ReqRAG: Enhancing Software Release Management through Retrieval-Augmented LLMs: An Industrial Study

Md Saleh Ibtasham[3], Sarmad Bashir[1,2] ✉, Muhammad Abbas[1,2], Zulqarnain Haider[3], Mehrdad Saadatmand[1], and Antonio Cicchetti[2]

[1] RISE Research Institutes of Sweden, Västerås, Sweden, `{first.last}@ri.se`
[2] Mälardalen University, Västerås, Sweden, `{first.last}@mdu.se`
[3] Alstom, Västerås, Sweden, `{first.last}@alstomgroup.com`

Abstract. [Context and Motivation] Engineers often need to refer back to release notes, manuals, and system architecture documents to understand, modify, or upgrade functionalities in alignment with new software releases. This is crucial to ensure that new stakeholder requirements align with the existing system, maintaining compatibility and preventing integration issues. **[Problem]** In practice, the manual process of retrieving the relevant information from technical documentation is time-intensive and frequently results in inefficient software release management. **[Principal ideas/results]** In this paper, we propose a question-answering chatbot, *ReqRAG*, leveraging Retrieval Augmented Generation (RAG) with Large Language Models (LLMs) to deliver accurate and up-to-date information from technical documents in response to given queries. We employ various context retrieval techniques paired with state-of-the-art LLMs to evaluate the *ReqRAG* approach in industrial settings. Furthermore, we conduct human evaluations of the results in collaboration with experts from Alstom to gain practical insights. Our results indicate that, on average, 70% of the generated responses are adequate, useful, and relevant to the practitioners. **[Contribution]** Fewer studies have comprehensively evaluated RAG-based approaches in industrial settings. Therefore, this work provides technical considerations for domain-specific chatbots, guiding researchers and practitioners facing similar challenges.

Keywords: Software Release Management · Large Language Models (LLMs) · Retrieval Augmented Generation (RAG) · Industry Study

1 Introduction

In the domain of safety-critical systems, such as those in the railways industry, accurate and comprehensive documentation is the foundation for ensuring the system's safety, reliability, and integrity [30]. During software development, especially when introducing new versions or updates, associated requirements must remain traceable throughout the system lifecycle [3]. In this regard, the EN 50128

[1] railway control and protection system standard outlines clear guidelines for documenting software changes. These guidelines provide a structured approach to software release management, enabling tracking of each version and its modifications. Furthermore, efficient access to critical information during software release management is essential to ensure that each new release meets all specified requirements and compliance guidelines. However, the system's complexity often influences how information is organized and collected in different formats, such as architectural documents, interface control documents, and release notes [18]. Such technical documentation provides practitioners with relevant insights into system changes and dependencies. In practice, engineers must ensure that new software version requirements are traceable and consistently align with the overall system. This is often achieved by cross-referencing technical documentation, which validates that each change meets system-level requirements and regulatory standards. Engineers, therefore, commonly raise queries such as:

– *Which new or updated features and functionalities are implemented in release 1.2.1.0?*
– *Is release 1.2.1.0 backward compatible with the release 1.2.0.2?*

Manually accessing relevant information to the user queries is often tedious and prone to inconsistencies, as the required information is distributed over several documents. This scattered nature of data makes manual information retrieval time-consuming, which may result in inefficient software release processes. Therefore, an automated approach to generating accurate answers for requirements-related queries in technical documentation is required to address these challenges and improve efficiency in managing new software versions in complex systems.

In this industry-driven research, we collaborated closely with Alstom Sweden (Alstom), a world-leading railway vehicle manufacturing company. The primary objective is to investigate and propose an automated solution that leverages NLP techniques, particularly Large Language Models (LLMs), to deliver precise answers to user queries based on Alstom's technical documentation, thereby enhancing efficiency in managing software releases.

While LLMs have shown great performance across various software engineering tasks due to their extensive training on diverse datasets, their inherent knowledge remains limited to their initial training [37]. As a result, pre-trained LLMs struggle to reliably answer domain-specific queries without additional fine-tuning, which is costly given the size of LLMs [34]. To address this, our approach *ReqRAG* incorporates a well-known Retrieval Augmented Generation (RAG) technique [15] coupled with pre-trained LLMs. *ReqRAG* first retrieves relevant information from a knowledge base of technical documents in response to user queries, then provides this context to LLMs to support the generation of accurate and domain-specific answers. We evaluated various information retrieval techniques and generative LLMs within *ReqRAG* approach to assess their performance. Additionally, we conducted a human evaluation of the results to further provide valuable practical insights into their effectiveness.

The rest of the paper is structured as follows: Section 2 provides a brief overview of the background and related work. Section 3 presents the study design of our proposed *ReqRAG* approach. Section 4 presents and discusses the results. Section 5 presents potential validity threats and limitations. Finally, Section 6 concludes the paper with future directions.

2 Background & Related Work

2.1 Background

In this section, we detail the fundamental concepts used in our *ReqRAG* approach, i.e., Information Retrieval and Large Language Models (LLMs).

Information Retrieval (IR). IR is an NLP technique that efficiently identifies and ranks relevant information from a large corpus in response to user queries [23]. Traditional IR techniques, such as TF-IDF and its variation BM25, laid the foundation for evaluating textual similarity in retrieval tasks such as document retrieval and recommender systems. These methods represent the textual data as sparse high-dimensional feature vectors for retrieval; however, they do not capture the semantic relationships between different textual attributes. On the other hand, dense-vector representations based on deep learning models effectively map semantic meaning, enabling accurate context retrieval [20].

Large Language Models. A Language Model (LM) learns a probability distribution over a sequence of words to perform various natural language processing (NLP) tasks, such as topic modeling and classification [8,7]. The advent of modern Large Language Models (LLMs) began with the introduction of transformer architecture proposed by Vaswani et al. [32]. The approach incorporated a self-attention mechanism to better capture the relationship between data attributes. This was followed by the release of the BERT (Bidirectional Encoder Representations from Transformers) model [11], an encoder-only model that improved language understanding by learning bidirectional representations. The evolution of transformer-based architecture continued with the arrival of Generative Pre-trained Transformers (GPT) [25] and its successive iterations [26,9], which significantly improved performance in various NLP tasks such as language translation, summarization, and question answering systems [17].

However, pre-trained LLMs often hallucinate in specialized domains and provide incorrect information because of their inability to access factual information, particularly from proprietary knowledge bases [31,40]. In this regard, IR techniques could leverage specialized knowledge sources to retrieve relevant and verified information that can inform the LLMs to generate factually accurate responses. This combination of IR techniques with LLMs, also known as Retrieval-Augmented Generation (RAG), enables LLMs to access up-to-date proprietary knowledge rather than relying solely on pre-existing training data [29].

2.2 Related Work

Recently, the RAG framework has been employed in software engineering tasks such as requirements traceability, test case generation, and code understanding

[14]. Ali et al. [5] proposed a RAG-based chatbot to enhance traceability between natural language requirements and code using Llama 3 LLM. Similarly, Ezzini et al. [13] develop a question-answering assistant (QAssist), which analyzes requirements to improve their quality. QAassist utilizes domain-specific corpus along with an external knowledge base to answer requirement-related queries. The tool uses BERT LLM variants, which highlight answers in the passages given user queries in a non-conversational manner.

In another work, Arora et al. [6] proposed a domain-specific RAG-based LLM approach (RAGTAG) to generate test cases using natural language requirements. They employed few-shot and zero-shot prompting techniques to compare RAGTAG using GPT-3.5-turbo and GPT-4.0. Chaudhary et al. [10] develop a chatbot for continuous integration and continuous delivery (CI/CD) documents. To evaluate retrieved-context, they compared multiple retrieval techniques, such as TF-IDF and BM25. For answer generation, the authors employed a Llama 2 LLM to respond to CI/CD-related queries. In another recent study, Abedu et al. [4] utilize a RAG-powered chatbot to process repositories for QA tasks dynamically. The authors used OpenAI ada-002 embeddings for context retrieval and generated answers using the GPT-3.5-turbo model. Similarly, Veturi et al. [33] used GPT-3.5-turbo to generate response suggestions for customer service agents in retail contact centers and compared multiple dense embedding techniques for context retrieval.

While the above-mentioned work shares the same general objective as our study, it does not specifically address our goal: answering requirement-related queries during software release management. Additionally, in our study, the units under analysis are comprised of PDF documents, which require different retrieval and preprocessing methods to enable effective context augmentation for LLMs. To the best of our knowledge, this study is the first to develop and comprehensively evaluate a chatbot specifically tailored to answer queries in the context of software release management.

3 Study Design

This section outlines the methodology for developing and evaluating *ReqRAG* at Alstom. Following, we describe the industrial context, research questions, data collection, *ReqRAG* approach and evaluation metrics.

3.1 Industrial context

Software release management is crucial in the rail manufacturing industry because of the safety-critical nature of systems and strict regulatory requirements. Engineers not only support end-users by patching new code with legacy systems and resolving compatibility issues but also ensure that each update aligns with existing system-level requirements. In this regard, engineers frequently consult technical documentation to ensure compatibility in these regulatory environments to access necessary information for software upgrades, requirement changes, or modifications. For instance, determining the exact features implemented in a release or verifying backward compatibility with previous releases is important to maintaining reliability and regulatory compliance. However, due to the ever-increasing customer requirements and market competition, Alstom continuously seeks to enhance its processes to support engineers in effective software release management. Therefore, in this paper, we investigate and propose a domain-specific chatbot, *ReqRAG*, that gives up-to-date answers for user queries in technical documentation. Seven technical documents from Alstom are the units under analysis, addressing requirements-related queries relevant to software release management.

3.2 Research Questions

The objective of the study is to enhance the software release management process for practitioners in the studied context by improving the efficient retrieval of relevant information and providing accurate answers to their domain-specific queries. Therefore, we explore multiple information retrieval techniques to extract relevant context for the queries and then employ various LLMs to generate accurate answers. We aim to identify the configurations that present the best results for our proposed *ReqRAG* approach. To this end, we define the following research questions:

- ***RQ1: Which information retrieval technique provides the most accurate context for ReqRAG?***
- ***RQ2: Which Large Language Model yields the best results in answer generation?***
- ***RQ3: How useful do practitioners find ReqRAG's responses?***

3.3 Data Collection

We had access to seven industrial documents from Alstom to create the knowledge base. These technical documents include the release notes, architectural documents, and interface control documents from the Train Control Management System (TCMS). Additionally, the documents contain relevant information that engineers must refer to during new software releases. This is because the release notes track changes in software modifications, while interface control documents define how different railway systems communicate and interact. Similarly, architectural documents outline the overall system design and safety mechanisms.

For the evaluation of *ReqRAG*, we obtained a diverse set of 27 queries with ground truth answers from Alstom. This collection represents real-world questions that practitioners ask and address regularly, which includes, but is not limited to, clarification and traceability of requirements for new software versions.

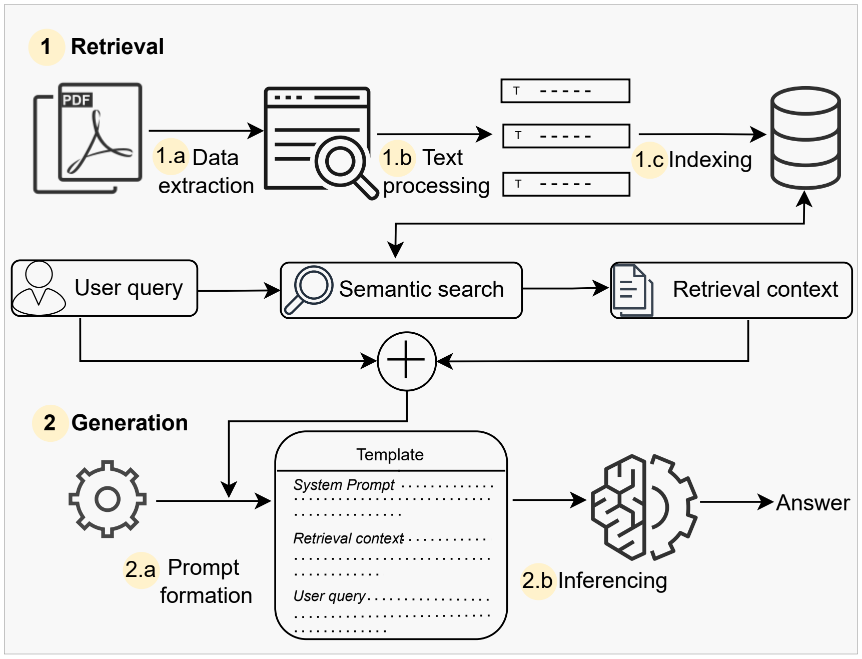


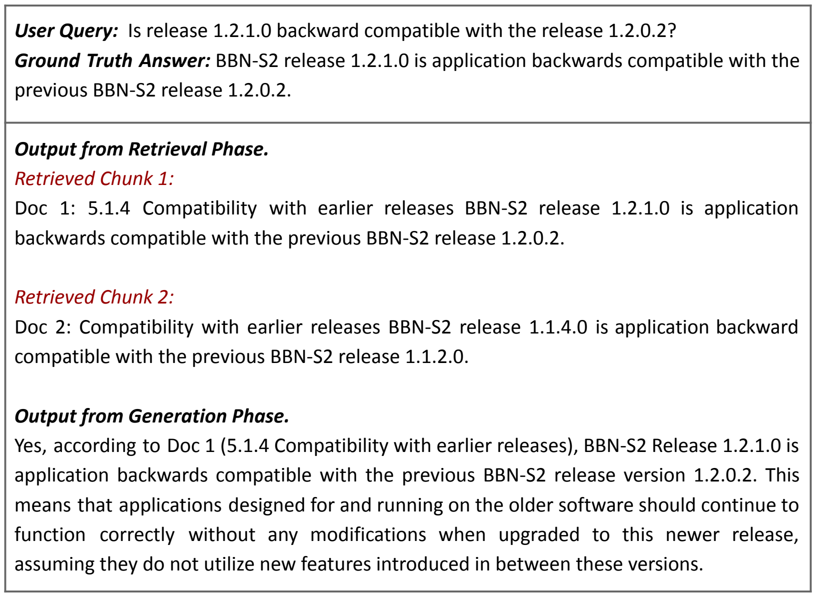
Fig. 1: Overview of the proposed *ReqRAG* approach

3.4 ReqRAG Approach

Figure 1 provides an overview of the proposed *ReqRAG* approach, highlighting its two primary phases: Retrieval and Generation. Figure 2 shows a working example, demonstrating a user query, ground truth answer, and its corresponding response from the *ReqRAG* phases. Below, we detail the *ReqRAG* phases and the design decisions involved.

1. Retrieval. In this phase, the aim is to extract the relevant information from different data sources to infer LLM for contextually aware and accurate answers. For this purpose, we must first create a knowledge base that stores the relevant information for retrieval. Therefore, we created a data ingestion pipeline that systematically transforms raw data—PDF technical documents in our case—to a searchable vector format that can be utilized for the semantic search of user queries and their relevant information. In this regard, we perform the following steps for the retrieval phase:

1.a Data extraction. In general, accurate data extraction from PDF documents presents significant challenges because of their complex layouts consisting of embedded fonts, tables, and images [38]. Therefore, for effective data extraction from PDF files, we employed Optical Character Recognition (OCR) techniques. OCR models support structured text extraction, which is crucial for efficient text analysis and retrieval in our case. In particular, we compare two



User Query: Is release 1.2.1.0 backward compatible with the release 1.2.0.2?

Ground Truth Answer: BBN-S2 release 1.2.1.0 is application backwards compatible with the previous BBN-S2 release 1.2.0.2.

Output from Retrieval Phase.

Retrieved Chunk 1:

Doc 1: 5.1.4 Compatibility with earlier releases BBN-S2 release 1.2.1.0 is application backwards compatible with the previous BBN-S2 release 1.2.0.2.

Retrieved Chunk 2:

Doc 2: Compatibility with earlier releases BBN-S2 release 1.1.4.0 is application backward compatible with the previous BBN-S2 release 1.1.2.0.

Output from Generation Phase.

Yes, according to Doc 1 (5.1.4 Compatibility with earlier releases), BBN-S2 Release 1.2.1.0 is application backwards compatible with the previous BBN-S2 release version 1.2.0.2. This means that applications designed for and running on the older software should continue to function correctly without any modifications when upgraded to this newer release, assuming they do not utilize new features introduced in between these versions.

Fig. 2: An example user query and corresponding output from the *ReqRAG* phases.

popular open-source OCR models for data extraction, namely YOLOX [16] and Detectron2 (Det2) [36] (see section 4 for details).

1.b Text processing. Once the raw structured data is extracted through the OCR model, refining or preprocessing the text is crucial so that only relevant information is stored for the subsequent stages. Moreover, raw data extracted from PDF documents often includes noisy and irrelevant elements, which reduces the effectiveness of relevant information retrieval. To address this, we employ rule-based filtering to remove common noise elements such as headers, footers, and page numbers and use regular expressions to remove table of contents pages from the PDF documents.

Following text cleaning, we segment the data into semantic units or so-called chunks to store them within a vector database. Chunking the data into smaller semantic units improves contextual relevancy and addresses the issue of LLMs' limited context window, which may be insufficient for processing entire documents [33]. In our case, we explored different sizes and determined that 1600 characters are optimal for each chunk document before exceeding the context window length of considered LLMs for answer generation.

1.c Indexing. In this step, we process the document chunks to create vector embeddings and store them within a vector database, Milvus [35]. The rationale behind selecting the Milvus database is its advanced query processing feature, which allows efficient retrieval of relevant document chunks. Specifically, we employ *IVF-FLAT* [19] indexing method, an Approximate Nearest Neighbor Search

(ANNS) technique, which organizes the data into clusters for efficient retrieval during semantic search. For a given query, the ANNS first identifies the clusters closest to the query vector embedding and then searches within these clusters to find the nearest neighbors to return the relevant context.

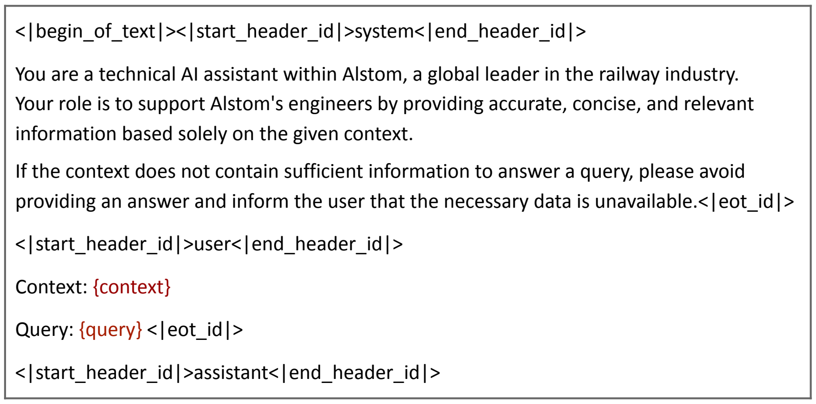
To create vector embeddings, we select and compare two techniques, i.e., mxbai-embed-large-v1 [4] (maxbai-L) and stella_en_400M_v5 [5] (stella_v5) because of their high performance in Massive Text Embedding Benchmark (MTEB) [24] (see section4 for results). Notably, the steps involved in the retrieval phase are executed only once during the initial setup, where the chunk vector embeddings are stored in the database. Subsequently, for each query, only the relevant context chunks are retrieved from the database to generate the final answers from LLMs.

2. Generation. During the Generation phase, we prompt the generative LLMs to provide accurate domain-specific answers by augmenting with the retrieved context documents given the user query. In the following, we explain the steps involved in the generation phase:

2.a Prompt formation. When a user provides a query, the retrieval phase selects the top-k document chunks from the knowledge base, combined with the task instructions for the LLMs and the original query to structure an input prompt template. Figure 3 shows an example of our prompt template. In this template, we define the *system* instructions to establish a *role-play* that guides the LLM in understanding the environment and ensures that its responses are well aligned with the specific domain context. Moreover, in cases where the LLM cannot provide an accurate answer against a user query because of insufficient context, it is instructed to inform the user that it lacks the necessary data. This approach helps prevent potential hallucinations, a common limitation in LLMs, by setting clear boundaries. The remaining part of the template contains temporary fields for context documents and the user's query, which are substituted at runtime.

2.b Inferencing. Once the prompt template is populated with the relevant context chunk documents and user query, we infer answers from the LLM. To select the best performing model for *ReqRAG*, we compare three open-source LLMs: *Phi3-mini (3.8B)*[6] *gemma-2 (2B)*[7] *Llama-3.2 (1B)*[8]. The rationale for selecting these LLMs for comparison is their smaller size (1B to 3.8B parameters), which provides a balance between performance and computational efficiency. These LLMs' open-source nature also meets our strict data privacy and confidentiality requirements. We specifically employ instruct-tuned variants of these LLMs over base versions because of their improved ability to understand and respond accurately to given prompts [39]. For evaluation, we set the LLMs'

[4] `https://huggingface.co/mixedbread-ai/mxbai-embed-large-v1`
[5] `https://huggingface.co/dunzhang/stella_en_400M_v5`
[6] `https://huggingface.co/microsoft/Phi-3-mini-128k-instruct`
[7] `https://huggingface.co/google/gemma-2-2b-it`
[8] `https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct`



```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are a technical AI assistant within Alstom, a global leader in the railway industry.
Your role is to support Alstom's engineers by providing accurate, concise, and relevant
information based solely on the given context.
If the context does not contain sufficient information to answer a query, please avoid
providing an answer and inform the user that the necessary data is unavailable.<|eot_id|>

<|start_header_id|>user<|end_header_id|>

Context: {context}

Query: {query} <|eot_id|>

<|start_header_id|>assistant<|end_header_id|>
```

Fig. 3: An example of prompt template.

temperature configuration to zero to minimize the randomness in the answers generated by the LLMs.

3.5 Evaluation Metrics

We comprehensively evaluate our *ReqRAG* approach by employing a set of evaluation metrics for both the retrieval and generation phases. These evaluation metrics provide quantitative scores and practical insights into the effectiveness of our approach in retrieving relevant context and generating accurate answers.

Retrieval Phase Evaluation. In the retrieval phase, we measure *Context Recall* using LLM as an evaluator. In particular, we calculate $R@k$ that measures the retrieval component by calculating the proportion of the relevant context within the top k results given the ground truth reference. The value of k represents the number of retrieved-context document chunks considered for the evaluation. We use the prompt template provided by Ragas [12] to calculate $R@k$ with an open-source Llama3 8B [9] instruction tuned model.

Generation Phase Evaluation. We employ widely adopted NLP metrics, *BLEU*, *ROUGE*, and *METEOR*, to evaluate the quality of generated answers [28]. These metrics are standard in evaluating the overlap between the generated text and reference ground truth. In particular, the *BLEU* metric focuses on precision by evaluating the *n-grams* co-occurrence between machine-generated text and those in human reference text. *ROUGE* metric, commonly used to evaluate summarization tasks, calculates recall by estimating the amount of reference text overlapping in the generated result. *METEOR*, an extension of *BLEU*, focuses on precision and recall and considers synonyms and stemming to evaluate linguistic similarities between generated text and human judgment.

[9] `https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct`

Table 1: Results of different embedding techniques with OCR models for *ReqRAG* retrieval phase.

Embedding	**OCR**	**R@1**	**R@2**	**R@3**
mxbai-L v1	YoloX	0.69	0.73	0.75
	Det2	0.72	0.79	0.74
stella_v5	YoloX	0.73	0.79	0.76
	Det2	0.62	0.83	0.90

In addition to using traditional NLP metrics for evaluation, we calculate answer similarity (*Answer Sim.*) by measuring the semantic relatedness between the LLM generated and the ground-truth answer. We first created dense embedding vectors for both LLM-generated answers and ground truth, then calculated their cosine similarity. We selected the stsb-roberta-base-v2[10] variant of sentence-bert to create the embeddings because of their good performance in semantic similarity tasks [2].

Human Evaluation. Apart from the quantitative evaluation of retrieval and generation phases of the proposed *ReqRAG* approach, we also performed human evaluation to gain insights into the practical utility and quality of the generated answers. We conducted the human evaluation based on the following criteria:

- *Adequacy* criterion evaluates the quality of the responses based on their completeness and richness.
- *Usefulness* measures the practical value of LLM-generated answers in helping practitioners understand and perform their tasks effectively.
- *Relevance* metric examines how closely the responses relate to the ground truth, evaluating its accuracy with the correct information.

Each above-mentioned criterion is evaluated by experts at Alstom using a five-point Likert-scale [21] (1 for poor, 2 for marginal, 3 for acceptable, 4 for good, and 5 for excellent).

4 Results and Discussion

Table 1 and Table 2 show the evaluation results for the *Retrieval* and *Generation* phases of our approach, respectively.

RQ1: **Retrieval performance.** For answering *RQ1*, we evaluated multiple embedding techniques coupled with OCR models on the context retrieval task. As Table 1 shows, maxbai-L embedding technique combined with YOLOX OCR, the *R@1* on average is 0.69, which increases to 0.72 when using Det2. However, *R@3* score slightly decreases from 0.75 with YoloX to 0.74, indicating

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

Table 2: The results of LLMs for *ReqRAG* generation phase.

LLM	Retrieval	Chunks	BLEU	ROUGE	METEOR	Answer Sim.
Phi-3-mini (3.8B)	mxbai-L v1, YoloX	1	0.08	0.28	0.22	0.56
		2	0.22	0.37	0.30	0.58
		3	0.20	0.37	0.34	0.62
	mxbai-L v1, Det2	1	0.08	0.23	0.18	0.51
		2	0.21	0.33	0.30	0.59
		3	0.17	0.36	0.34	0.62
	Stella_v5, YoloX	1	0.10	0.27	0.20	0.53
		2	0.18	0.33	0.27	0.55
		3	0.19	0.36	0.32	0.62
	Stella_v5, Det2	1	0.08	0.27	0.22	0.54
		2	0.18	0.36	0.30	0.63
		3	0.17	0.38	0.30	0.64
gemma-2 (2B)	mxbai-L v1, YoloX	1	0.08	0.31	0.23	0.55
		2	0.28	0.39	0.33	0.62
		3	0.25	0.42	0.36	0.64
	mxbai-L v1, Det2	1	0.06	0.23	0.17	0.50
		2	0.22	0.35	0.30	0.57
		3	0.19	0.33	0.30	0.60
	Stella_v5, YoloX	1	0.12	0.28	0.19	0.51
		2	0.22	0.37	0.29	0.58
		3	0.21	0.35	0.29	0.60
	Stella_v5, Det2	1	0.06	0.27	0.19	0.51
		2	0.17	0.38	0.30	0.60
		3	0.18	0.39	0.31	0.64
Llama-3.2 (1B)	mxbai-L v1, YoloX	1	0.08	0.30	0.26	0.60
		2	0.15	0.37	0.34	0.63
		3	0.13	0.35	0.32	0.66
	mxbai-L v1, Det2	1	0.08	0.29	0.24	0.61
		2	0.12	0.30	0.27	0.61
		3	0.09	0.28	0.26	0.61
	Stella_v5, YoloX	1	0.06	0.27	0.23	0.58
		2	0.08	0.30	0.27	0.62
		3	0.08	0.32	0.30	0.63
	Stella_v5, Det2	1	0.05	0.25	0.20	0.58
		2	0.12	0.32	0.29	0.60
		3	0.10	0.33	0.29	0.65

that YOLOX may retrieve more relevant contexts within the top three document chunks in some cases. For the stella_v5 embedding, *R@1* is higher with YOLOX (0.73) compared to Det2 (0.62). On the other hand, *R@3* with Det2 significantly increases to 0.90, outperforming all the other combinations and indicating that the correct context is highly likely to be retrieved within the top three results. Counter-intuitively, in some configurations, such as maxibai-L with Det2 and stella_v5 with YOLOX, *R@2* is slightly better than *R@3*. This could be explained by the fact that increasing the number of retrieved-context documents may not always increase recall, as additional documents may introduce irrelevant or redundant information, potentially degrading retrieval performance.

Nevertheless, ensuring that each retrieved document adds unique and relevant information is crucial to improving retrieval performance. Based on these results, we summarize an answer to RQ1.

Answer to RQ1. In our dataset evaluation, stella_v5 embeddings with Det2 OCR achieve the best average recall within the top three result document chunks (with a context recall score of 0.90). However, careful consideration is required to select the optimal number of context document chunks to ensure each adds relevant information to the retrieval process.

RQ2: **LLM Generation performance.** The results for the generation phase of *ReqRAG* are summarized in Table 2. To answer RQ2, we comprehensively evaluate the considered LLMs with different retrieval methods and document chunk sizes. Our findings indicate a clear relationship between the size of LLMs and their performance across traditional NLP metrics. In all configurations, Phi-3-mini achieved the highest average performance in the majority of cases, with BLEU = 0.15, ROUGE = 0.33, and METEOR = 0.27. Similarly, gemma-2 scored the highest on ROUGE (0.34) and performed moderately on others, while Llama-3.2 scored the lowest on traditional metrics on average. However, interestingly, the Answer Sim. metric shows that the smallest model, Llama-3.2, performed the best with an average score of 0.62, followed by phi3-mini (0.58) and gemma (0.57). Moreover, we observe that document chunk size directly influences the quality of the generated answers. In particular, ROUGE and METEOR metrics perform similarly to or slightly better with a document chunk size of 3 compared to a chunk size of 2 across all LLMs and configurations. In contrast, the BLEU score consistently performed better with chunk size 2 than with chunk size 3. This is because the BLEU metric *n-gram* precision approach penalizes irrelevant information that a larger chunk may introduce, which results in reduced alignment with the generated answer, while ROUGE focuses on *n-gram* recall and METEOR metric accounts for text variations. On the other hand, Answer Sim. metric, which calculates the semantic relatedness based on dense embedding vectors, shows a linear relationship with document chunk sizes, where an increase in chunk size corresponds to higher scores. For example, in the case of gemma-2 configured with Stella_v5 and Det2, the Answer Sim. score improved from 0.51 at document chunk size 1 to 0.60 at 2 and 0.64 at 3. A similar progression of score can be observed across all configurations, which shows that chunk size 3 enhances the LLM's semantic alignment with the reference answers by providing richer context information.

Answer to RQ2. In general, Phi-3-mini yields best results for NLP metrics, while Llama-3.2 excelled in Answer Sim. metric. Moreover, we observed that chunk size 3 improves the semantic alignment of generated answers with reference text and provides the best results across all configurations.

RQ3: **Human Evaluation results.** We conducted a human evaluation to assess further the quality of *ReqRAG* responses using a five-point Likert scale. Specifically, we select the responses from best-performing configurations of the retrieval phase (Stella_v5, Det2) paired with LLM (gemma-2). Based on the evaluation criteria, we asked four industry experts to rate *ReqRAG* responses. The results from human evaluation show that, on average, 70% of the responses met the criteria, with specific average (avg.) and standard deviation (std.) scores as follows: Adequacy avg. (std.) = 3.69 (1.54), Usefulness avg. (std.) = 3.44 (1.68), and Relevance avg. (std.) = 3.32 (1.64).

Adequacy. ReqRAG-generated responses achieved an average adequacy score of 3.69, which suggests that it effectively covers the important aspects of practitioners' queries, and the experts generally found the response to be acceptable in terms of quality. However, the standard deviation (1.54) shows moderate variability in experts evaluation.

Usefulness. The average score is 3.44, with experts agreeing that the responses are valuable in understanding and effectively performing their tasks. Additionally, experts noted that the answers provide actionable insights into requirement-specific queries, effectively supporting software release management. The standard deviation score of 1.68 shows noticeable variability in experts' perception of usefulness.

Relevance. The average relevance score of 3.32 indicates that experts found the responses reasonably aligned with the ground truth references, providing accurate information. However, the standard deviation of 1.64 shows significant variability in experts' perceptions, suggesting that some responses closely match the expected information while others are only partially aligned.

During the evaluation, experts also noted that some queries were not formulated clearly, which led to the responses not being completely aligned with the ground truth information. This could be an area of improvement to rewriting the queries in an automated fashion, as queries are not always well-defined or structured [22].

Answer to RQ3. The evaluation shows that experts find *ReqRAG*-generated responses acceptable across the evaluation criteria, with average scores of 3.69 for adequacy, 3.44 for usefulness, and 3.32 for relevance. However, the moderate variability in assessment suggests potential areas of improvement in refining the queries and generated responses.

5 Threats to Validity

Construct validity. To address this, we conducted both quantitative and human evaluations for *ReqRAG*. For quantitative analysis, we calculate context recall, traditional NLP metrics, and semantic similarity to assess the performance of our approach. These metrics have been widely adopted to assess the performance of RAG-based systems. Furthermore, we analyzed results with practitioners to validate our approach.

Internal validity. One possible internal validity threat could arise from using open-source LLMs in our approach. Given that our approach was evaluated on the proprietary industrial dataset, we mitigate the possibility of LLMs being familiar with the data during its pre-training.

External validity. Generalization has always been a concern in industrial case studies. We conducted our evaluations on a domain-specific dataset and plan to conduct further evaluations on extended datasets. However, we do not claim the generalizability of our results beyond the studied context.

6 Conclusion and Future Work

This study is oriented toward supporting engineers in software release management by providing efficient access to relevant information addressing requirement-related queries for new software version releases. Therefore, we introduce *ReqRAG*, an automated solution designed to deliver accurate answers to user queries within technical documentation at Alstom. *ReqRAG* leverages Large Language Models (LLMs) and adopts the Retrieval-Augmented Generation (RAG) technique to enhance answers quality with relevant, domain-specific context. We conducted an empirical evaluation on an industrial dataset, exploring various configurations to assess the performance and select the best approach for the final design. Moreover, we conducted a human evaluation to gather insights from the practitioners on the effectiveness of the approach. To this end, we plan to improve the *ReqRAG* approach by re-writing given user queries to ensure they are well-structured for processing by LLMs. Furthermore, we plan to extend the evaluation with other document sets to assess the generalizability of our approach.

Acknowledgement. The KKS has partially funded this research work through the ARRAY and INDTECH industrial schools at MDU, as well as the AIDOaRt (KDT) and SmartDelta [27] (ITEA) projects.
Competing Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Railway applications - communication, signalling and processing systems - software for railway control and protection systems (2020), `https://standards.globalspec.com/std/14317747/EN%2050128`
2. Abbas, M., Bashir, S., Saadatmand, M., Enoiu, E.P., Sundmark, D.: 3. requirements similarity and retrieval. pp. 61–87. Springer (December 2024). `https://doi.org/10.1007/978-3-031-73143-3_3`
3. Abbas, M., Ferrari, A., Shatnawi, A., Enoiu, E., Saadatmand, M., Sundmark, D.: On the relationship between similar requirements and similar software: A case study in the railway domain. Requirements Engineering **28**(1), 23–47 (2023)
4. Abedu, S., Abdellatif, A., Shihab, E.: Llm-based chatbots for mining software repositories: Challenges and opportunities. In: Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering. pp. 201–210 (2024)
5. Ali, S.J., Naganathan, V., Bork, D.: Establishing traceability between natural language requirements and software artifacts by combining rag and llms. In: International Conference on Conceptual Modeling. pp. 295–314. Springer (2024)

6. Arora, C., Herda, T., Homm, V.: Generating test scenarios from nl requirements using retrieval-augmented llms: An industrial study. arXiv preprint arXiv:2404.12772 (2024)
7. Bashir, S., Abbas, M., Ferrari, A., Saadatmand, M., Lindberg, P.: Requirements classification for smart allocation: A case study in the railway industry. In: 2023 IEEE 31st International Requirements Engineering Conference (RE). pp. 201–211. IEEE (2023)
8. Bashir, S., Abbas, M., Saadatmand, M., Enoiu, E.P., Bohlin, M., Lindberg, P.: Requirement or not, that is the question: A case from the railway industry. In: International Working Conference on Requirements Engineering: Foundation for Software Quality. pp. 105–121. Springer (2023)
9. Brown, T.B.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
10. Chaudhary, D., Vadlamani, S.L., Thomas, D., Nejati, S., Sabetzadeh, M.: Developing a llama-based chatbot for ci/cd question answering: A case study at ericsson. arXiv preprint arXiv:2408.09277 (2024)
11. Devlin, J.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
12. Es, S., James, J., Espinosa-Anke, L., Schockaert, S.: Ragas: Automated evaluation of retrieval augmented generation. arXiv preprint arXiv:2309.15217 (2023)
13. Ezzini, S., Abualhaija, S., Arora, C., Sabetzadeh, M.: Ai-based question answering assistance for analyzing natural-language requirements. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). pp. 1277–1289. IEEE (2023)
14. Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.S., Li, Q.: A survey on rag meeting llms: Towards retrieval-augmented large language models. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 6491–6501 (2024)
15. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., Wang, H.: Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997 (2023)
16. Ge, Z.: Yolox: Exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430 (2021)
17. Hadi, M.U., Qureshi, R., Shah, A., Irfan, M., Zafar, A., Shaikh, M.B., Akhtar, N., Wu, J., Mirjalili, S., et al.: A survey on large language models: Applications, challenges, limitations, and practical usage. Authorea Preprints (2023)
18. Jansen, A., Bosch, J., Avgeriou, P.: Documenting after the fact: Recovering architectural design decisions. Journal of Systems and Software **81**(4), 536–557 (2008)
19. Jegou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE transactions on pattern analysis and machine intelligence **33**(1), 117–128 (2010)
20. Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W.t.: Dense passage retrieval for open-domain question answering. arXiv preprint arXiv:2004.04906 (2020)
21. Likert, R.: A technique for the measurement of attitudes. Archives of Psychology (1932)
22. Ma, X., Gong, Y., He, P., Zhao, H., Duan, N.: Query rewriting for retrieval-augmented large language models. arXiv preprint arXiv:2305.14283 (2023)
23. Manning, C.D.: An introduction to information retrieval (2009)

24. Muennighoff, N., Tazi, N., Magne, L., Reimers, N.: Mteb: Massive text embedding benchmark. arXiv preprint arXiv:2210.07316 (2022). https://doi.org/10.48550/ARXIV.2210.07316, https://arxiv.org/abs/2210.07316
25. Radford, A.: Improving language understanding by generative pre-training (2018)
26. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
27. Saadatmand, M., Abbas, M., Enoiu, E.P., Schlingloff, B.H., Afzal, W., Dornauer, B., Felderer, M.: Smartdelta project: Automated quality assurance and optimization across product versions and variants. Microprocessors and microsystems **103**, 104967 (2023)
28. Sai, A.B., Mohankumar, A.K., Khapra, M.M.: A survey of evaluation metrics used for nlg systems. ACM Computing Surveys (CSUR) **55**(2), 1–39 (2022)
29. Schulhoff, S., Ilie, M., Balepur, N., Kahadze, K., Liu, A., Si, C., Li, Y., Gupta, A., Han, H., Schulhoff, S., et al.: The prompt report: A systematic survey of prompting techniques. arXiv preprint arXiv:2406.06608 (2024)
30. Smith, D.J., Simpson, K.G.: The safety critical systems handbook: a straightforward guide to functional safety: IEC 61508 (2010 Edition), IEC 61511 (2015 edition) and related guidance. Butterworth-Heinemann (2020)
31. Tonmoy, S., Zaman, S., Jain, V., Rani, A., Rawte, V., Chadha, A., Das, A.: A comprehensive survey of hallucination mitigation techniques in large language models. arXiv preprint arXiv:2401.01313 (2024)
32. Vaswani, A.: Attention is all you need. Advances in Neural Information Processing Systems (2017)
33. Veturi, S., Vaichal, S., Tripto, N.I., Jagadheesh, R.L., Yan, N.: Rag based question-answering for contextual response prediction system. arXiv preprint arXiv:2409.03708 (2024)
34. Wang, C., Liu, X., Yue, Y., Tang, X., Zhang, T., Jiayang, C., Yao, Y., Gao, W., Hu, X., Qi, Z., et al.: Survey on factuality in large language models: Knowledge, retrieval and domain-specificity. arXiv preprint arXiv:2310.07521 (2023)
35. Wang, J., Yi, X., Guo, R., Jin, H., Xu, P., Li, S., Wang, X., Guo, X., Li, C., Xu, X., et al.: Milvus: A purpose-built vector data management system. In: Proceedings of the 2021 International Conference on Management of Data. pp. 2614–2627 (2021)
36. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019)
37. Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., Zhong, S., Yin, B., Hu, X.: Harnessing the power of llms in practice: A survey on chatgpt and beyond. ACM Transactions on Knowledge Discovery from Data **18**(6), 1–32 (2024)
38. Zhang, Q., Huang, V.S.J., Wang, B., Zhang, J., Wang, Z., Liang, H., Wang, S., Lin, M., Zhang, W., He, C.: Document parsing unveiled: Techniques, challenges, and prospects for structured information extraction. arXiv preprint arXiv:2410.21169 (2024)
39. Zhang, S., Dong, L., Li, X., Zhang, S., Sun, X., Wang, S., Li, J., Hu, R., Zhang, T., Wu, F., et al.: Instruction tuning for large language models: A survey. arXiv preprint arXiv:2308.10792 (2023)
40. Zhang, Y., Li, Y., Cui, L., Cai, D., Liu, L., Fu, T., Huang, X., Zhao, E., Zhang, Y., Chen, Y., et al.: Siren's song in the ai ocean: a survey on hallucination in large language models. arXiv preprint arXiv:2309.01219 (2023)