



# Automated requirement contradiction detection through formal logic and LLMs

Alexander Elenga Gärtner<sup>1</sup> · Dietmar Göhlich<sup>2</sup>

Received: 25 September 2023 / Accepted: 24 May 2024 / Published online: 6 June 2024  
© The Author(s) 2024

## Abstract

This paper introduces ALICE (Automated Logic for Identifying Contradictions in Engineering), a novel automated contradiction detection system tailored for formal requirements expressed in controlled natural language. By integrating formal logic with advanced large language models (LLMs), ALICE represents a significant leap forward in identifying and classifying contradictions within requirements documents. Our methodology, grounded on an expanded taxonomy of contradictions, employs a decision tree model addressing seven critical questions to ascertain the presence and type of contradictions. A pivotal achievement of our research is demonstrated through a comparative study, where ALICE's performance markedly surpasses that of an LLM-only approach by detecting 60% of all contradictions. ALICE achieves a higher accuracy and recall rate, showcasing its efficacy in processing real-world, complex requirement datasets. Furthermore, the successful application of ALICE to real-world datasets validates its practical applicability and scalability. This work not only advances the automated detection of contradictions in formal requirements but also sets a precedent for the application of AI in enhancing reasoning systems within product development. We advocate for ALICE's scalability and adaptability, presenting it as a cornerstone for future endeavors in model customization and dataset labeling, thereby contributing a substantial foundation to requirements engineering.

**Keywords** Contradiction detection · Requirements engineering · Natural language processing

---

✉ Alexander Elenga Gärtner  
a.e.gaertner@outlook.de

Dietmar Göhlich  
dietmar.goehlich@tu-berlin.de

<sup>1</sup> IAV GmbH, Berlin, Germany

<sup>2</sup> TU-Berlin, Berlin, Germany

## 1 Introduction

Requirements are crucial in product development as they serve as a basis for communication among stakeholders, teams, and companies (Loucopoulos 2005; Gericke and Blessing 2012). Specification sheets are commonly used to capture project requirements, aiming for a complete, unambiguous, and contradiction-free system definition (DIN 69901-5:2009-01 2009; Bender and Gericke 2021). However, these sheets often contain thousands of requirements in natural language, derived from interdisciplinary collaboration, typically organized and managed in separate documents for different levels of the product (VDI 2221 Blatt 1:2019-11 2019). The complexity of modern systems and the need for distributed and concurrent development across various levels further complicate the process, which is why the requirements are typically organized and managed in multiple documents (Göhlich and Fay 2021). As a result, errors, including contradictions, are frequently found in these documents. Currently, there is a lack of comprehensive automated tools for detecting contradictions and performing quality analysis on industrial specification documents.

Product development is a complex and dynamic process that requires proper management and understanding of the requirements. Multiple studies have demonstrated the critical importance of high-quality requirements in the success of development projects. In the realm of requirements quality research, the attention is centered on individual characteristics of requirements, such as completeness, complexity, ambiguity, and consistency (Montgomery et al. 2022; IEEE/ISO/IEC 29148-2018 2018). This paper focuses on addressing inconsistencies in the form of contradictions, which can potentially cause misunderstandings and product defects.

Natural Language Inferencing (NLI) is a commonly encountered problem in the field of natural language processing (NLP), in which the objective is to determine the nature of the relationship between a premise and a hypothesis, both of which are represented as sentences (Jang et al. 2020). In recent years, NLP has become more sophisticated, with machine learning models capable of handling complex tasks such as question answering, text extraction, and sentence generation. It was shown that extending NLI to manage sentence relationships could have significant implications for scientific text analysis during product development (Ritter et al. 2008).

We emphasize formal requirements written in natural language that adhere to specific formulation guidelines, particularly using single-sentence requirements that follow the principle of atomicity. For example, employing standard sentence templates for requirements can ensure adherence to these guidelines. Such templates are widely used across different industries and sectors involved in hardware, software, or system development, especially for lower system levels (Dick 2017; GmbH 2016; Wieggers and Beatty 2013; Robertson and Robertson 2013). This approach facilitates consistency and minimizes errors arising from linguistic nuances.

Various solutions have been proposed to tackle this challenge, including knowledge graphs and thesauri. Knowledge graphs are formal representations

of concepts, ideas, or objects about a specific domain and all the relationships between those concepts. Thesauri group together words with similar (synonyms) and opposite (antonyms) meanings (Ahmad et al. 2020). While these solutions can be helpful in identifying contradictions in requirements, they have limitations in their practicality, as we will discuss further in Sect. 3.

Building upon our foundational work (Gärtner et al. 2022) which introduced a taxonomy for detecting contradictions and proposed a semi-automated approach, we made further strides in automation in Gärtner et al. (2023). Our contribution involved developing a method to identify conditions and other sentence components within requirements. This advancement builds upon our previous framework, emphasizing our ongoing commitment to enhancing the automated detection of contradictions in requirements engineering.

Our current research introduces ALICE (Automated Logic for Identifying Contradictions in Engineering). This system synergizes the capabilities of formal logic with large language models (LLMs) to automatically detect and resolve contradictions between two requirements.

Formal logic is a branch of mathematics that deals with the precise rules and structures for reasoning and inference. On the other hand, LLMs are artificial intelligence (AI) models that use machine learning algorithms to learn patterns and relationships between tokens in large natural language datasets. We will elaborate further on this in Sect. 2.2 and explain why LLMs are needed to detect contradictions. Combining these two solutions offers a unique approach to identifying and resolving contradictions automatically, leveraging the capabilities of formal logic and the data-driven approach of LLMs to identify inferences in natural language.

Through this research, we aim to provide an understanding of the strengths and limitations of this approach and its potential applications in product development. We evaluate the effectiveness using real requirements specifications, demonstrating its potential and paving the way for future optimization.

This paper significantly contributes to automated contradiction detection in requirements expressed in controlled natural language (Schwitter 2010). They are particularly relevant in the automotive domain and related fields, as sentence templates are standard for formulating requirements (Dick 2017; GmbH 2016). However, it is essential to note that *formal requirements* have different definitions. The requirements we consider are more natural and intuitive than those discussed in the related work (see Sect. 3.2).

To establish the groundwork for this research, we built upon our previous work (Gärtner et al. 2022), which proposed six essential questions for identifying contradictions. In this paper, we implemented these questions and improved their effectiveness by modifying and expanding them, resulting in a comprehensive set of seven categorized inquiries. These refined inquiries serve a dual purpose: firstly, to determine the presence of contradictions in the requirements, and secondly, to identify the specific types of contradictions that may be present.

We provide a detailed, step-by-step explanation of our implementation and discuss the evaluation using real-world requirements.

## 2 Fundamentals

There are several ways to identify contradictions between sentences without human intervention. In the following section, we clarify the term *contradiction*, discuss two prominent methods—formal logic and machine learning—and explain why we chose a combination of both.

### 2.1 Contradictions

Contradictions are statements that conflict with one another. However, the definition is not intuitively clear, so it must be explicitly defined:

Different definitions of contradictions can be found in the literature on requirements engineering (RE). To get the most generically valid and scientifically accepted definition, we build our theory on the logical philosophy of Aristotle (Aristotle 1999), as shown in Gärtner et al. (2022). The foundation of his logic—also known as term logic, traditional logic, or formal logic—developed in his work *Metaphysics* is the law of non-contradiction (LNC) (Horn 2018). Derived from this, Fig. 1 shows different types of RE-relevant contradictions.

In alignment with the interpretation of Karlova-Bourbonus (2019), it should be acknowledged that contraries, in addition to contradictories, will be considered. Despite falling under the term of contradictions, it is imperative to recognize that they are not synonymous and must be distinctly discerned.

- Contradictory opposites, such as *he is sick* and *he is not sick*, are mutually exclusive; one must be true if the other is false, without overlap.
- Contrary opposites, like *it is black* and *it is white*, cannot be true at the same time but can both be false, indicating they are not exhaustive.
- Subaltern relation follows, where if *everybody is sick* is true, it implies *some people are sick* must also be true, demonstrating a logical step-down.

In practice, identifying contradictions in natural language can be challenging. Based on empirical evidence, it is apparent that only a small number of contradictions observed in real-world situations exhibit explicit characteristics. *‘Rather, contradictions make use of a variety of natural language devices [...]. The most sophisticated kind of contradictions, the so-called implicit contradictions, can be found only when applying world knowledge and after conducting a sequence of logical*

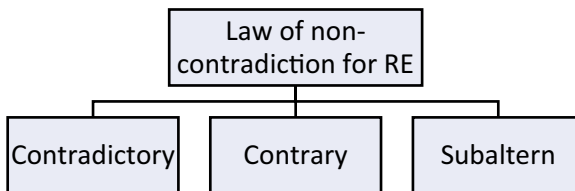


Fig. 1 Contradictions relevant to RE

operations' (Karlova-Bourbonus 2019) such as '*the car must be as fast as possible*' and '*the car must be as fuel efficient as possible.*' Those familiar with physical rules know that the faster a car drives, the more fuel it consumes. In this study, however, we focus solely on explicit (prototypical) (De Marneffe et al. 2008) contradictions, such as '*The car is fast*' and '*The car is slow,*' without conducting deep meaning processing.

## 2.2 Theoretical background for NLP

In the following section, we will explore formal logic as a subset of symbolic AI and Large Language Models (LLMs) as a subset of machine learning (ML).

### 2.2.1 Formal logic

Symbolic artificial intelligence represents intelligence using abstract symbolic representations and logical rules, which are the core elements of formal logic. Unlike earlier approaches aimed at imitating human thought processes (Newell and Simon 1956) the symbolic AI paradigm sought to replicate human knowledge and understanding at a conceptual level.

Formal logic plays a crucial role in symbolic AI, as it encompasses a wide range of applications, including handling simple mathematical tasks, performing string comparisons, and extending to more complex symbolic reasoning and logical operations such as rule-based reasoning, knowledge representation, expert systems, and automated theorem proving. Tasks such as parsing sentences, planning responses, reasoning about meaning, and building expert systems have relied on symbolic problem-solving techniques (Jurafsky and Martin 2019)

It is essential to acknowledge that symbolic AI, which relies on formal logic, has achieved significant milestones. However, it faces limitations when dealing with complex, ambiguous, and uncertain problems (Russell and Norvig 2016). It struggles with tasks that require fluid and intuitive thinking, perceiving subtle distinctions, and generalizing knowledge across domains: it would have difficulties determining whether something could be *fast* and *big* simultaneously but not *fast* and *slow*.

While rule-based systems alone cannot match human intelligence, they provide key capabilities such as logical reasoning, explanation, and abstraction integral to human cognition (Johnson-Laird 2006; Miller 2019).

### 2.2.2 Machine learning

Machine learning (ML) is an approach to AI that uses data-driven techniques to train models that can learn and make predictions. Unlike Symbolic AI, ML algorithms learn patterns and relationships from large amounts of data, enabling the systems to adapt to new information and make accurate predictions in real-world scenarios.

Classical ML models excel in specific tasks like image classification, speech recognition, or natural language processing. However, they may struggle to identify

contradictions across diverse contexts, as their knowledge is confined to the data used during training (Goodfellow et al. 2016; Kim 2018). Therefore, even these models would have difficulty determining the relationships between something being *fast* and *big* or *fast* and *slow* simultaneously.

On the other hand, LLMs like GPT (openai 2020) and LLaMA (Touvron et al. 2023) are pre-trained on extensive datasets, allowing them to learn general language patterns and relationships. This broad knowledge enables LLMs to better understand context and semantics, crucial for identifying contradictions (Surana et al. 2022). They can indeed identify that something could be *fast* and *big* simultaneously but not *fast* and *slow*. This is why, in this work, we use LLMs to detect contradictions.

### 3 Related work

Previous research in requirements management has proposed several methods for tackling inconsistencies between requirements. The following sections discuss taxonomies, NLP-based conflict detection, and ontology-based conflict detection approaches.

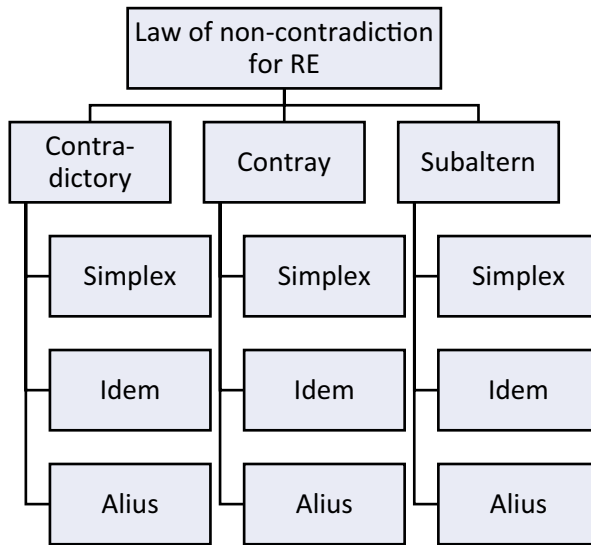
#### 3.1 Classifying conflicts

In our foundational work (Gärtner et al. 2022), we introduced a nuanced taxonomy for detecting contradictions based on Aristotle's Law of Non-Contradiction, recognizing the need for a systematic approach tailored to the intricacies of requirements engineering. This classification emerged from a rigorous analysis of contradictions commonly observed in requirements documents, ensuring a methodical and scientifically grounded approach rather than an arbitrary categorization. By adopting this classification, we aimed to bridge the gap between theoretical logic and practical application in software development, ensuring our methodology is grounded in logical rigor and applicable to the demands of modern engineering projects.

The proposed method categorizes contradictions into distinct types: contradictories, contraries, and subalterns, which fall into three subcategories: Simplex, Idem, and Alius, as shown in Fig. 2. The decision to focus on these subtypes was driven by their prevalence in real-world requirements and unique challenges in automated detection and resolution.

Simplex contradictions (from Latin '*simple*') are characterized by direct opposition without conditional statements. These contradictions are straightforward but crucial for establishing our classification framework. For example, '*The car must be red*' versus '*The car must be blue*' showcases apparent, uncomplicated contradictions.

Idem contradictions (from Latin '*same*') involve identical conditions leading to contradictory outcomes, presenting challenges due to their conditional nature. An example is '*If the customer wishes, the car must be red*' and '*If the customer wishes, the car must be blue*,' where the same condition yields conflicting requirements.



**Fig. 2** Nine types of contradictions for RE (Gärtner et al. 2022)

Alius contradictions (from Latin *‘different’*), distinguished by differing conditions that result in incompatible conclusions, illustrate the complexity of engineering requirements. An instance of this is *‘If the customer wishes, the car must be red’* versus *‘If the car has four doors, the car must be blue,’* demonstrating how different conditions can lead to contradictory outcomes.

These categories—Simplex, Idem, and Alius—were chosen to address the varied and complex nature of contradictions in real-world requirements. By providing a clear structure for identifying and analyzing these contradictions, our approach enhances the capability of automated tools to manage these challenges.

Different types of contradictions occur in varying numbers, have varying levels of project impact, and require different detection approaches. As a result, our previous study defines questions that can be used to identify these types of contradictions. This standardized solution is the basis for the fully automated contradiction detection presented in Sect. 4 of this paper.

Guo et al. (2021) suggest classifying conflicts into three fundamental types: inconsistencies, inclusions, and interlocks, each of which can be further subdivided into seven subcategories. Although this approach shows potential, the authors do not provide a clear method for identifying these categories.

Wu et al. (2022) categorize contradictions into six types: *negation*, *antonym*, *replacement*, *switch*, *scope*, and *latent*. In this classification, *negation*—identified through negative words like ‘no,’ ‘not,’ ‘never,’ and ‘neither–nor’—aligns with what we define as *contradictory*. Both *antonyms* and *replacements* fit within our category *contraries*. The concept of *scope*, which involves either narrowing or expanding the scope expressed in a sentence, corresponds to our *subset* category. However, *switch* is not a contradiction in the context of RE, exemplified by the sentences *‘Sally sold*

a boat to John' and 'John sold a boat to Sally'. This scenario does not necessarily indicate a contradiction, as it is possible for Sally to sell a boat to John while John simultaneously sells his boat to Sally. Lastly, *latent* contradictions, which are not yet developed, fall outside this paper's focus as we concentrate on explicit contradictions.

### 3.2 Natural language processing for detecting contradictions

On a general notice, Zhao et al. (2021) report that most studies (67.08%) in the domain of NLP combined with RE consist of solution proposals evaluated through laboratory experiments or example applications. In contrast, only a tiny percentage (7%) of these studies undergo evaluation in industrial settings, resulting in insufficient practical validation.

De Marneffe et al. (2008) suggest a definition for contradictions in NLP tasks and analyze the task's performance. However, their focus is on implicit rather than explicit contradictions, aligning closely with the *switch*-contradictions delineated by Wu et al. in the preceding section.

Li et al. (2017) state that traditional context-based word embedding learning algorithms are ineffective for mapping contrasting words. To address this issue, they developed a neural network to learn contradiction-specific word embeddings that can separate antonyms. However, their emphasis is not on explicit contradictions, e.g.: 'Some people and vehicles are on a crowded street' and 'Some people and vehicles are on an empty street' (Li et al. 2017). While these sentences contrast with each other, they might still simultaneously be correct. Other approaches with the same goal as Li et al. utilize *WordNet* and *Thesaurus* to obtain additional antonyms and synonyms as semantic constraints (Chen et al. 2015; Liu et al. 2015). These methods might be helpful for fine-grained optimization, as we will mention in Sect. 6.

Ritter et al. (2008) present an approach for automatic contradiction detection. However, their approach does not address explicit contradictions in the context of RE, so their technique cannot be directly applied to our case. Nevertheless, it may prove helpful for future optimization.

Heitmeyer et al. (1996) introduce a technique called consistency checking for detecting errors in requirements specifications using formal logic. Their paper analyzes formal requirements, called *SCR*, and addresses issues like type errors, non-determinism, missing cases, and circular definitions.

However, the requirements are not written using everyday language. Instead, they are expressed in a highly structured manner, where each requirement component has a predetermined category that it must be placed in during the writing process. Our method aims to meet requirements written more intuitively and naturally while still being formal.

Gervasi and Zowghi (2005) explore using formal logic and natural language parsing techniques to identify inconsistencies in requirements. The authors propose a method that automatically discovers and addresses these logical contradictions using theorem-proving and model-checking techniques. By translating the requirement



into a set of logic formulae, they can detect contradictions such as  $\alpha$  and its negation  $\neg \alpha$ . Hunter and Nuseibeh (1998) suggest utilizing formal logic, which records and tracks the information, enabling the tracking of inconsistent information by propagating labels and their associated data. While they do not only focus on negating assumptions like Gervasi et al., Hunter et al. can only detect conflicts if at least a part of the argument is negated, such as  $\alpha$  and  $\neg \alpha$ . Therefore, Gervasi et al. and Hunter et al. cover an essential part of the issue addressed in this paper, namely *contradictories* (see Fig. 2). However, they do not consider *contraries* and *subalterns*, such as ‘the car must be red’ and ‘the car must be blue’ or ‘it must be under 20’ and ‘it must be under 10’.

Other research papers that use NLP focus on categorizing requirements, such as distinguishing between functional and non-functional requirements (Kurtanovic and Maalej 2017), addressing security-related requirements (Jindal et al. 2016) or finding contradictions in prose texts (Karlova-Bourbonus 2019; Sepúlveda-Torres et al. 2021).

## 4 Automation method

The combination of formal logic and LLMs leverages their strengths to identify contradictions between statements. Formal logic offers a formal framework for reasoning about relationships between statements. At the same time, the LLMs ability to access *world knowledge* can be used to ask specific prompts, guiding ALICE to identify contradictions. The prompt engineering process involves designing and fine-tuning prompts to elicit the desired behavior from the model. This combination enables the model to perform logical inference and identify contradictions scalable and flexibly. The resulting system can be used to evaluate the consistency of arguments in natural language text. While we cannot provide a comprehensive account of all the detailed procedures and optimizations used in the code, we will explain the relevant steps in this context in the following sections.

### 4.1 Method fundamentals

In this section, we will explore the core components of our approach, including a decision tree, formal logic, and GPT prompting.

#### 4.1.1 Decision tree

In our previous work, we implemented a classifier system to identify if and what type of contradiction is present. In the present paper, we designed this system as a decision tree, as shown in Fig. 3. Here, seven questions must be answered to determine the specific type of contradiction. The first four questions refer to the effects of the requirements, and the following three questions refer to the condition, if any. One notable advantage of this method is its modular nature, allowing easy adaptability to future advances in the field. Therefore, if more efficacious methodologies

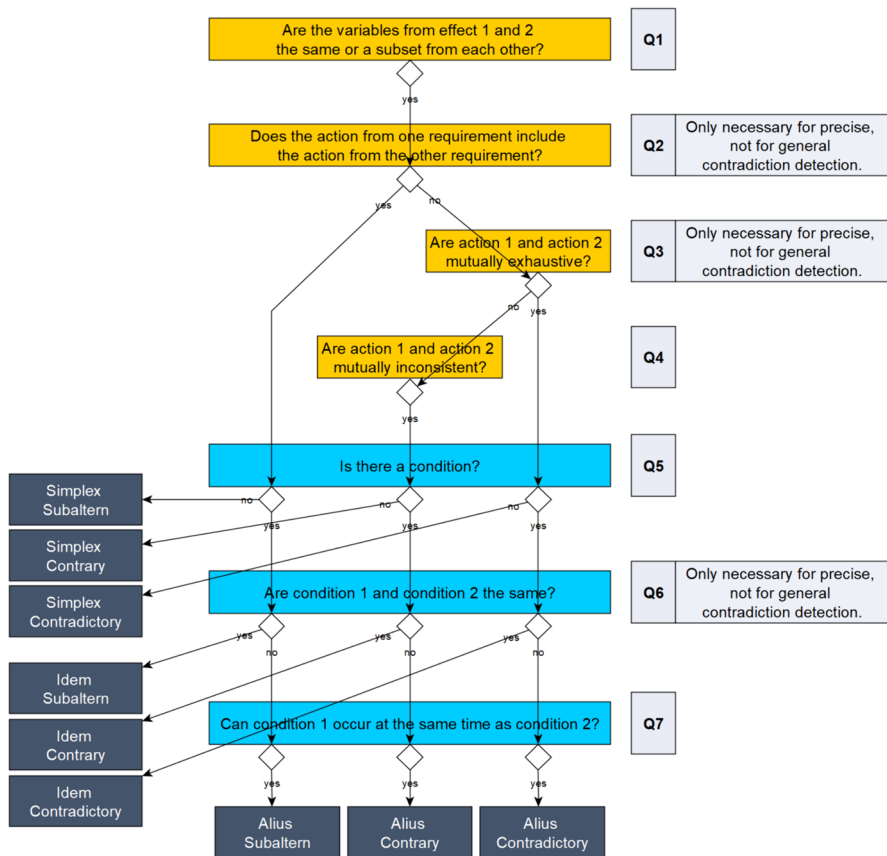


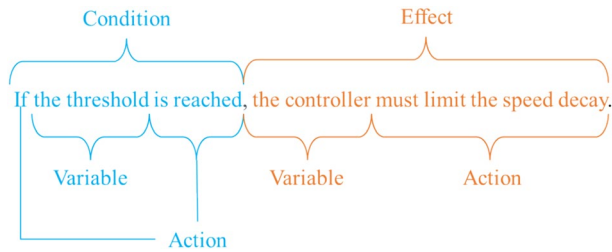
Fig. 3 Modular decision tree for contradiction detection in RE

for resolving the questions are developed subsequently, their integration would be unproblematic. Based on these seven questions, we show how ALICE leads to the desired goal and discuss the advantages of each approach (ALICE or LLMs-only) in Sect. 5.

#### 4.1.2 Formal logic

To detect contradictions, we need the constituents, i.e., *condition* and *effect*, as well as *variable* and *action*, as seen in Fig. 4. This analysis is pivotal, establishing the core structure of our method to address the seven questions proposed by our model effectively.

Take the requirement ‘*If the threshold is reached, the controller must limit the speed decay*’ as an example. The segments ‘*If the threshold is reached*’ and ‘*the controller must limit the speed decay*’ serve as the condition and the resultant effect, respectively. Within the condition, ‘*The threshold*’ represents the



**Fig. 4** Showcase of condition and effect, as well as variable and action in a formal requirement

variable, and '*is reached*' the action. Similarly, within the effect, '*the controller*' is the variable and '*must limit the speed decay*' the action. This is shown in Fig. 4.

In terms of implementation, this step involves utilizing symbolic AI using a grammatical model that incorporates grammatical rules, trigger words, and POS tags to detect conditions, as shown in Gärtner et al. (2023).

It is important to note that conditional statements are not statements of causality. The *presence of oxygen*, for example, is a condition for a *match to light*, whereas *striking the match* is the cause. The lighting of the match is called the *effect*. In terms of implementation, this step involves utilizing symbolic AI using a grammatical model that incorporates grammatical rules, trigger words, and POS tags to detect conditions.

For the contradiction detection presented in this paper, we further extended this logic by incorporating logical and mathematical rules, including string similarity comparisons and mathematical operations.

#### 4.1.3 GPT

To leverage the power of LLMs while addressing the unpredictability of their responses, we adopted a frozen model approach using GPT3. Utilizing this version ensured no updates would be applied to the model during our experiments, providing consistency in its behavior. The later models, such as GPT3.5 and GPT4, are subject to constant updates by Openai (2023a). For the sake of completeness, we have nevertheless tested and briefly evaluated the newer models. In our implementation, we employed the following configuration settings: the engine was set to text-davinci-003, the prompt was provided as the input, a temperature of 0 was used to ensure deterministic responses, the maximum number of tokens was set to 5, top-p sampling with a probability of 1 was employed to avoid randomness, and both the frequency penalty and presence penalty were set to 0 to encourage a neutral influence on the generation process. This configuration allowed us to explore the capabilities of the frozen GPT3 model and examine its outputs in a controlled and predictable manner.

## 4.2 Preprocessing

In NLP tasks, preprocessing is critical in preparing unstructured textual data for analysis. Preprocessing aims to convert raw text into a structured and meaningful format suitable for further analysis. Our research builds on our prior work (Gärtner et al. 2023), making it part of ALICE. It utilizes a fully automated preprocessing pipeline consisting of the following stages: (1) data reduction to replace or eliminate special characters such as ‘<’ with their corresponding expressions; (2) text parsing to generate dependency trees and perform part-of-speech tagging; and (3) identification of verbal expressions, including *conditions*, *effects*, *variables*, and *actions*, collectively referred to as constituents, as explained in Sect. 4.1.2.

## 4.3 Questions

The following section summarizes the seven key questions that form the basis of the ALICE methodology for detecting contradictions within requirement specifications. Each question targets a specific aspect of potential contradictions, enabling a structured and thorough analysis. Detailed examples and GPT-based prompts for each question are provided in the “[Appendix](#)”.

1. Variable identity: Identifies whether the two requirements variables are identical or subsets.
2. Effect inclusivity: Investigates whether one effect encompasses another, which is essential for identifying subaltern contradictions.
3. Mutual action exclusivity: Identifies directly opposing actions, indicating contradictions where the truth of one negates the other.
4. Mutual action inconsistency: Assesses if actions are inherently contradicting without being direct opposites, which is essential for identifying contraries.
5. Condition presence: Identifies conditional clauses that might affect the interpretation of requirements, helping eliminate Simplex-contradictions contradictions.
6. Condition equivalence: Compares conditions in two requirements to detect Idem-contradictions that require the same condition.
7. Condition co-occurrence: Evaluates the possibility of two conditions coinciding, helping eliminate Alius-contradictions contradictions.

## 5 Validation and results

We adopted a structured empirical validation protocol based on established research methodologies to ensure a rigorous and systematic evaluation of ALICE. Our approach is delineated by specific research questions (RQs) designed to assess the effectiveness and applicability of ALICE in identifying and resolving contradictions in engineering requirements.

RQ1: How does ALICE compare to existing LLM approaches regarding accuracy and recall in detecting contradictions within complex specification sheets?

RQ2: How does employing ALICE affect the efficiency and scalability of contradiction detection processes in real-world datasets?

## 5.1 Data analysis and methodology

We used three datasets in our study. Dataset 1 was compiled specifically for this study and served as an initial test suite for developing our method. This dataset, see “Supplementary Information”, includes all possible types of contradicting requirements pairs as well as non-contradicting pairs. Datasets 2 and 3 were obtained from a recent real-world electric bus project: A comprehensive set of requirements was created to outline a modular system intended to replace traditional bus powertrains with modern electric ones. The significance of requirements in developing electric buses can be found in the publication *Design of urban electric bus systems* (Göhlich et al. 2018). Usually, real-world specification sheets cannot be shared with the public due to confidentiality concerns. Nevertheless, we have taken steps to anonymize 210 requirement pairs and have provided access to them, as detailed in Sect. 9.

Dataset 2 contains 1071 requirement pairs, which have been manually checked and labeled for contradictions. Hence, it can be used to validate our method. Dataset 3 contains 3916 pairs, which were not manually checked. On the one hand, it was used to show that the method can handle large datasets, and on the other hand, it served to compare ALICE and GPT3.

Our datasets are inherently unbalanced, reflecting the real-world scenario where the vast majority of requirements do not contradict each other. This imbalance is not a flaw but a feature of using authentic datasets, where contradictions are naturally rare when comparing each requirement against others. Our analysis methods are tailored to recognize and effectively handle such skewness, aiming to accurately identify the relatively few, yet critically important, contradicting combinations. This approach underscores the practical applicability of our method in real-world settings, where the objective is not to balance data artificially but to mirror and navigate the complexities of actual requirement specifications.

Additionally, rather than employing machine learning techniques to train on the unbalanced dataset, our methodology leverages formal logic and pre-trained LLMs, reinforcing the method’s adaptability and efficacy in handling the intricacies of requirement specifications without the need for dataset balancing. Traditional machine learning techniques often necessitate a significant quantity of labeled data to effectively capture the underlying features and patterns within a specific problem domain. Due to the absence of a large dataset specifically tailored for contradicting requirements and inherently unbalanced datasets, incorporating such ML techniques becomes unfeasible.

We compared the results of ALICE to the results when using only LLMs. We conducted an evaluation solely focused on detecting the presence of contradictions without classifying the specific type of contradiction: While ALICE demonstrated the capability to identify specific types of contradictions, LLMs are not equipped

with this classification awareness. Contradictions in specialized industrial domains hinge on subtle semantic details. However, LLMs are trained on broad datasets, which limits their ability to specialize in such nuanced tasks. Thus, we limited the evaluation to general contradiction detection to ensure comparability.

Finally, our evaluation did not consider methods using only formal logic, see Sect. 3.2. As mentioned, traditional context-based word embedding algorithms like *Word2Vec* or *GloVe* are ineffective for mapping contrasting words (Li et al. 2017). For instance, they would both struggle to detect that a car *can drive* and *be red* simultaneously, but it cannot *drive* and *be in the air* simultaneously. The latter example represents an apparent contradiction that would require real-world knowledge to identify.

## 5.2 Results for dataset 1

We present the first dataset's results based on the method presented in Sect. 4. It comprises 87 requirement pairs, of which 61 exhibit contradictions. The selection of the contradictory pairs encompasses all nine types of contradictions and considers as many variations in their formulations as possible.

An extract of the 87 pairs is shown in Table 1, with one example for each type of contradiction and some examples that are not contradictions.

Table 2 displays the confusion matrices as tabular representations of ALICE and GPT3 performance. In these matrices, True Positives and True Negatives indicate correct predictions for contradictions and non-contradictions, respectively. False Positives and False Negatives represent errors where non-contradictions were mistaken for contradictions and vice versa. From these figures, we derive key metrics like accuracy, precision, and recall, which reflect the model's performance in identifying true contradictions among the dataset (Shultz et al. 2010).

The LLM-only prompts were formulated as shown in Fig. 5. The return value is then analyzed regarding whether it contains a string with the content 'yes,' after which the answer is considered as positive.

ALICE achieved an accuracy of 72%, a recall of 75%, and a precision score of 83%. On the other hand, the LLM-only method achieved an accuracy of 47%, a recall of only 32%, and a precision score of 80%. Attempts to guide the model's behavior using n-prompting proved unsuccessful. Several attempts were made, mainly using examples of contradictions and the seven questions described in Sect. 4.3. N-prompting, or "chain of thought" prompting, involves guiding a language model through a series of logical steps to improve its problem-solving accuracy. This technique mimics human reasoning by breaking down complex problems into simpler, articulated steps before concluding.

ALICE is more likely to identify contradictions (i.e.,  $9 + 45 = 44$ ) than the LLM (i.e.,  $5 + 20 = 25$ ). This is even more clear in the following datasets.

**Table 1** Extract from dataset 1

Expected type	Requirement pairs
Simplex subaltern	x must be less than 12
Simplex contrary	The playground must be fun for adults and children
Simplex contradictory	x must be equal to three
Idem subaltern	If it rains, x must be between 10 and 20
Idem contrary	If it rains, the car must stand
Idem contradictory	If it rains, x must be equal to 3
Alius subaltern	If it rains, x must be between 10 and 20
Alius contrary	If the value of the signal LapVeh_FueCur exceeds the value 0 (A), the signal Chrg must be set to TRUE
Alius contradictory	If y is equal to 4, x must be equal to 3
0	The application must be able to run on multiple platforms and operating systems
0	If the signal Pfx is equal to three, y must be equal to 5
0	If the signal Pfx is equal to three, y must be equal to 5
...	...

x must be less than 10  
The playground must only be fun for children  
x must not be equal to three  
If it rains, x must be between 12 and 15  
If it rains, the car must drive  
If it rains, x must be unequal to 3  
If I sing, x must be between 12 and 15  
If the parameter Chrg\_SubVal is set to TRUE, the signal Chrg corresponds to the parameterizable value Chrg\_SubValChrg  
If z is equal to 4, x must be unequal to 3  
The application must integrate easily with other existing software systems  
If the signal Pfx is equal to three, z must be equal to 5  
If the signal Pfx is equal to 4, y must be equal to 6  
...

**Table 2** Results for reference dataset in the form of confusion matrices (Color figure online)

ALICE	0 expected	1 expected	GPT3	0 expected	1 expected
0 calculated	17	15	0 calculated	21	41
1 calculated	9	45	1 calculated	5	20

```
Prompt: Do the following sentences contradict each other, yes or no:
1. {Req1}
2. {Req2}
```

**Fig. 5** LLM prompt-pseudo code

### 5.3 Results for dataset 2

In this section, we analyzed 1071 combinations to validate our method. ALICE achieved an accuracy of 99%, a recall of 60%, and a precision score of 94%. On the other hand, the LLM-only method achieved an accuracy of 97%, a recall of 0%, and a precision score of 0%. Due to unbalanced data, the accuracy alone may not be a highly relevant metric. The recall, which measures the proportion of correctly identified positives, provides a better understanding of the results. Accordingly, ALICE detected 60% of all contradictions, whereas the LLM method did not detect any. For a more comprehensive analysis of the results, the confusion matrices are shown in Table 3.

An example of an accurately identified contradicting pair (true positive) by ALICE is as follows:

**Table 3** Confusion matrices for the first dataset (Color figure online)

ALICE	0 expected	1 expected	GPT3	0 expected	1 expected
0 calculated	1044	9	0 calculated	1041	26
1 calculated	1	17	1 calculated	4	0



1. If the value of the signal  $B\_LdnTrvOfSin3Load\_fKILdngr$  exceeds the value of 0, the value of the segment output signal  $LdnProcTrvtv$  must be set to True.

Formal :  $if\ condition\_1 \Rightarrow signal \stackrel{!}{=} True$

2. If the Parameter  $Load\_DomWrtTgtLdnProcTrvtvToSW$  is set to True, the signal  $LdnProcTrvtv$  corresponds to the configurable value  $Load\_DomWrtLdnProcTrvtv$ .

Formal :  $if\ condition\_2 \Rightarrow signal \stackrel{!}{=} config.$

This is a contradiction of the type *Alius contrary* since it does not exclude that  $condition\_1$  and  $condition\_2$  may occur at the same time, while the effects ( $signal \stackrel{!}{=} True$  and  $signal \stackrel{!}{=} config$ ) contradict each other.

It is a *potential* contradiction because it is theoretically feasible that the conditions have been defined in another place so that parallel occurrence is excluded. Automated detection of such scenarios is not within the scope of this paper but is conceivable for the future. Nonetheless, a manual verification was conducted on a selective basis, where the absence of corresponding exclusions was confirmed for the prior example: The first requirement belongs to the *Function* section, while the subsequent requirement belongs to the *Manual Output* section. It could be assumed that the *Manual Output* supersedes all other requirements due to its later placement; however, this assumption lacks clarity and unambiguity. Sequential processing of requirements must be explicitly defined and is not inherently evident. On the contrary, in cases where sequentially overwriting actions are present, they must be distinctly indicated. Therefore, this is indeed a real contradiction.

An example of a requirement pair falsely identified as *Alius contrary* (false positive) by ALICE is as follows:

1. If the value of the signal  $B\_Mt[...]\text{Mngt}$  is above the value of  $Load\_DebVwTgtInpSig == 1$  s for longer than  $Load\_ThdOfLdnTrvMtl == 0.8$ , the value of the segment output signal  $LdnProcTrvtv$  must be set to True.

Formal :  $if\ condition\_1 > C \Rightarrow signal \stackrel{!}{=} True$

2. If none of the input signals is above the value of 0 and the value of the signal  $B\_Mt[...]\text{Mngt}$  falls below the value of  $Load\_DebVwTgtInpSig == 5$  s for longer than  $Load\_ThdOfLdnTrvMtl == 0.8$ , the value of the segment output signal  $LdnProcTrvtv$  must be set to False.

Formal :  $(if\ condition\_2) \wedge (if\ condition\_3) \Rightarrow signal \stackrel{!}{=} False$

The issue, in this case, traces back to the response provided for question 7. ALICE's approach fails to distinguish between 'above  $x$ ' ( $condition\_1$ ) and 'below  $x$ ' ( $condition\_3$ ) as mutually exclusive, leading it to believe that both conditions can

co-occur. This would result in contradicting effects, thereby rendering the requirements contradictive. It is important to note that—in the current configuration—question 7 involves GPT3 as LLM. It is possible that more advanced LLM models could yield better results in the future.

An example of a falsely identified contradicting pair (false positive) when relying solely on GPT3 (LLM-only) and not using ALICE is as follows:

1. If the CAN status bit B\_NchPsst12333 has the value False, an error must be written to the segment-internal error bus.

Formal :  $\text{if } condition\_1 \Rightarrow x \stackrel{!}{=} error$

2. If the CAN status bit BVwLoad\_sOfNch12333 has the value True, an error must be written on the segment-internal error bus.

Formal :  $\text{if } condition\_2 \Rightarrow x \stackrel{!}{=} error$

When asking GPT why it thinks like that, it explains, '*Since these are different CAN message status bits, both one and the other cannot be fulfilled at the same time. Therefore, the requirements contradict each other.*' ALICE correctly identified this as not contradicting.

## 5.4 Application to large datasets

The third dataset consists of 3916 combinations of requirements, which took approximately 4 h to complete. GPT-3 took 0.9 s on average per requirement pair, resulting in approximately one hour for the whole dataset. ALICE achieved a precision score of 86%, signifying high accuracy in its predictions. In contrast, GPT3's precision score was 0%. Our focus during this analysis was not on optimizing for speed or code efficiency; thus, not all combinations could be thoroughly analyzed within the scope of this research. We did not employ parallel processing techniques, indicating

**Table 4** Results for dataset 3 in the form of confusion matrices (Color figure online)

ALICE	0 expected	1 expected
0 calculated	3848	
1 calculated	6	38

GPT3	0 expected	1 expected
0 calculated	3881	
1 calculated	5	0

**Table 5** Different answers generated by GPT3, GPT3.5, GPT4 (22.03.2023) and LLaMA

GPT3: text davinci		ChatGPT: GPT3.5 Turbo		ChatGPT: GPT4	LLaMA
A1	No, they do not mean the same thing	Yes, 'must be set to true' can mean the same as 'corresponds to the parameterisable value Ldn_C' if that specific value requires the parameter to be set to true	No, 'must be set to true' means that the value must be set as true, while 'corresponds to the parameterisable value Ldn_C' means that it should match the given parameter value	No, Ldn_F is a switch in the SIM that sets the simulator to either use the local directory name (Ldn_F), or not (Ldn_T) [...]	
A2	Yes, these statements could potentially contradict each other	No, the two statements do not potentially contradict each other	Yes, these statements could potentially contradict each other if the parameterisable value Ldn_C is not set to true. If Ldn_C is set to true, then there is no contradiction	No, We cannot have more than one such statement in any given model, but if both are present in a particular model they could easily refer to different parameters	

a potential for future performance enhancements. Nevertheless, the analysis successfully showcased the method's capability to process large datasets and underlined the combined method's advantage over solely using GPT-3, as shown in Table 4.

## 5.5 LLM comparison

We evaluated the datasets using not only GPT3 but also GPT-3.5 turbo and GPT-4 as well as LLaMA by Meta. Our findings indicate that GPT-3.5 turbo had a limited performance, detecting only one contradiction. On the other hand, GPT-4 detected five contradictions, marginally underperforming when compared to GPT-3. LLaMA detected three contradictions. The difference between GPT-3 and GPT-3.5 is particularly evident in question 4, '*Are the actions mutually inconsistent?*' which involves two questions: Question 1: *Does 'must be set to true' mean the same as 'corresponds to the parameterisable value Ldn\_C'?* And Question 2: *could the following statements potentially contradict each other? 'it must be set to true' 'it must correspond to the parameterisable value Ldn\_C.'*

Table 5 depicts the divergent responses obtained from GPT-3, GPT-3.5, GPT-4, and LLaMA, with GPT-3.5 yielding a less satisfactory answer than the other two models. While GPT-4's response is akin to that of GPT-3, its computational time is notably higher, and thus, we opted to employ GPT-3. LLaMA comprehends problems or inquiries accurately but frequently derives erroneous conclusions, generating increasingly imaginative responses.

This outcome is likely not attributable to a lack of quality in the model but instead stems from an alternative objective for which the model was intended. An illustrative example of LLaMA's propensity to stray further from the target with increasing response length is provided in Table 5. The first answer was accurate; however, subsequent elaboration led to irrelevant content. The initial response to the second question was inaccurate, but the ensuing explanation was correct. Therefore, one could even argue that the second answer is an oxymoron, as it contradicts itself. Notably, no definitive comparison between LLaMA and GPT can be drawn from this study. It is plausible that the model's performance may vary depending on the nature of the input stimuli. Future investigations utilizing different input conditions may yield more informative insights into the comparative performance of the two models.

## 5.6 Criticality assessment

ALICE enables the conclusion on the criticality of requirements and, to some extent, on the development costs. Contradictions falling under the types Idem (e.g.,  $A \Rightarrow x \stackrel{!}{=} c$  and  $A \Rightarrow x \stackrel{!}{=} k$ ) and Simplex (e.g.,  $x \stackrel{!}{=} c$  and  $x \stackrel{!}{=} k$ ) are always considered real contradictions because either there is no condition or the conditions are identical. However, contradictions under the Alius type (e.g.,  $A \Rightarrow x \stackrel{!}{=} c$  and  $B \Rightarrow x \stackrel{!}{=} k$ ) are *potential* contradictions.

As noted in Sect. 5.3, it is a *potential* for contradiction as it is theoretically possible for the conditions to have been defined in a manner that precludes their parallel

occurrence. While this paper does not address the automated detection of such scenarios, it remains a possibility for future exploration.

Furthermore, question 5, *'Is there a condition?'* provides additional insight, as noted by Frattini et al. (2022), regarding the correlation between the existence of conditions, lead times, and volatility of requirements. Fischbach's findings suggest that conditions with a strict semantic structure can lead to a more understandable requirement, which can subsequently facilitate the translation of the requirement into downstream artifacts, such as code or test cases.

## 5.7 Conclusion

After conducting a thorough evaluation, we now return to our initial research questions to contextualize our findings within the broader objectives of our study.

RQ1: How does ALICE compare to existing LLM approaches regarding accuracy and recall in detecting contradictions within complex specification sheets?

Our analysis reveals that ALICE demonstrates a notable improvement in both accuracy and recall rates over LLM-only approaches. As evidenced by the results in Dataset 1, ALICE achieved an accuracy of 72%, a recall of 75%, and a precision score of 83%, significantly outperforming the LLM-only method, which registered an accuracy of 47%, a recall of 32%, and a precision score of 80%. For Dataset 2, ALICE achieved an accuracy of 99%, a recall of 60%, and a precision score of 94%. On the other hand, the LLM-only method achieved an accuracy of 97%, a recall of 0%, and a precision score of 0. These findings underscore ALICE's ability to detect contradictions accurately, highlighting the effectiveness of integrating formal logic with LLMs.

RQ2: How does employing ALICE affect the efficiency and scalability of contradiction detection processes in real-world datasets?

The approximately 4-h-long evaluation of Dataset 3, comprising 3916 requirement pairs, demonstrates ALICE's proficiency in managing large datasets efficiently with a precision score of 86%. Despite the inherently unbalanced nature of real-world datasets, ALICE's methodology was adept at identifying the relatively few but critically important contradicting combinations. These results have been critically evaluated and validated as actual contradictions by the second author, who brings over two decades of experience at a major automotive OEM in Germany. His last role involved responsibility for concept design and component integration of various vehicles, underscoring his requirements engineering expertise (Knothe et al. 2006; Göhlich 2008). This showcases ALICE's practical applicability and scalability, marking a significant step in automating contradiction detection in engineering requirements.

## 6 Limits

Several limits to the method must be considered: on the one hand, limits of the LLM, and on the other, limits of formal logic.

Prompt: Are the following sentences contradictory?

1: The car shall stand.

2: When it rains, the car shall drive.

Response: Yes, these sentences are contradictory.

Follow-up prompt: Elaborate.

Response: Sentence 1 implies that the car should always stand, no matter what the weather is like. Sentence 2 implies that when the car shall drive when it rains, which contradicts the first sentence.

**Fig. 6** GPT3 has enough knowledge to detect and explain the present contradiction

Prompt: Are the following sentences contradictory?

1: If the actual heater stage „Cbn[...]“ is greater than 0, the requested pump power „Cbn\_Sp[...]“ must be limited by the parameter „Cbn\_Tri[...]“.

2: If „Batt[...]“ is false, the requested pump power „Cbn\_Sp[...]“ must be limited to 20%

Response: No, these sentences are not contradictory. They are both conditions that could be true at the same time.

**Fig. 7** GPT3 is not able to properly detect the present contradictions. Firstly, the sentences are not conditions but mere sentences, and secondly, although the conditions can indeed be true at the same time, this would lead to contradicting effects

## 6.1 Limits of LLMs for contradiction detection

LLMs lack adequate data to facilitate the identification of specific types of contradictions. Therefore, we consider the limits of general contradiction detection with LLMs.

Detecting fundamental contradictions is a straightforward task, and the corresponding reasoning can be effectively deduced, as exemplified in the subsequent instance, as seen in Fig. 6:

Nevertheless, LLMs do not accurately detect contradictions according to the definition in Sect. 2.1. It is worth noting that the model's response is not necessarily incorrect but rather inappropriate in the RE context. It is plausible that a human, unaware of the context, might respond similarly to GPT3, as shown in Figure (Fig. 7).

Another example of LLM shortcomings was mentioned in Sect. 5.2 as the false positive example concerning the opposite words *below* and *above*, which were not correctly classified. One way to address such shortcomings is to combine machine learning models, such as detecting opposing words from Li et al. (2017).

Furthermore, an inconsequential usage of the words *must* and *shall* can lead to issues with LLM models. In some cases, the model did not recognize these words as synonyms and labeled the effects differently. In general, however, one should refrain from using both words in the same document when writing requirements (GmbH 2016).

Also, LLMs have trouble with strings that are too long. For example, a parameter name such as *OpfsOpfsCnvrGfgsddhFTiOutWRTEForHGfOufiToGhfS* cannot be analyzed as it is a '*long and complex string that doesn't provide any meaningful information to perform a character comparison*' according to GPT. The string is analyzed correctly after inserting an underscore between the signal name, such as *OpfsOpfsCnvrGfgs\_ddhFTiOutWRTEForHGfOufiToGhfS*.

Finally, for example, when specifying requirements, using the phrase

it must be set equal to TRUE

in one instance and in another instance

it must be not be manipulated

it would not result in a contradiction. However, substituting the second instance with

it must be not be set

it would produce a correct contradiction detection. Although this may not be immediately apparent, employing precise and consistent language in technical writing is critical to prevent errors and misinterpretations. Thus, adhering to formal language conventions when dealing with NLP tasks in technical documents is highly recommended.

## 6.2 Limits of formal logic

Filler words can introduce errors in natural language processing tasks. For instance, consider the sentence '*Amy and I both have to fight him,*' where the term *both* is treated as the variable instead of *Amy* and *I*. On the other hand, when comparing it to a second sentence, '*Neither Amy nor I have fought against him,*' *Amy* and *I* would appropriately be identified as the variables. No contradiction would be observed in this case because the variables are considered distinct. Eliminating the filler word *both* enables accurate identification of the contradiction in the sentence. These findings suggest that filler words should be cautiously treated in natural language processing applications to ensure accurate results.

Passive sentence constructions are prone to misinterpretation and should be avoided. For instance, the statement '*There must be no manipulation of Com\_Batt*' is better expressed as '*The function must prevent manipulation of Com\_Batt.*' Using active sentence structures enhances clarity and enables the algorithm to identify the variable performing the action. This aligns with general rules for writing requirements that aim to reduce ambiguity (Sophist GmbH 2016).

Improper placement of commas in complex sentences can result in erroneous condition detection, leading to inaccurate identification of variables and actions. As such, it is crucial to ensure that commas are placed correctly in technical writing. Failure to do so may introduce flawed interpretations and subsequent errors. Consequently, it is advisable to exercise caution when using complex sentence structures and to verify that commas are placed accurately to enhance clarity and precision.

The creation of neologisms should be minimized in technical writing. In cases where they cannot be avoided, adding them to a dictionary is crucial to avoid being detected as *out-of-vocabulary* terms. Therefore, it is advisable to employ established terminology whenever possible.

### 6.3 Threats to validity

In conducting this research, we have identified and addressed several potential threats to the validity of our findings, guided by established frameworks in empirical research (Wohlin et al. 2012).

**External validity:** The study does not provide sufficient grounds to make claims about the generalizability of the results for non-formal requirements, as only formal requirements were analyzed. Focusing on formal requirements was driven by their prevalence and criticality in the domains studied. However, it should be noted that the basis of our investigation is generally applicable grammatical rules. Also, the analyzed data sets follow rules automakers generally apply when writing formal requirements (Ahmad et al. 2020; Dick 2017; Sophist GmbH 2016).

Further investigation and comparison with additional data sets are necessary to address contextual factors such as the domain, development process, requirements engineering techniques, and technologies used.

**Internal validity:** We have minimized researcher and confirmation bias through a standardized methodology based on validated grammatical rules, reducing subjective interpretation. Future enhancements could benefit from practitioner insights and further validation studies.

**Construct validity:** In our study, we have taken careful steps to ensure that the operational definitions used closely align with the theoretical constructs we aim to investigate, a critical component for maintaining construct validity and ensuring our measurements reflect what we intend to measure.

**Conclusion validity:** We have not further detailed the statistical methods employed to affirm the reliability of our conclusions. Robust statistical analysis is essential to verify that the relationships observed in our study are statistically significant and not due to chance, thus underpinning the integrity of our study's conclusions. The potential for overfitting the methodology to the data is a legitimate concern, given the limited availability of a sizable dataset upon which the approach was originally devised.

By acknowledging these limitations and areas for improvement, we underscore our commitment to advancing automated contradiction detection in engineering requirements.



## 7 Conclusion and outlook

This paper contributes to a fully automated contradiction detection for requirements written in controlled natural language that combines formal logic and LLMs called ALICE. This hybrid method involves prompt engineering, which entails designing and fine-tuning prompts to elicit the desired behavior from the model, such as identifying contradictions. The seven-step process is scalable and can be adapted to future advancements in the field, providing a modular way to evaluate the consistency of arguments.

The performance of ALICE and a purely LLM-based method for automated contradiction detection was evaluated in this study. ALICE achieved a higher accuracy (99%), recall (60%), and precision (94%) than the LLM-only method (97%, 0%, and 0%). These findings show that ALICE is more effective than LLM approaches in identifying contradicting formal requirements.

However, the limitations of both LLMs and formal logic in detecting contradictions in technical documents were also highlighted. LLMs may produce false positives due to the use of opposite words and lack adequate data to accurately detect specific types of contradictions. On the other hand, formal logic may be misled by filler words and improper placement of commas, and neologisms should be avoided to ensure accurate condition detection. To ensure precision and reduce ambiguity, it is recommended to use active sentence structures and established terminology in technical writing.

Combining different machine learning models, such as the detection of opposing words by Li et al. (2017), could be a solution to address these shortcomings. Investigating tailored models should be a priority for future research, as our method is predestined to be customized. On the one hand, various approaches to answering the seven questions should be explored. On the other hand, experimenting with different thresholds for string comparisons might prove beneficial, especially when dealing with datasets that adhere to different formulation rules. Also, we validated the method on requirements at lower, more detailed system levels. Although we believe it would also work on higher levels, a thorough validation with actual requirements should be conducted since the reference dataset included such requirements.

Our study focuses on detecting contradictions between pairs of requirements due to their significant impact. We recognize the potential for contradictions among multiple requirements. Addressing these would require advanced analysis methods. Future work could explore extending our methodology to tackle these more complex scenarios, enhancing contradiction detection in requirements engineering for complex system development.

Furthermore, using the presented method, it is now possible to create labeled datasets—which do not yet exist, as explained in Sect. 5. This could enable the potential use of individual ML models for contradiction detection in the future.

Alius contradictions are *potential* contradictions due to theoretically infeasible conditions, as explained in Sects. 5.3 and 5.6. An automated analysis of such conditions was not within the scope of this paper; however, it is conceivable for the future.

A subsequent area of exploration involves the integration of ALICE into the product development lifecycle. In another work (Gärtner and Göhlich 2024), we have detailed a comprehensive overview for embedding ALICE into product development processes, offering developers and product owners a practical tool for identifying and resolving contradictions in the development phase. This integration streamlines the requirements engineering process and significantly reduces the risk of costly reworks and project delays associated with unresolved contradictions. Future work will continue to refine these implementation strategies, further elucidating how industry professionals can seamlessly adopt ALICE to bolster the development of error-free, high-quality products.

In conclusion, advanced AI systems can perform sophisticated reasoning tasks by combining formal logic and LLMs. ALICE provides a promising approach to detecting contradictions in natural language text requirements specifications.

## Appendix

### Question 1

The first question determines whether the variables of effect 1 and 2 are identical or if one set of variables is a subset of the other. This determines if two statements can exhibit an LNC contradiction; see Sect. 2.1. A contradiction can only occur when the variables are the same or when one set of variables is a part of the other, such as comparing:

1. *table*
2. *table leg*

First, we check if both variables 1 and 2 are considered as *out-of-vocabulary* labels, identifiers, naming conventions, or newly coined terms. If they are, we verify if the characters match exactly using formal logic. If there is a 100% match, the answer is '*yes, they are identical.*' If the match is not exact, the variables may still have the same meaning, like '*car*' and '*automobile.*' In this case, we query GPT-3 to assess whether the variables are synonymous, as shown in Fig. 8

If the LLM confirms the variables are synonymous, the answer to the first question is saved as '*yes, they are identical.*'. By explicitly asking to answer with '*Yes or No,*' the response can be framed to include '*Yes*' or '*No*' in the output.

```
Prompt: "are the words \"{}\" similar to words \"{}\"? Yes
or No?".format(variable1, variable2)

Exemplary Response: '\n\nYes'
```

Fig. 8 Prompt for the first question

## Question 2

As explained below, the second question was not processed in this implementation. It examines the potential inclusivity between the actions, determining whether one condition subsumes the other, with one effect serving as the superaltern and the other as the subaltern. For example, in the sentence '*It must be [action]*,' the actions

1. *between 15 and 30 m*
2. *between 20 and 22 m*

demonstrate this relationship, as the range of the second action is wholly encompassed within the range of the first action.

This step is only relevant when it is necessary to identify a precise contradiction type (i.e., subalterns). If identifying the general contradiction type suffices, this question can be omitted. Current LLMs cannot accurately answer this question due to GPT-3's limited mathematical and logical abilities (openai 2023b; Friedman 2020). In our case, all contradictions belonging to the subaltern category are automatically designated as contrary-type contradictions.

## Question 3

The third question identifies mutual exclusivity between two actions, detecting contradictory opposing actions such as

1. *the table must be blue*
2. *the table must not be blue*

When one statement is true, the other is false, and vice versa, indicating mutually exclusive actions. To answer this question, we explored a solution with formal logic and a solution with GPT-3.

Formal logic approach: First, we identify negation terms like '*not*,' '*neither*,' '*not equal*,' '*none*,' '*never*,' '*only*,' '*nowhere*,' or '*without*,' which hint at a potential contradiction. Next, we compare the *actions*, disregarding negation words, e.g.:

1. *must be blue*
2. *must be blue*

```
Prompt: "Are the following sentences similar to each other?:\n-
{}\n- {}".format(action_tokens1, action_tokens2)

Exemplary Response: '\n\nYes, they'
```

**Fig. 9** Prompt for the third question

```
Prompt: "could the following statements potentially contradict each other?\n- \"it must {}\" \n- \"it must {}\" \n\nAnswer 1: ".format(action1, action2)

Exemplary Response: 'Yes, the two statements could potentially contradict each other.'
```

**Fig. 10** Prompt for the fourth question

and evaluate their similarity. As in question 1, this is done by comparing the strings. We achieved the best results with a similarity factor of 90%.

GPT-3 approach: The GPT-3 prompt is shown in Fig. 9. This solution yielded better results than formal logic. Therefore, it was used for the evaluation in Sect. 5. If a similarity is observed, the answer to the third question is saved as ‘yes.’

#### Question 4

The fourth question assesses mutual inconsistency between two actions, identifying contrary (not contradictory, see Sect. 2.1) opposing actions like

3. *the table is red*
4. *the table is blue*

To answer this question, we evaluate whether the actions have the potential to contradict each other using GPT-3. If the model output yields a ‘yes’ as response, the answer to the fourth question is saved as ‘yes’, as shown in Fig. 10

#### Question 5

The fifth question determines the existence of a condition, such as

1. *If I am in France, ...*

```
Prompt: "Does the following sentence contain a condition: The car must stand."

Response: 'Yes, the sentence "The car must stand" contains a condition. This condition is that the car must remain in a standing position.'
```

**Fig. 11** Condition detection with GPT-3

```
Prompt: "can the following states occur at the same time? Yes
or no?:\n1. {} \n2. {}".format(condition1, condition2)

Exemplary Response: '\n\nYes'
```

**Fig. 12** Prompt for the seventh question

To eliminate potential Simplex-contradictions. We apply our previous model (Gärtner et al. 2023) using NLP methods based on grammatical rules, as explained in Sect. 4.1.

Although LLMs might seem suitable for resolving this query, their reliability is questionable, as shown in Fig. 11. Obviously, the sentence does not contain a condition; in fact, it is a simple statement. Consequently, we rely on the grammatical model for greater assurance and 100% replicability.

### Question 6

The sixth question identifies if the first condition is equivalent to the second condition to detect Idem-contradictions that require the same condition, such as

1. *If I am in France, (the table must be blue)*
2. *If I am in France, (the table must be red)*

This question is answered with formal logic using string compares. If similarity exceeds a predefined threshold, the answer is ‘yes.’ We achieved the best results with a threshold of 95%.

### Question 7

The seventh question determines if the first and second conditions can co-occur. The objective of this question is to assess whether two statements have the potential to contradict each other based on the theoretical possibility of both conditions happening at the same time. For example, the statements

1. *If I am in France, (the table must be blue)*
2. *If I am in Germany, (the table must be red)*

cannot co-occur and, therefore, are not contradicting regarding RE. However, the statements

1. *If I am in France, (the table must be blue)*
2. *If it is hot outside, (the table must be red)*

could theoretically contradict each other as they can occur at the same time.

To answer this question, we use GPT-3, as shown in Fig. 12. If the model output yields a 'yes' as response, the answer to the seventh question is saved as 'yes.'

**Supplementary Information** The online version contains supplementary material available at <https://doi.org/10.1007/s10515-024-00452-x>.

**Acknowledgements** We thank IAV GmbH for providing us with the requirements specifications for electric buses.

**Author Contributions** Conceptualization, A.E.G., and D.G.; methodology, A.E.G.; implementation, A.E.G.; validation, A.E.G.; resources, A.E.G.; data curation, A.E.G.; writing—original draft preparation, A.E.G.; writing—review and editing, D.G.; supervision, D.G. All authors have read and agreed to the published version of the manuscript.

**Funding** Open Access funding enabled and organized by Projekt DEAL.

## Declarations

**Competing interests** The authors declare no competing interests.

**Data availability** Dataset 1 is available. Datasets 2 and 3 are partially available, as described in Sect. 5. The data is available in the attachments.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Ahmad, A., Justo, J.L.B., Feng, C., Khan, A.A.: The impact of controlled vocabularies on requirements engineering activities: a systematic mapping study. *Appl. Sci.* **10**(21), 7749 (2020)
- Aristotle: Aristotle's *Metaphysics*. Unter Mitarbeit von Joe Sachs. Green Lion Press, Santa Fe (1999)
- Bender, B., Gericke, K.: Entwickeln der anforderungsbasis: requirements engineering. In: Bender, B., Gericke, K. (eds.) *Pahl/Beitz Konstruktionslehre: Methoden und Anwendung Erfolgreicher Produktentwicklung*, pp. 169–209. Springer, Berlin (2021)
- Chen, Z., Lin, W., Chen, Q., Chen, X., Wei, S., Jiang, H., Zhu, X.: Revisiting word embedding for contrasting meaning. In: Chengqing, Z., Michael, S. (eds.) *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp. 106–115. Association for Computational Linguistics, Beijing (2015)
- De Marneffe, M.C., Rafferty, A.N., Manning, C.D.: Finding Contradictions in Text. Hg. v. Stanford University. USA. Online verfügbar unter <https://nlp.stanford.edu/pubs/contradiction-acl08.pdf>, zuletzt geprüft am 13.04.2022 (2008)
- Dick, J.: *Requirements Engineering*, 4th edn. Springer, Cham (2017)
- DIN 69901-5:2009-01, 2009: *Project Management—Project Management Systems—Part 5: Concepts*

- Frattini, J., Fischbach, J., Mendez, D., Unterkalmsteiner, M., Vogelsang, A., Wnuk, K.: Causality in requirements artifacts: prevalence, detection, and impact. *Requir. Eng.* (2022). <https://doi.org/10.1007/s00766-022-00371-x>
- Friedman, L.: Grant Sanderson and Lex Fridman. Math, Manim, Neural Networks & Teaching with 3Blue1Brown (118), 08.2020. Online verfügbar unter <https://www.youtube.com/watch?v=TMxAbNAVrzI&t=14s> (2020)
- Gärtner, A.E., Göhlich, D.: Towards an automatic contradiction detection in requirements engineering. In: *International Design Conference (Hg.): Design 24* (2024)
- Gärtner, A.E., Fay, T.-A., Göhlich, D.: Fundamental research on detecting contradictions in requirements: taxonomy and semi-automated approach. *Appl. Sci.* **12**(15), 7628 (2022). <https://doi.org/10.3390/app12157628>
- Gärtner, A.E., Göhlich, D., Fay, T.-A.: Automated condition detection in requirements engineering. *Proc. Des. Soc.* **3**, 707–716 (2023)
- Gericke, K., Blessing, L.: An analysis of design process models across disciplines. In: Marjanovic, D., Storga, M., Pavkovic, N., Bojetic, N. (eds.) *DESIGN 2012. Proceedings of the 12th International Design Conference*, May 21–24, 2012, Dubrovnik, Croatia. Zagreb: Fac. of Mechanical Engineering and Naval Architecture (DS, 3), pp. 171–180 (2012)
- Gervasi, V., Zowghi, D.: Reasoning about inconsistencies in natural language requirements. *ACM Trans. Softw. Eng. Methodol.* **14**(3), 277–330 (2005). <https://doi.org/10.1145/1072997.1072999>
- Sophist GmbH: Schablonen für alle Fälle. Online verfügbar unter <http://www.sophist.de/MASTeR-Broschuere/>. zuletzt geprüft am 28.03.2023 (2016)
- Göhlich, D.: Innovationen der fahrzeugtechnik am beispiel der Mercedes-Benz S-Klasse. In: Schindler, V. (ed.) *Forschung für das Auto von morgen*, pp. 129–154. Springer, Berlin (2008)
- Göhlich, D., Fay, T.-A.: Arbeiten mit anforderungen: requirements management. In: Bender, B., Gericke, K. (eds.) *Pahl/Beitz Konstruktionslehre. Methoden und Anwendung erfolgreicher Produktentwicklung*, pp. 211–229. Springer, Berlin (2021)
- Göhlich, D., Fay, T.A., Jefferies, D., Lauth, E., Kunith, A., Zhang, X.: Design of urban electric bus systems. *Des. Sci.* **4**, e15 (2018). <https://doi.org/10.1017/dsj.2018.10>
- Goodfellow, I., Bengio, Y., Courville, A.: *Deep Learning*. The MIT Press, Cambridge (2016)
- Guo, W., Zhang, L., Lian, X.: Automatically detecting the conflicts between software requirements based on finer semantic analysis. Online verfügbar unter <http://arxiv.org/pdf/2103.02255v1> (2021)
- Heitmeyer, C.L., Jeffords, R.D., Labaw, B.G.: Automated consistency checking of requirements specifications. *ACM Trans. Softw. Eng. Methodol.* **5**(3), 231–261 (1996). <https://doi.org/10.1145/234426.234431>
- Horn, L.R.: Contradiction. Hg. v. *The Stanford Encyclopedia of Philosophy*. The Metaphysics Research Lab. Online verfügbar unter <https://plato.stanford.edu/archives/win2018/entries/contradiction/>. zuletzt geprüft am 21.04.2022 (2018)
- Hunter, A., Nuseibeh, B.: Managing inconsistent specifications. *ACM Trans. Softw. Eng. Methodol.* **7**(4), 335–367 (1998). <https://doi.org/10.1145/292182.292187>
- IEEE/ISO/IEC 29148-2018: ISO/IEC/IEEE International Standard—Systems and Software Engineering—Life Cycle Processes—Requirements Engineering. Online verfügbar unter <https://standards.ieee.org/ieee/29148/6937/>. zuletzt geprüft am 15.03.2023 (2018)
- Jindal, R., Malhotra, R., Jain, A.: Automated classification of security requirements. In: *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 2027–2033. IEEE, Jaipur, 21.09.2016–24.09.2016 (2016)
- Johnson-Laird, P.N.: *How We Reason*, 1. Oxford University Press, New York (2006)
- Jurafsky, D., Martin, J.H.: *Speech and Language Processing*. Online verfügbar unter <https://web.stanford.edu/~jurafsky/slp3/>. zuletzt geprüft am 08.04.2023 (2019)
- Karlova-Bourbonus, N.: Automatic detection of contradictions in texts. Gießen: Universitätsbibliothek. Online verfügbar unter <http://geb.uni-giessen.de/geb/volltexte/2019/14447/> (2019)
- Kim, H.K.: 2018 IEEE/ACIS 19th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD). June 27–29, 2018, Busan, Korea: Proceedings. Institute of Electrical and Electronics Engineers; IEEE Computer Society; International Association for Computer and Information Science, IEEE, Piscataway, NJ, Online verfügbar unter <http://ieeexplore.ieee.org/servlet/opac?punumber=8422066> (2018)
- Knothe, F., Mast, J., Böttger, M., Pfeiffer, P., Futschik, H.-P., Hutzenlaub, H.: Die neue CL-Klasse von Mercedes-Benz. *ATZ Automob. Z.* **108**(10), 800–813 (2006). <https://doi.org/10.1007/BF03221821>

- Kurtanovic, Z., Maalej, W.: Automatically classifying functional and non-functional requirements using supervised machine learning. In: 2017 IEEE 25th International Requirements Engineering Conference (RE), pp. 490–495. IEEE, Lisbon, 04.09.2017–08.09.2017 (2017)
- Jang, A., Uzsoy, A.S., Culliton, P.: Contradictory, My Dear Watson. Hg. v. Kaggle. Online verfügbar unter <https://kaggle.com/competitions/contradictory-my-dear-watson>. zuletzt geprüft am 13.03.2023 (2020)
- Li, L., Qin, B., Liu, T.: Contradiction detection with contradiction-specific word embedding. *Algorithms* **10**(2), 59 (2017). <https://doi.org/10.3390/a10020059>
- Liu, Q., Jiang, H., Wei, S., Ling, Z.-H., Hu, Y.: Learning semantic word embeddings based on ordinal knowledge constraints. In: Chengqing, Z., Michael, S. (eds.) Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Association for Computational Linguistics, pp. 1501–1511. Beijing (2015)
- Loucopoulos, P.: Requirements engineering. In: John, C., Claudia, E. (eds.) Design Process Improvement, pp. 116–139. Springer, London (2005)
- Miller, T.: Explanation in artificial intelligence: insights from the social sciences. *Artif. Intell.* **267**, 1–38 (2019). <https://doi.org/10.1016/j.artint.2018.07.007>
- Montgomery, L., Fucci, D., Bouraffa, A., Scholz, L., Maalej, W.: Empirical research on requirements quality: a systematic mapping study. *Requir. Eng.* **27**(2), 183–209 (2022). <https://doi.org/10.1007/s00766-021-00367-z>
- Newell, A., Simon, H.: The logic theory machine—a complex information processing system. *IEEE Trans. Inf. Theory* **2**(3), 61–79 (1956). <https://doi.org/10.1109/TIT.1956.1056797>
- openai: Models. GPT-3. Hg. v. openai. Online verfügbar unter <https://platform.openai.com/docs/models/gpt-3>. zuletzt geprüft am 15.03.2023 (2020)
- openai: ChatGPT—Release Notes. The Latest Update for ChatGPT. Online verfügbar unter <https://help.openai.com/en/articles/6825453-chatgpt-release-notes>. zuletzt aktualisiert am 19.09.2023 (2023a)
- openai: GPT-4 Technical Report, p. 7, Online verfügbar unter <https://cdn.openai.com/papers/gpt-4.pdf> (2023b)
- Ritter, A.; Soderland, S.; Downey, D.; Etzioni, O.: It's a contradiction—no, it's not: a case study using functional relations. In: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing. Unter Mitarbeit von Association for Computational Linguistics, pp. 11–20 (2008)
- Robertson, S., Robertson, J.: Mastering the Requirements Process Getting Requirements Right. Addison-Wesley, Upper Saddle River (2013)
- Russell, S.J., Norvig, P.: Artificial intelligence. A modern approach. Unter Mitarbeit von Ernest Davis und Douglas Edwards, 3rd edn, Global edition. Pearson, Boston, Columbus, Indianapolis, New York, San Francisco, Upper Saddle River, Amsterdam, Cape Town, Dubai, London, Madrid, Milan, Munich, Paris, Montreal, Toronto, Delhi, Mexico City, Sao Paulo, Sydney, Hong Kong, Seoul, Singapore, Taipei, Tokyo (2016)
- Schwitter, R.: Controlled natural languages for knowledge representation. In: Coling 2010: Posters, pp. 1113–1121. Coling 2010 Organizing Committee (2010). <https://aclanthology.org/C10-2128>
- Sepúlveda-Torres, R., Bonet-Jover, A., Saquete, E.: “Here are the rules: ignore all rules”: automatic contradiction detection in Spanish. *Appl. Sci.* **11**(7), 3060 (2021). <https://doi.org/10.3390/app11073060>
- Shultz, T.R., Fahlman, S.E., Craw, S., Andritsos, P., Tsaparas, P., Silva, R.: Confusion matrix. In: Claude, S., Webb, G.I. (eds.) Encyclopedia of Machine Learning, p. 209. Springer US, Boston (2010)
- Surana, S., Dembla, S., Bihani, P.: Identifying contradictions in the legal proceedings using natural language models. *SN Comput. Sci.* **3**, 187 (2022). <https://doi.org/10.1007/s42979-022-01075-3>
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T.: LLaMA: open and efficient foundation language models (2023)
- VDI 2221 Blatt 1:2019-11: design of technical products and systems—model of product design (2019)
- Wiegiers, K., Beatty, J.: Software Requirements, 3rd edn. Microsoft Press, Redmond (2013)
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer, Berlin (2012)
- Wu, X., Niu, X., Rahman, R.: Topological analysis of contradictions in text. In: Amigo, E., Castells, P., Gonzalo, J., Carterette, B., Shane Culpepper, J. Gabriella, K. (eds.) Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. Madrid Spain, 11 07 2022 15 07 2022. pp. 2478–2483. ACM, New York (2022)



Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K.J., Ajagbe, M.A., Chioasca, E.-V., Batista-Navarro, R.T.: Natural language processing for requirements engineering. *ACM Comput. Surv.* **54**(3), 1–41 (2021). <https://doi.org/10.1145/3444689>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.