Bootstrapping Role and Reference Grammar Treebanks via Universal Dependencies

Kilian Evang and **Tatiana Bladier** and **Laura Kallmeyer** and **Simon Petitjean**
Heinrich Heine University Düsseldorf
Universitätsstraße 1, 40225 Düsseldorf, Germany
{evang,bladier,kallmeyer,petitjean}@phil.hhu.de

Abstract

We describe ud2rrg, a rule-based approach for converting UD trees to Role and Reference Grammar (RRG) structures. Our conversion method aims at facilitating the annotation of multilingual RRG treebanks. ud2rrg uses general and language-specific conversion rules. In order to evaluate ud2rrg, we approximate the subsequent annotation effort via measures of tree edit distance. Our evaluation, based on English, German, French, Russian, and Farsi, shows that the ud2rrg transformation of UD-parsed data constitutes a highly useful starting point for multilingual RRG treebanking. Once a sufficient amount of data has been annotated in this way, the automatic conversion can be replaced by a statistical parser trained on that data for an even better starting point.

1 Introduction

Role and Reference Grammar (RRG) (Van Valin Jr. and LaPolla, 1997; Van Valin Jr., 2005) is a grammar theory for natural language that shares with Universal Dependencies (Nivre et al., 2016; Nivre et al., 2020) the aim of being descriptively adequate across typologically diverse languages while reflecting their commonalities in its analyses. It also shares with UD a number of design characteristics, such as recognizing dissociated nuclei and the principle to "annotate what is there", eschewing the use of empty elements, cf. de Marneffe et al. (2021). In addition, RRG's separation between constituent structure and operator structure (the latter reflecting the attachment of functional elements) offers an explanatory framework for certain word-order and semantic phenomena. In recent years, the computational linguistics community has become increasingly interested in RRG and has started to formalize RRG (Osswald and Kallmeyer, 2018) and to build resources and tools to support data-driven linguistic research within RRG (Bladier et al., 2018; Bladier et al., 2020; Chiarcos and Fäth, 2019).

As illustrated in the examples in Figure 1, an important feature of RRG is the layered structure of the clause. The nucleus (NUC) contains the (verbal) predicate, arguments attach at the core (CORE) layer, and extracted arguments at the clause (CLAUSE) layer. Each layer also has a periphery, where adjuncts attach (marked PERI). Operators (closed-class elements encoding tense, modality, aspect, negation, etc.) attach at the layer over which they take scope. They are assumed to be part of a separate projection, but we collapse both projections into a single tree structure for convenience. Elements like *wh*-words in English are placed in the pre-core slot (PrCS), and the pre-clausal elements like fronted prepositional or adverbial phrases are placed in the pre-detached position (PrDP).

In this paper we describe the ongoing effort to build RRGparbank[1], a novel-length parallel RRG treebank for English, German, French, Russian, and Farsi, in a semi-automatic fashion. We focus on the automatic part. Exploiting an off-the-shelf UD parser, the text (George Orwell's novel *1984* and translations) is parsed into UD. Then, exploiting structural similarities between UD and RRG, the UD trees are automatically converted into RRG trees (§2). This conversion accelerates the process of manually annotating the corpus (§3). Once enough data has been collected in this way, we replace the rule-based conversion with a statistical RRG parser trained on the collected data. A series of experiments shows that the statistical RRG parser offers a better starting point for annotating once approximately 2 000 sentences

[1] https://rrgparbank.phil.hhu.de

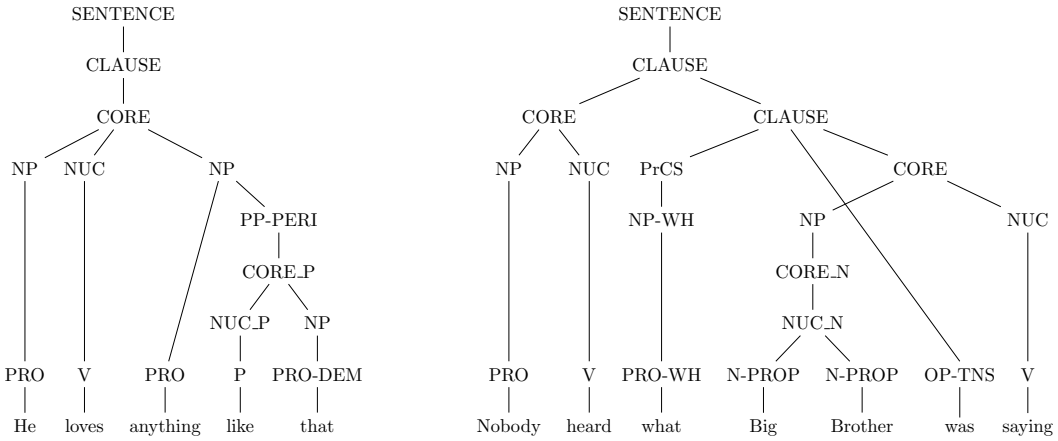


Figure 1: Examples of RRG annotation. Punctuation marks are omitted.

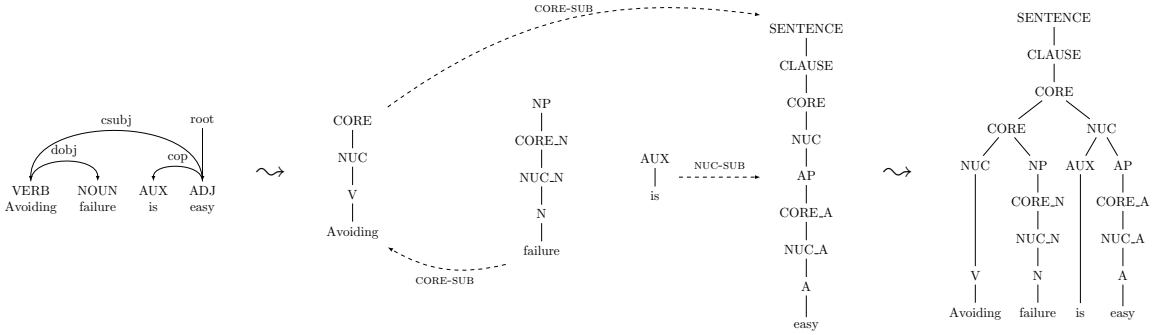


Figure 2: UD tree, RRG derivation tree and resulting RRG tree for the sentence *Avoiding failure is easy*.

are available for training (§4). Finally, we give a qualitative comparison between our converter and that of Chiarcos and Fäth (2019), which targets a slightly different flavor of RRG (§5).

2 UD to RRG Conversion

2.1 Auxiliary Formalism

We define a custom formalism, inspired by tree grammar formalisms such as LTAG (Joshi and Schabes, 1997) that allows us to treat RRG trees as being composed from lexically anchored *elementary trees* via a number of composition operations. Figure 2 and Figure 3 show examples: in the middle, there are a number of elementary trees and the operations with which they are combined (the derivation) and on the right there is the resulting RRG tree. The set of operations is RRG-specific:

- NUC-SUB: presupposes that the host tree is clausal.[2] If the guest tree is clausal, attach its NUC node under the host tree's NUC node, and merge its CORE, CLAUSE and SENTENCE nodes (if any) into the corresponding nodes of the host tree.[3] If not, attach its root under the host tree's NUC node.

- NUC-COSUB: presupposes that both trees are clausal. Create a NUC node above the host tree's NUC node (if it doesn't exist yet), and attach the guest tree's NUC node to that. Merge the CORE, CLAUSE, and SENTENCE nodes.

- NUC-COORD: presupposes that both trees are clausal. Attach the guest tree's NUC node under the host tree's CORE node. Merge the CORE, CLAUSE, and SENTENCE nodes.

[2] A *clausal* tree is an elementary tree whose lexical anchor is the head of a clause. Its spine starts with SENTENCE, CLAUSE, CORE, or NUC, followed by nodes for the lower clausal layers.

[3] By merging we mean attaching any children of the former under the latter.

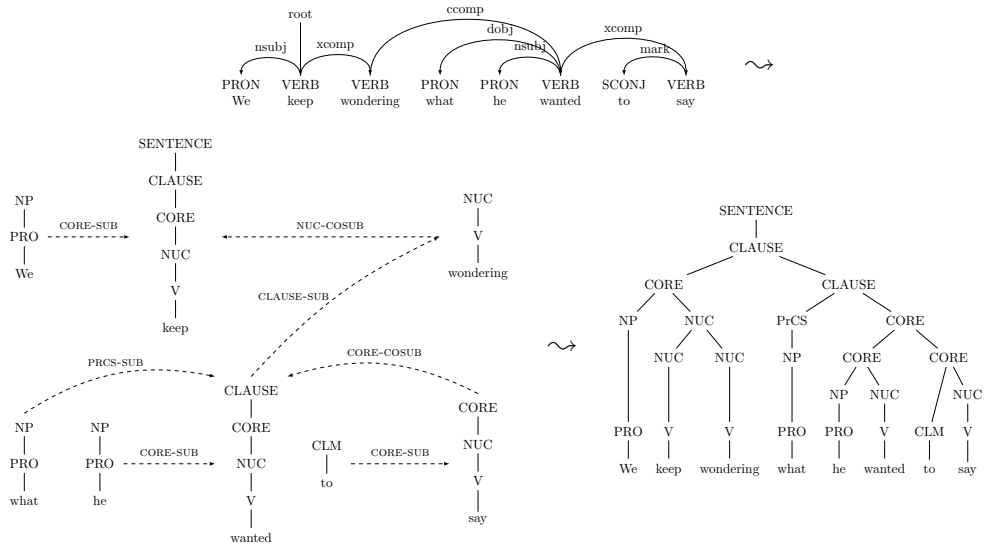


Figure 3: UD tree, RRG derivation tree and resulting tree for the sentence *We keep wondering what he wanted to say*.

- CORE-SUB: presupposes that the host tree is clausal. If the guest tree is clausal, attach its CORE node under the host tree's CORE node, and merge the CLAUSE and SENTENCE nodes (if any). If not, attach its root under the host tree's CORE node.
- CORE-COSUB: presupposes that both trees are clausal. Create a CORE node above the host tree's CORE node (if it doesn't exist yet) and attach the guest tree's CORE node to that. Merge the CLAUSE and SENTENCE nodes.
- CORE-COORD: presupposes that both trees are clausal. Attach the guest tree's CORE node under the host tree's CLAUSE node. Merge the CLAUSE and SENTENCE nodes.
- CLAUSE-SUB: presupposes that the host tree is clausal. If the guest tree is clausal, attach its CLAUSE node under the host tree's CLAUSE node, and merge the SENTENCE nodes (if any). If not, attach its root under the host tree's CLAUSE node.
- PRCS-SUB: create a left PrCS daughter of the host tree's CLAUSE node (if it doesn't exist yet) and attach the guest tree's root under it.
- PRDP-SUB: create a left PrDP daughter of the host tree's SENTENCE node (if it doesn't exist yet) and attach the guest tree's root under it.
- WRAP: attach the host under a designated node (marked $\star$) of the guest (used for attaching preposition complements under PPs).

2.2 General Conversion Rules

If we view the derivations, as exemplified in Figures 2 and 3, as *derivation trees* where nodes are labeled with elementary trees and edges are labeled with operations, then this derivation tree is isomorphic to a corresponding UD tree. What remains to do to convert UD trees to RRG trees is to specify a set of rules that relabel nodes in UD trees with RRG elementary trees, and edges with operations. We try to keep these rules as local as possible, ideally looking only at one UD node and its incoming edge at a time, so a simple recursive traversal of the UD tree suffices. However, as we will see, in some cases we need to take a little more context into account. Table 1 shows the rules used in the example conversions.

UD's content-word-centric approach is a good fit for the conversion to RRG regarding, e.g., copulas, modal, tense, and aspect operators, which RRG treats not as heads of verb phrases but as additional

	label	POS	additional conditions	elementary tree	operation
1	root	ADJ		(SENTENCE (CLAUSE (CORE (NUC (AP (CORE_A (NUC_A (A <>))))))))	
2	cop	AUX		(AUX <>)	NUC-SUB
3	csubj	VERB		(CORE (NUC (V <>)))	CORE-SUB
4	dobj	NOUN		(NP (CORE_N (NUC_N (N <>))))	CORE-SUB
5	root	VERB		(SENTENCE (CLAUSE (CORE (NUC (V <>)))))	
6	nsubj	PRON		(NP (PRO <>))	CORE-SUB
7	dobj	WP or WP$		(NP (PRO <>))	PRCS-SUB
8	mark	SCONJ		(CLM <>)	CORE-SUB
9	ccomp	VERB		(CORE (NUC (V <>)))	CORE-SUB
10	ccomp	VERB	verb of cognition/saying	(CLAUSE (CORE (NUC (V <>))))	CLAUSE-SUB
11	xcomp	VERB		(CORE (NUC (V <>)))	CORE-COSUB
12	xcomp	VERB	phase verb	(NUC (V <>))	NUC-COSUB
13	xcomp	VERB	raising verb	(CORE (NUC (V <>)))	CORE-COORD
14	case	ADP		(PP (CORE_P* (NUC_P (P <>))))	WRAP

Table 1: Examples of rules. Rules 1–12 are the ones used in Figures 2 and 3. Rules 7, 10, 12, and 13 are examples of rules whose implementation requires language-specific POS tags or lexicons.

operators attaching to clauses, cores, or nuclei. By contrast, prepositions are treated as heads of PPs and thus necessitate a slightly more complicated rule (WRAP) to wrap the prepositional complement in a PP. Overall, RRG's approach can be characterized as more content-word-centric than function-word-centric. This and the ready availability of UD resources made UD a more natural starting point for our conversion project than more function-word-centric variants such as SUD (Gerdes et al., 2018).

2.3 Special Conversion Rules

Rules can also make reference to lexical and other language-specific knowledge. One area where this is important is clause linkage, which in UD is always represented with `conj`, `ccomp`, or `xcomp` relations, but in RRG splits up into a more fine-grained set of *juncture-nexus* types. For example, while we subordinate clausal complements at the CORE level by default (Rule 9 in Table 1), clausal complements of verbs of cognition and saying typically require subordination at the CLAUSE level (Rule 10). Similarly, while we cosubordinate open clausal complements at the CORE level by default (Rule 11), open clausal complements of phase verbs as in *starts walking*, *keep wondering* or *stopped believing* require cosubordination at the NUC level (Rule 12). This is illustrated in Figure 3. We have so far implemented this rule for English and for German. For English, we determine the verb class by lookup in the VerbNet lexical database (Kipper-Schuler, 2005). For German, as far as we are aware, similar resources such as GermaNet (Hamp and Feldweg, 1997), provide sets of verb classes which are less fine-grained. Therefore, the equivalent conversion rule for German uses a handwritten set of verbs instead of a lexical database. We have also defined rules to recognize `xcomp` instances encoding the raising construction and trigger core coordination by its associated lemma *seem* in English or *scheinen* in German (Rule 13). Due to the low frequency of relevant phenomena and the inevitable brittleness of rules, we have left these specialized conversion rules as a proof-of-concept and not aimed for more extensive coverage, relying on statistical predictions (see below) and annotators instead for the purpose of building RRGparbank.

2.4 Implementation and Workflow

The text basis for RRGparbank is provided by George Orwell's novel *1984* as well as translations to German, French, Russian and Farsi. The English and Farsi texts, their segmentation into sentences and tokens as well as POS tags and lemmas are taken from the MULTEXT-East dataset (Erjavec, 2017), which also provides the (non-annotated) Russian text. The French and German data was built using the Orwell (1972) and Orwell (2003) editions, respectively. A large part of the German data was annotated by hand following the guidelines of the MULTEXT-East dataset. We used UDpipe2 (Straka, 2018) for segmentation, tagging, and lemmatization of the Russian, French and the non-annotated German data. UD parses for all languages are also provided by UDPipe2.[4]

[4]The reported Labeled Attachment Scores for the 5 languages on UD treebanks are as follows: 85.8% for English, 81.2% for German, 84.3% for Farsi, 83.5% for French and 85.3% for Russian.

Timestamp	nBURP	LF1	failed
#1	0.66	61.02	1 100
#2	0.57	64.09	773
#3	0.47	68.75	355
#4	0.33	72.51	221
#5	0.20	79.96	0

Table 2: nBURP and LF1 scores for the output of ud2rrg using Russian data (4 635 sentences), at different steps of development. Sentences that could not be converted are replaced with flat dummy trees.

In the next step, we use a script called ud2rrg[5], which we developed based on the formalism described above, to convert the UD trees to RRG. It performs a traversal of each UD tree and at each node applies the matching rule, thereby gradually building up an RRG tree. In the rare cases where conversion fails for a node because there is no matching rule (e.g., with rare combinations of POS and grammatical relation), conversion fails and a dummy tree is generated where all tokens are attached to the root.

For the results reported in this paper, 13 annotators with training in RRG annotated 5 453 English, 5 723 German, 2 177 French, 4 675 Russian, and 1 110 Farsi sentences over a time period of 21 months.[6] They were provided the output of ud2rrg and corrected the trees using a graphical interface. The graphical interface and the annotation guidelines were based on RRGbank (Bladier et al., 2018). Development of ud2rrg was ongoing during this period and informed by manual inspection of sentences that failed to convert and of changes annotators made to the ud2rrg output.

Annotation on different languages started at different times. We used the same ud2rrg for all languages, but each new language typically brings with it a number of POS tags and constructions that have not or not much been seen in the data so far, meaning that rules have to be refined and added before ud2rrg performs as well on the new as on the old languages. As an example, consider the case of Russian. Table 2 shows the performance of ud2rrg on Russian at different points in time after its introduction as a new language. We measure the performance in nBURP (smaller is better), LF1 (larger is better), and number of failed sentences (smaller is better) – details are given below in Section 3. Timestamp #1 corresponds to the introduction of the Russian data in the annotation interface. Between #1 and #2, ongoing development of ud2rrg took into account, among other data, Russian gold data produced by annotators. The first rules making specific reference to Russian lexemes were added between #2 and #3 (7 new rules and 2 extensions of existing rules), leading to significantly better performance. Timestamp #4 is a week after #3, while #5 is the time of redaction of this article. The scores keep improving with time, as regular evaluations on the updated gold data indicates which transformation rules are missing, or need an update. Annotators are also encouraged to report sentences for which the transformation is problematic, or fails.

As of this writing, ud2rrg contains about 278 rules, 4 of which depend on language-specific semantic lexical resources to select a juncture-nexus type. In addition, we have language-specific routines to determine finer-grained parts of speech for function words, such as negation particles, negative determiners, indefinite pronouns, demonstrative pronouns, clitic pronouns, or negative pronouns.

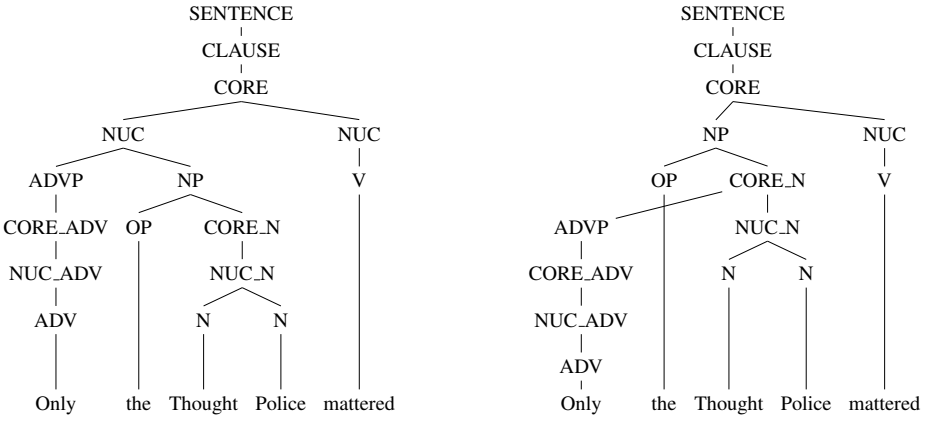
3 Impact on Annotation Effort

Correcting automatically pre-annotated data facilitates the annotation of the treebank because the data no longer need to be annotated from scratch. In this section we estimate the impact of pre-annotation on the human effort of creating treebank data. Specifically, we try to measure the mechanical effort it takes annotators to move, insert, delete, and relabel tree nodes in our graphical drag-and-drop annotation interface (Bladier et al., 2018). For this study, we ignore the cognitive cost of annotation decisions, which is much harder to measure.

Roughly speaking, the more similar a pre-annotated tree to the gold tree, the fewer drag-and-drop

[5] https://gitlab.com/treegrasp/ud2rrg

[6] The data is a snapshot of RRGparbank as of 2021-09-17.



(a) ud2rrg tree (b) gold tree

Figure 4: Example pair of gold tree and corresponding ud2rrg output.

operations annotators will need. Established tree similarity measures include tree edit distance (TED) (Zhang and Shasha, 1989) and EVALB (Collins, 1997). However, it is well known that these measures tend to over-penalize attachment errors (Bangalore et al., 1998; Emms, 2008) because constituents that have to be reattached do not incur a unitary cost but rather a cost proportional to their size or to the length of the path between the predicted and the correct attachment site. This contrasts with our graphical annotation interface where reattachment is a single drag-and-drop operation. As an example, consider Figure 4. Here, we have to reattach one ADVP subtree and delete one NUC node in order to transform the ud2rrg tree into the gold tree. However, the tree edit distance is 6 because reattachment incurs a cost not only for the ADVP node but for all its descendants. Similarly, EVALB will count not 2 but 3 spans (NUC, NP, CORE_N) as "false positives" in the ud2rrg tree. This effect gets worse with longer reattachments.

We are not aware of a polynomial algorithm to compute optimal edit scripts between trees when reattachment is allowed as a single operation. Instead, we use an approximate but principled algorithm that counts the number of operations needed to turn the predicted tree into the gold tree when recreating the constituents of the gold tree by modifying the predicted tree in a strict bottom-up fashion, recreating smaller constituents first and then moving on to larger ones. This algorithm, called "bottom-up replugging" (BURP), is described in detail in Appendix B. In our example, BURP first recreates the CORE_N subtree, for which the ADVP subtree needs to be moved down (cost 1). It then recreates the NP subtree and deletes the NUC node (cost 1). The trees are now identical, with total cost 2, which is exactly the number of operations intuitively needed. While not necessarily optimal, we conjecture that BURP approximates the strategies that human annotators use to edit trees, and that its scores are therefore a better predictor of human annotation effort than TED or EVALB.

For our evaluation, we use all three measures. For TED and BURP, we normalize the score by the number of brackets in the gold RRG tree, since trees with a more complex internal structure require more editing than simpler trees. The results are given in Table 3.

4 Comparison with statistical parsing

We compare the output of ud2rrg with parsing the sentences using the statistical neural parser ParTAGe (Bladier et al., 2020), developed for RRG-based tree rewriting grammars. We evaluate how much training data is needed for the statistical parser to outperform the rule-based conversion approach. For the experiments, we did not distinguish between silver and gold data[7] but split all gold and silver data up into 4 385 training, 542 development, and 526 test sentences for English. We randomly shuffle the training data and use the first n trees for training. Our experiments show that the statistical parser needs around

[7]Silver sentences are annotated by one annotator whereas gold sentences are annotated by at least two annotators. We had 5 228 gold and 225 silver sentences in the English subcorpus in total.

approach	train sz.	failures		nTED		LF1 (exact match)		nBURP	
		dev	test	dev	test	dev	test	dev	test
ud2rrg		0	0	0.32	0.34	76.97 (90)	76.51 (84)	0.20	0.21
statist.	500	137	131	0.43	0.42	62.65 (70)	63.45 (85)	0.64	0.63
parser	1 000	0	1	0.35	0.35	68.93 (88)	70.27 (85)	0.30	0.29
	2 000	0	0	0.27	0.27	75.35 (128)	76.13 (113)	0.22	0.21
	3 000	0	0	0.25	0.24	77.93 (135)	78.73 (133)	0.19	0.18
	4 000	0	0	0.23	0.22	79.56 (149)	80.62 (135)	0.18	0.17
	>4 000	0	0	0.23	0.22	79.75 (157)	80.30 (137)	0.17	0.16
# sent.		542	526	542	526	542	526	542	526
∅ len.		13.99	14.02	13.99	14.02	13.99	14.02	13.99	14.02

Table 3: Comparison of UD parsing for English followed by rule-based ud2rrg conversion vs. statistical RRG parsing (Bladier et al., 2020), depending on the amount of RRG training data available. The evaluation does not consider function tags and punctuation. The numbers in brackets indicate the amount of exactly matched produced trees. Sentences that could not be converted/parsed are counted in the evaluation as flat dummy trees. We use our BURP measure (normalized by number of constituents in the gold tree) as well as tree edit distance (Zhang and Shasha, 1989) normalized in the same way, and EVALB LF1 (Collins, 1997). All three measures show that statistical parsing starts to outperform ud2rrg at around 2 000 training sentences.

2 000 pre-annotated trees for training to surpass the rule-based conversion.

We also evaluate the UD conversion on other languages (see Table 4).[8] In cases where ud2rrg could not convert UD parses to trees, we evaluate the scores as if the trees were annotated from scratch. Concretely, we measure the distance from flat dummy trees where each pre-terminal has a dummy POS tag and attaches directly to the root. The results show that about a fifth of the sentences are converted directly to the gold standard for different languages and in general the annotators' effort is reduced for the majority of sentences compared to annotation from scratch (represented as baseline in Table 4). These findings clearly show that using the rule-based UD conversion approach can be a good starting point for pre-annotation of a multilingual treebank.

language	baseline		ud2rrg		# sents	∅ len.	failures	# sents
	nBURP	LF1	nBURP	LF1	(annot.)	(annot.)		(entire corpus)
de	1.24	6.56	0.18	79.24 (926)	5723	17.00	9	6661
fr	1.22	8.97	0.21	79.80 (402)	2177	12.57	1	7261
ru	1.18	7.64	0.20	79.96 (939)	4635	11.76	0	6669
fa	1.16	9.14	0.30	72.09 (211)	1110	9.01	37	6604

Table 4: Comparison of normalized BURP and EVALB F1 scores of ud2rrg for German, French, Russian, and Farsi evaluated on the full set of annotated sentences without taking into account punctuation and function tags. The baseline is annotation from scratch, starting with flat dummy trees. For sentences where ud2rrg fails, we fall back to the baseline. The numbers in brackets show produced trees exactly matching with gold annotations.

5 Related Work

The availability of UD corpora for a big variety of languages makes them appealing to use for creating linguistic resources for different NLP tasks. Fancellu et al. (2017) and Reddy et al. (2017) describe algorithms for conversion of UD structures to logical forms enabling an almost language-independent transformation. Ranta and Kolachina (2017) develop an approach to convert UDs into abstract syntactic

[8] Note that these data do not fully reflect differences between languages in RRGparbank, since the annotation is still ongoing and the current amount of covered data and annotated syntactic phenomena is different for each language.

annotations to create treebanks based on the Grammatical Framework (GF) formalism for multilingual grammars (Ranta, 2011).

Closest to our work is that of Chiarcos and Fäth (2019) who define a RDF/SPARQL-based converter to RRG, using as input not only UD but also semantic role annotation. The data for which both the input (partially manually corrected UD) and the output is publicly available[9] consists of 351 examples from the textbook of Van Valin Jr. and LaPolla (1997). While their converter was developed on this kind of data, for us it presents a new domain. After normalizing away notational differences and ignoring operator attachment as well as POS tags (see below) but without any updates to ud2rrg, we obtained an nBURP of 0.16, an nTED of 0.18, and an F1 score of 85.75, with 15.38% exact matches. We then performed a qualitative comparison on 100 randomly chosen sentences to gain insights into types of mistakes and to inform future development. We summarize our findings here; the full results are provided in Appendix A.

Notational conventions A large part of the differences are purely notational and can be automatically normalized away: C&F attach all punctuation at the root whereas we leave it attached at smaller phrases as in UD. We mark *wh*-phrases with -REL or -WH labels, C&F don't. C&F mark arguments with ARG nodes, which we don't have, and peripheries with PERIPHERY nodes, while we mark the children with -PERI instead. Some nonterminals have slightly different names, e.g., COREn vs. CORE_N, LDP vs. PrDP, or ADJ vs. AP. We also ignore part-of-speech tags because C&F do not attempt to convert the input POS tags into the POS tags conventionally used in RRG analyses.

Tense, modality, and aspect operator attachment In RRG, tense operators attach at the CLAUSE level, modal operators at the CORE level, and aspect operators at the NUC level. This means that, e.g., a tensed auxiliary verb as in *she has seen* or a tensed modal verb as in *he can see* attach at more than one level, namely at both CLAUSE and NUC, and at both CLAUSE and CORE, respectively. In C&F's annotation, this is so. By contrast, our guidelines limit annotations to trees for ease of processing and by convention only attach at one level, which is typically CLAUSE. We tried to ignore these differences in attachment by removing the non-CLAUSE additional edges from C&F's annotation. 10 differences due to operator attachment remain. Since auxiliary and modal verbs form a closed class, the multiple attachment would be easy to restore.

Theoretical assumptions (51 instances) Some of the remaining differences can be explained by ud2rrg following conventions set down in our RRGbank-based annotation guidelines which differ from those followed in C&F's data. These are not mere notational differences but have potential theoretical significance because they reflect different assumptions about the internal structure of phrases, etc. These differences will be used to check and revise our annotation guidelines. For example, we annotate numerals using "quantifier phrases" (QPs), attach attributive APs at CORE_N rather than NUC_N level, assume a full AP rather than a simple nucleus for predicative adjectives, treat possessive pronouns like determiners and do not place them under NPIP, treat prepositions introducing adverbial clauses as clause linkage markers (CLM) rather than prepositions, do not distinguish CONJ from CLM, etc.

Bugs (25 instances) Some differences are bugs in ud2rrg which can easily be fixed in future development, e.g., failure to convert prepositions `mark`ing clauses into PPs rather than CLM-marked clauses, failure to handle `nmod:tmod` dependents as adverbial modifiers, attachment of *wh*-PPs in PrDP rather than PrCS, or failure to recognize *wh*-movement when the subject is a passive subject.

Limitations (84 instances) Telling the differences between an argument PP and a peripheral (adjunct) PP is hard and currently out of scope for ud2rrg. C&F use semantic role information to predict this. Relatedly, ud2rrg currently does not distinguish between PPs with and without internal layers (CORE_P and NUC_P).

Clause linkage (20 instances) Similarly, mapping `conj`, `ccomp`, and `xcomp` dependencies to the appropriate juncture-nexus type for linking clauses together is complex. As described in Section 2, we

[9] `https://github.com/acoli-repo/RRG`

have some rules to address this heuristically, but many cases are not yet covered and may also require semantic role or other lexical information to resolve.

Bad input (12 instances) Incorrect UD input sometimes leads to incorrect ud2rrg output. For example, the input data contain a number of unspecific `dep` relations which are then not correctly handled. There are also instances of wrongly resolved PP attachment ambiguity and the occasional confusion of a relative clause with an adverbial clause, and vice versa.

Error in gold standard (7 instances) Finally, we also discovered a handful of apparent errors in C&F's annotation. For example, the genitive suffix *'s* is always attached to the root instead of inside the NPIP, and some PrCS nodes appear to be spurious.

6 Conclusions

We have presented a rule-based algorithm for converting Universal Dependencies to RRG trees as a way to bootstrap RRG treebanks. By mapping UD nodes to RRG fragments and grammatical relations to operations that combine these fragments, it provides a principled mapping between the two formalisms. Language-independent at core, the algorithm can be extended with language-specific rules to incorporate lexical and other language-specific knowledge. We have shown that by basing RRG annotation on automatically converted trees, the number of tree manipulation operations that annotators have to perform is considerably reduced compared to annotating from scratch. We have also shown that for annotating English, a statistical parser trained on sentences annotated so far starts to produce more accurate trees than our rule-based conversion at around 2 000 training sentences. Finally, we have performed a detailed qualitative comparison with the output of another converter and pinned down the remaining issues for ours. In future work, we will consider applying the ud2rrg algorithm to the data from Parallel Universal Dependencies corpora (Zeman et al., 2020). Moreover, ud2rrg allows bootstrapping of further RRG treebanks for different languages, based on existing UD treebanks.

Acknowledgments

We thank two anonymous reviewers for their insightful comments, as well as all people involved in the creation and annotation of RRGparbank (David Arps, Davy Baardink, Kata Balogh, Elif Benli, Sarah Fabian, Valeria Generalova, Zahra Ghane, Robin Möllemann, Natalia Moors, Rainer Osswald) for their help with collecting the data and fruitful discussions. We also thank Behrang QasemiZadeh, Jule Pohlmann and Roland Eibers for preparing the German data, Rainer Osswald for integrating the French data and completing the German data, Naima Grebe for helping with the implementation of special conversion rules for English phase verbs, and Robert D. Van Valin Jr. for contributing to annotation decisions. This work was carried out as a part of the research project TreeGraSP[10] funded by a Consolidator Grant of the European Research Council (ERC).

References

Srinivas Bangalore, Anoop Sarkar, Christy Doran, and Beth-Ann Hockey. 1998. Grammar and parser evaluation in the XTAG project. In *Proceedings of LREC Workshop on Evaluation of Parsing Systems*.

Tatiana Bladier, Andreas van Cranenburgh, Kilian Evang, Laura Kallmeyer, Robin Möllemann, and Rainer Osswald. 2018. RRGbank: a role and reference grammar corpus of syntactic structures extracted from the penn treebank. In *Proceedings of the 17th International Workshop on Treebanks and Linguistic Theories (TLT 2018)*. Linköping Electronic Conference Proceedings.

Tatiana Bladier, Jakub Waszczuk, and Laura Kallmeyer. 2020. Statistical parsing of tree wrapping grammars. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6759–6766, Barcelona, Spain (Online), December. International Committee on Computational Linguistics.

[10] `https://treegrasp.phil.hhu.de`

Christian Chiarcos and Christian Fäth. 2019. Graph-Based Annotation Engineering: Towards a Gold Corpus for Role and Reference Grammar. In Maria Eskevich, Gerard de Melo, Christian Fäth, John P. McCrae, Paul Buitelaar, Christian Chiarcos, Bettina Klimek, and Milan Dojchinovski, editors, *2nd Conference on Language, Data and Knowledge (LDK 2019)*, volume 70 of *OpenAccess Series in Informatics (OASIcs)*, pages 9:1–9:11, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain, July. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47(2):255–308, 07.

Martin Emms. 2008. Tree-distance and some other variants of evalb. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC 2008)*, Marrakech, Morocco.

Tomaž Erjavec. 2017. MULTEXT-East. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, pages 441–462, Dordrecht. Springer Netherlands.

Federico Fancellu, Siva Reddy, Adam Lopez, and Bonnie Webber. 2017. Universal dependencies to logical forms with negation scope. *arXiv preprint arXiv:1702.03305*.

Kim Gerdes, Bruno Guillaume, Sylvain Kahane, and Guy Perrier. 2018. SUD or surface-syntactic Universal Dependencies: An annotation scheme near-isomorphic to UD. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 66–74, Brussels, Belgium, November. Association for Computational Linguistics.

Birgit Hamp and Helmut Feldweg. 1997. GermaNet - a lexical-semantic net for German. In *Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*.

Aravind K. Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, pages 69–123. Springer, Berlin.

Karin Kipper-Schuler. 2005. *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia, May. European Language Resources Association (ELRA).

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France, May. European Language Resources Association.

George Orwell. 1972. *1984*. Gallimard. French translation by Amélie Audiberti (first published 1950).

George Orwell. 2003. *1984*. Ullstein, 37th edition. German translation by Kurt Wagenseil (first published 1950 by Alfons Bürger Verlag).

Rainer Osswald and Laura Kallmeyer. 2018. Towards a formalization of Role and Reference Grammar. In Rolf Kailuweit, Eva Staudinger, and Lisann Künkel, editors, *Applying and Expanding Role and Reference Grammar*, (NIHIN Studies), pages 355–378. Albert-Ludwigs-Universität, Universitätsbibliothek, Freiburg.

Aarne Ranta and Prasanth Kolachina. 2017. From universal dependencies to abstract syntax. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 107–116.

Aarne Ranta. 2011. *Grammatical framework: Programming with multilingual grammars*, volume 173. CSLI Publications, Center for the Study of Language and Information Stanford.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. *arXiv preprint arXiv:1702.03196*.

Brian Roark. 2002. Evaluating parser accuracy using edit distance. In *Proceedings of the LREC 2002 workshop: Beyond PARSEVAL: Towards Improved Evaluation Measures for Parsing Systems*.

Geoffrey Sampson and Anna Babarczy. 2003. A test of the leaf-ancestor metric for parse accuracy. *Journal of Natural Language Engineering*, 9:365–380.

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium, October. Association for Computational Linguistics.

Robert D. Van Valin Jr. and Randy J. LaPolla. 1997. *Syntax: Structure, meaning and function*. Cambridge University Press.

Robert D. Van Valin Jr. 2005. *Exploring the Syntax-Semantics Interface*. Cambridge University Press.

Daniel Zeman, Joakim Nivre, Mitchell Abrams, Noëmi Aepli, Željko Agic, Lars Ahrenberg, et al. 2020. Universal dependencies 2.5. *LINDAT/CLARIAHCZ digital library at the Institute of Formal and Applied Linguistics (UFAL), Faculty of Mathematics and Physics, Charles University. url: http://hdl. handle. net/11234/1-3226*.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262.

A Qualitative Comparison with Chiarcos and Fäth (2019)

The following table contains the results of the qualitative evaluation of our converter on 100 randomly selected example sentences from Van Valin Jr. and LaPolla (1997) as annotated by Chiarcos and Fäth (2019). The first column indicates the sentence number in their release, the second the type of difference (different **op**erator attachment, **theo**retical assumption, **bug**, **limit**ation, clause **link**age, bad input (**ud**), or error in **gold** standard), and the third column contains a brief description of the difference. Empty second and third columns indicates sentences from our sample with no difference after normalization.

sentence	type	description
5	op	neg at CLAUSE vs. CORE
5	bug	failure to attach fronted non-wh object in PrCS
8	ud	dobj → dep
8	limit theo	PP internal structure
9	bug	fronted wh-PP attached to PrDP instead of PrCS
9	limit theo	PP internal structure
12	limit	failure to recognize PERI
17	limit theo	PP internal structure
17	ud	fronted wh-object attached with dep
18	ud	nmod:tmod → dep
18	limit	failure to recognize PERI
18	limit theo	PP internal structure
24		
26	limit theo	PP internal structure
29	limit theo	PP internal structure
30	limit theo	PP internal structure
31	limit theo	PP internal structure
31	limit theo	PP internal structure
35	acoli	”that” treated as NP when it is a determiner
48	acoli	’s attached to root
48	theo	AP-PERI always attaches at CORE_N, not NUC_N
48	limit theo	PP internal structure
50	limit theo	PP internal structure
54	limit theo	PP internal structure
54	theo	ADVP-PERI always attaches at CORE, not NUC
54	theo	AP-PERI always attaches at CORE, not NUC
55	theo	ADVP-PERI always attaches at CORE, not NUC
55	theo	AP-PERI always attaches at CORE, not NUC
55	bug	failure to recognize PoDP
55	limit theo	PP internal structure
58	theo	”by” passive subject treated as argument, not adjunct
58	limit theo	PP internal structure
59	limit theo	PP internal structure
62		
67	theo	”by” passive subject treated as argument, not adjunct
68	limit theo	PP internal structure
71	limit	failure to recognize PERI
74	ud	compound → dobj
74	limit	failure to recognize PERI
76	theo	possessive pronoun treated as definiteness operators, not placed in NPIP
76	theo	AP-PERI always attaches at CORE_N, not NUC_N
76	limit theo	PP internal structure

89		
91		
99	bug	raising construction where the subordinated predicate is an adjective is wrongly classified as dependency parsing error
103	ud	acl:relcl → advcl
103	ud	”to whom” is not a subtree
103	limit theo	PP internal structure
104	limit theo	PP internal structure
104	limit theo	PP internal structure
104	bug	failure to recognize PrCS when subject is marked nsubjpass
105	limit theo	PP internal structure
105	limit theo	PP internal structure
105	bug	failure to recognize PrCS when subject is marked nsubjpass
111		
112		
120	limit	failure to recognize PERI
120	limit theo	PP internal structure
120	limit	failure to recognize PERI
123	limit theo	PP internal structure
123	limit	failure to recognize PERI
123	limit	failure to recognize PERI
125	limit	failure to recognize PERI
125	limit	failure to recognize PERI
127		
132	theo	to-infinitive that replaces a relative clause treated as CLAUSE-PERI and attached in NUC_N instead of CORE attached at CORE_N
132	limit theo	PP internal structure
133	limit theo	PP internal structure
134	ud	NP with relcl instead of SENTENCE with fronted dobj
135	limit theo	PP internal structure
139		
140	limit theo	PP internal structure
141	limit theo	PP internal structure
146	limit theo	PP internal structure
150	limit theo	PP internal structure
153	theo	AP-PERI always attaches at CORE_N, not NUC_N
153	limit theo	PP internal structure
153	limit theo	PP internal structure
153	limit theo	PP internal structure
153	limit theo	PP internal structure
153	limit theo	PP internal structure
159	limit theo	PP internal structure
159	limit theo	PP internal structure
160	limit theo	PP internal structure
160	limit theo	PP internal structure
162	limit theo	PP internal structure
168	limit theo	PP internal structure
168	limit	failure to recognize PERI
171	limit	failure to recognize PERI
171	limit theo	PP internal structure
172	limit theo	PP internal structure

172	limit	failure to recognize PERI
174	limit theo	PP internal structure
182		
183	limit	failure to recognize PERI
183	limit theo	PP internal structure
188	bug	fronted wh-PP attached to PrDP instead of PrCS
188	limit theo	PP internal structure
190	op	neg at CLAUSE vs. CORE
190	link	parataxis handled as CORE cosubordination instead of SENTENCE coordination
190	limit	failure to recognize/handle cleft construction
200	theo	single complex NUC instead of NUC cosubordination
200	theo	predicative adjective: simple NUC vs. AP
200	link	CORE coordination vs. cosubordination
202	link	CORE coordination vs. cosubordination
205	link	CORE coordination vs. subordination
205	theo	predicative adjective: simple NUC vs. AP
207		
211	theo	single complex NUC instead of NUC cosubordination
211	theo	predicative adjective: simple NUC vs. AP
217	bug	failure to handle nmod:tmod as adverbial
217	theo	single complex NUC instead of NUC cosubordination
217	theo	predicative adjective: simple NUC vs. AP
225	link	CORE coordination vs. CLAUSE subordination
229	bug	advcl as PP-PERI vs. CLAUSE
229	limit theo	PP internal structure
229	limit theo	PP internal structure
231	op	modal verbs attach at CORE
233	op	modal verbs attach at CORE
233	link	CORE coordination vs. cosubordination
236	limit	failure to recognize PERI
236	limit theo	PP internal structure
237		
239	link	CLAUSE vs. CORE subordination
239	acoli	spurious PrCS?
240	ud	nominalized clause parsed wrong
240	theo	”by” passive subject treated as argument, not adjunct
247	limit theo	PP internal structure
247	bug	failure to handle nmod:tmod as adverbial
247	link	CLAUSE vs. CORE subordination
248	bug	advcl as PP-PERI vs. CLAUSE
248	limit theo	PP internal structure
255	ud	xcomp $\rightarrow$ dep
258	link	CORE coordination vs. cosubordination
259	link	CORE coordination vs. subordination
260	link	CORE coordination vs. cosubordination
260	theo	possessive pronoun treated as definiteness operators, not placed in NPIP
260	theo	possessive pronoun treated as definiteness operators, not placed in NPIP
260	bug	advcl as PP-PERI vs. CLAUSE
261	link	CORE coordination vs. cosubordination
261	theo	possessive pronoun treated as definiteness operators, not placed in NPIP

261	theo	possessive pronoun treated as definiteness operators, not placed in NPIP
261	bug	advcl as PP-PERI vs. CLAUSE
263	link	CORE coordination vs. subordination
265	link	CLAUSE coordination vs. cosubordination
265	bug	advcl as PP-PERI vs. CLAUSE
265	theo	possessive pronoun treated as definiteness operators, not placed in NPIP
265	limit theo	PP internal structure
276	theo	”by” passive subject treated as argument, not adjunct
276	theo	”by” passive subject attached at higher vs. lower CORE
278	bug	fronted advcl not in PrDP
278	bug	ud2rrg advcl as PP vs. CLAUSE
292	link	CORE subordination vs. cosubordination
293		
294	limit theo	PP internal structure
294	link	CLAUSE in CORE vs. CORE subordination
294	acoli	spurious PrCS?
294	theo	predicative adjective: simple NUC vs. AP
295	limit theo	PP internal structure
295	link	CLAUSE under CORE vs. CORE subordination
297	limit theo	PP internal structure
297	acoli	’s attached to root
297	acoli	’s attached to root
297	theo	QP
297	theo	QP
297	theo	AP-PERI always attaches at CORE, not NUC
297	theo	AP-PERI always attaches at CORE, not NUC
297	theo	we don’t distinguish between CONJ and CLM
298	theo	we don’t distinguish between CONJ and CLM
298	theo	QP
298	theo	QP
298	limit	article of coordinated NP attaches too low
301	theo	QP
301	theo	AP-PERI always attaches at CORE, not NUC
305	op	at CLAUSE vs. CORE
305	limit theo	PP internal structure
305	limit theo	PP internal structure
305	ud	PP attachment
305	limit	failure to recognize PERI
305	op	modal verbs attach at CORE
305	link	NP-CLAUSE subordination vs. CORE subordination
309	theo	neg at CLAUSE vs. CORE
309	limit theo	PP internal structure
309	ud	advcl $\rightarrow$ acl:relcl
309	op	modal verbs attach at CORE
309	limit	failure to recognize PERI
312	op	modal verbs attach at CORE
312	bug	neg determiner attached wrongly
312	limit	failure to recognize PERI
318	link	CORE coordination vs. CORE cosubordination
318	limit	failure to recognize PERI
324	theo	QP

324	theo	AP-PERI always attaches at CORE_N, not NUC_N
324	ud	discontinuous wh-PP (stranding) not parsed correctly
324	bug	failure to handle nmod:tmod as adverbial
324	theo	relative clause attaches at NUC_N, not CORE_N
325	theo	QP
325	theo	QP
325	theo	AP-PERI always attaches at CORE_N, not NUC_N
325	theo	AP-PERI always attaches at CORE_N, not NUC_N
325	limit	PP attaches too low in coordinated NP
325	theo	we don't distinguish between CONJ and CLM
326	limit theo	PP internal structure
326	op	modal verbs attach at CORE
326	theo	we don't distinguish between CONJ and CLM
326	bug	failure to handle elliptical conjunct
326	bug	failure to recognize PoDP
331	theo	single complex NUC instead of NUC cosubordination
331	acoli	's attached to root
336	bug	untensed auxiliary treated as OP-TNS
337	bug	failure to handle nmod:tmod as adverbial
348		
350		
350	bug	advcl as PP-PERI vs. CLAUSE
350	limit	failure to recognize PERI
350	theo	we don't distinguish between CONJ and CLM
350	bug	NP conjunction attaches to PP
350	bug	failure to recongize PrDP
350	link	CORE cosubordination vs. CORE under NUC
350	op	modal verbs attach at CORE
350	theo	AP-PERI always attaches at CORE_N, not NUC_N
350	theo	predicative adjective: simple NUC vs. AP
350	limit theo	PP internal structure

B Computing Tree Distance Using Bottom-up Replugging (BURP)

We describe BURP ("bottom-up replugging"), an algorithm that computes an edit script between two trees with identical spans, such as two different natural-language constituent parse trees over the same sentence. Potential applications include evaluating the performance of constituent parsers and estimating the annotator effort in a semi-automatic annotation scenario.

Similar metrics include tree-distance (Zhang and Shasha, 1989; Emms, 2008), EVALB (Collins, 1997), string-distance applied to tree linearizations (Roark, 2002), and the leaf-ancestor metric (Sampson and Babarczy, 2003). None of them explicitly models the possibility of re-attaching a subtree to a different node, and they thus tend to over-penalize attachment errors as every constituent containing a moved subtree is affected (Bangalore et al., 1998). Although Roark (2002) and Emms (2008) propose strategies that mitigate this, subtree re-attachment is still handled as pair of delete and insert operations, thus its cost cannot be freely chosen but is necessarily the sum of the two.

BURP differs from all these algorithms by trying to explicitly simulate the way human annotators using graphical annotation interface correct trees. We assume the following basic operations to be availalbe: relabeling a node, deleting an internal node (implicitly reattaching all its children to its parent), inserting a node below another node (so that children of the existing node become children of the new node), and moving a node (that has at least one sibling) to a different parent. Given these operations, one question to ask is what is the optimal set of operations to transform the source (or predicted) tree into the target (or gold) tree, given some cost for each operation (in the following, we assume that every operation has cost

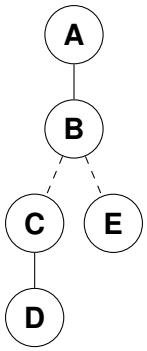


Figure 5: An example tree consisting of three maximal unary chains. Dashed edges indicate the boundaries between chains.

1, but they can easily be weighted differently). This is an NP-complete problem. Another, and perhaps more interesting question is how many operations human annotators need. We conjecture that BURP mimicks the human annotation process to some degree, and thus gives script lengths that correlate better with human annotator effort than other measures.

Sketch of the algorithm BURP transforms the source tree into the target tree in a bottom-up fashion, recreating smaller subtrees of the target tree in the source tree first and then moving on to larger ones until the root is reached and the whole tree transformed. To this end, the target tree is first divided up into maximal unary *chains*, as illustrated in Figure 5. To simplify the description, we will often refer to a chain as if all its nodes have been contracted into one, i.e. we say that in the example, chain AB has two children CD and E. We will also occasionally refer to a subtree and its root as the same entity if the context makes it clear. BURP does a post-order traversal of the chains in the target tree and at each chain transforms a part of the source tree into the subtree under that chain. All children of the chain have at this point already been recreated, and they are re-used, even if this is not guaranteed to give an optimal edit script. This is how BURP cuts the NP-complete problem down to a polynomial one. The local decisions, *viz.*, which part of the source tree to transform into the current target chain, however, are optimized for minimal local cost.

Definitions The *span* (or *yield*) of a tree is the set of its leaves. Note that we do not assume spans to be contiguous.

Inputs The inputs to BURP are a source tree T_1 and a target tree T_2 with identical spans.

Data Structures and Initialization While transforming T_1 into T_2, we will temporarily remove subtrees from T_1 and thus take it apart into multiple parts. We maintain a set P of these parts, which we initialize as $P := \{T_1\}$. We say that a node is "free" if it is the root of a tree in P.

The Traversal We do a post-order traversal of the chains in T_2, recreating the subtrees under them in as subtrees of trees in P. Thus, when we visit a chain, all of its children have already been recreated. Let p_2 be the currently visited target chain and C its recreated children in the trees in P. In the example in Figure 6, $p_2 = \mathsf{BDE}$ and $C = \{\mathsf{F}, \mathsf{G}, \mathsf{H}, \mathsf{I}, \mathsf{J}\}$. We then pick an *extended source chain* or *x-chain*, i.e., a path p_1 in some tree in P such that $p_1 = n_1 n_2 \ldots n_N$ with $N \geq 1, n_N \in C$. In the example, $p_1 = \mathsf{ABCF}$. The subtree under p_1 is then deterministically transformed into that under p_2 with the minimal number of operations and assuming that the subtrees under all $c \in C$ remain unchanged.[11] The transformation consists of the following steps:

[11]The reason for including one of the transformed children in p_1 is to allow for the case where the rest of the source chain is empty and all nodes in p_2 have to be inserted during the transformation. An empty source chain would not specify where to insert these nodes. Note also that $n_1 \ldots n_{N-1}$ need not be a unary chain (in the example, B has another child P); it will be transformed into one.

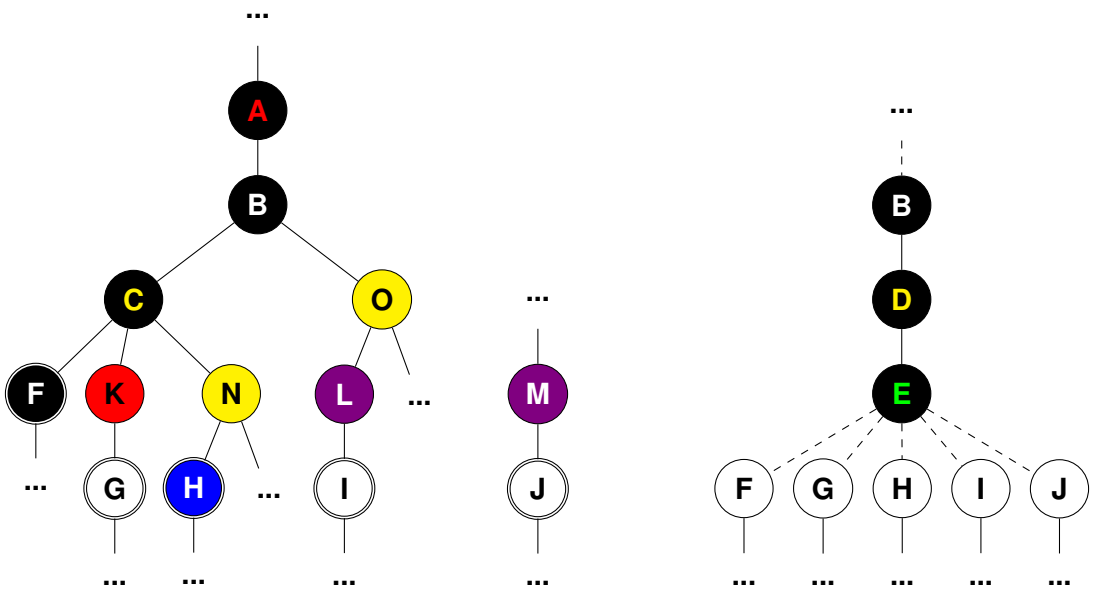


Figure 6: Example state of the algorithm, with two source tree fragments on the left and a target tree fragment on the right. Black nodes are parts of the target chain or source x-chain. Doubly circled nodes are already-recreated children in the source tree fragments. Yellow nodes will be freed, blue ones moved, red ones pruned, and violet ones moved and pruned.

1. **Move down.** For each topmost descendant of $n_1 \dots n_{N-2}$ whose span is a subset of the target span (meaning the span of p_2) but which is not dominated by n_{N-1}, move it to n_{N-1}. In the example, L is moved to C (cost 1).

2. **Free above.** For each child of $n_1 \dots n_{N-2}$ that is not in p_1, "free" it, i.e., remove it and add it to P as a free subtree. It will find its place is the transformed tree later. In the example, O is freed (cost 1).

3. **Edit chain.** Insert, delete, and relabel nodes in $n_1 \dots n_{N-1}$ so as to make the chain identical to p_2. The cost is the Levenshtein distance (Levenshtein, 1966) between both sequences.[12] In our example, A is deleted, C is relabeled D, and E is inserted (cost 3).

4. **Move up.** For each topmost descendant of n_{N-1} whose span is a subset of the target span but which is not a child of n_{N-1}, move it to n_{N-1}. In the example, H is moved to C (cost 1).

5. **Free below.** For each child of n_{N-1} whose span is not a subset of the target span, free it. In our example, N is freed (cost 1).

6. **Move in.** For each topmost node whose span is a subset of the target span and that is not yet a child of n_{N-1}, move it there. If that node is a root, there is no cost because the cost of moving was already incurred when the node was freed.[13] In the example, M is moved in (cost 1).

7. **Prune.** For every node between n_{N-1} and any $c \in C$, delete it. In the example, K, L, and M are deleted (cost 3).

Postcondition After visiting the root of T_2, P contains exactly one tree, which is identical to T_2.

[12] Our graphical annotation interface does not currently allow for inserting a node directly above another node in a unary chain if the latter has siblings. This could be taken into account by disallowing insertions at the beginning of the Levenshtein edit script.

[13] The cost is incurred early, by the freeing operation, not by the subsequent "moving in", so it can be attributed to the x-chain that necessitates the moving. Annotators often do not have a place where they can put removed subtrees temporarily; we assume that they will immediately move the subtree to the node where it will eventually end up.

Search For every visited target chain, we pick an x-chain that minimizes the local cost. The final edit chain and cost still depends on how ties between locally optimal x-chains are broken, and on the exact order of traversal. In our current implementation[14], the leaves are assumed to be ordered (as the words are in natural language), and the postorder traversal proceeds from left to right. Ties between x-chains are currently broken by preferring chains that are in more recently freed subtrees, further to the right in the tree, and longer. A closer approximation to the optimal edit script could be achieved, e.g., by randomizing this and doing multiple restarts.

[14] `https://github.com/texttheater/burp`