

Semi-Automated Labeling of Requirement Datasets for Relation Extraction

Jeremias Bohn

Technical University of Munich, Germany
jeremias.bohn@tum.de

Jannik Fischbach

Qualicen GmbH, Germany
jannik.fischbach@qualicen.de

Martin Schmitt

Center for Information and Language Processing
LMU Munich, Germany
martin@cis.lmu.de

Hinrich Schütze

Center for Information and Language Processing University of Cologne, Germany
LMU Munich, Germany
inquiries@cislmu.org

Andreas Vogelsang

University of Cologne, Germany
vogelsang@cis.uni-koeln.de

Abstract

Creating datasets manually by human annotators is a laborious task that can lead to biased and inhomogeneous labels. We propose a flexible, semi-automatic framework for labeling data for relation extraction. Furthermore, we provide a dataset of preprocessed sentences from the requirements engineering domain, including a set of automatically created as well as hand-crafted labels. In our case study, we compare the human and automatic labels and show that there is a substantial overlap between both annotations.

1 Introduction

While recent advances in Natural Language Processing have yielded high-quality language models such as BERT (Devlin et al., 2019), GPT-3 (Brown et al., 2020) and ELECTRA (Clark et al., 2020) which are able to continue sentences, fill in masked words and correctly parse human language, using these models for most use-case scenarios still requires them to be trained on a down-stream task using labeled data. For some tasks, e.g. sentiment analysis of reviews, creating datasets is relatively easy as large databases with annotations already exist (such as the IMDb movie review dataset (Maas et al., 2011)). However, training a model on niche tasks often demands hand-crafting new datasets from spread-out documents. This is usually done by humans who collect, preprocess, and annotate sentences which is a laborious task and can result in biased and/or inhomogeneous labeling, e.g. if annotation instructions were not understood correctly

or left room for subjective interpretation. This becomes especially apparent if multiple, non-expert individuals are involved in this process.

In requirements engineering, we usually work with large documents written in natural language (Mich et al., 2004; Kassab et al., 2014) which describe the specifications of a software project, usually classified as either functional requirements, specifying what functionality the system should provide, and non-functional requirements, specifying in what way the system should implement those functions. However, these documents are often updated during the life cycle of the project and span up to multiple hundreds of pages, depending on the project size. Keeping track of all the changes and maintaining the software based on the requirement document can soon become a challenge (Fischbach et al., 2020) which is why an automatic conversion to, e.g., UML diagrams can come in handy. To do so, it is necessary to parse the relations between entities from the written text into a structured format, thus creating a comparable corpus of requirements in natural language and the same relation in a formal language.

In this paper, we propose a semi-automatic approach that, given a clean, grammatically correct sentence stating a software requirement, outputs a labeling corresponding to the relation the requirement describes based on a small set of pre-defined rules of word dependency relations. This should reduce human bias manifesting in labels as the annotator does not actively choose the labels for each word anymore but instead defines abstract rules

which provide for homogeneous, deterministic labeling and reduce the amount of labor for creating such datasets. This automatically annotated data can then be used for training a more powerful model, as shown by [Schmitt et al. \(2020\)](#).

We summarize our main contributions as follows:

- We provide a high-quality, preprocessed dataset of 2,093 requirement sentences together with 1,848 automatically created labels and another 199 manually created labels for a subset of the automatically labeled sentences as a resource for further research projects.
- We provide a flexible, semi-automatic framework for data annotation of the relation extraction domain based on dependency parsing and pattern matching.
- We conduct a case study on the said framework on requirement document sentences, showing its annotation results are matching those of humans to a substantial degree.

2 Related Work

[Gamallo et al. \(2012\)](#) propose a simple Open Information Extraction system based on dependency parse trees. The algorithm extracts triples with two arguments and a sentence part relating those. However, the patterns are not very sophisticated and put a large part of the sentence into the relation. Hence, this approach is not suitable for our use case as we would eventually like to generate object diagrams from the relations we extracted. [Erkan et al. \(2007\)](#) use dependency parse trees to extract relations between proteins from sentences. They do so by classifying whether a sentence, given a dependency tree, describes a relation between any pair of proteins occurring in the sentence using semi-supervised harmonic functions and support vector machines. However, their entities (the protein names) are already annotated which is not the case if we only have the raw sentences as in our approach. [Mausam et al. \(2012\)](#) use dependency trees and a labeled bootstrap dataset to automatically generate patterns for information extraction, unlike our approach which does not require to annotate any data manually but instead to produce patterns. While this approach might be able to extract simple triples well, one needs either a larger annotated dataset, defeating the purpose of our work, or the patterns might not generalize well, thus being

unsuitable for constructing a qualitative annotated corpus. [Reddy et al. \(2016\)](#) propose an algorithm to automatically extract logical expressions from dependency parse trees for question answering. These were then converted into a graph indicating the relations between the named entities in the sentence by applying semantic parsing. However, this approach always converts the entire sentence into a graph and may include information that is irrelevant for a dataset that is to be generated. [Inago et al. \(2019\)](#) use a rule-based approach on dependency trees to process natural language car parking instructions with decision trees for automated driving systems. Unlike our data (or most datasets in general), sentences of the application domain are very short and similar in structure. While our approach could be effectively converted into a decision tree, it is easier to construct rules with our pattern engine for more complex data.

3 Corpus Creation

3.1 Dataset

For our dataset, we use 19 publicly available requirement documents in the English language from the PURE dataset ([Ferrari et al., 2017](#)), with a large topical variety, including governmental institution software in military and scientific fields, inventory management systems and video games. All documents are provided in .PDF, .HTML or .DOC format. From these, we manually extracted 2,104 requirement sentences (1,639 functional, 465 non-functional requirements).

3.2 Preprocessing

As we want to automatically dependency parse our sentences, we have to ensure that all input to the model is grammatically and orthographically sound. We also have to ensure that any unnecessary information is removed to not confuse the parser. Therefore, we manually applied the following formatting operations to each sentence during data extraction:

- Splitting of enumerations into multiple sentences, adjusting words if necessary to make the sentence sound (e.g., nounification of verbs); e.g., "The system has to include a) [...] b) [...] c) [...]" becomes 3 sentences, each including exactly one of the requirements
- Removal of extra inter-punctuation (additional spaces, dots, commas, etc.)

- Removal of references to sections, tables, figures, or other requirements of the document as they are not relevant for extracting the relation of the sentence itself
- Removal of abbreviations after written-out expressions (e.g., in "automated teller machine (ATM)", the "(ATM)" is dropped)
- Removal of requirement reference numbers
- Correction of spelling mistakes where obvious
- Adding of dots at the end of each sentence if missing
- Changing the first letter of a sentence to upper case if it is not yet
- Removal of quotation marks around pseudo-correct terms (e.g., 'the "processor" will [...] becomes 'the processor will [...]'')
- Removal of explicit explanations of what is included in some term (e.g., "errors of either kind, i.e. hardware and software, [...]")
- Lower-casing of words if they are not abbreviations (e.g., "NOT" becomes "not")
- Remove brackets around additional plural 's' (e.g., "socket(s)" becomes "sockets")
- Exchanging "/" with "and" or "or" where applicable and possible given the context (e.g. "The system should support adding/deleting files" becomes "The system should support adding and deleting files")
- Unification of the possessive 's' preceding symbols ("'" and "''" are changed to """)
- Removal of duplicate sentences (11 in total)

After these preprocessing steps, the average sentence length is 19.87 words, the maximum is 69 words and the minimum 4 words.

3.3 Labeling

These final 2,093 sentences (1,628 functional, 465 non-functional requirements) are parsed to extract dependencies using the Neural Adobe-UCSD Parser (Mrini et al., 2020) which achieved state-of-the-art performance on the Penn Treebank dataset (Marcus et al., 1993). Based on these dependencies, we handcraft a total of 102 patterns to label 91.03%

of the functional and 78.71% of the non-functional sentences without any further human interaction. Each pattern is a sequence of triples (l, dp, c) where l is a label, dp a sequence of dependency labels forming a path downwards a dependency tree and c a Boolean value indicating whether all children (direct and indirect) should be left out from labeling or not. Each sequence applies all or a subset of the following entity tags to the sentences:

- `ent1`: The main entity of the requirement. Either the acting component or the component on which a constraint is applied (if there is no second entity)
- `rel`: The relation/action of the requirement.
- `ent2`: The passive entity of the requirement. Either the component on which an action is performed or which is involved in the action passively
- `cond`: Any modifier of the requirement. Can further specify the requirement or put conditions on it how or when it will be applied.

An excerpt of automatic annotations can be found in Table 1. Each pattern is applied using tree traversal: for each label that is to be applied, a sequence of dependency labels (optionally with modifiers) is given, starting at the root. The algorithm checks whether the current nodes have any direct children connected to them with the current dependency label of the sequence. If so, we check whether these children have children connected to them with the next label in the sequence. If not, the pattern fitting is stopped and no labeling is applied to the sentence. If we reach the end of the sequence, the final node is labeled with the given label and, depending on a parameter, all of its children, too. A simple example can be found in Table 2, row 1. Dependency labels can include modifiers to allow for more complex patterns:

- Starting with `!`, the pattern matching will remove any node that has one or more children with the given dependency label. Thus, no step downwards the tree is taken
- Followed by `=[placeholder]` where `[placeholder]` is any word, only those nodes are considered where the label is the given label and the actual word of the node is specified by `[placeholder]`

Sentence
While flying two MAE AVs Beyond Line Of Sight _{cond} , the TCS _{ent1} shall provide _{rel} full control functionality _{ent2} of each AV _{cond} . NPAC SMS _{ent1} shall default _{rel} the EDR Indicator _{ent2} to False _{cond} . A bulk entry _{ent1} can be used to add _{rel} many assets _{ent2} .
The HATS-GUI _{ent1} shall interact with the Host OS to compare _{rel} time stamps _{ent2} for files _{cond} . The BE _{ent1} shall be able to apply _{rel} corrections _{ent2} based on state count and/or quantizer power measurement data _{cond} .

Table 1: Examples of Labeling

- . . . lets us traverse back to the parent of the current node. This allows us to check nodes for their existence without including them in the actual labeling

A selection of patterns used can be found in Table 2. In our setting, one sentence usually holds one relation, however, this is not the case for conjunctions of multiple main clauses or instructions. Due to current limitations of our engine (see Section 6), the relation of the first main clause is always chosen, however, this depends on the pattern design. Even though we only use requirements written in English, a large portion of the rules could be applied to data in different languages as the Universal Dependencies (Schuster and Manning, 2016) rely on the concept of primacy of content, allowing for very similar dependency trees. However, patterns explicitly using keywords may not generalize well for other languages. The code for the labeling task as well as the labeled data can be found on GitHub¹.

4 Evaluation

Given our automatically labeled data, we evaluate the quality of the labels by comparing its output to human annotations. To do so, we randomly sample 199 sentences (10.77%) from the 1848 sentences which were automatically labeled. Two of the authors then annotated these sentences manually. The annotators were given the descriptions of each label type, but had no access to the actual labeling from the algorithm. Annotators collaboratively labeled the data, discussing the labeling for each sentence and agreeing upon a single valid labeling. We then calculate inter-rater reliability with the Cohen’s κ between the human annotators and the automatic annotator, once over all labels and once as average inter-reliability per sentence (i.e., we calculate one Cohen’s κ score per sentence

¹<https://github.com/JeremiasBohn/RequirementRelationExtractor>

and average over all sentences –this considers each sentence equally while the overall score puts more weight on longer sentences). The results can be found in Table 3. While the overall score puts more weight on long sentences, the sentence average provides us an approximation of the reliability of our automatic annotator for any sentence. According to the taxonomy of Landis and Koch (Landis and Koch, 1977), the per sentence average κ value indicates a substantial inter-annotator agreement, the overall κ a moderate agreement. While the main acting entity is extracted very well with almost perfect agreement according to Landis and Koch, extracting relational modifiers proves to be the hardest with only moderate agreement between our automatic approach and the human annotators. This is mostly due to the nature of the label itself, spanning a large variety of modifiers from conditions to entities not involved in the relation itself. While one could split the `cond` label into multiple different labels, this would increase the number of patterns required a lot. Alternatively, one might reduce the coverage of the labeling in general but we focused on including as much information as possible. The relatively low score for `ent2` mainly arises from sentences containing multiple relations where many words describe a passive entity for other relations than the one of the main sentence. Our approach currently is not able to effectively extract multiple relations from a single sentence yet. This is also the reason why the score `rel` is lower than the one for `ent1`.

5 Limitations

While our approach works well for requirements documents - after all, relations between software entities and modifications of these relations can be extracted well by syntactically parsing the sentence structure - this does not apply to word labels which require a semantic understanding of the input. For example, if we were to create labels for Named Entity Recognition, our algorithm would fail as

Pattern	Description
<pre>(‘rel’, [‘root’], True) (‘ent1’, [‘root’, ‘nsubj’], False) (‘ent2’, [‘root’, ‘dobj’], False) (‘cond’, [‘root’, ‘advcl’], False)</pre>	Simple pattern, sets the root of the sentence as the relation (only this single word), the entire nominal subject as the acting entity, the entire direct object as the passive entity. An adverbial clause is treated as a relation modifier.
<pre>(‘rel’, [‘root=capable’, ‘prep=of’, ‘pcomp’], True) (‘ent1’, [‘root’, ‘nsubj’], False) (‘ent2’, [‘root’, ‘prep=of’, ‘pcomp’, ‘prep=in’, ‘pobj’], False) (‘cond’, [‘root’, ‘advcl’], False)</pre>	Catches phrases like "The system should be capable of [...]" and searches for the passive entity in the prepositional object of the prepositional clause starting with "in".
<pre>(‘rel’, [‘root’, ‘!dobj’], True) (‘ent1’, [‘root’, ‘nsubjpass’], False) (‘cond’, [‘root’, ‘prep=in’, ‘pobj=case’, ‘.’], False) (‘cond’, [‘root’, ‘advmod’], False)</pre>	Pattern is only applied if the sentence has no direct object (which could serve as the passive entity). Prepositional sentences starting with "in case" are labeled as requirement modifier (we have to traverse the tree upwards again to include the 'in' as well).

Table 2: Examples of Patterns

Labels considered	Sentence Avg.	Overall
All labels	0.632	0.576
rel only	0.790	0.720
ent1 only	0.855	0.822
ent2 only	0.619	0.561
cond only	0.532	0.543

Table 3: Cohen’s Kappa Results

it is not possible to find syntactic rules to distinguish between, e.g., an organization and a person. Also, the algorithm fails in some cases if either rules are not specific enough or the dependency parser mistakenly adds dependencies between sentence parts where there is no dependency between them. The latter may especially occur frequently if the sentences were not preprocessed well which is why our algorithm is not suitable as a classifier in general (if we, on the other hand, use our data as training input for a Transformer model (Vaswani et al., 2017), it may overcome these strict syntactic requirements and generalize better on real-world data).

6 Conclusion & Outlook

In this paper, we present a novel approach for data labeling which allows users to annotate sentences for relation extraction within a shorter time period compared to manual annotation while at the same time having a consistent labeling scheme for the entire dataset. Our approach exploits syntactic features which are the integral foundation of most relation extraction tasks.

For the future, it would be helpful to implement an automatic extraction of requirement sentences by, e.g., training a classifier to identify relevant sen-

tences in plain text or .PDF documents as well as a semi-automatic approach with human validation for preprocessing sentences into grammatically and orthographically sound ones. We plan on extending the pattern engine our algorithm relies on, e.g., allowing for recursive patterns to parse nested sentences and to extract multiple relations from one sentence as well as optional pattern parts to reduce redundancy (e.g., a sentence where the active entity is the nominal subject, the relation the dependency tree root and the passive entity the direct object may have a relation modifier in an adverbial clause. As of the current state, this requires two patterns (exponentially increasing with the number of optional dependencies) while with a pattern where this adverbial clause is considered optional, we only need a single pattern).

References

- Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.
- Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. **BERT: Pre-training of deep bidirectional transformers for language understanding**. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

- Güneş Erkan, Arzucan Özgür, and Dragomir R. Radev. 2007. [Semi-supervised classification for extracting protein interaction sentences using dependency parsing](#). In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 228–237, Prague, Czech Republic. Association for Computational Linguistics.
- Alessio Ferrari, Giorgio Ortonzo Spagnolo, and Stefania Gnesi. 2017. [Pure: A dataset of public requirements documents](#). In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 502–505.
- Jannik Fischbach, Henning Femmer, Daniel Mendez, Davide Fucci, and Andreas Vogelsang. 2020. [What makes agile test artifacts useful? an activity-based quality model from a practitioners’ perspective](#). In *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), ESEM ’20*, New York, NY, USA. Association for Computing Machinery.
- Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. 2012. [Dependency-based open information extraction](#). In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, pages 10–18, Avignon, France. Association for Computational Linguistics.
- Akari Inago, Hiroshi Tsukahara, and Ichiro Kobayashi. 2019. Parsing parking instructions for self-driving cars into spatial semantic descriptions. *J. Comput.*, 14(5):328–338.
- Mohamad Kassab, Colin Neill, and Phillip Laplante. 2014. State of practice in requirements engineering: contemporary data. *Innovations in Systems and Software Engineering*.
- J. Richard Landis and Gary G. Koch. 1977. [The measurement of observer agreement for categorical data](#). *Biometrics*, 33(1):159–174.
- Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. [Learning word vectors for sentiment analysis](#). In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.
- Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. [Building a large annotated corpus of English: The Penn Treebank](#). *Computational Linguistics*, 19(2):313–330.
- Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. [Open language learning for information extraction](#). In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534, Jeju Island, Korea. Association for Computational Linguistics.
- Luisa Mich, Mariangela Franch, and Pierluigi Novi Inverardi. 2004. Market research for requirements analysis using linguistic tools. *Requirements Engineering*.
- Khalil Mrini, Franck Deroncourt, Quan Hung Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. 2020. [Rethinking self-attention: Towards interpretability in neural parsing](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 731–742, Online. Association for Computational Linguistics.
- Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. [Transforming dependency structures to logical forms for semantic parsing](#). *Transactions of the Association for Computational Linguistics*, 4:127–140.
- Martin Schmitt, Sahand Sharifzadeh, Volker Tresp, and Hinrich Schütze. 2020. [An unsupervised joint system for text generation from knowledge graphs and semantic parsing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7117–7130, Online. Association for Computational Linguistics.
- Sebastian Schuster and Christopher D. Manning. 2016. [Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks](#). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, pages 2371–2378, Portorož, Slovenia. European Language Resources Association (ELRA).
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in neural information processing systems*, pages 5998–6008.