<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper presented at *7th International Workshop on Artificial Intelligence and Requirements Engineering, AIRE 2020, Zurich, Switzerland, 1 September 2020*.

Citation for the original published paper:

Frattini, J. (2020)

CEREC: Causality Extraction from Requirements Artifacts

In: *Proceedings - 7th International Workshop on Artificial Intelligence and Requirements Engineering, AIRE 2020*, 9233006 (pp. 79-82). Institute of Electrical and Electronics Engineers Inc.

<https://doi.org/10.1109/AIRE51212.2020.00018>

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:bth-20837>

CEREC: Causality Extraction from Requirements Artifacts

Julian Frattini

Department of Software Engineering
Blekinge Institute of Technology
37141 Karlskrona, Sweden
Email: julian.frattini@bth.se

Abstract—The cause-effect recognition (CEREC) system provides an API for causality extraction tailored to the requirements engineering context. The library is written in Java and is released under the MIT open source license. In this paper, the underlying algorithm is described, and a demonstration of the active learning component for causality extraction is outlined. The results are promising and strengthen the confidence in exploring automation approaches for model-based testing.

I. INTRODUCTION

Causality extraction has excellent use in many fields of research [1], as causal sentences clearly describe a semantic relationship between two events where one event, called the cause, triggers another event, called the effect. Extracting this relationship is, for example, used in the medical domain to gather causal relations between symptoms and diseases and deriving insights for diagnostics [2], for question answering [3], and in the economic domain, where causal relations can be used to unravel insight on stock market movements [4].

The domain of requirements engineering is eligible for causality extraction for three reasons: (1) requirements are predominantly written in natural language [5], (2) formalized requirements are reusable for other artifacts like test cases [6], and (3) causal relations in requirements convey important conditions, which explicitly define the described system. Existing approaches to causality extraction fail to translate into requirements engineering for two reasons: existing approaches are (1) based on semantic causality, while relations described in requirements engineering are commonly not of semantic nature, and (2) they only extract single words as causal events, which omits important information necessary to reuse the extracted causality for further processing [7]. Our approach on automatic causality extraction is therefore distinct from the related work in being tailored specifically to the context of requirements engineering.

The CEREC library provides a syntactic approach to causality extraction, which is applicable to requirements engineering and can be integrated into a model-based testing pipeline visualized in Figure 1. The machine learning core of the library is realized with the DKPro framework using external OpenNLP libraries. A separate tool demonstrates the active learning part of the library.

Fig. 1. Model-based testing pipeline transforming a NL requirement into a test case

II. CAUSALITY EXTRACTION ALGORITHM

A cause-effect graph can be automatically extracted from a natural language sentence if a sentence of a similar grammatical structure has already been discovered. The algorithm achieves this by establishing *causality patterns* for each grammatical equivalence class utilizing a machine learning algorithm rather than static linguistic rules. A causality pattern connects the syntactic structure of a sentence to an extraction algorithm. This makes use of the circumstance that grammatically similar causal sentences have their cause- and effect-expression located at similar positions within the sentence [8]. The syntactic structure of a sentence is retrieved utilizing a natural language processing (NLP) pipeline, which formalizes a sentence into a tree-like structure, called the constituency structure. Sentences with a similar constituency structure are grammatically similar and likely to have their cause- and effect-expression located in a similar position within the constituency structure. The *extraction algorithm* associated with the causality pattern selects and retrieves the expressions that convey the causal relations of the sentence.

The library provides an API covering two relevant interactions with the CEREC system: testing and training. While training, the CEREC system is provided with a sentence and—if the sentence is causal—a cause-effect graph conveying the sentence causality. With this sample, the machine learning algorithm can establish a new causality pattern, hence covering the sentence grammatical equivalence class in the future. While testing, the system determines whether a given sentence matches one of the established patterns. If it does, then a cause-effect graph is extracted from the sentence and returned to the user of the library.

The machine learning algorithm can be trained beforehand via batch learning. As the system can be tailored to specific requirements artifacts, writing styles, and use of linguistic patterns, it also offers online learning. An agent using the system

for online learning acts as the oracle in an active learning process for the algorithm by manually generating the cause-effect graph, which conveys a causal sentence's causality. The agent is improving the CEREC system's causality extraction by establishing new or improving existing causality patterns.

III. TOOL

The CEREC and its APIs are intended to be used as a library in a larger pipeline—e.g., for automatically generating test cases in a model-based testing environment, as shown in Figure 1. However, a simple graphical user interface, shown in Figure 2, was built around the CEREC API to showcase the active learning part of the system¹.

A sentence can be entered in section A of the tool and analyzed. Selecting *Analyze* will apply the CEREC testing operation to the sentence. The result of the pattern matching process is shown in section B, where the systems classification of the sentence and—if the sentence was classified as causal—the extracted cause-effect-graph is shown. If the prediction of CEREC is incorrect, the user can manually correct the system decision. First, the general classification of causal or non-causal can be corrected by selecting *Incorrect* in the result row. If a non-causal sentence was mistakenly detected to be causal, the system will adjust the erroneously compliant pattern to fix the acceptance. If a causal sentence was not compliant to any pattern, the user can manually provide the cause-effect-graph and make the system to actively learn a new grammatical equivalence class by entering *Confirm*. If the sentence is determined to be causal, the tool will display the pattern index and all other sentences accepted by this pattern in section C.

An additional operation provided by the tool allows to export all generated patterns to a GraphML file—for each pattern, the index, the pattern signature, and the accepted sentences are visualized in graph form. Figure 3 shows one such pattern. The exported pattern list gives a good overview of the distribution of sentences among the patterns. If many sentences are associated with few patterns, as shown in Figure 4, then the rate of desired automatic causality extraction is high, as only the first sentence had to be annotated manually. However, a pattern overview, as in Figure 5 shows, that each pattern has only one accepted sentence. Therefore, the CEREC system did not provide any automation of causality extraction. The overview of all patterns gives insight in general usage of causality in the natural language sentences, as the second example indicates that different linguistic patterns were used for writing causal sentences, which represent a high number of grammatical equivalence classes. The number of used grammatical equivalence classes should be kept low to make the best use of automatic causality extraction.

IV. DEMONSTRATION

The active learning of the CEREC system can be demonstrated using the tool together with the automatically generated

pattern overview. A fraction of a publicly available requirements artifact [9] can be used to demonstrate the causality extraction in a requirements engineering context. Actively training the system with sample sentences while observing the pattern overview will showcase how patterns are established and incrementally specified. With further learning, the rate of sentences that are classified correctly increases and more cause-effect graphs are extracted automatically.

V. CONCLUSION

The tool demonstrates the applicability of syntactic causality extraction in the requirements engineering context. Paradigms like model-based testing can profit from increased automation, as it reduces manual effort and accelerates the process of formalization.

The general approach of connecting a syntactic-based signature with an arbitrary extraction algorithm can be extended to other semantic natural language relations. CEREC can be tailored towards any semantic relation with the configuration of a set of constraint generators. The relationship between types of constraints and a semantic relation may be subject to research based on the approach presented in this paper.

ACKNOWLEDGMENT

I would like to acknowledge that this work was supported by the KKS foundation through the S.E.R.T. Research Profile project at Blekinge Institute of Technology. I further thank my supervisors and colleagues Max Junker, Daniel Mendez, Michael Unterkalmsteiner, and Davide Fucci for their feedback and support.

REFERENCES

- [1] I. Hendrickx, S. N. Kim, Z. Kozareva, P. Nakov, D. O. Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz, "Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals," in *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 2010, pp. 33–38.
- [2] C. S. Khoo, S. Chan, and Y. Niu, "Extracting causal knowledge from a medical database using graphical patterns," in *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2000, pp. 336–343.
- [3] R. Girju, "Automatic detection of causal relations for question answering," in *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering-Volume 12*. Association for Computational Linguistics, 2003, pp. 76–83.
- [4] K. Chan and W. Lam, "Extracting causation knowledge from natural language texts," *International Journal of Intelligent Systems*, vol. 20, no. 3, pp. 327–358, 2005.
- [5] S. Wagner, D. M. Fernández, M. Felderer, A. Vetrò, M. Kalinowski, R. Wieringa, D. Pfahl, T. Conte, M.-T. Christiansson, D. Greer *et al.*, "Status quo in requirements engineering: A theory and a global family of surveys," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 28, no. 2, pp. 1–48, 2019.
- [6] H. S. Son, R. Y. C. Kim, and Y. B. Park, "Test case generation from cause-effect graph based on model transformation," in *2014 International Conference on Information Science & Applications (ICISA)*. IEEE, 2014, pp. 1–4.
- [7] J. Fischbach, M. Junker, A. Vogelsang, and D. Freudenstein, "Automated generation of test models from semi-structured requirements," in *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2019, pp. 263–269.
- [8] B. Altenberg, "Causal linking in spoken and written english," *Studia linguistica*, vol. 38, no. 1, pp. 20–69, 1984.

¹Kindly note that at the time of writing this paper, we are not allowed yet to disclose the code as open source.

CEREC

Automatic Cause-Effect-Recognition

The screenshot displays the CEREC tool interface. At the top, a grey box contains the text "The audio configuration is unavailable if no audio input is connected." To its right is an "ANALYZE" button. Below this, the text "Result: causal" is shown, followed by an "INCORRECT" button. A large central area contains a causal diagram with two nodes: "no audio input is connected" and "The audio configuration is unavailable", connected by a right-pointing arrow. Below the diagram are three buttons: "+", "→", and a trash icon, followed by a "CONFIRM" button. On the right side, three red brackets labeled A, B, and C group the interface elements. Bracket A groups the top grey box and the "ANALYZE" button. Bracket B groups the "Result: causal" text, the "INCORRECT" button, the central diagram area, and the three buttons below it. Bracket C groups the "Syntactically equivalent sentences" section and the two sentences below it.

The audio configuration is unavailable if no audio input is connected.

ANALYZE

Result: causal

INCORRECT

no audio input is connected → The audio configuration is unavailable

+ →

CONFIRM

Syntactically equivalent sentences

Pattern index: 0

The redo action is unavailable if the previous action was not an 'undo ' action.

The menu selection is unavailable if the associated action is unavailable.

Fig. 2. Screenshot of the tool

- [9] A. Ferrari, G. O. Spagnolo, and S. Gnesi, "Pure: A dataset of public requirements documents," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 2017, pp. 502–505.

Fig. 3. Example of a causality pattern: a pattern's signature (yellow) and a compliant sentence (grey/white)

Fig. 4. Example for training with few linguistic patterns, resulting in few grammatical equivalence classes and a therefore higher rate of automatic causality extraction

Fig. 5. Example for training with many linguistic patterns, resulting in many grammatical equivalence classes and a therefore lower rate of automatic causality extraction