Requirements Ambiguity Detection and Explanation with LLMs: An Industrial Study

Sarmad Bashir[1,2], Alessio Ferrari[3], Abbas Khan[1,2], Per Erik Strandberg[4], Zulqarnain Haider[5], Mehrdad Saadatmand[1] and Markus Bohlin[2]
[1]RISE Research Institutes of Sweden, Västerås, Sweden, {first.last}@ri.se
[2]Mälardalen University, Västerås, Sweden, {first.last}@mdu.se
[3]University College Dublin, Dublin, Ireland {first.last}@ucd.ie
[4]Westermo, Västerås, Sweden, {first.last}@westermo.com
[5]Alstom, Västerås, Sweden, {first.last}@alstomgroup.com

***Abstract*—Developing large-scale industrial systems requires high-quality requirements to avoid costly rework and project delays. However, linguistic ambiguities in natural language (NL) requirements have been a long-standing challenge, often introducing misinterpretations and inconsistencies that propagate throughout the development lifecycle. Such ambiguous NL requirements necessitate early detection and well-reasoned explanations to clarify and prevent further misunderstandings among stakeholders. While solutions have been developed to detect ambiguities in NL requirements, the advent of generative large language models (LLMs) offers new avenues for explanation-augmented requirements ambiguity detection. This paper empirically investigates LLMs for ambiguity detection and explanation in real-world industrial requirements by adopting an in-context learning paradigm. Our results from three industrial datasets show that LLMs achieve a 20.2% average performance increase in classifying ambiguous requirements when prompted with ten relevant in-context demonstrations (10-shot), compared to no demonstrations (0-shot). Additionally, we conducted human evaluations of the LLM-generated outputs with eight industry experts along four dimensions—naturalness, adequacy, usefulness and relevance—to gain practical insights. The results show an average rating of 3.84 out of 5 across evaluation criteria, indicating that the approach is effective in providing supporting explanations for requirement ambiguities.**

***Index Terms*—requirements classification, requirements ambiguity, large language models, in-context learning**

I. Introduction

In the safety-critical domain, such as the railway industry, contractual specification documents establish the high-level objectives and constraints for complex systems, which are subsequently translated into actionable natural language (NL) requirements guiding the system development life cycle. [1] While NL remains the de facto industry standard for expressing requirements, it is inherently prone to ambiguities [2]. Requirements ambiguities pose a challenge not only for requirements engineers, who must ensure clarity and precision to facilitate accurate implementation, but also for other stakeholders, such as software developers and system integrators, who are involved in implementing and validating the requirements. Specifically, the prevalence of linguistic ambiguities in NL requirements—arising from stakeholders' diverse backgrounds and varying technical expertise—requires early detection during the requirements engineering (RE) process to avoid scope creep and ensure alignment in large-scale projects [3], [4].

In this regard, automated requirements ambiguity detection can significantly help industrial practitioners by reducing feedback loops and minimizing the time and effort required for iterative clarifications. Further, identifying potential ambiguous requirements earlier in the RE process can enable stakeholders to address unclear specifications before they propagate to downstream phases of system design and implementation.

Over the years, several studies have focused on detecting and resolving different types of ambiguities in NL requirements and advancing state-of-the-art solutions. For instance, early-stage approaches rely on traditional natural language processing (NLP) techniques that focus on identifying ambiguities through rule-based methods and linguistic pattern analysis [5], [6]. This is followed by recent works applying deep learning techniques to detect and resolve requirements ambiguities by leveraging contextual embeddings and domain-specific corpora [7], [8]. Despite the remarkable advancements in this domain, one critical issue of most existing solutions, with few exceptions [9], [10], is their lack of performance evaluation on industrial datasets, reducing practical applicability [11]. Moreover, most of the existing solutions focus solely on detecting ambiguities and provide limited human-like explanations of their outputs, which are crucial for industrial practitioners to make well-informed decisions.

To address the aforementioned issues, we propose leveraging the capabilities of generative large language models (LLMs), which are pre-trained on diverse and vast amounts of corpora, for the task of ambiguity detection and explanation. The decision to utilize LLMs is driven by two primary considerations. First, the extensive pre-training and auto-regressive architecture of modern LLMs not only enhances their capability to classify ambiguous requirements but simultaneously enables the generation of explanations that closely resemble human reasoning, offering valuable

insights for practitioners. Second, there is a lack of empirical investigation of LLMs for the task of requirements ambiguity detection, especially with real-world industrial datasets, emphasizing the need to assess their practical effectiveness and reliability in industry-specific scenarios.

In this regard, we closely collaborated with two companies: Alstom Rail AB[1] (Alstom), a global leader in railway manufacturing, and Westermo Network Technologies AB[2] (Westermo), a key provider of industrial communication systems for railways and other critical sectors. Specifically, we employ LLMs with in-context learning [12], where models make predictions using a few task-specific examples. This approach has been applied to software engineering tasks [13], [14], enhancing model performance with task-specific context [15]. In general, in-context learning involves constructing a prompt template that includes NL instructions, a few demonstration examples to establish the context, and a query for the LLM to resolve [16]. The effectiveness of such a learning paradigm largely depends on a customized demonstration selection strategy that closely aligns with the task, enabling the model to better understand the query. Therefore, to select demonstration examples, we empirically evaluate two retrieval strategies, one based on random sampling and the other on semantic similarity to the input query. In addition, we assess the impact of demonstration quantity (0-shot, 1-shot, 5-shot, and 10-shot) on performance, comparing three different LLMs on two project datasets from Alstom (179 and 265 requirements) and one from Westermo (219 requirements) for the task of classifying each requirement as ambiguous or unambiguous, along with the explanation. Our results indicate that the in-context learning approach increases performance by an average of 20.2% across three industrial datasets and models when provided with ten demonstrations (10-shot), compared to zero demonstrations (0-shot). Further, the industry experts evaluated the explanations generated by LLMs, finding them to be linguistically natural, adequate and relevant. However, they also identified limitations, particularly the inadequate understanding and use of domain-specific terminology, highlighting the need to incorporate domain knowledge when applying LLMs to improve performance.

The rest of the paper is organized as follows. Section II outlines the background on requirement ambiguities, large language models, and in-context learning. Section III details our explanation-augmented classification approach. Section IV describes the evaluation methodology, while Section V presents and discusses the results. Section VI provides a qualitative analysis, and Section VII discusses the related work in relation to this paper. Section VIII addresses potential threats to validity. Finally, Section IX concludes with a summary and future research directions.

[1]https://www.alstom.com/se/alstom-sverige

[2]https://www.westermo.com/se

II. Background

In this section, we introduce the task of requirements ambiguity detection and discuss recent developments in LLMs, followed by an explanation of the in-context learning paradigm.

A. Requirements Ambiguity

Ambiguity in NL requirements has been a significant source of frustration among stakeholders and a primary reason for undermining project success in industrial contexts [17], [18]. In general, ambiguity in NL can be classified into multiple linguistic categories—commonly lexical, syntactic, semantic, and pragmatic—each with multiple subtypes and variations [19]. In practice, requirements often show overlapping forms of ambiguity and belong to multiple categories, complicating efforts to place them within a single category [20], [21]. In our context, we frame the problem of requirements ambiguity detection as an explanation-augmented binary classification task, identifying whether a requirement statement is ambiguous or unambiguous and generating concise explanations to support the decision.

The decision to frame the problem as a binary classification task (ambiguous *versus* unambiguous) and not identifying subcategories, aligns with the constraints of our industrial partners' dataset and the limited practical value of subcategory distinctions for our stakeholders. Additionally, in our context, the binary classification of requirements, supplemented with rationales, provides a practical approach and actionable insights for improving the clarity and quality of requirements.

B. Large Language Models (LLMs)

Language models are probabilistic in nature and designed to capture the distributions of words and sequences, enabling them to effectively perform a variety of tasks, such as semantic parsing [22], classification [23], and reasoning [24]. Recent advancements and the development of LLMs—trained on huge text corpora—started with the introduction of transformer architecture by Vaswani et al. [25], which utilized the self-attention mechanisms to capture long-range contextual dependencies. This is followed by Delvin et al. [26], who developed Bidirectional Encoder Representations from Transformers (BERT), employing bidirectional training to enhance data representation learning. Subsequently, Radford et al. [27], [28] introduced the GPT series, focusing on autoregressive next-word prediction, which was further advanced by Brown et al. [16] with GPT-3, showing significant improvements in few-shot generalization across diverse NLP tasks.

These improvements resulted in multiple variations of language models in different sizes and state-of-the-art performance on a wide range of benchmark tasks. In addition, recent efforts in LLMs have shown significant results in human-like reasoning based on the given context [29].

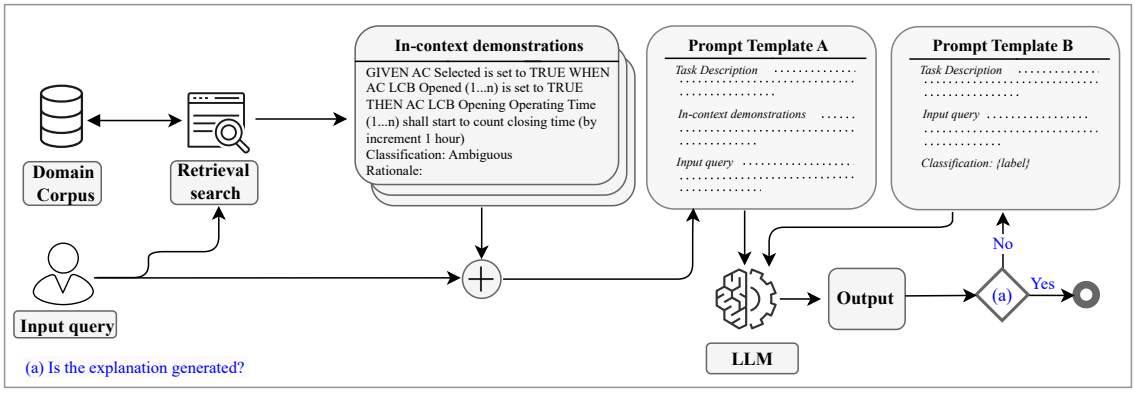

Fig. 1: Overview of our In-context Learning Approach

C. In-Context Learning

The optimal performance of LLMs in specialized tasks, such as detecting ambiguity in domain-specific requirements, typically necessitates adaptation to the particular domain. In this regard, fine-tuning LLMs has traditionally been the default approach for domain-specific tasks [30], although their growing size requires substantial domain-specific datasets and extensive computational resources. Recently, the architecture of modern LLMs has increasingly supported in-context learning, enabling adaptation to specialized domains by providing task instructions and examples directly in the prompt [15], [31]. This significantly reduces computational overhead and the need for additional training data, as the model utilizes provided examples and task instructions at inference time without updating model parameters. Within this concept, the LLMs can be adapted for the target task by dynamically retrieving relevant examples and combining them with prompting strategies [16]. This allows the models to generalize effectively and achieve high performance in domain-specific contexts [12].

In this study, we share the same motivation by employing an in-context learning approach, utilizing prompt engineering techniques with LLMs for ambiguity detection in requirements. This approach provides greater flexibility, enabling effective model adaptation to domain-specific data with limited demonstrations.

III. Approach

Fig. 1 provides an overview of our approach for detecting requirements ambiguity using in-context learning with LLMs. Below, we discuss different phases of the approach.

A. Prompt template Creation

Fig. 2 shows the prompt template constructed for our task of requirement ambiguity detection. Here, the prompt template is defined as PT = (*TD*, r_{examples}, r_{test}), where *TD* is task description that establishes *system* instructions, including a role-play scenario to ensure that the models are aware of the domain context (Fig. 2, ①) and formulating a dual-task that assigns a binary label and requires a concise rationale to

```
<|im_start|>system                                                    Task Description
As a requirements engineer, you are tasked with evaluating new project specifications.
For each requirement, please follow these steps:                                    ①

1. Classification: Determine if the requirement is "Ambiguous" or "Unambiguous".
2. Rationale: Briefly explain your classification decision in 1-3 sentences.
                                                                                    ②
For the "Ambiguous" requirement, identify vague terms or missing details. For the
"Unambiguous" requirement, identify what makes it clear and complete.
                                                                                    ③
For your reference, below are few requirement statements along with their ground truth
labels. Although these examples lack rationale explanations, you must provide your
rationale for classification.
<|im_end|>
                                                                    In-context
<|im_start|>user                                                    Demonstration 1
Failure of the line trip chain shall trigger a maintainer alert via TCMS.
<|im_end|>
<|im_start|>assistant
Classification: Unambiguous
Rationale:
<|im_end|>
                                                                    In-context
. . .                                                               Demonstration N
                                                                    Input Query
<|im_start|>user
Components with a long design life must still be able to be removed without the
need to remove other significant equipment cases or any structural partitions.
<|im_end|>
<|im_start|>assistant
```

Fig. 2: Prompt template (A) for requirements ambiguity detection

```
                                                                    Task Description
<|im_start|>system
As a requirements engineer, you have reviewed project specifications and
classified each requirement as either "Ambiguous" or "Unambiguous."
Given the requirement and its classification, provide a brief explanation in 1-3
sentences.
For an "Ambiguous" requirement, identify vague terms or missing details. For an
"Unambiguous" requirement, explain what makes it clear and complete.
<|im_end|>

<|im_start|>user
{input query}
Classification: {label} <|im_end|>

<|im_start|>assistant
```

Fig. 3: Prompt template (B) to generate missing explanations

justify that label based on the provided definitions (Fig. 2, ②). Further, r_{examples} consists of a set of k in-context demonstrations $(d_1, d_2, \ldots, d_k)$, which are automatically retrieved from a ground-truth. Each demonstration $d_i = (r_i, l_i, \emptyset)$ comprises a requirement statement r_i, its ground-truth label l_i, and an empty placeholder $\emptyset$ for the rationale (i.e., the ambiguity explanation). Since our datasets lack ground-truth rationales, these placeholders ensure the model follows the specified generation structure for a given

query requirement r_{test}. In the prompt template, $< im\_start >$ and $< im\_end >$ are special tokens that define instruction blocks within the prompt, and their format is adapted based on the underlying model architecture.

Note that we structure each d_i as multi-turn conversations in the *user-assistant* format, which aligns with the supervised fine-tuning (SFT) approach used for instruction-tuned language models [32], improving consistency in output generation. However, in some cases, the models fail to generate a rationale for a given r_{test}, likely due to the *recency bias* [33] caused by multiple in-context demonstrations. Specifically, the repeated inclusion of $\emptyset$ with each d_i establishes a pattern that ignores the earlier *system* instructions to generate explanations. As a result, even when explicitly instructed to generate rationales (see Fig. 2, ③), the model disproportionately focuses on recent prompt blocks and omits the rationale. To address this, we automatically detect such missing explanations from the output and issue a follow-up prompt (Fig. 3) aligned closely with the instructions of *TD* in Fig. 2 to generate the missing rationale.

B. Requirement Demonstration Retrieval

For each r_{test}, we retrieve top-k candidate requirements statements r_i from a domain-specific corpus to construct in-context learning demonstrations. Specifically, we consider three configurations to determine the number of k for selecting r_{examples} demonstrations. In the *zero-shot* setting ($k = 0$), the PT includes only the *TD* and r_{test}. In the *one-shot* setting ($k = 1$), the PT includes only one demonstration from r_{examples} retrieved through the selection strategy. Similarly, in the *few-shot* setting ($k > 1$), we retrieve multiple r_{examples} preceding the r_{test} in the prompt.

For the selection of r_{examples} demonstrations, we employ two retrieval strategies: random retrieval and semantic-based retrieval. In the random retrieval, the top-k demonstrations are sampled uniformly from the corpus without considering semantic alignment with r_{test}. In contrast, the semantic-based retrieval strategy utilizes an off-the-shelf sentence transformer (ST) *stsb-roberta-base-v2* [3], which is pre-trained using a contrastive learning objective on multiple semantic textual similarity datasets [34]. The ST model captures both lexical and contextual attributes, enabling it to produce 768-dimensional dense embeddings that accurately represent requirement semantics. Notably, the ST model has shown competitive performance in semantic similarity tasks [13], [22].

For semantic-based retrieval, both r_{test} and all the r_i statements in the domain corpus are encoded into dense vectors. Then, cosine similarity scores are computed between r_{test} and each r_i statement, and the top-k most similar r_i are selected to construct r_{examples} for in-context learning.

[3]https://huggingface.co/sentence-transformers/stsb-roberta-base-v2

Input Requirement (R1): Routing switches ability to filter network traffic between data-plane interfaces and management data traffic.
Ground truth: 'Ambiguous'
Output Label: 'Ambiguous'
Output Rationale: The requirement is ambiguous because it does not specify what type of filtering is required (e.g. packet filtering, protocol filtering, etc.) and what kind of management data traffic is being referred to (e.g. SNMP, HTTP, etc.). Additionally, it does not provide any specific details about the data-plane interfaces, such as the number of interfaces or the type of traffic that will be passing through them.

Input Requirement (R2): GIVEN AC Selected is set to TRUE WHEN AC LCB Opened (1...n) is set to TRUE THEN AC LCB Opening Operating Time (1...n) shall start to count closing time (by increment 1 hour).
Ground truth: 'Ambiguous'
Output Label: 'Ambiguous'
Output Rationale: The requirement is ambiguous because it does not clearly specify what should happen when AC LCB Opened (1...n) is set to FALSE. Additionally, the phrase "by increment 1 hour" is unclear and may be interpreted in different ways.

Listing 1: Example outputs.

C. LLM-based Ambiguity Detection

After constructing the prompt template PT with the retrieved r_{examples}, the final step is to invoke a large language model (LLM) to classify the query requirement r_{test} and generate a rationale in the specified output format. As discussed previously, if the model omits the rationale due to multiple in-context demonstrations, we issue a follow-up prompt to output an explanation. The examples (shown in Listing 1) above report the output generated by our in-context learning approach for two ambiguous input requirements.

In this study, we consider three LLMs: *Phi-3 Mini*[4] (3.8B), *Llama-3*[5] (8B), and *Qwen-2.5*[6] (1.5B), for our task of explanation-augmented requirements ambiguity detection. We use *instruct-tuned* LLM variants instead of base versions due to their superior performance in understanding the instructions in prompts [35]. The models represent different architectural families and are within the 1.5B-8B parameter range. We selected the models because of their (i) availability under open-source licenses that comply with our data privacy and confidentiality requirements, (ii) strong performance on software engineering tasks [13], and (iii) compatibility in terms of size for deployment in

[4]https://huggingface.co/microsoft/Phi-3-mini-128k-instruct
[5]https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct
[6]https://huggingface.co/Qwen/Qwen2.5-1.5B-Instruct

resource-constrained environments. Further, the model selection is guided by Measuring Massive Multitask Language Understanding (MMLU) benchmark scores, which assess the general knowledge and reasoning capabilities of LLMs. At the time of experimentation, *instruct-tuned* variants of *Phi3-mini* (3.8B) and *Llama-3* (8B) achieved comparable MMLU scores of 69% and 68.4%, respectively, ranking as the best-performing models in their respective size categories. Following the release of *Qwen-2.5* (1.5B), we include the model in the evaluation due to its state-of-the-art performance in the sub-2B category [36] and practicality for deployment in our resource-constrained environment.

IV. Evaluation

A. Study Context

This study is conducted in close collaboration with Alstom, a global leader in railway vehicle manufacturing and Westermo, a prominent provider of industrial network solutions. Both organizations require strict requirements engineering practices to ensure system safety and performance. In such specialized domains, project initiation often requires reviewing extensive textual documentation that includes system requirements and standards, regulatory mandates, and safety guidelines. The complexity of these artifacts and processes requires detailed analysis to ensure compliance with safety and performance criteria. Traditionally, requirements analysts rely on manual methods to detect ambiguous or conflicting requirements, a laborious and time-consuming process. In addition, clarifying requirements requires involving multiple stakeholders, which extends the feedback cycle and ultimately delays the time-to-market. Consequently, both organizations continuously seek efficient early-stage requirements ambiguity detection methods to facilitate their RE process.

B. Research Questions

The study aims to assist practitioners by detecting ambiguous requirements and generating concise explanations to support decision-making. Since different LLMs may perform differently when given the same in-context learning demonstrations within a standardized prompt template, we systematically evaluate their effectiveness in detecting requirement ambiguities. In this regard, we evaluate the impact of the number of demonstration examples (i.e., 0-shot, 1-shot, 5-shot and 10-shot) on the performance of LLMs. We also investigate the effect of customized demonstration selection, assessing whether highly relevant in-context examples improve performance compared to randomly selected ones. Finally, we examine the usefulness of LLM-generated explanations for practitioners. To this end, we formulate the following research questions (RQs).

- **RQ1.** *Which LLM yields the best results in requirements ambiguity detection task?*
- **RQ2.** *What is the impact of demonstration quantity on LLM performance?*

TABLE I: Datasets

Dataset	Reqs.	AW	Amb.	Valid.	Test
Westermo	219	26	56	44	175
$\text{Alstom}_{\text{ProjectA}}$	179	33	74	36	143
$\text{Alstom}_{\text{ProjectB}}$	265	37	48	55	219

*AW = average words per requirement; Amb. = number of ambiguous requirements (remaining are unambiguous); Valid. = number of validation requirements; Test = number of test requirements.

- **RQ3.** *Which demonstration selection technique improves the effectiveness of LLMs?*
- **RQ4.** *How do practitioners perceive and evaluate the explanations of LLMs?*

C. Data collection & Preparation

Table I outlines the attributes of the datasets under consideration. In particular, we utilize three industrial datasets: one from Westermo and two from Alstom, referred to as $\text{Alstom}_{\text{ProjectA}}$ and $\text{Alstom}_{\text{ProjectB}}$ dataset. Each dataset consists of requirements previously annotated with ground-truth binary labels representing ambiguous or unambiguous. To perform the evaluation, we prepared the datasets using the following steps. First, we processed the requirements to identify and remove duplicates. For Westermo, duplicate requirements were removed within the dataset, while for the Alstom datasets, we detected and removed duplicates both within each dataset and across projects. Then, we created three stratified random splits of each dataset into validation (20%) and test (80%) subsets to address model performance variability.

Note that we did not create the training subsets, as our evaluation followed an in-context learning paradigm that does not require LLM training and update of its parameters. Instead, we treated the validation set as a pool to select demonstration examples for in-context learning.

D. Experimental Setup

To evaluate the performance of LLMs, we employ 0-shot and 1-shot settings, as well as demonstration selection with 5- and 10-shots for in-context learning. The reason for limiting the maximum number of demonstration shots to ten is driven by two factors. First, the limited context window of LLMs restricts the tokens that can be processed in a single input. Second, larger prompts increase memory usage and processing time, leading to practical computational challenges [37]. Furthermore, we selected the demonstrations both randomly and based on their semantic similarity to the given query to evaluate their impact on model performance and generalization. In both approaches, we set the temperature parameter to zero to reduce the randomness in model responses and consistently select the most probable tokens at each decoding step. The maximum length of the generated sequence is set to 156 tokens to limit output size.

In addition, we repeated each experiment three times, each time using a different stratified random split of the data, and

LLM	Method	Westermo			$\text{Alstom}_{\text{ProjectA}}$			$\text{Alstom}_{\text{ProjectB}}$		
		P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
	Random	59.1	48.7	52.2	51.9	53.4	53.5	69.3	50.8	56.5
Qwen-2.5 (1.5B)	0-shot	58.9	42.1	44.8	61.3	54.3	52.8	68.8	38.5	43.4
	1-shot ($Selection_{random}$)	63.6	45.0	47.1	55.1	54.3	54.6	71.6	39.2	43.4
	1-shot ($Selection_{semantic}$)	63.2	49.3	52.2	50.5	53.1	50.8	69.9	46.7	52.5
	5-shot ($Selection_{random}$)	60.8	52.8	55.3	57.1	56.4	56.6	74.5	59.7	64.4
	5-shot ($Selection_{semantic}$)	62.5	49.9	52.8	58.0	58.7	58.2	74.3	59.7	64.3
	10-shot ($Selection_{random}$)	63.6	65.7	64.3	60.1	60.4	60.0	74.6	71.1	72.5
	10-shot ($Selection_{semantic}$)	59.6	57.1	57.8	62.7	63.2	62.7	73.1	73.0	72.9
Phi3-mini (3.8B)	0-shot	68.4	61.7	63.8	60.6	61.8	59.7	66.5	45.9	52.1
	1-shot ($Selection_{random}$)	66.5	49.7	52.3	61.3	59.7	60.0	78.3	55.5	60.5
	1-shot ($Selection_{semantic}$)	60.5	45.0	47.9	65.5	63.2	63.4	72.9	47.2	52.6
	5-shot ($Selection_{random}$)	63.5	56.2	58.6	62.3	61.8	61.9	74.1	68.2	70.6
	5-shot ($Selection_{semantic}$)	63.2	53.5	56.3	68.9	**67.1**	**67.4**	76.4	68.9	71.6
	10-shot ($Selection_{random}$)	66.2	63.2	64.3	64.3	64.6	64.3	70.5	70.5	72.0
	10-shot ($Selection_{semantic}$)	65.3	64.2	64.4	**67.1**	**67.1**	67.1	74.1	76.6	75.2
Llama-3 (8B)	0-shot	64.3	59.4	61.3	59.5	60.8	56.0	67.6	59.7	63.1
	1-shot ($Selection_{random}$)	68.1	66.1	67.0	53.8	58.0	49.6	74.7	**77.4**	**75.8**
	1-shot ($Selection_{semantic}$)	66.7	65.3	65.9	60.8	61.8	56.3	69.5	71.2	70.3
	5-shot ($Selection_{random}$)	65.6	58.7	60.6	57.6	59.4	56.3	78.2	66.8	70.4
	5-shot ($Selection_{semantic}$)	70.5	57.7	60.3	61.6	62.9	60.3	70.7	57.1	61.8
	10-shot ($Selection_{random}$)	70.4	**67.8**	**68.6**	64.4	64.6	61.3	**79.0**	69.5	72.6
	10-shot ($Selection_{semantic}$)	**71.9**	63.2	65.3	64.6	64.6	61.8	72.7	62.9	66.5

TABLE II: The performance comparison across different LLMs and demonstration selection strategies (in %).

reported the average scores across runs. This reduces the variance in performance estimates and provides a more reliable assessment of models by averaging results across multiple splits. As a point of reference, we also include a *random* classifier as a baseline, representing chance-level performance. Due to the use of an in-context learning paradigm, our validation set is small, so we sample labels uniformly rather than by frequency, reflecting true random chance performance.

E. Evaluation Metrics

To address our research questions, we employ a two-fold evaluation. First, we adopted standard metrics, Precision (P), Recall (R), and F1 score, to evaluate the performance (RQ1–RQ3) of different LLMs for the task of requirements ambiguity detection. As Table I shows, our datasets are highly imbalanced, reflecting real-world scenarios where ambiguous and unambiguous requirements may not be equally distributed, we report weighted average scores to account for this imbalance. In addition, we use the non-parametric Friedman test, which determines the statistically significant differences among multiple classifiers evaluated across multiple datasets [38]. Specifically, we use the Friedman test followed by the Nemenyi post-hoc test to assess the performance of different LLMs (RQ1) and the effect of demonstration quantity (RQ2). For RQ3, we use the Wilcoxon signed-rank test to perform a pair-wise comparison and determine whether the differences in performance between demonstration selection techniques (random *vs* semantic) are statistically significant.

Second, we conducted a human evaluation to gain expert insights (RQ4) into the practical utility and quality of the results. In this regard, a total of eight domain experts—three from Westermo and five from Alstom—performed the human evaluation. From the Westermo side, two experts were from the technical pre-sales team with expertise in the company's technical offerings, customer needs and extensive experience with internal and external requirements. The third expert was a certified requirements engineer working as a project manager in the R&D with a background in software test automation. From the Alstom side, four evaluators were from the Train Control and Information System (TC&IS) department and one from the Propulsion department. All were seasoned engineers who worked in various roles, responsible for handling requirements from the tender stage to the product delivery. The experts evaluated the LLMs' explanations using the four criteria listed below.

1) **Naturalness** evaluates the fluency, readability, and linguistic quality of the generated explanations. The motivation behind this criterion is to evaluate how well the practitioners can understand the reasoning behind classification decisions without misinterpretation.
2) **Adequacy** reflects the extent to which the rationale provides comprehensive and relevant information. The criterion measures whether the explanation includes information richness to justify the classification decision.
3) **Usefulness** examines the extent to which the generated classification labels and their explanations assist practitioners in making well-informed decisions.
4) **Relevance** evaluates the degree to which the assigned

classification labels and rationales are relevant and aligned with the query requirements.

The human evaluation is conducted using a five-point Likert scale (1 for poor, 2 for marginal, 3 for acceptable, 4 for good, and 5 for excellent). Such an experimental setting, along with similar evaluation criteria, has been adopted in previous studies [13], [14]. However, based on discussions and agreement with our industrial partners, we revised the definitions to align with our specific use case.

F. Implementation

Our experimental setup is implemented primarily in Python. The *sentence-transformers*[7] library is utilized for semantic-based retrieval of demonstrations. We perform inference with open-source LLMs using Hugging Face protected and dedicated inference endpoints for secure and scalable model deployment and execution.

V. Results and Discussion

Table II presents the overall performance of selected LLMs on industrial datasets for the task of requirements ambiguity detection. The *Method* column specifies the demonstration configuration, indicating both the number of in-context examples (e.g., 0-shot, 1-shot, 5-shot, and 10-shot) and the retrieval selection strategy (random or semantic). In the following, we discuss the results based on the evaluation metrics defined in IV-E.

RQ1: LLM Performance. Table II shows the results of various LLMs across multiple datasets, highlighting how both model and dataset choice significantly affect results. Notably, no single model consistently outperforms others across all considered methods. In general, larger models such as Phi3-mini (3.8B) and Llama-3 (8B) perform better than the smaller Qwen-2.5 (1.5B), suggesting that model size may affect the performance. Similarly, in the Alstom datasets, Phi3-mini outperforms other models with the highest F1-score on $\text{Alstom}_{\text{ProjectA}}$ (67.1%) and $\text{Alstom}_{\text{ProjectB}}$ (75.2%), both with 10-shot semantic selection. However, Qwen-2.5, despite its smaller size, achieved superior results under specific configurations. Notably, on the $\text{Alstom}_{\text{ProjectA}}$ dataset with a 10-shot semantic selection, Qwen-2.5 resulted in an F1-score of 72.9%, outperforming Llama-3 (8B), which scores 66.5%.

To further assess the effect of model choice on performance, we applied the Friedman test to the weighted F1 scores obtained across three datasets given seven prompt configurations, resulting in 21 blocks, each containing the three F1 scores of the models for a specific *dataset–method* pairing. The test revealed statistically significant differences in performance between model sizes, $p < 0.05$. Further, post-hoc Nemenyi comparisons showed that Qwen-2.5 performed significantly worse than both Phi3-mini and Llama-3 This could be influenced by the more limited size of Qwen-2.5. However, the performance of Phi3-mini and Llama-3 did not differ significantly despite their difference in size, suggesting that model size alone does not determine performance, and prompt design and dataset characteristics could also play critical roles.

> **Answer to RQ1.** No single model consistently outperforms the rest across all considered datasets and configurations. Although larger models (Phi3-mini and Llama-3) tend to show better performance, factors like prompt configuration and domain-specific dataset characteristics may influence the overall performance.

RQ2: Demonstration Quantity. Table II shows that in general, the increase in the number of demonstration examples improves the in-context learning performance across the models and datasets. For instance, on the Westermo dataset, Llama-3 achieves an F1 score of 61.3% without any requirement demonstrations (0-shot), which improves to 68.6% once ten randomly selected demonstrations are used. A similar trend is observed in the $\text{Alstom}_{\text{ProjectA}}$ and $\text{Alstom}_{\text{ProjectB}}$ datasets, where Llama-3 achieves F1 scores of 56.0% and 63.1% without demonstration examples and with ten randomly selected examples, the scores improved to 61.3% and 72.6%, respectively. These findings are consistent with existing work [14], which suggests that inferring LLMs with more in-context learning demonstrations often improves performance.

To evaluate the statistical significance of the demonstration quantity effect on performance, we conducted a Friedman test across nine *model-dataset* combinations (blocks), each consisting of four levels (0-shot, 1-shot, 5-shot, and 10-shot). For each block, F1 scores were averaged across demonstration selection variants (i.e., random and semantic) within the same shot level to focus the analysis specifically on the impact of demonstration quantity. The Friedman test showed a statistically significant impact ($p < 0.05$) of demonstration quantity on performance, indicating that the number of in-context demonstrations greatly influences F1 scores across all considered models and datasets. The follow-up Nemenyi post-hoc test showed that only the pair-wise comparisons between 10-shot *vs*. 0-shot and 10-shot *vs.* 1-shot achieved statistical significance. This suggests that the differences between lower shot counts (0, 1, 5) are not significantly different across all settings, while most of the performance gain comes from increasing the demonstration count to ten[8].

[7]https://huggingface.co/sentence-transformers

[8]Although the statistical difference between between five and ten examples is not significant, the results in Table II suggest that the use of ten examples lead to higher performance for most of the cases.

Answer to RQ2. While the performance of LLMs improves with additional in-context demonstrations, only the 10-shot setting produced statistically significant gains. This suggests that, in our case, the models require around ten examples to achieve consistent improvements.

RQ3: Demonstration Selection. The comparative analysis in Table II indicates that the effectiveness of demonstration selection techniques—*random* and *semantic*—varies with the number of demonstrations provided to the model. In the 1-shot setting, the *semantic* selection shows a slight advantage over *random* selection in specific cases. For instance, the Qwen-2.5 model for the Westermo dataset improved the F1 score from 47.1% (random) to 52.2% (semantic). Similarly, for Phi3-mini on $\text{Alstom}_{\text{ProjectA}}$, the *semantic* demonstration selection achieves a considerable improvement from 60.0% (random) to 63.4%. However, this improvement is not consistent across all datasets or models. Notably, the Llama-3 model shows the opposite pattern on $\text{Alstom}_{\text{ProjectB}}$, where *random* selection achieves a significantly higher F1 score of 75.8% compared to 70.3% score from *semantic* selection. As the number of demonstrations increases to five, the performance differences between selection techniques continue to vary across models and datasets. For Qwen-2.5 on $\text{Alstom}_{\text{ProjectA}}$, the *semantic* selection technique provides a slight improvement of F1 score 58.2% over *random* 56.6%. Phi3-mini shows a clear advantage with *semantic* selection, achieving higher F1 scores on both $\text{Alstom}_{\text{ProjectA}}$, increasing from 61.9% to 67.4%, and $\text{Alstom}_{\text{ProjectB}}$, rising from 70.6% to 71.6%. However, for Llama-3 on the $\text{Alstom}_{\text{ProjectB}}$ dataset, the F1 score drops significantly from 70.4% with *random* selection to 61.8% with *semantic* selection. With 10 demonstrations, the Phi3-mini model achieves higher F1 scores with *semantic* selection, reaching 67.1% on $\text{Alstom}_{\text{ProjectA}}$ compared to 64.3% with *random* selection, and 75.2% on $\text{Alstom}_{\text{ProjectB}}$ compared to 72.0%. In contrast, Qwen-2.5 performs better with *random* selection on the Westermo dataset, achieving 64.3%, while *semantic* selection yields 57.8%. Similarly, Llama-3 consistently favors *random* selection, particularly on $\text{Alstom}_{\text{ProjectB}}$, where it achieves 72.6% F1 score compared to *semantic* selection score of 66.5%. These findings suggest that while *semantic*-based technique can enhance performance in multiple cases, its impact remains varying across all models and datasets.

To further analyze the statistical significance, we performed the Wilcoxon signed-rank test to determine whether the choice of demonstration selection strategy affects the F1 performance across all considered configurations. The test, based on 27 paired comparisons, revealed no statistically significant difference ($p > 0.05$), indicating that the *semantic*-based demonstration selection neither consistently outperforms nor underperforms *random* selection.

Criterion	Westermo		$\text{Alstom}_{\text{ProjectA}}$		$\text{Alstom}_{\text{ProjectB}}$	
	μ	σ	μ	σ	μ	σ
Naturalness	4.34	0.80	4.28	1.06	3.65	1.50
Adequacy	4.34	1.13	3.91	1.19	3.82	1.32
Usefulness	3.26	1.33	3.52	1.38	3.28	1.45
Relevance	4.08	1.19	3.97	1.15	3.60	1.32

TABLE III: Human evaluation results

* μ=average; σ=standard deviation

Answer to RQ3. No single demonstration selection strategy consistently outperformed the other across all configurations. The effectiveness of demonstration selection is influenced by how each model represents and processes semantic information, as well as by the characteristics of the dataset, such as domain specificity.

RQ4: Human Evaluation. We conducted a human evaluation using a five-point Likert scale to assess the quality of LLM-generated explanations and gain practical insights. A 20% of the generated explanations—including both true positives and false positives[9]—from each test dataset were selected and rated by domain experts based on predefined evaluation criteria, and also provided qualitative feedback on these instances. We randomly sample the instances for evaluation because (i) no single model consistently outperformed others across datasets or metrics and (ii) in the absence of ground truth rationales for requirement labels, cross-sampling across models is valuable for capturing diverse reasoning strategies that can guide future improvements in prompt tuning. Table III presents the descriptive statistics from the study for each dataset. For discussion, we report the *weighted* averages μ and standard deviations (σ) aggregated across all datasets. The *weighted* scores are calculated due to the varying number of evaluators (total 8): Westermo (3 evaluators), $\text{Alstom}_{\text{ProjectA}}$ (5 evaluators), $\text{Alstom}_{\text{ProjectB}}$ (4, shared with $\text{Alstom}_{\text{ProjectA}}$). The results are as follows: Naturalness $\mu(\sigma)$ = 4.08 (1.21), Adequacy $\mu(\sigma)$ = 3.99 (1.24), Usefulness $\mu(\sigma)$ = 3.38 (1.40), and Relevance $\mu(\sigma)$ = 3.87 (1.24).

Naturalness. Practitioners rated the naturalness of LLM responses positively with a μ of 4.08, indicating that outputs were generally fluent and easy to comprehend. However, the σ of 1.21 indicates moderate variability, suggesting a few inconsistencies regarding the clarity of specific outputs.

Adequacy. μ = 3.99 shows that practitioners generally agree that the provided rationales are comprehensive and information-rich, effectively supporting the classification decisions. The σ = 1.24 suggests variability among experts, with specific explanations missing relevant details. This is particularly evident in Alstom projects (*weighted* σ = 1.25),

[9]Previous studies observed that even in presence of false positives, experts can have insight on ambiguities that they did not previously considered [39].

where experts noted insufficient justification for the given query requirement and its generated classification label.

Usefulness. $\mu = 3.38$ shows experts found the explanations satisfactory for informed decision-making. However, the high $\sigma = 1.40$ reflects considerable variation in expert responses, indicating that while some found the outputs practically helpful, others noticed limited value. In this regard, there is still a need to improve the usefulness of the generated responses.

Relevance. $\mu = 3.87$ suggests that experts found the generated responses reasonably aligned and relevant to their queries. However, the notable variability $\sigma = 1.24$ suggests differences in experts' perceptions of relevance, with some experts noting only partial alignment or insufficient context-specific relevance in certain instances.

Answer to RQ4. Domain experts perceive LLM-generated explanations as linguistically natural, adequate, and relevant. However, the low usefulness ratings suggest shortcomings in the explanations and highlight the need for better contextual alignment with domain specificity.

VI. Expert Feedback

Domain experts reviewed the results of our case study on requirements ambiguity detection, providing detailed analysis and feedback to guide future improvements. In this section, we present a brief qualitative analysis of their responses to discuss the implications of our study.

Insights into the Variability of Standards Interpretation. The experts pointed out the variability in the interpretation of standards, noting that only referencing them in requirements is often insufficient because standards themselves can be broad, complex, and open to interpretation. The LLMs frequently accepted references to standards without verifying the relevance or completeness of the cited information. An expert emphasized this issue, stating, *"Standards-compliance is a very tricky topic... [The LLM] just saying that the standard is 'well-defined' is not good enough,"* in response to the LLM's justification for classifying a requirement as unambiguous. Similarly, in another instance, the LLM failed to identify ambiguity when a requirement referenced a standard but did not specify the details for compliance testing. The expert noted, *"What test levels are required? The Requirement specifies a standard but not what test levels to comply to"*, showing that referencing standards alone can still result in unclear requirements.

Furthermore, the reviewers concerns suggest that standards often require careful, *clause-by-clause* reading to ensure that requirements fully capture necessary constraints, a level of analysis that LLMs consistently failed to achieve. In this regard, retrieval-augmented generation (RAG) [13] techniques can be employed to access the standard document relevant to a specific requirement, enabling LLMs to retrieve necessary details for more informed analysis.

On the importance of Domain-Specific and Contextual Knowledge. The experts emphasized the need for domain-specific and contextual knowledge to interpret and evaluate requirements accurately. In this regard, experts mentioned multiple instances where incompleteness or inaccuracy of generated rationales was due to the model's limited understanding of industry terminology, relevant standards, and operational context. For instance, an expert from Westermo noted that a rationale related to railway standard *"IEC 61375-2-5 ETB physical train naming convention"* overlooked the specifications *"...defined in the 61375-standard that the AI does not seem to know,"* showing the model's lack of familiarity with essential domain-specific concepts. Similarly, experts from Alstom also pointed out that specific terminologies in requirements, such as *grease free chains, single failure,* and *rising edge,* which are ambiguous to model, have well-established meanings within the domain, often defined by industry standards.

These insights emphasize the need to incorporate domain-specific information into the LLM's knowledge base, which further improves its reasoning capabilities and, in turn, the practical utility of the models within specialized domains. This problem has been also observed by other authors who used LLMs for model generation from requirements [40].

Reasoning quality and human-like judgments. In addition to acknowledging the limitations of language models in standard interpretation and domain-specific knowledge, industry experts consistently highlighted LLMs strong logical reasoning and human-like judgment in classifying requirements. One reviewer observed that, even when the LLM initially misclassified a requirement as ambiguous, its detailed rationale was often persuasive enough to change their judgment, *"I felt that this [requirement] was unambiguous, but reading the response from the LLM, I agree that it's somewhat ambiguous... the motivation is very good, and could have come from a colleague."* Similarly, the LLMs were further commended by reviewers for identifying poor requirement formulations, including ambiguous phrases (e.g., "provide to ensure"), non-quantifiable terms (e.g., "minimized"), and logical inconsistencies (e.g., mixing WHEN conditions with THEN actions), and particularly noted their precision in handling ambiguous constructs (e.g., "at least," "preferably").

In our context, although LLMs exhibit domain-specific limitations, experts value them for their ability to detect structural inconsistencies and well-formulated reasoning.

VII. Related Work

There is a substantial body of literature on requirements ambiguities—often referred to as *smells* [41] or *defects* [39]—and on approaches for automatic ambiguity detection [11]. A seminal work is the handbook by Berry et al. [42], which discusses the different types of ambiguities (i.e., lexical, syntactic, semantic, pragmatic), and provides extensive examples. Other theoretical contributions are those by Gervasi et al. [43], providing a unifying framework for

ambiguity in requirements specifications and interviews, and a recent chapter of the Handbook of Natural Language Processing for Requirements Engineering [44], which gives an overview of the established concepts and recent works.

Several tools and solutions have been developed to detect ambiguities. Part of them use rule-based or heuristic NLP approaches, while others propose machine learning techniques, including language models (LMs).

The first group includes tools like QuARS [45] and SREE [46], which are based on the automatic identification of ambiguous keywords or key-phrases in the requirements text (e.g., pronouns, adverbs, the terms *all* and *as possible*, etc.). With a more sophisticated approach, Chantree et al. [47] propose a set of heuristics to detect coordination ambiguities, which occur when words such as *and* or *or* are used, while other works, e.g., [48], [49], adopted knowledge graphs and search algorithms to detect pragmatic ambiguities, i.e., those occurring when the interpretation of the requirements depends on the context. Some industrial works have also been published using rule-based and heuristic approaches, such as Ferrari et al. [39], with an application in the railway domain, and Femmer et al. [41], who introduced the Smella tool and applied it to requirements in the automotive sector. More recently, Veizaga et al. [10] proposed Paska, a tool that uses rule-based NLP techniques combined with a controlled natural language to detect requirements smells, and applied the tool to the financial domain.

Among the works using machine learning solutions, Yang et al. [6] use a set of heuristics combined with the k-nearest neighbour algorithm to discover anaphoric ambiguities, i.e., those associated with pronouns such as *it*, *they*, etc. Bequiri et al. [50] propose a combination of NLP techniques and traditional machine learning algorithms for the detection of ambiguity in the railway domain, and include a module to support understanding of the words that could cause ambiguity. Another line of work from Ferrari and colleagues [21], [51], [52] uses LMs based on word embeddings to discover pragmatic ambiguities, with a particular focus on those ambiguities that can emerge when two readers belong to different domains. Ezzini et al. [9] uses a combination of machine learning solutions and LMs, i.e., BERT, for the detection of anaphoric ambiguities. Still with a focus on anaphoric ambiguities, Yildrim et al. [53] is one of the first works that employs generative LLMs to address the ambiguity problem in RE, showing that these solutions outperform more traditional classifiers.

Our research belongs to the group of approaches using machine learning and (L)LM-based solutions. The most similar works to ours are those by Yildrim et al. [53], using LLMs, and by Beqiri et al. [50], providing some forms of explanations for the ambiguities. Compared to Yildrim et al. [53], which focuses on anaphoric ambiguities only, we (i) address a broader perspective, (ii) consider an industrial case, and (iii) leverage LLMs to also provide *explanations* of the ambiguities, besides detection. Compared to Beqiri et al. [50], our explanations are not simply based on keywords, but are more articulated, human-like expressions. Overall, to our knowledge, this is the first contribution that: compares different LLMs; provides support for ambiguity explanation; considers an industrial case study; and includes an evaluation with practitioners.

VIII. Threats to Validity

Internal Validity. To address the variability introduced by LLM sampling and demonstration selection, we repeat each experiment three times using stratified random data splits and report average scores. Furthermore, we implemented our approach using standard libraries for both development and metric computation, and engaged multiple domain experts to evaluate the results independently. A common internal validity threat when using open-source LLMs is data leakage, as they may have been trained on publicly available data. However, our use of proprietary industrial datasets——which are not accessible online——eliminates this risk, making model familiarity with our data highly unlikely.

External Validity. Generalization is a known challenge in industry-driven research such as ours. While our results are based on two industrial datasets from Alstom and one from Westermo, thus covering multiple companies, they may not be generalizable beyond the studied contexts. Nevertheless, following case-based generalization principles [54], our findings could be transferable to similar domains, such as the railway and automotive industries.

IX. Conclusion and Future Directions

Detecting ambiguities in requirements and resolving them with multiple stakeholders is a common challenge and practice in large-scale companies to avoid project scope creep. Recent advances in LLMs provide support for this task due to their effective understanding with few examples and adaptability across datasets. However, there is limited empirical evidence on the performance of LLMs using in-context learning in industrial settings. In this paper, we present an empirical analysis across three industrial datasets to evaluate different configurations and measure the performance of LLMs in detecting ambiguities in requirements. Our results show that LLMs achieve competitive classification performance when provided with sufficient few-shot in-context demonstrations and generate explanations that practitioners find useful for clarifying and making well-informed decisions.

In the future, we aim to improve our approach by addressing the limitations identified by domain experts, with a focus on retrieving relevant, domain-specific glossaries and standards to improve the performance of LLMs.

Acknowledgement. This research has been partially supported by the KKS INDTECH industrial school (Grant No. 20200132) at MDU and by the AIDOaRt (Grant No. 101007350) and MATISSE (Grant No. 101140216) projects under the KDT program.

REFERENCES

[1] S. Bashir, M. Abbas, M. Saadatmand, E. P. Enoiu, M. Bohlin, and P. Lindberg, "Requirement or not, that is the question: A case from the railway industry," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2023, pp. 105–121.

[2] D. M. Berry, E. Kamsties, and M. M. Krieger, "From contract drafting to software specification: Linguistic sources of ambiguity," *A Handbook*, 2003.

[3] D. M. Fernández, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetrò, T. Conte, M.-T. Christiansson, D. Greer, C. Lassenius *et al.*, "Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice," *Empirical software engineering*, vol. 22, pp. 2298–2338, 2017.

[4] A. Bajceta, M. Leon, W. Afzal, P. Lindberg, and M. Bohlin, "Using NLP tools to detect ambiguities in system requirements-a comparison study." in *REFSQ Workshops*, 2022.

[5] G. Lami, *QuARS: A tool for analyzing requirements*. Carnegie Mellon University, Software Engineering Institute, 2005.

[6] H. Yang, A. De Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, "Analysing anaphoric ambiguity in natural language requirements," *Requirements engineering*, vol. 16, pp. 163–189, 2011.

[7] S. Alharbi, "Ambiguity detection in requirements classification task using fine-tuned transformation technique," in *CS & IT Conference Proceedings*, vol. 12, no. 21. CS & IT Conference Proceedings, 2022.

[8] A. Moharil and A. Sharma, "Tabasco: A transformer based contextualization toolkit," *Science of Computer Programming*, vol. 230, p. 102994, 2023.

[9] S. Ezzini, S. Abualhaija, C. Arora, and M. Sabetzadeh, "Automated handling of anaphoric ambiguity in requirements: a multi-solution study," in *Proceedings of the 44th international conference on software engineering*, 2022, pp. 187–199.

[10] A. Veizaga, S. Y. Shin, and L. C. Briand, "Automated smell detection and recommendation in natural language requirements," *IEEE Transactions on Software Engineering*, vol. 50, no. 4, pp. 695–720, 2024.

[11] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, and R. T. Batista-Navarro, "Natural language processing for requirements engineering: A systematic mapping study," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–41, 2021.

[12] M. Luo, X. Xu, Y. Liu, P. Pasupat, and M. Kazemi, "In-context learning with retrieved demonstrations for language models: A survey," *arXiv preprint arXiv:2401.11624*, 2024.

[13] M. S. Ibtasham, S. Bashir, M. Abbas, Z. Haider, M. Saadatmand, and A. Cicchetti, "Reqrag: Enhancing software release management through retrieval-augmented llms: An industrial study," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2025, pp. 277–292.

[14] M. Geng, S. Wang, D. Dong, H. Wang, G. Li, Z. Jin, X. Mao, and X. Liao, "Large language models are few-shot summarizers: Multi-intent comment generation via in-context learning," in *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, 2024, pp. 1–13.

[15] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, B. Chang *et al.*, "A survey on in-context learning," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 1107–1128.

[16] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[17] K. E. Wiegers and J. Beatty, *Software requirements*. Pearson Education, 2013.

[18] J. Dick, E. Hull, and K. Jackson, *Requirements engineering*. Springer, 2017.

[19] D. M. Berry and E. Kamsties, "Ambiguity in requirements specification," in *Perspectives on software requirements*. Springer, 2004, pp. 7–44.

[20] A. K. Massey, R. L. Rutledge, A. I. Antón, and P. P. Swire, "Identifying and classifying ambiguity for regulatory requirements," in *2014 IEEE 22nd international requirements engineering conference (RE)*. IEEE, 2014, pp. 83–92.

[21] A. Ferrari and A. Esuli, "An NLP approach for cross-domain ambiguity detection in requirements engineering," *Automated Software Engineering*, vol. 26, no. 3, pp. 559–598, 2019.

[22] M. Abbas, S. Bashir, M. Saadatmand, E. P. Enoiu, and D. Sundmark, "Requirements similarity and retrieval," in *Handbook on Natural Language Processing for Requirements Engineering*. Springer, 2025, pp. 61–88.

[23] S. Bashir, M. Abbas, A. Ferrari, M. Saadatmand, and P. Lindberg, "Requirements classification for smart allocation: A case study in the railway industry," in *2023 IEEE 31st International Requirements Engineering Conference (RE)*. IEEE, 2023, pp. 201–211.

[24] A. Plaat, A. Wong, S. Verberne, J. Broekens, N. van Stein, and T. Back, "Reasoning with large language models, a survey," *arXiv preprint arXiv:2407.11511*, 2024.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[27] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[28] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[29] Q. Chen, L. Qin, J. Liu, D. Peng, J. Guan, P. Wang, M. Hu, Y. Zhou, T. Gao, and W. Che, "Towards reasoning era: A survey of long chain-of-thought for reasoning large language models," *arXiv preprint arXiv:2503.09567*, 2025.

[30] X. ZHAO, J. LU, C. DENG, C. ZHENG, J. WANG, T. CHOWDHURY, L. YUN, H. CUI, Z. XUCHAO, T. ZHAO *et al.*, "Beyond one-model-fits-all: A survey of domain specialization for large language models," *arXiv preprint arXiv*, vol. 2305, 2023.

[31] A. Edwards and J. Camacho-Collados, "Language models for text classification: Is in-context learning enough?" *arXiv preprint arXiv:2403.17661*, 2024.

[32] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma *et al.*, "Scaling instruction-finetuned language models," *Journal of Machine Learning Research*, vol. 25, no. 70, pp. 1–53, 2024.

[33] X. Guo and S. Vosoughi, "Serial position effects of large language models," *arXiv preprint arXiv:2406.15981*, 2024.

[34] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[35] S. Zhang, L. Dong, X. Li, S. Zhang, X. Sun, S. Wang, J. Li, R. Hu, T. Zhang, F. Wu *et al.*, "Instruction tuning for large language models: A survey," *arXiv preprint arXiv:2308.10792*, 2023.

[36] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei *et al.*, "Qwen2. 5 technical report," *arXiv preprint arXiv:2412.15115*, 2024.

[37] Z. Li, Y. Liu, Y. Su, and N. Collier, "Prompt compression for large language models: A survey," *arXiv preprint arXiv:2410.12388*, 2024.

[38] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[39] A. Ferrari, G. Gori, B. Rosadini, I. Trotta, S. Bacherini, A. Fantechi, and S. Gnesi, "Detecting requirements defects with NLP patterns: an industrial experience in the railway domain," *Empirical Software Engineering*, vol. 23, no. 6, pp. 3684–3733, 2018.

[40] A. Ferrari, S. Abualhaijal, and C. Arora, "Model generation with llms: From requirements to uml sequence diagrams," in *2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*. IEEE, 2024, pp. 291–300.

[41] H. Femmer, D. M. Fernández, S. Wagner, and S. Eder, "Rapid quality assurance with requirements smells," *Journal of Systems and Software*, vol. 123, pp. 190–213, 2017.

[42] A. Handbook, "From contract drafting to software specification: Linguistic sources of ambiguity," 2003. [Online]. Available: /https://cs.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf

[43] V. Gervasi, A. Ferrari, D. Zowghi, and P. Spoletini, "Ambiguity in requirements engineering: Towards a unifying framework," *From Software Engineering to Formal Methods and Tools, and Back: Essays Dedicated to Stefania Gnesi on the Occasion of Her 65th Birthday*, pp. 191–210, 2019.

[44] D. M. Berry, E. Kamsties, C. Ribeiro, and S. F. Tjong, "Detecting defects in natural language requirements specifications," in *Handbook on Natural Language Processing for Requirements Engineering*. Springer, 2025, pp. 117–151.

[45] A. Bucchiarone, S. Gnesi, G. Lami, G. Trentanni, and A. Fantechi, "Quars express-a tool demonstration," in *2008 23rd IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 2008, pp. 473–474.

[46] S. F. Tjong and D. M. Berry, "The design of sree—a prototype potential ambiguity finder for requirements specifications and lessons learned," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2013, pp. 80–95.

[47] F. Chantree, B. Nuseibeh, A. De Roeck, and A. Willis, "Identifying nocuous ambiguities in natural language requirements," in *14th IEEE International Requirements Engineering Conference (RE'06)*. IEEE, 2006, pp. 59–68.

[48] A. Ferrari, G. Lipari, S. Gnesi, and G. O. Spagnolo, "Pragmatic ambiguity detection in natural language requirements," in *2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*. IEEE, 2014, pp. 1–8.

[49] A. Ferrari and S. Gnesi, "Using collective intelligence to detect pragmatic ambiguities," in *2012 20th IEEE International Requirements Engineering Conference (RE)*. IEEE, 2012, pp. 191–200.

[50] L. Beqiri, C. S. Montero, A. Cicchetti, and A. Kruglyak, "Classifying ambiguous requirements: An explainable approach in railway industry," in *2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*. IEEE, 2024, pp. 12–21.

[51] A. Ferrari, B. Donati, and S. Gnesi, "Detecting domain-specific ambiguities: an NLP approach based on wikipedia crawling and word embeddings," in *2017 IEEE 25th international requirements engineering conference workshops (REW)*. IEEE, 2017, pp. 393–399.

[52] A. Ferrari, A. Esuli, and S. Gnesi, "Identification of cross-domain ambiguity with language models," in *2018 5th international workshop on artificial intelligence for requirements engineering (AIRE)*. IEEE, 2018, pp. 31–38.

[53] S. Yildirim, G. Malik, M. Cevik, and A. Başar, "Anaphora resolution in software requirements engineering: A comparison of generative NLP pipelines and encoder-based models," in *2024 34th International Conference on Collaborative Advances in Software and COmputiNg (CASCON)*. IEEE, 2024, pp. 1–6.

[54] R. Wieringa and M. Daneva, "Six strategies for generalizing software engineering theories," *Science of computer programming*, vol. 101, pp. 136–152, 2015.