RST-Tace
A tool for automatic comparison and evaluation of RST trees

Shujun Wan
Humboldt University of Berlin
Berlin, Germany
shujun.wan@hu-berlin.de

Tino Kutschbach
Independent Researcher
Berlin, Germany
tino.kutschbach@mailbox.org

Anke Lüdeling
Humboldt University of Berlin
Berlin, Germany
anke.luedeling@hu-berlin.de

Manfred Stede
University of Potsdam
Potsdam, Germany
stede@uni-potsdam.de

Abstract

This paper presents RST-Tace, a tool for automatic comparison and evaluation of RST trees. RST-Tace serves as an implementation of Iruskieta's comparison method, which allows trees to be compared and evaluated without the influence of decisions at lower levels in a tree in terms of four factors: constituent, attachment point, nuclearity as well as relation. RST-Tace can be used regardless of the language or the size of rhetorical trees. This tool aims to measure the agreement between two annotators. The result is reflected by F-measure and inter-annotator agreement. Both the comparison table and the result of the evaluation can be obtained automatically.

1 Introduction

Rhetorical Structure Theory (Mann and Thompson, 1988) is intended to describe discourse structure and text organization by labeling the discourse relations that hold between *elementary discourse units (EDU)* or between larger spans of text. It is widely used throughout the discourse community as a theory for discourse analysis. RST is defined as the reconstruction of the author's plan from the perspective of the reader (Stede, 2017), that is to say it implies a certain subjectivity. According to this view, different annotators might very well produce different analyses, which can nonetheless be equally legitimate (Das et al., 2017).

However, differences in the analysis based on the legitimate scope of explication ought to be distinguished from unexpected errors or ambiguities resulting from unclear annotation guidelines. In order to assess and ensure the accuracy and reliability of the annotation, it is crucial to measure the agreement between the annotators. Compared with other types of annotation, evaluating rhetorical structures and calculating the inter-annotator agreement are not trivial. There are several challenges: 1) RST tree parsing, 2) finding an appropriate method for comparison and evaluation, 3) applying this method efficiently.

So far, Marcu's (2000) method for the comparison of RST annotations by several annotators is widely-used. Building on Marcu's method, Maziero and Pardo (2009) developed *RSTeval* in order to obtain the results of comparison automatically. While being widely used, the method has also been criticized. For instance, da Cunha and Iruskieta (2010) argue that it amalgamates agreement coming from different sources, with the result that decisions at lower levels in the tree significantly affect agreement at the upper rhetorical relations in a tree (Iruskieta et al., 2015), and relations cannot be able to be compared where constituents do not coincide.

In this regard, Iruskieta et al. (2015) proposed an evaluation method which accepts that constituents do not need to coincide in their entirety to be compared. Iruskieta's method provides a qualitative description of dispersion annotation while allowing quantitative evaluation (details are introduced in section 2). Nevertheless, using this method to evaluate discourse structures manually is an extremely time- and resource-consuming task. Thus, inspired by RSTeval, we have developed RST-Tace as a tool for automatic comparison and evaluation of RST trees based on Iruskieta's method.

Proceedings of Discourse Relation Parsing and Treebanking (DISRPT2019), pages 88–96
Minneapolis, MN, June 6, 2019.

Example	CS1	CS2
[1]	1	1
[2]	23	23|24
[3]	17-18|26	17-18|26
[4]	10|15	11|15

Table 1: Examples of matching central subconstiutents (extracted from *CMN_008, RST German Learner Treebank*[8])

This research paper focuses on the theoretical foundations of RST-Tace as well as the implementation process. In addition, an example of using RST-Tace to compare and evaluate rhetorical trees (extracted from a self-built RST treebank) by two linguists will be presented in the final section.

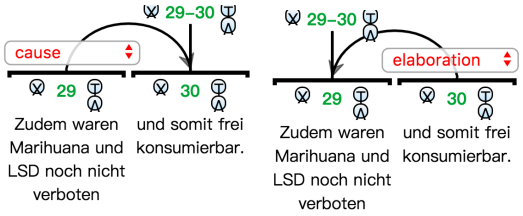
2 Theoretical Framework

According to Iruskieta's method, the correspondence of constituents is not a necessary condition for comparison. Only the central subconstituent (CS)[1] which indicates the most important unit of the satellite span, has to be identical. With this restriction, discourse structures are compared using four independent factors:

- Constituent (C): the unit(s) where the satellite (or one of the nuclei in case of multinuclear relations) is located.
- Attachment point (A): the unit(s) where the constituent is linked.
- Nuclearity (N): the direction of the relation.
- Relation (R): the name of relations.

In order to use Iruskieta's method, each RST tree must first be converted into a table which consists of the four above factors as well as the central subconstituent. Subsequently, pairs should be matched according to the central subconstituent. The third stage is evaluation. According to Iruskieta's method, both agreement and disagreement are considered. Lastly, the result of the evaluation (F-measure) is calculated.

[1]According to Iruskieta (2015), there is an agreement that the most important unit of an RST tree is the "central unit(s)" (Stede, 2008) and the most important unit of a span is the "central subconstituent" (Egg and Redeker, 2010). Following this framework, Iruskieta et al. use the term "Central Subconstituent(s)" of a relation for the most important unit of the modifier span that is the most important unit of the satellite span.



(a) Annotator I (b) Annotator II

Figure 1: Annotations from two annotators (extracted from *DEU_006, RST German Learner Treebank*[2])

Anno	CS	R	N	C	A
I	29	cause	→	29S[3]	30N[4]
II	30	elaboration	←	30S	29N

Table 2: Matching table of Figure 1

Modifications

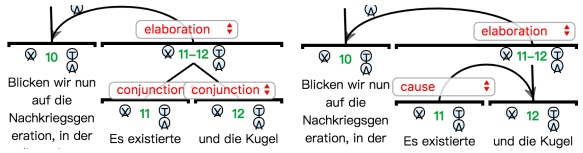
In light of the basic principles of Iruskieta's method, we highlight, in this part, some points which are crucial for the use of our tool or are slightly modified by us:

1. The use of this method for comparison and evaluation takes the harmonization of discourse segmentation as a given.
2. As a general rule, CS has to be the same so that the relations are able to be compared (see example [1] in Table 1). A case complicating the comparison occurs when multinuclear relations become involved. When there is a multinuclear relation, all of its constituents must be described as CS. Consequently, a multinuclear relation has more than one CS. Under such a circumstance, when a relation with more than one CS is able to be compared with another that has only one CS, at least one of the CSs has to be identical (see example [2] in Table 1). When two multinuclear relations are to be compared, their CSs do not have to remain the same entirely (see example [3] in Table 1). Similarly, they need to possess at least one identical CS (see example [4] in Table 1).
3. The association of CS is the prerequisite for

[2]Since we aim to show the tree structures rather than the linguistic decision, we decided not to translate the language of RST trees into English in this paper.

[3]S represents Satellite

[4]N refers to Nucleus



(a) Annotator I (b) Annotator II

Figure 2: Annotations from two annotators (Extracted from *DEU_006, RST German Learner Treebank*)

Anno	CS	R	N	C	A
I	11-12	elaboration	←	11-12S	10N
II	12	elaboration	←	11-12S	10N

Table 3: Matching table of Figure 2

comparison of relations according to Iruskieta's method. However, there are two cases that still deserve to be compared, even though the CSs are not identical.

The RST tree of the example in Figure 1(a) is quite similar to the RST tree of 1(b). Apart from the names of relations which are coincidentally not the same in this example, the main reason why the two trees have different CSs, constituents and attachment points, is that they differ merely in nuclearity. However, due to the discrepancy in CS, this pair cannot be detected using Iruskieta's method. From Table 2 which is converted from the RST trees, we can observe that the CS of Figure 1(a) is 29 whereas the CS of Figure 1(b) is 30; yet, C1 equals A2 as well as C2 equaling A1.

The other case in which relations are still associated despite distinct CSs is when C1 equals C2 and A1 equals A2. The relation 10 and 11-12 of Figure 2(b) is not able to be compared with the one of Figure 2(a), because they have different CSs (see Table 3). This discrepancy stems from the micro level, i.e. the relation between EDU 10 to EDU 11.

In brief, we match relations following the decision tree below (see Figure 3).

4. Iruskieta's method is originally designed for comparing and evaluating discourse structures in different languages and/or by different annotators. Hence, in the case of disagreement, two sources of disagreement are distinguished: type A and type L for

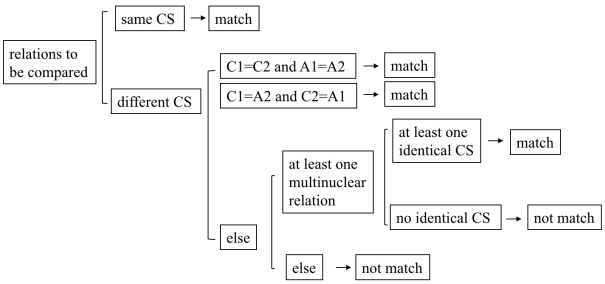


Figure 3: Decision tree for matching

annotator-based discrepancies and language-based discrepancies respectively[5]. In comparison to Iruskieta's method, we focus more on agreement instead of disagreement, because we aim to compare and evaluate annotations which are in the same language but annotated by different annotators. To check agreement in rhetorical relation, the constituent of this relation must have the same central subconstituent. If this condition is fulfilled, relation (R), constituent (C) and attachment point (A) will be further checked.

5. Concerning the results of evaluation, both F-measure and inter-annotator agreement for each factor are calculated. The agreement score for the whole RST trees is the mean value of the four factors.

3 Implementation

For evaluation, comparison, and analysis of RST trees, RST-Tace supports three main use cases:

1. Analysis of a single RST tree (i.e. extraction and listing of all annotated relations), see section 3.3.

2. Comparison of the annotated relations of a pair of RST trees, see section 3.4.

3. Statistical evaluation of a whole dataset.

These use cases depend on each other, e.g. in order to compare the annotated relations of a pair of RST trees (second use case), the annotated relations of both trees are first extracted and listed (first use case).

[5]For more details and examples regarding the evaluation of disagreement, refer to (Iruskieta et al., 2015), p.15-20.

3.1 Commandline Interface

In its current state, RST-Tace can be used via a commandline interface (command: *rsttace*), and each of the main use cases has its own command (*analyse* / *compare* / *evaluate*):

```
$ rsttace
Usage: rsttace [OPTIONS] COMMAND [ARGS]...

This tool analyses, compares, and evaluates RST trees.

Options:
--help  Show this message and exit.

Commands:
analyse   Parse single RST trees, analyse
          and list their annotated relations.
compare   Parse RST tree pairs and compare
          them with each other.
evaluate  Perform a statistical evaluation
          of a set of RST tree pairs.
```

The results are either written back to the commandline, or to csv-files if needed. For example, the command for comparing two rs3-files and writing the result to a CSV-file is:

```
$ rsttace compare file1.rs3 file2.rs3 \
                -o result.csv
```

As the tool is currently under active development the user interface may still be subject to change. For up-to-date information and complete documentation, please refer to the GitHub repository of *RST-Tace*.[6]

Before we discuss how each of the three tasks is performed in detail, the parsing process of rs3-files will be described.

3.2 RST Tree Parsing

After a set of texts has been annotated with tools such as *RSTTool* (O'Donnell, 1997) and *rstWeb* (Zeldes, 2016), the resulting RST trees are typically exported as **.rs3*-files. In order to work with the RST trees efficiently, these files have to be parsed and converted into an internal tree based data structure, which allows convenient data access for the desired evaluation task.

File Format

RSTTool and *rstWeb* both use the same file format for exported rs3-files with only minor differences. The file parser of *RST-Tace* is designed to handle the rs3-files of both tools.

The file format is based on XML and contains a header and a body. Figure 4 shows an example of a RST tree annotated with *RSTTool*. The header defines which relations are used in the RST tree and

```
<rst>
<header>
 <relations>
  <rel name="background" type="rst"/>
  <rel name="antithesis" type="rst"/>
  ...
  <rel name="reason" type="rst"/>
  <rel name="summary" type="rst"/>
  ...
  <rel name="sequence" type="multinuc"/>
  <rel name="joint" type="multinuc"/>
  <rel name="contrast" type="multinuc"/>
  <rel name="list" type="multinuc"/>
 </relations>
</header>
<body>
 <segment id="1" parent="45" relname="span">Man muss vor ...
 <segment id="2" parent="1" relname="reason">da die erst ...
 <segment id="4" parent="5" relname="background">Nehmen wir.
 ...
 <segment id="8" parent="7" relname="condition">wenn sie ...
 <segment id="46" parent="68" relname="span">Wenn man es ...
 <segment id="9" parent="69" relname="contrast">Die Kriegs..
 <segment id="10" parent="11" relname="span">Blicken wir ...
 <segment id="13" parent="51" relname="conjunction">Es ex...
 <segment id="12" parent="51" relname="conjunction">und die.
 ...
 <group id="3" type="span" parent="49" relname="contrast"/>
 <group id="47" type="span" parent="6" relname="evidence"/>
 <group id="48" type="span" parent="49" relname="contrast"/>
 ...
 <group id="75" type="span" parent="76" relname="span"/>
 <group id="76" type="span"/>
 ...
</body>
</rst>
```

Figure 4: Example of an rs3-file, encoding an RST tree

their corresponding type, i.e. either mono- or multinuclear. The body represents the actual RST tree. An rs3-file consists of a list of XML elements (either *segments* and *groups*), each of which has the following attributes: *Node ID*, *parent ID*, *relation name*. Elements of the type *segment* correspond to the EDUs and contain the corresponding text of the EDU. *Group* elements have an additional attribute *type*, encoding whether the element is a span or corresponds to a multinuclear relation.

As shown in Figures 1(a), 1(b), and 2(a), 2(b): each leaf node of an RST tree has an EDU-ID, and higher level nodes have ranges of EDU-IDs which depend on the EDU-IDs of their child nodes.

In the XML file format, the *segment* elements correspond to the EDUs; their order inside the body corresponds the order of occurence of the EDUs in the original text. For example, the first *segment* element in the body corresponds to the first EDU (i.e. having the EDU-ID "1"), the third *segment* element in the body corresponds to the third EDU (i.e. having the EDU-ID "3"), and so on.

For the case of rs3-files from *RSTTool*, it is important to note that the IDs encoded in the *segment* elements do not necessariliy correspond to the desired EDU-IDs, as shown in Figure 4[7].

[6] https://github.com/tkutschbach/RST-Tace

[7] In the example in Figure 4, the third *segment* has the ID

Because the proposed evaluation method of RST trees relies on correct EDU-IDs, the parser of *RST-Tace* infers the EDU-ID of each segment itself based on its position inside the list. The IDs of the *segment* and *group* elements are then used to reconstruct the tree structure. Afterwards, the ranges of EDU-IDs that higher level nodes span (i.e. represented by the *group* elements) can be inferred.

Internal Data Representation

As mentioned above, the data arrangement of rs3-files does not precisely represent an RST tree structure, which would allow convenient data access for the proposed evaluation method. As a remedy, the given data is parsed and converted into an internal tree representation, which consists of nodes and three different types of edges.

The three different types of edges are:

1. Horizontal edges, connecting two nodes on the same level: Encoding mononuclear relations.

2. Vertical edges, connecting one parent node with multiple child nodes one level below: Encoding multinuclear relations.

3. Vertical edges, connecting one parent node with one child node one level below: Encoding spans.

The tree nodes correspond to the *segment* and *group* elements in the rs3-file format. Each node is connected to a parent node via an edge and can have one horizontal edge connecting it to another node on the same level. Furthermore, it can have one vertical edge, connecting it to one or several child nodes on the next lower level. Additionally, the nodes encode their corresponding EDU ID or ID-range. If a node is a leaf node, then it also contains the text information of its corresponding EDU.

Parsing Process

The encoding step is implemented straightforwardly by first reading all XML elements and searching the element which corresponds to the root node of the RST tree (which is characterized by a missing parent ID, e.g. the element with $id = 76$ in Figure 4). Afterwards, for all elements that refer to this root as parent, nodes and edges are generated and connected depending on the type of their relation. In the next step, all elements that refer to those nodes are processed respectively. This process is repeated until all elements have been appended as nodes to the internal data structure.

Finally, after the complete tree has been built, the EDU ID-ranges of the higher level nodes have to be inferred based on their corresponding lower level children, because only the EDU IDs of the leaf nodes can be directly extracted from the rs3-file. This inference is done by iterating over the whole tree bottom-up, i.e. from the leaf nodes towards the root node, and gradually augmenting the higher nodes level-by-level until the root node is reached.

3.3 Extraction of Annotated Relations

In order to compare and evaluate an RST tree using Iruskieta's method, its annotated relations have to be extracted and listed together with additional information (e.g. *constituent*, *nuclearity*).

Once the RST tree is available in the form of the previously described data structure, the extraction of relations becomes a simple task of iterating over the set of edges in the tree and listing their corresponding relations. Also, the additional information is directly accessible: The *nuclearity* corresponds to the direction of an edge, and the other information such as *constituent*, *attachment-point* and *central sub-constituent* can be directly acquired from EDU IDs and ID-ranges encoded in the nodes that each edge is connected to.

3.4 Comparison of RST Tree Pairs

An important task in RST research is comparing different RST annotations (of different annotators) for a single text or a set of texts. Under the condition that two annotations of a text are based on the same segmentation of EDUs, RST-Tace can compare the two different RST trees and calculate an equivalence score.

In order to compare the RST annotations of two trees using Iruskieta's method, the annotation lists of both RST trees are generated first, as described in section 3.3. Afterwards, the annotated relations of both RST trees have to be associated to each other.

As mentioned above, two different annotators might create RST trees with different structures for the same text; thus, it is not always clear which annotated relation in one tree corresponds

"4" instead of "3", and the on the *segment* with ID "8" follows the *segment* with ID "46", which itself is followed by the *segment* with ID "9".

Equivalence	Cost
Same CS	0
$C1 = C2$ and $A1 = A2$	1
$C1 = A2$ and $C2 = A1$	2
At least one identical CS	3
No matching	4

Table 4: Cost values used for matching annotated relations of two RST trees

to which one in the other. This ambiguity means that the association is not a trivial task.

Optimal Association

RST-Tace deals with this ambiguity by searching for an optimal association. For this, each annotated relation of the first RST tree is compared to each annotated relation of the second RST tree by the scheme introduced previously and shown in Figure 3. Because each of the possible matching outcomes stands for a different degree of equivalence, they can be prioritized by assigning cost values to each of them (low cost values for high priorities, high cost values for low priorities). The cost values used in this work are shown in Table 4.

While comparing all annotated relations of both RST trees with each other, these cost values are used to populate a cost matrix $\mathbf{C}$. With N being the number of relations annotated in the first RST tree and M being the number of relations annotated in the second tree, the matrix $\mathbf{C}$ has the form $N \times M$. An element $\mathbf{C}_{i,j}$ represents the cost of matching relation i in the first tree with relation j of the second tree.

The optimal association is then calculated by applying the Hungarian algorithm (Kuhn, 1955), also known as Kuhn-Munkres algorithm, to this cost matrix. Matches are categorized as completely identical CS, C1=C2 and A1=A2, C1=A2 and C2=A1, partially identical CS as well as no matching.

Evaluation and Results

After the annotations of both RST trees have been associated, all annotation pairs are compared according to Iruskieta's method, i.e. their nuclearities (N), relations (R), constituents (C), and attachment points (A) are compared and marked as equal or non-equal. These values are then used to calculate F-measure and inter-annotator agreement.

RST-Tace also offers the possiblity to process a whole batch of RST tree pairs and calculate the equivalence scores and inter-annotator agreement over a whole dataset.

4 An Example of Comparison and Evaluation using RST-Tace

In this section, we provide an example of using RST-Tace to compare and evaluate RST trees. Extracted from *RST German Learner Treebank*[8], two annotations [9] on the same German text by two linguists are compared and evaluated. A part of the two RST trees where the annotations are different is shown in Figure 5; the comparison table and the results of evaluation are presented in Figure 6.

5 Summary

To conclude, RST-Tace allows comparison and evaluation of RST trees by different annotators automatically. It can be used for rhetorical structures in any language as well as with any size. The modifications that are made based on Iruskieta's method provide a further perspective of RST related theories. Currently, the statistical part of the implementation, i.e. the automatic calculation of F-measure and inter-annotator agreement, is under active development. In the future, additional features could be added to RST-Tace, for instance, a user-friendly interface, or a more sophisticated statistical analysis of larger datasets and RST treebanks.

Acknowledgments

This work was financially supported by the *China Scholarship Council*. We wish to express our appreciation to Felix Golcher from IDSL, Humboldt University of Berlin, who provided his statistical expertise that greatly assisted the research, and to Shuyuan Cao from Polytechnic University of Catalonia for his theoretical support. We would like to thank Matthew Plews and Dawn Nichols for proofreading this paper. We also owe our special thanks to the anonymous reviewers for their valuable comments.

[8] RST German Learner Treebank consists of 40 RST trees. The texts in the treebank are argumentative essays (around 25,000 tokens in total) extracted from Kobalt-DaF Corpus (Zinsmeister et al., 2012). All the data was annotated by two professional linguists according to the guidelines of the Potsdam Commentary Corpus (Stede, 2016), which was designed for German pro-contra essays. The quantity of EDUs of each text is between 40-80.

[9] Segmentation has been harmonized before annotating.

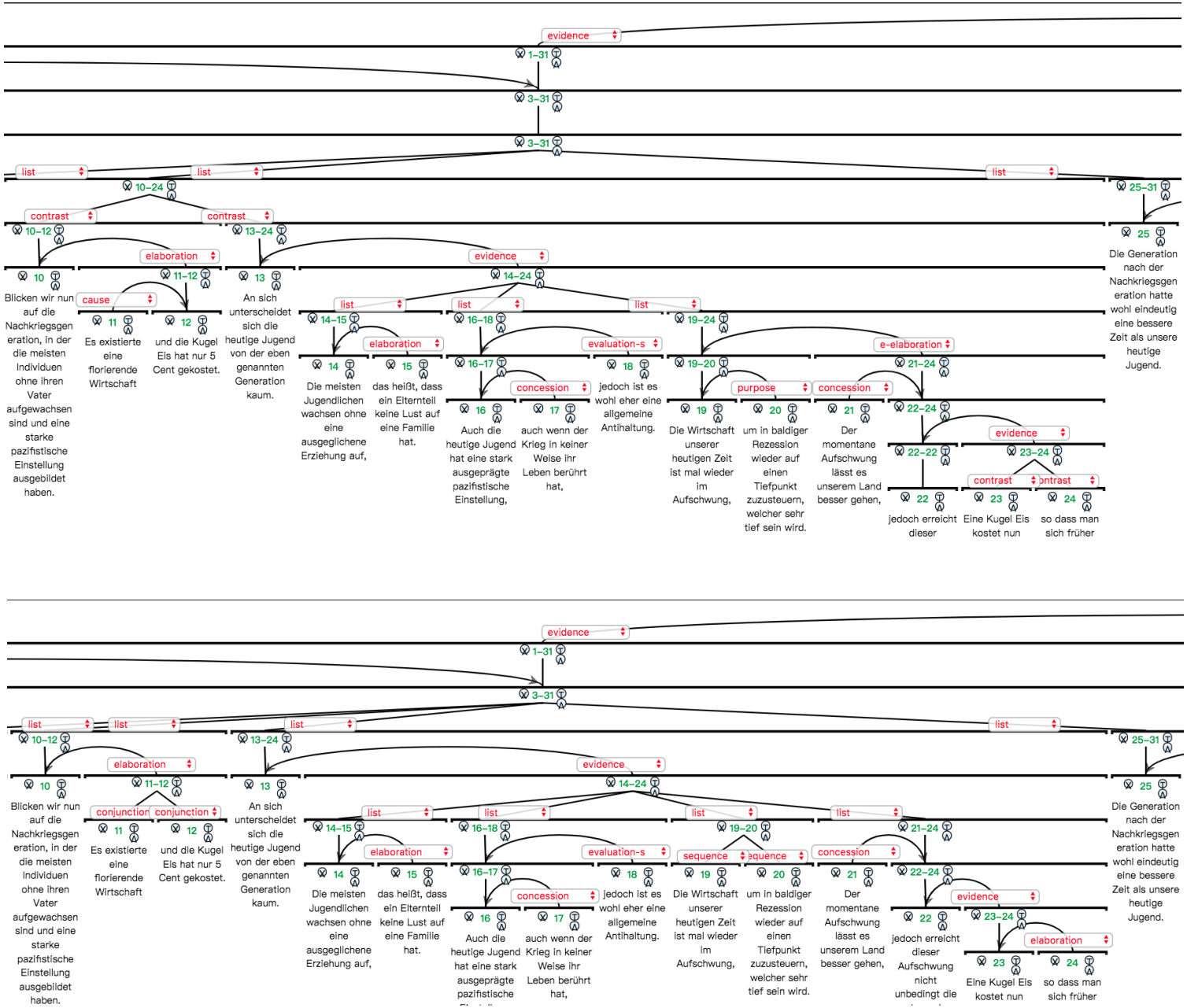


Figure 5: RST trees from two annotators. *DEU_006, RST German Learner Treebank*

ID	CS-A	Relation-A	Nuc-A	C1-A	C2-A	CN-A	A1-A	A2-A	AN-A	CS-B	Relation-B	Nuc-B	C1-B	C2-B	CN-B	A1-B	A2-B	AN-B	Matching	N	R	C	A
1	1	preparation	→	1	2	S	3	31	N	1	preparation	→	1	2	S	3	31	N	Completely identical CS	✓	✓	✓	✓
2	3-31	evidence	→	1	31	S	32	37	N	3-31	evidence	→	1	31	S	32	39	N	Completely identical CS	✓	✓	✓	×
3	32-37|38-41	joint	↔	1	37	N	38	41	N										No matching	×	×	×	×
4										40-41	evaluation-s	←	40	41	S	1	39	N	No matching	×	×	×	×
5	2	reason	←	2	2	S	1	1	N	2	reason	←	2	2	S	1	1	N	Completely identical CS	✓	✓	✓	✓
6	3	background	→	3	3	S	4	4	N	3	background	→	3	3	S	4	4	N	Completely identical CS	✓	✓	✓	✓
7	3-7	evidence	→	3	7	S	8	8	N	3-7	evidence	→	3	7	S	8	8	N	Completely identical CS	✓	✓	✓	✓
8	8|9	contrast	↔	3	8	N	9	9	N	8|9	contrast	↔	3	8	N	9	9	N	Completely identical CS	✓	✓	✓	✓
9	4|5	contrast	↔	3	4	N	5	7	N	4|5	contrast	↔	3	4	N	5	7	N	Completely identical CS	✓	✓	✓	✓
10	3-9|10-24|25	list	↔	3	9	N	10	31	N	3-9|10|13|25	list	↔	3	9	N	10	31	N	C1=C2 and A1=A2	✓	✓	✓	✓
11	6	evidence	←	6	7	S	5	5	N	6	evidence	←	6	7	S	5	5	N	Completely identical CS	✓	✓	✓	✓
12	7	circumstance	←	7	7	S	6	6	N	7	condition	←	7	7	S	6	6	N	Completely identical CS	✓	×	✓	✓
13	10-24|25	list	↔	10	24	N	25	31	N	10|13|25	list	↔	10	12	N	13	31	N	Partially identical CS	✓	✓	×	×
14	10|13	contrast	↔	10	12	N	13	24	N	13|25	list	↔	13	24	N	25	31	N	Partially identical CS	✓	×	×	×
15	11	cause	→	11	11	S	12	12	N	11|12	conjunction	↔	11	11	N	12	12	N	C1=C2 and A1=A2	×	×	×	✓
16	12	elaboration	←	11	12	S	10	10	N	11-12	elaboration	←	11	12	S	10	10	N	C1=C2 and A1=A2	✓	✓	✓	✓
17	14-24	evidence	←	14	24	S	13	13	N	14-24	evidence	←	14	24	S	13	13	N	Completely identical CS	✓	✓	✓	✓
18	14|16-17|19-20	list	↔	14	15	N	16	24	N	14|16-17|19-20|22-24	list	↔	14	15	N	16	24	N	C1=C2 and A1=A2	✓	✓	✓	✓
19	15	elaboration	←	15	15	S	14	14	N	15	elaboration	←	15	15	S	14	14	N	Completely identical CS	✓	✓	✓	✓
20	16-17|19-20	list	↔	16	18	N	19	24	N	16-17|19-20|22-24	list	↔	16	18	N	19	24	N	C1=C2 and A1=A2	✓	✓	✓	✓
21	17	concession	←	17	17	S	16	16	N	17	concession	←	17	17	S	16	16	N	Completely identical CS	✓	✓	✓	✓
22	18	evaluation-s	←	18	18	S	16	17	N	18	evaluation-s	←	18	18	S	16	17	N	Completely identical CS	✓	✓	✓	✓
23	20	purpose	←	20	20	S	19	19	N	19|20	sequence	↔	19	19	N	20	20	N	C1=A2 and A1=C2	×	×	×	×
24	21	concession	→	21	21	S	22	24	N	21	concession	→	21	21	S	22	24	N	Completely identical CS	✓	✓	✓	✓
25	22-24	e-elaboration	←	21	24	S	19	20	N	19-20|22-24	list	↔	19	20	N	21	24	N	C1=A2 and A1=C2	×	×	×	×
26	23-24	evidence	←	23	24	S	22	22	N	23	evidence	←	23	24	S	22	22	N	C1=C2 and A1=A2	✓	✓	✓	✓
27	23|24	contrast	↔	23	23	N	24	24	N	24	elaboration	←	24	24	S	23	23	N	C1=A2 and A1=C2	×	×	×	×
28	26-31	evidence	←	26	31	S	25	25	N	26-31	evidence	←	26	31	S	25	25	N	Completely identical CS	✓	✓	✓	✓
29	26|27|28|30|31	list	↔	26	26	N	27	31	N	26|27|28-30|31	list	↔	26	26	N	27	31	N	C1=C2 and A1=A2	✓	✓	✓	✓
30	27|28|30|31	list	↔	27	27	N	28	31	N	27|28-30|31	list	↔	27	27	N	28	31	N	C1=C2 and A1=A2	✓	✓	✓	✓
31	28|30|31	list	↔	28	28	N	29	31	N	28|29	list	↔	28	28	N	29	30	N	Partially identical CS	✓	✓	✓	×
32	29	interpretation	→	29	29	S	30	30	N	30	elaboration	←	30	30	S	29	29	N	C1=A2 and A1=C2	×	×	×	×
33	30|31	list	↔	29	30	N	31	31	N	28-30|31	list	↔	28	30	N	31	31	N	Partially identical CS	✓	✓	×	✓
34	32	concession	→	32	32	S	33	35	N	32	concession	→	32	32	S	33	35	N	Completely identical CS	✓	✓	✓	✓
35	33-35	antithesis	→	32	35	S	36	36	N	33-35	antithesis	→	32	35	S	36	39	N	Completely identical CS	✓	✓	✓	×
36	36	reason	→	32	36	S	37	37	N	36	reason	→	36	36	S	37	37	N	Completely identical CS	✓	✓	×	✓
37	34	circumstance	←	34	34	S	33	33	N	34	circumstance	←	34	34	S	33	33	N	Completely identical CS	✓	✓	✓	✓
38	35	evidence	←	35	35	S	33	34	N	35	evidence	←	35	35	S	33	34	N	Completely identical CS	✓	✓	✓	✓
39	38	circumstance	→	38	38	S	39	39	N	38	cause	→	38	38	S	39	39	N	Completely identical CS	✓	×	✓	✓
40	39|40-41	joint	↔	38	39	N	40	41	N	39	elaboration	←	38	39	S	36	37	N	Partially identical CS	×	×	×	×
41	40|41	list	↔	40	40	N	41	41	N	40|41	list	↔	40	40	N	41	41	N	Completely identical CS	✓	✓	✓	✓

Results	Nuclearity	Relation	Constituent	Attachment point	RST trees
F-Measure:	(33 of 40) → 0.804	(30 of 41) → 0.731	(29 of 41) → 0.707	(29 of 41) → 0.707	0.737
Inter-Annotator Agreement:	0.71	0.698	0.57	0.59	0.567

Figure 6: Comparison table and the results for text *DEU_006, RST German Learner Treebank*, using RST-TACE

References

Iria da Cunha and Mikel Iruskieta. 2010. Comparing rhetorical structures in different languages: The influence of translation strategies. *Discourse Studies*, 12(5):563–598.

Debopam Das, Maite Taboada, and Manfred Stede. 2017. The Good , the Bad , and the Disagreement : Complex ground truth in rhetorical structure analysis. In *In workshop on Recent Advances in RST and Related Formalisms*, Santiago de Compostela, Spain.

Markus Egg and Gisela Redeker. 2010. How Complex is Discourse Structure ? How Complex is Discourse Structure ? (May 2014).

Mikel Iruskieta, Iria da Cunha, and Maite Taboada. 2015. A qualitative comparison method for rhetorical structures: identifying different discourse structures in multilingual corpora. *Language Resources and Evaluation*, 49(2):263–309.

H. W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

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. 2000. The Rhetorical Parsing of Unrestricted Texts: A Surface-based Approach. *Computational Linguistics*, 26(3):395–448.

E Maziero and Thiago AS Pardo. 2009. Metodologia de avaliação automática de estruturas retóricas. In *Proceedings of the III RST Meeting (7th Brazilian Symposium in Information and Human Language Technology), Brasil*.

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

Manfred Stede. 2008. RST Revisited : Disentangling Nuclearity. In Benjamins, editor, *'Subordination' versus 'coordination' in sentence and text – from a cross-linguistic perspective.* Amsterdam.

Manfred Stede. 2016. *Handbuch Textannotation. Potsdamer Kommentarkorpus 2.0*. Universitätsverlag Potsdam, Potsdam.

Manfred Stede. 2017. Annotation Guidelines for Rhetorical Structure.

Amir Zeldes. 2016. rstWeb – A Browser-based Annotation Interface for Rhetorical Structure Theory and Discourse Relations. 2016:1–5.

Heike Zinsmeister, Marc Reznicek, Julia Ricart Brede, Christina Rosén, and Dirk Skiba. 2012. Das Wissenschaftliche Netzwerk „Kobalt-DaF". *Zeitschrift für germanistische Linguistik*, 40(3):457–458.