rstWeb – A Browser-based Annotation Interface for Rhetorical Structure Theory and Discourse Relations

Amir Zeldes
Department of Linguistics, Georgetown University
amir.zeldes@georgetown.edu

Abstract

This paper presents rstWeb, a new browser-based interface for Rhetorical Structure Theory and other discourse relation annotations. Expanding on previous tools for RST, rstWeb allows annotators to work online using only a browser. Project administrators can easily collect multiple annotations of the same documents on a central server, keep track of annotation processes and assign tasks and annotation schemes to users. A local version using an embedded web framework is also available, running offline on a desktop browser under the localhost.

1 Introduction

Since its introduction by Mann & Thompson (1988) Rhetorical Structure Theory has enjoyed continuing interest as a framework for the analysis of discourse relations, including the development of large scale corpora (especially the RST Discourse Treebank; RSTDT, Carlson et al. 2003) and automatic parsers (Joty et al. 2013, Surdeanu et al. 2015). However while the development of RST corpora and parsing has continued, there has been less progress in creating more up-to-date, collaborative and online interfaces for annotation, which would facilitate the development of new manually annotated data sets. Most work to date has used either the original RSTTool (O'Donnell 2000), a local desktop application written in Tcl/Tk, or its extension, the ISI RST Annotation Tool by Daniel Marcu (see: `http://www.isi.edu/~marcu/discourse/AnnotationSoftware.html`). Both tools are not being actively developed at present, and installing and running them across platforms can be challenging.

Meanwhile for other annotation tasks, online web interfaces have been developed which allow annotators to be trained and to work using only a browser, substantially facilitating the recruitment, curation and validation of data (e.g. Arborator, Gerdes 2013 for dependency syntax, or WebAnno, Yimam et al. 2013, for a variety of tasks). These server-based tools let project managers collect data centrally, without exchanging files with annotators, and track progress or log annotation processes automatically, while substantially reducing administration effort. The software presented here is meant to do the same for RST. Specifically it allows:

- Annotation using only a browser
- Import and export of RSTTool's .rs3 format
- Import of plain text (discourse unit per line)
- Support for multiple annotated versions of documents across users
- Enforcement of uniform annotation schemes across users
- Undo/redo functionality
- Logging of annotation steps
- Administration for user assignments, projects and guideline links
- Single mode for adding/deleting spans, multinuclear relations and satellite linking (no mode switching, see below)

The following section describes the technical infrastructure of rstWeb and the main requirements and workflows of the software. Section 3 briefly reports on a project employing rstWeb as an annota-

Proceedings of NAACL-HLT 2016 (Demonstrations), pages 1–5,
San Diego, California, June 12-17, 2016.

tion interface and estimates the reduction of user actions compared to previous tools based on annotation logs from RSTDT. Section 4 discusses some applications to discourse annotation outside RST. Section 5 ends with discussion for further work.

2 Software architecture

rstWeb[1] is written in Python with a SQLite backend, and these are required for the server running the software. In order to stay light-weight and responsive, JavaScript is used for the browser-based client, making the server-side demand almost no resources. jQuery and jsPlumb are used to render edges and animations. Following a static form-submit architecture (cf. Arborator, Gerdes 2013), no running services are used: Python scripts are exposed via a Web server (e.g. Apache), and calling them from a browser accesses the DB to serialize HTML for the client. For local machine use, a service script using the CherryPy framework can be used, requiring local users to install Python and CherryPy (`http://www.cherrypy.org/`). The software is platform independent, running on Mac, Linux and Windows platforms. Figure 1 gives a schematic overview of the system's architecture.

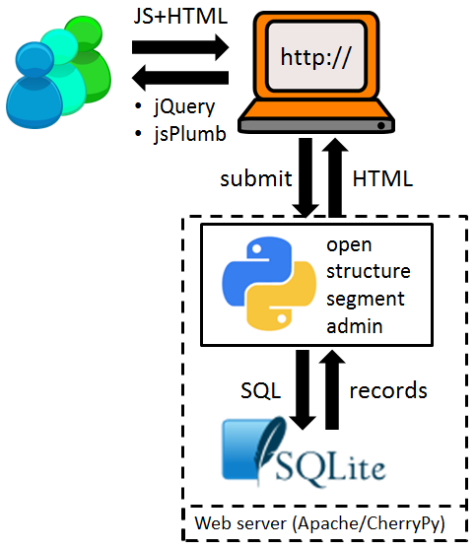

Figure 1: rstWeb schematic architecture.

Four scripts are exposed to the user, used to open and administrate projects ('open' and 'admin' scripts), and to annotate in two modes described below: 'segmentation' and 'structuring'.

[1] `http://corpling.uis.georgetown.edu/rstweb/info`

To annotate documents, users log in to the interface, where they can open any documents that have been assigned to them. Each user has their own copy of each assigned document, meaning that multiple users can annotate the same document in parallel for inter-annotator agreement experiments, though the tool does not support automatic calculation of agreement measures at present. Once a document has been opened, the user can move freely between two modes: segmentation of Elementary Discourse Units (EDUs), and structuring the units into an RST tree (see Figure 3 below).

In designing the annotation workflow, a central objective was to avoid constant switching between modes: in RSTTool, segmenting units, linking, unlinking, grouping them in spans or adding multinuclear relations, all required changing the 'mode' to do just that task; single clicks could then be used to carry out the action. This meant it was more convenient to complete multiple tasks of the same kind (e.g. spanning or unlinking) consecutively, which required some planning and reduced flexibility, or alternatively that frequent switching needed to be done. For rstWeb, the attempt was made to allow all operations on any node to be available simultaneously. This attempt has been successful for all tasks except for segmentation. An initial attempt to allow users to segment units within the RST diagram proved cumbersome, since reading EDUs in small boxes left-to-right is more difficult than reading the running text in one big box.

As a result, a dedicated segmentation mode was developed, the interface for which is shown in Figure 2. This interface closely resembles RSTTool's segmentation mode.

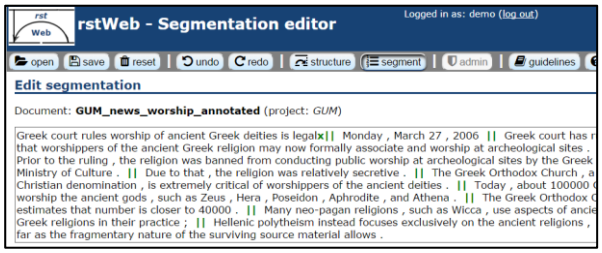

Figure 2: Discourse segmentation editor.

Users can move between modes and choose to re-segment while structuring: if a unit in a tree is segmented, the first portion of the divided segment retains the original function, and the second is created without attachment. Merging two units causes

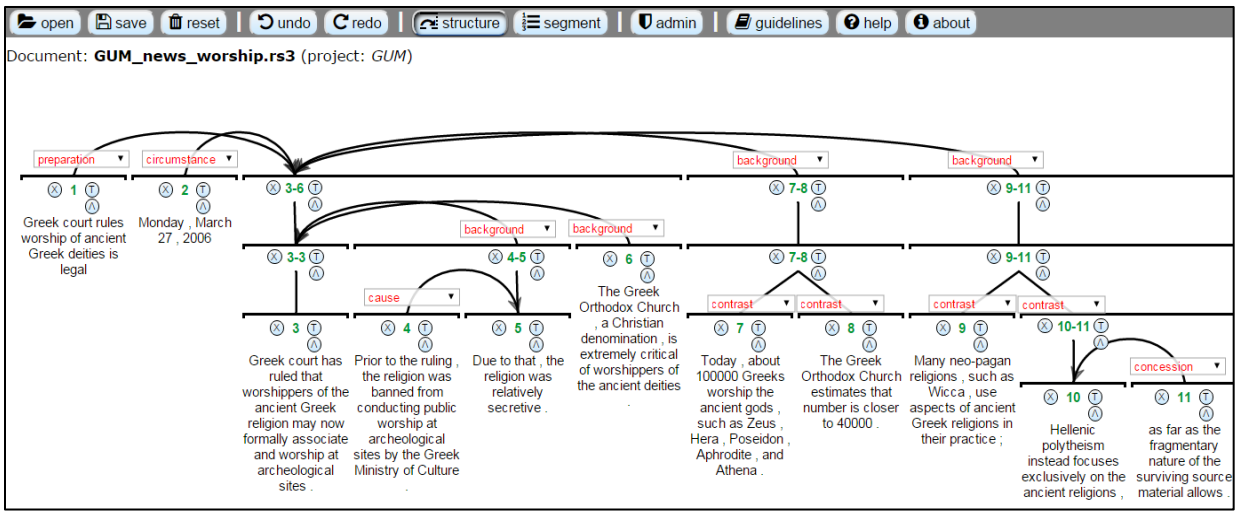

Figure 3: Structurer interface with an RST tree. The three buttons around each node allow users to unlink edges, create grouping spans or add multinuclear clusters above nodes, without switching annotation modes.

them to retain the attachment and label of the first unit. The tool has client-side undo/redo functionality, without submitting to the server, though undo/redo steps are logged as in the ISI tool.[2]

The other mode, structuring, is where the bulk of annotation work is done (see Figure 3). rstWeb supports the same tree structures as other tools, including crossing edges. However unlike earlier tools, there is no need to switch between annotation modes to connect or unlink nodes, add spans, or add multinuclear relations. These actions are handled by small buttons surrounding each node junction: X for unlinking, T for adding a span and Λ for multinuclear nodes (see Figure 3). User reports suggest that this facilitates annotation substantially.

Finally, administrators can manage user assignments and import documents from plain text files (one EDU per line) or .rs3 files (RSTTool format), or export annotations in .rs3 format.[3] Documents can be grouped into projects, which can be given a guidelines URL for users to consult.

3 Annotating in rstWeb

rstWeb has been employed in the annotation of the GUM corpus (Zeldes 2016)[4], an open-access multilayer corpus including RST analyses, constructed via classroom annotation and extended yearly. The corpus contains texts from 4 genres: travel guides, how-to guides, online news and interviews. In the most recent round of data collection, encompassing 29 documents, RST annotation was done with rstWeb, instead of the previously used RSTTool. Documents were comparable in length (Ø 58.31 EDUs) with those in the RST Discourse Treebank with Ø 56.59 EDUs (Carlson et al. 2003). This suggests that the system can be used successfully for text sizes on par with the benchmark resource for RST. The amount of errors based on instructor corrections using rstWeb compared to RSTTool was very similar (see Zeldes 2016).

To give an idea of the mode changes required by a multi-mode workflow, switching between linking/unlinking/grouping and creating multinuclear clusters as in older tools, we can examine annotation step files from the RST Discourse Treebanks. Table 1 gives the necessary mode change rates per node (including non-terminals), and the proportion of changes per annotation step in 10 random Wall Street Journal documents from RSTDT (including undo actions, but excluding segmentation operations).

Although the tools are different and therefore hard to compare directly, rstWeb logs from the GUM data suggest a similar rate of Ø 0.43 action type changes per step, indicating that annotators generally use mode changes as needed in either environment, meaning the multimode interface should save a substantial amount of clicking.

[2] Step logging has been used in the evaluation of annotation methodology, for example in Marcu et al. (1999).
[3] This format can also be imported into corpus search tools supporting RST, such as ANNIS (Krause & Zeldes 2016).
[4] `http://corpling.uis.georgetown.edu/gum`

doc	cha	steps	nodes	cha/stp	cha/node
wsj_0602	74	143	128	0.5174	0.5781
wsj_0654	16	30	37	0.5333	0.4324
wsj_0667	18	25	33	0.72	0.5454
wsj_1146	207	546	636	0.3791	0.3254
wsj_1169	15	30	34	0.5	0.4411
wsj_1306	32	72	93	0.4444	0.3440
wsj_1387	113	209	271	0.5406	0.4169
wsj_2336	25	45	61	0.5555	0.4098
wsj_2373	12	39	58	0.3076	0.2068
wsj_2386	55	177	255	0.3107	0.2156
Ø	56.7	131.6	160.6	0.4809	0.3916

Table 1: Mode change proportions per step and node in 10 WSJ documents from the RST Discourse Treebank.

During a previous round of data collection for GUM, RST annotations for the same corpus with the same text types were created using RSTTool. Feedback from students who switched from working with RSTTool to rstWeb, as well as from instructors (including a trained teaching assistant), has been very positive.

4 Using rstWeb for other resources

Data has successfully been imported into rstWeb from several existing RST-annotated sources, including the RST Discourse Treebank (converted to .rs3) and the German Potsdam Commentary Corpus (Stede & Neumann 2014). Although the software has been designed specifically for RST annotation, it may be possible to use it for other types of annotation, especially those representing binary relations between clauses. In particular, it is possible to disable the buttons generating spans and/or multinuclear nodes: this could be useful for other (shallow) discourse parsing frameworks or subsets of these, in which annotators would not be allowed to create multinuclear nodes or possibly any form of hierarchy.

For some forms of annotation, and particularly for explicit connectives (e.g. marking up a word such as 'because') and gaps inside clauses (clause parts with no relations), as used e.g. in the Penn Discourse Treebank (Prasad et al. 2008), the interface is not suitable, since each unit of annotation must be broken off as a segment. For connectives, this could be a single word, which would be impractical to view in the RST style diagram. However for simple binary relation classification between clauses with similar schemas, the advantages of the online, browser-based interface may make it a useful option (cf. Figure 4, using the *Expansion.Conjunction* and *Expansion.Restatement* relations from PDTB; multinuclear buttons have been disabled, but hierarchies are still enabled).

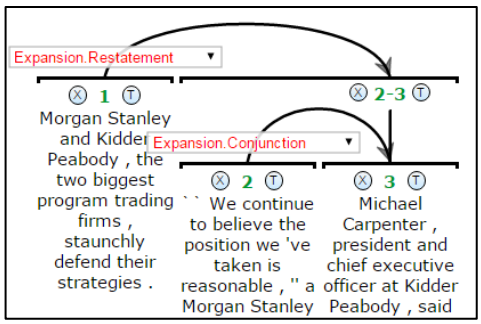

Figure 4: PDTB style hierarchical binary relations without connective annotation. Multinuclear buttons are disabled.

5 Conclusion

rstWeb offers a new, actively maintained tool for online, browser-based annotation of Rhetorical Structure Theory. The static script strategy of the backend means that server load when running rstWeb is negligible: it is not running at all unless a user has just submitted or requested data. Using CherryPy as a localhost container means that server code can be used offline or by single users who do not have access to a server – all code updates to the server version carry over to the local version. Using the browser as an interface means that users can work in a familiar environment, without installing software (at least for server based projects), that administrators do not need to exchange files with annotators, and that the system is cross-platform compatible without resorting to heavier Java based frameworks.

In future work, some additional features could be added to the software. In particular, it is currently not possible to edit the inventory of RST relations after the import of a document. Also, support for 'schemas', i.e. added span annotations to mark a unit as a 'title' etc., which was supported in previous tools, is not currently implemented, but is planned for an upcoming version. Finally, built in facilities for measuring inter-annotator agreement are interesting possible addition to the software.

References

Lynn Carlson, Daniel Marcu and Mary Ellen Okurowski. 2003. Building a Discourse-Tagged Corpus in the Framework of Rhetorical Structure Theory. In *Current and New Directions in Discourse and Dialogue*. (Text, Speech and Language Technology 22.) Kluwer, Dordrecht, 85–112.

Kim Gerdes. 2013. Collaborative Dependency Annotation. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*. Prague, 88–97.

Shafiq Joty, Giuseppe Carenini, Raymond Ng, and Yashar Mehdad. 2013. Combining Intra- and Multi-Sentential Rhetorical Parsing for Document-Level Discourse Analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, 486–496.

Thomas Krause and Amir Zeldes. 2016. ANNIS3: A New Architecture for Generic Corpus Query and Visualization. *Digital Scholarship in the Humanities* 31(1):118–139.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text* 8(3):243–281.

Daniel Marcu, Estibaliz Amorrortu, and Magdalena Romera. 1999. Experiments in Constructing a Corpus of Discourse Trees. In *Proceedings of the ACL Workshop Towards Standards and Tools for Discourse Tagging*. College Park, MD, 48–57.

Michael O'Donnell. 2000. RSTTool 2.4 - A Markup Tool for Rhetorical Structure Theory. In *Proceedings of the International Natural Language Generation Conference (INLG 2000)*. Mitzpe Ramon, Israel, 253–256.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse Treebank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*. Marrakech, Morocco.

Manfred Stede and Arne Neumann. 2014. Potsdam Commentary Corpus 2.0: Annotation for Discourse Research. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2014)*. Reykjavik, Iceland, 925–929.

Mihai Surdeanu, Thomas Hicks, and Marco A. Valenzuela-Escarcega. 2015. Two Practical Rhetorical Structure Theory Parsers. In *Proceedings of NAACL-HLT 2015*. Denver, CO, 1–5.

Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, 1–6.

Amir Zeldes. 2016. The GUM Corpus: Creating Multilayer Resources in the Classroom. *Language Resources and Evaluation*. Available online at http://dx.doi.org/10.1007/s10579-016-9343-x.