TreeAnnotator: Versatile Visual Annotation of Hierarchical Text Relations

Philipp Helfrich, Elias Rieb, Giuseppe Abrami, Andy Lücking, Alexander Mehler
Goethe University Frankfurt, Text-Technology Lab
Robert-Mayer-Straße 10, 60325 Frankfurt am Main
helfrich@cs.uni-frankfurt.de, s0294036@stud.uni-frankfurt.de, {abrami, luecking, mehler}@em.uni-frankfurt.de

Abstract

We introduce TreeAnnotator, a graphical tool for annotating tree-like structures, in particular structures that jointly map dependency relations and inclusion hierarchies, as used by Rhetorical Structure Theory (RST). TreeAnnotator is browser-based, embedded within the UIMA framework and provides two visualization modes. TreeAnnotator's interoperability exceeds similar tools, providing a wider range of formats, while annotation work can be completed more quickly due to a revised input method for RST dependency relations. TreeAnnotator offers a multiple window view, which allows users to inspect several annotations side by side. For storing and versioning annotations, the UIMA Database Interface (UIMA DI) was developed to save documents based on a pre-defined type system. These features not only connect TreeAnnotator annotations to modern technological and dialog theoretical work, but set it apart from related tools. The ease of use of TreeAnnotator and its newly designed user interface is evaluated in a user study consisting of annotating rhetorical relations with TreeAnnotator and the classic *RSTTool*.

Keywords: text annotation, text visualization, multiple views, rhetorical structure theory, UIMA, database

1. Introduction

Over the past years, a lot of 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)).

For some reason, however, the domain of discourse annotation, most notably in terms of *Rhetorical Structure Theory* (RST) (Mann and Thompson, 1988), has not gained much attention for quite a long time. Most RST annotations, with the RST Discourse Treebank (Carlson et al., 2003)(Carlson et al., 2014) leading the way, are carried out with the original *RST Annotation Tool* (O'Donnell, 1997) or its extension, the *ISI RST Annotation Tool* (Marcu, 1999). While still usable, both tools, however, are not maintained any more and do not comply to current annotation frameworks. Due to such reasons, a third, browser-based RST annotation tool has been developed recently: *rstWeb* (Zeldes, 2016). Why, then, is there need for a new tool like TreeAnnotator?

We highlight three reasons in the following: Firstly, TreeAnnotator, unlike *rstWeb*, follows the UIMA framework, which supports current requirements of annotation pipelines (Wilcock, 2017). Note that also the Atomic/ANNIS frameworks (Druskat et al., 2014; Krause and Zeldes, 2016) are not based on UIMA, and hence TreeAnnotator is a module following a technologically complementary approach, more related to the "NLP-affine" text annotation tool *BRAT* (Stenetorp et al., 2012) (which in turn is the technological and design source for UIMA-based *WebAnno* (de Castilho et al., 2014)). Even extending such frameworks, we developed an UIMA database interface in order to provide a fine-grained type system-compliant storage of documents.

Secondly, users of TreeAnnotator benefit from different visualizations. The usefulness of visualizing annotations has been independently highlighted, for instance by (Finlayson and Erjavec, 2017, p. 183) and (Biemann et al., 2017, p. 236). In case of polynuclear hierarchies there are several ways to display the resulting structures, which in turn emphasize different structural features. Accordingly, TreeAnnotator provides – in addition to the classical RST view – an inclusion view that adopts rhetorical structure representations in the tradition of *Discourse Representation Theory* (DRT) (Kamp and Reyle, 1993), namely *Segmented Discourse Representation Theory* (SDRT) (Asher and Lascarides, 2003). The visualization flexibility not only provides alternative views on given rhetorical structures, it also makes TreeAnnotator's output more "readable" to modern theoretical linguistics. Thus, TreeAnnotator may reach a new group of users.

Thirdly, the newly designed graphical interface of TreeAnnotator significantly improves the tool's usability for rhetorical annotations: this has been evaluated in a study in comparison to the *RSTTool*.

The components, annotation processes, and visualizations of TreeAnnotator are described in Sec. 2. The usability study is reported in Sec. 3. Future developments are summarized in Sec. 4.

2. TreeAnnotator

2.1. Annotating Tree-like Text Structures in TreeAnnotator

TreeAnnotator is an up-to-date, browser-based and freely available device for discourse annotation. It is designed as a module of the still-developing annotation suite TextAnnotator. TextAnnotator in turn draws on the architecture of TextImager (Hemati et al., 2016), an UIMA-based framework that offers a wide range of NLP and visualization tools through a user-friendly GUI. All NLP tools that are available via TextImager can be utilized for preprocessing input documents, say, in terms of tokenization, lemmatization or pos tagging. TreeAnnotator's front-end is based on the *Ext JS* framework[1] and

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

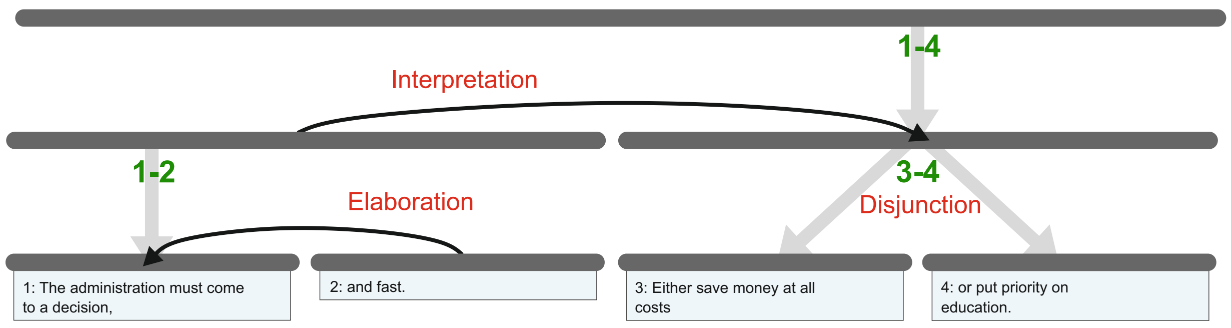
1-4
Interpretation
1-2
3-4
Elaboration
Disjunction
1: The administration must come to a decision,
2: and fast.
3: Either save money at all costs
4: or put priority on education.

Figure 1: TREEANNOTATOR's classical RST mode, illustrating the same RST structure as shown in Fig. 2

uses the *D3.js*[2] library for visualizations. It was designed for annotating RST trees (Mann and Thompson, 1988) and structures that jointly map dependency relations and inclusion hierarchies. As such, TREEANNOTATOR covers and extends the functionality of previous RST annotation tools, but is completely newly developed in order to meet current technological standards and requirements.
The user interface characteristics of TREEANNOTATOR give rise to the following features, which are elaborated subsequently:

- the number of mouse clicks for carrying out an RST annotation is significantly reduced, thanks to a new approach of how to annotate rhetorical structures. In particular, spans no longer have to be added manually, but are included automatically based on selected relation types – a respective user study is reported in Sec. 3.;
- the flexibility and interoperability exceeds comparable tools by offering new output formats (e.g. LaTeX and XMI; see Sec. 2.2. and Sec. 2.3., respectively);
- TREEANNOTATOR provides an alternative mode of interactive visualization (and, hence, annotation), including a split-window facility, which extends the classical RST view (Sec. 2.2.);
- finally, TREEANNOTATOR includes a redo/undo functionality due to an underlying versioning system (Sec. 2.4.).

2.2. Visualizing Annotations in TextAnnotator

TREEANNOTATOR's visualization addresses current workflow needs more specifically than comparable tools: In particular, (a) a second mode of visualization is available, independent from classical RST trees; (b) LaTeX output for scientific publications can be generated; (c) a multiple window view to inspect annotations side by side is provided.

(a) Visualization Modes Representing data by means of alternative modes helps in understanding and interpreting the subject matter. To this end, TREEANNOTATOR supports two modes of visualizing tree-like structures: *classical RST trees* and *dependency-oriented inclusion hierarchies* (DIH) (see Fig. 1 and 2). TREEANNOTATOR allows for switching between the two available visualizations even during annotation.
Visualizations in RST mode are similar to those of *(ISI) RSTTool* and *rstWeb*. However, TREEANNOTATOR's mouse wheel zooming and dragging functionality adds a useful feature, especially for annotating larger tree structures: TREEANNOTATOR has been tested with trees of more than one hundred *Elementary Discourse Units* (EDUs, the text spans connected by rhetorical relations); *RSTTool* and *rstWeb* in comparison are limited to displaying roughly two dozen EDUs simultaneously on screen.
Unlike *ANNIS*, which was developed for generic graph visualizations, TREEANNOTATOR offers two dedicated RST modes, specifically designed for quick annotation work. ANNIS' latest version, ANNIS3, provides a solution for RST trees by means of a visualization plugin (Krause and Zeldes, 2016). However, the application is limited to only displaying previously annotated corpora. The corresponding annotation tool *Atomic* (Druskat et al., 2014) is currently still in development and not available as a browser-based tool.

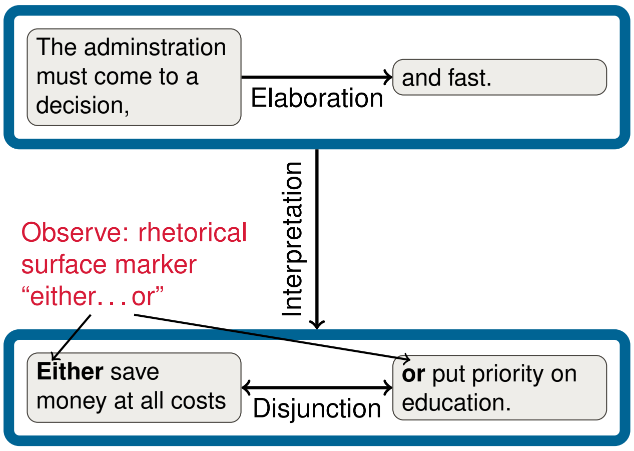
The adminstration must come to a decision,
Elaboration
and fast.
Interpretation
Observe: rhetorical surface marker "either...or"
Either save money at all costs
Disjunction
or put priority on education.

Figure 2: DIH: Mode to view integrated dependency and inclusion hierarchies in TREEANNOTATOR. The comment is added to the PGF/TikZ source code.

The DIH mode is not just a further means of visualization, it opens up a new domain of application, since it connects RST annotations to dynamic semantics in the DRT (Kamp and Reyle, 1993) tradition. The nested boxes representation format is familiar to formal semanticists from SDRT (Asher and Lascarides, 2003, e.g. p. 33), from where it is adopted in the first place. A particularly nice feature of the inclusion hierarchy representation – in contrast to the original RST rendering – is that the right frontier of a text becomes visible: the right frontier constitutes the structural

[2]https://d3js.org/

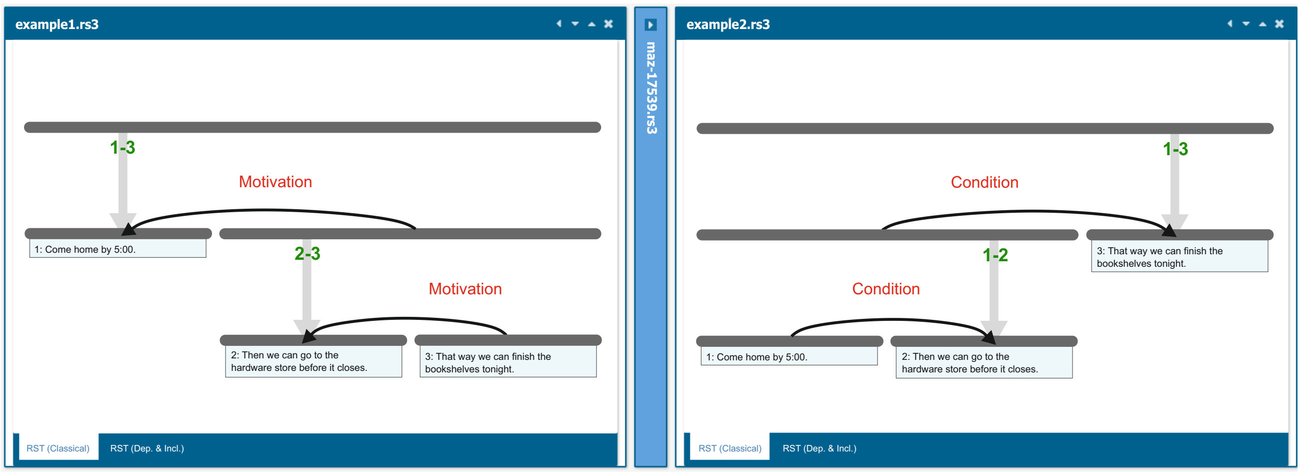
example1.rs3
1-3
Motivation
1: Come home by 5:00.
2-3
Motivation
2: Then we can go to the hardware store before it closes.
3: That way we can finish the bookshelves tonight.
RST (Classical)
RST (Dep. & Incl.)
maz-17539.rs3
example2.rs3
1-3
Condition
3: That way we can finish the bookshelves tonight.
1-2
Condition
1: Come home by 5:00.
2: Then we can go to the hardware store before it closes.
RST (Classical)
RST (Dep. & Incl.)

Figure 3: Multiple window view, showing two concurring annotations (Moore and Pollack, 1992, p. 542 f.). Several collapsible annotation views can be visualized.

boundary for anaphoric attachment sites and, hence, is of importance for any further co-reference annotation.

(b) Graphics Output Besides screenshot-like graphics (PNG), TREEANNOTATOR allows for generating LaTeX output: RST trees are exported using the *rst package* (Reitter, 2002), DIHs are output to PGF/TikZ (Tantau, 2015). PGF/TikZ are formal languages that can be interpreted by TeX (Knuth, 1984), LaTeX (Lamport, 1994) (including XeLaTeX), and ConTeXt (Hagen and Hoekwater, 2013) (including LuaTeX) to produce vector graphics output. The PGF/TikZ format is not only suitable for direct inclusion in LaTeX documents – the standard format of many scientific publication organs –, but enables further modifications within the graphic's source code. By this means, relevant features can be highlighted for publications – as illustrated in Fig. 2 –, or visualizations can be uncovered piecewise in presentations for talks.

(c) Multiple Window View A well-known problem of RST annotations is that there may be several, equally justified but concurring rhetorical analyses and structures of text spans (Moore and Pollack, 1992). Since this pluralism of rhetorical analyses is perfectly legitimate according to RST (Das et al., 2017), it has to be accounted for in the annotation workflow. Accordingly, a multiple window view was developed, which allows users to simultaneously inspect multiple annotations of the same text side by side, see Fig. 3. In the current version, this is achieved by simply creating a copy of the document. In future releases, the different annotations will be stored in the same document by using different *Subjects of Analysis*, that is, specific UIMA objects[3], which will further open up the possibility of having different annotations even for fragments of a text.

2.3. Storing Annotations in TextAnnotator

Since the Apache UIMA framework (Ferrucci et al., 2009) is one of the standards in projects within an NLP context (Wilcock, 2017), it is also used in the TEXTANNOTATOR suite. However, somewhat surprisingly this document-based schema does not provide native database support. To this end, we developed the *UIMA Database Interface* (UIMA DI) (Abrami and Mehler, 2018): it stores arbitrary UIMA documents based on their associated *UIMA Type System Descriptions* (UIMA TSD) in any database back-end (see Fig. 4). In the current version[4], the connections to MongoDB[5] and Neo4J[6] are already implemented and further database back-ends can be added easily.

For the general purpose of storing and using tree-like structures in UIMA frameworks, we developed a new UIMA TSD for graphs[7], compliant to the GraphML[8] format. Based on the UIMA TSD we defined an object model for RST trees, which is used in TEXTANNOTATOR (and thereby in TREEANNOTATOR). In order to efficiently store and retrieve annotations created by those tools, we employ established database technologies. Within this technological context, annotations are stored in the XMI[9] format during annotation processes. The resulting XMI document can be stored in any database integrated into UIMA DI or downloaded to a text file.

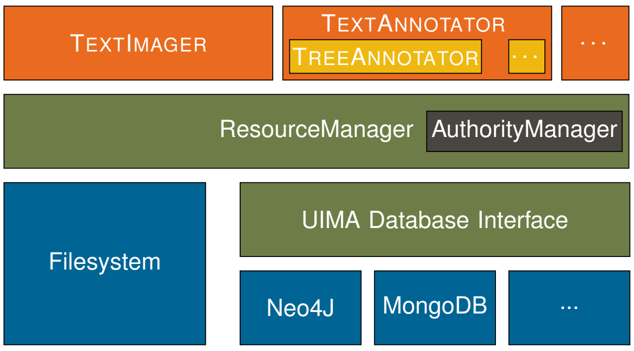

Figure 4: Components of the architecture

Table 1 summarizes the input/output formats which are cur-

[3]https://uima.apache.org

[4]Available on GitHub, under GPLv3.
[5]https://www.mongodb.com/
[6]https://neo4j.com
[7]http://www.textannotator.hucompute.org/typesystem/graph
[8]http://graphml.graphdrawing.org/index.html
[9]http://www.omg.org/spec/XMI/

Input	Data output	Graphical output
plain text, RST file format	UIMA-conform XMI file	LaTeX (PGF/TikZ), PNG

Table 1: Input/output formats of TREEANNOTATOR.

rently supported by TREEANNOTATOR.

2.4. Versioning Annotations in TextAnnotator

When dealing with error-prone, highly interpretive annotation tasks like RST analysis, undoing and redoing operations should be a fundamental part of the workflow (see e.g. the results in Sec. 3.2.). TEXTANNOTATOR currently supports an undo/redo functionality within the same session by versioning the XMI file in the internal memory. TREEANNOTATOR inherits TEXTANNOTATOR's versioning facility and thus provides the same functionality as *rstWeb* (v2.0) and *RSTTool* (v3.0), but exceeds tools such as *WebAnno* (v3.2.2), which does not offer an undo function. Thus, any annotation can be revised during the actual session. Since storing a copy of the complete file for every revision leads to a high memory usage, especially for larger documents, we are currently looking into genuine version control systems. Using such a system will not only lead to reduced memory usage, but will also allow TREEANNOTATOR to provide an undo/redo functionality even beyond sessions.

2.5. Managing Annotations in TextAnnotator

Since it cannot be guaranteed in any case that the natural language texts to be annotated are free of copyright restrictions, a rights and resource management is required. To this end, TEXTANNOTATOR uses components of the *eHumanities Desktop* (Gleim et al., 2012) which allows managing user rights and resources: *AuthorityManager* and *ResourceManager* (see Fig. 4). By integrating UIMA DI into *ResourceManager*, any UIMA document becomes accessible as a resource and organized in projects, which are defined as repositories. That is, user rights can be inherited from higher-level groups via restriction or extension in order to specify resource-specific permissions. Apart from that, TEXTANNOTATOR can also be used anonymously.
The need for a fine-grained rights management is evidenced by sample use-cases. A common classroom situation, for instance, involves students that work on an annotation task. The original texts are provided in a folder for which the group of students have only reading rights so that it is secured that the files cannot be modified or deleted by accident. Annotation results are stored in a second folder where students have write permission. The results of the other students cannot be changed or seen, as the students do not have reading rights to each other's documents, as required for inter-annotator evaluations.
In the context of research projects, files to be annotated are usually stored in a folder that is shared among the different project partners. All project partners have at least writing permissions and can therefore jointly work on the annotation sources. Now annotation results can be published in a targeted manner: they can be shared with individual users or made public to a wider audience simply by assigning appropriate authentications to files and users.
Tab. 2 compares TEXTANNOTATOR's rights and resource management, as facilitated by *AuthorityManager* and *ResourceManager*, to that of *rstWeb* and the widespread *WebAnno*: The access options applied by TEXTANNOTATOR are more elaborate than those of the compared tools. Especially, TEXTANNOTATOR acknowledges the level of groups, allows to share results with third parties and provides data access for external tools.

Feature	WebAnno	rstWeb	TextAnnotator
user management	✓	✓	✓
group management	✗	✗	✓
user permissions on projects / collections	✓	✓	✓
user permissions on documents	✗	✓	✓
group permissions on projects	✗	✗	✓
group permissions on documents	✗	✗	✓
project-independent document usage	✗	✗	✓
document organisation in repositories	✗	✗	✓
hierarchical repositories	✗	✗	✓

Table 2: Comparison of TEXTANNOTATOR, *WebAnno* and *rstWeb* with regards to rights and resource management

3. Evaluation

3.1. Setting

In a usability study, two groups of five test subjects each were asked to annotate pre-specified RST trees, comparing TEXTANNOTATOR to the most widely used RST annotation software, *RSTTool*. The subjects had no prior acquaintance with RST tools and RST annotations. However, they had a brief introduction into Rhetorical Rtructure Theory and its polynuclear relations. Twelve text segments (with three, five and seven RST relations, respectively) of the *Potsdam Commentary Corpus* (Stede, 2004) (Applied CL Discourse Research Lab, 2017) were handed out on paper. Group one annotated using TREEANNOTATOR, group two worked with *RSTTool*. The results are shown below.

3.2. Results

Using TREEANNOTATOR resulted in a decrease of mouse clicks per annotation. On average, annotators working with TREEANNOTATOR needed 14.9 clicks less compared to annotators using *RSTTool*; the median was improved by 18.0 clicks (or −32.7%, see Tab. 3).

	RSTTool		TreeAnnotator		improvement	
	avg	med.	avg	med.	avg %	med. %
short	36.8	36.5	22.4	19.0	−39.0%	−47.9%
medium	45.4	46.5	32.1	28.0	−29.3%	−39.8%
long	54.9	55.5	37.8	36.0	−31.2%	−35.1%
total	45.7	46.5	30.8	28.5	−32.7%	−38.7%

avg – average (arithmetic mean), med. – median

Table 3: Number of mouse clicks for 12 pre-specified RST annotations with *RSTTool* and TREEANNOTATOR

In terms of annotation time, TREEANNOTATOR again performed better in all measured categories. Using *RSTTool*, annotators needed 104.9 sec for the twelve texts on average, while user of TREEANNOTATOR managed an average time of 96.3 sec (an improvement of −8.6 sec or −8.2%; see Tab. 4).

	RSTTool		TreeAnnotator		improvement	
	avg	med.	avg	med.	avg %	med. %
short	84.3	78.0	70.8	58.0	−16.0%	−25.6%
medium	101.9	101.5	92.9	82.5	−8.8%	−18.7%
long	128.6	124.5	125.3	116.5	−2.5%	−6.4%
total	104.9	101.5	96.3	95.0	−8.2%	−6.4%

avg – average (arithmetic mean), med. – median

Table 4: Required time (in sec) for 12 pre-specified RST annotations with *RSTTool* and TREEANNOTATOR

The number of annotation errors for both tools is virtually the same (see Tab. 5). However, the types of errors differed: annotators working with *RSTTool* mainly erred with respect to RST tree structures, while annotators using TREEANNOTATOR mainly produced incorrect text segmentations. As a result, TREEANNOTATOR's text segmentation was subsequently altered to make the currently selected segment more obvious for users.

	RSTTool	TreeAnnotator
Incorrect relation name	3	1
Incorrect RST structure	6	2
Incorrect text segmentation	0	5
Output file missing entirely	1	1
total	10	9

Table 5: Number of errors for each annotation tool (60 pre-specified RST annotations per group in total)

Users of TREEANNOTATOR (on average) needed 28.5% of their time for segmentation, while the program's undo function was used very sparsely. One concern illustrated in Tab. 6 is the fact that the time per newly included relation increases with text length, a problem partly due to the hardware used during the study, which did not allow for smooth zooming and moving of larger RST trees.

text length	time for segmentation (% of total time)	Undo actions (per text)	Time per new EDU (in sec)	Time per new relation (in sec)
short	29.6%	0.30	4.4	5.1
medium	28.6%	0.35	4.1	8.2
long	27.3%	0.15	4.1	12.1
total	28.5%	0.27	4.2	8.5

Table 6: Average user behavior for pre-specified RST annotations in TREEANNOTATOR

3.3. Summary

TREEANNOTATOR provides a state of the art approach to rhetorical structure annotation. TREEANNOTATOR outperforms *RSTTool* in all categories shown above (albeit only minimally in terms of error-proneness). Improvements in the number of needed mouse actions is a particularly remarkable result. Furthermore, we found a reduced annotation time, especially for shorter to medium length texts.

4. Outlook

TREEANNOTATOR offers a modern and arguably intuitive approach to discourse annotations for RST-like structures, with unique features that set it apart from related tools.
Future versions will add support for annotations based on representations from DRT and SDRT and a newly designed module for semantic analysis, utilizing various lexical resources to create a comprehensive model of semantic representation. However, rhetorical structure annotations will not be overloaded by adding, say, semantic or anaphoric annotations. Rather, a multilayer annotation scenario will be implemented, which has already proven to be useful, for instance, in the GUM project (Zeldes, 2017). TEXTANNOTATOR's code will be made available on GitHub in the future. Meanwhile, the tool can be reviewed at http://www.textannotator.hucompute.org (login: textAnnotator, password: textAnnotator).

5. Bibliographical 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.

Asher, N. and Lascarides, A. (2003). *Logics of Conversation*. Cambridge University Press, Cambridge.

Biemann, C., Bontcheva, K., Eckart de Castilho, R., Gurevych, I., and Yimam, S. M. (2017). Collaborative web-based tools for multi-layer text annotation. In Nancy Ide et al., editors, *Handbook of Linguistic Annotation*, pages 229–256. Springer Netherlands, Dordrecht.

Carlson, L., Marcu, D., and Okurowski, M. E. (2003). Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In Jan van Kuppevelt et al., editors, *Current and New Directions in Discourse and Dialogue*, pages 85–112. Springer Netherlands, Dordrecht.

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.

Das, D., Taboada, M., and Stede, M. (2017). The good, the bad, and the disagreement: Complex ground truth in rhetorical structure analysis. In *Proceedings of the 6th Workshop Recent Advances in RST and Related Formalisms*, pages 11–19, Santiago de Compostela, Spain.

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

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

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.

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*.

Hagen, H. and Hoekwater, T., (2013). *ConTEXt reference manual*.

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.

Kamp, H. and Reyle, U. (1993). *From Discourse to Logic*. Kluwer Academic Publishers, Dordrecht.

Knuth, D. E. K. (1984). *The TEX Book*. Addison-Wesley.

Krause, T. and Zeldes, A. (2016). ANNIS3: A new architecture for generic corpus query and visualization. *Digital Scholarship in the Humanities*, 31(1):118–139.

Lamport, L. (1994). *LATEX – A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley.

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

Marcu, D., (1999). *Instructions for Installing the Rhetorical Annotation Tool*.

Moore, J. D. and Pollack, M. E. (1992). A problem for RST: The need for multi-level discourse analysis. *Computational Linguistics*, 18(4):537–544.

O'Donnell, M. (1997). RST-Tool: An RST analysis tool. In *Proceedings of the 6th European Workshop on Natural Language Generation*.

Reitter, D., (2002). *Rhetorical theory in LATEX with the rst package. Technical Manual*.

Stede, M. (2004). The Potsdam Commentary Corpus. In *Proceedings of the 2004 ACL Workshop on Discourse Annotation*. Association for Computational Linguistics.

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 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL 2012, pages 102–107, Avignon, France.

Tantau, T., (2015). *The TikZ and PGF Packages*, manual for version 3.0.1a edition.

Wilcock, G. (2017). The evolution of text annotation frameworks. In Nancy Ide et al., editors, *Handbook of Linguistic Annotation*, pages 193–207. Springer Netherlands, Dordrecht.

Zeldes, A. (2016). rstWeb – a browser-based annotation interface for rhetorical structure theory and discourse relations. In *Proceedings of NAACL-HLT 2016 System Demonstrations*, pages 1–5.

Zeldes, A. (2017). The GUM corpus: creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

6. Language Resource References

Applied CL Discourse Research Lab. (2017). *Potsdam Commentary Corpus*. 2.0, ISLRN 867-681-112-435-1.

Carlson, Lynn and Marcu, Daniel and Okurowski, Mary Ellen. (2014). *RST Discourse Treebank*. Linguistic Data Consortium, 1.0, ISLRN 299-735-991-930-2.