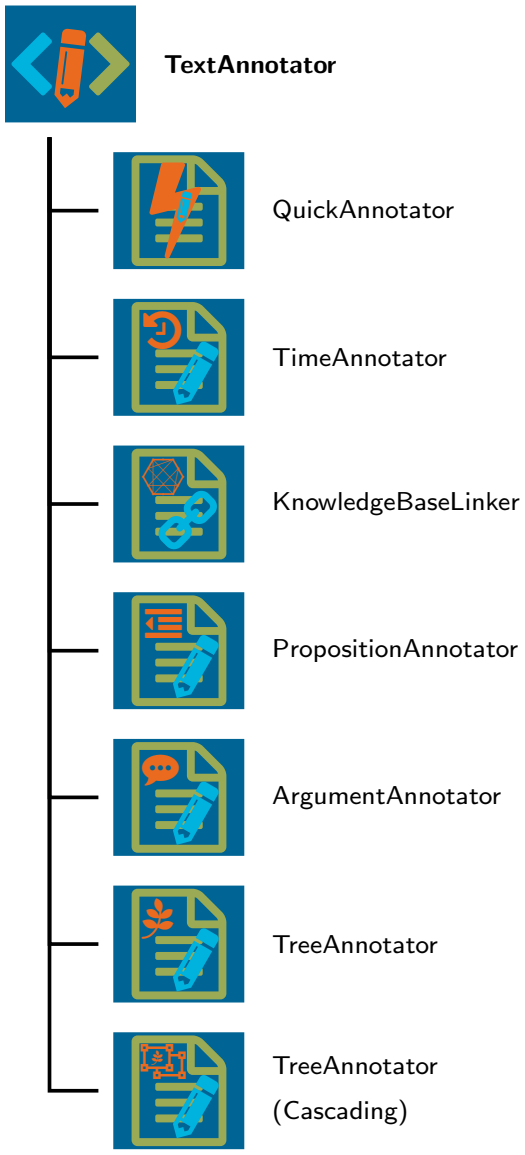

Figure 2: The annotation capabilities of TEXTANNOTATOR.

There are currently seven tools that deal with different annotation tasks (Figure 2). With the help of this tool palette TEXTANNOTATOR fulfills the requirement **(c)**, since several complex annotations can be performed with graphical support. As the tools *TimeAnnotator*, *ArgumentAnnotator*, *KnowledgeBaseLinker* (Abrami et al., 2019), *TreeAnnotator* (Helfrich et al., 2018), have already been described elsewhere, we only add descriptions of *QuickAnnotator*, *TimeAnnotator* and *KnowledgeBaseLinker*.

QuickAnnotator

Using *QuickAnnotator* (Abrami et al., 2019) annotators can quickly perform the same or different annotations on tokens by drag and drop as well as through clicking. This includes the possibility to annotate or de-annotate named entities and to merge single tokens into multi-word expressions. The merging of tokens can be cascaded, enabling nested annotations; at the same time, multi-word expressions can be mapped to multiple named entities.

Figure 3 illustrates a snapshot of an annotation process of the *BIOfid*[5] project which uses *QuickAnnotator*. This project (Koch et al., 2017; Driller et al., 2018) on historical biological literature includes many machine learning tasks to identify biological species that require the generation of training data and the correction of automatic annotations. In addition, *BIOfid*'s longer-term goal is to extract and make available data from historical texts from the field of biology that can be used to investigate processes of climate change over long periods of time. To be flexible, all selectable visual components of *QuickAnnotator* are dynamically customizable, but currently not configurable via a separate graphical user interface.

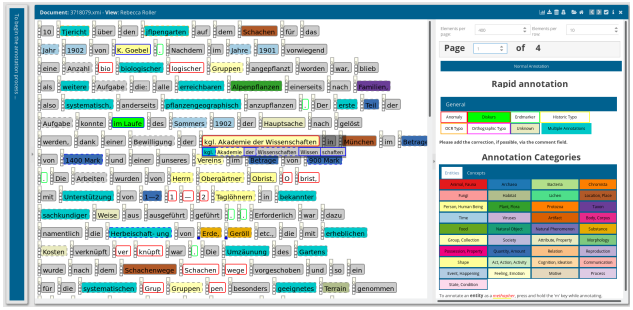

Figure 3: A snapshot of *QuickAnnotator*: The tokens of the input text are presented as interactive elements; on the right side one sees the classes to be annotated.

TimeAnnotator

TimeAnnotator enables the annotation of temporal structures in texts. The underlying annotation scheme (Mani and Pustejovsky, 2004) allows the annotation of a temporal structure as a relative sequence of events described in the input text and represented as a tree. To complement this, we additionally apply the approach of Stede (2007) so that points in time can also be annotated. In other words, each annotated node in a time tree managed by *TimeAnnotator* has the ability to receive temporal annotations in the form of concrete timestamps, time spans or relational expressions (Abrami et al., 2019).

KnowledgeBaseLinker

TEXTANNOTATOR provides an annotation component for annotating named entities, the so-called *KnowledgeBaseLinker* (KBL). The visualization of annotations by KBL is inspired by BABELFY[6]. However, KBL supports the integration and linking of a wide range of knowledge databases based on configurations managed by TEXTANNOTATOR.

Designed to expand on existing tools, KBL 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. 4). Currently, Wikidata, Wikipedia, Wiktionary, Geonames[7], the German National Library[8], GermaNet and Babelfy (Moro et al., 2014) as well as ontologies from die BIOfid project are accessible and can quickly be searched within TEXTANNOTATOR. For a more fine-grained NE detection, 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[9] and the ICOS List of Key Onomastic Terms[10]).

4. Authoring Annotations

All annotations are created using TEXTANNOTATOR's API and each annotation is connected with an active user who has the permission to modify the documents accordingly. These annotations are stored in a UIMA document which is stored in a database managed by UIMA-DI. However, UIMA documents have their own internal structures and in this context only the views are of interest. Regarding text documents, each UIMA document contains the text to be annotated and the set of all annotations in that text. Each annotation is assigned to a *view*, which provides a logical view of the document. In line with UIMA, TEXTANNOTATOR implements this approach and therefore stores annotations within views.

TEXTANNOTATOR distinguishes between so-called *UserViews* and *AnnotationViews*: a separate *UserView* is created for each user who annotates a document. All *UserViews* are owned by certain users, meaning that these views are managed by the AUTHORITYMANAGER. With the help of *UserViews* every user is enabled to annotate in her or his area, which does not only protect their annotations but also enables the distributed annotation of documents. In contrast to this, *AnnotationViews* are created independently of the user and managed as separate resources by RESOURCEMANAGER. This allows arbitrary document views to be created and shared based on user and group permissions. New views can be derived from existing views by copying their contents (see Fig. 5). Thus, documents can be annotated according to different schemata, in different projects or contexts at the same time; a new view can be created for each topic-specific project. In addition, annotators can annotate not only the same documents in different views, but also the same documents in the same views simultaneously. Because of TEXTANNOTATOR's functionality, this simultaneous annotation of identical views is platform independent. All this is performed by means of the TA's WebSocket; Figure 6 gives a visual depiction thereof.

Due to the simultaneity of annotations by different annotators, the question of the authorship of individual annotations arises. This task can also be solved with UIMA: UIMA has not a concept of authorship, so that we go the way of implicitly annotating this information. In addition to the *TSD*'s assigned to the project or task, TEXTANNOTATOR therefore stores annotations to identify their authors. This implicit annotation is defined in UIMA as a *TSD*, so that it can be exchanged platform-independently;

[5] https://www.biofid.de/en/
[6] http://babelfy.org
[7] http://www.geonames.org/
[8] http://www.dnb.de
[9] www.urbandictionary.com, accessed 2019-11-26
[10] https://tinyurl.com/sq4cpbc, accessed 2019-11-26

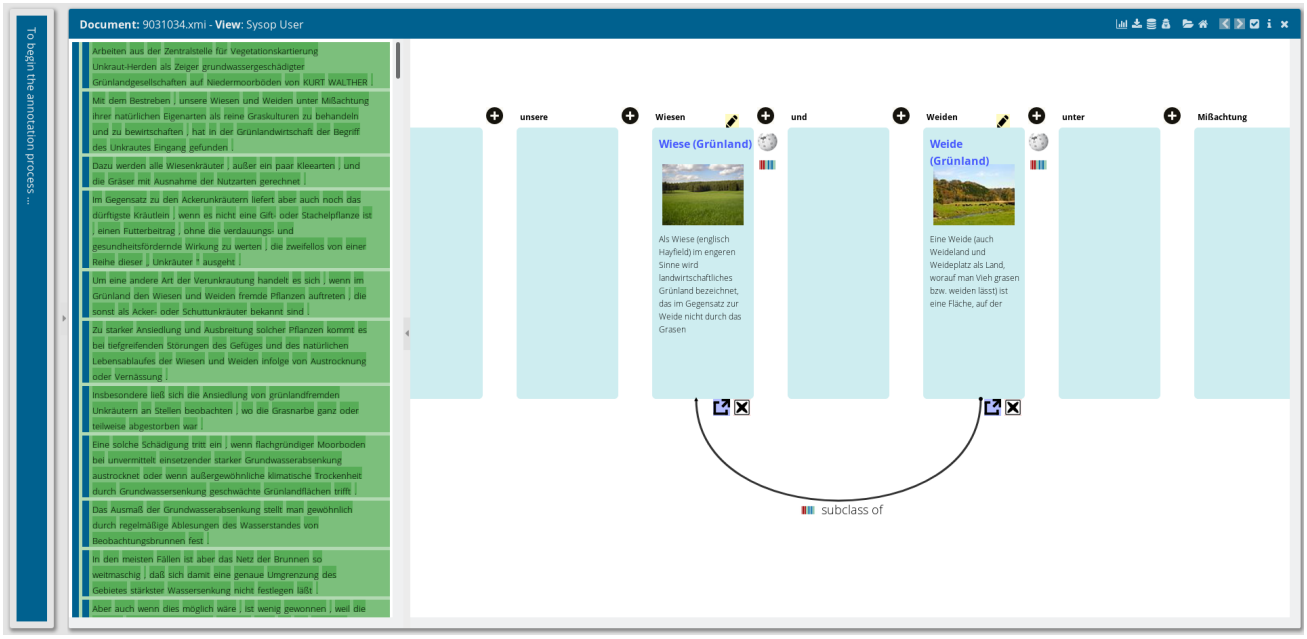

Figure 4: The selected text (left) "With the effort to treat and cultivate our meadows and pastures as pure grass cultures, disregarding their natural characteristics, the concept of weeds has found harmony in grassland farming." (Translated from: "Mit dem Bestreben, unsere Wiesen und Weiden unter Mißachtung ihrer natürlichen Eigenarten als reine Graskulturen zu behandeln und zu bewirtschaften, hat in der Grünlandwirtschaft der Begriff des Unkrautes Einklang gefunden.") is graphically displayed in the *AnnotationPanel* (right), which is extracted from the BIOfid Project. Through the pre-processing of TEXTIMAGER, already recognized named entities are directly linked and visualized with their identified knowledge resources. Furthermore, a query can be executed in the connected knowledge databases for each token, which is then linked. A correction of existing automatic recognition is also possible. In addition, all tokens connected by a relation in Wikidata are visualized with an arc and the corresponding relation name is displayed. In this example, the "subclass" relation between *meadow* and *pasture* is visualized.

one therefore speaks of so-called *Fingerprint*s. The *Fingerprint* specifies for each annotation which user executed it at which time (timestamp). In this way, annotations can be differentiated down to the token level in terms of authorship and time.

5. Inter-Annotator-Agreement

When creating annotated corpora for NLP tasks, the *annotation quality* is a key concern. This is especially important when multiple annotators are involved in the corpus creation. In order to control the quality of the annotated data, all documents should be annotated at least by two annotators and these annotations should then be compared. To this end, one can employ *inter-annotator agreement* (IAA) methods. In this section, we will present the IAA methods implemented for TEXTANNOTATOR.

As mentioned in Section 4, TEXTANNOTATOR creates a single view for each annotator on a given document. These *user views* are the base for our IAA implementation. We use the *DKPro Agreement* module (Meyer et al., 2014) to compute IAA scores. This module distinguishes two different data models for annotations: *coding studies* and *unitizing studies*. *Coding studies* contain annotations on a set of items with fixed boundaries, such as entire documents or tokens, whereas in *unitizing studies* annotations are created as spans over a continuum of units. Our implementation contains UIMA analysis engines for both kinds of studies, which allow for:

- the selection of views to be compared,
- filtering of non-human made annotations,
- the selection of any applicable agreement measure and
- the integration of the agreement scores into the CAS structure.

DKPro Agreement (Meyer et al., 2014) implements a total of ten different agreement measures. As the purpose of the IAA in our context is to enable reviewing the annotation process, we only select measures that can give a score for each category, as opposed to an overall score. Thus, five measures remain (see Table 2). Note that all but Cohen's κ (1960) support any number of annotators.

The annotation categories to be included in the agreement calculation can be selected interactively in TEXTANNOTATOR (see Figure 5). Programmatically, this is done by passing a set of qualified category class or super-class names from any given *TSD* to the analysis engines, for both white- and blacklisting. The scores can be computed on-demand or on-change and are displayed in the annotation view overview (see Figure 7). AUTHORITYMANAGER can be used to restrict access to the IAA results. The UIMA IAA module is available on GitHub[11].

[11] https://github.com/texttechnologylab/UIMA-Agreement

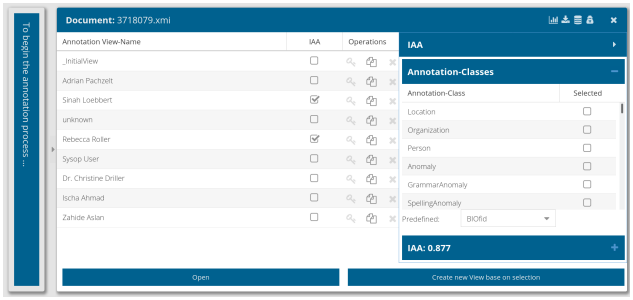

Figure 5: After opening a document in the TEXTANNOTATOR, the user is offered all available annotation views (left list). These annotation views are visualized in a three-column table: (1) Name of the annotation view, (2) IAA and (3) operations. While (1) is self-explanatory, (2) is a selection option for the later calculation of IAA values. The last column (3) allows duplication, removal and adjustment of user access permissions. The right side of the panel shows, for the later IAA calculation, the available annotation classes according to the *TSD* available in the TEXTANNOTATOR. After selecting the required views (left) and related classes (right), the IAA value of the document is automatically calculated in real time in the context of the selection. The calculated IAA value is displayed in the lower part of the right panel. In addition, to display and calculate the IAA value, suitable access permissions to the document must be available. This prevents annotators from viewing the current IAA value during annotations in order to align each other.

Agreement Measure	Type	Raters
Cohen's κ (1960)	coding	2
Percentage agreement	coding	≥ 2
Fleiss's κ (1971) [multi-π]	coding	≥ 2
Krippendorff's α (1980)	coding	≥ 2
Krippendorff's αu (1995)	unitizing	≥ 2

Table 2: The subset of DKPro inter-annotator agreement measures implemented for the TEXTANNOTATOR API.

6. Evaluation

To evaluate TEXTANNOTATOR, the *Usability Metric for User Experience* (UMUX) (Finstad, 2010) test was performed with 15 participants as part of a teaching course. During the development of TEXTANNOTATOR, this evaluation was the first to be executed and the task consisted of annotating two texts each with *TimeAnnotator* and *KnowledgeBaseLinker*. All participants had the same prior knowledge of the annotation subject and were equally unfamiliar with the subject matter. For the annotation corpus, texts from the BIOfid project were selected, whereby an IAA value was not calculated, since the focus was not on the agreement of the annotations itself, instead it was on the ubiquity. The following questions were answered after the annotation with a scale of 1-7, where 7 means complete consent:

1. The annotation tool's capabilities meet my requirements.
2. Using the annotation tool is a frustrating experience.

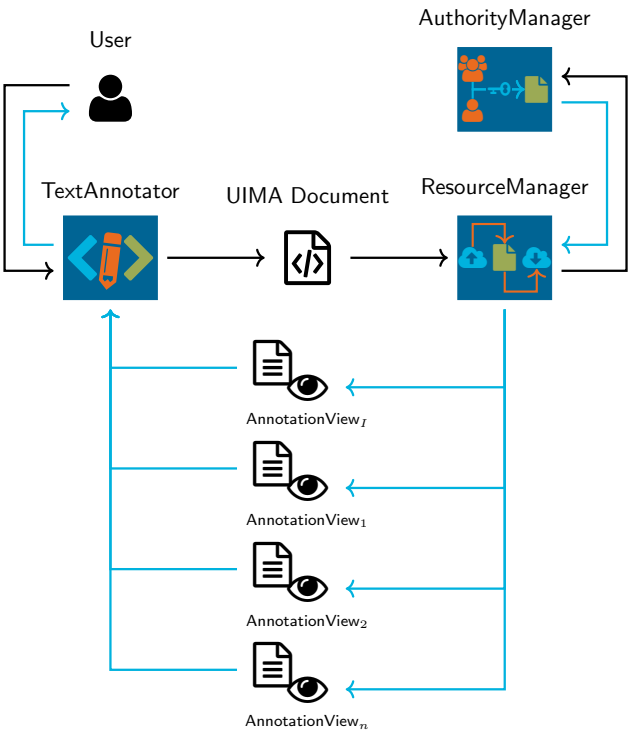

Figure 6: Simplified representation of the selecting process for *AnnotationView*s. After successful authentication of a user and the following list of all available documents, the user selects a document. The selected (UIMA) document is then browsed for *AnnotationView*s by RESOURCEMANAGER and its access permissions are validated w.r.t. the user. As a result, the available *AnnotationView*s are returned and presented so that the user can select the input view for one of the annotation tools (see Figure 2).

3. The annotation tool is easy to use.
4. I have to spend too much time correcting things with the annotation tool.

At first glance, the evaluation results do not look overwhelmingly good (Figure 8). But this is deceptive, especially if one looks at question one and considers the other questions in the context of the annotation task. The majority of the probands were able to perform the tasks with the annotation tool (Question 1). On the other hand, questions 2-4 suggest something out of the ordinary that made using TEXTANNOTATOR to be a frustrating experience: This can be traced back to the premise that the probands were supposed to complete the evaluation within a defined period of time. However, as the evaluation period increasingly came to an end without the necessary practice, the non-short texts could only be completed under a pressure factor. Regardless of how intuitively a tool is designed, if the underlying semantics (Stede, 2007) must be understood, as in the case of the *TimeAnnotator*, annotating is much more complex. The annotation with the KBL, on the other hand, was less time-consuming because only named entities had to be corrected in knowledge resources. However, since there was no distinction between the two tools in the study, no further conclusions can be drawn here. In summary, it can

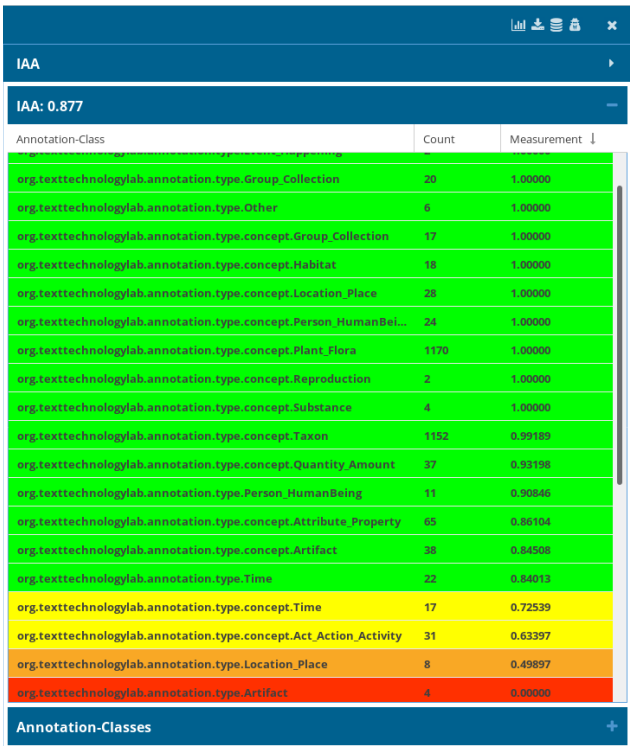

Figure 7: After selecting the views and annotation classes (see Figure 5) the IAA can be calculated and displayed as seen in this image. The first column shows the annotation class, the second how often it appears across all selected views and the third one holds the agreement score. In this particular case, the list is sorted by descending agreement score and each row is given a color depending on this score for a quick overview. Typically, a value of 0.8 (green) is considered reliable for most tasks (Krippendorff, 1980; Krippendorff, 2018).

be stated that the annotation task could be solved by the probands, but the usability – with regard to the evaluation environment – suffered to some degree. In addition, however, this shows that the development of an annotation tool in general should be more closely linked to the community and potential users, since usability and acceptance reflect the most important factor in the reuse of a tool. For this reason, these results will of course be incorporated into the next development milestones, which will be presented in the next section.

7. 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. For all 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*. In addition, TEXTANNOTATOR will implement a generic active learning unit (Fang et al., 2017) that supports the annotators. An important issue in this respect is the easy inclusion of annotation schemes: currently, new annotation schemes can only be applied by creating new *TSD*s, which requires a restart of

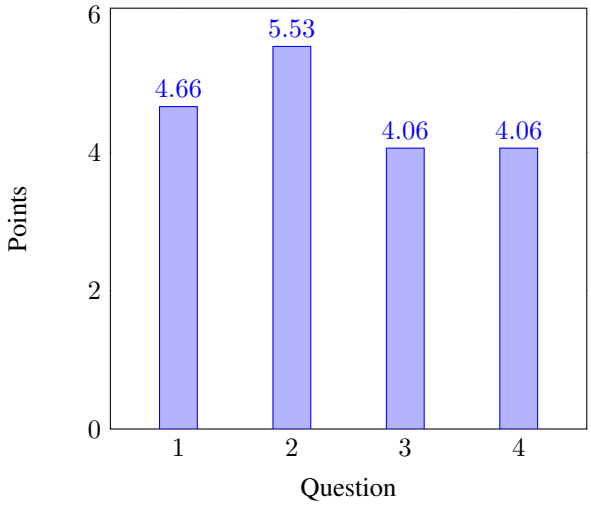

Figure 8: The results of the UMUX test with 15 participants.

the underlying database. To avoid this and ensure greater flexibility, it should be possible within TEXTANNOTATOR to use schemes based on individual or groups of users without defining them as individual *TSD*s. UIMA-based approaches (c.f. Verspoor et al. (2009), Roeder et al. (2010), Rak and Ananiadou (2013)) for this purpose should be evaluated, extended and applied as required.
For the integration of the community the current version of TEXTANNOTATOR will be available via GitHub[12]. TEXTANNOTATOR will be published under the *AGPL v3* license and can then be used and supplemented by the community, e.g. with their own visualizations. In addition, further evaluations are planned in various scenarios.

8. Conclusion

In this paper, we presented the current development state of TEXTANNOTATOR as a browser-based and collaborative text annotation tool that performs various linguistic annotation tasks within the same framework. By the multitude of annotation tools existing at the present time and referring to their functional heterogeneity, requirements have been defined which should be applied to annotation tools in terms of content and technology. Due to the state of the art (backend, frontent, external tools), the linking of multi-level annotations, the collaborative user interface and the connection to external knowledge resources, etc., we feel confident that TEXTANNOTATOR is currently one of the most advanced annotation tools.

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.

[12] `https://github.com/texttechnologylab`

References

Abrami, G. and Mehler, A. (2018). A uima database interface for managing nlp-related text annotations. In *Proceedings of the 11th edition of the Language Resources and Evaluation Conference, May 7 - 12*, LREC 2018, Miyazaki, Japan.

Abrami, G., Mehler, A., Helfrich, P., and Rieb, E. (2018). TextAnnotator: A browser-based framework for annotating textual data in digital humanities. In *Proceedings of the Digital Humanities Austria 2018*.

Abrami, G., Mehler, A., Lücking, A., Rieb, E., and Helfrich, P. (2019). TextAnnotator: A flexible framework for semantic annotations. In *Proceedings of the Fifteenth Joint ACL - ISO Workshop on Interoperable Semantic Annotation, (ISA-15)*, ISA-15, May.

Bethard, S., Ogren, P., and Becker, L. (2014). ClearTK 2.0: Design patterns for machine learning in UIMA. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3289–3293, Reykjavik, Iceland, May. European Language Resources Association (ELRA).

Brendler, S. (2004). Klassifikation der Namen. In Andrea Brendler et al., editors, *Namenarten und ihre Erforschung. Ein Lehrbuch für das Studium der Onomastik*, chapter 2, pages 69–92. Baar, Hamburg.

Cassidy, S. and Schmidt, T. (2017). Tools for multimodal annotation. In Nancy Ide et al., editors, *Handbook of Linguistic Annotation*, pages 209–227. Springer Netherlands, Dordrecht.

Chen, W.-T. and Styler, W. (2013). Anafora: A web-based general purpose annotation tool. In *Proceedings of the 2013 NAACL HLT Demonstration Session*, pages 14–19. Association for Computational Linguistics.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

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

Driller, C., Koch, M., Schmidt, M., Weiland, C., Hörnschemeyer, T., Hickler, T., Abrami, G., Ahmed, S., Gleim, R., Hemati, W., Uslu, T., Mehler, A., Pachzelt, A., Rexhepi, J., Risse, T., Schuster, J., Kasperek, G., and Hausinger, A. (2018). Workflow and current achievements of biofid, an information service mobilizing biodiversity data from literature sources. *Biodiversity Information Science and Standards*, 2.

Eckart de Castilho, R. and Gurevych, I. (2014). A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pages 1–11, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Eckart de Castilho, R., Mújdricza-Maydt, É., Yimam, S. M., Hartmann, S., Gurevych, I., Frank, A., and Biemann, C. (2016). A web-based tool for the integrated annotation of semantic and syntactic structures. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 76–84, Osaka, Japan, December. The COLING 2016 Organizing Committee.

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

Ferrucci, D., Lally, A., Verspoor, K., and Nyberg, E. (2009). Unstructured Information Management Architecture (UIMA) Version 1.0. OASIS Standard, Mar.

Finlayson, M. A. and Erjavec, T. (2017). Overview of annotation creation: Processes and tools. In Nancy Ide et al., editors, *Handbook of Linguistic Annotation*, pages 167–191. Springer Netherlands, Dordrecht.

Finstad, K. (2010). The usability metric for user experience. *Interacting with Computers*, 22:323–327, 09.

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Gleim, R., Mehler, A., and Ernst, A. (2012). Soa implementation of the ehumanities desktop. In *Proceedings of the Workshop on Service-oriented Architectures (SOAs) for the Humanities: Solutions and Impacts, Digital Humanities 2012, Hamburg, Germany*.

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

Helfrich, P., Rieb, E., Abrami, G., Lücking, A., and Mehler, A. (2018). Treeannotator: Versatile visual annotation of hierarchical text relations. In *Proceedings of the 11th edition of the Language Resources and Evaluation Conference, May 7 - 12*, LREC 2018, Miyazaki, Japan.

Hemati, W., Uslu, T., and Mehler, A. (2016). Textimager: a distributed uima-based system for nlp. In *Proceedings of the 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.

Kano, Y., Baumgartner Jr, W., Mccrohon, L., Ananiadou, S., Cohen, K., Hunter, L., and Tsujii, J. (2009). U-compare: Share and compare text mining tools with uima. *Bioinformatics (Oxford, England)*, 25:1997–8, 06.

Koch, M., Kasperek, G., HÃ¶rnschemeyer, T., Mehler, A., Weiland, C., and Hausinger, A. (2017). Setup of biofid, a new specialised information service for biodiversity research. *Biodiversity Information Science and Standards*, 1.

Krippendorff, K. (1980). *Content analysis: An introduction to its methodology*.

Krippendorff, K. (1995). On the reliability of unitizing continuous data. *Sociological Methodology*, pages 47–76.

Krippendorff, K. (2018). *Content analysis: An introduction to its methodology*. Sage publications.

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

Mehler, A., Abrami, G., Bruendel, S., Felder, L., Ostertag, T., and Spiekermann, C. (2017). Stolperwege: an app for a digital public history of the Holocaust. In *Proceedings of the 28th ACM Conference on Hypertext and Social Media*, HT '17, pages 319–320, New York, NY, USA. ACM.

Meyer, C. M., Mieskes, M., Stab, C., and Gurevych, I. (2014). DKPro agreement: An open-source Java library for measuring inter-rater agreement. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 105–109, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Moro, A., Raganato, A., and Navigli, R. (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.

Rak, R. and Ananiadou, S. (2013). Making UIMA truly interoperable with SPARQL. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 89–97, Sofia, Bulgaria, August. Association for Computational Linguistics.

Rak, R., Rowley, A., and Ananiadou, S. (2012). Collaborative development and evaluation of text-processing workflows in a UIMA-supported web-based workbench. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2971–2976, Istanbul, Turkey, May. European Language Resources Association (ELRA).

Rim, K. (2016). Mae2: Portable annotation tool for general natural language use. In *Proceedings of 12th Joint ACL-ISO Workshop on Interoperable Semantic Annotation*, pages 75–80.

Roeder, C., Jonquet, C., Shah, N. H., Baumgartner, William A., J., Verspoor, K., and Hunter, L. (2010). A UIMA wrapper for the NCBO annotator. *Bioinformatics*, 26(14):1800–1801.

Spiekermann, C., Abrami, G., and Mehler, A. (2018). VAnnotatoR: a gesture-driven annotation framework for linguistic and multimodal annotation. In *Proceedings of the Annotation, Recognition and Evaluation of Actions (AREA 2018) Workshop*, AREA.

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

Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., and Tsujii, J. (2012). Brat: A web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 102–107, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vasil'eva, N. (2011). Die Terminologie der Onomastik, ihre Koordinierung und lexikographische Darstellung. In Karlheinz Hengst et al., editors, *Namenkundliche Informationen*, volume 99/100, pages 31–45. Leipziger Universitätsverlag, Leipzig.

Verspoor, K., Baumgartner Jr, W., Roeder, C., and Hunter, L. (2009). Abstracting the types away from a uima type system. *From Form to Meaning: Processing Texts Automatically. C. Chiarcos, Eckhart de Castilho, Stede, M*.

Wu, S. T., Kaggal, V. C., Savova, G. K., Liu, H., Zheng, J., Chapman, W. W., Chute, C. G., and Dligach, D. (2011). Generality and reuse in a common type system for clinical natural language processing. In *Proceedings of the First International Workshop on Managing Interoperability and Complexity in Health Systems*, MIXHS '11, pages 27–34, New York, NY, USA. ACM.