# Automatic Extraction of Cause-Effect-Relations from Requirements Artifacts

Julian Frattini
julian.frattini@bth.se
Blekinge Institute of Technology
Karlskrona, Sweden

Maximilian Junker
maximilian.junker@qualicen.de
Qualicen GmbH
Munich, Germany

Michael Unterkalmsteiner
michael.unterkalmsteiner@bth.se
Blekinge Institute of Technology
Karlskrona, Sweden

Daniel Mendez
daniel.mendez@bth.se
Blekinge Institute of Technology and fortiss GmbH
Karlskrona, Sweden

## ABSTRACT

*Background:* The detection and extraction of causality from natural language sentences have shown great potential in various fields of application. The field of requirements engineering is eligible for multiple reasons: (1) requirements artifacts are primarily written in natural language, (2) causal sentences convey essential context about the subject of requirements, and (3) extracted and formalized causality relations are usable for a (semi-)automatic translation into further artifacts, such as test cases.
*Objective:* We aim at understanding the value of interactive causality extraction based on syntactic criteria for the context of requirements engineering.
*Method:* We developed a prototype of a system for automatic causality extraction and evaluate it by applying it to a set of publicly available requirements artifacts, determining whether the automatic extraction reduces the manual effort of requirements formalization.
*Result:* During the evaluation we analyzed 4457 natural language sentences from 18 requirements documents, 558 of which were causal (12.52%). The best evaluation of a requirements document provided an automatic extraction of 48.57% cause-effect graphs on average, which demonstrates the feasibility of the approach.
*Limitation:* The feasibility of the approach has been proven in theory but lacks exploration of being scaled up for practical use. Evaluating the applicability of the automatic causality extraction for a requirements engineer is left for future research.
*Conclusion:* A syntactic approach for causality extraction is viable for the context of requirements engineering and can aid a pipeline towards an automatic generation of further artifacts from requirements artifacts.

## CCS CONCEPTS

• **Computing methodologies** → *Rule learning*; • **Software and its engineering** → Genetic programming.


*ASE '20, September 21–25, 2020, Virtual Event, Australia*

ACM ISBN 978-1-4503-6768-4/20/09…$15.00
https://doi.org/10.1145/3324884.3416549

## KEYWORDS

causality extraction, natural language processing, pattern matching, requirements artifacts

**ACM Reference Format:**
Julian Frattini, Maximilian Junker, Michael Unterkalmsteiner, and Daniel Mendez. 2020. Automatic Extraction of Cause-Effect-Relations from Requirements Artifacts. In *35th IEEE/ACM International Conference on Automated Software Engineering (ASE '20), September 21–25, 2020, Virtual Event, Australia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3324884.3416549

## 1 INTRODUCTION

The detection of semantic relations in natural language text has proven to be an aspect of the field of information extraction which has great potential for many areas of application as outlined by the SemEval 2010 task set [14][27][20].
The automatic extraction of causal relations from large corpora of natural language text has proven useful to many fields of research. Some examples for these fields of research are healthcare, where causal relations between symptoms and diseases yield insights for diagnostics [15], and economy, where financial relations can be derived from analysing stock reports [4].
These domain-specific examples limit the scope of an automatic phrase extraction method to causal relations but they in turn increase their precision, as specific semantic structures and lexical cue phrases reoccur in domain-specific texts. Automatic extraction of causal relations is especially applicable in domains where causal relations are explicitly stated.
One field of research that is often overlooked in this context, but shares the qualifying attributes to excel in automated recognition and extraction of causal relations, is requirements engineering. Artifacts in requirements engineering are predominantly written in natural language [28] and describe conditions of a system in sentences conveying a causality relation. This can be seen in the example "If registration is not successful an audible and visual indication shall be provided." [7], where the relation between the unsuccessful registration and the audible and visual indication is expressed in a causal sentence.
Furthermore, the implementation of a system based on requirements engineering artifacts needs to be validated by determining whether the defined requirements are fulfilled. Tests of various granularity are usually a formalized version of specific requirements. Manually transforming natural language requirements into

formalized test cases can be a tedious and error-prone task, as natural language often lacks specificity and contains ambiguous phrasing, which may be interpreted in more than one way.
Approaches to minimize ambiguity in natural language requirements exist in the form of controlled natural language [10][17], which can be easily reused for further formalization [22] or generating executable scenarios from requirements [13] among others. Though controlled natural language poses clear advantages for formalization techniques, it is still underrepresented in requirements engineering in practice. For this reason we do not assume any form of control on the natural language requirements.
An automatic recognition and extraction of causal relations in natural language requirements texts can mitigate the problem of ambiguity by reducing the manual work necessary for transforming causal relations conveyed by sentences into formalized notation.
Above that recent studies have shown that an automatic test generation from cause-effect graphs are effective and viable [25][24]. Our proposed system fills the gap between natural language requirements artifacts and cause-effect graphs: the automatic extraction of causal relations may be used in a pipeline to automatically generate test cases from natural language requirements artifacts.
Our contribution is threefold: first, we introduce a causality pattern structure that is based on the syntactic rather than the semantic structure of a sentence. This structure is specified incrementally to prevent over-fitting. Second, we present an interactive, online machine learning framework that tailors the causality patterns to its input, therefore adapting to any given context. Third, we evaluate the framework by analysing a set of publicly available requirements artifacts and assess the system's capability to automatically extract cause-effect graphs from causal sentences.
Following Roel Wieringa's design science approach [29] our work relies on the problem investigation observed in practice in the industry, where requirements engineering experts at Qualicen GmbH identified a need for an automatic extraction of causal relations from natural language requirements. Our contribution covers the treatment design and provides a preliminary evaluation of the prototypical implementation.
The rest of this paper is structured as follows: Section 2 covers related work in the field of automatic extraction of causal relations. Section 3 introduces the used concepts and Section 4 presents our approach in detail, which is evaluated in Section 5. A conclusion is drawn in Section 6 alongside considerations for future work.

## 2 RELATED WORK

Automatic extraction of causal relations from natural language texts is typically split into two categories [1]: non-statistical techniques that rely on pattern matching and statistical techniques that utilize machine learning for classification of sentences.
Early approaches attempted an automatic causality detection by matching sentences to manually defined lexico-syntactic patterns. A highly cited, early work by Girju et al. [12] investigated a specific pattern <$NP_1$ VP $NP_2$>, where NP represented a noun phrase and VP a verb phrase. A set of semantic constraints on the three pattern elements filtered the matching sentences and allowed for a causality detection via a specific, semantic ranking. Other pattern-based approaches include [4], which utilizes a hierarchy of templates to increase the granularity of patterns.
Another popular approach consists of feature-based classification methods, where causality is detected based on syntactic, semantic or lexical features. A recent approach by Ayyanar et al. [2] based the classification on grammar tags of specific elements of a sentence as well as the distance between the causally related nouns. Other approaches on classification via features and decision trees are [3], where syntactic patterns were manually generated and then used to train a decision tree for the detection of causality, and [11], a refinement of [12] by adding a decision tree for causality detection.
Rink et al. achieved the best result for the SemEval 2010 task 8 challenge of detecting causality [1]. Their proposed framework combines graphical patterns consisting of syntactic, semantic and lexical constraints with a binary classifier to identify causally related events in sentences [21].
Prominent domains for application of causality extraction are question answering [11], the medical domain [15], and economics [4].
The technique can be utilized in the context of requirements and software engineering to reuse extracted cause-effect graphs for other downstream artifacts. One example of this is to integrate an automatic extraction algorithm into a pipeline for automatic test case generation: As test cases are the validation of a built system and this validation is based on the context described in the requirements artifacts [26], an automatic pipeline for formalizing natural language requirements artifacts into test cases provides viable support for the software engineering process [19].
Research on the steps of automatically transforming cause-effect graphs into decision tables [25][24] exists as well as transforming decision tables into test suites [23].
Other possible fields of application include traceability link recovery, where the extracted semantic, causal relation could add to the processing of natural language requirements [16]. The extracted cause-effect graph provides additional semantic information on relations in the requirements and might add to the precision of link recovery.
But most of the existing approaches on causality extraction are not eligible to the context of requirements engineering for three major reasons:

(1) Causal relations described in requirements artifacts are rarely connected in an obvious semantic manner
(2) Semantic causality extraction is based on large corpora of natural language text, which is often not available in the requirements engineering phase
(3) The extracted causal relation is often limited to a word pair, omitting information of the causal sentence vital for reusing the extracted cause-effect graph

Tailoring causality extraction for requirements engineering is explored in a more recent approach by Fischbach et al. [8], where test cases are automatically generated from acceptance criteria. The usage of predefined, manual patterns based on the dependency structure of sentences specifies their approach on the artifact type of user stories, which is avoided in our approach to ensure a greater generalizability.
Our approach is eligible to extract cause-effect graphs from causal

sentences in requirements engineering by dealing with the aforementioned flaws of existing approaches as follows:

(1) Causal relations are detected via syntactic and lexical attributes
(2) The online learning approach continuously improves the causality detection without the need for a large training corpus beforehand
(3) Specific phrase extraction methods are capable of extracting an arbitrary portion of a sentence rather than only a single word.

This approach can be applied to requirements engineering as well as any other domain.

## 3 CONCEPTS

This section introduces and formalizes all concepts used in the context of the implementation.

### 3.1 Foundation

*Sentences.* We denote sentences as $s \in S$, where S is the set of all sentences, called the corpus, and a sentence s can be causal or non-causal.

*Causality.* Causal sentences contain a causality $c \in C$. We introduce the following predicate to express the causality of a sentence s:

| Predicate | Explanation |
|---|---|
| causality(s, c) | $c \in C$ is the causality of sentence $s \in S$ |

A causal sentence and its conveyed causality is illustrated in the following example:

| | |
|---|---|
| **Sentence** | The application is terminated when the x-button is pressed. |
| **Causality** | the x-button is pressed $\Longrightarrow$ The application is terminated |

The relation of causality is reduced to its most simple form, consisting of a cause- and an effect-phrase. Extracting these two phrases from the sentence is subject to the causality extraction.

### 3.2 Formalization

*Structure.* A sentence written in natural language can be processed by natural language processing (NLP) tools in order to derive syntactic and semantic information about the sentence. A sentence can be structured in two ways: by constituency or dependency. A constituency parser groups adjacent words into grammatical units and builds a tree structure, where the root node represents the full sentence. A dependency parser also constructs a tree structure, but in this case the inner nodes of the tree are not chunks grouping words or other chunks into units, but rather the words itself. A word $w_1$ is the parent node of a word $w_2$ if $w_2$ is semantically dependent of $w_1$. The root node of the semantic structure is the root dependency, which is usually the sentence's predicate.
Elemental to the identification of causal sentences in our approach is the syntactic structure, which will be denoted as $t \in T$ and applicable to the following predicates:

| | |
|---|---|
| structureOf(s, t) | $t \in T$ is the structure of sentence $s \in S$ |

*Internal Representation.* A sentence $s \in S$ is formalized by NLP tools into an internal representation. This internal representation of sentences consists of the following aspects:

- Words: each word contained in the sentence in order of appearance with a corresponding part-of-speech-tag indicating the words role within the sentence
- Constituency structure: adjacent words clustered together to chunks of words which represent a specific grammatical unit
- Dependency structure: semantic relations of words to each other

An example for a sentence formalized into the internal representation is given in Figure 1. The tree-structure composed of grey nodes represents the constituency structure, where each node contains its constituent tag. The list of white nodes represents the words of the sentence annotated with their respective part-of-speech-tag. The lines connecting the word nodes represent the dependency structure, where each label on the connection represents the dependency relation type.

*Structural Equivalence.* The premise of our approach is that similarly structured causal sentences also have their cause- and effect-phrases located at similar positions within the sentence. This premise will be explained in detail in Section 4.2. Two tree-like structures are equivalent if both structures contain the same nodes in the same order. Furthermore, a structure $t_1 \in T$ is the subtree of a structure $t_2 \in T$ when $t_1$ contains all nodes in the same order as $t_2$ and potentially more, but not necessarily the other way around. We introduce the following predicate:

| | |
|---|---|
| subtree($t_1$, $t_2$) | structure $t_1 \in T$ is a subtree of structure $t_2 \in T$ |

For example, the structure depicted in Figure 2 is a subtree of the structure in Figure 3, but not a subtree of structure in Figure 4. Hence: $subtree(t_1, t_2)$ but not $subtree(t_1, t_3)$, because $t_3$ lacks a node with the tag SBAR as the first child of the root node.

*Cause-Effect Graphs.* Cause-effect graphs are used to represent the causality, which a sentence conveys, in a formalized, graphical notation [5]. In our approach we limit the extent of cause-effect graphs to their most simple form, consisting of two nodes, one representing the cause and one the effect of the causality relation. Cause-effect graphs will be denoted as $g \in G$ and are applicable to the following predicate:

| | |
|---|---|
| conveys(g, c) | CEG $g \in G$ conveys causality $c \in C$ |

For every causal sentence there exists a cause-effect graph that conveys its causality:

$$\forall s \in S, \exists c \in C : causality(s, c) \rightarrow \exists g \in G : conveys(g, c)$$

Note that the cause-effect graph is not necessarily equivalent to the causality. Ensuring that a cause-effect graph extracted from a sentence actually represents the causality of that sentence is the center of the challenge of correct causality extraction.

### 3.3 Phrase Extraction Methods

Other attempts at causality extraction covered the part of phrase extraction by selecting certain semantic elements from the sentence, for example taking the two noun phrases $NP_1$ and $NP_2$ of the sentence. Introducing phrase extraction methods is necessary

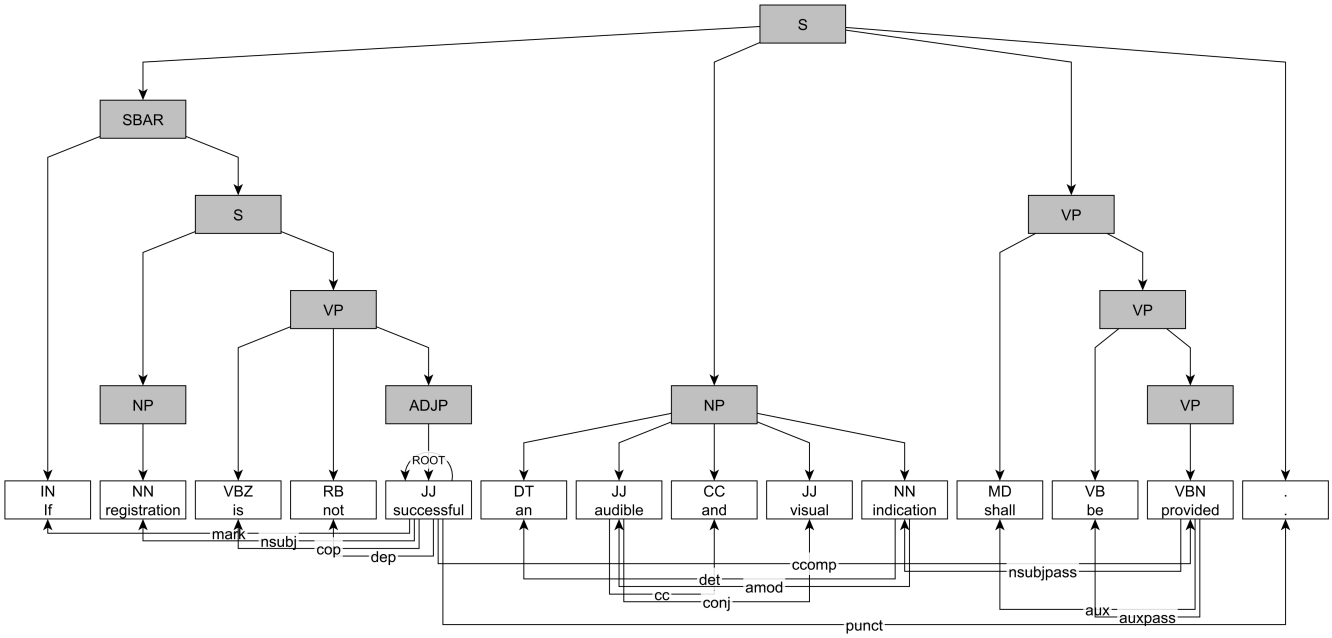


**Figure 1: Internal representation of a natural language sentence formalized via constituency and dependency parsing**

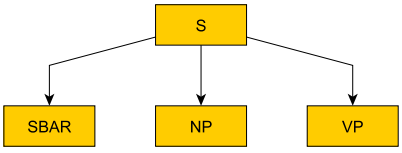


**Figure 2: Syntactic sentence structure $t_1$ of sentence $s_1$**

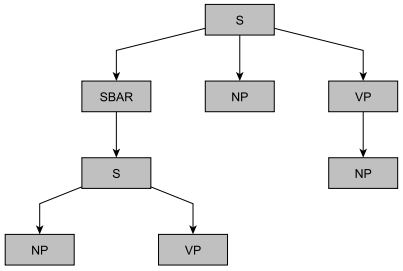


**Figure 3: Syntactic sentence structure $t_2$ of sentence $s_2$**

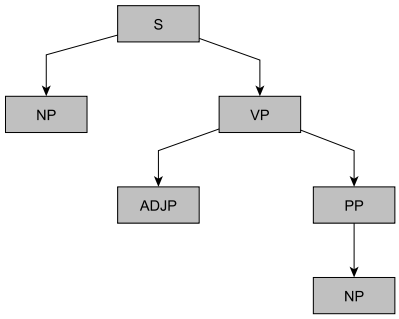


**Figure 4: Syntactic sentence structure $t_3$ of sentence $s_3$**

for our approach for two reasons: first, to provide independence of semantic information, the extraction of phrases must be based on syntactic or lexical information. Second, the extent of the phrase, which represents a part of the causality, shall be arbitrary to ensure that the causality patterns meet the needs for the requirements engineering process. An example from the prominent SemEval 2010 Task 8 set of annotated sentences is the following sentence [14]:

| | |
|---|---|
| **Sentence** | The current view is that the chronic inflammation in the distal part of the stomach caused by Helicobacter pylori infection results in an increased acid production from the non-infected upper corpus region of the stomach. |
| **Causality** | infection $\Longrightarrow$ inflammation |

The causality relation annotated in the SemEval 2010 set of sentences is semantically correct, but causality detection algorithms trimmed to identifying causal relations of this type are restricted to ignore a lot of data in comparable sentences. Without claiming to define the borders of causality in a linguistic or philosophical scope, it is feasible to expect a more encompassing phrase from a causality phrase extraction method, for example as follows:

| | |
|---|---|
| **Sentence** | The current view is that the chronic inflammation in the distal part of the stomach caused by Helicobacter pylori infection results in an increased acid production from the non-infected upper corpus region of the stomach. |
| **Causality** | Helicobacter pylori infection $\Longrightarrow$ chronic inflammation in the distal part of the stomach |

The context of causality extraction in requirements engineering is to extract phrases from a natural language requirements artifact

that can be reused to generate test cases, possibly even automatically. The information value of an extracted causality is highest when the extraction is not limited to the two events in a causal relation represented by a single word, but includes all relevant information like participating actors and conditions. This becomes evident in the following example, where a sentence from the PURE dataset of requirements artifacts [7] is annotated by a causality, which only takes into account the connected events:

| | |
|---|---|
| **Sentence** | If registration is not successful an audible and visual indication shall be provided. |
| **Causality** | registration $\Longrightarrow$ indication |

The causal relation of the registration and the indication is evident, but vital information is lost in the extraction process. This vital information is necessary to more precisely define the initial situation and the expected result of described context. In contrast, the following cause- and effect-phrase would provide the necessary information for later processing, like a test case generation:

| | |
|---|---|
| **Sentence** | If registration is not successful an audible and visual indication shall be provided. |
| **Causality** | registration is not successful $\Longrightarrow$ an audible and visual indication shall be provided |

An phrase extraction method $e \in E$ is a process that, when applied to a sentence $s \in S$, returns a specific phrase from this sentence. The extracted phrase is a substring of the sentence. An phrase extraction method is defined as follows: for a sentence s with causality c, the application of the phrase extraction method on s will yield a cause-effect graph g. The following predicate is introduced for phrase extraction methods $e \in E$:

| | |
|---|---|
| extracts(e, s, g) | Phrase extraction method $e \in E$ extracts cause-effect graph $g \in G$ from sentence $s \in S$ |

A phrase extraction method is generated as outlined in Algorithm 1. In the simplified case of cause-effect graphs for this approach the cause-effect graph g only contains 2 nodes n, one for the cause and one for the effect.

**Algorithm 1** Generate phrase extraction method $e \in E$ for causal sentence $s \in S$ with causality $c \in C$

**Require:** $s \in S$, cause-effect graph $g \in G$ where causality(s, c)
  **for** node $n \in g$ **do**
    find phrase of n in s
    find nodes in the syntactic structure of s, which parent the phrase of n
    create selector of these nodes and add selector to e
  **end for**
  **return** $e$, where extracts(e, s, g) holds

The resulting phrase extraction method works by selecting specific nodes within a sentence and combining all word nodes, that are direct or indirect children of the selected node, together. Multi-word expressions are handled by selecting parenting nodes of the sentence structure, which cover the full expression.
In case of the sentence described above and shown in Figure 1, the phrase extraction method extracting the cause-phrase would select the node S under the node SBAR, which transitively parents all word nodes of which the cause-phrase consists. The phrase extraction method extracting the effect-phrase would select the node NP and VP, which are direct children of the root node, and combine their word nodes to the effect-phrase.

## 3.4 Causality Patterns

A causality pattern combines the recognition of a causal sentence with extraction of a cause- and effect-phrase from this unknown sentence. This connection is based on the grammatical equivalence to a previously known and processed causal sentence example. A causality pattern consists of three elements:

- Signature: a subtree of a syntactic sentence structure, which is the indicator for compliance to a sentence
- Phrase extraction method: an algorithm, which extracts the cause- and effect-phrase when applied to a sentence
- Accepted sentences: a list of all causal sentences, that are compliant to the pattern and applicable by its phrase extraction method

The signature associated with a causality pattern is similar to the syntactic structure $t \in T$ of a sentence, as it represents a sub-tree of a syntactic structure.
The following predicates are introduced:

| | |
|---|---|
| signatureOf(p, t) | $t \in T$ is the signature of the pattern $p \in P$ |
| compliant(p, s) | sentence $s \in S$ is compliant to pattern $p \in P$ |
| extractionOf(p, e) | $e \in E$ is the phrase extraction method of pattern $p \in P$ |
| applicable(p, s) | sentence $s \in S$ is applicable by the pattern $p \in P$ |
| accepted(p, s) | sentence $s \in S$ is accepted by the pattern $p \in P$ |

*Generation.* A causality pattern can be generated from a causal sentence and a manually created cause-effect graph. The signature is extracted from the sentence's syntactic structure to represent its grammatical structure, and the phrase extraction methods are generated from the elements of the cause-effect graph, which are located within the sentence. The generation and maintenance is further explained in Section 4.2. The acceptance of a sentence by a pattern is determined by evaluating compliance and applicability, which are explained next.

*Compliance.* Compliance between a sentence and a pattern determines, whether the sentence belongs to the grammatical equivalence class which the pattern represents. A sentence is compliant to a pattern if and only if the signature of the pattern is a subset of the sentence structure. This is formally expressed as:

$$\forall s \in S, \forall p \in P, \exists t_1, t_2 \in T : signatureOf(p, t_1) \wedge structureOf(s, t_2) \wedge subtree(t_1, t_2) \rightarrow compliant(p, s)$$

Our approach explores an incremental signature: the pattern's signature consists of as many nodes necessary, such that the following two conditions hold:

- The pattern's signature is a subtree to the structure of all accepted sentences

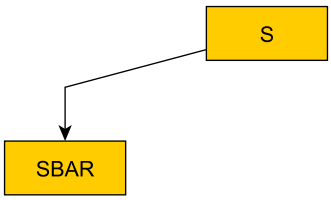


**Figure 5: Signature $t_{p2}$ of a hypothetical pattern $p_2$**

- The pattern's signature is not a subtree to the structure of all non-causal sentences and causal sentences of different patterns.

This can be visualized when reviewing existing examples: assume sentence $s_2$ with structure $t_2$ visualized in Figure 3 is causal and sentence $s_3$ with structure $t_3$ visualized in Figure 4 is not causal. Then a pattern $p_2$ would minimally require a signature $t_{p2}$ as visualized in Figure 5 to differentiate the two sentences. Since $t_{p2}$ is a subtree of $t_2$, sentence $s_2$ is compliant to pattern $p_2$. Sentence $s_3$ is not compliant to pattern $p_2$, because $t_{p2}$ is not a subtree of $t_3$. Note that the pattern $p_2$ would also differentiate between $s_2$ and $s_3$ if it had a the signature $t_1$ shown in Figure 2, but this signature would not be minimal because it contains more nodes than necessary. The conditions ensure that only sentences of similar syntactic sentence structure are compliant to a pattern and therefore eligible to be accepted.

*Applicability.* Applicability of a sentence by a pattern determines, whether the phrase extraction methods of the pattern extract the desired cause-effect graph. A sentence is applicable by a pattern if the phrase extraction method associated with the pattern extracts the cause- and effect-expressions from the sentence that convey its causality relation. This is formally:

$$\forall p \in P, \forall s \in S, \exists e \in E, \exists c \in C, \exists g \in G : causalityOf(s,c) \wedge conveys(g,c) \wedge extractionOf(p,e) \wedge extracts(e,s,g) \rightarrow applicable(p,s)$$

*Acceptance.* A sentence is accepted by a pattern if the sentence's structure is compliant to the pattern's signature and the sentence is applicable by the patterns phrase extraction method.

$$\forall s \in S, \forall p \in P : compliant(p,s) \wedge applicable(p,s) \rightarrow accepted(p,s)$$

# 4 IMPLEMENTATION

This Section describes the implementation of concepts and the functionality of the pattern-based machine learning algorithm, which will then be evaluated in the following chapter. We invite fellow researchers to inspect and use our implementation [9] for further studies.

## 4.1 System Architecture

The **c**ause-**e**ffect **rec**ognition system (cerec) provides an interface which allows for two operations: *training*, where the system is provided with a sentence and a desired cause-effect graph, which conveys the sentence's causality. Here the system is induced to create a new causality pattern for the sentence's syntax, if it does not already exist. When using the operation *testing* the system is provided only with a sentence. If the sentence is causal and its syntactic structure is already covered by a causality pattern, the system will attempt to generate a cause-effect graph from the sentence using the phrase extraction methods associated with the pattern. If a sentence is not correctly processed – meaning that a causal sentence is either not recognized as causal or a non-causal wrongfully recognized as causal – providing a manual correction will result in a training operation to generate a new or adapted pattern.
The *cause-effect recognition* is the core of the cerec system and tasked with the maintenance of pattern correctness, which is in detail explained in Section 4.2. This system contains three major elements:

- *NLP* unit: used to formalize natural language sentences into the internal representation mentioned in Section 3.2
- *CommandGenerator*: used to generate phrase extraction methods mentioned in Section 3.3
- database of *patterns*: contains all generated causality patterns, where each pattern has the structure described in Section 3.4

Furthermore, the cerec system keeps track of all discovered non-causal sentences: as non-causal sentences are not associated with any pattern, they are stored by the system directly, which is of importance for the maintenance of pattern correctness, later described in Section 4.2. We use the OpenNLP and Maltparser [18] for the NLP unit.

## 4.2 Maintenance of Pattern Correctness

*Principles of Maintenance.* In order to ensure the correctness of the database of patterns, a set of principles must be upheld. These principles formalize how the training data – both causal and non-causal sentences – constructs the knowledge database represented by the existing patterns.
The most basic principle is the core assumption of the syntactic cause-effect recognition approach: causal sentences of syntactic similarity are likely to have their causal phrases located at the same position within the sentence, which is the foundation of pattern-based detection and extraction algorithms [11][12][8]. These causal phrases are therefore retrievable by the same phrase extraction methods. This derivation can be denoted as follows:

$$\forall s \in S, \exists c \in C, \exists p \in P : causality(s,c) \wedge compliant(p,s) \rightarrow applicable(p,s)$$

This principle justifies to have sentences of similar syntactic structure accepted by the same causality pattern. All accepted sentences of a pattern are compliant to the signature of a pattern and therefore share syntactic similarities. This makes it likely for the phrase extraction methods associated with the pattern to extract the correct cause-effect graph.
The second principle is that every already discovered non-causal sentence must be non-compliant to every pattern, which formally translates to:

$$\forall p \in P, \forall s \in S, \neg\exists c \in C : causality(s,c) \rightarrow \neg compliant(p,s)$$

To ensure that already discovered non-causal sentences are not compliant to a newly introduced pattern, each signature must be specified in a way to comply to both principles of maintenance. The process of achieving this will be technically explained in next section and brought into context in section thereafter.

*Signature Specification.* An important process for the maintenance of pattern correctness in regard to incremental patterns is the specification of a pattern's signature. The incremental nature of the signature, which defines the grammatical class of each pattern and is used to determine a sentence's compliance to it, derives that the structure consists only of as many nodes such that it fulfills all principles of maintenance.
The overall goal of the machine learning algorithm is to identify all sentences, which are applicable to the same phrase extraction method. To ensure that as many sentences as possible are identified by their respective causality patterns, the pattern's identifying signature must be minimal in size of nodes: each additional node risks that a sentence, which would be applicable by the phrase extraction method of the causality pattern, is not compliant to the pattern anymore, because one node of the sentence's structure differentiates it from the pattern's signature.
The specification process is applied when a sentence is compliant to a pattern, but the sentence is not applicable to the pattern's phrase extraction method, therefore violating the first principle of maintenance. A sentence which is compliant to a pattern but not applicable by its phrase extraction method will be referred to as an *intruder*, because the sentence is falsely compliant to the pattern. An intruder can be causal or non-causal:

- Causal intruders are compliant to a pattern but the phrase extraction method of the pattern does not extract the desired cause-effect graph
- Non-causal intruders are compliant to a pattern, although they should never be

An intruder can only be detected if his actual causality is provided. Knowing the actual causality of a sentence requires semantic domain knowledge and must be provided by a human, either beforehand when preparing training samples or as an correction during the online usage of the system. If an intruder is detected, the pattern's signature must be more specific in order to prevent the pattern's compliance to the intruder. This correction improves the causality detection and extraction of the cerec system in providing further data for the machine-learning algorithm.
The specification of the signature can take two forms: in the most general form, additional nodes are added to the pattern's structure. These nodes are taken from the set of accepted sentences, in order to maintain compliance of the accepted sentences. The algorithm specifically searches for a position in the signature where all accepted sentences contain the same node while the intruding sentence is different. This node is identified as an differentiator between the accepted sentences and the intruder. Adding the differentiator to the pattern's signature will maintain the compliance to the accepted sentences, while revoking the compliance to the intruder. During this process the pattern's signature incrementally reconstructs the structure of the accepted sentences.
The other form of specifying a signature is enforcing additional constraints on certain nodes of the pattern's signature. The implementation explored in our approach uses lexical constraints: explicitly causal cue phrases like "because" or "if" are highly eligible indicators for differentiating between syntactically similar, but causally different sentences. A lexical constraint added to a node of the pattern's signature means that any sentence, that is compliant, must also contain this specific keyword as a word node transitively parented by the node, to which the constraint is applied.
When a signature specification process is triggered, all eligible differentiating specifications are listed and ordered by degree of precision. A specification is more precise when the changes to the signature are minimal, which reduces the risk of later disregarding sentences, which would be accepted but are not compliant anymore due to an overfitting of the signature. The most precise specification is applied.

*Training causal Sentences.* When training a causal sentence in the cerec system, a sentence alongside a cause-effect graph, which conveys its causality relation, is given. The first step of the system is to check if any existing pattern is compliant to the given sentence.
If no pattern is found, then the given sentence structure is novel to the system and needs to be established. A new pattern is generated in the following steps: first, an identifying signature is associated with the pattern. Following the nature of the incremental signatures, this signature initially consists just of the minimal representation of the structure of its first accepted sentence: the root node of the sentence's structure, which is an S-node identifying the tree as structure of a sentence.
With just this minimal S-node as a signature the new pattern would violate the second principle of maintenance, as all previously discovered non-causal sentences would be falsely compliant to the pattern, since every sentence's structure is rooted in the S-node. This is overcome by applying the specification algorithm described in the previous section: the pattern's signature is specified against every previously discovered non-causal sentence, until no non-causal sentence is compliant to the new pattern and therefore the second principle upheld.
Next, a phrase extraction method is tailored to retrieve the cause-effect graph that conveys the sentence's causality relation from the sentence. This is done by dynamically constructing an algorithm as described in Section 3.3. In a last step the given sentence is added as an accepted sentence to the new pattern, as this sentence's affiliation with the pattern is confirmed.
The training algorithm is different when an existing pattern is found to which the causal sentence is compliant. Two cases are possible from here: if the sentence is also applicable to the phrase extraction method associated with the pattern, then this causal sentence is already covered by the cerec system and correctly recognized. The sentence is added to the accepted sentences of the pattern and adds no new pattern to the database.
If the sentence is not applicable to the phrase extraction method associated with the pattern, to which the sentence is compliant, then the sentence violates the first principle of maintenance: the sentence is compliant to a pattern, but not applicable by its phrase extraction methods.

This is resolved by specifying the pattern against the intruding sentence and creating a new pattern for the causal intruder. The result will be two patterns with very similar structures associated to them, but different enough to differentiate the accepted sentences of the original pattern from the intruding sentence. The new pattern will be provided with a specific phrase extraction method to preserve the first principle of maintenance.

*Training non-causal Sentences.* When training a causal sentence in the cerec system, only a sentence is given. The first step of the system is again to check if any existing pattern is compliant to the given sentence.
If no pattern is found, then the non-causal sentence is correctly discarded. The cerec system adds the sentence to its list of non-causal sentences for further processing.
If a pattern is found, to which the sentence is compliant, then the second principle of maintenance is violated. This is resolved by specifying the structure of the compliant pattern against the non-causal intruder. The specification process results in the incremental addition of nodes or constraints to the pattern's signature, until the intruding sentence is no longer compliant to the pattern and the second principle of maintenance therefore upheld.

## 5 EVALUATION

The purpose of the following preliminary study is to prove the feasibility of the causality recognition and phrase extraction method based on syntactic similarities. For this, we use the cerec system as described in Section 4 with lexical constraints and train a set of requirements artifacts containing natural language sentences.

### 5.1 Design

A publicly available dataset $A$ containing requirements artifacts $a \in A$ was selected for the evaluation of the implemented approach. The obvious choice is the SemEval 2010 Task 8 dataset [14], which is often used for the evaluation of causality extraction approaches because causal sentences are already annotated. However, the annotated causal phrases are limited to pairs of events, predominantly two nouns, in semantically causal relation. Our proposed algorithm works on the dataset, but evaluating its effectiveness by detecting causally related noun-pairs defeats the purpose of tailoring the proposed algorithm to requirements engineering. Instead we used the PURE dataset of public requirements documents [7][6] and manually annotated the 18 available artifacts for evaluation. Only full natural language sentences were regarded, as these are subject to our approach. The 18 datasets collectively contained 4457 sentences, 558 of which (12.52%) are causal.
To assess our approach we formulated the following research questions:

- **RQ1**: How effective is the algorithm in automatically detecting and extracting causal relations in a single requirements document?
- **RQ2**: How effective is the algorithm in automatically detecting and extracting causal relations in a single requirements document with previous training?

To evaluate the effectiveness of the algorithm, the system will be *fully trained*: a dataset will be randomly ordered and provided to the system sentence by sentence in a *training* operation. The system will perform this operation and evaluate the process with one of the following flags:

- creation successful/failed (crea+/crea-): a causal sentence has not been accepted by an existing pattern. The creation of a new pattern is attempted
- recognition successful (rec+): a causal sentence is accepted by an existing pattern and the causal relation has been extracted accordingly
- specification successful/failed (spec+/spec-): a sentence has been compliant to an existing pattern, but not applicable to its phrase extraction method, so a pattern specification process to restore the principles of maintenance has been attempted
- discarding successful (disc+): a non-causal sentence has not been accepted by an existing pattern
- deflection successful/failed (defl+/defl-): a non-causal sentence has been compliant to an existing pattern, so a pattern specification process to restore the principles of maintenance has been attempted

The eight flags compose the containing set $F$:

$$F = \{rec+, disc+, crea+, crea-, spec+, spec-, delf+, delf-\}$$

The full training process simulates the expected user interaction with the system: while writing natural language sentences of a requirements document, the system attempts to detect a causality in every sentence upon its completion. If a causal sentence is recognized and the causal phrases are extracted correctly, the process is flagged as *rec+*. If the causal sentence is not recognized, the interacting user can manually provide the cause- and effect-phrase of the sentence and train the system by providing the causal relation alongside the sentence. This information is used for a training process and upon success will generate a new pattern, which flags the process as *crea+*. Should this process fail for any reason, the process is flagged as *crea-*. Should the algorithm detect the causality of a sentence but extract a cause- and effect-expression that does not align with the user's perception of the conveyed causality, he can manually correct the cause- and effect-expression of the sentence, which will be used for a specification process as described in Section 4.2, as the sentence is seemingly compliant to a pattern but not applicable by its phrase extraction method.
If the specification succeeds, the run is flagged as *spec+* – otherwise *spec-*. If a non-causal sentence is non-compliant to any pattern, the sentence is correctly discarded, which is flagged as *disc+*. If the non-causal sentence is wrongfully compliant to a pattern, a specification algorithm has to be performed again to deflect the intruder, but with the exception that no new pattern is created for the new, non-causal sentence. If the specification succeeds, the run is flagged as *defl+* – otherwise *defl-*. The overall training process is illustrated in Figure 6. Here processes are written in round-edge boxes and can result in success or failure.

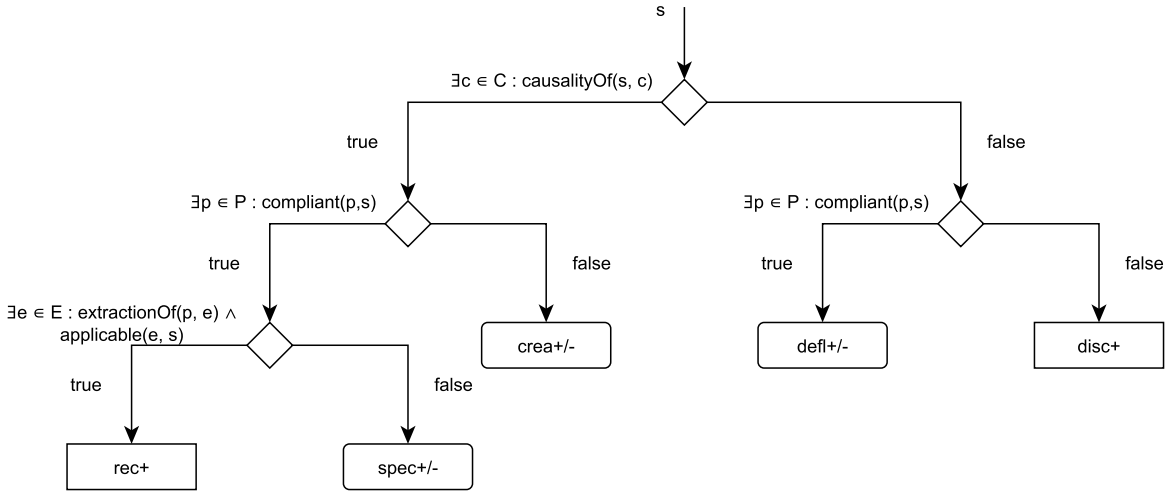


**Figure 6: Decision diagram on the training process**

## 5.2 Measures

The distribution of occurring flags allows to evaluate the effectiveness of the algorithm. We introduce the notation $n_f(a)$ as the number of training processes of the artifact $a \in A$ tagged with a specific flag $f \in F$.

The system's task is twofold: the *detection* of a causal relation in a sentence, which is equivalent to a classification as either causal or non-causal, and the *extraction* of a cause-effect graph from a causal sentence. Both tasks are evaluated using the precision, recall and f-score measures.

In case of detection, true positive cases are all processed sentences flagged as *rec+*, *spec+*, and *spec-*, since these are the cases, where a causal sentence is compliant to an existing pattern. The recall value of this task is calculated as described in Equation 1 and represents, how many of the causal sentences were classified as such.

$$recall(a) = R(a) = \frac{n_{rec+}(a) + n_{spec+}(a) + n_{spec-}(a)}{n_c(a)} \quad (1)$$

In case of extraction, true positive cases are all processed sentences flagged as *rec+*. The recall value of this task is calculated as described in Equation 2 and represents, from how many sentences the cerec system automatically and correctly extracted the cause-effect-graph, hence it is also referred to as *recognition rate*.

$$recall(a) = R(a) = \frac{n_{rec+}(a)}{n_c(a)} \quad (2)$$

## 5.3 RQ1: Effectiveness of the algorithm without training

The study of RQ1 simulates the use of the system in an environment without any previous training, which means that no causality patterns exist at the start of the evaluation. Table 1 shows the results of the measures on the requirements artifacts on the left side.

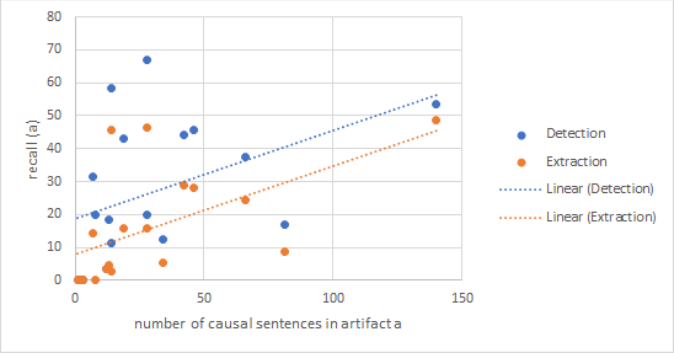


**Figure 7: Recall trend in relation to $n_c(a)$**

Two major insights can be derived from the results of this study: first, the overall number of causal sentences is directly proportionate to the recognition rate. The recall of artifacts with more causal sentences tends to be higher as indicated in Figure 7. The relation between number of causal sentences and recognition rate is due to the nature of the machine learning approach, where an increased number of training data provides more patterns, which the system can learn and reuse for detection and extraction later.

The second insight is that also the number of causal linguistic patterns used is directly proportionate to the recognition rate. This can be observed when comparing the results of the two artifacts *phin* and *qheadache* with each other: even though they contain almost the same amount of causal sentences, qheadache presents a significantly higher precision and recall in both detection and extraction. Investigating the resulting patterns yields that the repeated use of a causal linguistic pattern "If <S>, <NP> <VP>." generated a pattern with the signature shown in Figure 8. Once the pattern was established by manually annotating the first occurrence

| Artifact a | n(a) | $n_c(a)$ | RQ1: without previous training | | | | | | RQ2: with previous training | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Detection | | | Extraction | | | Detection | | | Extraction | | |
| | | | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 |
| blitdraft | 67 | 19 | 89.13 | 43.15 | 58.15 | 75.0 | 15.78 | 26.08 | 100.0 | 21.05 | 34.78 | 100.0 | 21.05 | 34.78 |
| cctns | 183 | 28 | 87.5 | 20.0 | 32.55 | 84.61 | 15.71 | 26.5 | 100.0 | 14.28 | 25.0 | 100.0 | 7.14 | 13.33 |
| dii | 40 | 1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| eirene | 520 | 66 | 81.57 | 37.57 | 51.45 | 74.31 | 24.54 | 36.9 | 84.73 | 33.63 | 48.15 | 83.47 | 30.6 | 44.78 |
| eirene fun | 616 | 140 | 94.0 | 53.71 | 68.36 | 93.4 | 48.57 | 63.9 | 95.81 | 48.99 | 64.83 | 95.44 | 44.85 | 61.03 |
| ertms | 223 | 42 | 90.29 | 44.28 | 59.42 | 85.91 | 29.04 | 43.41 | 100.0 | 23.8 | 38.46 | 100.0 | 19.04 | 32.0 |
| gammaj | 203 | 3 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 66.66 | 66.66 | 66.66 | 50.0 | 33.33 | 40.0 |
| gemini | 392 | 12 | 33.33 | 3.33 | 6.06 | 33.33 | 3.33 | 6.06 | 66.66 | 16.66 | 26.66 | 50.0 | 8.33 | 14.28 |
| getreal | 99 | 2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| keepass | 191 | 46 | 91.3 | 45.65 | 60.86 | 86.66 | 28.26 | 42.62 | 100.0 | 36.95 | 53.96 | 100.0 | 23.91 | 38.59 |
| microcare | 39 | 8 | 100.0 | 20.0 | 33.33 | 0.0 | 0.0 | 0.0 | 100.0 | 12.5 | 22.22 | 0.0 | 0.0 | 0.0 |
| peering | 131 | 13 | 60.0 | 18.46 | 28.23 | 27.27 | 4.61 | 7.89 | 100.0 | 23.07 | 37.5 | 100.0 | 15.38 | 26.66 |
| peppol | 656 | 81 | 71.87 | 17.03 | 27.54 | 56.45 | 8.64 | 14.98 | 95.45 | 10.37 | 18.7 | 87.5 | 3.45 | 6.65 |
| phin | 184 | 14 | 72.72 | 11.42 | 19.75 | 40.0 | 2.85 | 5.33 | 86.84 | 47.14 | 61.11 | 54.54 | 8.57 | 14.81 |
| qheadache | 107 | 14 | 100.0 | 58.57 | 73.87 | 100.0 | 45.71 | 62.74 | 100.0 | 42.85 | 60.0 | 100.0 | 35.71 | 52.63 |
| tcs | 548 | 34 | 65.62 | 12.35 | 20.79 | 45.0 | 5.29 | 9.47 | 67.39 | 18.23 | 28.7 | 55.88 | 11.17 | 18.62 |
| themas | 178 | 28 | 97.91 | 67.14 | 79.66 | 97.01 | 46.42 | 62.8 | 100.0 | 31.42 | 47.82 | 100.0 | 21.42 | 35.29 |
| video search | 80 | 7 | 91.66 | 31.42 | 46.8 | 83.33 | 14.28 | 24.39 | 100.0 | 14.28 | 25.0 | 100.0 | 14.28 | 25.0 |

**Table 1: Performance of automatic causality detection and extraction**

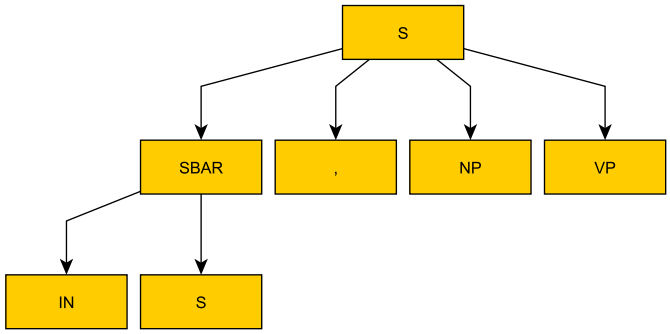


**Figure 8: Signature of the most prominent pattern generated from the *qheadache* requirements artifact**

of this linguistic pattern, all comparable sentences were compliant and also applicable by the generated phrase extraction method. Sentences like "If the file was correctly updated, there is no output." and "If the game is not saved, a dialog box is displayed that asks to the player if he wants to save the game." are compliant to the pattern and the phrase extraction method generated the correct cause- and effect-phrase, therefore successfully automatizing the causality recognition and extraction for these sentences.
The consequence of the two insights are discussed in the conclusion.

## 5.4 RQ2: Effectiveness of the algorithm with previous training

The study of RQ2 simulates the use of the system in an environment with previous training, which means that causality patterns already exist at the start of the evaluation. To simulate this, each of the requirements artifacts was tested as in RQ1, but only after the system has been previously trained with all other artifacts beforehand, effectively starting the training with all patterns extracted from the other artifacts. The results are shown in Table 1 on the right side.
Compared to the results of RQ1, the effect of previous training of the cerec system is not entirely positive: the precision has increased partially, which means that more of the compliant sentences were actually causal and recognized successfully. This comes at the cost of a partially decreased recall, which means that less sentences were compliant to patterns in general. In total these changes result in no overall improvement of the f-score compared to RQ1.
The training beforehand established a large set of patterns based on a corpus vastly outnumbering the sentences contained by the artifact under test. These patterns have undergone a multitude of specification processes as described in Section 4.2, which accounts for the increase in precision of the overall algorithm. On the other hand, the increased specificity of patterns causes less sentences of the artifact under test to be compliant. This represents that the specification process is focused on preventing false positives, hence reducing the recall.
The evaluation of RQ2 shows that the system does not yet profit from an increased beforehand training, which leaves room for improving the scalability of the approach. Reevaluating the focus on precision by balancing the specification process more in favor of recall is to be explored in future approaches.
It is worth mentioning that the lack of improvement in recall when performing RQ2 in contrast to RQ1 is further rooted in the fact that the evaluation was performed on a very heterogeneous set of requirements, written by different authors from different domains. A

common template or guideline for writing requirements was most probably absent, which would have increased the similarity of linguistic causality patterns and therefore the recognition rate. The evaluation was from this perspective performed under worst-case conditions with artifacts that are biased against a learning system that relies on consistency and patterned expressions, which mitigates the benefit from previous training.

# 6 CONCLUSION

## 6.1 Limitations

Two major limitations restrict the generalizability of our study approach: first, the approach is restricted to recognizing and extracting a simplified form of cause-effect-graphs, consisting only of a cause- and a effect-phrase, when in reality conveyed causalities may be a lot more complex. Regarding arbitrarily complex cause-effect graphs as formalized causalities improves the applicability of the causality recognition and extraction approach, but may impose a challenge on the cerec system in terms of the recognition rate.

Secondly, the study only investigates the theoretic capability of an automatic causality extraction system while disregarding an actual implementation in practice. Following Roel Wieringa's design science approach [29], the process framed as *scaling up to practice*, which covers treatment validation, implementation and implementation evaluation, is still required in order to explore the sensitivity of our solution proposal to the practical context.

Further limitations include the exclusive focus on intersentential causality and that patterns' signatures are restricted to lexico-syntactic structures, which disregards the semantic component of causality. The implication of these limitations for the detection and extraction of causal relations from natural language requirements is part of an ongoing study.

## 6.2 Discussion

Though the range of recall values peaks at a promising 48.57%, some artifacts are evaluated with a recognition rate of 0.0%. The deviation of success in recognition leads to two observations: first of all, the proposed cerec system lacks in maturity and has still difficulties in maintaining a consistent level of causality recognition. Possible improvements presented in Section 6.3 may improve the robustness of the approach.

Secondly, some requirements artifacts are more eligible for an automatic causality recognition and extraction. Two major factors influence the applicability of a syntactic causality recognition approach:

(1) Number of causal sentences: a higher number of causal sentences increases the number of causality patterns, which are directly proportionate to the recognition rate
(2) Recurring linguistic causality patterns: reusing formulations for conveying causal relations increases the probability of pattern compliance and phrase extraction method applicability

These derivations are especially interesting when bringing the proposed approach of automatic causality recognition in relation to user interaction: encouraging to comply to the derived recommendations may drastically increase the usage of causal sentences as well as the recognition rate of the cerec system, further benefiting the automatized requirements formalization.

## 6.3 Future work

Apart from the future work of overcoming the mentioned limitations by utilizing the derivations outlined in the discussion, other aspects of the an automatic causality extraction approach are eligible for further investigation.

The signature of a causality pattern does not only consist of a subtree of a syntactic sentence structure, but may apply additional constraints to increase the specificity of the signature. The possibilities of different constraints and their influence on the recognition rate may yield interesting results on the relation between causal sentences an specific semantic, syntactic and lexical attributes.

Furthermore an integration of the causality recognition and extraction system into a full pipeline dedicated to automatically generating artifacts like test cases from natural language requirements documents may be of interest for validating the context, in which the cerec system is supposed to be used. Research in this direction approaches the application of the causality recognition and extraction and may generate actual value for the requirements engineering phase as well as other downstream phases.

Lastly, the potential of the syntactic patterns might be utilized with different semantic relationships: as the patterns used in our approach simply associate a lexico-syntactic structure to a phrase extraction method retrieving certain parts of the sentence, the algorithm could be tailored towards any semantic relation and context. This may initiate a discussion about the degree of connection between semantic relations and the syntactic structure of sentences.

Overall, the results presented in Section 5 prove, that the isolated process of automatic causality extraction based on lexico-syntactic patterns holds great potential with an recognition rate of up to 48.57%, supporting our confidence in the viability of this approach for the requirements engineering context.

# ACKNOWLEDGMENTS

We 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. We would further like to thank Diego Marmsoler and the reviewers of this paper for their valuable feedback.

# REFERENCES

[1] Nabiha Asghar. 2016. Automatic extraction of causal relations from natural language texts: a comprehensive survey. *arXiv preprint arXiv:1605.07895* (2016).
[2] Raja Ayyanar, George Koomullil, and Hariharan Ramasangu. 2019. Causal Relation Classification using Convolutional Neural Networks and Grammar Tags. In *2019 IEEE 16th India Council International Conference (INDICON)*. IEEE, 1–3.
[3] Eduardo Blanco, Nuria Castell, and Dan I Moldovan. 2008. Causal Relation Extraction.. In *Lrec*.
[4] Ki Chan and Wai Lam. 2005. Extracting causation knowledge from natural language texts. *International Journal of Intelligent Systems* 20, 3 (2005), 327–358.
[5] William R Elmendorf. 1973. *Cause-effect graphs in functional testing*. IBM Poughkeepsie Laboratory.
[6] Alessio Ferrari. [n.d.]. PURE: publicly available datasets. https://zenodo.org/record/1414117#.X0pItDVCRPY. Accessed: 2020-03-21.

[7] Alessio Ferrari, Giorgio Oronzo Spagnolo, and Stefania Gnesi. 2017. Pure: A dataset of public requirements documents. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 502–505.

[8] Jannik Fischbach, Maximilian Junker, Andreas Vogelsang, and Dietmar Freudenstein. 2019. Automated Generation of Test Models from Semi-Structured Requirements. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. IEEE, 263–269.

[9] Julian Frattini. [n.d.]. Source code of the cerec system. https://zenodo.org/record/4009160#.X0zuUYtCRPY. Accessed: 2020-08-31.

[10] Norbert E Fuchs and Rolf Schwitter. 1995. Controlled natural language for requirements specifications. (1995).

[11] Roxana Girju. 2003. 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, 76–83.

[12] Roxana Girju, Dan I Moldovan, et al. 2002. Text mining for causal relations.. In *FLAIRS conference*. 360–364.

[13] Michal Gordon and David Harel. 2009. Generating executable scenarios from natural language. In *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 456–467.

[14] Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid O Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. 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, 33–38.

[15] Christopher SG Khoo, Syin Chan, and Yun Niu. 2000. 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, 336–343.

[16] Raúl Lapeña, Jaime Font, Carlos Cetina, and Óscar Pastor. 2018. Exploring new directions in traceability link recovery in models: The process models case. In *International Conference on Advanced Information Systems Engineering*. Springer, 359–373.

[17] Alistair Mavin, Philip Wilkinson, Adrian Harwood, and Mark Novak. 2009. Easy approach to requirements syntax (EARS). In *2009 17th IEEE International Requirements Engineering Conference*. IEEE, 317–322.

[18] Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing.. In *LREC*, Vol. 6. 2216–2219.

[19] Amit Paradkar, Kuo-Chung Tai, and Mladen A. Vouk. 1997. Specification-based testing using cause-effect graphs. *Annals of Software Engineering* 4, 1 (1997), 133–157.

[20] Marta Recasens, Lluís Màrquez, Emili Sapena, M Antònia Martí, Mariona Taulé, Véronique Hoste, Massimo Poesio, and Yannick Versley. 2010. Semeval-2010 task 1: Coreference resolution in multiple languages. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 1–8.

[21] Bryan Rink, Cosmin Adrian Bejan, and Sanda Harabagiu. 2010. Learning textual graph patterns to detect causal event relations. In *Twenty-Third International FLAIRS Conference*.

[22] Matt Selway, Georg Grossmann, Wolfgang Mayer, and Markus Stumptner. 2015. Formalising natural language specifications using a cognitive linguistic/configuration based approach. *Information Systems* 54 (2015), 191–208.

[23] Mamta Sharma et al. 2010. Automatic generation of test suites from decision table-theory and implementation. In *2010 Fifth International Conference on Software Engineering Advances*. IEEE, 459–464.

[24] Hyun Seung Son, R Young Chul Kim, and Young B Park. 2014. Test case generation from cause-effect graph based on model transformation. In *2014 International Conference on Information Science & Applications (ICISA)*. IEEE, 1–4.

[25] Praveen Ranjan Srivastava, Parshad Patel, and Siddharth Chatrola. 2009. Cause effect graph to decision table generation. *ACM SIGSOFT Software Engineering Notes* 34, 2 (2009), 1–4.

[26] Mark Utting, Alexander Pretschner, and Bruno Legeard. 2012. A taxonomy of model-based testing approaches. *Software testing, verification and reliability* 22, 5 (2012), 297–312.

[27] Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 Task 13: TempEval-2. In *Proceedings of the 5th international workshop on semantic evaluation*. 57–62.

[28] Stefan Wagner, Daniel Méndez Fernández, Michael Felderer, Antonio Vetrò, Marcos Kalinowski, Roel Wieringa, Dietmar Pfahl, Tayana Conte, Marie-Therese Christiansson, Desmond Greer, et al. 2019. Status quo in requirements engineering: A theory and a global family of surveys. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 28, 2 (2019), 1–48.

[29] Roel J Wieringa. 2014. *Design science methodology for information systems and software engineering*. Springer.