Graph-Based Annotation Engineering: Towards a Gold Corpus for Role and Reference Grammar

Christian Chiarcos
Applied Computational Linguistics Lab, Goethe University Frankfurt, Germany
http://acoli.informatik.uni-frankfurt.de/
chiarcos@informatik.uni-frankfurt.de

Christian Fäth
Applied Computational Linguistics Lab, Goethe University Frankfurt, Germany
faeth@em.uni-frankfurt.de

Abstract

This paper describes the application of annotation engineering techniques for the construction of a corpus for Role and Reference Grammar (RRG).

RRG is a semantics-oriented formalism for natural language syntax popular in comparative linguistics and linguistic typology, and predominantly applied for the description of non-European languages which are less-resourced in terms of natural language processing. Because of its cross-linguistic applicability and its conjoint treatment of syntax and semantics, RRG also represents a promising framework for research challenges within natural language processing. At the moment, however, these have not been explored as no RRG corpus data is publicly available. While RRG annotations cannot be easily derived from any single treebank in existence, we suggest that they can be reliably inferred *from the intersection* of syntactic and semantic annotations as represented by, for example, the Universal Dependencies (UD) and PropBank (PB), and we demonstrate this for the English Web Treebank, a 250,000 token corpus of various genres of English internet text. The resulting corpus is a gold corpus for future experiments in natural language processing in the sense that it is built on existing annotations which have been created manually.

A technical challenge in this context is to align UD and PB annotations, to integrate them in a coherent manner, and to distribute and to combine their information on RRG constituent and operator projections. For this purpose, we describe a framework for flexible and scalable annotation engineering based on flexible, unconstrained graph transformations of sentence graphs by means of SPARQL Update.

2012 ACM Subject Classification Computing methodologies → Language resources; Information systems → Semantic web description languages; Computing methodologies → Natural language processing; Computing methodologies → Lexical semantics

Keywords and phrases Role and Reference Grammar, NLP, Corpus, Semantic Web, LLOD, Syntax, Semantics

Digital Object Identifier 10.4230/OASIcs.LDK.2019.9

Category Short Paper

Supplement Material The software described in this paper are available under the Apache 2.0 license from https://github.com/acoli-repo/RRG. This includes build scripts for the data. We aim to provide the data under the same license as the annotations it is derived from (CC-BY-SA), but we are still in the process of copyright clearance for the original text.

Funding The research described in this paper was conducted in the context of the project *Linked Open Dictionaries* (LiODi, 2015-2020), funded by the German Ministry for Education and Research (BMBF), as well as the project *Specialised Information Service Linguistics* (Fachinformationsdienst Linguistik, funding period 2017-2019) funded by the German Research Foundation (DFG).

1 Introduction

Annotation engineering can be defined as the task to transform linguistic annotations from one or several specific source representations into representations of a different type or quality in an automated or semi-automated fashion.

The goal of annotation engineering is to produce high-quality annotations (gold data) for training state of the art tools in natural language processing. It is thus an essential aspect for the growth of the discipline beyond earlier annotation efforts, as it allows legacy resources created with great manual effort to be re-used as test and training data for a novel theoretical framework. The goal of annotation engineering is *not* to replace annotation tools, but rather to provide training data for them. As it posits particularly high standards of annotation quality, annotation engineering is to be done in a transparent, rule-based fashion rather than by machine learning. The outcome of annotation engineering is a resource, i.e., an annotated corpus or information extracted from it, so that transformation efficiency plays a considerably less important role than conversion quality. Typical examples for annotation engineering include, for example, transformations of annotations between different theoretical frameworks, e.g., and this will be illustrated here for the case of Role and Reference Grammar.

Annotation engineering differs from plain conversion in the sense that its output is typically qualitatively different or richer than the source annotations. This can be achieved, for example, by including additional knowledge sources, e.g., lexical resources, or additional sources of annotation. To a large extent, such resources are already available from the web of data, where specifications for the publication of (open) language resources are developed in the context of the Linguistic Linked Open Data (LLOD) cloud and associated W3C community and business groups.[1] In the last decade, a large number of language resources have been published in this context, in accordance with W3C standards and as Linked Data, and with their formal metadata registered at portals such as LingHub,[2] thus facilitating their usability and interoperability. The Linguistic Linked Open Data Cloud comprises corpora, dictionaries, resource metadata and terminology repositories and knowledge bases which are made interoperable through the use of shared vocabularies and ontologies such as GOLD, ISOcat, OLiA, lexvo and lexinfo. As of January 2019, over 100,000 resources covering more than 1,000 languages are listed on LingHub, a curated subset of open resources of these is the basis for the LLOD cloud diagram.

In order to facilitate the integration of such resources in annotation engineering workflows, with CoNLL-RDF [5], we developed an approach based on LOD standards, most notably RDF for representing annotations, and SPARQL Update[2] for their transformation. In conceptual terms, these allow to render, manipulate and create arbitrary linguistic annotations in the form of labeled directed multi-graphs – and, as established already by Bird and Liberman [1] –, they are thus capable of encoding *every* kind of linguistic annotation.

In this paper, we illustrate the application of our annotation engineering approach on the creation of a Role and Reference Grammar treebank. Aside from the conceptual challenge to derive a gold corpus from existing manual annotations, a technical challenge in this context is that RRG syntax cannot be derived from any common treebank formalism, but instead, it requires to create and to process the intersection of independent annotations for syntax and semantics.

[1] `http://linguistic-lod.org/llod-cloud`
[2] `http://linghub.org/`

2 Role and Reference Grammar

Role and Reference Grammar [8, RRG] is a theory of grammar developed by Robert D. Van Valin, Jr. and William A. Foley during their research of Austro-Asiatic and native American languages. Encountering various problems in applying established theories of grammar, they aimed to overcome the European bias in language description by devising a descriptive grammar formalism that integrates structural analyses with semantics and pragmatics.

RRG features several *projections*, i.e., interdependent levels of analyses all grounded in the surface expression of a particular sentence. The RRG *constituent projection* is a phrase structure representation of content elements (excluding operators), complemented by the *operator projection* that formalizes scope and attachment of function words, and the *focus projection* that identifies speech acts and information structures. Semantics has a different status, as (the constituent projection in) RRG syntax is designed to reflect frame semantics. In fact, argument linking has been a priority in the design of RRG: RRG formalizes a linking algorithm that connects semantic (macro) roles (ACTOR and UNDERGOER) in an underlying lexical inventory with the surface grammar and vice versa: The CORE of an utterance contains the semantic predicate and its core arguments, the PERIPHERY contains its (semantic) modifiers. Every CORE thus corresponds exactly to an instance of a verbal frame in the sense of FrameNet.[3] In this paper, we focus on constituent projection and operator projections in RRG as illustrated in Fig. 2.[4]

For reasons of space, we cannot explain the specifics of RRG here in detail. Yet, one aspect that is worth mentioning is that RRG postulates a direct relationship between functional operators and the nodes they modify, e.g., aspect applies to the semantic nucleus (NUC) of a clause, modality applies to its CORE, and tense to the CLAUSE itself. This has implications for juncture and nexus: If two verbs are connected by a paratactic relation in surface syntax, the scope of shared operators (and shared arguments) define whether this is a co(sub)ordination at the level of NUC, CORE, CLAUSE or SENTENCE. As a result, RRG constituent projections cannot be derived from any other syntax formalism but need to be adjusted to account for theory-specific constraints arising from the occurrence of function words and/or overlap in semantic arguments. While this raises problems for bootstrapping RRG syntax from existing annotations, it is an advantage in terms of expressivity, as RRG structure is designed to provide an appropriate representation of both frame semantics and operator scope.

Because of its semantics-based approach to syntax, RRG poses a promising framework for developing innovative applications in natural language understanding processing and natural language understanding. Furthermore, RRG ties in with the current trend towards intensified research of less-resource languages, as indeed, RRG is a popular theoretical framework for language documentation and linguistic typology. If RRG can be operationalized for one language, say, English, this paves the way for improved NLP support for languages to which RRG has been previously applied, e.g., languages from the Americas (Yaqui, Zoque, Lakota, etc.), the Pacific (Amele, Kankanaey, Amis, etc.), Asia (Zaza, Farsi, Tibetan, etc.), and to

[3] See `http://framenet.icsi.berkeley.edu/`. As for the relation between CORE and higher elements of syntactic analysis such as recursive COREs, CLAUSE, SENTENCE and *TEXT*, these are created for the purpose of juncture, nexus and extraposition, but all require a non-recursive CORE as their basis.

[4] This visualization has been created with TIGER Search [10] and follows its graphical design: Primary edges (black) represent the constituent projection, secondary edges (green) represent the operator projection, the target nodes of secondary edges would be shared in both projections.

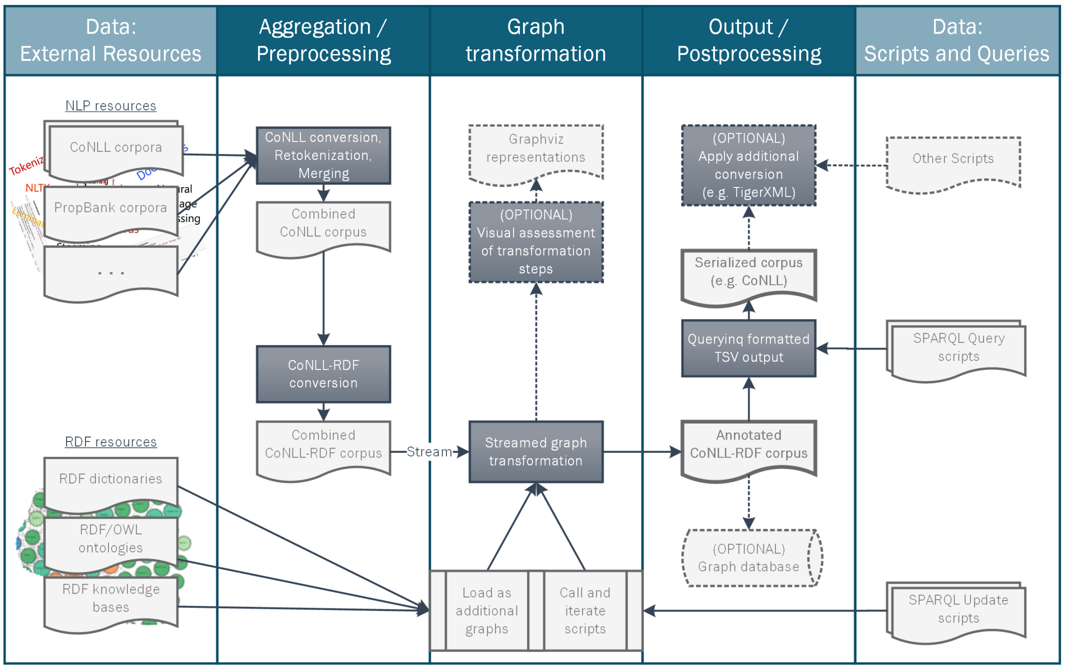
Data: External Resources
Aggregation / Preprocessing
Graph transformation
Output / Postprocessing
Data: Scripts and Queries
NLP resources
CoNLL corpora
PropBank corpora
. . .
CoNLL conversion, Retokenization, Merging
Combined CoNLL corpus
CoNLL-RDF conversion
Combined CoNLL-RDF corpus
Stream
RDF resources
RDF dictionaries
RDF/OWL ontologies
RDF knowledge bases
Graphviz representations
(OPTIONAL) Visual assessment of transformation steps
Streamed graph transformation
Load as additional graphs
Call and iterate scripts
(OPTIONAL) Apply additional conversion (e.g. TigerXML)
Serialized corpus (e.g. CoNLL)
Queryinq formatted TSV output
Annotated CoNLL-RDF corpus
(OPTIONAL) Graph database
Other Scripts
SPARQL Query scripts
SPARQL Update scripts

Figure 1 Annotation Engineering Workflow.

some extent Africa (Hausa, Gikuyu).

A fundamental problem in the practical application of state-of-the-art NLP techniques to RRG is, however, that no annotated data is currently available in order to assess potential challenges and benefits RRG may pose to state-of-the-art NLP techniques based on Machine Learning. Here, we address this deficit by creating an English RRG Treebank from manual annotations of syntax and frame semantics by means of annotation engineering.

3 Graph-based Annotation Engineering

The goal of annotation engineering is to produce high-quality annotations (gold data) for training and evaluating NLP tools. In our case, we ground RRG clausal syntax in existing manual annotations for semantic frames (PropBank, [15, PB]), and derive remaining aspects of constituent and operator projection from independent manual annotations for dependency syntax and morphosyntax provided as part of the Universal Dependencies [14, UD] corpora. PropBank and UD overlap in the English Web Treebank (EWT), a corpus of 250,000 words representing various genres of English internet text, and we adopt the English Web Treebank [17, EWT] as the basis for bootstrapping an RRG Treebank for English.

However, these formats bear structural limits which have to be worked around. This applies especially for back-reference in natural language resulting in separated phrases sharing the same head, thus violating the purely hierarchical tree structure. While this can be avoided by restructuring the tree or inserting NULL tokens (as in PTB) it produces challenges in data modeling which could easily be avoided by implementing a purely graph-based format which is able to host and integrate all data structures at the same time using labeled edges.

3.1 Annotation engineering with the ACoLi CoNLL libraries

The ACoLi CoNLL libraries [7] provide native support for the CoNLL-U format as provided by UD, and they support specific vocabulary extensions for semantic frames (SRL annotations) in CoNLL TSV formats, they are thus adopted here as the technical basis for annotation engineering. Building on that, we provide a generic, customizable workflow (Fig. 1) for the integration of heterogeneously annotated corpora into a common data structure and the induction of novel data structures. We employ SPARQL Update rules to perform advanced graph transformations, and the CoNLL-RDF API[5] supports both their aggregation and organization in files, as well as their sequential iteration.

CoNLL-RDF supports various output formats, most importantly tabular data in TSV or CoNLL, RDF in different serializations, a "canonical" Turtle representation that emulates the appearance of CoNLL-TSV formats, human-readable representations, and Dot/GraphViz plots. As exchange format for the RRG corpus, we provide its data (without diagnostic information) as TIGER/XML [11] using a specialized export functionality [6].

3.2 Integration of concurrent annotations

Resource integration can be performed on multiple levels. A particularly challenging aspect of annotation engineering is the combination of concurrent, independently performed annotations of the same text, by different groups using different tools and formats. Using converters bundled with the ACoLi CoNLL libraries or available from third parties, most annotation formats can be converted to a CoNLL format or a corresponding TSV representation. CoNLL-RDF supports the flexible handling of differences with respect to the order and naming of columns. Conflicting tokenizations (i.e., differences in the definition of rows) are handled by CoNLL Merge.[6] We merged UD files with the corresponding PropBank files via their shared part-of-speech columns (the PB skel files do not include the text), successfully merging 99.99% of the EWT tokens (254,564 of 254,593) in default mode. The remaining 29 mis-aligned tokens were manually corrected.[7] Merging resulted in a single TSV file holding all information from UD and PB, ideally suited for subsequent conjoint processing.

3.3 Graph creation

CoNLL-RDF allows us to render tabular data structures from CoNLL or related one-word-per-line TSV formats as an RDF graph. The `CoNLLStreamExtractor` retains the order of tokens, columns and sentences within a corpus but adds minimal additional overhead to make it RDF-compliant. The idea of the converter is to identify words by URIs, to define them as `nif:Words` and as being connected by `nif:nextWord`, and to use user-provided labels (say, `WORD`) to map every column to a property of the same name in the `conll:` namespace. Listing 1 is an example fragment for the analysis of the sentence *Where can I get morcillas in Tampa Bay* (EWT, dev/answers/20070404104007AAY1Chs_ans).

[5] `https://github.com/acoli-repo/conll-rdf`
[6] `https://github.com/acoli-repo/conll`
[7] CoNLL Merge also provides a *force* mode that enforces the tokenization of one file. Because of the minimal number of misalignments, however, this was not necessary here.

Listing 1 CoNLL-RDF canonical format.

```
:s1_1 a nif:Word; conll:WORD "where"; conll:HEAD :s1_4; ... ; nif:nextWord s1_2 .
...
:s1_4 a nif:Word; conll:WORD "get"; conll:HEAD :s1_0; ... ; nif:nextWord s1_5 .
...
:s1_6 a nif:Word; conll:WORD "in"; conll:HEAD :s1_4; ... ; nif:nextWord s1_7 .
:s1_7 a nif:Word; conll:WORD "tampa"; conll:HEAD :s1_8; ... ; nif:nextWord s1_8 .
:s1_8 a nif:Word; conll:WORD "bay"; conll:HEAD :s1_6; ... .
```

3.4 Graph transformation

The `CoNLLRDFUpdater` is designed for the stream processing of large corpora that cannot be held in memory: It reads a single sentence, creates an RDF graph for it, optionally adds context information (e.g., from preceding context or from a background knowledge graph loaded at initialization), applies SPARQL update transformations and spells out the results. Sentence-by-sentence processing minimizes memory usage and search space for the transformations, and in addition, the process is parallelized to improve run-time performance.

When processing corpora, we read the input from stdin, apply the `CoNLLStreamExtractor` to produce RDF graphs and the `CoNLLRDFUpdater` for graph transformation. Graph transformations are implemented with SPARQL Update rules, in that a sequence of SPARQL files can be provided as arguments for the `CoNLLRDFUpdater`. Each file contains a number of SPARQL update operations which are executed in their sequential order. In addition, numerical flags allow each file to be executed multiple times or until no further changes occur. The results are flushed through `stdout` while next sentence is read. Lookahead and lookback parameters allow cross-sentence analyses (e.g. of text coherence) while parallelization speeds up the process. Furthermore, native LOD and RDF resources can be easily integrated by either federated queries or preloading RDF dumps into separate named graphs making them available within the update scripts.

In annotation engineering and rule-based parsing continuous revision and reorganization of rules is a crucial aspect. By separating transformation operations (SPARQL) from the actual code (CoNLL-RDF classes, JAVA), the transformations become easily portable between the CoNLL-RDF environment and off-the-shelf triple stores as well as between different workflows that require the same functionality. SPARQL statements can thus be run independently, e.g., on a database snapshot and easily optimized on that basis. Additionally, this architecture improves reusability of modular scripts and encourages contribution in community projects.

4 Use Case: Building an RRG Treebank

Using our annotation engineering workflow, we implemented a rule-based conversion routine for transforming the UD and PropBank representation of the EWT corpora into an RRG representation.

In addition to EWT data, we also digitized and converted the complete body of English examples (429 examples, 3,766 tokens) drawn from [18, 19], using UD v.1-compliant annotations produced by the Stanford parser [3, manually corrected].[8] As these are lacking

[8] With respect to RRG annotations, we primarily follow the notational conventions of [18], in that we use the labels `ARG` and `NP`. This is upward-compatible, though: In more recent editions, the `ARG` would be just omitted (as it can be inferred from the underlying lexical inventory – which does not exist in our

semantic role annotations, heuristic rules have been devised to identify verbal predicates and their semantic arguments from the syntactic annotation alone. This extrapolation does not replace full-fledged PropBank annotations, so, where it failed, we provide explicit patterns for specific verbs and their grammatical roles as part of the transformation workflow in order to reproduce textbook examples (cf. Fig. 2.)

For representing RRG data structures, we define an RDF vocabulary for representing constituency projection and operator projection:[9] All RRG node types from the constituency projection are defined by concepts such as `rrg:NUC`, `rrg:CORE`, etc. The relations between them are formalized by two navigational properties, `rrg:has` (pointing from parent node to children), and `rrg:next` (connecting RRG nodes with the next following siblings).

The operator projection is represented by RDF properties such as `rrg:TNS` (tense) or `rrg:IF` (illocutionary force) which point from the words or phrases that evoke a particular operator to the RRG node that they are associated with (e.g., a `rrg:CLAUSE`). Listing 2 shows a (slightly simplified) example rule that creates a `rrg:NUC` from existing SRL annotations[10]

Listing 2 Graph transformation with SPARQL.

```
INSERT { _nuc a rrg:NUC; rrg:has ?pred. }
WHERE {  ?pred conll:SRL []. };
```

This rule reads the `SRL` column of the PropBank skel files that holds the disambiguated identifier of the semantic predicate (normally preceding the argument columns) and creates a syntactic nucleus if one is encountered.

4.1 Transformation steps

The transformation workflow consists of the following steps, each corresponding to a SPARQL Update file. The actual implementation of this workflow in CoNLL-RDF merely requires to provide this list of SPARQL Update files to the `CoNLLRDFUpdater` after `-update`:

constituents. Identify noun phrases, prepositional phrases, modifier (adverb, adjective) phrases.

clausal structure. For every (verbal) PB predicate, create clausal NUC, CORE, CLAUSE, SENTENCE together with arguments, periphery, PrCS and dislocation positions. Note that the resulting data structure is not a tree: Multiple COREs can share their arguments, etc.

operators. For every word, identify all NUC- (CORE-, CLAUSE-) level operators and to the NUC (CORE, CLAUSE) that comprises the UD head of the word. It is important here that this builds on the operator hierarchy specified by RRG as it guides constituent pruning during juncture assessment.

juncture and nexus. This is the most challenging and RRG-specific part of the conversion process:

1. Constituent pruning: Eliminate all SENTENCE (CLAUSE, CORE) nodes that feature neither a operator nor an argument.

case), and `NP` would be renamed to `RP`. Furthermore, we adopt a simplified handling of `ADJ` and `ADV` phrases for which a novel `MP` node has been introduced at a later point in time.

[9] `PREFIX rrg: <http://www.acsu.buffalo.edu/~rrgpage/rrg.html#syn_>`

[10] For identifying the newly created NUC element, this listing uses a blank node. In the actual implementation, we provide full URIs.

2. Co(sub)ordination: If a UD `conj` (similar for `ccomp`, `xcomp`, `parataxis`, etc.) holds between two unconnected NUCs (COREs, CLAUSEs), create NUC (etc.) coordination under the same CORE (CLAUSE, SENTENCE) parent. If two coordinated COREs (CLAUSEs) share an argument *in the same semantic role* and use the same (or no) CORE- (CLAUSE-) level operators, transform the coordination into CORE (CLAUSE) cosubordination by inserting a CORE (CLAUSE) node that contains both COREs (CLAUSEs).
3. Establish tree structures by eliminating shared arguments.
4. Subordination: If an UD `acl` (similar for `advcl`, etc.) holds between two unconnected tree fragments, attach the tree containing the dependent to the smallest suitable constituent that contains the UD head.
5. Sentence completion: For every tree fragment that does not contain a SENTENCE, create its RRG parent nodes up to the level of SENTENCE. Connect multiple SENTENCE nodes within an utterance by a TEXT node.

4.2 Evaluation

Beyond the textbook examples mentioned above, we are not aware of any source for RRG gold annotations for English, so that we currently have no basis for a quantitative extrinsic evaluation. Instead, we performed an intrinsic evaluation by means of two optional validation steps, provided in two SPARQL files:

Structural validation. Establishes tree structures. If a node has more than two parents, it will be disconnected and assigned a new `MPARENT` node as parent. If an utterance contains more than one partial tree, these will be joined under a new `FRAG` node. This includes any unattached word, but exceptions apply (e.g., RRG does not account for punctuation).

Pattern validation. The original RRG parsing algorithm defines parsing templates. As these have not been used for creating the parses, we adopt them for validating parses. Here, we implement each of these templates as WHERE conditions in SPARQL INSERT statements, with diagnostic information (e.g., provenance of this particular pattern in the textbook) inserted as an `rdfs:isDefinedBy` reference. The number of templates is relatively high, so that pattern validation is slow (and optional).

The RRG generation workflow was exclusively modeled on RRG text book examples and the EWT development set from the answers domain. We focused on answers as this portion is particularly challenging in that it often contains grammatical errors. This may include, for example, omissions of function words, insertion of incorrect function words when written by non-natives or in a careless fashion. Like most linguistic phenomena, errors, and annotation gaps follow a distribution with a long tail, we thus do not guarantee convertability for the entire EWT data, but we aimed for an RRG-compliant conversion of 90% of the corpus to provide a suitable starting point for developing NLP tools. For the remaining (up to) 10%, subsequent efforts for manual correction of this data are necessary.

Ultimately we were able to convert 98.4% (418/425) of dev/answers sentences into structurally valid RRG representations, resp. 92.6% (1828/1974) sentences from the EWT development set in general, and 91.2% (1880/2061) from the EWT test set.

We also performed pattern validation, with 74.9% (1479/1974) and 73.9% (1524/2061) structurally and pattern-valid sentences on development set and test set, respectively. It should be noted, however, that we only validated against patterns as explicitly found in the text books or necessary for text book examples, so that low number for pattern validation may less reflect invalid RRG parses than they reflect gaps in the pattern inventory.

Along with this publication, we prepare the release of the corpus under `https://github.com/acoli-repo/RRG`, with the aim to reach out to the RRG community to elicit feedback and bug reports as a source of extrinsic (qualitative) evaluation and further improvement.

5 Discussion

Graph-based or graph-assisted parsing has had a long history in natural language processing. In the late 1980s [16] proposed the usage of graphs instead of trees for representing syntactic structures. Over the past decades, new approaches emerged resulting in growing performance improvements.[9] showed how statistical parsers could be augmented by graph transformation. [13] gained similar improvements with post processing the results of data-driven dependency parsers. In semantics, knowledge bases are mostly represented as graphs leading to numerous parsing approaches, e.g. using staged query graph generation [20]. Since history on this field is very diverse, it is impossible to list all relevant publications. For further reading we recommend the overview provided by [12].

Neither of these graph-based approaches, however, does address the specific task pursued here, i.e., the combination of existing annotated corpora and their transformation into a completely new representation formalism in order to alleviate the generation of gold data. Instead, they try to improve existing annotations mostly by training neural networks to efficiently stack mathematical algorithms and general transformation rules. The goal of annotation engineering, however, is *not* to replace (nor to improve) annotation tools, but rather to provide training data for them. As it posits particularly high standards of annotation quality, annotation engineering is to be done in a transparent, rule-based fashion rather than by machine learning.

In both regards, high demand for quality and the need of human supervision, annotation engineering is comparable to traditional grammar engineering (i.e., rule-based parsing). On the one hand, it is a simpler task in the sense that it transforms existing annotations rather to create them from scratch, and in particular, it does not depend on the conjoint development of lexical resources along with the rules. In particular, annotation engineering differs from grammar engineering as it is concerned with the analysis of a *finite* set of symbols rather than with the analysis of an *infinite* set of symbols from a finite set of categories.

On the other hand, the data structures encountered in annotation engineering can be *much* more diverse than those encountered in grammar engineering: In symbolic parsing, input data is a plain sequence of tokens, whereas internal and output structures are implementation-specific. In annotation engineering, input data can carry *any* kind of annotation originating from multiple sources. In comparison to grammar engineering, unrestricted annotation engineering is thus less challenging in terms of coverage, but more challenging in terms of diversity.

This paper showed that graph-based annotation engineering does have a place in NLP and that existing technologies developed in the context of the Linked Open Data community can be applied for the purpose. It does not replace machine learning, but rather serves as a technique for generating gold data for underresourced languages or annotation schemes.

Upon copyright clearance for the contained text, we will provide the corpus in three editions: Development edition (using the internal RDF vocabulary), an OWL release version in RDF, using the POWLA [4] vocabulary for generic linguistic annotations, and an release version in TIGER/XML [11] for further processing in conventional corpus tools.

References

1 Steven Bird and Mark Liberman. Annotation Graphs as a Framework for Multidimensional Linguistic Data Analysis. In Marilyn Walker, editor, *Towards Standards and Tools for Discourse Tagging*, Maryland, USA, June 1999. Association for Computational Linguistics. URL: `http://aclweb.org/anthology/W99-0301`.

2 Carlos Buil Aranda, Olivier Corby, Souripriya Das, Lee Feigenbaum, Paula Gearon, Birte Glimm, Steve Harris, Sandro Hawke, Ivan Herman, Nicholas Humfrey, Nico Michaelis, Chimezie Ogbuji, Matthew Perry, Alexandre Passant, Axel Polleres, Eric Prud'hommeaux, Andy Seaborne, and Gregory Todd Williams. SPARQL 1.1 Overview. `https://www.w3.org/TR/sparql11-overview`, 2013.

3 Danqi Chen and Christopher D Manning. A Fast and Accurate Dependency Parser using Neural Networks. In *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), Doha, Qatar, 2014. Association for Computational Linguistics (ACL)*, 2014.

4 Christian Chiarcos. POWLA: Modeling Linguistic Corpora in OWL/DL. In Elena Simperl, Philipp Cimiano, Axel Polleres, Oscar Corcho, and Valentina Presutti, editors, *The Semantic Web: Research and Applications*, pages 225–239, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

5 Christian Chiarcos and Christian Fäth. CoNLL-RDF: Linked Corpora Done in an NLP-Friendly Way. In *Language, Data, and Knowledge - First International Conference, LDK 2017, Galway, Ireland, June 19-20, 2017, Proceedings*, pages 74–88, 2017. `doi:10.1007/978-3-319-59888-8_6`.

6 Christian Chiarcos, Benjamin Kosmehl, Christian Fäth, and Maria Sukhareva. Analyzing Middle High German Syntax with RDF and SPARQL. In *In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, 2018. European Language Resources Association (ELRA)*, 2018. URL: `http://www.lrec-conf.org/lrec2018`.

7 Christian Chiarcos and Niko Schenk. The ACoLi CoNLL Libraries: Beyond Tab-Separated Values. In *In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, 2018. European Language Resources Association (ELRA)*, May 2018.

8 William A. Foley and Jr. Robert D. Van Valin. Role and Reference Grammar. In E.A. Moravscik and J.A. Wirth, editors, *Current approaches to syntax*, pages 329–352. Academic Press, New York, 1980.

9 Dan Klein and Christopher D. Manning. Accurate Unlexicalized Parsing. In *In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003), Sapporo, Japan, 2003. Association for Computational Linguistics (ACL)*, 2003.

10 Wolfgang Lezius. TigerSearch – Ein Suchwerkzeug für Baumbanken. In *Proceedings of the 6. Konferenz zur Verarbeitung natürlicher Sprache (6th Conference on Natural Language Processing, KONVENS 2002), Saarbrücken, Germany*, 2002.

11 Andreas Mengel and Wolfgang Lezius. An XML-based Representation Format for Syntactically Annotated Corpora. In *In Proceedings of the Second International Conference on Language Resources and Evaluation (LREC 2000), Athens, Greece, 2000. European Language Resources Association (ELRA)*, 2000.

12 Vivi Nastase, Rada Mihalcea, and Dragomir R. Radev. A survey of graphs in natural language processing. *Natural Language Engineering*, 21(5):665?698, 2015. `doi:10.1017/S1351324915000340`.

13 Jens Nilsson, Joakim Nivre, and Johan Hall. Graph Transformations in Data-Driven Dependency Parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, January 2006. `doi:10.3115/1220175.1220208`.

14 Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut

Tsarfaty, and Daniel Zeman. Universal Dependencies v1: A Multilingual Treebank Collection. In *In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), Portorož, Solvenia, 2016. European Language Resources Association (ELRA)*, May 2016.

15 Tim O'Gorman, Sameer Pradhan, Martha Palmer, Julia Bonn, Kathryn Conger, and James Gung. The New Propbank: Aligning Propbank with AMR through POS Unification. In *In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, 2018. European Language Resources Association (ELRA)*, 2018.

16 Jungyun Seo and Robert F. Simmons. Syntactic Graphs: A Representation for the Union of All Ambiguous Parse Trees. *Computational Linguistics*, 15(1):19–32, March 1989. URL: `http://dl.acm.org/citation.cfm?id=68960.68962`.

17 Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. A Gold Standard Dependency Corpus for English. In *In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014), Reykjavik, Iceland, 2014. European Language Resources Association (ELRA)*, 2014.

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

19 Robert D Van Valin Jr. *Exploring the syntax-semantics interface.* Cambridge University Press, 2005.

20 Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In *In Proceedings of the Joint Conference of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL) and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (AFNLP), Beijing, China, 2015. Association for Computational Linguistics (ACL)*, pages 1321–1331, January 2015. `doi:10.3115/v1/P15-1128`.

A RRG Example

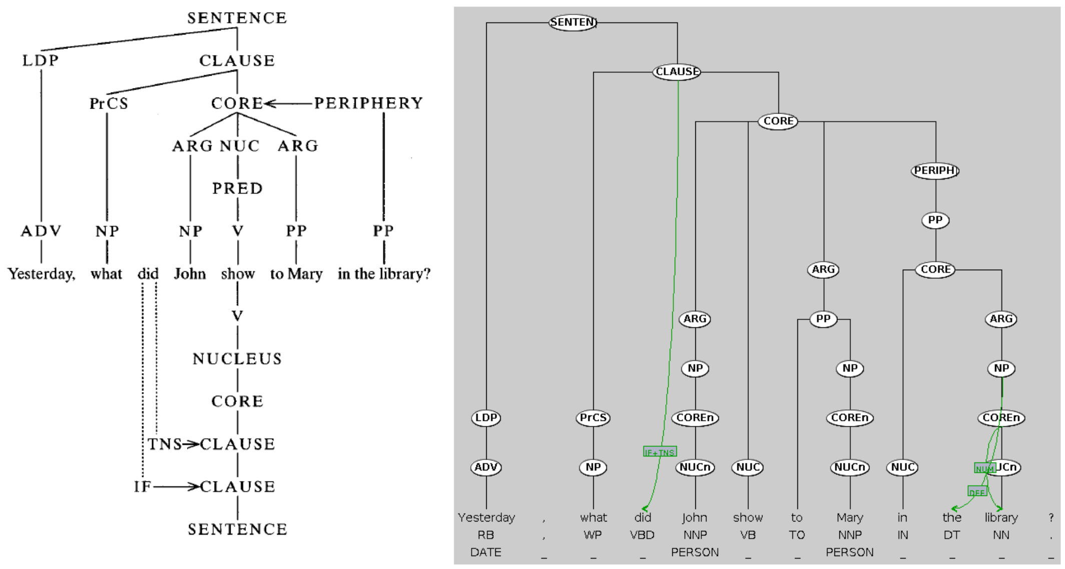

Figure 2 RRG Textbook Example(left, [18, p. 50]) compared to Synpathy rendering (right).