Noname manuscript No.
(will be inserted by the editor)

Detecting Requirements Defects with NLP Patterns: an Industrial Experience in the Railway Domain

Alessio Ferrari · Gloria Gori · Benedetta Rosadini · Iacopo Trotta · Stefano Bacherini · Alessandro Fantechi · Stefania Gnesi

Received: date / Accepted: date

Acknowledgments. The authors would like to thank Daniel M. Berry for kindly providing the dictionaries used by SREE. This work was partially supported by the H2020-S2RJU-OC-2017 Project ASTRail (ID: 777561).

A. Ferrari
ISTI-CNR, Via G. Moruzzi, 1, 56124, Pisa, Italy
Tel.: +39 050 621 2806
Fax: +39 050 315 2810
E-mail: alessio.ferrari@isti.cnr.it

G. Gori
University of Florence, Dipartimento di Ingegneria dell'Informazione,
Via Santa Marta, 3, 50139 Florence, Italy
E-mail: gloria.gori@unifi.it

B. Rosadini
Alstom Signalling Solutions, Via Pietro Fanfani, 21, 50127, Florence, Italy
E-mail: benedetta.rosadini@alstomgroup.com

I. Trotta
Alstom Signalling Solutions, Via Pietro Fanfani, 21, 50127, Florence, Italy
E-mail: iacopo.trotta@alstomgroup.com

S. Bacherini
Alstom Signalling Solutions, Via Pietro Fanfani, 21, 50127, Florence, Italy
E-mail: stefano.bacherini@alstomgroup.com

A. Fantechi
University of Florence, Dipartimento di Ingegneria dell'Informazione,
Via Santa Marta, 3, 50139 Florence, Italy
Tel.: +39 055 275 8639
Fax: +39 055 275 8570
E-mail: alessandro.fantechi@unifi.it

S. Gnesi
ISTI-CNR, Via G. Moruzzi, 1, 56124, Pisa, Italy
Tel.: +39 050 621 2918
Fax: +39 050 315 2810
E-mail: stefania.gnesi@isti.cnr.it

Abstract **[Context and Motivation]** In the railway safety-critical domain requirements documents have to abide to strict quality criteria. Rule-based natural language processing (NLP) techniques have been developed to automatically identify quality defects in natural language requirements. However, the literature is lacking empirical studies on the application of these techniques in industrial settings. **[Question/problem]** Our goal is to investigate to which extent NLP can be practically applied to detect defects in the requirements documents of a railway signalling manufacturer. **[Principal idea/results]** To address this goal, we first identified a set of typical defects classes, and, for each class, an engineer of the company implemented a set of defect-detection patterns by means of the GATE tool for text processing. After a preliminary analysis, we applied the patterns to a large set of 1866 requirements previously annotated for defects. The output of the patterns was further inspected by two domain experts to check the false positive cases. Additional discard-patterns were defined to automatically remove these cases. Finally, SREE, a tool that searches for typically ambiguous terms, was applied to the requirements. The experiments show that SREE and our patterns may play complementary roles in the detection of requirements defects. **[Contribution]** This is one of the first works in which defect detection NLP techniques are applied on a very large set of industrial requirements annotated by domain experts. We contribute with a comparison between traditional manual techniques used in industry for requirements analysis, and analysis performed with NLP. Our experience shows that several discrepancies can be observed between the two approaches. The analysis of the discrepancies offers hints to improve the capabilities of NLP techniques with *company specific* solutions, and suggests that also company practices need to be modified to effectively exploit NLP tools.

Keywords Natural Language Processing · Requirements Engineering · Natural Language Requirements · Requirements Analysis · Defect Detection · Ambiguity · Precision · Recall · Industrial Case Study · Railway

1 Introduction

The CENELEC norms provide standards for the development of railway safety-critical systems in Europe. The CENELEC EN 50128:2011, specific for software, asks requirements documents for railway systems to be *complete, clear, precise, unequivocal, verifiable, testable, maintainable, and feasible* – clause 7.2.4.4 of the norm (CENELEC, 2011). To ensure that these quality attributes are met, companies developing railway products have a Verification Engineer (VE) who reviews for defects any requirements document produced along the development process. This review activity is time consuming and error prone, and an automated review assistant might help VEs in their task. As well known, requirements are normally edited in natural language (NL) (Mich et al, 2004; Kassab et al, 2014), and the railway domain makes no exception. Several natural language processing (NLP) approaches have been developed to assist requirements review. Part of this work focusses on the identification of typical defective terms and constructions (Fabbrini et al, 2001; Berry et al, 2003; Gnesi et al, 2005; Gleich et al, 2010; Tjong and Berry, 2013; Arora et al, 2015; Femmer et al, 2017), while other focus on artificial intelligence techniques (Chantree et al, 2006; Yang et al, 2011; Ferrari and Gnesi, 2012). However, the literature is lacking large-scale case studies concerning industrial applications of NLP approaches for defect detection (Femmer et al, 2017).

This paper aims at filling this research gap, by providing the experience done within a collaboration between a world-leading railway signalling company, the University of Florence, and ISTI-CNR to investigate the feasibility of using NLP for defect identification in the requirements documents of the company. In this experience, a professional VE of the company developed a set of NLP-based *defect detection patterns* by means of the GATE tool (General Architecture for Text Engineering) for text analysis (Cunningham, 2002). The VE applied the patterns on a dataset of 241 requirements, previously annotated for defects by the VE. A recall of 88.33% and a precision of 64.24% were obtained. Given these encouraging results, the patterns were applied on a larger dataset of 1866 requirements, previously annotated by another VE of the company. This time, the performance was poorer, with a recall of 85.39%, and a disappointing precision of 5.81%. The requirements were inspected by two VEs, to analyse the false positive cases that led to the observed value of precision. This analysis showed that many *true* linguistics defects were not considered in the initial annotation. After marking these defects as true positive cases, the precision increased to 77.37%. To further improve the performance, a set of *discard patterns* were defined to eliminate systematic false positive cases. The final precision value achieved was 83.16%. After this activity, experiments were performed with SREE (Tjong and Berry, 2013), a tool for defect detection in requirements, which searches for typically defective terms. SREE allowed the detection of defects that were not identified by means of our patterns, although at the cost of lower precision. Further analysis showed that SREE and our patterns may play complementary roles in defect detection.

This experience, which involved three professional VEs and a large-scale experimentation on 1866 requirements, shows that NLP technologies can be used to develop *in-house* tools for defect identification. Furthermore, the internal development of the tools can enable the VEs of the company to tune the tools to account for part of the discrepancies that occur between manual reviews and automated ones. This work is an extension of a previous conference paper (Rosadini et al, 2017). With respect to the original paper, the current work provides an improved structure, according to the guidelines of Runeson et al (2012) for reporting case studies in software engineering, and adds the following relevant contributions: (a) a thorough discussion of the false positive cases of the large-scale study on the 1866 requirements dataset; (b) the introduction of discard patterns to reduce the false positive cases, and increase the precision of the approach; (c) an experience in which the SREE tool for term-based defect detection is applied on the requirements.

The remainder of the paper is structured as follows. Sect. 2 summarises related works. In Sect. 3 we describe the patterns for defect detection used in the study. In Sect. 4, we present our research methodology and the case study design. In Sect. 5, the execution of the case study is described, while Sect. 6 presents the results. Sect. 8 highlights the lessons learned, and Sect. 9 provides final remarks, with implications for practice and future research.

2 Related Works

NLP techniques have been largely applied to automate several requirements engineering tasks, including model synthesis (Robeer et al, 2016), classification of requirements into functional/non-functional categories (Casamayor et al, 2012), classification of online product reviews (Maalej and Nabil, 2015), traceability (Sultanov and Hayes, 2013; Cleland-Huang et al, 2010), detection of equivalent requirements (Falessi et al, 2013), completeness evaluation (Ferrari et al, 2014), information extraction (Gacitua et al, 2010; Quirchmayr et al, 2017; Lian et al, 2016), ambiguity detection (Tjong and Berry, 2013; Berry et al, 2003), and its generalisation, defect detection. Since in this paper we focus on defect detection, we will discuss related works in this field. Techniques developed to address the problem of defects in written requirements can be broadly partitioned into two sets. The first set of techniques suggests to use constrained NL or formal/semi-formal languages to prevent or limit defects. The second set of techniques starts from unconstrained NL and generally aims at detecting defects, either by means of manual verification, or by means of automated tools.

2.1 Preventing and Limiting Defects

In the literature, several strategies were defined to *prevent* defects by means of constrained natural languages (Mavin et al, 2009; Pohl and Rupp, 2011) or (semi-)formal approaches (Mich, 1996; Ambriola and Gervasi, 2006; Kof, 2010; Gervasi and Zowghi, 2005).

Concerning the use of constrained natural languages, the EARS (Mavin et al, 2009) and the Rupps template (Pohl and Rupp, 2011) are well known constrained formats for editing requirements. Arora et al (2015) defined an approach to check the conformance of requirements to these templates. Although the adoption of constrained natural languages is not widespread in industry, recent studies have shown that templates can be proficiently used by domain experts (Mavin et al, 2016). On the other hand, templates can limit the amount of requirements defects at the syntactic level, but linguistic defects may still be present at the lexical, semantic and pragmatic levels. Addressing these defects requires other techniques (Arora et al, 2015).

Among the works on (semi-)formal approaches, one of the earlier contributions with a focus on defect prevention is the tool LOLITA (Mich, 1996), which implements an approach for translating NL requirements into object-oriented models. Similarly, Circe-Cico (Ambriola and Gervasi, 2006), starts from NL requirements to generate models to support requirements analysis. Zowghi et al (2001) and Gervasi and Zowghi (2005) suggest logic as a tool to identify and analyse inconsistency in requirements from multiple stakeholders. More specifically, they propose a tool, named CARL, that automatically translates NL into logic and then uses theorem proving and model checking to detect inconsistency in the requirements. The works of Kof aim to semi-automatically formalise NL requirements into message sequence charts (Kof, 2008) and automata (Kof, 2009). More recently, Yue et al (2015) proposed a method and a tool, called *aToucan*, to automatically generate a UML software analysis model from textual, functional requirements specifications expressed in the form of use cases. A systematic study of defects in use case specifications expressed in restricted NL is presented by Zhang et al (2016).

The idea behind the works on (semi-)formal approaches is that the formalisation process may help in identifying requirements defects, since errors in requirements would lead to inconsistencies or omissions in models, and, due to the more formal nature of models, defects are easier to detect in models than in textual requirements. However, through an analysis of two empirical studies, Kamsties (2005) concludes that formalization does not help to eliminate defects from informal requirements documents. Indeed, during the formalization process the analyst makes implicit assumptions, transforming defects into errors. Therefore, even when formal modelling is applied, other techniques for defect defection shall be used as a complement.

2.2 Detecting Defects

Approaches for defect *detection* can be categorised into manual approaches and automated ones, mostly based on NLP. Early and successful techniques for manual requirements inspection were provided by Fagan (1976) and Shull et al (2000). Inspection checklists were developed, among others, by Anda and Sjøberg (2002) and by Kamsties et al (2001), while a survey on the topic of requirements inspection was published by Aurum et al (2002).

Automated NLP approaches for defect detection can be be categorised into those that use rule-based techniques (Wilson et al, 1997; Berry et al, 2003; Gnesi et al, 2005; Gleich et al, 2010; Tjong and Berry, 2013; Arora et al, 2015; Femmer et al, 2017) and those that leverage artificial intelligence techniques (Chantree et al, 2006; Yang et al, 2011; Ferrari and Gnesi, 2012). Our contribution falls into the first category, which collects all the works in which defects are identified based on linguistic patterns.

The Ambiguity Handbook of Berry et al (2003) includes one of the most influential classification of ambiguity-related defects in requirements, and provides a large set of examples of typically dangerous words and constructions. Wilson et al (1997) define a quality model composed of quality attributes and quality indicators, and develop an automatic tool (called ARM: Automated Requirement Measurement) to perform the analysis against the quality model aiming to detect defects and to collect metrics. The tool was applied to industrial requirements from NASA (Rosenberg et al, 1998). Gnesi et al (2005) present QuARS, a tool for defect detection based on a quality model developed by the authors. Similarly, Gleich et al (2010) implemented a grep-like, pattern-based technique to detect defects, supported by statistical NLP techniques such as POS tagging. Kiyavitskaya et al (2008) propose a two-step approach to identify ambiguities in NL requirements. In the first step, a tool applies a set of ambiguity measures to the requirements, in order to identify potentially ambiguous sentences. In the second step, a (manually simulated) tool shows the specific parts that are potentially ambiguous in the set of sentences identified. Tjong and Berry (2013) developed SREE, a tool that identifies defects based on a pre-defined list of dangerous terms. Arora et al (2015) use patterns of linguistic defects as the other works, and, in addition, checks the conformance of the requirements to a given template.

Among the works that use artificial intelligence techniques, Chantree et al (2006) present a technique that helps requirements analysts to identify so-called innocuous ambiguities, i.e., linguistic ambiguities that have a single reading in practice. The focus of this work is on *coordination* ambiguities (i.e., due to the usage of coordinating conjunctions), and a set of heuristics, developed according to a data-set built by human assessors, is presented to discriminate between innocuous and nocuous ambiguities. This approach was extended for *anaphoric* ambiguities (i.e., due to the usage of pronouns) by Yang et al (2011). Finally, Ferrari and Gnesi (2012) propose a graph-based technique to detect *pragmatic* ambiguities (i.e., ambiguities that depend on the context) in NL requirements defined for a specific application domain.

All these works, and in particular the ones employing rule-based techniques, were used as fundamental references to define the defect detection patterns of our study. On the other hand, all the listed works provide limited validation in real industrial contexts, as noted also by Femmer et al (2017). Large data-sets annotated by experts were considered by Falessi et al (2013). However, their focus is solely on redundancy defects (i.e., equivalent requirements), detected by means of information retrieval techniques. The task of finding couples of equivalent requirements is radically different from the one we are dealing with in our study, in which multiple linguistic defects occurring in single requirements are considered. To our knowledge, the more general industrial work on defect detection is the one presented by Femmer et al (2017), who experimented their tool named *Smella* on several datasets provided by three companies. Although domain experts were interviewed to assess the effectiveness of the tool, analysis of the results was performed by two researchers. Another industrial case study on defect detection was presented by Wilmink and Bockisch (2017). Two datasets of 293 requirements in total were used as a benchmark, and term-based defect detection techniques were employed to detect ambiguities. The results were reviewed by domain experts.

Our work contributes to the recent literature on the industrial application of defect detection NLP techniques (Femmer et al, 2017; Wilmink and Bockisch, 2017). Compared to the other studies, in our work the techniques are implemented, tailored, and validated by domain experts. Furthermore, this is the first work that shows how rule-based NLP patterns for defect detection can be *incrementally tuned* to the needs of a company, to address the systematic – and domain-dependent – false positive cases typically raised by these techniques.

3 A Rule-based Approach to Predict Defects

In this section, we first give a background on the NLP technologies used in the study (Sect. 3.1). Then, we describe the NLP-patterns used (Sect. 3.2), and the discard patterns developed to address systematic false positive cases (Sect. 3.3). Finally, we also describe the tool SREE from Tjong and Berry (2013), and we outline how the tool was used in our study (Sect. 3.4).

3.1 NLP Technologies

In this section, we list the natural language processing (NLP) technologies included in the tool GATE (Cunningham, 2002) that was adopted to define the patterns:

- **Tokenization:** this technology partitions a document into separate *tokens*, e.g., words, numbers, spaces, and punctuation.
- **Part-of-Speech (POS) Tagging:** this technology associates to each token a Part-of-Speech, e.g., noun (NN), verb (VB), adjective (JJ), *etc.* Common POS taggers are statistical in nature, i.e., they are trained to predict the POS of a token based on a manually annotated corpus.
- **Shallow Parsing:** this technology identifies noun phrases (NP) – in this case we speak about Noun Chunking – and verb phrases (VP) – in this case we speak about Verb Chunking – in sentences. For example, given the sentence *Messages are received by the system*, a shallow parser identifies {*Messages*, *the system*} as NP, and {*are received*} as VP.
- **Gazetteer:** this technology searches for occurrences of terms defined in a list of terms. In our case, we used it to check the presence of vague terms.
- **JAPE Rules:** this technology allows defining rules (i.e., high-level regular expressions) over tokens and other elements in a text (Cunningham, 2002). A rule identifies sequences of elements that match the rule. Rules are expressed in the intuitive JAPE grammar, which is similar to regular expressions. JAPE rules can be rather long to report. In this paper, for the sake of space, to describe JAPE rules we will use a more concise and intuitive pseudo-code inspired to the JAPE grammar. In JAPE, and in our rules, the symbols reported in Table 1 are used. Furthermore, when we use a term in capital letters, this indicates a form of *macro* that identifies terms of the specific type, e.g., NUMBER identifies numbers, while ELSE identifies the term *else* in its various orthographic forms. Although these

macros differ in terms of semantics, we expect that the reader can infer their meaning.

Table 1: Symbols used in the JAPE grammar.

Expression	**Meaning**
$< expr_1 > \mid < expr_2 >$	$< expr_1 >$ OR $< expr_2 >$
$< expr_1 >, < expr_2 >$	$< expr_1 >$ AND $< expr_2 >$
$! < expr >$	NOT $< expr >$
$< expr > +$	One or more elements matching $< expr >$
$< expr > *$	Zero or more elements matching $< expr >$
$< expr >?$	Zero or one element matching $< expr >$

3.2 Patterns for Defect Detection

This section lists the classes of language defects considered, together with the patterns (i.e., JAPE rules) defined to identify them. Patterns are defined in terms of sequences of tokens to be matched within a requirement. Hence, the output produced by one pattern when applied to a requirement is zero or n requirement fragments (i.e., contiguous sequences of tokens in the requirement) that match the pattern. In Table 2 we report the patterns in a compact version. The JAPE implementation of the patterns, together with the discard-patterns that will be introduced in Sect. 3.3, is available in our public repository[1]. Below, we describe the defect classes addressed by each pattern.

Table 2: Pattern adopted for each defect class.

Defect Class	**Pattern**
Anaphoric ambiguity	P_{ANA} = (NP)(NP)+ (Split)[0,1] (Token.POS == PP | Token.POS =∼ PR*)
Coordination ambiguity	P_{CO_1} = ((Token)+ (Token.string == AND | OR)) [2] P_{CO_2} = (Token.POS == JJ) (Token.POS == NN | NNS) (Token.string == AND | OR) (Token.POS == NN | NNS)
Vague terms	P_{VAG} = (Token.string ∈ Vague)
Modal adverbs	P_{ADV} = (Token.POS == RB | RBR), (Token.string =∼ "[.]*ly$")
Passive voice	P_{PV} = (AUXVERB)(NOT)?(Token.POS == RB | RBR)? (Token.POS ==VBN)
Excessive length	P_{LEN} = Sentence.len > 60
Missing condition	P_{MC} = (IF)(Token, !Token.kind == punctuation)* (Token.kind == punctuation)(!(ELSE | OTHERWISE))
Missing unit of measurement	P_{MU_1} = (NUMBER)((Token)[0, 1](NUMBER))?(!MEASUREMENT) P_{MU_2} = (NUMBER)((Token)[0, 1](NUMBER))?(!PERCENT)
Missing reference	P_{MR} = (Token.string == "Ref")(Token.string == ".") (SpaceToken)?(NUMBER)
Undefined term	P_{UT} = (Token.kind == word, Token.orth == mixedCaps)

[1] `https://github.com/ISTI-FMT/QUARS_plus_plus`

- **Anaphoric ambiguity** Anaphora occurs in a text whenever a pronoun (e.g., *he*, *it*, *that*, *this*, *which*, etc.) refers to a previous part of the text. The referred part of the text is normally called *antecedent*. An anaphoric ambiguity occurs if the text offers more than one antecedent options (Yang et al, 2011), either in the same sentence (e.g., *The system shall send a message to the receiver, and* ***it*** *provides an acknowledge message* - *it* = *system* or *receiver*?) or in previous sentences. The potential antecedents for the pronouns are noun phrases (NP), which can be detected by means of a shallow parser. The pattern P_{ANA} matches any sequence of two or more noun phrases (NP), followed by zero or one sentence separators (Split), followed by a personal pronoun (PP), or other types of pronouns (PR*).
- **Coordination ambiguity** Coordination ambiguity occurs when the use of coordinating conjunctions (e.g., *and* or *or*) leads to multiple potential interpretations of a sentence (Chantree et al, 2006). Two types of coordination ambiguity are considered here. The first type includes sentences in which more than one coordinating conjunction is used in the same sentence (e.g., *There is a* 90° *phase shift between sensor 1* ***and*** *sensor 2* ***and*** *sensor 3 shall have a* 45° *phase shift*). The second type includes sentences in which a coordinating conjunction is used with a modifier (e.g., *Structured approaches and platforms* – *Structured* can refer to *approaches* only, or also to *platforms*). Two patterns were defined, one for each type. P_{CO_1} matches exactly two occurrences (notation "[2]") of one or more Tokens followed by a coordinating conjunction. P_{CO_2} matches cases in which an adjective (JJ) precedes a couple of singular (NN) or plural nouns (NNS), joined by *and* or *or*.
- **Vague terms** Vagueness occurs whenever a sentence admits borderline cases, i.e., cases in which the truth value of the sentence cannot be decided (Berry et al, 2003). Vagueness is associated with the usage of terms without a precise semantics, such as *minimal*, *as much as possible*, *later*, *taking into account*, *based on*, *appropriate*, etc. In our context, we use the list of 446 vague terms provided by the QuARS tool (Gnesi et al, 2005). The list includes single-word and multi-word terms that were collected as source of vagueness in requirements. P_{VAG} matches any term included in the set *Vague* of vague terms.
- **Modal adverbs** Modal adverbs (e.g., *positively*, *permanently*, *clearly*) are modifiers that express a quality associated to a predicate. As noted by Gleich et al (2010), adverbs are discouraged in requirements as potential source of ambiguity. We noticed that, in the requirements of the company, most of the adverbs causing ambiguity were modal adverbs ending with the suffix -*ly*. For this reason, P_{ADV} matches adverbs in normal form (RB) or in comparative form (RBR) that terminate ($ indicates string termination) with -*ly*.
- **Passive voice** The use of passive voice is a defect of clarity in requirements, and can lead to ambiguous interpretations in those cases in which the passive verb is not followed by the subject that performs the action expressed by the verb (e.g., *The system shall be shut down* – by which ac-

tor?). Passive voice detection is also considered by Gleich et al (2010) and by Femmer et al (2014). To identify passive voice expressions, P_{PV} matches auxiliary verbs followed by a verb in past participle (VBN), possibly with negations and adverbs.

– **Excessive length** Longer sentences are typically harder to process than short sentences, and can be source of unclarity. It was chosen to identify all the sentences that are longer than 60 tokens. Although this is a rather weak threshold – for generic English texts, Cutts (1996) recommends not to exceed 40 tokens –, we considered this value appropriate for the length of the sentences in her domain.
– **Missing condition** To be considered complete, each requirement expressing a condition through the *if* clause, shall have a corresponding *else* or *otherwise* clause. P_{MC} checks whether an *if* clause is followed by an *else/otherwise* clause in the same sentence.
– **Missing unit of measurement** Each number is required to have an associated unit of measurement, unless the number represents a reference (see below). Hence, the patterns check whether a number has an associated unit, or a percentage value associated to it.
– **Missing reference** This defect occurs when a reference that appears in the text in the form *Ref.* <X> does not appear in the list of references of the requirements document. To detect this defect we leverage the pattern P_{MR} to extract references in the text, and then – through Java code not reported here – we check whether each number found appears in the list of references.
– **Undefined term** This pattern searches all the terms that follow the textual form used in the company for defining glossary terms (e.g., *restrictiveAspect*), which are expressed in camelCase format (i.e, *mixedcap* orthography). As for the *missing reference* case, we leverage the P_{UT} pattern to search for terms expressed in camelCase, and then we automatically search the glossary to check whether the term is present or not.

The defect classes associated to the patterns can be related to part of the broad quality criteria specified by the CENELEC norms, and reported in Sect. 1. Furthermore, they can be related to the different levels of language to which the defect belong, namely lexical, syntactic, semantic and pragmatic – see, e.g., Berry et al (2003) for a discussion in the context of NL requirements. They can also be related to the level of detection, which, in our case, is either lexical or syntactic. Table 3 reports these relationships, using a structure similar to the one adopted by Gleich et al (2010).

3.3 Discard Patterns

A set of patterns was defined along the case study based on an analysis of the false positive cases produced by the defect detection patterns (see Sect. 6.3.2). For the sake of clarity, we refer to these additional patterns as *discard patterns*. Each discard pattern is associated to one defect class. The defect class is the

Table 3: Patterns associated to the different CENELEC criteria, and to the different levels of language.

Defect Class	**Criterion**	**Lev. of Language**	**Detection**
Anaphoric ambiguity	Unequivocal	Syntactic, Semantic, Pragmatic	Syntactic
Coordination ambiguity	Unequivocal	Syntactic, Semantic	Syntactic
Vague terms	Precise	Pragmatic	Lexical
Modal adverbs	Precise	Pragmatic	Syntactic
Passive voice	Clear	Semantic, Pragmatic	Syntactic
Excessive length	Clear	Semantic, Pragmatic	Lexical
Missing condition	Complete	Semantic, Pragmatic	Syntactic
Missing unit of measurement	Complete	Semantic, Pragmatic	Lexical
Missing reference	Complete	Semantic, Pragmatic	Lexical
Undefined term	Complete	Semantic, Pragmatic	Lexical

one whose patterns generate the systematic false positive cases. The discard patterns, adapted from the JAPE rules reported in our repository, are reported in Table 4, and briefly described below.

– **Anaphoric ambiguity:** the pattern D_{ANA} detects the pronoun within the expression *it shall be possible.* The notation IT_SHALL_BE_POSSIBLE indicates another utility pattern that matches the expressions *it shall be possible*, *it may be possible* and *it should be possible*, in their orthographic variants, and possibly including other terms within the pattern (e.g., *it should **also** be possible*). The JAPE notation "within" indicates that the first argument is completely included in the second argument. Each ambiguity detected through the pattern P_{ANA} is discarded when it includes D_{ANA}.
– **Vague terms:** the pattern D_{VAG_1} matches all the tokens in which the terms *sound* and *light* are used as nouns, according to the annotations of the POS Tagger. The JAPE notation "(?i)" indicates that all orthographic variants of the string shall be matched. Instead, the pattern D_{VAG_2} matches the term *possible* when used within the pattern IT_SHALL_BE_POSSIBLE. D_{VAG_3} matches any vague term included in the list of stop phrases $StopPhrasesVague$, which collects the set of domain specific terms that include vague terms (e.g., *distant signalling distance*, *near miss*), according to our analysis of the false positive cases. Each vague term detected through P_{VAG} is discarded when it includes D_{VAG_1}, D_{VAG_2} or D_{VAG_3}.
– **Modal Adverbs:** the pattern D_{ADV_1} matches the terms *manually* and *automatically.* Instead, D_{ADV_2} matches the term *only* within the expression *information purposes only.* Each modal adverb detected through P_{ADV} is discarded when it includes D_{ADV_1} or D_{ADV_2}.
– **Undefined term:** the pattern D_{UT} matches any unknown term annotation (P_{UT}) that contains a known acronym, i.e., a term included in the list *KnownAcronym.* Any P_{UT} annotation is discarded when it includes D_{UT}.

Table 4: Discard patterns.

Defect Class	**Discard Pattern**
Anaphoric ambiguity	D_{ANA} = ((Token.POS == PP | Token.POS = PR*) within IT_SHALL_BE_POSSIBLE)
Vague terms	D_{VAG_1} = (P_{VAG}, Token.string ==∼ “(?i)sound” | “(?i)light”, Token.POS == NN | NNS) D_{VAG_2} = (P_{VAG} within IT_SHALL_BE_POSSIBLE) D_{VAG_3} = (P_{VAG} within *StophPhrasesVague*)
Modal adverbs	D_{ADV_1} = (Token.string ==∼ “(?i)manually” | “(?i)automatically”) D_{ADV_2} = (P_{ADV} within INFORMATION_PURPOSES_ONLY)
Undefined term	D_{UT} = (P_{UT} contains *KnownAcronym*)

3.4 SREE Patterns

The tool SREE (Tjong and Berry, 2013) is a defect detection tool for NL requirements that is oriented to achieve 100% recall for the defects in its scope, even at the cost of lower precision. SREE leverages a set of dictionaries of typically defective terms (single and multi-word). A requirement that includes a term that matches one of the terms of the dictionaries is returned by SREE as a potentially defective requirement. Furthermore, the matched term is also returned. The key feature of SREE resides in searching only for *lexical* matches, without leveraging POS Taggers or other statistical tools that may, in principle, decrease the recall. The approach is analogous to the one adopted in our work for the pattern for *Vague terms* (see Sect. 3.2).

SREE employs ten dictionaries, and each dictionary is associated to a defect class. The defect classes, together with representative examples of the terms included – called indicators by Tjong and Berry (2013) – are:

- **Continuance:** as follows, below, following, in addition, in particular, *etc.*;
- **Coordinator:** and, and/or, or;
- **Directive:** e.g., etc. , figure, for example, i.e., note, table.
- **Incomplete:** TBA, TBD, as a minimum, as defined, as specified, *etc.*;
- **Optional:** as desired, at last, either, eventually, if appropriate, in case of, if necessary, *etc.*;
- **Plural:** contains a list of 11,287 plural nouns, each ending in “s”;
- **Pronoun:** anyone, he, her, this, they, which, whom, yourself, *etc.*;
- **Quantifier:** all, any, few, little, many, much, several, some;
- **Vague:** ( ), [], as far as, as required, eventually, mutually-agreed, *etc.*;
- **Weak:** can, could, may, might, ought to, preferred, should, will, would.

The complete list of terms for each dictionary, with the exception of the *plural* class, can be found in the work of Tjong and Berry (2013). For the *plural* class, the authors of the current paper contacted Daniel M. Berry, who kindly provided the list. In our study we adopted the dictionaries of SREE. Specifically, each SREE dictionary was imported in GATE as a separate Gazetteer. In our evaluation we apply all the SREE dictionaries, with the exception of the dictionary of the *weak* class, since this class was initially excluded from the analysis.

SREE-reduced A subset of SREE was also adopted in our case study. The selection, which we call *SREE-reduced*, is composed of the terms that are specific to SREE, and are not considered in our patterns. In particular, pronouns are sources of anaphoric ambiguities, and are considered in our P_{ANA} pattern. Furthermore, the coordinators *and* and *or* are sources of coordination ambiguities and are considered in our P_{CO_1} and P_{CO_2}, while the expression *and/or* was considered in our list *Vague* of vague terms. Finally, also part of the terms included in the different SREE dictionaries are included in our *Vague* list. Therefore, *SREE-reduced* is composed of the dictionaries of SREE but excluding: (a) the dictionaries of the *coordinator*, *pronoun* and *weak* class; (b) all the terms in the other dictionaries that were already part of the *Vague* list.

4 Research Methodology and Case Study Design

The experience presented in this paper shares the typical characteristics of case study research, in that the phenomenon under study is analysed within its natural context – i.e., a railway company –, and the boundary between the context and the phenomenon are not clearly evident, and cannot be fully controlled (Yin, 2013). It also includes iterative and improving aspects that are closer to action research (Baskerville and Wood-Harper, 1996), and technology transfer (Gorschek et al, 2006). Overall, our empirical design can be regarded as an *exploratory* and *iterative* case study. Its design largely follows the guidelines of Runeson et al (2012), adapted to the iterative context of our experience. Specifically, in our study, each iteration follows a template reference structure, which includes research question (RQ) definition, data collection procedures, and data analysis procedures. Each iteration is based on specific RQs, and its results are used as triggers to define additional RQs to be answered in the next iteration. In the following, we first outline the RQs produced, and then we describe the template structure adopted in each iteration.

4.1 Research Objective and Research Questions

The research objective of this study is as follows:

Research Objective: Understand to which extent NLP technologies can be used by a railway company to detect defects in NL requirements.

The research objective can be decomposed into the following RQs. Each RQ will be associated to one or more iterations of the case study. It should be noticed that the RQs have been generated along with the case study iterations, and were not already defined at the beginning of the study.

- **RQ1: What is the accuracy of the NLP patterns for defect detection?**
  We want to provide a *quantitative* measure of the effectiveness of the patterns in identifying requirements defects. The assumption is that the higher the measures of accuracy, the more effective are the patterns. To this end, we want to compare the results of the application of the patterns with the defects identified by domain experts, i.e., VEs. The accuracy is measured in terms of precision and recall. The former indicates how many of the defects identified by a tool are considered as defects by VEs. The latter indicates how many of the defects identified by VEs are actually identified by a tool. Precise definitions will be given in Sect. 4.3, and will consider single defects – i.e., requirements fragments that are considered defective according to a specific defect class – and defective requirements – i.e., requirements that include at least one defective fragment.
- **RQ2: Which are the cases of inaccuracy of the NLP patterns for defect detection?**
  We want to provide a *qualitative* analysis of the effectiveness of the patterns. More specifically, we want to understand which are the specific cases in which the patterns fail in identifying defects. This is done in terms of (a) defects identified by VEs that are not detected by the patterns, i.e., *false negative* cases – which impact on recall; and (b) in terms of defects that are detected by the patterns, but are not considered as defects by the VEs, i.e., *false positive* cases – which impact on precision.
- **RQ3: What is the precision of NLP patterns for defect detection when complemented with discard patterns?**
  This question was generated after answering RQ2. Indeed, it was observed that the defect detection patterns generate *systematic* false positive cases, which could be addressed with discard patterns. The application of discard patterns is expected to increase the precision of the overall approach, and this question aims at quantitatively evaluating to which extent the precision can be increased.
- **RQ4: Can a third-party tool identify additional defects?**
  We want to understand whether the usage of an additional tool can allow us to address false negative cases, and to identify additional defects not considered in the patterns. To this end, we apply the dictionaries of SREE, a tool specifically designed to achieve 100% recall on the defects considered. To answer this broad question, we decompose it into the following sub-questions.
  - **RQ4.1: What is the accuracy of SREE with respect to the NLP patterns for defect detection complemented with discard patterns?**
    We first want to understand whether SREE identifies defective requirements identified by the VEs, and not identified by the patterns – i.e., false negatives. By answering this question, we provide a quantitative evaluation of the accuracy of SREE in identifying defective requirements, in terms of recall and precision. The comparison with the pat-

terns is useful to understand whether SREE and the patterns can be considered as complementary tools.

- **RQ4.2: What is the precision of SREE for the defects in its scope?**
  This question was generated after answering RQ4.1, and noticing that SREE generates a large number of false positive requirements. This suggested that SREE may be less precise than the patterns also at the level of single defects. So, we wanted to further assess the precision of SREE for the defects in its scope.
- **RQ4.3: Which additional defects can be identified with SREE?**
  This question was generated after answering RQ4.1. Indeed, we considered that some of the false positive requirements issued by SREE could include specific defects not considered by the patterns. Therefore, the goal was to understand whether novel categories of defects can be identified with SREE.
- **RQ4.4: Which are the false positive cases for SREE?**
  This question was generated after answering RQ4.1, and as a qualitative complement to RQ4.2, to check which are the typical sources of false positives at the level of defects.

4.2 Case and Subjects Selection

The selection of the case study is triggered by the involved company, and by its need to support VEs in their task of requirements review with automated tools. Specifically, the company, represented by the 5th author, contacted two research institutions, namely ISTI-CNR, represented by the 7th author, and University of Florence, represented by the 6th author. To experiment the feasibility of using defect detection NLP techniques, the company allocated one VE (VE1, 3rd author) dedicated to the task, ISTI-CNR provided an Expert in defect detection through NLP (NLP-E, 1st author), and the University of Florence provided a second VE (VE2, 2nd author), who worked at the company as VE, before moving to academia, inside a collaborative PhD program. A third VE, (VE3, 4th author) had already conducted within the company a quality review of parts of the datasets considered in the study. The characteristics of the involved subjects will be described in Sect. 5.1.

4.3 Data Collection and Analysis Procedures

To collect and analyse the data necessary to answer the RQs, each iteration followed a template structure. The template structure of the iterations is depicted in Fig. 1. The template is composed of eight tasks, which are further grouped into three main phases, namely Preparation, Data Collection, and Data Analysis. The phases are designed to ensure a minimal intervention of NLP-E in the execution of the case study. Specifically, the contribution of

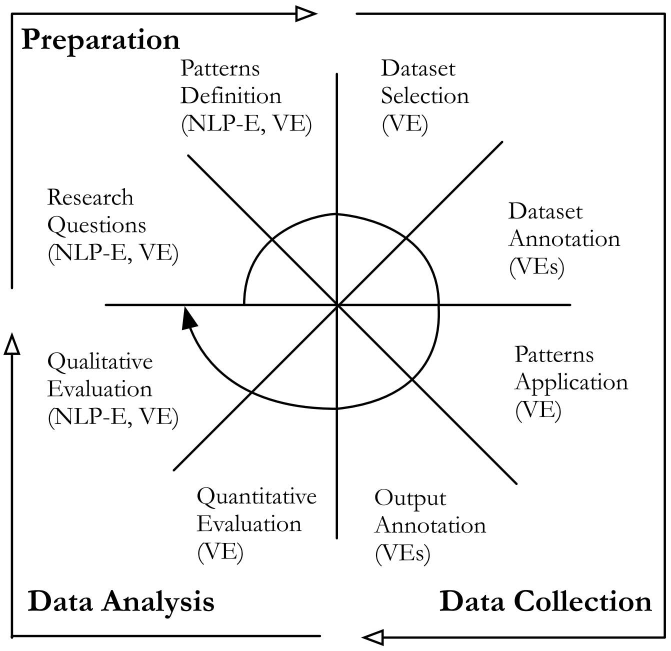
Preparation
Patterns Definition (NLP-E, VE)
Dataset Selection (VE)
Research Questions (NLP-E, VE)
Dataset Annotation (VEs)
Qualitative Evaluation (NLP-E, VE)
Patterns Application (VE)
Quantitative Evaluation (VE)
Output Annotation (VEs)
Data Analysis
Data Collection

Fig. 1: Template structure adopted in the iterations of the case-study.

NLP-E was limited to the Preparation and Data Analysis phases. The Data Collection phase was carried out by the VEs involved in the specific iterations.

Preparation The preparation phase consists of two tasks, described below.

– **Research Questions:** RQs are defined, which are going to be answered by the iteration. If in the previous iteration, the RQs are considered to require another iteration to be answered, the previous RQs are kept. Furthermore, in this phase, a specific instance of the template is chosen so that this is appropriate to answer the questions. In particular, the phases of the template that will be performed are selected – not all the phases are required for each iteration. This phase is led by NLP-E, in collaboration with the VE involved in the iteration.
– **Patterns Definition:** patterns are defined and implemented to support defect detection. The patterns will be employed in the iteration. In this phase, we consider the definition of defect detection patterns, the definition of discard patterns, and also the implementation of the patterns that support the dictionaries of SREE. If the patterns are defined in previous iterations, this phase is not performed. This phase is led by the VE involved in the iteration, under the guidance of NLP-E.

Data Collection Procedure Data are collected according to the following tasks. All the tasks are performed by the VEs involved in the iterations.

- **Dataset Selection:** a requirements dataset is selected, to which we apply the patterns.
- **Dataset Annotation:** the dataset is manually annotated for defects by one or more VEs. Annotations may have been performed also before the current study, as for *Large-scale Study - 1st Iteration*, see Sect. 5.4. In this case, for the sake of structure and clarity of the presentation, we consider the annotation as it would be performed during this task. If the annotations come from a previous iteration, this phase is not performed. The output of this phase is a set of requirements which are annotated as *accepted*, if they do not contain defects, or *rejected*, if they contain at least one defect. Furthermore, depending on the iteration considered, annotations associated to specific defects are also provided. More specifically, the annotation was performed as follows. Given a requirement, this was labelled as *accepted* if it appeared to fulfill the criteria normally adopted by the company. These criteria are derived from the more general guidelines provided by the CENELEC EN 50128:2011 norm (CENELEC, 2011), and considering also the IEEE Std 1233-1998 as a reference (IEEE, 1998)[2]. In particular, a requirement was labeled as *accepted* if it was: (a) *feasible*: what is required is physically and technologically possible, can be done with available resources and is not against laws and regulations; (b) *testable*: can be demonstrated through repeatable tests or is at least verifiable through inspection; (c) *complete*: stand-alone, no missing references, undefined terms, to-be-defined parts, or missing conditions; (d) *clear and unambiguous*; (e) *uniquely identifiable*; (f) *consistent*: no internal contradiction and no contradiction with other requirements. The requirement was labeled as *rejected* in case it did not fulfill one of the criteria. In case the requirement was marked as *rejected* for criterion (c) or criterion (d), the VE involved stated whether the rejection was due to one or more linguistic defect classes associated to the patterns listed in Sect. 3.2. In this case, the VE involved labelled as $defective(i)$ each requirement fragment that included the i-th defect.
- **Patterns Application:** the patterns are applied on the annotated dataset, and potentially defective requirements are produced as output.
- **Output Annotation:** the output of the patterns is annotated for defects by one or more VE. In each iteration, this task is considered as mutually exclusive with the Dataset Annotation task. Indeed this task is mainly oriented to assess the precision of the output of the patterns, and has been introduced when doubts were raised about the quality of the original annotations, or whenever further assessment was required. This task is performed as follows. For each requirement fragment labelled as defective according to pattern i, each VE annotated the fragment as $defective(i)$, if the VE considered the defect as a true defect[3]. Overall, if a fragment was

[2] The standard is currently replaced by ISO/IEC/IEEE 29148:2011 (ISO, IEC, IEEE, 2011).

[3] In this context, we consider as a pattern i also a dictionary from *SREE-reduced*, as defined in Sect. 3.4

annotated as $defective(i)$ by at least one VE, the fragment was marked as $defective(i)$ in the annotated set used for the evaluation.

Data Analysis Procedure Data analysis is performed according to the following tasks.

– **Quantitative Evaluation:** the accuracy of the patterns in detecting defects is evaluated. In particular, we compare the annotations performed by the VEs with the annotations performed by the patterns. Specifically, we evaluate the values of precision and recall of the patterns with respect to the annotations performed. Evaluation measures for single defects and for entire requirements are provided, and defined as follows.
  – *Evaluation Measures by Defect:* To measure the effectiveness of the patterns, we first provide a set of measures that focus on single defective fragments identified by the patterns. Given the pattern associated to the i-th defect, we consider the amount of true positive tp^D as the number of requirements fragments labeled as $defective(i)$ and correctly identified by the pattern; the amount of false positive fp^D as the number of requirements fragments wrongly identified as defective by the pattern; the amount of false negative fn^D as the number of requirements fragments labeled as $defective(i)$ that are not discovered by the pattern. Based on these definitions, we define the measure of precision (p^D) and recall (r^D) as:

$$p^D = \frac{tp^D}{tp^D + fp^D} \quad r^D = \frac{tp^D}{tp^D + fn^D}$$

    The precision p^D is negatively influenced by the amount of defects wrongly identified (fp^D). The recall r^D is negatively influenced by the amount of undetected defects (fn^D).
  – *Evaluation Measures by Requirement:* to have a view of the effectiveness of the patterns applied together, we provide a set of measures that focus on the number of requirements, instead of on the number of defective fragments. Here, we consider the amount of true positive tp^R as the number of requirements labeled as *rejected* for which at least one of the patterns correctly identified a defective requirement fragment; the amount of false positive fp^R as the number of requirements wrongly identified as defective (i.e., at least one of the patterns triggered a defect while the requirement was marked as *accepted*); the amount of false negative fn^R as the number of requirements marked as *rejected* for which none of the patterns triggered a defect. The measures of precision p^R and recall r^R are defined as for p^D and r^D, but considering tp^R, fp^R, and fn^R.

  This task was performed by the VE involved in the iteration. The VE provided NLP-E with the tables with the quantitative evaluation. Depending on the iteration, different evaluation measures are used, among those listed above.

– **Qualitative Evaluation:** cases of inaccuracy of the patterns are evaluated and classified. In particular, the results produced by the patterns are inspected by the VEs, and classes of inaccuracy cases are provided. This task was supported by NLP-E, who was provided with representative examples for each class, and supported the VEs in refining the classes. The interaction was performed by means of on-line calls, and shared documents.

4.4 Validity Procedure

The validity procedure adopted aims to ensure the validity of the data used in the study, and reported in this paper.

To ensure the validity of the annotations performed on the datasets during the Output Annotation task, the annotation process is independently performed by two VEs. The inter-rater agreement is computed by means of the Cohen's Kappa (Landis and Koch, 1977). In case of disagreement, if at least one of the annotators considered a requirement as defective, the requirement was considered defective in the final set used during the analysis. This validity procedure was not followed in the *Pilot Study*, due to its preliminary nature (Sect. 5.3). Furthermore, it was not followed during the Dataset Annotation task. Specific threats associated to this aspect are discussed in Sect. 7.

Second, we ensure the validity of the quantitative results reported, by replicating part of the study. In particular, the *Large Scale Study – 1st and 2nd Iterations*, initially conducted by VE1 were partially replicated by VE2. Discrepancies of the results were evaluated and root causes of the discrepancies were assessed.

Third, to limit the researcher bias, the intervention of NLP-E was limited to the Preparation and Data Analysis phases, while Data Collection was entirely performed by the VE involved in each iteration. NLP-E never had access to the datasets used, but solely to the quantitative results produced, and to specific examples to be used to support the Qualitative Analysis task, and to report the case.

5 Case Study Execution

This section describes the execution of the case study. We first describe the characteristics of the case and the subjects involved, and then we describe the different iterations performed in relation to the RQs.

5.1 Case and Subjects Description

The Company and its Process The company produces signalling equipment for both railway and urban transport applications. In order to efficiently produce such systems, the company develops a set of different products aimed to provide generic functionalities; specific projects based on their product lines are then developed in order to satisfy customer's specific needs. These needs are usually expressed in requirements released by the customer to the companies tendering for contract. The requirements are then elaborated and refined by the company, without relying on standard editing guidelines. The company, for both products and projects, applies the V-model for life-cycle management according to the CENELEC standard (CENELEC, 2011). As dictated by the standard, a requirements' review activity is performed by the Validation Team, according to the criteria reported in Sect. 4.3.

The Subjects Involved The subjects involved in the case are VE1, VE2 and VE3. The background of the three VEs is as follows:

- VE1 has a 3-year working experience in requirements review, as well as testing and validation;
- VE2 has a 2-year working experience in requirements review, as well as process metrics and traceability;
- VE3 has a 10-year working experience in requirements review, and other tasks performed in the Validation Team. VE3 has a strong expertise in contract requirements review, and has an in-depth knowledge of the project associated to the dataset *D-Large*, described later in this paragraph.

VE1, VE2 and VE3 belonged to different groups within the same company, but they were subject to the same company practices. VE1 and VE2 voluntarily participated to the study. VE3 participated to the study since the requirements reviewed by him before this work was conceived (*D-Large*, see below) were used in the case study.

Datasets The datasets made available by the company for this research activity consist of:

- **Pilot Dataset (*D-Pilot*):** this dataset consists of 241 system requirements. This dataset was randomly selected from the requirements documents of a wayside Automatic Train Protection (ATP) system and an interlocking (IXL) system belonging to the same product. ATP systems are embedded platforms that enforce the rules of signaling systems, by adding an on-board automatic control over the speed limit allowed to trains along the track. Instead, IXL systems controls the movement of trains in the railway yard, by setting signal statuses, and moving railway switches. This dataset is composed by the following requirements types: functional, architectural, interface and performance.
- **Large-scale Dataset (*D-Large*):** this dataset consists of 1866 requirements. The requirements belong to a requirements document concerning a system-of-systems that includes an interlocking system, an ATP, a CTC (Centralised Traffic Control) and an Axle Counter. Interlocking and ATP systems have been briefly described above. CTC systems monitor and dispatch trains. Axle Counters are embedded systems located along the railway line, which detect the passing of a train between two points on a track.

Table 5: Outline of the different iterations performed.

ID	Iteration Name	Nature	RQs	Patterns	Dataset
0	Pilot	Exploratory	RQ1 RQ2	Def. Det. Patterns	D-Pilot
1	Large-scale - 1st	Exploratory	RQ1 RQ2	Def. Det. Patterns	D-Large
2	Large-scale - 2nd	Explanatory	RQ1 RQ2	Def. Det. Patterns	D-Large
3	Large-scale - 3rd	Improving	RQ3	Def. Det. Patterns + Discard Patterns	D-Large
4	Large-scale - 4th	Improving	RQ4.1	SREE	D-Large
5	Large-scale - 5th	Explanatory	RQ4.2 RQ4.3 RQ4.4	SREE-reduced	D-Large

Table 6: Tasks performed and subjects involved in each iteration.

ID	Res. Quest.	Pat. Def.	Data. Sel.	Data. Ann.	Pat. App.	Out. Ann.	Quant. Eval.	Qual. Eval.
0	VE1 NLP-E	VE1 NLP-E	VE1	VE1	VE1	-	VE1	VE1 NLP-E
1	VE1 NLP-E	-	VE1	VE3	VE1/ VE2	-	VE1/ VE2	VE1 NLP-E
2	VE1 NLP-E	-	-	-	-	VE1 VE2	VE1/ VE2	VE1/VE2 NLP-E
3	VE2 NLP-E	VE2 NLP-E	-	-	VE2	-	VE2	-
4	VE2 NLP-E	VE2 NLP-E	-	-	VE2	-	VE2	VE2 NLP-E
5	VE2 NLP-E	VE2 NLP-E	-	-	VE2	VE1 VE2	VE2	VE2 NLP-E

They were originally written by the customer in international English language and refined by the company. No particular glossary restrictions are applied and no guideline was provided. This dataset is composed by the following requirements types: functional, architectural, interface and ergonomical.

In all these datasets safety requirements are not included, since they are handled by an independent safety assessment process, which produces separate safety requirements documents.

5.2 Iterations

The execution of the case study consists in a set of iterations, which follow the template structure outlined in Sect. 4. Each iteration is aimed at answering one or more RQs, and, although the overall case study is exploratory, each iteration has a different flavour, which range from *exploratory*, to *explanatory* and to *improving*. Furthermore, in each iteration, different tasks of the template are performed. Tables 5 and 6 give an outline of the different iterations. Overall, the case study consists of six iterations. The first one is a *Pilot Study*, based on a preliminary requirements dataset (*D-Pilot*), while the others belong to the *Large-scale Study*, based on a larger requirements dataset (*D-Large*). Table 5 shows the nature of the iteration, the associated RQs, the patterns and dataset used. Iterations from 0 to 2 were dedicated to investigate the accuracy of NLP patterns (RQ1, RQ2), with different levels of insight. Iteration 3 was dedicated to improve the precision of the patterns (RQ3). Iteration 4 and 5 were focused on the application of the SREE dictionaries (RQ4.1-4). Table 6 shows the tasks performed together with the subjects who participated to the task. The notation VE1/VE2 indicates that the task, initially conducted by VE1, was replicated by VE2.

Here, we briefly summarise the rationale, execution and results of each iteration, with reference – explicit or implicit – to Table 5 and 6. We do not provide all the justifications for the content of the tables, since extensive details are given in the subsequent sections.

- ***Pilot Study:*** this iteration was oriented to have a first understanding of the applicability of NLP patterns for defect detection in the context of the company. To this end, the defect detection patterns (Def. Det. Patterns in Table 5, reported in Sect. 3.2) were defined by VE1 under the guidance of NLP-E, with the objective of maximizing recall, as suggested by Berry et al (2012). Then, they were applied by VE1 on a limited dataset of the company, i.e., *D-Pilot*, which was previously annotated for defects by VE1. A recall of 88.33% (r^R) and a precision of 64.24% (p^R) were obtained, and the recall r^D for single defects reached 100% for the majority of the patterns.
- ***Large Scale Study - 1st Iteration:*** given the encouraging result of the previous iteration, the defect detection patterns were applied by VE1 on *D-Large*, annotated for defects by VE3. The goal was now to understand whether the approach was applicable on a larger set of requirements of the company, annotated by a subject who did not participate to the definition of the patterns. Furthermore, the tasks named Patterns Application and Quantitative Evaluation, originally performed by VE1, were replicated by VE2 (VE1/VE2 in Table 6), to confirm the validity of the produced data. In this iteration, the results were acceptable in terms of recall ($r^R = 85.39\%$), but particularly poor in terms of precision, with $p^R = 5.81\%$. A non-systematic Qualitative Evaluation performed by VE1 suggested that many potential *linguistic* defects were ignored by VE3 in his annotation, thus leading to the low value of precision observed.
- ***Large Scale Study - 2nd Iteration:*** this iteration aimed at systematically explaining the poor results of the previous one. In particular, we were interested in understanding whether the false positive cases produced according to the annotations of VE3 could be considered as true positives (i.e., defects), if an additional annotation was performed with a focus on

linguistic defects. Therefore, the output of the Pattern Application task from the previous iteration was considered – as shown in Table 6, the tasks from Patterns Definition to Patterns Application were not performed again. The Output Annotation task was carried out by VE1 and VE2, and their agreement was assessed. Quantitative Evaluation was performed by VE1, and then replicated by VE2. The precision obtained was $p^R = 77.37\%$, and the average precision at defect level – average of p^D for the different defects – was 72.81%. This confirmed the effectiveness of the patterns for linguistic defects. The Qualitative Evaluation, also replicated, was supported by NLP-E, and allowed the identification of classes of *systematic* false positive cases, which could be potentially discarded with additional patterns.

– ***Large Scale Study - 3rd Iteration:*** based on the Qualitative Evaluation of the previous iteration, we wanted to understand to which extent the precision could be further increased through additional patterns, designed to discard false positive cases (Discard Patterns in Table 5, reported in Sect. 3.3). VE2 took the lead in this activity due to other company-related commitments of VE1, and defined a set of discard patterns under the guidance of NLP-E. With these patterns, the precision p^R further increased to 83.16%, and the average p^D reached 81.36%.

– ***Large Scale Study - 4th Iteration:*** this iteration aimed at understanding whether the defect-detection capabilities of the approach could be complemented with the usage of an additional tool, namely SREE (see Sect. 3.4). To have a general, initial indication, we considered the annotations performed by VE3 on *D-Large* (annotations already used in *Large Scale Study - 1st Iteration*), and we checked whether SREE was able to identify *requirements* that were annotated as defective by VE3, but were not identified by our patterns. To this end, the performance of SREE, in terms of p^R and r^R, were compared with those of the defect detection patterns complemented with discard patterns. VE2 performed all the tasks included in this iteration. The Quantitative Evaluation task showed that SREE achieved higher recall with respect to our patterns ($r^R = 96.63\%$ vs 85.39%), but at the cost of lower precision ($p^R = 5.45\%$ vs 6.24% – i.e., 351 additional false positive requirements). SREE was therefore recognised as an appropriate complement to our patterns, i.e., undetected defective requirements could be identified, but further investigation was required to explain its poor performance in terms of precision.

– ***Large Scale Study - 5th Iteration:*** this iteration was driven by the low value of precision obtained with SREE at the level of requirements, and was oriented to have a fine-grained assessment of the performance of SREE. Specifically, we wanted to assess the precision of SREE at the level of the single *defects* in its scope. VE2 used a subset of the SREE dictionaries, i.e., *SREE-reduced* (see Sect. 3.4), including solely those terms that were specific to SREE and were not already considered in our patterns. The Output Annotation task was performed in parallel by VE1 and VE2 on the single defects produced by *SREE-reduced*, and their agreement was assessed. Although the average p^D for the different defects resulted to be only 11.29%, the Qualitative Evaluation, performed by VE2 and NLP-E, showed that several novel classes of *defects* discovered were not considered by our patterns. This confirmed the complementary role of SREE with respect to our patterns.

In the following sections, we report how each specific iteration was executed. The reader should refer to Table 5 and Table 6 to have a structured summary of the information provided in each section.

5.3 Pilot Study

Fig. 2 gives an outline of the iteration. The iteration involved NLP-E and VE1, and aimed to address RQ1 and RQ2. In this iteration, all the tasks of the template are performed, with the exception of Output Annotation. This iteration was *exploratory*, in that it aimed to assess the accuracy of NLP patterns on a limited dataset of the company. The tasks performed are as follows:

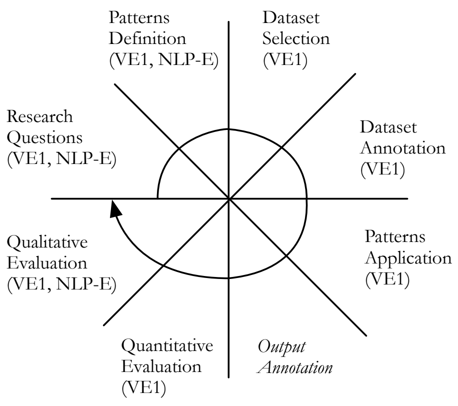
Patterns Definition (VE1, NLP-E)
Dataset Selection (VE1)
Dataset Annotation (VE1)
Patterns Application (VE1)
Output Annotation
Quantitative Evaluation (VE1)
Qualitative Evaluation (VE1, NLP-E)
Research Questions (VE1, NLP-E)

Fig. 2: Structure of the Pilot Study.

- **Research Questions:** RQ1 and RQ2 were defined in collaboration between NLP-E and VE1. In this iteration, the underlying goal was to establish whether the patterns were able to achieve a recall value close to 100%. As noted by Berry et al (2012), defect detection techniques shall favor recall over precision since the cost of undetected *true* defects is much higher than the cost of manually discarding false positive cases.
- **Patterns Definition:** NLP-E considered that assessing the effectiveness of a domain-generic tool for defect detection (e.g., QuARS, presented by Gnesi et al (2005)) would have required a strong *expertise* in the domain of the requirements documents. In addition, he considered that, if the tool had provided too many false positive cases, e.g., *innocuous ambiguities* (Chantree

et al, 2006), the company would not have considered the tool as appropriate for its needs. Hence, it was decided to let VE1 develop the tool *in-house*, with the support of NLP-E.
VE1 was initially required to study the papers of Berry et al (2003), Gnesi et al (2005), Gleich et al (2010), Tjong and Berry (2013) and Arora et al (2015). Then, she was required to perform the tutorials provided by GATE (General Architecture for Text Engineering, see Cunningham (2002)), which was the generic NLP tool selected to be tailored to support defect detection. The tool was chosen since it was considered sufficiently easy to use for an engineer, and sufficiently powerful for the task. After this training, VE1 and NLP-E met to define the defect classes on which to focus. Priority was given to those defect classes that were considered more relevant from the point of view of VE1 – taking into account the defect classes provided by Berry et al (2003), and by the other papers she had studied – and whose identification was considered feasible by NLP-E. VE1 autonomously implemented the patterns, under the supervision of NLP-E. The patterns developed are reported in Sect. 3.2.
- **Dataset Selection:** *D-Pilot* was selected by VE1 under the guidance of representatives of the company.
- **Dataset Annotation:** the dataset was manually annotated by VE1. After this task, 120 requirements were marked as *rejected*, while 121 were marked as *accepted*[4].
- **Patterns Application:** the task was then carried out using the support of GATE.
- **Quantitative Evaluation:** VE1 provided NLP-E with a table with the results of the evaluation. The measures used are for defects, tp^D, fp^D, fn^D, p^D, r^D, and for requirements, tp^R, fp^R, fn^R, p^R, r^R.
- **Qualitative Evaluation:** VE1 evaluated false positive and false negative cases, and provided representative examples. VE1 and NLP-E interacted so that NLP-E could tailor the cases and examples for reporting.

5.4 Large-scale Study - 1st Iteration

Fig. 3 gives an outline of the iteration. The iteration involved NLP-E, VE1, VE2 and VE3. This iteration is still based on RQ1 and RQ2, in that it aims to further answer the RQs with a case modification – in terms of dataset used and annotator –, and the nature of the iteration is still *exploratory.* All the tasks, with the exception of Patterns Definition and Output Annotation are performed. The patterns were the one used in the previous iteration. To confirm the validity of the produced data, VE2 replicated part of the tasks. The parts replicated by VE2 are represented in dashed line in Fig. 3. The tasks performed are as follows.

[4] The dataset appears balanced since VE1 continued to randomly select new requirements from the original requirements considered, until a balanced number of accepted and rejected requirements was obtained.

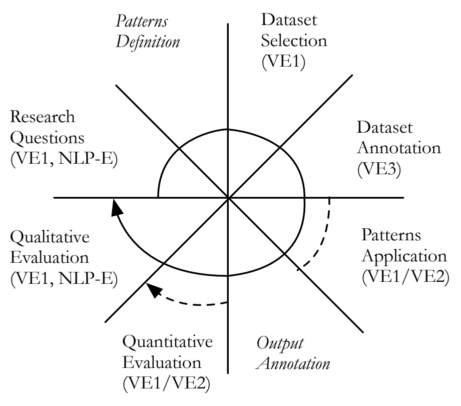
Patterns Definition
Dataset Selection (VE1)
Dataset Annotation (VE3)
Patterns Application (VE1/VE2)
Output Annotation
Quantitative Evaluation (VE1/VE2)
Qualitative Evaluation (VE1, NLP-E)
Research Questions (VE1, NLP-E)

Fig. 3: Structure of the Large-scale Study – 1st Iteration.

- **Research Questions:** the research questions RQ1 and RQ2 were kept from the previous task. The objective of this iteration was to perform an assessment of the patterns on a larger requirements dataset of the company, previously validated by another VE (i.e, VE3), to understand to which extent the approach could be applicable more widely within the company.
- **Dataset Selection:** *D-Large* was selected by VE1, under the guidance of representatives of the company.
- **Dataset Annotation:** the defects of the document were previously annotated by VE3, following the criteria of the company already outlined in Sect. 4.3, and employed by VE1 for the *Pilot Study*. Since this task was performed before this work was conceived, the annotation of the defective fragments was not performed by VE3, who just marked requirements as *accepted* or *rejected*, and described the reasons for rejection in a specific requirements validation document. From the 1866 requirements, 1733 were marked as *accepted*, while 93 were marked as *rejected*.
- **Patterns Application:** the task was initially carried out using the support of a tool developed by VE1 on top of GATE to facilitate the analysis of the results. In the replication, the task was performed by VE2, but using solely the support of GATE.
- **Quantitative Evaluation:** the measures adopted to evaluate the effectiveness of the patterns in identifying defective requirements are tp^R, fp^R, fn^R, p^R and r^R. Intuitively, these measures indicate whether the application of the different patterns simultaneously allows the identification of requirements that were marked as *rejected* by VE3. Since VE3 did not annotate fragments, for this analysis we do not consider evaluation measures for the single defects as in the *Pilot Study*.
- **Qualitative Evaluation:** given the poor results obtained from the Quantitative Evaluation (see Sect. 6.2), especially in terms of precision, this task was performed by VE1 as a non-systematic inspection of the false negative

and false positive cases. The inspection of the false positive cases was oriented to understand whether these cases included defective requirements not initially annotated by VE3. This evaluation triggered the *Large-scale Study – 2nd Iteration*, which aimed to more rigorously explain the poor results.

5.5 Large-scale Study - 2nd Iteration

Fig. 4 gives an outline of the iteration. The iteration involved NLP-E, VE1, VE2, and was performed to provide a more informed answer to RQ1 and RQ2. The iteration has an *explanatory* nature, in that its underlying goal was to explain whether the false positive cases identified in the previous iteration could be considered as true positive cases, from the point of view of more strict annotators. To confirm the validity of the produced data, VE2 replicated part of the tasks. The parts replicated by VE2 are represented in dashed line in Fig. 4. The tasks performed are as follows.

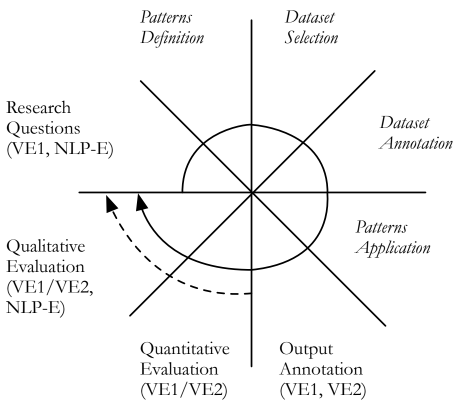
Patterns Definition
Dataset Selection
Research Questions (VE1, NLP-E)
Dataset Annotation
Patterns Application
Qualitative Evaluation (VE1/VE2, NLP-E)
Quantitative Evaluation (VE1/VE2)
Output Annotation (VE1, VE2)

Fig. 4: Structure of the Large-scale Study – 2nd Iteration.

- **Research Questions:** RQ1 and RQ2 were considered not sufficiently answered by the previous iteration, and the iteration was designed to understand to which extent the low value of precision observed was due to inaccuracies in the annotation process performed by VE3.
- **Output Annotation:** a second annotation process was performed on the requirements marked as defective by at least one of the patterns. In this annotation process, two VEs (VE1 and VE2) independently annotated the output of the patterns. The agreement between annotators was estimated

with the Cohen's Kappa, resulting in $k = 0.82$, indicating an almost perfect agreement[5].

- **Quantitative Evaluation:** since in this analysis we focus solely on the output produced by the patterns, we consider neither the amount of false negative cases, nor the measure of recall. Hence, we consider tp^D, fp^D, p^D, for each defect class i, and tp^R, fp^R, p^R, as measures of the precision over requirements.
- **Qualitative Evaluation:** the task was performed by VE1 first, and was later reviewed VE2, to give a first categorisation of the false positive cases. The categorisaiton was refined by NLP-E based on the examples given by the VEs, with a particular focus on systematic categories of false positives, which could be potentially discarded with additional patterns.

5.6 Large-scale Study - 3rd Iteration

Fig. 5 gives an outline of the iteration. This iteration involved NLP-E and VE2, was aimed at answering RQ3, and had an *improving* nature. Indeed, the goal of this iteration was to understand whether the performance of the patterns in terms of precision could be improved with discard patterns. To implement the foreseen improvement of the patterns, VE2 was actively involved in the activity. Indeed, at this stage, VE1 was committed to a mentoring program within the company, to disseminate the best practices for requirements quality learned throughout the experience. The task performed in this iteration are as follows.

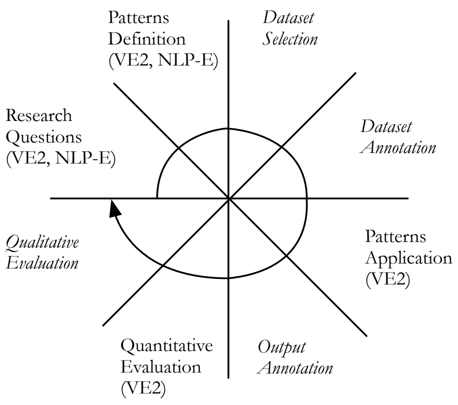
Patterns Definition (VE2, NLP-E)
Dataset Selection
Research Questions (VE2, NLP-E)
Dataset Annotation
Patterns Application (VE2)
Qualitative Evaluation
Quantitative Evaluation (VE2)
Output Annotation

Fig. 5: Structure of the Large-scale Study – 3rd Iteration.

[5] According to (Landis and Koch, 1977), the following qualitative measures are associated to the different ranges of the Cohen's Kappa: $k < 0$, no agreement; $0 \leq k \leq 0.20$, slight; $0.21 \leq k \leq 0.40$, fair; $0.41 \leq k \leq 0.60$, moderate; $0.61 \leq k \leq 0.80$ substantial; and $0.81 \leq k \leq 1$ almost perfect agreement.

- **Research Questions:** the Qualitative Analysis performed in the previous iteration allowed NLP-E, VE1 and VE2 to observe that a set of systematic false positive cases could be addressed with specific patterns designed to discard these cases (see Sect. 6.3.2). Therefore RQ3 was defined, and the iteration was designed to define, apply and evaluate the discard patterns in conjunction with the defect detection patterns.
- **Patterns Definition:** VE2 performed a self-training, analogous to the one performed by VE1 (i.e., a study of the selected literature, and a tutorial on GATE) during the *Pilot Study*. Afterwards, VE2 implemented the discard patterns, under the supervision of NLP-E. The discard patterns are reported in Sect. 3.3.
- **Patterns Application:** the patterns were applied by means of GATE.
- **Quantitative Evaluation:** the evaluation was performed by VE2 considering the annotations produced in the previous Output Annotation task. As in the previous iteration, the evaluation measures used are tp^D, fp^D, p^D, for each defect class i, and tp^R, fp^R, p^R.

The Qualitative Evaluation was not performed, since the goal was only to assess whether the discard patterns could improve the performance of the overall approach in terms of precision.

5.7 Large-scale Study – 4th Iteration

Fig. 6 gives an outline of the iteration. The iteration involved NLP-E and VE2, and aimed to give an answer to RQ4.1. In the context of the case study, this analysis was performed to understand whether the dictionaries of SREE could be used to identify additional requirements defects that could not be identified with our patterns. The nature of the iteration was again *improving*, and consisted of the following tasks.

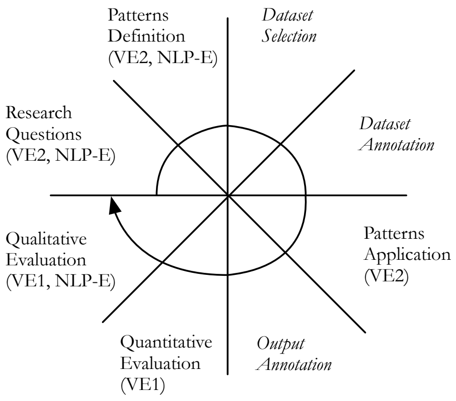
Patterns Definition (VE2, NLP-E)
Dataset Selection
Research Questions (VE2, NLP-E)
Dataset Annotation
Patterns Application (VE2)
Qualitative Evaluation (VE1, NLP-E)
Quantitative Evaluation (VE1)
Output Annotation

Fig. 6: Structure of the Large-scale Study – 4th Iteration.

- **Research Questions:** the iteration was designed to compare the defect detection capabilities of SREE with respect to our patterns, and in particular, whether SREE actually allows to achieve higher values of recall. Therefore, RQ4, and its first refinement, RQ4.1, were defined by NLP-E and VE2.
- **Patterns Definition:** under the guidance of NLP-E, each SREE dictionary, as reported in Sect. 3.4, was imported in GATE by VE2 as a separate Gazetteer. As mentioned, in our evaluation we apply all the SREE dictionaries, with the exception of the dictionary of the *weak* class (see Sect. 3.4).
- **Patterns Application:** the patterns implementing the SREE dictionaries were applied by VE2 by means of GATE.
- **Quantitative Evaluation:** the annotations considered for these requirements are those of V3 only, from *Large-scale Study – 1st Iteration*. Indeed, in this phase, we are interested in understanding whether the dictionaries of SREE applied altogether are able to detect defects, identified by VE3, that our patterns were not able to detect. To this end, SREE is compared with our patterns according to the values of tp^R, fp^R, fn^R, p^R, r^R. The patterns considered include the defect-detection patterns, plus the discard patterns.
- **Qualitative Evaluation:** this task was performed by VE2 with the support of NLP-E in a non systematic way, to observe defective requirements that could be detected by SREE.

5.8 Large-scale Study – 5th Iteration

Fig. 7 gives an outline of the iteration. The iteration involved NLP-E, VE1 and VE2, and aimed to answer RQ4.2, RQ4.3 and RQ4.4. The iteration had an *explanatory* nature. Indeed, from the previous iteration, a high amount of false positive requirements was returned by SREE with respect to our patterns. This suggests that SREE may be less precise also at the level of defects. On the other hand, these false positive requirements may conceal defects that were not considered by VE3. Therefore, it was decided to evaluate the potential degree of precision for the single defects identified by SREE. The tasks performed in this iteration are as follows.

- **Research Questions:** NLP-E and VE2 considered that further investigation was required to answer RQ4, and its refinement RQ4.2, 4.3 and 4.4 were defined. Specifically, with RQ4.2 we wanted to assess which was the precision of SREE at the level of single defects, since low precision was observed at the level of requirements, after answering RQ4.1. Furthermore, we wanted to systematically study the specific defects that could be detected with SREE, and that could not be detected with our patterns (RQ4.3). With RQ4.4, we wanted to provide a qualitative evaluation of the false positive cases at the level of single defects.
- **Patterns Definition:** to evaluate the false positive cases issued by SREE at the level of defects, a selection of the SREE dictionaries was adopted for

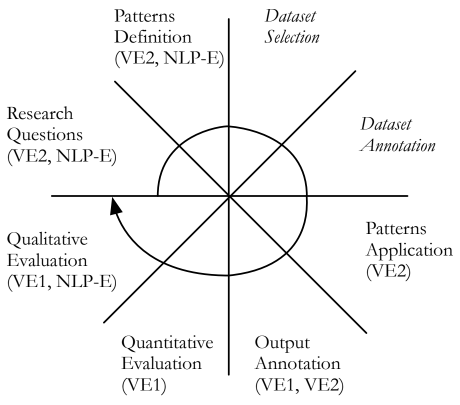
Patterns Definition (VE2, NLP-E)
Dataset Selection
Dataset Annotation
Patterns Application (VE2)
Output Annotation (VE1, VE2)
Quantitative Evaluation (VE1)
Qualitative Evaluation (VE1, NLP-E)
Research Questions (VE2, NLP-E)

Fig. 7: Structure of the Large-scale Study – 5th Iteration.

the analysis, which we call *SREE-reduced* (see Sect. 3.4). Indeed, we recall that, to address RQ4, this analysis was oriented to understand to which extent the SREE dictionaries could complement our patterns.

- **Patterns Application:** the patterns were applied by means of GATE.
- **Output Annotation:** a second annotation process was performed on the requirements marked as defective by at least one of the patterns derived from the dictionaries of SREE. VE1 and VE2 independently vetted the output derived from the application of *SREE-reduced*, and decided whether the defects issued were true positive or false positive cases. For each SREE defect class associated to one *SREE-reduced* dictionary, all the requirements labelled as defective according to the dictionary were considered. An exception is the *plural* class, for which a sample of 50 requirements labelled as defective was randomly chosen. The annotator agreement was estimated with the Cohen's Kappa, resulting in $k = 0.79$, indicating substantial agreement.
- **Quantitative Evaluation** the values of tp^D, fp^D and p^D were used for each single defect class of SREE considered.
- **Qualitative Evaluation:** true positive and false positive cases were analysed and classified by VE2, under the supervision of NLP-E, for each dictionary of *SREE-reduced.* True positives were analysed to answer RQ4.3, while false positives were analysed to answer RQ4.4.

Table 7: Results for single defects and requirements for the *Pilot Study*.

Defect Class	tp^D	fp^D	fn^D	p^D	r^D
Anaphoric ambiguity	22	8	0	73.33%	100%
Coordination ambiguity	16	8	0	66.66%	100%
Vague terms	21	16	10	56.75%	67.74%
Modal adverbs	28	14	0	66.66%	100%
Passive voice	343	60	0	85.11%	100%
Excessive length	200	30	133	86.95%	60.06%
Missing condition	66	14	2	82.5%	97.05%
Missing unit of measurement	2	2	2	50%	50%
Missing reference	10	0	0	100%	100%
Undefined term	208	76	0	73.23%	100%
Requirements	tp^R	fp^R	fn^R	p^R	r^R
	106	59	14	64.24%	88.33%

6 Results

6.1 RQ1, RQ2: Pilot Study

6.1.1 RQ1: What is the accuracy of the NLP patterns for defect detection?

In Table 7 we report the results of the different evaluation measures to establish the accuracy of the patterns. We see that, although the patterns for *anaphoric ambiguity* and *coordination ambiguity* are both based on shallow parsing, which normally has a typical accuracy of 90-95% (Kang et al, 2011), we achieve the objective of 100% recall. Similarly, for *modal adverbs* and *passive voice*, we achieve 100% recall, although these patterns employ POS tagging, which has an accuracy around 97% (Manning, 2011). Two of the patterns that employ only lexical-based pattern matching, namely *missing reference* and *undefined term*, also achieve 100% recall. Lower values of recall are instead achieved for the patterns associated to *vague terms* (67.74%), *excessive length* (60.06%), *missing unit of measurement* (50%) and *missing condition* (97.05%).

6.1.2 RQ2: Which are the cases of inaccuracy of the NLP patterns for defect detection?

Vague terms By inspecting the ten false negative defects for vague terms, VE1 found that they were all due to the absence of the quantifier *some* in the list of vague terms provided by QuARS. Hence, requirements such as the following were not marked as defective by the pattern: *In case the boolean logic evaluates the permissive state, the system shall activate* ***some*** *redundant output* – which output shall be activated? VE1 resolved the problem by simply adding the term *some* to the list of vague terms. Since also p^D was particularly low (56.75%), VE1 inspected the false positives and saw that they were due to domain-specific terms, namely ***raw*** *data*, ***hard*** *disk*, ***short***-*circuit*, ***logical*** *or*,

***logical** and*, ***green** LED*. These terms were used to discard false positives in future analysis.

Excessive length By inspecting the false negative cases for excessive length, VE1 saw that they were due to a limitation of the GATE Tokenizer. For nested bullet point lists, the Tokenizer considers each item as a separate sentence. Hence, very long and deeply nested bullet point lists were not considered as sentences of excessive length. However, VE1 also argued that the length of a sentence, and the hard readability due to complex nested lists are different kinds of defects. Hence, she decided not to change the pattern for excessive length, and to consider the problem of nested lists as a defect that, at the moment, was left uncovered.

Missing unit of measurement Concerning the two false negative cases for missing unit of measurement, VE1 observed that these were due to the presence of ranges of numerical values, e.g., *[4,20]*, without the specification of the unit of measurement. To address these cases, the pattern was adjusted.

Missing condition The two false negative cases for missing condition appeared to be due to the presence of multiple *if* statements in the same sentence, with one *else* statement only, as in the following case: ***If** the initialization starts, **if** the board is plugged in and **if** the operator has sent the running command the system shall start, **else** it shall go in failure mode.* For requirements as the one presented, it is difficult to understand which specific *if* is covered by the *else* statement. Since the majority of missing condition defects were identified (66 out of 68), and considering that a VE has to manually review the requirements anyway, as required by the norm (CENELEC, 2011), VE1 decided not to add additional rules for this defect class. It could be noticed that the specific defect could be detected also with techniques that check the *readability* of the text (Collins-Thompson, 2014), an emerging topic in requirements (Ferrari et al, 2017), which is however outside of the scope of this paper.

False negative requirements It is also useful to look at the values of false negative cases fn^R and recall r^R for the requirements. These 14 false negative cases not only include those already discussed, but also cases of defective requirements that could not be identified with our patterns – but which were annotated by VE1 following the guidelines of the company. In particular, interesting cases are those in which we have *inconsistent requirements* (e.g., 1: *The system shall accept only read access to file X*; 2: *The system shall accept read and write access to file X.*) that violate guideline (f), which asks requirements to be *consistent*. Other cases are those for which we have problems of *testability* (guideline (b)), as in the case of *under-specified statements* (e.g., *The system shall go in error mode when an internal asynchronism has* *been detected*; asynchronism among which components?), or *incomplete statements* (e.g., *The system shall make available its internal status*; through which interface?). Finally, other cases are those associated to other defects of completeness of the requirements document, as in the case of requirements for which it is expressed only the best-case scenario, and not the worst-case (e.g., *The system shall go at runtime state from power off state in 3 minutes in the best case.*; which is the requirement for the worst case?). Although some false negative cases were found, the evaluation of the patterns was considered successful in terms of recall by VE1. Hence, we decided to experiment the use of the patterns on a larger requirements dataset.

6.2 RQ1, RQ2: Large-scale Study – 1st Iteration

6.2.1 RQ1: What is the accuracy of the NLP patterns for defect detection?

In Table 9 we report[6] the output of the patterns on the dataset in terms of defects identified (**D**), and in terms of defective requirements (**R**) – the other columns of the table will be discussed in Sect. 6.3.

We see that the majority of the defects are due to *passive voice*. This is in line with the results of Femmer et al (2014). The use of passive voice appears to be a sort of writing style of these requirements, since 824 out of 1866 (44%) include this defect. However, the most interesting – and disappointing – aspect comes from the evaluation presented in Table 8. The number of false positive requirements is extremely high, and the precision is only 5.81%. This value is comparable with the precision obtained through a random predictor (for which $p^R = r^R = 93/1866\% = 5\%$, see Alvarez (2002)). Hence, it appears not acceptable if the tool needs to be used in a real-world setting. Furthermore, also the value of r^R (85.39%) is slightly lower if compared with the one obtained in our preliminary study, for which $r^R = 88.33\%$.

6.2.2 RQ2: Which are the cases of inaccuracy of the NLP patterns for defect detection?

In this iteration, we give general observations of false negative cases, which impact the value of r^R, and false positive cases, which impact on p^R. Given the low value of p^R observed, the evaluation of false positives, and their classification was systematically performed during *Large-scale Study – 2nd Iteration* (Sect. 6.3).

[6] The results presented in Table 9 and Table 8 differ from those presented in our original conference paper. When VE2 replicated the experiments performed by VE1, discrepancies in the results emerged. These were traced back to the usage of a support tool, developed by VE1 on top of GATE, to ease the analysis of the requirements. The tool introduced further manipulations, which led to incorrect numerical results. The results presented in this paper are produced based solely on the analysis of the output of GATE, and are, to the best of our knowledge, correct.

Table 8: Results for the *Large-scale Study – 1st Iteration*.

tp^R	fp^R	fn^R	p^R	r^R
76	1232	13	5.81%	85.39%

False negative cases As for the preliminary analysis, the false negative cases are due to requirements that include defects that were not considered by any of the patterns, but that violate one or more criteria adopted by the company. Interesting examples are requirements that do not fulfill the criterion of *testability* (guideline (b)), as e.g., *The system shall be in continuous operation for 24 hours a day and 7 days a week*; requirements that are not *feasible* (guideline (a)), e.g., *The core of the system shall use TCP/IP protocol in order to communicate with peripheral boards* – in this case, this requirement was considered not feasible since the only communication protocol that was considered applicable was UDP; requirements that include *inconsistent* statements (guideline (f)), e.g., *The brake symbol shall be able to show the following colors: Green when the brake is not active, Grey when the brake is not active.* Overall, these cases show that there is a variety of defects of semantic nature that are hardly identifiable with the applied NLP techniques – which focus on lexical and syntactic aspects –, and hence require a human expert to accurately assess them.

False positive cases VE1 inspected the output of the tool, and saw that part of the false positive requirements were, in her opinion, actually defective. For example, the following requirement marked as *accepted*, was evidently defective due to several vague terms (highlighted in bold): ***Depending on*** *the technical or functional solution selected, there shall be time parameters in the control system, that the Purchaser shall be able to adjust during operation in order for the registration/deregistration to be made* ***as effectively as possible***.[7] In other terms, her opinion was that VE3, when evaluating the requirements, actually tolerated several linguistic defects, and marked as *rejected* only those requirements that appeared to include severe conceptual defects. When consulted by VE1, VE3 observed that he also had an in-depth knowledge of the project of the requirements, which allowed him to disambiguate, or tolerate, certain defects. To assess how many of the false positive cases could be considered as linguistic defects from the point of view of a more strict annotator that did not have prior knowledge of the project, a second annotation process was performed to evaluate the false positive cases (*Large-scale Study – 2nd Iteration*, Sect. 5.5).

[7] The requirement was not rejected since it was clarified by other subsequent requirements. This violates the guideline (c) that require requirements to be stand-alone, but the defect was not considered crucial.

6.3 RQ1, RQ2: Large-scale Study – 2nd Iteration

6.3.1 RQ1: What is the accuracy of the NLP patterns for defect detection?

Table 9 reports the results of this phase. For each defect class, the precision reaches an average value of **72.81%** for what concerns the number of defects (average of different p^D). Overall p^R resulting from the application of all the patterns together, raises from the 5.81% of Table 8, to **77.37%**.

6.3.2 RQ2: Which are the cases of inaccuracy of the NLP patterns for defect detection?

From the results presented in the previous section, there is still a significant amount of false positive cases that should be noticed. Part of these cases are *systematic*, and they can be discarded with additional patterns. Here we will discuss relevant examples of false positive cases for each class, specifically focusing on the systematic cases, and mentioning non-systematic ones when this is considered relevant.

Table 9: Results for the *Large-scale Study – 2nd Iteration.*

Defect Class	**D**	**R**	tp^D	fp^D	p^D
Anaphoric ambiguity	391	342	198	193	50.64%
Coordination ambiguity	261	215	190	71	72.80%
Vague terms	857	580	392	465	45.74%
Modal adverbs	478	379	333	145	69.67%
Passive voice	1317	824	888	429	67.43%
Excessive length	13	13	13	0	100%
Missing condition	185	147	127	58	68.65%
Missing unit of measurement	0	0	0	0	-
Missing reference	2	1	2	0	100%
Undefined term	61	57	49	12	80.33%
Average					**72.81%**
			tp^R	fp^R	p^R
Requirements			1012	296	**77.37%**

Anaphoric ambiguity The majority of the false positive cases for anaphoric ambiguities are due to the usage of the pronoun *it* in its impersonal form, especially in the expression ***It** shall be possible [...]*. This expression, and its variants – *it shall also be possible*, *it should be possible*, etc. – is often used as a preamble in the requirements of the company. These cases are systematic sources of false positives, and appropriate patterns can be defined to discard them.

The remaining, non-systematic cases, include situations in which the referent of the pronoun is disambiguated by the context, as in the following requirement: ***Trains** that arrive on the automatically controlled **stretches** shall* *continue to be directed to* ***their*** *correct destinations*. The pronoun *their* is clearly referred to the trains, but the pattern P_{ANA} recognises two nouns (i.e., *trains* and *stretches*), to which the pronoun may refer. To detect these non-systematic false positive cases, machine learning approaches, such as those applied by Yang et al (2011) should be applied.

Coordination ambiguity The false positive cases for coordination ambiguity, in line with those identifyied by Chantree et al (2006), are non-systematic cases, in which the potentially ambiguous fragment is disambiguated by the context. For example, consider a requirement such as: *It shall be possible to print out the* ***whole timetable or part*** *of it*, in which the fragment in bold is detected by means of pattern P_{CO_2}. In this requirement, it is clear that the adjective *whole* refers solely to the noun *timetable*. Similarly, consider the following requirement: *A train can consist of one,* ***two or three cars for services between Station A and*** *Station B*, in which the fragment in bold is detected by means of pattern P_{CO_1}. Also in this case, it is clear that the conjunctions *and* and *or* refer to their nearby terms. However, these cases are non-systematic, and can hardly be detected by means of rule-based patterns. Other heuristics, such as those presented by Chantree et al (2006) should be used.

Vague terms A large number of false positive cases (465) is identified for this defect. These cases can be partitioned into the following typical situations:

1. **Lexical Ambiguity:** the vague term is *lexically ambiguous* (Berry et al, 2003). For example, the term *light*, considered as adjective, is vague, but when playing the role of noun, as in the requirement *Yellow Stop* **lights** *do not have to be monitored*, is not vague. Cases such as the one in this example can be systematically detected by applying POS tagging, and considering a term as vague only if it plays the role of adjective. A similar systematic case, which can be addressed with the same approach, is the case of the term *sound*, as in the requirement fragment *Blue arrows, and their associated* ***sound****, shall not be presented to the driver [...]*;
2. **Domain-specific Term:** a vague word is part of a domain-specific multi-word term, as for the term *distant* of the following example: *The operator shall use "****distant*** *signalling distance" to apply the brake*. Another interesting case is the term *near* in the typical railway expression ***near*** *miss* – indicating an unplanned event that has the potential to cause, but does not actually result in human injury. To discard these cases, techniques for multi-word term identification (Bonin et al, 2010) may be applied. Otherwise, a list of stop phrases to be ignored can be defined based on the false positives identified. In our case, this second option will be chosen.
3. **Accepted Expressions:** the term *possible* is used in the phrase *It shall be* ***possible*** *[...]*, considered an accepted requirement preamble within the company, as previously mentioned.

4. **Internal Clarification:** the vague term is later clarified with the specification of numerical quantities, as in the following fragment: *[...] for a* ***short*** *stretch (maximum 3 meters) on tramcars [...].* In this case, the term *short* is clarified by the phrase *maximum 3 meters.*
5. **Domain Clarification:** the vague term is clarified by the domain, as in the case of the term *adjacent* in the following requirement: *In the case of a train passing* ***adjacent*** *to a level crossing, each train shall register its own priority.* Physical adjacency among elements in the railway line is a well defined concept in the domain. However, we found also cases in which the term *adjacent* was considered vague, as in the fragment ***adjacent*** *track*, in which it is not specified whether the referenced track is on the left-hand or on the right-hand side.

The first three cases can be systematically detected. By contrast, the last two are hard to be detected in a systematic manner. Indeed, although for case 4, patterns that check numerical quantities nearby the vague term can be defined, but it is not sure how "nearby" should be intended. In addition, these false positive cases are rather easy to discard, and, for this reason, patterns will not be defined to address these cases.

Modal adverbs For modal adverbs the great majority of the false positive cases are due to the usage of the terms *manually* and *automatically.* These terms are not considered defective in the context of the requirements, since they are used to distinguish between the duties of the system (*automatically*), and the duties of the operator (*manually*). The remaining false positive cases are due to the usage of the term *only.* Consider the following requirement: *In case there are two coupled points the system shall select* ***only*** *the point with identifier equal to 1.* Here, the term *only* is used to distinguish between multiple choices. Since the term *only*, especially when misplaced, may be ambiguous, as noted by Berry et al (2003), the usage of this term cannot be regarded as a systematic source of false positives. An exception in this sense is the occurrence of *only* in the fragment *information purposes* ***only***, an expression frequently used in the requirements. When *only* occurs in this fragment, it can be considered as a systematic false positive case.

Passive voice For the false positive cases of this class, we can identify four typical situations, listed below:

1. **Irrelevant Actor:** the actor performing the action is sometimes considered as not relevant, as in the requirement: *Air conditioning units* ***are installed*** *in some of the technical equipment areas.* This sentence provides information about a certain environment, and the reader does not need to know who installed the air conditioning units. Similar cases are those in which the passive voice *is connected*, or *is disconnected* are used;
2. **Implicit Actor:** the actor – often, the system or the operator – can be inferred from the context, as in *Error signals shall* ***be displayed*** *in the MMI above the speedometer* (the actor is the system), or *The emergency*

brake restore shall be performed with the green signal (the actor is the operator).

3. **Explicit Actor:** the actor is actually expressed, as the passive voice is used in conjunction with prepositions (e.g., *by*, *from*), after which the actor is clarified, as in the following example: *All views shall **be developed** by the Supplier in consultation with the Purchaser.*
4. **Intransitive Verb:** the passive voice is used with intransitive verbs, such as the verb *log-in*, e.g., *if a workstation fails and the operator **is still logged in** [...]*;

The first two cases cannot be identified systematically. However, the latter two can be, in principle, identified with appropriate patterns, which detect the prepositions *by* and *from* in conjunction with passive voice (case 3), or which identify intransitive verbs (case 4). However, since the number of these cases was considered negligible, VE2 decided not to implement these patterns.

Missing condition False positives for this defect class occur when the term *if* is not used to express a condition over the system behaviour. For example, the requirement *The system shall check if there is a train in the route* does not require an *else* statement. In other cases, the *else* condition is expressed in another requirement, e.g., *1: If the precondition satisfies all initialization check the system shall set its internal state to running; 2: In case an initilization check fails, the system shall set its internal state to failure.* These cases can hardly be detected with patterns, and require the knowledge of the context to be disambiguated.

Undefined term The entirety of the false positive cases for undefined terms are due to the identification of units of measures, or known acronyms in their plural forms, such as, e.g., kVA, dB, LEDs. A list of known unit of measurement and known acronyms can easily be defined to discard these cases.

6.4 RQ3: Large-scale Study – 3rd Iteration

RQ3: What is the precision of NLP patterns for defect detection when complemented with discard patterns?

Table 10 reports the results obtained when applying the discard patterns. We notice a substantial increase, in terms of p^D. In particular, compared with the results of Table 9, p^D increases by 22.69% for anaphoric ambiguity, by 24.89% for vague terms, by 11.75% for modal adverbs, and by 17.67% for undefined term. Overall, the average p^D raises to 81.36% (an increase of 8.55% with respect to Table 9), and also p^R increases by a non negligible 5.79%. This increase of precision saves, in principle, a considerable amount of checks to the VE, who has to vet a lower number of requirements. More specifically, if we look at the values of fp^R in Table 10 (296) and in Table 9 (205), we see that 91 requirements do not have to be vetted after the introduction of the discard patterns.

Table 10: Results for the *Large-scale Study – 3rd Iteration*.

Defect Class	**D**	**R**	tp^D	fp^D	p^D
Anaphoric ambiguity	270	251	198	72	73.33%
Coordination ambiguity	261	215	190	71	72.80%
Vague terms	555	384	392	163	70.63%
Modal adverbs	409	330	333	76	81.42%
Passive voice	1317	824	888	429	67.43%
Excessive length	13	13	13	0	100%
Missing condition	185	147	127	58	68.65%
Missing unit of measurement	0	0	0	0	-
Missing reference	2	1	2	0	100%
Undefined term	50	47	49	1	98%
Average					**81.36%**
Requirements			tp^R	fp^R	p^R
			1012	205	**83.16%**

Table 11: Results for the *Large-scale Study – 4th Iteration*, SREE vs Patterns.

Tool	tp^R	fp^R	fn^R	p^R	r^R
SREE	86	1492	3	5.45%	96.63%
Patterns	76	1141	13	6.24%	85.39%

As noticed in Sect. 6.3, the majority of the remaining false positive cases cannot be systematically detected, and require the judgment of a human assessor. These types of situations can be potentially addressed through statistical techniques, as, e.g., Chantree et al (2006) and Yang et al (2011). Typical examples have already been reported in Sect. 6.3.

6.5 RQ4.1: Large-scale Study – 4th Iteration

RQ4.1: What is the accuracy of SREE with respect to the NLP patterns for defect detection complemented with discard patterns?

Table 11 compares the performance of the SREE dictionaries and our patterns against the annotations of VE3. From the table, we see that SREE outperforms our patterns by 11.24% in terms of recall on the requirements originally annotated by VE3, and its precision is 0.79% lower. Hence, SREE dictionaries may contain terms that help to identify defective requirements that were not detected through our patterns, and were therefore part of the false negative cases issued. On the other hand, a 0.79% gap in terms of precision, implies that 351 additional false positive requirements (fp^R) are generated by SREE with respect to our patterns.

Let us first analyse the false negative cases of our patterns that are detected through the SREE dictionaries, and then we will investigate the issue of precision.

A representative example of the requirements detected through SREE, and not with our patterns, is the following one: ***Normal*** *and abnormal changes in* *the status of the Facility shall warrant special treatment [...].* VE3 rejected the requirement, and stated that *"Normal and abnormal changes" are not defined and shall be agreed.* SREE identifies this requirement as defective, since its dictionary for the *vague* class includes the term *normal.* On the other hand, it is worth noticing that SREE dictionaries do not include the term *abnormal*, which is also a defective term, according to the statements of VE3. A similar case is the following requirement: *When the driver follows indications as to the **maximum** speed limit the Facility shall not cause braking that produces jolty and uneven driving.* The requirement was marked as rejected by VE3, because it does not fulfill the criterion of *testability* (guideline (b)). This is due to the presence of the adjectives *maximum*, *jolty* and *uneven.* Here, the SREE dictionaries correctly detects the vague term *maximum*, but do not detect the defective terms *jolty* and *uneven.* Hence, although including the SREE dictionaries in our patterns can help to increase the recall, novel terms may be needed in the future to address other, previously unseen, defects.

Another interesting aspect concerns other requirements that (a) are marked as defective by SREE, (b) are marked as rejected by VE3, but for which (c) the cause of the rejection is not the defect indicated by SREE. Exemplary cases are mostly related to the usage of plurals, which have 3377 occurrences in 1250 requirements (see Table 12, discussed in Sect. 6.6). An example is the following requirement: *It shall be possible to turn **trains** at the intended turning points without restriction.* SREE identifies the source of the defect in the plural term *trains.* However, VE3 marked the requirements as rejected because it violates the criterion of *testability* (guideline (b)). This is due to the expression *without restriction*, which does not allow the definition of a finite number of tests to verify the requirement.

Of course, there are entire defect classes considered by our patterns, which are not detected by the dictionaries of SREE, such as *passive voice*, *missing condition*, *missing reference*, *missing unit of measurement*, *etc.* Given these observations, SREE dictionaries can be considered as complementary to our patterns. Still, SREE and our patterns altogether are insufficient to detect all the potential defects, and should be complemented with additional terms, as, e.g. *jolty* and *uneven.*

As mentioned, a high amount of false positive requirements was returned by SREE with respect to our patterns. This suggests that SREE may be less precise also at the level of defects. On the other hand, these false positive cases may conceal defects that were not considered by VE3.

Therefore, it was decided to evaluate the potential degree of precision for the *single defects* identified by SREE. An analysis of the false positive cases was performed at the level of the single defects, similar to the one applied on the output of our patterns during *Large-scale Study – 2^{nd} Iteration.*

Table 12: Results of the *Large-scale Study – 5th Iteration*. False positive evaluation for *SREE-reduced* dictionaries.

Defect Class	**D**	**R**	tp^D	fp^D	p^D
Continuance	181	155	41	140	22.65%
Directive	123	102	0	123	0%
Optional	102	92	26	76	25.49%
Incomplete	32	31	2	30	6.25%
Plural	3377	1250	*6*	*125*	*4.58%*
Quantifier	308	264	25	283	8.11%
Vague	931	665	111	820	11.92%
Average					**11.29%**

6.6 RQ4.2, RQ4.3, RQ4.4: Large-scale Study – 5th Iteration

6.6.1 RQ4.2: What is the precision of SREE for the defects in its scope?

Table 12 reports the results of the analysis of the false positive defects. The average value of p^D is 11.29%, which indicates that a large amount of false positive cases are issued, which is much lower, compared with the 81.36% obtained through our patterns (Sect. 6.5)[8].

6.6.2 RQ4.3, RQ4.4: Which additional defects can be identified with SREE, and which are the false positive cases?

Below, we provide an analysis of the true positive and false positive cases.

Continuance The continuance class includes terms that, when present, indicate a reference between a statement and a previous one (e.g., *in addition*, *in particular*), or a subsequent one (e.g., *following*, *below*). True positive cases occur when the referred statement is absent, and, therefore, the requirement is incomplete. The number of these cases is not negligible, and are all associated to the terms *as follows* and *below*. False negative cases occur anytime the referred statement appear in the requirement. These cases occur especially when the referent is a *previous* statement, and the terms *in addition* and *in particular* are used.

Directive The directive class includes terms that indicate the presence of a reference to an element in the requirement (e.g., *e.g.*, *i.e.*) or in the document (e.g., *figure*, *table*). As for the continuance class, true positives may occur when the referred element is absent, while false positive occur when the referred element is present. In the considered requirements, no true positive case was identified.

[8] The value of p^R that considers the analysis of the false positive cases for the SREE dictionaries cannot be provided, since we analysed only a subset of the defects for the *plurals* class. However, the average value of p^D gives a clear indication of the precision of SREE at the level of defects.

Incomplete The incomplete class includes terms that may indicate a form of internal incompleteness of the requirement (e.g., *TBD*, *to be defined*). The dictionary of this class raises a limited number of defects (32). Indeed, expressions as, e.g., *TBA*, *TBD* do not occur in the requirements, and the great majority of the false positive cases occur when the term *in addition* is used – a term that is included also in the continuance class. Another typical case of false positive is the following requirement fragment: *[...] functions shall be performed in a secure way, **as defined** in the CTC security requirements.* Here, the requirement is not incomplete, since it refers to another document in which the required information is available. Instead, the cases evaluated as true positives are similar to the following one: *All alarms [...] shall be shown in track plan views **as specified** above.* Here, the problem is with the term *above*, rather than with the term *as specified*, since the VEs could not find the referred information in the document. However, the defect was considered a true positive by the VEs, since the tagging of the term *as specified* allowed the identification of the defect.

Optional The optional class includes terms that indicate subjective optionality. Anytime an expression such as *if needed*, *if necessary*, *if appropriate* occurred, this was marked as a true positive case. Similarly, many true positive cases occur with the term *either*, as in the requirement: *A cable run shall be laid on **either** side of the track.*

False positive cases occur when terms such as *either*, or *neither*, are used in the expressions *either [...] or* , or *neither [...] nor.* Another typical, systematic false positive case occurs with the usage of the term *in case of*, when this expresses a condition that depends on actions that are external to the system, as in: ***In case of** a restart of the system [...].*

Plurals Plurals are ambiguous when they are used to describe a property of a set or sets, and it is not clear if the property is that of each element or of the whole set (Berry and Kamsties, 2005), as in the requirement fragment *[...] **printers** shall have a sound [...].* In the considered sample of 50 defective requirements for the plurals class, cases such as this one were extremely rare. A large amount of false positive cases was instead observed.

Typical false positive cases belong to two classes. The first class includes lexically ambiguous verbs used in third person singular form, as, e.g., *means*, *passes*, *leaves.* The second class includes cases in which the plural term indicates a set of objects or subjects, such as *trains*, *boards*, *tracks*, *operators*, etc., and it is clear from the context that the requirement refer to all the elements in the set, as in the following fragment: *Control **orders** that are executed by **operators** shall be registered [...].* Since the requirements are high-level system requirements, the use of plurals in the form exemplified is rather common, and accepted by the VEs.

Quantifier Quantifiers that express quantities in a vague form such as *few*, *little*, *many*, are included in the quantifier class. The occurrence in the requirements of these vague terms was always considered by the VEs as a true positive defect. False positive cases are due to universal quantifiers, such as *all* or *any*. Indeed, although, as noted by Berry and Kamsties (2005), these terms may be source of ambiguity (e.g., *All lights have a switch* – one switch for each light, or a common switch?), in the considered requirements these terms are not used in ambiguous forms. Instead, non ambiguous requirements fragments such as the following are common: *[...] **all** equipped tramcars [...] shall be able to operate on **all** track networks [...].*

Vague The vague class includes additional terms with respect to the *Vague* dictionary of our patterns. Part of these terms appear to be useful to identify extremely vague requirements that were not identified through our patterns. A representative example is the following requirement, which includes two vague expressions: *Communication shall **as far as** possible be redundant, with separate cable runs, for the **various** communication links.* False positive cases are mainly due to the usage of terms such as *also*, and *but*, which are rather frequent in the requirements, but are not considered sources of vagueness by the VEs. Indeed, the presence of these terms sometimes indicates that a requirement includes more than one statement, as in the fragment: *the [...] system shall not be reused **but** shall be dismantled [...].* However, since the considered requirements are high-level system requirements, the VEs accepted these situations.

6.6.3 General Observations

From this analysis, we see that additional defects, which were not previously considered by our VEs, are actually detected thanks to SREE. This confirms that SREE may play a complementary role with respect to our patterns. On the other hand, the value of precision of SREE, at the level of defects, is poorer than the precision of our patterns, i.e., a larger number of false positive cases is issued. However, this numerical difference should be considered with care. Indeed, there are two main reasons that explain and justify this result:

1. **SREE Philosophy:** the philosophy of SREE, as we interpret it through its usage, is to identify terms that, when present, may indicate that also a defect *may* be present. If the defect is not present, it is easy for the analyst to vet the requirement. Representative examples in this sense are the terms in the *continuance* class: terms such as *as follows* and *below* were judged as particularly useful by the VEs to detect incomplete requirements, although their occurrence was not always associated to a defect. The VEs said that vetting the false positive cases was straightforward for this class. Hence, the low value of precision was sufficiently counter-balanced by the usefulness of the terms included in the defect class.
2. **Subset of SREE:** a subset of SREE dictionaries was used, instead of the whole SREE. Hence, the comparison cannot be considered complete. However, our goal in this case study was not to identify the best tool for

defect detection, but rather to investigate whether additional defects could be found by means of the SREE dictionaries. This goal also mitigates a potential annotators' bias that may have occurred in the evaluation of the false positives of SREE dictionaries. Although this bias cannot be totally eliminated in the context of our case study, our patterns, as well as the SREE dictionaries, are available for the research community, who can independently compare the different strategies.

7 Threats to Validity

In this section, we discuss threats to validity according to the structure recommended by Runeson et al (2012).

Construct Validity Objective and widely used metrics, i.e., precision and recall, were used in this work to assess the accuracy of the adopted NLP technologies. To derive measures of precision and recall, subjective evaluations were performed by VE1, VE2, and VE3 during the Dataset Annotation and Output Annotation tasks. In the Pilot Study, only VE1 annotated the dataset, and no countermeasure was taken to assess the validity of the annotation, given the preliminary nature of the study. Similarly, in the *Large-scale Study – 1st Iteration*, only VE3 annotated *D-Large*, and the same annotation was used for the *Large-scale Study – 4th Iteration.* On the one hand, also in the real-world context of the company, requirements review is performed by one subject, and the subjectivity threat can be considered as partially mitigated by the realism of this annotation. On the other hand, the Output Annotation on *D-Large*, was independently performed by VE1 and VE2, and the inter-rater agreement was computed by means of the Cohen's Kappa. The agreement resulted in $k = 0.82$ (almost perfect) for *Large-scale Study – 2nd Iteration*, and $k = 0.79$ for *Large-scale Study – 5th Iteration* (substantial). Therefore, we believe that the threat is further mitigated by these measures of agreement, at least for those requirements that were produced as output by the NLP patterns. Therefore, construct validity threats are mitigated for *Large-scale Study – 2nd, 3rd and 5th Iteration*, while they are only partially mitigated for *Pilot Study*, and *Large-scale Study – 1st and 4th Iteration*, in which only one subject was involved in the annotation process.

Internal Validity The main threats to the internal validity of the study are due to the personal objectives of the involved subjects, which may have had an impact on the results. Indeed, the annotations performed by VE1 and VE2 in the tasks in which they were involved may be biased by their need to show that the implemented patterns were successful, hence annotating as defective also requirements that were not. In the case of the *Pilot Study*, this threat is mitigated by the fact that the annotation was performed *before* applying the patterns, and hence without exactly knowing their output. In the *Large-scale Study* iterations, the threat is mitigated by (a) by the pragmatics of the case study, and (b) the independent Output Annotation process performed. Indeed, since VE1 works as VE in the company, she is also interested on improving her job, besides showing that the implemented technology is effective. VE2 may be less keen to this type of integrity, since she is not part of the company anymore. However, since the Output Annotation task was always performed independently by the two VEs, we argue that this threat is sufficiently controlled. Furthermore, as noticed in Sect. 6.6.3, since this threat cannot be totally mitigated, we share our patterns so that other researchers can apply them to their contexts, and check their effectiveness. It should be noted that the annotations of VE3 are not subject to this threat, since they were performed before this work was conceived. Validity issues related to the discrepancies between the annotations performed by VE3 compared to the ones of VE1 and V2, are discussed in the *External Validity* paragraph, since we argue that the annotations represent different contexts, from which different generalisation criteria may apply.

Another internal validity threat is associated to the tool-suite initially used by VE1 in the *Large-scale Study – 1st and 2nd Iterations*, to compute the data for the case. Indeed, she used an internally developed tool on top of GATE to produce the results. To mitigate potentially unsound manipulation of the data by this prototype tool, part of *Large-scale Study – 1st and 2nd Iterations*, were replicated by VE2, with the support of GATE only. Discrepancies in the results were observed, and root causes were analysed. The rest of the analysis were performed by means of GATE only. Since GATE is a widely used tool – see the list of companies using GATE[9] and, e.g., Arora et al (2015) and Derczynski et al (2015), for relevant scientific works in which GATE was employed –, we believe that the results produced with its support are correct.

External Validity Our discussion on the external validity of the study is loosely based on the principles of case-based generalisation proposed by Wieringa and Daneva (2015), and of similarity-based generalisation proposed by Ghaisas et al (2013). Specifically, we describe the main architectural aspects of our study, i.e., domain, requirements, subjects, that can be considered as a term of comparison for other studies. In this way, other researchers and practitioners can reason by analogy, and possibly profit from our results (Ghaisas et al, 2013).

– *Domain:* our study covers a company of a specific domain, i.e., the railway domain. In Europe, railway companies have to follow the general guidelines of the CENELEC norms (CENELEC, 2011), and their work practices at process level can be considered comparable. Furthermore, the railway domain is characterised by a limited number of suppliers, who often deal with the same customers – i.e., the national or private railway companies, who provide infrastructure, and services to passengers. This increases the homogenisation of processes and, in part, requirements documents. While we cannot generalise our results for any type of domain, we argue that

[9] `https://gate.ac.uk/commercial.html`

similar results may be obtained in other railway companies. On the other hand, the following limitations to the external validity of our results shall be considered.

- *Requirements:* the requirements considered in the study have been selected by VE1, with the support of the company, as benchmarks to represent typically defective requirements of the firm. VE1 and VE2 admits that, depending on the subjects involved in the production of requirements, the documents may have different degrees of quality, and the documents belonging to the study are requirements of lower quality than average. Furthermore, along the process, system requirements such as those analysed are normally refined into lower level requirements. Hence, the results produced shall be considered representative for (a) system requirements, (b) requirements with a poor degree of quality. Since the requirements concerns several types of railway signalling systems, they are sufficiently representative of the types of product developed in railways.
- *Subjects:* Overall, three VEs were involved in this study. The sample is limited, but it shall be considered that all the VEs are normally subject to the same company practices and process, and can therefore be considered representative VEs for the company. Considering the characteristics of the railway domain mentioned above, they can be considered, to a certain extent, also representative of VEs in railways. Discrepancies were observed between the annotations performed by VE3 on *D-Large* during *Large-scale Study – 1st Iteration*, and the annotations on the output of the patterns performed by VE1 and VE2, during *Large-scale Study – 2nd Iteration*. In principle, the discrepancies may be associated to the different degree of experience of the subjects. VE1 and VE2 had 3 and 2 years experience, respectively, while VE3 had 10 years of experience. We believe that the discrepancies observed are only partially associated to the experience. Instead, we believe that the discrepancies are due to the differences in terms of contextual knowledge, and goals. VE3 had in in-depth knowledge of the project that allowed him to disambiguate, or tolerate, certain defects, and focused on severe conceptual problems. Instead, VE1 and VE2 did not have any prior knowledge on the project, and focused on linguistic aspects, given the research-based, exploratory nature of their work.
  For these reasons, the different iterations have different degrees of external validity – notwithstanding the construct validity threats already discussed. Specifically, *Pilot Study*, and *Large-scale Study – 2nd, 3rd, and 5th Iterations* can be considered representative for those cases in which the annotation is performed by VEs who do not have prior knowledge of the project of the requirements, and focus on *linguistics* defects. Instead, *Large-scale Study – 2nd and 4th Iterations* are representative for those cases in which the annotation is performed by a VE who has an in-depth knowledge of the project, and focuses on *conceptual* defects.

As mentioned, our results can be generalised to other domains only to a limited extent. Our work focusses on a single railway company, and railway companies have a well-defined processes to follow, that is not shared by other context. The degree of rigour of the railway process is comparable to the one employed in the avionic sector, in which the DO-178C norm applies for software development (RTCA Inc. and EUROCAE, 2012). However, the products developed in railways and avionics are highly different, and use domain specific terminology. Many of our patterns are domain independent, but, given the large variability of NL, and of domain specific NLs, the generalisation of our results to other domains requires further research.

Reliability The results provided are mainly quantitative, and we argue that a common understanding on their meaning was achieved when the values of precision and recall had to be computed. Concerning the qualitative data, these were provided by the VEs and were refined with the support of NLP-E. We argue that this interaction increased the reliability of the qualitative results.

8 Lessons Learned

From the experience presented in this paper, a set of lessons learned were discussed among the authors, and are reported below.

Domain-customisable NLP Tools Our experience shows that NLP technologies are available for requirements analysts with limited NLP training, and that these technologies can be proficiently used for the detection of several typical requirements defects. Rule-based NLP patterns tend to generate large numbers of false positives (Chantree et al, 2006; Yang et al, 2011). If the results come from a tool that the requirements analyst cannot control, the analyst is likely to distrust the tool. Instead, if the analyst understands the inherent principles of the tool – and implementing the tool is a proper way for understanding its principles –, they can understand its weaknesses and use it at its best. Furthermore, it is also important that domain experts develop the tools, since, to reduce the amount of false positive cases, tailoring the patterns for the specific needs of the domain is required. If the VEs implement the patterns, they can customise them according to the language used in the domain, as, e.g., to account for terms such as *raw data*, *hard disk* (Sect. 6.1), and phrases such as *it shall be possible* (Sect 6.3). The introduction of the discard patterns, to remove systematic false positive cases, allowed an increase of the average p^D from 72.81% to 81.36% (Sect. 6.3). It should be noticed that, if a company defines a set of patterns to be applied for defect detection, a maintenance cost should be taken into account, since, as any software tool, patterns may need to evolve. While for COTS tools the software house who develops them takes care of their evolution, and maintenance costs, the railway company has to take the burden of maintenance in case of internally developed tools.

Requirements Language Counts Looking at the large number of passive voice defects in *Large-scale Study – 2nd Iteration*, it appeared that the use of passive voice was a form of writing style. As a consequence, the patterns generated a large number of detected defects (i.e., 1317). This tells us that, to effectively use NLP, one cannot simply implement appropriate defect detection patterns: one should change also the language adopted in the requirements, to make it more error free, so that the VE can focus on a smaller amount of defects. For this reason, we argue that NLP tools should be first used by the requirements editors, to limit the amount of poor writing style, and only *afterwards* by a VE. However, this is not always practicable, especially in those cases in which requirements are produced by the customer, and assessed by the company who has to develop the product. As acknowledged by the company, the requirements considered in this study are particularly rich in defects, also with respect to other requirements of the company. However, it is worth noting that, upon suggestion of NLP-E, and taking inspiration from the work of Terzakis and Gregory (2016), VE1 is currently involved in a mentoring program within the company, to educate the requirements authors towards the production of higher quality requirements.

Requirements Level Counts During the analysis of the false positive cases of SREE, a large number of plurals (3377) was identified, which were tolerated in most of the cases. Furthermore, also the presence of conjunctions such as *also* and *but*, which indicate non-atomic requirements, was tolerated in these requirements. This was motivated by the level of the requirements. The considered dataset was composed of high-level system requirements, for which, according to the VEs, a certain degree of generality can be accepted. These requirements will be refined into lower-level technical requirements, for which a greater degree of precision is expected. As we notice in a recent work (Ferrari et al, 2017), this suggests that requirements at different degrees of abstractions may need different treatments. More specifically, patterns to check presence of plurals, as well as *also* and *but* conjunctions, may need to be applied for low-level requirements, while they do not need to be used for high-level ones.

Validation Criteria Count Considering the *Large-scale Study – 1st and 2nd Iterations*, we saw that a large part of the false positive cases encountered in the *Large-scale Study – 1st Iteration* could be associated with a weaker validation performed by VE3, who did not focus on linguistic defects, but more on severe conceptual defects, also given his in-depth knowledge of the project. For this reason, the results obtained in terms of precision were extremely poor. When changing criteria, p^R varied from 5.81% to 77.37% (Sect. 6.3). Hence, to perform an appropriate validation of rule-based NLP patterns, it is advisable to start from an annotated dataset that has been defined *knowing* the classes of defects that will be checked by the patterns, and specifically stating that the focus is on *linguistic* defects. Otherwise, the results might be misleading. This observation might appear counter-intuitive, since we suggest to adapt human operators to tools. However, when dealing with the complexity of NL, we argue that the adaptation between humans and NLP tools should be bi-directional.

NLP is Only a Part of the Answer In our large-scale study, several false negative cases occurred, which can hardly be detected with NLP. These are examples of conceptual defects that require a human with knowledge of the domain and of the specific project. In recent years, NLP technologies have seen radical progress (Goth, 2016). Linguistic tasks at the semantic level, such as, e.g., question-answering, became possible. However, the *pragmatic* nature of ambiguity (Ferrari et al, 2016), and the contextual knowledge needed to understand a requirements document, make the problem of automatic defect detection in requirements hardly solvable with current technologies. Therefore, NLP represents only a part of the answer to defect detection, while the other part is represented by human analysts with domain expertise. It should also be considered that relying on a tool for defect detection may also change company practices, in that a VE may rely too faithfully on the tool's output. This reasonable hypothesis requires further empirical investigation, but its potential implications should be considered when introducing an automated tool to support practices that are normally manually conducted.

Statistical NLP vs Lexical Techniques Our patterns make use of POS tagging and shallow parsing, which are statistical techniques that can hamper the objective of 100% recall (Berry et al, 2012). However, in Sect. 6.1, we showed that 100% recall was achieved for those patterns that used these techniques, while it was *not* achieved for the pattern adopted for *vague terms*, which uses a lexical based approach. Hence, we argue that the argument in favour of a"dumb" lexical-based defect detection approach instead of an approach that leverages statistics-based techniques (Berry et al, 2012) should be partially revised. If one wants to use lexical-based detection approaches, then one should use only defect indicators belonging to closed word classes (e.g., pronouns, conjunctions). Instead, if one uses open word classes (e.g., adjective, adverbs), the problems are not different from those that *might* emerge with statistical techniques. As statistical techniques may fail, also lists of dangerous adjectives and adverbs may fail, because they might not include words that were not considered until they appear in the requirements (as e.g., the word *some*, as noted in Sect. 6.1, or the words *jolty* and *uneven*, as noted in Sect. 6.5).

9 Conclusion, Implications for Practice and Future Research

This paper presents the experience of a railway signalling manufacturer in implementing a set of NLP patterns to detect defects in NL requirements. A pilot study on 241 requirements is presented, as well as a large-scale study on 1866 requirements. After a refinement of the patterns, a precision of 83.16% and a recall of 85.39% are obtained. Recall can be increased by using term-based defect detection tools such as SREE (Tjong and Berry, 2013), although at the cost of a lower precision. From this experience, we can derive a set of implications for practice and directions for future research, which are summarised below.

Implication for Practice Overall, the experience was considered extremely useful by the company. In particular, VE1 says that, after studying the literature on defect identification, and implementing the patterns, also her way of judging requirements defects became stricter. This is also one of the reasons why requirements marked as *accepted* by VE3, were afterwards *rejected* by VE1 and VE2. This implies that, while on the one hand tools have to be adapted to company practices, also company practices can be modified by tools. In our study, we also observed that an increase in the performance can be obtained by *incrementally* tuning the patterns based both on the defects encountered in practice, and through the inclusion of other defect-detection criteria from the research literature – in particular, the SREE dictionaries. Therefore, regardless of the NLP technologies used to detect defects, technologies need to be adapted to the specific language of the company, to be fruitfully used.

It should also be observed that, based on the lessons learned from the current study, VE1 is now involved in a mentoring program within the company, oriented to teach requirements authors how to write linguistically clear requirements. The idea is that editors should be aware of linguistic defects, so that the work of VEs can focus on conceptual ones. In this sense, we argue that, by working with NLP techniques for defect detection, one can have an effect also in terms of organisational learning. Another relevant implication for practice concerns the complementary role of NLP techniques, and human analysis. We observed that part of the conceptual defects present in the requirements could not be detected with the patterns, but some ignored linguistic defects could be identified by the patterns. This suggests that, although human analysts cannot be replaced, tools can help them to perform a better job.

Future Research Within the context of the industrial collaboration that made this paper possible, our future work will go towards four main directions. (1) The first direction is to understand to which extent the NLP patterns have to be tuned to analyse requirements at different levels of abstractions, and to understand which patterns are appropriate for which level. (2) The second direction is studying to which extent *language errors* – a defect not considered here, but mentioned by Berry et al (2003) – may impact on the quality of the requirements. The VEs noticed that large part of the requirements considered were not expressed in correct English, since they were written by Italian editors, who tended to use Italian syntactic constructions. However, apparently, these language errors did not have an impact on the subsequent phases of the process, since the readers of the requirements were also Italian. (3) The third direction is to evaluate the cost of using NLP techniques for defect detection, compared to the cost of manual review. Cost-based evaluation approaches suitable for our context have been recently discussed by Berry et al (2017). (4) The fourth direction is to leverage NLP technologies also for other tasks of the company, which are dominated by NL. One particular task of interest is the support towards the automated generation of summary documents from multiple sources. Indeed, the railway process produces a large amount of documents, which are often hard to navigate, and summary documents can provide a substantial help in controlling the process itself.

The directions outlined come from the needs of the company, and from the interests of the researchers involved in this case study. However, they can be considered by the research community also as inspiration for future investigation in the field of applications of NLP to requirements and technical documentation in general. An additional direction for future research triggered by the current work, but that go beyond the collaboration with the considered company, includes the extension of our results to domains that are different from railways, to assess to which extent the adaptation of NLP patterns to the language of a company can lead to improved results in terms of defect detection accuracy.

References

Alvarez SA (2002) An exact analytical relation among recall, precision, and classification accuracy in information retrieval. Tech. Rep. BCCS-02-01, Computer Science Department, Boston College

Ambriola V, Gervasi V (2006) On the systematic analysis of natural language requirements with Circe. Automated Software Engineering 13(1):107–167

Anda B, Sjøberg DI (2002) Towards an inspection technique for use case models. In: Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE'02), ACM, pp 127–134

Arora C, Sabetzadeh M, Briand L, Zimmer F (2015) Automated checking of conformance to requirements templates using natural language processing. IEEE Transactions on Software Engineering 41(10):944–968

Aurum A, Petersson H, Wohlin C (2002) State-of-the-art: software inspections after 25 years. Software Testing, Verification and Reliability 12(3):133–154

Baskerville RL, Wood-Harper AT (1996) A critical perspective on action research as a method for information systems research. Journal of information Technology 11(3):235–246

Berry D, Gacitua R, Sawyer P, Tjong SF (2012) The case for dumb requirements engineering tools. In: Proceedings of the 18th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'12), Springer, LNCS, vol 7195, pp 211–217

Berry DM, Kamsties E (2005) The syntactically dangerous all and plural in specifications. IEEE Software 22(1):55–57

Berry DM, Kamsties E, Krieger MM (2003) From contract drafting to software specification: Linguistic sources of ambiguity. URL `https://cs.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf`

Berry DM, Cleland-Huang J, Ferrari A, Maalej W, Mylopoulos J, Zowghi D (2017) Panel: Context-dependent evaluation of tools for nl re tasks: Recall vs. precision, and beyond. In: 2017 IEEE 25th International Requirements Engineering Conference (RE), pp 570–573, DOI 10.1109/RE.2017.64

Bonin F, DellOrletta F, Venturi G, Montemagni S (2010) A contrastive approach to multi-word term extraction from domain corpora. In: Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'10), pp 19–21

Casamayor A, Godoy D, Campo M (2012) Functional grouping of natural language requirements for assistance in architectural software design. KBS 30:78–86

CENELEC (2011) EN 50128:2011: Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems. Tech. rep.

Chantree F, Nuseibeh B, Roeck AND, Willis A (2006) Identifying nocuous ambiguities in natural language requirements. In: Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06), IEEE, pp 56–65

Cleland-Huang J, Czauderna A, Gibiec M, Emenecker J (2010) A machine learning approach for tracing regulatory codes to product specific requirements. In: ICSE (1), ACM, pp 155–164

Collins-Thompson K (2014) Computational assessment of text readability: A survey of current and future research. ITL-International Journal of Applied Linguistics 165(2):97–135

Cunningham H (2002) GATE, a general architecture for text engineering. Computers and the Humanities 36(2):223–254

Cutts M (1996) The plain English guide. Oxford University Press

Derczynski L, Maynard D, Rizzo G, van Erp M, Gorrell G, Troncy R, Petrak J, Bontcheva K (2015) Analysis of named entity recognition and linking for tweets. Information Processing & Management 51(2):32–49

Fabbrini F, Fusani M, Gnesi S, Lami G (2001) The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool. In: Proceedings of the 26th Annual NASA Goddard Software Engineering Workshop, IEEE, pp 97–105

Fagan ME (1976) Design and code inspections to reduce errors in program development. IBM Systems Journal 15(3):182–211

Falessi D, Cantone G, Canfora G (2013) Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques. IEEE Transactions on Software Engineering 39(1):18–44

Femmer H, Kučera J, Vetrò A (2014) On the impact of passive voice requirements on domain modelling. In: Proceedings of the 8th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'14), Art. 21, ACM

Femmer H, Fernández DM, Wagner S, Eder S (2017) Rapid quality assurance with requirements smells. Journal of Systems and Software 123:190–213

Ferrari A, Gnesi S (2012) Using collective intelligence to detect pragmatic ambiguities. In: Proceedings of the 20th IEEE International Requirements Engineering Conference (RE'12), IEEE, pp 191–200

Ferrari A, dellOrletta F, Spagnolo GO, Gnesi S (2014) Measuring and improving the completeness of natural language requirements. In: Proceedings of the 20th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'14), Springer, pp 23–38

Ferrari A, Spoletini P, Gnesi S (2016) Ambiguity and tacit knowledge in requirements elicitation interviews. Requirements Engineering 21(3):333–355

Ferrari A, Dell'Orletta F, Esuli A, Gervasi V, Gnesi S (2017) Natural Language Requirements Processing: a 4D Vision. IEEE Software (to appear)

Gacitua R, Sawyer P, Gervasi V (2010) On the effectiveness of abstraction identification in requirements engineering. In: Proceedings of the 18th IEEE International Requirements Engineering Conference (RE'10), IEEE, pp 5–14

Gervasi V, Zowghi D (2005) Reasoning about inconsistencies in natural language requirements. ACM Transactions on Software Engineering and Methodology 14(3):277–330

Ghaisas S, Rose P, Daneva M, Sikkel K, Wieringa RJ (2013) Generalizing by similarity: Lessons learnt from industrial case studies. In: Proceedings of the 1st international workshop on conducting empirical studies in industry, IEEE Press, pp 37–42

Gleich B, Creighton O, Kof L (2010) Ambiguity detection: Towards a tool explaining ambiguity sources. In: Proceedings of the 16th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'10), Springer, LNCS, vol 6182, pp 218–232

Gnesi S, Lami G, Trentanni G (2005) An automatic tool for the analysis of natural language requirements. International Journal of Computer Systems Science and Engineering 20(1):53–62

Gorschek T, Garre P, Larsson S, Wohlin C (2006) A model for technology transfer in practice. IEEE software 23(6):88–95

Goth G (2016) Deep or shallow, nlp is breaking out. Communications of the ACM 59(3):13–16

IEEE (1998) IEEE Guide for Developing System Requirements Specifications. IEEE Std 1233, 1998 Edition pp 1–36, DOI 10.1109/IEEESTD.1998.88826

ISO, IEC, IEEE (2011) ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering. ISO/IEC/IEEE 29148:2011(E) pp 1–94, DOI 10.1109/IEEESTD.2011.6146379

Kamsties E (2005) Understanding ambiguity in requirements engineering. In: Engineering and Managing Software Requirements, Springer Berlin Heidelberg, pp 245–266

Kamsties E, Berry DM, Paech B (2001) Detecting ambiguities in requirements documents using inspections. In: Proceedings of the 1st Workshop on Inspection in Software Engineering (WISE01), pp 68–80

Kang N, van Mulligen EM, Kors JA (2011) Comparing and combining chunkers of biomedical text. Journal of biomedical informatics 44(2):354–360

Kassab M, Neill C, Laplante P (2014) State of practice in requirements engineering: contemporary data. Innovations in Systems and Software Engineering 10(4):235–241

Kiyavitskaya N, Zeni N, Mich L, Berry DM (2008) Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Requirements Engineering 13(3):207–239

Kof L (2008) From textual scenarios to message sequence charts: Inclusion of condition generation and actor extraction. In: Proceedings of the 16th IEEE International Requirements Engineering Conference, (RE'08), IEEE, pp 331–332

Kof L (2009) Translation of textual specifications to automata by means of discourse context modeling. In: Proceedings of the 15th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'09), Springer, LNCS, vol 5512, pp 197–211

Kof L (2010) From requirements documents to system models: A tool for interactive semi-automatic translation. In: Proceedings of the 18th IEEE International Requirements Engineering Conference (RE'10), IEEE, pp 391–392

Landis JR, Koch GG (1977) The measurement of observer agreement for categorical data. Biometrics pp 159–174

Lian X, Rahimi M, Cleland-Huang J, Zhang L, Ferrari R, Smith M (2016) Mining requirements knowledge from collections of domain documents. In: Proceedings of the 24th IEEE International Requirements Engineering Conference (RE'16), IEEE, pp 156–165

Maalej W, Nabil H (2015) Bug report, feature request, or simply praise? on automatically classifying app reviews. In: Proceedings of the 23rd IEEE International Requirements Engineering Conference, (RE'15), IEEE, pp 116–125

Manning CD (2011) Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In: Proceedings of the 12th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing'11), LNCS, vol 6608, Springer, pp 171–189

Mavin A, Wilkinson P, Harwood A, Novak M (2009) Easy approach to requirements syntax (ears). In: Proceedings of the 17th IEEE International Requirements Engineering Conference (RE'09), IEEE, pp 317–322

Mavin A, Wilksinson P, Gregory S, Uusitalo E (2016) Listens learned (8 lessons learned applying EARS). In: Proceedings of the 24th IEEE International Requirements Engineering Conference (RE'16), IEEE, pp 276–282

Mich L (1996) NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA. NLE 2(2):161–187

Mich L, Franch M, Inverardi PN (2004) Market research for requirements analysis using linguistic tools. Requirements Engineering 9(1):40–56

Pohl K, Rupp C (2011) Requirements engineering fundamentals. Rocky Nook, Inc.

Quirchmayr T, Paech B, Kohl R, Karey H (2017) Semi-automatic software feature-relevant information extraction from natural language user manuals. In: Proceedings of the 23rd International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'17), Springer, pp 255–272

Robeer M, Lucassen G, van der Werf JME, Dalpiaz F, Brinkkemper S (2016) Automated extraction of conceptual models from user stories via nlp. In: Proceedings of the 24th IEEE International Requirements Engineering Conference (RE'16), IEEE, pp 196–205

Rosadini B, Ferrari A, Gori G, Fantechi A, Gnesi S, Trotta I, Bacherini S (2017) Using NLP to detect requirements defects: An industrial experience in the railway domain. In: Proceedings of the 23rd International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'17), LNCS, vol 10153, pp 344–360

Rosenberg LH, Hammer F, Huffman LL (1998) Requirements, testing and metrics. In: In 15th Annual Pacific Northwest Software Quality Conference

RTCA Inc, EUROCAE (2012) DO-178C: Software Considerations in Airborne Systems and Equipment Certification. Tech. rep.

Runeson P, Host M, Rainer A, Regnell B (2012) Case study research in software engineering: Guidelines and examples. John Wiley & Sons

Shull F, Rus I, Basili V (2000) How perspective-based reading can improve requirements inspections. IEEE Computer 33(7):73–79

Sultanov H, Hayes JH (2013) Application of reinforcement learning to requirements engineering: requirements tracing. In: Proceedings of the 21st IEEE International Requirements Engineering Conference (RE'13), IEEE, pp 52–61

Terzakis J, Gregory S (2016) Ramp: requirements authors mentoring program. In: Proceedings of the 24th IEEE International Requirements Engineering Conference (RE'16), IEEE, pp 323–328

Tjong SF, Berry DM (2013) The design of SREE: A prototype potential ambiguity finder for requirements specifications and lessons learned. In: Proceedings of the 19th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'13), Springer, LNCS, vol 7830, pp 80–95

Wieringa R, Daneva M (2015) Six strategies for generalizing software engineering theories. Science of computer programming 101:136–152

Wilmink M, Bockisch C (2017) On the ability of lightweight checks to detect ambiguity in requirements documentation. In: Proceedings of the 23rd International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'17), Springer International Publishing, LNCS, vol 10153, pp 327–343

Wilson WM, Rosenberg LH, Hyatt LE (1997) Automated analysis of requirement specifications. In: Proceedings of the 19th international conference on Software engineering, ACM, pp 161–171

Yang H, Roeck AND, Gervasi V, Willis A, Nuseibeh B (2011) Analysing anaphoric ambiguity in natural language requirements. Requirements Engineering 16(3):163–189

Yin RK (2013) Case study research: Design and methods. Sage publications

Yue T, Briand LC, Labiche Y (2015) atoucan: an automated framework to derive uml analysis models from use case models. ACM Transactions on

Software Engineering and Methodology (TOSEM) 24(3):13

Zhang H, Yue T, Ali S, Liu C (2016) Towards mutation analysis for use cases. In: Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, ACM, pp 363–373

Zowghi D, Gervasi V, McRae A (2001) Using default reasoning to discover inconsistencies in natural language requirements. In: Proceedings of the 8th Asia-Pacific Software Engineering Conference (APSEC'01), pp 133–140