electronics MDPI

Article

Document GraphRAG: Knowledge Graph Enhanced Retrieval Augmented Generation for Document Question Answering Within the Manufacturing Domain

Simon Knollmeyer [1,*], **Oğuz Caymazer** [2] and **Daniel Grossmann** [1]

[1] AImotion Bavaria, Technische Hochschule Ingolstadt, 85049 Ingolstadt, Germany
[2] Department of Information Systems, University of Münster, 48149 Munster, Germany
* Correspondence: simon.knollmeyer@thi.de; Tel.: +49-841-9348-1118

Abstract: Retrieval-Augmented Generation (RAG) systems have shown significant potential for domain-specific Question Answering (QA) tasks, although persistent challenges in retrieval precision and context selection continue to hinder their effectiveness. This study introduces Document Graph RAG (GraphRAG), a novel framework that bolsters retrieval robustness and enhances answer generation by incorporating Knowledge Graphs (KGs) built upon a document's intrinsic structure into the RAG pipeline. Through the application of the Design Science Research methodology, we systematically design, implement, and evaluate GraphRAG, leveraging graph-based document structuring and a keyword-based semantic linking mechanism to improve retrieval quality. The evaluation, conducted on well-established datasets including SQuAD, HotpotQA, and a newly developed manufacturing dataset, demonstrates consistent performance gains over a naive RAG baseline across both retrieval and generation metrics. The results indicate that GraphRAG improves Context Relevance metrics, with task-dependent optimizations for chunk size, keyword density, and top-k retrieval further enhancing performance. Notably, multi-hop questions benefit most from GraphRAG's structured retrieval strategy, highlighting its advantages in complex reasoning tasks.

Keywords: Retrieval-Augmented Generation; large language model; knowledge graph; production domain; manufacturing; design science research

check for updates

Academic Editor: Mohammad Shahin

Received: 14 March 2025
Revised: 13 May 2025
Accepted: 15 May 2025
Published: 22 May 2025

Citation: Knollmeyer, S.; Caymazer, O.; Grossmann, D. Document GraphRAG: Knowledge Graph Enhanced Retrieval Augmented Generation for Document Question Answering Within the Manufacturing Domain. *Electronics* **2025**, *14*, 2102. https://doi.org/10.3390/electronics14112102

1. Introduction

The rapid advancements in Artificial Intelligence (AI), especially in the field of Large Language Models (LLMs), have transformed the way complex problems are solved across various industries. Although they have achieved remarkable progress, LLMs still struggle with significant challenges, particularly their limited ability to handle domain-specific and long-tail knowledge, which includes less common yet industry-critical information essential for specialized applications. Key limitations include their susceptibility to hallucinations, inability to self-update, and reliance on memorized patterns from pre-training datasets [1,2]. These challenges highlight the need for more robust frameworks to manage and retrieve knowledge effectively in specialized contexts.

In modern automotive manufacturing settings, for instance, production planning often involves reviewing extensive technical documentation, sometimes hundreds of pages long, describing assembly line configurations, supplier requirements, and operational guidelines. Planners and engineers must carefully verify that each technical requirement is met: a process that can be both error-prone and time-consuming. Projects can span multiple years and rely on a mix of structured data (e.g., engineering databases) and unstructured content (e.g., PDF documents), leading to fragmented knowledge repositories. When new team members or departments take over, the lack of institutional memory and the scattering of domain-specific jargon across different sources further complicate efficient knowledge reuse. In this context, ensuring rapid access to accurate information is paramount, motivating the development of systems that can intelligently retrieve and integrate data from diverse domains [3].

Retrieval-Augmented Generation (RAG) has emerged as a promising solution to extend LLMs' capabilities by integrating external knowledge sources into the generation process [4,5]. However, traditional RAG systems frequently encounter limitations when queries require multi-hop reasoning or when crucial context is dispersed across multiple documents and data formats. To address these gaps, recent research has explored enhancing RAG systems with Knowledge Graphs (KGs). By structuring data into entities and relationships, KGs enable more precise retrieval and reasoning, making them particularly valuable for managing long-tail knowledge [6,7].

While some early efforts in integrating KGs with RAG systems show promise, systematic guidance on how to design and implement such systems remains limited [8,9]. Specifically, industrial environments require solutions that integrate smoothly with existing workflows and infrastructure. Developing fully featured ontologies for KGs can be time consuming, and even small delays in retrieving critical insights can hinder fast-paced operational decision making. This emphasizes the practicality of Document Knowledge Graphs (DKGs), which offer a more targeted, lighter-weight approach to structuring domain knowledge without immediately requiring comprehensive ontological modeling [10]. Therefore, the presented solution leverages document structures combined with keyword-based linking to enhance RAG systems specifically within manufacturing environments.

To address the presented challenges, this study adopts a Design Science Research (DSR) methodology, which is particularly well suited for creating and evaluating innovative IT artifacts. DSR emphasizes iterative cycles of design, development, and evaluation to solve real-world problems and generate theoretical insights. By focusing on DKGs rather than fully expanded KGs, our approach aims to achieve a scalable solution that can be deployed and adapted within industrial settings with minimal reconfiguration. Following this methodology, this study is structured around three key research questions:

- How to design a KG-enhanced LLM RAG system?
- How to implement and instantiate a KG-enhanced LLM RAG system in an industrial setting?
- How does the proposed system compare to a naive RAG system used as a baseline in an industrial setting?

The DSR process involves reviewing potential strategies for integrating KGs with RAG systems, developing and implementing a design artifact, and application of an evaluation framework to assess its performance. This structured approach ensures actionable insights for both theory and practice. By addressing gaps in integration design and implementation, this study advances knowledge in AI-driven solutions for domain-specific challenges while offering practical guidance for industrial applications.

The remainder of this paper is structured as follows: Section 2 reviews related work on KG-enhanced RAG systems, establishing the foundation for the proposed approach. Section 3 outlines the research methodology employed in this study. Section 4 defines the design requirements that inform the system architecture, followed by Section 5, which details the proposed system's design. Section 6 describes the implementation and evaluation setup. Section 7 presents the evaluation results alongside a detailed analysis. Section 8 discusses the findings, addresses limitations, and provides contextual insights. Finally, Section 9 concludes the paper by summarizing key contributions and outlining directions for future research.

2. Related Work

2.1. Retrieval-Augmented Generation

RAG enhances LLMs by dynamically incorporating external knowledge into the generation process, addressing key limitations such as hallucinations and the inability to handle domain-specific, long-tail knowledge [4,11]. Unlike traditional LLMs that rely solely on parametric memory, RAG introduces a retrieval module that retrieves relevant documents from an external knowledge base, grounding the model's responses in factual information. However, while naive RAG systems improve response accuracy, they often suffer from retrieval inefficiencies, semantic mismatches, and context fragmentation. To mitigate these issues, advanced retrieval and ranking techniques refine various stages of the pipeline, improving overall system performance. The structure of a naive RAG system is shown in Figure 1.

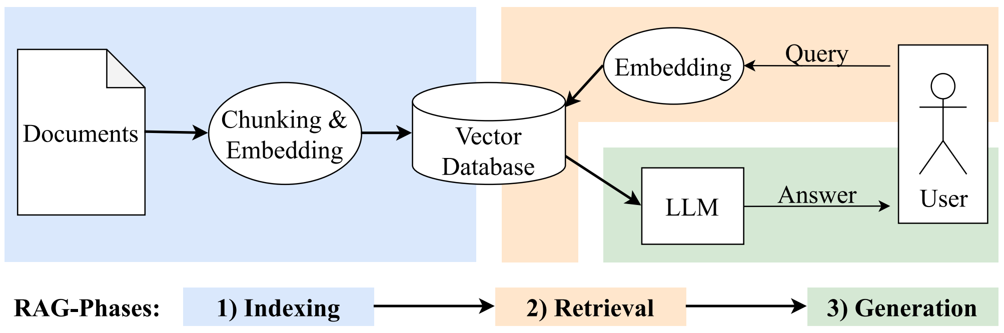

Figure 1. Architecture of a naive RAG system.

A naive RAG system consists of three key stages: indexing, retrieval, and generation. In the indexing phase, raw text data are pre-processed, transformed into chunks, and embedded into a high-dimensional vector space using transformer-based models [9,12]. Data preprocessing is crucial, as it involves cleaning, handling different file formats such as PDFs and Markdown, and converting them into plain text [9]. The chunking process, which breaks text into smaller, semantically coherent segments, significantly impacts retrieval effectiveness. The choice of chunking strategy is critical: larger chunks preserve more context but may introduce irrelevant information, whereas smaller chunks improve granularity at the cost of losing crucial context [12,13]. One common approach is adaptive chunking, which aligns splits with sentence or paragraph boundaries rather than arbitrary character limits [9,12]. Following chunking, embeddings are generated, capturing semantic meaning in a high-dimensional vector space [4,14]. The choice of embedding model significantly affects retrieval performance, with dense embeddings offering better semantic granularity at the cost of higher computational complexity, while sparse embeddings trade retrieval precision for efficiency [4,14–16].

The retrieval phase translates user queries into vector embeddings and retrieves the most semantically relevant chunks using similarity measures such as cosine similarity [17]. Ensuring consistency in vector representation between indexed chunks and queries is crucial to retrieval accuracy [9,16]. The system retrieves the top-k chunks based on similarity scores, where k serves as a tuneable parameter that balances recall and precision [9,12]. However, naive retrieval often suffers from semantic mismatches, where lexically similar but contextually irrelevant chunks are retrieved [18–20]. To mitigate this, advanced retrieval techniques refine chunk selection and ranking. Query rewriting dynamically reformulates user queries to improve alignment with indexed content, reducing ambiguity and increasing retrieval precision [21]. Additionally, reranking methods reorder retrieved chunks by incorporating additional semantic and contextual relevance factors, ensuring that the most useful information is prioritized for generation [9,22]. More sophisticated hybrid retrieval models combine dense and sparse vector retrieval, balancing efficiency and accuracy by leveraging complementary strengths [4,16].

The generation phase integrates retrieved chunks into the LLMs prompt, allowing it to generate responses grounded in external knowledge while mitigating hallucinations [4,19]. A well-structured prompt ensures the model effectively leverages retrieved information by prioritizing contextual relevance and coherence [11,19,23]. However, excessive or redundant chunks can lead to prompt overload, where irrelevant information dilutes response quality. Advanced context filtering techniques, such as passage scoring and chunk compression, optimize chunk selection to enhance response coherence [23,24]. Additionally, long-context models extend the window size for incorporating larger retrieved chunks while maintaining coherence [25].

Despite these improvements, RAG systems still face challenges, particularly in multi-hop reasoning and semantic cohesion. Chunking disrupts document continuity, making it difficult for the model to synthesize information across multiple sources, particularly when complex reasoning is required [25–27]. Embedding-based retrieval can lose fine-grained contextual links, reducing the system's ability to trace and integrate knowledge across documents [26,28]. These limitations are especially problematic for knowledge-intensive tasks requiring multiple pieces of evidence to be combined into a coherent answer. Addressing these challenges requires advancements in hierarchical retrieval augmentation, better long-range coherence mechanisms, and multi-document fusion strategies to enhance knowledge synthesis and ensure reliable, high-quality responses [25,27].

2.2. Knowledge Graphs for Retrieval-Augmented Generation

KGs represent structured relationships between entities, capturing semantic information through interconnected nodes and edges [6]. They originated from the Semantic Web vision, aiming to make information machine-processable by addressing issues of data heterogeneity and enabling structured knowledge representation [29]. Each node in a KG represents an entity, such as a person, organization, or concept, while edges define the relationships between these entities. The structured nature of KGs facilitates advanced reasoning, making them valuable for various applications such as search, recommendation, and Question Answering (QA) [30]. Unlike unstructured text or isolated vector embeddings, KGs explicitly encode semantic relationships, providing a robust framework for knowledge-intensive applications.

In recent years, research has explored the integration of KGs into RAG systems to enhance retrieval precision and facilitate multi-hop reasoning. While traditional RAG systems rely solely on vector-based similarity retrieval, KG-enhanced RAG systems leverage structured knowledge to improve semantic understanding and refine document retrieval. By explicitly encoding relationships between entities, KGs mitigate the fragmentation introduced by naive chunking in RAG, allowing models to retrieve contextually relevant information with higher accuracy [8,31]. Instead of retrieving passages based purely on lexical or embedding similarity, KG-enhanced RAG enables retrieval mechanisms that factor in explicit relationships, improving interpretability and accuracy in question-answering tasks. Beyond hybrid retrieval, recent research increasingly positions the KG as a reasoning framework or semantic routing layer. Emerging patterns include [8,32,33]:

- KG-as-index: entities serve as primary retrieval keys, later expanded with dense contextual information;

- KG-guided chunking: structural signals within the KG inform how documents are segmented prior to embedding;
- KG-constrained decoding: triples are used as hard or soft constraints to guide the generation process; and
- KG-as-memory: newly generated triples are written back into the KG, supporting continual knowledge refinement.

Within the pattern KGs-as-index for RAG, the structure of the graph can be broadly categorized into Information Knowledge Graphs (IKGs) and DKGs, each serving distinct purposes [10]. IKGs focus on representing extensive domain knowledge using detailed ontologies, supporting deep reasoning and inference capabilities [34,35]. These graphs model complex interdependencies between entities, making them well suited for applications requiring semantic search and intelligent decision making [36]. However, constructing IKGs is often labor intensive, requiring extensive domain expertise and manual effort to define detailed ontologies [37]. As a result, while IKGs offer powerful reasoning capabilities, their complexity makes them challenging to integrate into practical RAG systems within special domains.

In contrast, DKGs organize knowledge at the document level, structuring relationships between document elements rather than broad domain-wide entities. This approach is particularly relevant for RAG systems, as DKGs optimize document retrieval by encoding metadata, inter-document relationships, and contextual linkages, improving search efficiency and retrieval accuracy [10]. Unlike IKGs, DKGs require less ontology development, making them a more scalable and practical solution for RAG integration.

Figure 2 illustrates the differences between an IKG and a DKG. The IKG represents domain-wide entities and relationships, enabling semantic reasoning, while the DKG structures document-level relationships to enhance retrieval in RAG systems.

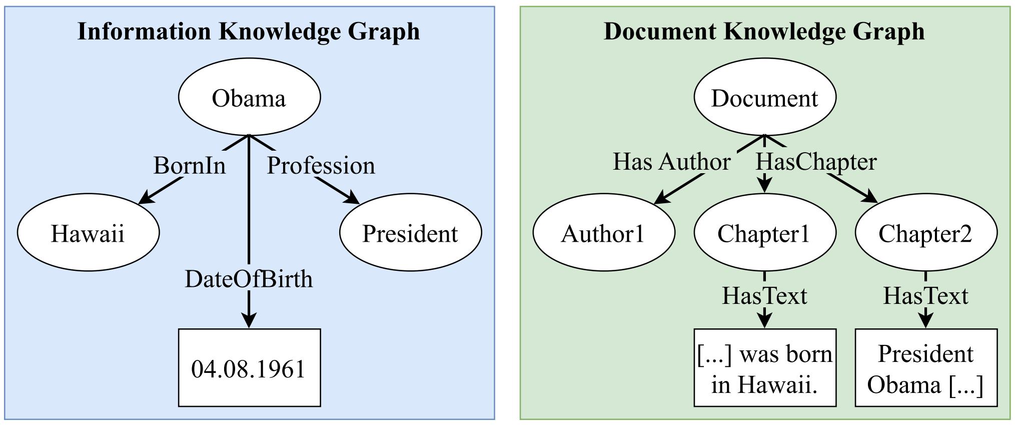

Figure 2. Comparison of Information Knowledge Graph and Document Knowledge Graph adapted from [10].

In specialized domains such as manufacturing, source materials span standards, patents, and maintenance logs across multiple languages and dialects. Constructing a unified Knowledge Graph (KG) in this context inherits the core challenges of Multilingual Large Language Models (MLLMs), including data sparsity, script diversity, and cross-lingual entity alignment. A recent survey on MLLMs [38] identifies two challenges that directly translate to KG construction:

- Data quality and diversity: high-resource languages dominate training corpora, leaving low-resource manufacturing languages underrepresented;

- Cross-lingual knowledge transfer: factual consistency declines when identical queries are issued in different languages, indicating that multilingual entity linking must explicitly propagate facts, often originating in English, across languages.

Existing KG-enhanced RAG approaches differ considerably in their graph construction strategies and retrieval mechanisms. The method proposed in [32] emphasizes LLM-based extraction of entities and relations to build an IKG. However, retrieval in this case is limited to the IKG layer and lacks integration with vector-based methods. A more hybrid strategy is seen in [39], where IKG-based retrieval is extended by incorporating parallel vector search, improving flexibility in capturing semantic relevance. Similarly, ref. [33] enhances IKG construction through the use of document structure for better entity extraction and linking but does not leverage structural information during retrieval.

In contrast, ref. [40] employs a DKG by connecting documents via existing hyperlinks or citations, yet it omits the use of an IKG, potentially limiting semantic granularity in intra-document relations. The work in [41] combines both DKG and IKG elements but is constrained by its reliance on highly structured PDF documents with a table of contents, limiting applicability in real-world scenarios where unstructured or semi-structured formats dominate. Furthermore, the use of the KG during retrieval lacks detailed description.

Some KG-enhanced RAG systems target domain-specific applications. For example, ref. [42] addresses customer service, while [43] focuses on Failure Mode and Effects Analysis (FMEA), a relevant use case in manufacturing. The latter leverages semi-structured input to extract domain-specific relations, such as cause–effectlinks, enriching the IKG with task-specific semantics. Lastly, ref. [44] shifts focus from graph construction to graph traversal, exploring various path-finding strategies for improved retrieval. While methodologically valuable, this work assumes a pre-constructed graph and does not address the challenges of graph formation in complex domains.

Despite recent progress in KG-enhanced RAG systems, existing research lacks a systematic approach to designing knowledge graphs specifically tailored for document-based QA. Most prior work emphasizes entity-centric IKGs, which are effective in structured domains but are not well suited for retrieval tasks involving complex, unstructured documents. Moreover, constructing high-quality IKGs in specialized domains remains largely manual and time consuming, as current automated approaches still struggle with precision [8,45]. To address these limitations, this work introduces a DKG-driven retrieval framework complemented by a lightweight, automatically constructed IKG component. The proposed solution is designed to enhance document-level question answering by combining structured document representation with retrieval-augmented generation.

Complementary to graph-centric methods, other approaches focus on integrating diverse knowledge sources to dynamically adapt the retrieval pipeline. For instance, the recently proposed *Collaborative Legal Expert* (CoLE) framework decomposes legal reasoning into four prompt-based stages: Query-Intent Identification, Domain-Knowledge Encoding, Best-Demonstration Retrieval, and Prompt Generation. It demonstrates substantial improvements in reasoning performance without requiring additional model fine-tuning [46]. CoLE's modular design is orthogonal to, and potentially compatible with, graph-based RAG frameworks.

2.3. Integration in Industrial Context

The manufacturing industry is undergoing significant transformation due to the increasing complexity of production processes and the growing volume of unstructured information that must be efficiently managed. A key challenge in this domain is the fragmentation of knowledge across multiple systems and documents, often stored in heterogeneous formats such as PDFs [3,30,47]. This lack of centralized and structured access to information hinders operational efficiency, decision making, and the reuse of critical knowledge [36,48]. To address these issues, manufacturers are focusing on improving the accessibility and retrieval of domain-specific information, particularly in factory planning and production operations [10]. Planners and engineers frequently engage in QA tasks, where retrieving precise information from scattered technical documents is essential [3,30]. Similarly, comparing existing knowledge on specific topics requires consolidating information from multiple sources while ensuring traceability and accuracy [30,49]. These challenges emphasize the need for advanced IT solutions capable of efficiently handling vast amounts of unstructured data and enhancing knowledge retrieval processes.

The integration of RAG and KGs offers a strategic opportunity to enhance knowledge utilization in manufacturing environments. RAG systems improve information retrieval by dynamically extracting relevant content from large-scale repositories. For instance, a manufacturing-specific RAG architecture designed for Manufacturing Execution System environments has been introduced, enabling real-time data processing [50]. While this approach effectively leverages structured data, the integration of domain-specific document knowledge into RAG systems remains an open challenge.

Despite their advantages, RAG systems are inherently constrained by retrieval quality, often yielding incomplete or noisy results, which in turn reduces the accuracy of generated responses [18,51,52]. KGs, in contrast, offer a structured representation of knowledge, organizing entities and relationships to enable precise, multi-hop reasoning over complex information [6,30].

For retrieving information from internal technical documents, RAG systems can index and retrieve data efficiently but struggle with semantic integrity and cross-document reasoning [10,23,28]. KGs enhance retrieval by structuring document relationships and supporting logical inference across multiple sources. This structured approach mitigates inconsistencies in retrieval but incurs higher computational costs [30]. Ensuring grounded responses from relevant sources requires adapting RAG systems to improve retrieval accuracy and document alignment [11,23,53].

From an operational perspective, both technologies present integration challenges. RAG systems need refinements beyond naive implementations to improve retrieval precision [22,52], whereas KGs require automation in knowledge extraction and graph construction to be effectively utilized in production environments [30,47]. Addressing these challenges is crucial for achieving a cost-efficient and scalable knowledge retrieval solution in manufacturing.

3. Research Methodology

This study follows a DSR methodology to develop and evaluate a KG-enhanced RAG system within an industrial manufacturing setting. Given the complexity of LLM, RAG, and KGs—technologies with relatively low solution maturity—the structured, iterative nature of DSR provides a systematic framework to address both theoretical and practical challenges.

The chosen research approach follows an echelon-based DSR methodology [54], which decomposes the research process into logically structured and self-contained phases. Each phase—Problem Identification, Design Requirements, Solution Design, Demonstration, and Evaluation—contributes to the incremental development and validation of the artifact. This stepwise approach ensures theoretical rigor and practical applicability by incorporating targeted validation at each stage. Through iterative refinements and empirical validation, this study aims to derive actionable design knowledge that enhances the integration of KG-enhanced RAG systems in industrial contexts.

3.1. Problem Identification

The first phase of DSR establishes the core problem by aligning theoretical foundations with the industrial context [55,56]. In this work, the identified and motivated problem can be stated as follows:

> *In manufacturing environments, the fragmentation of domain-specific knowledge across various systems and documents impedes efficient decision making and hinders the reuse of critical information.*

Existing RAG systems struggle with domain-specific and long-tail knowledge, limiting their effectiveness. KGs enhance retrieval and reasoning in RAG but pose significant challenges when integrating them into complex industrial environments. This problem statement underscores the inadequacy of current solutions and highlights the necessity of a KG-enhanced RAG system to overcome these limitations.

This phase ensures theoretical grounding and practical relevance by systematically defining the problem [54,57,58]. The validated problem statement guides the design, development, and evaluation of an artifact tailored to both academic and industrial needs.

3.2. Design Requirements

Building on the validated problem statement, this DSR phase translates the identified challenges into concrete design objectives for a KG-enhanced RAG system. The goal is to establish actionable requirements that ensure alignment between the artifact and the industrial challenges [54,57,58].

To derive these objectives, a literature review was conducted, systematically exploring the intersection of LLMs, RAG, and KGs. This review identified feasible integration strategies and technological synergies to construct an effective KG-enhanced RAG artifact. By synthesizing the SLR findings, key design principles were extracted, directly addressing the challenges outlined in the problem statement.

The resulting design objectives (c.f. Section 4) were further validated through a focus group consisting of experts from the automotive industry. Their insights ensured the technical feasibility and industrial applicability of the proposed solution. This structured approach guarantees that the artifact is grounded in both theoretical insights and practical constraints, forming a solid foundation for the subsequent design and development phases.

3.3. Solution Design

Based on the validated design objectives, the research progressed into the design and development phase, focusing on the creation of an architectural model for the KG-enhanced RAG artifact. This model serves as a structured blueprint, defining the functional and structural components necessary to achieve the specified design objectives [54]. Each component was deliberately designed to address the identified challenges, ensuring a cohesive and purpose-driven solution.

The architectural model (c.f. Section 5) was developed iteratively, incorporating insights, logical reasoning, and the literature to refine interactions between LLM, RAG, and KG components. A key aspect of this process was enhancing the retrieval algorithm of the naive RAG system by leveraging structured knowledge from KGs. Additionally, industrial utility was improved by incorporating mechanisms such as chain-of-reasoning at the LLM output, ensuring greater transparency and reliability in decision making.

To ensure alignment with practical requirements, the model was validated by the same focus group involved in the previous phase. Special attention was given to scalability and flexibility, enabling seamless integration into existing systems in industrial settings while maintaining adaptability to future technological advancements. This structured design approach ensures that the developed artifact remains both theoretically robust and practically viable for industrial deployment.

3.4. Demonstration

The demonstration phase involved the practical implementation of the validated architectural model to assess its effectiveness in an industrial context [54,57]. Following a case study approach, the artifact was applied to a real-world problem at a production planning department in the automotive industry, providing concrete evidence of its utility.

To ensure a controlled yet realistic validation, the artifact was instantiated in a simulated environment that closely mirrors real-world conditions within the industry (c.f. Section 6). Each iteration of the case study systematically assessed the artifact's performance across various evaluation dimensions, focusing on improving retrieval quality. Particular attention was given to enhancing the performance of the artifact compared to the naive RAG system, which served as the baseline.

Through iterative refinements, the demonstration phase validated the artifact's technical feasibility and its potential impact in an industrial setting. The insights gained from this process informed the subsequent evaluation phase, ensuring a rigorous assessment of the artifact's real-world applicability and effectiveness.

3.5. Evaluation

The final phase of the DSR process involves the rigorous evaluation of the instantiated artifact to assess its effectiveness in addressing the identified problem. This step ensures that the artifact is both theoretically sound and practically applicable in an industrial setting [57,58].

The evaluation focuses on effectiveness as the primary validation criterion, measuring the artifact's success in solving the intended problem and outperforming the naive RAG baseline. A formative evaluation approach was applied, iteratively testing different instantiations of the artifact (c.f. Section 6). This method enables empirical feedback to refine the solution design, ensuring it meets the defined design objectives [54,55].

A comparative analysis was conducted, systematically assessing the artifact against the naive RAG system using the identified evaluation dimensions. The outcome of this phase is a validated Proof-of-Value (PoV), demonstrating the practical benefits of the KG-enhanced RAG system. These findings provide a solid foundation for broader implementation within industrial settings.

4. Design Requirements

Design requirements represent the second phase of the DSR approach, building on the problem identified in Phase 1 (c.f. Section 3.1). This chapter presents the results of this phase, addressing the research questions. Following the DSR methodology, the chapter outlines the key design requirements derived from the literature, which highlight the limitations of current RAG systems and define the essential capabilities needed to overcome these challenges. These requirements define the essential capabilities an artifact must possess to overcome existing challenges.

To ensure both theoretical grounding and industrial relevance, the identified design objectives underwent validation through a focus group consisting of industry experts from an automotive factory planning division. These experts assessed the objectives based on their feasibility, relevance, and alignment with real-world manufacturing challenges. Their feedback refined the requirements, ensuring coherence with theoretical foundations while maintaining practical applicability.

The following sub-chapters present the validated design requirements D1–D4 in detail, outlining the specific challenges and considerations that guided the development of the KG-enhanced RAG system.

4.1. Retrieval Accuracy (D1)

To enhance retrieval accuracy in domain-specific RAG systems, researchers propose leveraging KGs to capture semantic and structural relationships within documents [59]. By integrating these relationships, relevant information can be more effectively identified and prioritized, addressing a key limitation of existing RAG approaches [7,10,60,61].

Semantic relationships can be established by linking keyword-related connections between chunks across different documents, enabling more contextually relevant retrieval. Additionally, structural relationships between chunks, even within the same document, are often lost in current RAG implementations. Restoring these connections allows for supplementary retrieval from related passages, improving contextual integrity [7,10,60].

Thus, the objective is to exploit both structural and semantic relationships in and between documents, enabling advanced search strategies within a KG. This extension of RAG is expected to significantly enhance retrieval accuracy [7,10,60,61].

4.2. Multi-Hop Questions-Answering (D2)

Existing RAG systems struggle to provide complete and accurate responses in multi-hop QA scenarios, where answers are distributed across multiple documents and sections [23,26,28]. To address this limitation, the design objective focuses on improving context integration across multiple sources, enabling more advanced reasoning [7,10,62].

A key requirement for enhancing multi-hop QA is enabling multiple retrievals from the KG, where each retrieval builds upon prior results in an informed manner [10,60]. Additionally, the KG must be structured so that nodes are meaningfully connected, ensuring that edges reflect semantic proximity and shared concepts [7].

By improving multi-hop QA capabilities, the RAG system can generate more comprehensive and accurate responses to complex queries, enhancing its effectiveness for advanced, reasoning-intensive applications [26].

4.3. Error Mitigation (D3)

Preventing retrieval errors is crucial in RAG systems, as incorrect information can propagate through the pipeline, leading to misleading outputs. This issue is particularly critical in domain-specific applications, where answer reliability is essential. Implementing mechanisms to detect and correct errors at intermediate stages enhances both accuracy and trustworthiness [11,22,24].

To mitigate retrieval errors, various strategies can be employed, such as intermediate validation steps, self-critique mechanisms, or filtering approaches [60,61]. By integrating such techniques, the system can reduce noise in retrieval, preventing the LLM from generating unreliable responses. This ensures a more robust and dependable RAG system, particularly for industrial use cases requiring high information precision.

4.4. Document Knowledge Management (D4)

In RAG systems, the document structure and contextual relationships between chunks are lost after embedding, making it difficult to maintain coherence during retrieval [7,10]. However, preserving document structure is essential for intelligent retrieval, as it allows the system to leverage metadata such as publication year, author, and hierarchical organization, including chapters and sections [10,61].

Representing document structures as nodes in a KG enables retrieval based on structural elements, allowing targeted searches within specific chapters or sections. Additionally, metadata can refine search precision, provided they are properly stored and utilized during retrieval [7,61].

By integrating document structures and metadata, the system can prioritize and filter chunks more effectively, improving retrieval relevance and enabling more sophisticated queries. To meet this requirement, the system must incorporate robust document knowledge management capabilities [7,10].

5. Solution Design

Building on the design objectives established earlier, this section introduces the architectural model for integrating KGs into RAG systems. While the design objectives define the goals for a successful integration, the architecture provides a structured blueprint to achieve these goals, detailing the necessary structural and functional components [54].

An overview of the developed architectural model is shown in Figure 3. The solution follows the three core RAG phases: indexing, retrieval, and generation.

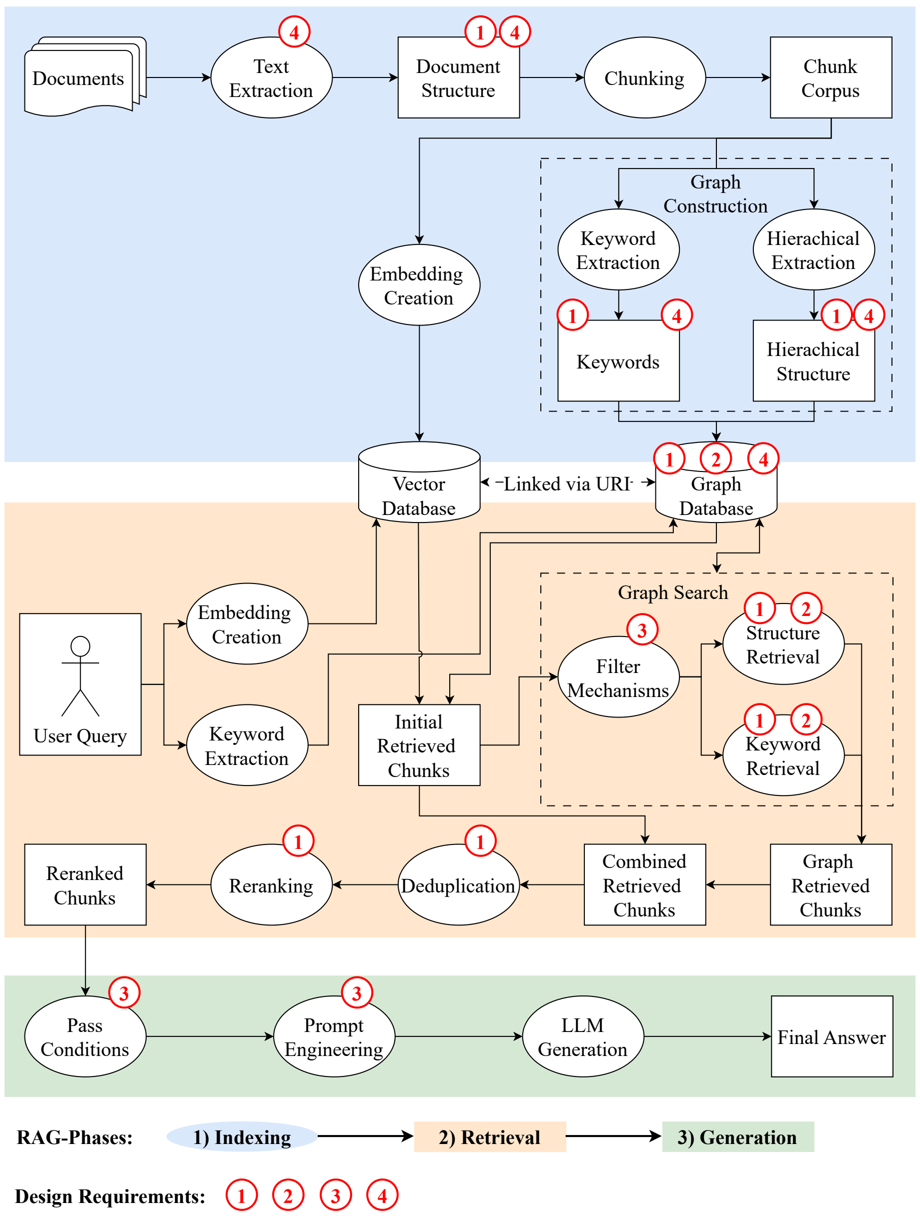

Figure 3. Architectural model with references to design requirements.

The process begins with document processing, preparing documents for indexing within both the vector and graph databases. Once indexed, the RAG system can handle queries for QA tasks. The retrieval phase selects the most relevant information from both databases, ensuring an optimized input for the final generation phase, which synthesizes the answer. The following sections provide a detailed explanation of each phase.

The architectural model also references relevant key components to the described design objectives (D1–D4). To ensure the architectural model meets practical and technical requirements in automotive production, a two-stage validation was conducted. First, three focus group sessions with domain experts evaluated the design decisions, leading to refinements in key aspects such as search strategies within the DKG. Second, simulations tested the model's feasibility and consistency through iterative instantiations in real-world scenarios, identifying and resolving issues related to graph construction. These validation steps ensured that the architecture effectively supports QA tasks in factory planning while aligning with the defined design objectives.

5.1. Creation of the Knowledge Graph

The indexing process begins with documents as the primary data source, including technical manuals, tenders, and reports. These documents often contain complex, unstructured information, making retrieval challenging. To address this, the text extraction component identifies and preserves structural elements such as chapters, sections, and tables, ensuring contextual integrity (D4). Efficiency and scalability are critical considerations when selecting the extractor, as a slow process can become a system bottleneck. The extracted output includes the document's hierarchical structure and metadata, such as the author and publication year, with the option to store additional production-related information.

The chunking process extends a standard RAG system by producing a structured chunk corpus that retains the hierarchical organization of the input document. It builds on the document structure identified during text extraction, ensuring that each chunk remains semantically coherent and aligned with logical section boundaries. By avoiding chunk boundaries that span across different subchapters, the retrieval performance is optimized, as the inclusion of disjointed content within a single chunk can negatively impact semantic understanding [63]. Each chunk is assigned a Unique Resource Identifier (URI), enabling it to be traced back to its original position within the document. This guarantees that, while the text is segmented for retrieval and embedding, its structural and contextual relationships remain intact, supporting intelligent and context-aware querying (D1, D4).

As illustrated in Figure 3, the chunk corpus serves two primary purposes. First, embeddings are generated from the chunks and stored in a vector database, following the standard procedure of a naive RAG system. Second, a KG is constructed from the same corpus (D1, D2, D4), leveraging the extracted structural information without the need for manual annotation or human refinement during graph creation. The structure of the indexed graph is illustrated in Figure 4.

The upper section of Figure 4 illustrates the hierarchically structured DKG derived from the generated chunk corpus. As shown, metadata are linked to each document node to support enhanced retrieval. The document structure is modeled as a tree-like graph, hierarchically organized into chapters and nested sub-chapters (lower-level nodes are omitted from the illustration for clarity).The DKG terminates at the chunk level, where each chunk is assigned a unique URI, which is also stored in the vector database to maintain a consistent link between both databases. Since this process involves transforming the structured chunk corpus into a graph representation using predefined rules and algorithms, the construction of the DKG is computationally efficient and does not require additional models or training.

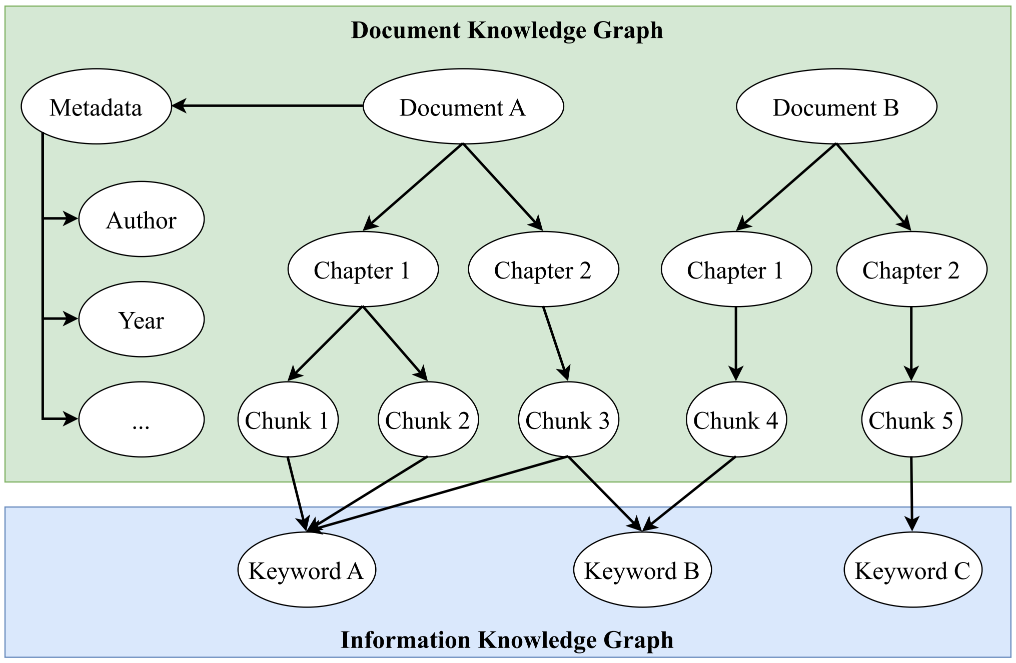

Figure 4. DKG and IKG indexing structure.

To further optimize retrieval, an IKG layer is created during the indexing phase. In this step, relevant keywords are extracted at the chunk level and stored within the graph structure. This enables deeper semantic insights into the indexed documents, as chunks from different chapters or documents can be interlinked via shared keywords. Since this layer directly influences retrieval performance, the number of extracted keywords must be carefully selected to prevent unnecessarily long retrieval times. Keyword extraction can be performed using algorithmic methods that offer fast processing without requiring external models or training. Alternatively, LLMs can be applied to extract domain-specific and contextually nuanced keywords, supported by prompt engineering techniques.

5.2. Retrieval Algorithms

This subsection addresses the retrieval phase of the RAG pipeline, highlighted in orange in Figure 3. The process begins with a user query and returns a ranked list of relevant chunks to support response generation. By leveraging the URI-based linking between the vector and graph databases, multiple retrieval strategies can be implemented. This work introduces three distinct graph-based retrieval approaches, each of which is described in the following sections, along with the reranking method employed to optimize retrieval performance (D1, D3).

The corresponding pipelines for the three graph-based retrieval strategies are illustrated in Figure 5, with the retrieval operations based on the underlying graph structure shown in Figure 4.

The first retrieval strategy, *Informed Chapter Search* (ICS), is shown in blue within Figure 5. It leverages both the vector and graph databases for optimized retrieval, focusing solely on the DKG component of the KG. ICS begins with a standard RAG retrieval, where the query is embedded, and the top-k most similar chunks are retrieved from the vector database. These initial chunks serve as the starting point for ICS. Subsequently, for each retrieved chunk, all other chunks within the same chapter are also retrieved using the DKG. This approach is based on the premise that relevant content is often concentrated within the same chapter, particularly in technical documents, where chapters provide a structured organization of information.

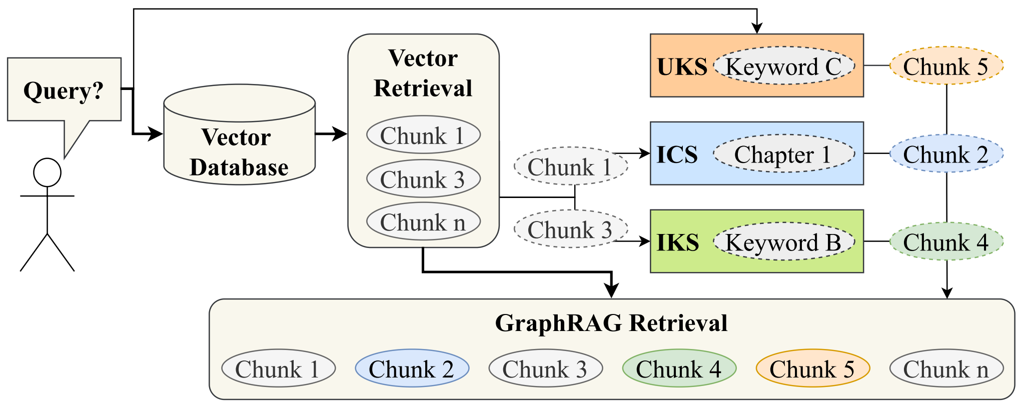

Figure 5. Pipeline of GraphRAG retrieval algorithms.

ICS enhances retrieval by filtering the DKG based on the initial vector search results, ensuring structural retrieval (D1, D2) while improving efficiency and relevance (D3). In the DKG representation within Figure 4, if *Chunk 1* appears in the top-k initial results, ICS would automatically retrieve *Chunk 2* as part of the same *Chapter 1*, as shown in Figure 5, demonstrating its ability to expand retrieval while maintaining structural coherence.

The second retrieval strategy, *Informed Keyword Search* (IKS), builds upon the initial vector search while leveraging both the DKG and the IKG layer. It is shown in green within Figure 5. Similar to ICS, IKS begins by retrieving the top-k most relevant chunks from the vector database. For each retrieved chunk, all associated keywords are extracted from the KG, and subsequently, all chunks linked to these keywords are included in the graph retrieval set.

IKS enhances naive RAG retrieval by incorporating domain-specific keywords to refine search results. In manufacturing contexts, texts often contain specialized terminology, which can be effectively utilized to improve retrieval accuracy. As illustrated in Figure 3, IKS applies a filtering mechanism (D3) within the graph search, followed by keyword-based retrieval (D1, D2). For example, in the KG representation in Figure 4, if *Chunk 3* appears in the top-k initial results, IKS would also retrieve *Chunk 4*, as both share *Keyword B*, as shown within Figure 5. This approach enables inter-document connections within the KG, enhancing retrieval by linking related content across multiple documents.

The third retrieval strategy, *Uninformed Keyword Search* (UKS), introduces an additional retrieval pathway that operates in parallel with vector-based retrieval, expanding the initial set of retrieved chunks. It is shown in orange within Figure 5. In this approach, keywords are extracted directly from the user query and used to retrieve all associated chunks from the KG. These retrieved chunks are then incorporated into the initial results for subsequent ICS and IKS retrieval steps. Similar to IKS, UKS leverages domain-specific terminology to enhance retrieval performance. For example, with reference to the graph in Figure 4, if *Keyword C* is extracted from the query, *Chunk 5* would be included in the initial retrieval set, as displayed in Figure 5.

The proposed Document GraphRAG (GraphRAG) framework integrates the three retrieval strategies introduced earlier. After combining the chunks from both initial vector-based retrieval and graph-based retrieval, two post-processing steps are required before passing the results to the generation phase. First, deduplication is performed to ensure that each chunk appears only once, as multiple retrieval strategies may return overlapping results. Next, reranking is applied to prioritize the most relevant chunks for answering the user query. This step is essential, as the GraphRAG approach can significantly expand the number of retrieved chunks, necessitating an effective ranking mechanism.

These post-processing steps enhance retrieval accuracy and efficiency (D1, D3), resulting in a reranked list of chunks that serves as the input for the generation phase.

5.3. Answer Generation

The final step of the solution design focuses on generating accurate and contextually grounded answers to user queries (D3). Before leveraging an LLM for answer generation, two preparatory steps are required: applying pass conditions to select relevant chunks and constructing an effective prompt.

First, pass conditions determine how many top-k reranked chunks are forwarded to the LLM, ensuring that it processes only the most relevant information while preventing overload, which could otherwise reduce answer accuracy.

Once the relevant chunks have been retrieved and selected, prompt engineering is applied to structure the input effectively, ensuring that the LLM generates coherent, contextually relevant, and factually grounded responses. To address the specific requirements of the manufacturing domain, the prompt was refined in collaboration with domain experts, with the final version provided in Appendix A.

This approach ensures that the LLM produces responses that remain faithful to the retrieved content, incorporating direct citations from the selected chunks (D3). Accordingly, the prompt is designed to include the user query, the retrieved chunks, and explicit instructions, such as enforcing source citations and requiring the model to indicate when sufficient information is unavailable.

With these pass conditions and structured prompt, the LLM generates the final response, ensuring that the output remains contextually accurate, relevant, and traceable to the retrieved sources.

6. Demonstration and Evaluation Setup

This section corresponds to the fourth and fifth phases of the DSR approach. To assess its effectiveness in an industrial context, the proposed architectural model is implemented and evaluated within a structured evaluation framework, ensuring its practical applicability in real-world scenarios. The chapter begins with a detailed technical implementation of the proposed architecture, followed by a description of the datasets used for evaluation and the evaluation strategy employed. Finally, the chapter concludes with an overview of the evaluation metrics used to assess the system's performance.

6.1. Technical Implementation

This chapter provides an overview of the technology stack used to implement the proposed architectural model. The key components are described along the processing pipeline shown in Figure 3. The implementation is designed as an initial demonstration of the architecture, with a primary focus on evaluating the quality of the generated answers rather than optimizing system performance in terms of latency or computational efficiency. To support modular development and ensure transparency during evaluation, JavaScript Object Notation (JSON) files are used for data transfer between pipeline components. For example, intermediate outputs are stored as files after the retrieval, reranking, and generation stages.

The overall implementation was developed using *Python Version 3.10* and executed on a standard office notebook without the use of specialized hardware. For the integration of AI components, including text extraction, LLMs, and embeddings, *Amazon Web Services* (AWS) cloud services were utilized. These components were accessed via Application Programming Interface (API) calls from the local implementation, enabling the use of scalable cloud-based models within the otherwise locally run pipeline.

In the indexing phase, *AWS Textract* [64] was utilized to extract text in markdown format while preserving the structural layout of the original documents. Although AWS Textract is sufficient for this proof of concept, the pipeline remains extractor-agnostic: any engine capable of outputting text, token positions, and reading order in JSON format can be integrated. More advanced layout-aware multimodal Transformers, such as LayoutLMv3 [65] or the earlier LAMBERT architecture [66], offer the potential to more tightly integrate visual and textual cues. These models could provide richer structural signals, which in turn could inform more expressive edge construction within the downstream DKG. This OCR-based service offers high flexibility for processing the diverse data formats typically found in the manufacturing domain. The extracted content, provided on a page-wise basis, is post-processed to retain only high-confidence elements and subsequently merged into complete documents. Converting the output to markdown enables effective capture of the hierarchical structure, comprising headings and corresponding numbering, thus facilitating the identification of chapters and sections. To ensure the robustness of the extracted hierarchy, several back-off algorithms are applied to validate and refine the structural integrity based on the detected heading levels and numbering schemes. This guarantees a consistent and semantically useful input for the subsequent processing steps. The evaluation of such layout-aware extractors and the analysis of how geometry-aware structural links influence retrieval accuracy are left for future work (see Section 9).

The chunking process was carried out using *LangChain's Character Text Splitter* [67]. The chunk size was determined based on the evaluation setup, as described in the following section. No overlap between chunks was applied, and splitting was configured to occur at paragraph or sentence boundaries to preserve semantic coherence.

Embeddings were generated using *AWS Titan* [68] and stored in a locally hosted *Chroma* vector database [69], which was configured to compute vector similarity using cosine distance. In parallel, the KG was constructed according to the *RDF* standard [70] and stored in a locally running *GraphDB* instance [71], following the structural model illustrated in Figure 4.

Keyword extraction was conducted using the *YAKE* algorithm [72] to enhance retrieval performance within the RAG system. The process was executed locally, with the language parameter set according to the respective dataset. YAKE was configured to extract unigrams and bigrams, applying a deduplication threshold of 0.8 to avoid nearly identical keywords. An algorithmic approach was chosen to ensure a standardized and reproducible evaluation. The extraction parameters were fine-tuned in collaboration with the focus group to optimize domain relevance and retrieval quality.

Section 5.1 discusses LLM-based, domain-specific keyword extraction as a potential alternative, offering highly specific and context-aware keywords at the cost of greater computational resources and integration effort. To explore this option, initial tests were conducted using LLM-extracted keywords, based on the prompt provided in Appendix B. *Anthropic's Claude 3 Haiku* [73] was employed via *AWS Bedrock* [74], selected for its lower computational demands, faster response times, and cost efficiency. Its capabilities make it well suited for simpler tasks like keyword extraction, which require less reasoning than answer generation.

During the retrieval phase, embeddings and keywords are generated in alignment with the indexing phase to ensure consistency. The graph search algorithms utilize *SPARQL Protocol and RDF Query Language* (SPARQL) queries to retrieve relevant information from the *RDF* graph.

In the deduplication module, URIs are used to ensure that each chunk appears only once in the retrieval results. Reranking was implemented using an LLM, which was prompted to assign a relevance score between one and ten to each chunk based on its relevance to the user query. To enhance interpretability, the LLM was also instructed to provide a justification for each score, enabling the system to leverage the model's reasoning capabilities. The prompt used for reranking is provided in Appendix C.

For LLM-based reranking, again *Anthropic's Claude 3 Haiku* was used due to the complexity of the task. The implementation includes a verification step that checks the LLM's output, and if results are missing or unprocessable, the reranking task is automatically repeated to ensure a complete and valid score is assigned to each chunk.

During the generation phase, *Claude Sonnet 3.5* [75] was selected for answer generation, as this task requires advanced reasoning capabilities and contextual synthesis. The corresponding prompt template, as provided in Appendix A, was used to guide the model. To support structured prompt construction and manage LLM interaction, the *LangChain* framework [76] was employed, facilitating efficient and consistent execution.

6.2. Evaluation Process

The evaluation process represents the final phase of the DSR approach, where the performance of the instantiated artifact is systematically assessed. The following subsections introduce the datasets used for evaluation, followed by the evaluation dimensions and metrics.

To benchmark the performance of the GraphRAG system, it is compared against a naive RAG system (c.f. Figure 1). The naive system was instantiated in parallel with GraphRAG, maintaining a similar technical implementation to ensure a fair comparison.

To benchmark the performance of the GraphRAG system, we compared it against a naive RAG baseline, as shown in Figure 1. The naive RAG system was implemented in parallel with GraphRAG, using a comparable technical architecture to ensure a fair and consistent evaluation. In current industrial manufacturing environments, naive RAG systems represent the predominant architecture for QA applications. This context motivated the selection of the benchmark, enabling an initial assessment of GraphRAG's effectiveness in addressing the challenges identified through the DSR process, while also highlighting its technical feasibility and potential for immediate industrial impact.

The GraphRAG system incorporates several tunable parameters, providing data-driven insights into its performance. For evaluation purposes, the following parameters and their impact on retrieval effectiveness are analyzed:

- chunk size,
- number of extracted keywords,
- top-k retrieval and pass conditions.

Since these parameters primarily influence the retrieval phase of the RAG process, their evaluation aligns with the core objective of GraphRAG, which is to enhance retrieval performance in an industrial context.

Given the large number of potential parameter variations, a strategic evaluation approach was adopted to ensure efficient use of computational resources. Standard values for each parameter were determined based on related work, preliminary experiments, and expert insights. This led to the definition of a standard GraphRAG pipeline, which serves as the baseline configuration for comparison with the naive RAG system.

To systematically assess the impact of each parameter, isolated adjustments were performed to measure their influence on system performance. Since all parameters are quantifiable on a linear scale, their values were both increased and decreased to observe corresponding effects. Furthermore, for parameters applicable to both GraphRAG and the naive RAG setup (e.g., chunk size), evaluations were conducted for both configurations to ensure a direct and fair comparison.

This methodology facilitates comparisons not only with the standard GraphRAG pipeline but also against a naive baseline with equivalent parameter adjustments, providing a comprehensive analysis of parameter sensitivity and system behavior.

Furthermore, initial tests evaluating the performance of LLM-extracted keywords were conducted on the manufacturing domain-specific dataset to provide a preliminary assessment of this alternative approach. As with the algorithmic extraction, the evaluation was performed against the standard parameter settings to ensure comparability of results.

This evaluation strategy maximizes insights into the performance trade-offs of the GraphRAG approach, ensuring an optimal balance between depth of analysis and resource efficiency, a crucial factor in fast-paced domains such as RAG.

6.3. Datasets for Evaluation

To demonstrate the performance of the GraphRAG system, it is essential to utilize suitable datasets. This study focuses on applying RAG in the manufacturing domain. To the best of our knowledge, no publicly available dataset exists for evaluating RAG performance in this domain, particularly for German-language, automotive-specific content.

To address this gap, we constructed a proprietary, domain-specific dataset containing internal information from an automotive manufacturer. The dataset creation process is presented later within this section. While this approach ensures high domain relevance, it also imposes limitations on dataset publication due to confidentiality constraints.

To mitigate this limitation, parallel evaluations are conducted using established open-domain QA datasets for RAG benchmarking. This approach ensures both traceable evaluation results and demonstrates the generalizability of the proposed GraphRAG framework. As outlined in [77], open-domain datasets include different question types. To ensure robust and comprehensive evaluation results, the selected datasets contain a diverse set of question types, facilitating a balanced assessment of retrieval and reasoning capabilities.

For single-hop reasoning, the Stanford Question Answering Dataset (SQuAD) [78] was selected due to its high-quality, manually curated questions. Answering these questions requires deriving information from a single piece of evidence, typically from one retrieved chunk. Additionally, SQuAD includes unanswerable questions, requiring RAG systems to detect instances where insufficient information is available to generate a valid response. The presence of unanswerable questions enhances real-world applicability, as it reflects scenarios in which no definitive answer exists for a given user query [78].

In contrast, multi-hop reasoning necessitates retrieving and synthesizing multiple pieces of information, often from different documents or distant sections of the same document. This process involves establishing logical connections between retrieved chunks to construct a comprehensive response. Since the GraphRAG system is specifically designed to enhance retrieval across multiple documents and facilitate reasoning across diverse sources, the HotpotQA dataset [79] was selected as the second evaluation dataset. HotpotQA is explicitly designed for multi-hop reasoning, making it a suitable benchmark for evaluating the system's ability to integrate and reason across multiple information sources.

Despite being published prior to the introduction of RAG, these datasets remain highly suitable for RAG evaluation, as they provide question–answer pairs alongside their retrieval context, facilitating a structured assessment of retrieval and generation performance. Since both datasets are based on a corpus of *Wikipedia* articles, they can be seamlessly adapted for GraphRAG evaluation. Each *Wikipedia* article corresponds to a single document, while its sections naturally map to the chapter structure within the DKG, ensuring a consistent representation of the dataset in the proposed framework.

In addition to these open-domain datasets, a domain-specific internal dataset was created to evaluate the GraphRAG system in an industrial setting. The dataset consists of

German-language production planning documents, aligning with this study's objective of assessing RAG performance in specialized domains. The dataset integrates both external and internal sources, including reference books, industry standards, technical specifications, and supplier offers. These documents were carefully selected and validated by domain experts from an automotive manufacturer to ensure relevance and applicability. The final dataset comprises 17 documents, totaling over 5500 pages of text, making it comparable in scale to the context size of SQuAD.

The internal dataset was designed to mirror the characteristics of both SQuAD and HotpotQA, incorporating single-hop, multi-hop, and unanswerable questions. The selected open-domain datasets were manually created by their respective publishers or developed using crowdsourcing platforms [78,79]. However, for the internal dataset, these approaches were not feasible due to resource constraints on domain experts and the confidential nature of the data, which precludes crowdsourcing.

To address these limitations, the dataset was constructed using an LLM-based approach, following and replicating established methodologies for multi-hop dataset creation [79,80]. As a foundation for generating question–answer pairs, the DKG was first constructed from the provided documents. Utilizing the IKS layer, extracted keywords were used to identify and connect semantically related chunks.

An LLM was employed to generate diverse question types, followed by an automated quality assessment to filter low-quality outputs. Additionally, domain experts manually reviewed a sample of the generated question–answer pairs, refining the prompt engineering process to ensure domain relevance for manufacturing-specific queries.

The final step in dataset preparation involves the selection of evaluation questions. Given the large volume of questions in open-domain datasets, evaluating the entire set is infeasible due to resource constraints. To address this, the datasets are downsampled to 200 questions each, following the approach of prior studies that have demonstrated the effectiveness of evaluating a subset of 100 questions [81,82]. To ensure representativeness, questions are randomly sampled while adhering to predefined constraints that maintain a balanced distribution of question types, reflecting the original dataset composition.

6.4. Evaluation Metrics

The evaluation metrics were selected based on the RAG evaluation framework presented in [77]. The assessment is structured into distinct evaluation dimensions, measuring the performance of both the retrieval and generation phases.

Metrics for the retrieval phase are specifically used to evaluate the effectiveness of the retrieval component within RAG systems. Given that the proposed approach primarily focuses on retrieval, these metrics serve as the key indicators of system performance. Additionally, the influence of retrieval on the generated answer is assessed through generation-related evaluation dimensions, though its impact is considered secondary in comparison. It is important to note that other factors, such as prompt engineering, also influence generation performance, making it necessary to interpret results holistically [77].

Furthermore, the evaluation framework distinguishes between different types of evaluators: lexical metrics, which can be computed efficiently and objectively, and more advanced evaluation methods that rely on the use of LLM-as-a-Judge. While the latter enables a more flexible and semantic assessment of RAG systems, it is important to acknowledge that the generated metrics are heavily influenced by the choice of LLM, prompt design, and potential model biases. These factors can limit the reproducibility and comparability of results, as they could be mainly interpreted on a relative basis with controlled input parameters. Consequently, combining such advanced evaluation methods with independent lexical metrics is essential to provide a more robust and balanced assessment of the artifact's performance [77].

To evaluate the retrieval stage, the *Context Relevance* dimension is applied. This ensures that the retrieved context provides relevant chunks to answer the question while minimizing irrelevant information. As per the principles of RAG, a fixed number of top-k retrieved chunks is passed to the LLM. Consequently, the primary objective is to ensure that the correct chunks are retrieved, as the pipeline cannot directly constrain the number of irrelevant chunks [77,83].

To evaluate *Context Relevance*, we apply the lexical metrics **Recall@k** and **MRR@k** (Mean Reciprocal Rank). Recall@k measures the proportion of relevant passages retrieved within the top-k results, regardless of the presence of irrelevant passages. As previously discussed, irrelevant chunks are less critical, since the LLM is expected to focus on relevant information during the generation phase. MRR@k, on the other hand, prioritizes the ranking position of the first relevant chunk within the top-k retrieval, providing insights into how effectively the retrieval system ranks relevant content.

Both metrics, along with all other evaluation measures used in this study, range between 0 and 1, where higher values indicate better performance. The formulas for these calculations are provided in [77].

In addition to these lexical metrics, LLM-as-a-Judge-based metrics **Context Precision** (CP) and **Context Recall** (CR) are employed to assess retrieval performance without relying on a predefined "golden retrieval" from the dataset. These metrics are particularly valuable for evaluating RAG systems in real-world, unstructured data environments, where predefined relevance labels may not always be available and unique. For further details on their calculation, refer to [83–85].

To evaluate the quality of the generation process, we assess the dimensions of *Faithfulness*, *Answer Relevance*, and *Correctness*, as recommended in [77]. *Faithfulness* specifically pertains to the generation phase and measures the extent to which the LLM's response aligns with the retrieved context, ensuring that all information presented in the answer can be directly inferred from the retrieved content. This dimension is critical for detecting hallucinations and ensuring factual accuracy [86].

One approach to quantifying *Faithfulness* is the lexical metric **K-Precision**, which calculates the proportion of tokens in the LLM-generated response that appear in the retrieved context. A higher K-Precision score indicates a stronger alignment between the generated output and the provided retrieval context. Further details on the calculation of this metric can be found in [77]. Additionally, another applied metric with fewer dependencies on predefined labels is the LLM-based **Faithfulness Score** (FS). This metric evaluates whether each statement in the generated response is supported by the retrieved chunks, ensuring faithful adherence to the provided context and correct prompt following during generation. For further details on its computation, refer to [77,83].

The *Answer Relevance* dimension assesses whether the LLM-generated response directly addresses the user query, penalizing incomplete or redundant answers, regardless of factuality [83,87]. To quantify this, the **AnswerRelevance Score** (ARS) is computed using an LLM-as-a-Judge approach, which systematically evaluates response relevance and completeness. The methodology for calculating this metric is detailed in [77,83].

The third evaluation dimension, *Correctness*, pertains to the generation step and assesses whether the LLM-generated response accurately aligns with the ground-truth reference answer, often referred to as the "golden answer". In the case of open-domain datasets, these reference answers are provided by human annotators, whereas for the internal dataset, they are generated using an LLM. To measure *Correctness*, we apply the lexical metric **Recall**, as it has been shown to correlate well with human evaluations [86].

Recall quantifies how much of the essential content in the reference answer is captured in the LLM's response at the token level, without penalizing additional, non-contradictory information. The methodology for calculating this metric is described in [77,86]. In addition, an LLM-as-a-Judge approach is employed to compute the **Correctness Coefficient** (CC), which enables a more semantic evaluation beyond solely token-based comparisons. This metric ensures a higher-level assessment of correctness, accounting for paraphrasing and meaning equivalence rather than just verbatim matching. Further details on its calculation can be found in [77].

To ensure an automated and robust evaluation framework, the proposed lexical metrics were implemented as described in [77]. The assessment of LLM-as-a-Judge metrics was conducted using the established *RAGAS* framework introduced in [83].

Since LLM-based metrics are inherently non-deterministic, as they rely on the model's interpretative response, the robustness of the evaluation process needed to be ensured. Therefore, multiple evaluations were performed with identical inputs, resulting in highly consistent outputs with statistically insignificant variations. This confirmed that the evaluation framework maintains stability, despite the stochastic nature of LLM-based assessments.

As the evaluation process requires substantial reasoning capabilities, *Claude Sonnet 3.5* [75] was employed for metric computation. Notably, it was observed that the choice of LLM influenced the computed evaluation scores, emphasizing the necessity of standardizing the evaluation model. To ensure comparability across all runs, the same LLM was consistently applied, guaranteeing reproducible results.

7. Results

The evaluation results are first presented by introducing the standard GraphRAG pipeline, which serves as the baseline configuration for comparison against the naive RAG approach, before analyzing the effects of parameter variations as described in Section 6.2. In addition, a dedicated subsection presents the initial results of using LLM-extracted keywords as an alternative to the standard keyword extraction method. The chapter concludes with a validation of the practical application of the artifact addressing result quality and system performance from an end-user perspective as introduced in Section 3.5.

The standard pipeline parameters include a chunk size of approximately 1000 tokens. Additionally, the IKS layer of the graph is constructed by extracting five keywords per chunk to establish interconnections between chunks. The top-k value is set to 10, which applies both to the initial retrieval within the vector database, serving as the starting point for the graph search algorithms, and to the pass conditions that limit the number of chunks after reranking, before they are passed to the LLM. This configuration ensures comparability with the naive RAG approach.

Table 1 presents the evaluation metrics for these standard parameters, comparing the GraphRAG pipeline against the naive RAG system across the different datasets. **Bold** formatting indicates the best value for each metric–dataset pair, highlighting the strongest performance achieved in the respective configurations.

Table 1. Evaluation results for standard GraphRAG pipeline.

Approach	Dataset	Recall @10	MRR @10	CP	CR	K-Precision	FS	ARS	Recall	CC
Naive RAG	SQuAD	0.725	0.498	0.510	0.723	0.736	**0.945**	0.666	0.764	**0.377**
	HotpotQA	0.500	0.610	0.432	0.703	0.637	0.884	0.497	0.693	0.253
	Internal	0.055	0.040	0.485	0.729	0.680	0.898	0.663	0.680	0.505
GraphRAG	SQuAD	**0.760**	**0.613**	**0.635**	**0.793**	**0.759**	0.941	**0.693**	**0.795**	0.370
	HotpotQA	**0.650**	**0.668**	**0.537**	**0.770**	**0.691**	**0.901**	**0.590**	**0.772**	**0.257**
	Internal	**0.075**	**0.044**	**0.498**	**0.794**	**0.693**	**0.936**	**0.726**	**0.702**	**0.571**

The results presented in Table 1 indicate a consistent performance improvement for the GraphRAG approach across nearly all evaluation dimensions and datasets. As the proposed architecture primarily focuses on the retrieval phase of the RAG pipeline (c.f. Figure 3), this is particularly reflected in the *Context Relevance* evaluation dimension, where all four retrieval metrics show an increase. On average, improvements of 15.5% for SQuAD, 18.3% for HotPotQA, and 14.5% for the internal dataset were observed, demonstrating a substantial performance boost over the naive baseline under standard parameter settings.

For the generation-related evaluation dimension, performance improvements are observed across most metrics, although two metrics show a minor decrease for the SQuAD dataset. Overall, the GraphRAG approach achieves an average improvement of 1.8% for SQuAD, 8.4% for HotPotQA, and 6.4% for the internal dataset across all generation-related metrics. These results suggest that the proposed artifact not only enhances retrieval performance but also positively impacts the generation phase. The impact is especially pronounced in the more complex, multi-hop queries of HotpotQA, suggesting that GraphRAG is particularly well suited to advanced question-answering tasks. Nevertheless, for the internal dataset, lexical measures of *Context Relevance* remain relatively low, whereas LLM-as-a-Judge evaluations align more closely with results from open-domain benchmarks. This discrepancy likely stems from automatically created datasets in real-world scenarios and will be addressed further in Section 8.

Based on these initial findings, the next sections investigate the effect of adjusting the above parameters and discuss how such refinements might further optimize performance.

7.1. Adjust Chunk Size

To assess the impact of chunk size on retrieval performance, the GraphRAG approach was evaluated using both smaller chunks (700) and larger chunks (1300) while keeping all other parameters constant. Since chunk size also affects the performance of the naive RAG system, a corresponding evaluation was conducted for the adjusted naive approach to ensure a fair comparison.

As GraphRAG primarily focuses on retrieval improvements, Table 2 presents the performance results for different chunk sizes, focusing on the Context Relevance evaluation dimension. The results for all evaluation dimensions are provided in Appendix D.

For readability, the datasets are denoted as DS1 (SQuAD), DS2 (HotPotQA), and DS3 (internal dataset). As in previous evaluations, **bold** values indicate the best overall performance across both chunk sizes and RAG approaches, while underlined values highlight the highest performance within each individual approach.

Table 2. Evaluation results for context relevance with adjusted chunk size.

Approach	Chunk Size	Recall@k			MRR@k			Context Precision			Context Recall		
		DS1	DS2	DS3	DS1	DS2	DS3	DS1	DS2	DS3	DS1	DS2	DS3
Naive RAG	700	0.750	0.400	0.060	0.477	0.505	0.043	0.519	0.343	0.443	0.745	0.614	0.729
	1000	0.725	0.500	0.055	0.498	0.610	0.040	0.510	0.432	0.485	0.723	0.703	0.729
	1300	0.740	0.510	0.060	0.508	0.651	0.036	0.499	0.442	0.440	0.740	0.703	0.730
GraphRAG	700	**0.790**	0.626	**0.080**	0.568	0.626	0.043	**0.665**	0.525	0.470	0.788	0.773	**0.803**
	1000	0.760	**0.668**	0.075	0.613	**0.668**	0.044	0.635	0.537	**0.498**	**0.793**	0.770	0.794
	1300	0.765	0.665	**0.080**	**0.638**	0.665	**0.045**	0.637	**0.569**	0.478	0.775	**0.812**	0.749

In general, Table 2 demonstrates that GraphRAG consistently outperforms the naive approach across all chunk sizes and datasets for the evaluated retrieval metrics, reinforcing its effectiveness as a retrieval enhancement technique.

The results indicate that each dataset exhibits an optimal chunk size, suggesting that the ideal configuration is task-dependent. For example, SQuAD achieves better performance with smaller chunk sizes, whereas HotpotQA benefits from larger chunks. This underscores the importance of tuning chunk size based on task-specific requirements, while also highlighting the robust performance of GraphRAG across diverse configurations.

Notable variations in performance are observed across different chunk sizes. While Recall@k favors smaller chunk sizes, MRR@k tends to perform better with larger chunks, potentially influenced by differences in metric computation methodologies. Analyzing the LLM-based metrics, CP and CR, reveals a distribution of peak performances across different chunk sizes and datasets, further justifying the selection of 1000 tokens as the baseline chunk size.

These findings are further corroborated by the generation-phase metrics (c.f. Appendix D). While GraphRAG maintains a consistent advantage over the naive approach, the presence of task-dependent optimal chunk sizes remains evident. SQuAD continues to perform best with smaller chunks, while HotpotQA benefits from larger ones. For the internal dataset, the optimal results are achieved with the middle chunk size of 1000 tokens, emphasizing the pipeline's adaptation for retrieval in the manufacturing domain.

7.2. Adjust Number of Keywords

The second parameter analyzed is the number of keywords used to populate the IKG layer within the GraphRAG approach. To evaluate its impact on retrieval performance, the standard setting of five keywords per chunk was adjusted, increasing it to eight and decreasing it to three. This modification allows for an investigation into the relationship between keyword density and retrieval effectiveness.

Table 3 presents the evaluation results for different keyword settings. Since the number of keywords exclusively affects the GraphRAG approach, the naive RAG system is not included in this evaluation. As in previous tables, the highest values are highlighted in **bold** to indicate the best-performing configuration for each metric.

Table 3. Evaluation results for context relevance with adjusted number of keywords.

Approach	Keywords	Recall@10			MRR@10			Context Precision			Context Recall		
		DS1	DS2	DS3	DS1	DS2	DS3	DS1	DS2	DS3	DS1	DS2	DS3
GraphRAG	3	0.745	0.635	0.070	**0.615**	**0.698**	0.040	0.643	**0.578**	**0.518**	0.761	**0.814**	0.814
	5	0.760	0.650	**0.075**	0.613	0.668	0.044	0.635	0.537	0.498	0.793	0.770	0.794
	8	**0.790**	**0.700**	0.065	0.609	0.672	**0.050**	**0.656**	0.554	0.503	**0.795**	0.806	**0.816**

Analysis of Table 3 reveals a clear trade-off between recall-oriented measures and ranking precision. Increasing the number of keywords (e.g., to eight) often enhances Recall@10 and CR but may lead to a slight reduction in MRR@10. Conversely, reducing the keyword count (e.g., to three) tends to improve ranking-focused metrics, such as MRR@10 and CP, but at the cost of overall retrieval coverage. Notably, the default configuration of five keywords appears to provide a balanced performance across multiple datasets, suggesting that this setting effectively mitigates the trade-off between recall and ranking precision.

These findings indicate that optimal keyword selection is highly task-dependent, with different priorities influencing parameter tuning. If maximizing retrieval coverage is the primary goal, a higher keyword count may be preferable. Conversely, if precision in top-ranked results is more critical, a lower keyword count could be advantageous.

This trade-off is also reflected in the generation-related metrics (c.f. Appendix D), further emphasizing the importance of domain-specific optimization. Adjusting the number of keywords can impact both retrieval effectiveness and answer generation quality, underscoring the need for task-specific parameter tuning to align with domain requirements.

7.3. Adjust Top-k Retrieval

The final parameter adjusted in the evaluation is top-k retrieval, which influences both initial vector retrieval and pass conditions. The standard value of 10 was modified to 5 and 20 to assess its impact on GraphRAG performance. Since top-k retrieval also affects the naive RAG approach, evaluations were conducted for both systems to ensure a comprehensive comparison.

Table 4 presents the evaluation results, with the highest-performing values highlighted, both for each approach individually and overall across configurations.

Table 4. Evaluation results for context relevance with adjusted top-k retrieval.

Approach	Top-k	Recall@k			MRR@k			Context Precision			Context Recall		
		DS1	DS2	DS3	DS1	DS2	DS3	DS1	DS2	DS3	DS1	DS2	DS3
Naive RAG	5	0.655	0.395	0.045	0.500	0.652	0.043	0.520	0.440	0.494	0.638	0.655	0.660
	10	0.725	0.500	0.055	0.498	0.610	0.040	0.510	0.432	0.485	0.723	0.703	0.729
	20	0.830	0.585	0.080	0.512	0.622	0.040	0.507	0.415	0.453	0.809	0.801	0.829
GraphRAG	5	0.705	0.550	0.065	**0.894**	**0.701**	**0.058**	0.642	0.537	**0.544**	0.722	0.770	0.746
	10	0.760	0.650	0.075	0.613	0.668	0.044	0.635	0.537	0.498	0.793	0.770	0.794
	20	**0.840**	**0.735**	**0.085**	0.645	0.649	0.037	**0.657**	**0.578**	0.472	**0.847**	**0.833**	**0.892**

Table 4 illustrates the impact of varying the top-k retrieval parameter from 5 to 20 on both Naive RAG and GraphRAG, focusing on *Context Relevance* metrics. As expected, higher k-values lead to an increase in Recall@k and CR, particularly for SQuAD and HotpotQA, while lower k-values tend to improve MRR@k and CP. Across all configurations, GraphRAG consistently outperforms the naive approach in most metrics, demonstrating the robustness of the graph-based retrieval architecture.

However, the results also highlight a clear trade-off: when k is too large, the system retrieves a broader range of relevant content, but at the cost of introducing more noise into the top-ranked results, thereby reducing precision-based scores. Consequently, the optimal top-k selection depends on the specific application requirements, whether the priority is maximizing retrieval coverage (favoring higher k) or ensuring high-ranking precision (favoring lower k).

Notably, with lower top-k values, the retrieval performance of the naive approach deteriorates more significantly than that of GraphRAG. This further underscores the advantages of the proposed graph-based architecture, as its multi-step retrieval process within the KG mitigates the loss of relevant content.

Analysis of the generation-related evaluation metrics (c.f Appendix D) reveals a parallel trend between retrieval and generation performance. Specifically, for higher top-k values, performance also improves within the naive RAG setup, in some cases surpassing the performance of GraphRAG with lower k-values. However, even under these conditions, GraphRAG maintains superior performance compared to the naive approach at equivalent retrieval settings, reinforcing the effectiveness of its structured retrieval framework.

7.4. Initial Evaluation of LLM-Based Keyword Extraction

Another dimension of the evaluation involves the adjustment of the keyword extraction method. As previously described, the main evaluation was conducted using YAKE as an algorithmic baseline to ensure reproducibility and computational efficiency. To demonstrate the adaptability of GraphRAG for domain-specific QA scenarios, an alternative approach based on LLM-driven keyword extraction was explored.

This section presents initial results using an LLM-based method to extract relevant keywords. While the public datasets focus on open-domain content, this evaluation leverages the internal dataset to assess the impact of domain-specific keywords in the manufacturing context.

Table 5 summarizes the results, with the highest-performing values highlighted in **bold**. The evaluation was conducted using the standard parameter settings applied to the internal dataset.

Table 5. Evaluation results for context relevance with adjusted keyword extraction for the internal dataset with chunk size 1000 and top 10 retrieval.

Approach	Keywords	Extractor	Recall@10	MRR@10	CP	CR
Naive	–	–	0.055	0.040	0.485	0.729
GraphRAG	5	YAKE	**0.075**	**0.044**	0.498	0.794
GraphRAG	5	LLM	0.070	0.041	**0.508**	**0.809**

Interpreting the results in Table 5 reveals an improvement when using LLM-based keyword extraction compared to both the naive baseline and the algorithmic approach based on YAKE. Although the performance gains are moderate—especially in contrast to the significant improvement from the naive method to YAKE—the results indicate that leveraging a large language model for domain-specific keyword extraction can further enhance answer quality.

A closer examination of the extracted keywords for a specific chunk confirms improved relevance and specificity with the LLM-based approach, as also noted by the focus group. While YAKE tends to extract general terms such as "*General Requirements*" ("*Allgemeine Anforderungen*"), the LLM identifies more targeted keywords like "*Pressure Reductions*" ("*Druckreduzierungen*") and "*Storage Charge Circuits*" ("*Speicherladeschaltungen*"), which can themselves be considered types of requirements. Although these LLM-based keywords more precisely reflect the content of the chunk, it should be noted that increased specificity also results in a denser IKG layer with fewer interconnections between chunks. Moreover, for such keywords to contribute effectively to retrieval, they must also be identifiable within the user query, which is essential for use in the Informed Keyword Search (IKS).

While this constitutes an initial assessment of LLM-based keyword extraction, the results demonstrate its potential. The observed gains in semantic clarity and contextual completeness may contribute to more accurate downstream answers, particularly in manufacturing-specific question-answering tasks.

7.5. Validation of Practical Application

To ensure a comprehensive evaluation of the implemented artifact, the practical applicability was qualitatively validated in addition to the quantitative evaluation. For this purpose, the end user focus group involved in the solution design process reviewed and assessed the generated results, providing domain-specific feedback on their relevance and usefulness.

The primary focus of the qualitative validation was to assess the quality of the generated results through the review of a random subset of answers produced within the dataset. The focus group concluded that, particularly for complex multi-hop queries, the quality of answers generated by GraphRAG was notably higher compared to those from the naive pipeline. This observation is consistent with the results of the quantitative evaluation and supports the validity of the applied metrics and evaluators. Moreover, the findings confirm the practical applicability of the proposed approach from the perspective of answer quality.

The second perspective of the qualitative evaluation focuses on the practical assessment of processing times and latency. Although the technical implementation primarily aimed to evaluate the system's output quality, initial measurements were conducted to gain insights into runtime behavior. It should be noted, however, that a real-world deployment would require implementation on more powerful and robust hardware to meet practical performance requirements.

The results showed that only a relatively small amount of time was required to construct the KG as part of the indexing process. Based on the chunk corpus generated using the standard parameters, which would also be required in a naive RAG setup, one dataset comprising approximately 5500 pages of text was successfully transferred into the KG within 195 s. As this process is executed during the initial index construction and does not affect retrieval latency, it was not considered a concern by the users in the focus group.

To assess practical applicability, end-to-end latencies were measured using the standard evaluation parameters. Following Xu et al. [88], total response time is treated as a user-centered efficiency metric. In the naive RAG pipeline, retrieval required 0.2 s and answer generation 1.5 s, resulting in a total latency of 1.7 s per query. In contrast, the GraphRAG implementation recorded 7.4 s for retrieval, 0.9 s for reranking, and 1.5 s for generation, yielding a total latency of 9.8 s. Although this represents a fivefold increase, it remains well below the ten-second threshold deemed acceptable by focus group participants. Moreover, it is significantly faster than manual searches across heterogeneous file systems, which may take several minutes per query [3]. The additional delay introduced by GraphRAG is therefore considered a reasonable trade-off for improved answer quality, consistent with the speed–accuracy trade-offs reported for recent LLMs such as OpenAI's o1 [89].

In conclusion, the qualitative validation confirmed that GraphRAG provides substantial practical benefits in terms of answer quality, particularly for complex, multi-hop queries, despite an increase in retrieval latency. These findings suggest that the trade-off between enhanced accuracy and slightly higher response times is acceptable for the implementation of GraphRAG within the manufacturing domain.

8. Discussion

The results across all datasets demonstrate that GraphRAG consistently outperforms the naive RAG baseline in both retrieval and generation metrics, reinforcing the notion that enhanced retrieval directly supports improved answer generation in RAG-based QA systems. From a broader perspective, the importance of a graph-based retrieval structure becomes evident: by explicitly modeling relationships between chunks via keyword-based edges, GraphRAG systematically retrieves relevant evidence, leading to meaningful improvements across datasets. While SQuAD's relatively straightforward queries benefit from more precise retrieval, the complex multi-hop questions in HotpotQA see even greater performance gains, leveraging GraphRAG's structured knowledge representation. The results also indicate that each dataset exhibits distinct optimal configurations for specific metrics, emphasizing that no single parameter setting universally optimizes all dimensions. Instead, task-specific parameter tuning is essential for maximizing performance. The systematic approach of the DSR methodology has proven well suited for developing and customizing a domain-specific RAG solution within the manufacturing context.

A key objective of this study was to examine the impact of specific parameter choices on system performance. Notably, a chunk size of 1000 tokens and a moderate number of keywords proved to be effective across multiple tasks, striking a balance between retrieval coverage and computational efficiency. Furthermore, the results highlight the necessity of adapting parameters to domain-specific requirements, as optimal configurations vary depending on dataset characteristics and question types.

Regarding top-k retrieval optimization, results indicate that increasing the number of retrieved chunks significantly improves recall-oriented metrics such as Recall@k and

CR, but at the cost of introducing noise that reduces precision-based scores. Interestingly, GraphRAG demonstrates greater robustness than the naive RAG approach across varying top-k values, suggesting that its structured retrieval process retains relevant information more effectively, even under lower k-values.

The initial assessment of LLM-based keyword extraction highlights the potential of incorporating domain-specific terms into the IKG. While the results indicate performance improvements over the algorithmic baseline, there remains room for further enhancement, particularly through refined prompt design and optimized retrieval strategies. Such improvements are essential to justify the increased computational cost and latency associated with LLM-based approaches when compared to more lightweight algorithmic solutions.

The integration of GraphRAG into the manufacturing domain yielded valuable insights from the internal dataset. Due to noise in the data, particularly from tender specifications with large sections of repeated text, lexical metrics struggled to accurately assess performance. The evaluation required both relevant documents to be retrieved for a multi-hop question to be counted as correct. In practice, the system often retrieved semantically equivalent content from different documents, leading to false negatives. While such redundancy is common in industrial settings, LLM-based metrics offered a more context-aware evaluation. These results confirmed GraphRAG's strong performance on the internal dataset, aligning with findings from open-domain benchmarks. Despite limitations, the consistent interpretation of LLM-based metrics alongside lexical ones supports their validity in less standardized environments. Overall, the results demonstrate the approach's adaptability and suggest that, with proper parameter tuning, graph-based retrieval remains effective in complex, domain-specific scenarios.

Despite the demonstrated benefits, certain limitations must be acknowledged. The first concern pertains to dataset creation. Although a comprehensive approach was employed, the dataset was primarily LLM-generated, with subsamples manually verified by domain experts. Incorporating a greater proportion of handcrafted questions could further enhance the quality of evaluation results.

Furthermore, the evaluation process uses a naive RAG system as the baseline, as this currently represents the state of the art within the targeted industrial domain. While GraphRAG demonstrates improved performance in this setting, several advanced RAG architectures have been developed within the scientific community. A comparison with these more sophisticated approaches could provide a deeper assessment of the proposed solution's performance and further validate its effectiveness beyond the industrial baseline.

Another limitation concerns the implemented text and keyword extraction components. Despite reviewing the document structure and optimizing post-processing, some edge cases in the real-world dataset may still result in incomplete structural capture. Likewise, even with parameter tuning supported by domain experts, less relevant keywords may occasionally be extracted, affecting retrieval precision. Nonetheless, GraphRAG outperforms the naive baseline, illustrating the approach's robustness. These findings point to clear opportunities for improvement, as discussed in the future work section.

A further limitation concerns the evaluation process. While LLM-as-a-Judge is a practical and scalable method for QA tasks, incorporating human-in-the-loop evaluation could improve result reliability, despite the focus group reviewing a subset of answers. The influence of prompt engineering must also be acknowledged. Preliminary tests showed that small changes to the prompt template and switching the evaluator to Haiku [73] significantly increased evaluation scores. This partly accounts for the relatively low values in the *Correctness* dimension, which is highly sensitive to both prompt design and model choice. As such, correctness scores should be interpreted relatively rather than in absolute terms.

The latency evaluation was based on a prototype implementation without optimization, running on low-end hardware. While the results provided useful insights and were acceptable to the focus group, further improvements are needed for real-world QA deployment. Scalable database solutions are especially recommended to reduce latency for graph-based queries. Future work will address scalability and resource efficiency to assess feasibility for large-scale use.

Overall, these findings affirm the potential of GraphRAG to enhance retrieval quality across a range of QA tasks, while also emphasizing the importance of careful parameter tuning, thoughtful data curation, and continued research into scalable, domain-adaptive retrieval solutions.

9. Conclusions and Outlook

The development and evaluation of the GraphRAG framework were guided by the DSR methodology, ensuring a structured and iterative approach to addressing real-world retrieval challenges in RAG-based QA systems. Through systematic problem identification, definition of design requirements, artifact development, implementation, and evaluation, this study has demonstrated the effectiveness of integrating KGs into RAG. By applying DSR principles, the research not only developed a novel DKG-based retrieval framework but also established a methodological foundation for designing task-specific and domain-adaptive QA systems.

The key contributions of this work are twofold: first, the systematic design and empirical validation of GraphRAG as a scalable and adaptable retrieval solution, and second, the critical assessment of parameter dependencies, offering insights into task-specific optimization strategies. These findings reinforce the value of DSR as an effective methodology for developing and refining AI-driven retrieval systems in complex industrial environments.

While several areas for further refinement have already been discussed, additional directions for future research offer potential to further strengthen the proposed framework. As noted, a deeper focus on KG construction and retrieval could improve answer quality, particularly by incorporating chapter summaries into the DKG to enhance the guidance provided during retrieval.

Beyond *AWS Textract*, the pipeline is extractor-agnostic and supports layout-aware multimodal Transformers. Their geometry-aware token positions could enable the injection of structural edges (e.g., *section*, *table-cell*, *figure-caption*) into the DKG, allowing systematic evaluation of their impact on retrieval accuracy.

Furthermore, keyword extraction for the IKG layer could be more closely aligned with domain-specific needs by further refining the LLM-based approach. The IKG itself may benefit from filtering overly generic keywords, as high-frequency terms linked to many chunks can reduce retrieval specificity. Given the strong performance of both unsupervised and initial LLM-based extraction methods, fine-tuning this component offers a promising path to enhance performance in specialized domains. It would also be beneficial to assess the individual impact of each graph search algorithm to better understand their respective contributions to retrieval effectiveness.

Another promising avenue for future work is enhancing the interaction between the vector and graph databases. The document management functionalities of the DKG could be leveraged to narrow the search space within the vector database before initial retrieval, either through user-defined filters or automatically extracted constraints.

An additional interesting approach involves utilizing the DKG to establish links between identical or highly similar chunks, thereby preventing the retrieval of redundant or non-diverse content, a challenge that frequently arises in real-world applications where documents contain overlapping sections, as observed in the internal dataset.

Extending GraphRAG to multilingual settings is a natural next step. Balanced multilingual LLMs as in [90], which produce language-agnostic embeddings, enable semantic edges linking concept nodes with equivalent meanings across languages. These cross-lingual links could replace manual synonym lists, enrich the IKG beyond basic keyword matches, and support multilingual QA without requiring retraining or re-indexing of the full pipeline.

Inspired by the recently introduced CoLE framework [46], a promising direction for future work is to incorporate a lightweight reasoning layer on top of GraphRAG. This could begin with a brief prompt that classifies the user's query and assigns semantic tags to relevant graph nodes. The existing document-graph traversal would then serve as the primary retrieval mechanism. A second prompt could be used to select a particularly instructive passage from the retrieved set. The final answer could follow a simple, fixed structure, explicitly stating the task, the governing rule, the supporting observations, and the resulting conclusion.

Finally, an investigation into expanded use cases could further validate and refine the GraphRAG approach. While the current evaluation includes both English and German datasets, future research could explore the system's performance in multilingual settings. Additionally, it would be valuable to assess whether the methodology and findings can be effectively transferred to other domain-specific QA tasks, broadening the applicability of GraphRAG beyond the manufacturing domain.

Author Contributions: Conceptualization, S.K., O.C. and D.G.; methodology, S.K. and O.C.; software, S.K. and O.C.; investigation, S.K. and O.C.; resources, S.K. and D.G.; data curation, S.K. and O.C.; writing—original draft preparation, O.C.; writing—review and editing, S.K.; visualization, S.K.; supervision, D.G. All authors have read and agreed to the published version of the manuscript.

Funding: The research was funded by the AUDI AG.

Data Availability Statement: The data supporting the findings of this study are owned by AUDI AG and are not publicly available due to proprietary restrictions. However, data may be available from the authors upon reasonable request and with permission of Audi.

Acknowledgments: The present approach was developed within the institute "AImotion Bavaria" at the Technische Hochschule Ingolstadt.

Conflicts of Interest: The authors declare that this study received funding from AUDI AG. The funder was involved in the study through the provision of data and computing resources.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
API	Application Programming Interface
ARS	Answer Relevance Score
AWS	Amazon Web Services
CC	Correctness Coefficient
CoLE	Collaborative Legal Expert
CP	Context Precision
CR	Context Recall
DKG	Document Knowledge Graph
DS1	SQuAD Dataset
DS2	HotpotQA Dataset
DS3	Internal Dataset
DSR	Design Science Research
FMEA	Failure Mode Effect Analysis

FS	Faithfulness Score
GraphRAG	Document GraphRAG
ICS	Informed Chapter Search
IKG	Information Knowledge Graph
IKS	Informed Keyword Search
JSON	JavaScript Object Notation
KG	Knowledge Graph
LLM	Large Language Model
MLLM	Multilingual Large Language Model
MRR	Mean Reciprocal Rank
OCR	Optical Character Recognition
PoV	Proof of Value
QA	Question Answering
RAG	Retrieval-Augmented Generation
SPARQL	SPARQL Protocol and RDF Query Language
SQuAD	Stanford Question Answering Dataset
UKS	Uninformed Keyword Search
URI	Uniform Resource Identifier

Appendix A. Prompt Template Generation

System: You are an AI model working in a Retrieval Augmented Generation pipeline. Your task is to generate an answer based on pre-retrieved context.

Human: You will be provided with some context and a question. Your goal is to answer the question using only the information given in the context. Additionally, you should cite exactly where you are grounding your answer from within the context.

Here is the context you should use to answer the question:
<context>
CONTEXT
</context>

Here is the question you need to answer:
<question>
QUESTION
</question>

Instructions for answering:
1. Carefully read and analyze the provided context.
2. Read and understand the question.
3. Search for relevant information in the context that can help answer the question.
4. If you find relevant information, use it to formulate your answer. Only use information from the provided context.
5. If you cannot find any relevant information to answer the question, state that you cannot answer the question based on the given context by responding: "This question cannot be answered with the provided context.".
6. Do not make assumptions or use external knowledge not present in the context.
7. Cite exactly where you are grounding your answer from within the context by referencing the relevant sections or chunks.

Please provide your answer in the following format:

<answer>
Your response here. If you can answer the question, provide a clear and concise answer based solely on the context, and cite exactly where you are grounding your answer from. If you cannot answer the question, state: "This question cannot be answered with the provided context."
</answer>

Assistant: <answer>

Appendix B. Prompt Template Keyword Extraction

System: You are an AI assistant specialized in identifying and extracting key technical terminology and significant concepts from domain-specific texts within the manufacturing domain. Your task is to analyze provided text passages and identify the most relevant and distinctive keywords that capture the main concepts and important terms.

Human: Please extract the top 5 keywords from the provided text. The keywords can be unigrams or bigrams. The keywords should be unique and significant in the text to be considered as keywords. Do not use any special characters or symbols in the keywords; only use letters, numbers, spaces, hyphens (-), and underscores (_). Common nouns or regular words should not be considered. You are also allowed to use years (integers) as keywords if they seem significant in the context of the text. Special and unique words or years are suitable as keyword candidates.

<context>
text_chunk
</context>

Instructions for answering:
1. Carefully read and analyze the provided text in the context.
2. Extract unique and significant unigrams or bigrams within the context.
3. Ensure that the keywords do not contain any special characters or symbols; only letters, numbers, spaces, hyphens (-), and underscores (_) are allowed.
4. If you think there are less than 5 keywords, that's okay.
5. Do not make assumptions or use external knowledge not present in the context.
6. Do not include any explanations or additional text outside the list of keywords.
7. Ensure the output is a valid JSON array with double quotes only.

Please return the keywords in the following format:
["first keyword", "second keyword", ..., "final keyword"]

Assistant: [

Appendix C. Prompt Template Reranking

System: You are an expert in processing and filtering search results. Help the user in sorting the search results.

Human: You will be provided with a list of text chunks, each enclosed within <text_chunk> </text_chunk> tags. Every text chunk comes with a unique index and uri attribute. It is crucial that you exactly copy these index and uri values without any modifications. Additionally, a question is provided within <question_text> </question_text> tags that you must use to evaluate the relevance of each text chunk.

Your tasks are as follows:

1. Analyze the Question:
- Carefully read and understand the question provided within the <question_text> </question_text> tags.

2. Evaluate Each Text Chunk in <text_chunk>:
- Determine its relevance to the question.
- Assign a relevance_score between 1 and 10, where:
- 10 indicates high relevance to answer the question.
- 1 indicates that the text chunk is not relevant at all for answering the question.
- Provide a justification for the assigned score, explaining why the text chunk is relevant or not relevant to the question.

3. Assign Scores Even to Irrelevant Chunks:
- Even if a text chunk does not contribute to answering the question, you must still assign it a relevance_score and provide a justification.

4. Sort the Results:
- After scoring, sort all text chunks in descending order based on their relevance_score.

5. Format Your Response:
- Your entire response must be enclosed within <ranking></ranking> tags.
- Follow the structure provided in the example below to ensure consistency.

Example:
<Example>
<ranking>
<text_chunk index="0" uri="http://example.com/chunk0">
<justification>The content directly addresses the main aspects of the question.</justification>
<relevance_score>9</relevance_score>
</text_chunk>
<text_chunk index="1" uri="http://example.com/chunk1">
<justification>The information is somewhat related but lacks specific details.</justification>
<relevance_score>5</relevance_score>
</text_chunk>
<text_chunk index="2" uri="http://example.com/chunk2">
<justification>The content is not relevant to the question at all.</justification>
<relevance_score>2</relevance_score>
</text_chunk>
</ranking>
</Example>

Now, proceed with the following data:

<text_chunks>
text_chunks_for_data_point_id
</text_chunks>

<question_text>
question_text_for_data_point_id
</question_text>

Assistant: Here is the new ranking of the text chunks based on their relevance to the question:

<ranking>

Appendix D. Evaluation Results

Appendix D.1. SQuAD Dataset

Table A1. Evaluation results for the SQuAD dataset.

Approach	Chunk Size	Key-words	Top-k	Recall @k	MRR @k	CP	CR	K-Prec.	FS	ARS	Recall	CC
Naive RAG	700	-	10	0.750	0.477	0.519	0.745	0.729	0.949	0.675	0.773	0.383
	1000	-	5	0.655	0.500	0.520	0.638	0.675	0.939	0.603	0.707	0.353
	1000	-	10	0.725	0.498	0.510	0.723	0.736	0.945	0.666	0.764	0.377
	1000	-	20	0.830	0.512	0.507	0.809	0.797	0.939	0.714	0.816	0.380
	1300	-	10	0.740	0.508	0.499	0.740	0.752	0.944	0.670	0.755	0.373
GraphRAG	700	5	10	0.790	0.568	0.665	0.788	0.749	0.958	0.732	0.817	0.402
	1000	3	10	0.745	0.615	0.643	0.761	0.752	0.958	0.665	0.775	0.375
	1000	5	5	0.705	0.594	0.642	0.722	0.705	0.960	0.672	0.774	0.370
	1000	5	10	0.760	0.613	0.635	0.793	0.759	0.941	0.693	0.795	0.370
	1000	5	20	0.840	0.645	0.657	0.847	0.812	0.943	0.720	0.861	0.400
	1000	8	10	0.790	0.609	0.656	0.795	0.759	0.942	0.713	0.822	0.387
	1300	5	10	0.765	0.638	0.637	0.775	0.764	0.950	0.674	0.790	0.382

Appendix D.2. HotpotQA Dataset

Table A2. Evaluation results for the HotpotQA dataset.

Approach	Chunk Size	Key-words	Top-k	Recall @k	MRR @k	CP	CR	K-Prec.	FS	ARS	Recall	CC
Naive RAG	700	-	10	0.400	0.505	0.343	0.614	0.597	0.879	0.398	0.628	0.217
	1000	-	5	0.395	0.652	0.440	0.655	0.580	0.863	0.458	0.642	0.230
	1000	-	10	0.500	0.610	0.432	0.703	0.637	0.884	0.497	0.693	0.253
	1000	-	20	0.585	0.622	0.415	0.801	0.719	0.891	0.573	0.724	0.261
	1300	-	10	0.510	0.651	0.442	0.703	0.659	0.888	0.496	0.697	0.229
GraphRAG	700	5	10	0.580	0.626	0.525	0.773	0.725	0.896	0.571	0.769	0.260
	1000	3	10	0.635	0.698	0.578	0.814	0.737	0.906	0.613	0.774	0.264
	1000	5	5	0.550	0.701	0.537	0.770	0.629	0.888	0.555	0.759	0.252
	1000	5	10	0.650	0.668	0.537	0.770	0.691	0.901	0.590	0.772	0.257
	1000	5	20	0.735	0.649	0.578	0.833	0.762	0.929	0.665	0.828	0.266
	1000	8	10	0.700	0.672	0.554	0.806	0.751	0.924	0.621	0.787	0.256
	1300	5	10	0.665	0.665	0.569	0.812	0.751	0.924	0.631	0.810	0.269

Appendix D.3. Internal Dataset

Table A3. Evaluation results for the internal dataset.

Approach	Chunk Size	Key-words	Top-k	Recall @k	MRR @k	CP	CR	K-Prec.	FS	ARS	Recall	CC
Naive RAG	700	-	10	0.060	0.043	0.443	0.729	0.658	0.904	0.636	0.662	0.506
	1000	-	5	0.045	0.043	0.494	0.66	0.592	0.911	0.543	0.636	0.479
	1000	-	10	0.055	0.040	0.485	0.729	0.680	0.898	0.663	0.680	0.505
	1000	-	20	0.080	0.040	0.453	0.829	0.746	0.907	0.752	0.717	0.539
	1300	-	10	0.060	0.036	0.44	0.73	0.684	0.881	0.632	0.642	0.484
GraphRAG	700	5	10	0.080	0.043	0.470	0.803	0.682	0.935	0.718	0.691	0.543
	1000	3	10	0.070	0.040	0.518	0.814	0.691	0.933	0.704	0.707	0.575
	1000	5	5	0.065	0.058	0.544	0.746	0.612	0.935	0.636	0.683	0.527
	1000	5	10	0.075	0.044	0.498	0.794	0.693	0.936	0.726	0.702	0.571
	1000	5	20	0.085	0.037	0.472	0.892	0.755	0.934	0.972	0.734	0.578
	1000	8	10	0.065	0.050	0.503	0.816	0.700	0.925	0.712	0.702	0.552
	1300	5	10	0.080	0.045	0.478	0.749	0.687	0.894	0.666	0.675	0.489

References

1. Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y.T.; Li, Y.; Lundberg, S.; et al. Sparks of Artificial General Intelligence: Early experiments with GPT-4. *arXiv* **2023**. [CrossRef]
2. Kandpal, N.; Deng, H.; Roberts, A.; Wallace, E.; Raffel, C. Large Language Models Struggle to Learn Long-Tail Knowledge. *arXiv* **2022**. [CrossRef]
3. Knollmeyer, S.; Mroß, B.; Müller, R.K.; Großmann, D. Ontology based knowledge graph for information and knowledge management in factory planning. In Proceedings of the 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), Sinaia, Romania, 12–15 September 2023; pp. 1–4. [CrossRef]
4. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-T.; Rocktäschel, T.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS 2020), Vancouver, BC, Canada, 6–12 December 2020; Volume 33, pp. 9459–9474. Available online: https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf (accessed on 15 February 2025).
5. Izacard, G.; Lewis, P.; Lomeli, M.; Hosseini, L.; Petroni, F.; Schick, T.; Dwivedi-Yu, J.; Joulin, A.; Riedel, S.; Grave, E. Atlas: Few-shot Learning with Retrieval Augmented Language Models. *arXiv* **2022**. [CrossRef]
6. Pan, J.Z.; Vetere, G.; Gomez-Perez, J.M.; Wu, H. *Exploiting Linked Data and Knowledge Graphs in Large Organisations*; Springer International Publishing: Cham, Switzerland, 2017. [CrossRef]
7. Wang, Y.; Lipka, N.; Rossi, R.A.; Siu, A.; Zhang, R.; Derr, T. Knowledge Graph Prompting for Multi-Document Question Answering. *arXiv* **2023**. [CrossRef]
8. Pan, S.; Luo, L.; Wang, Y.; Chen, C.; Wang, J.; Wu, X. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Trans. Knowl. Data Eng.* **2024**, *36*, 3580–3599. [CrossRef]
9. 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* **2023**. [CrossRef]
10. Knollmeyer, S.; Akmal, M.U.; Koval, L.; Asif, S.; Mathias, S.G.; Großmann, D. Document Knowledge Graph to Enhance Question Answering with Retrieval Augmented Generation. In Proceedings of the 2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA), Padova, Italy, 10–13 September 2024; pp. 1–4. [CrossRef]
11. Asai, A.; Wu, Z.; Wang, Y.; Sil, A.; Hajishirzi, H. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. *arXiv* **2023**. [CrossRef]
12. Wang, X.; Wang, Z.; Gao, X.; Zhang, F.; Wu, Y.; Xu, Z.; Shi, T.; Wang, Z.; Li, S.; Qian, Q.; et al. Searching for Best Practices in Retrieval-Augmented Generation. *arXiv* **2024**. [CrossRef]
13. Wu, S.; Xiong, Y.; Cui, Y.; Wu, H.; Chen, C.; Yuan, Y.; Huang, L.; Liu, X.; Kuo, T.W.; Guan, N.; et al. Retrieval-Augmented Generation for Natural Language Processing: A Survey. *arXiv* **2024**. [CrossRef]
14. Ni, J.; Qu, C.; Lu, J.; Dai, Z.; Hernandez Abrego, G.; Ma, J.; Zhao, V.; Luan, Y.; Hall, K.; Chang, M.W.; et al. Large Dual Encoders Are Generalizable Retrievers. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Abu Dhabi, United Arab Emirates, 7–11 December 2022; pp. 9844–9855. [CrossRef]
15. Chen, J.; Lin, H.; Han, X.; Sun, L. Benchmarking Large Language Models in Retrieval-Augmented Generation. *arXiv* **2023**. [CrossRef]

16. 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* **2020**. [CrossRef]
17. Malkov, Y.A.; Yashunin, D.A. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *arXiv* **2016**. [CrossRef] [PubMed]
18. Asai, A.; Schick, T.; Lewis, P.; Chen, X.; Izacard, G.; Riedel, S.; Hajishirzi, H.; Yih, W.-T. Task-aware Retrieval with Instructions. *arXiv* **2022**. [CrossRef]
19. Mallen, A.; Asai, A.; Zhong, V.; Das, R.; Khashabi, D.; Hajishirzi, H. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. *arXiv* **2022**. [CrossRef]
20. Sidiropoulos, G.; Voskarides, N.; Vakulenko, S.; Kanoulas, E. Combining Lexical and Dense Retrieval for Computationally Efficient Multi-hop Question Answering. In Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing, Virtual, 10 November 2021; pp. 58–63. [CrossRef]
21. Ma, X.; Gong, Y.; He, P.; Zhao, H.; Duan, N. Query Rewriting for Retrieval-Augmented Large Language Models. *arXiv* **2023**. [CrossRef]
22. Asai, A.; Zhong, Z.; Chen, D.; Koh, P.W.; Zettlemoyer, L.; Hajishirzi, H.; Yih, W.-T. Reliable, Adaptable, and Attributable Language Models with Retrieval. *arXiv* **2024**. [CrossRef]
23. Hei, Z.; Liu, W.; Ou, W.; Qiao, J.; Jiao, J.; Song, G.; Tian, T.; Lin, Y. DR-RAG: Applying Dynamic Document Relevance to Retrieval-Augmented Generation for Question-Answering. *arXiv* **2024**. [CrossRef]
24. Yan, S.Q.; Gu, J.C.; Zhu, Y.; Ling, Z.H. Corrective Retrieval Augmented Generation. *arXiv* **2024**. [CrossRef]
25. He, X.; Tian, Y.; Sun, Y.; Chawla, N.V.; Laurent, T.; LeCun, Y.; Bresson, X.; Hooi, B. G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering. *arXiv* **2024**. [CrossRef]
26. Besta, M.; Kubicek, A.; Niggli, R.; Gerstenberger, R.; Weitzendorf, L.; Chi, M.; Iff, P.; Gajda, J.; Nyczyk, P.; Müller, J.; et al. Multi-Head RAG: Solving Multi-Aspect Problems with LLMs. *arXiv* **2024**. [CrossRef]
27. Hu, X.; Ru, D.; Qiu, L.; Guo, Q.; Zhang, T.; Xu, Y.; Luo, Y.; Liu, P.; Zhang, Y.; Zhang, Z. RefChecker: Reference-based Fine-grained Hallucination Checker and Benchmark for Large Language Models. *arXiv* **2024**. [CrossRef]
28. Dong, J.; Fatemi, B.; Perozzi, B.; Yang, L.F.; Tsitsulin, A. Don't Forget to Connect! Improving RAG with Graph-based Reranking. *arXiv* **2024**. [CrossRef]
29. Hitzler, P.; Krötzsch, M.; Rudolph, S.; Sure, Y. *Semantic Web: Grundlagen*; Springer: Berlin/Heidelberg, Germany, 2008.
30. Peng, C.; Xia, F.; Naseriparsa, M.; Osborne, F. Knowledge Graphs: Opportunities and Challenges. *Artif. Intell. Rev.* **2023**, *56*, 13071–13102. [CrossRef]
31. Saad-Falcon, J.; Barrow, J.; Siu, A.; Nenkova, A.; Yoon, D.S.; Rossi, R.A.; Dernoncourt, F. PDFTriage: Question Answering over Long, Structured Documents. *arXiv* **2023**. [CrossRef]
32. Edge, D.; Trinh, H.; Cheng, N.; Bradley, J.; Chao, A.; Mody, A.; Truitt, S.; Metropolitansky, D.; Ness, R.O.; Larson, J. From local to global: A graph rag approach to query-focused summarization. *arXiv* **2024**. [CrossRef]
33. Zhu, X.; Guo, X.; Cao, S.; Li, S.; Gong, J. StructuGraphRAG: Structured Document-Informed Knowledge Graphs for Retrieval-Augmented Generation. *Proc. Aaai Symp. Ser.* **2024**, *4*, 242–251. [CrossRef]
34. Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; Yu, P.S. A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 494–514. [CrossRef] [PubMed]
35. Song, Y.; Zhao, X.; Watson, R.T. Digitised knowledge-based literature reviewing: A tutorial on coding causal and process models as graphs. *J. Decis. Syst.* **2024**, *33*, 601–612. [CrossRef]
36. Birtic, M.; Senington, R.; Syberfeldt, A. Exploring Production System Knowledge Graph Applications Using a Simulation Framework. In *Sustainable Production Through Advanced Manufacturing, Intelligent Automation and Work Integrated Learning*; Andersson, J., Joshi, S., Malmsköld, L., Hanning, F., Eds.; Advances in Transdisciplinary Engineering; IOS Press: Amsterdam, The Netherlands, 2024. [CrossRef]
37. Tamašauskaitė, G.; Groth, P. Defining a Knowledge Graph Development Process Through a Systematic Review. *ACM Trans. Softw. Eng. Methodol.* **2023**, *27*, 1–40 [CrossRef]
38. Zhu, S.; Supryadi; Xu, S.; Sun, H.; Pan, L.; Cui, M.; Du, J.; Jin, R.; Branco, A.; Xiong, D. Multilingual Large Language Models: A Systematic Survey. *arXiv* **2024**. [CrossRef]
39. Zhu, X.; Xie, Y.; Liu, Y.; Li, Y.; Hu, W. Knowledge Graph-Guided Retrieval Augmented Generation. *arXiv* **2025**. [CrossRef]
40. Hu, Y.; Lei, Z.; Zhang, Z.; Pan, B.; Ling, C.; Zhao, L. Grag: Graph retrieval-augmented generation. *arXiv* **2024**. [CrossRef]
41. Alkouz, A.; Al-Saleh, M.I.; Alarabeyyat, A.; Bouchahma, M. UKRAG: A Unified Knowledge Graph to Enhance Retrieval Augmented Generation Performance. In Proceedings of the International Symposium on Intelligent Computing Systems, Sharjah, United Arab Emirates, 6–7 November 2024; Springer: Cham, Switzerland, 2024; pp. 1–19.
42. Xu, Z.; Cruz, M.J.; Guevara, M.; Wang, T.; Deshpande, M.; Wang, X.; Li, Z. Retrieval-augmented generation with knowledge graphs for customer service question answering. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, Washington, DC, USA, 14–18 July 2024; pp. 2905–2909.

43. Bahr, L.; Wehner, C.; Wewerka, J.; Bittencourt, J.; Schmid, U.; Daub, R. Knowledge graph enhanced retrieval-augmented generation for failure mode and effects analysis. *J. Ind. Inf. Integr.* **2025**, *45*, 100807. [CrossRef]
44. Liu, R.; Jiang, H.; Yan, X.; Tang, B.; Li, J. PolyG: Effective and Efficient GraphRAG with Adaptive Graph Traversal. *arXiv* **2025**. [CrossRef]
45. Bytyci, A.; Ramosaj, L.; Bytyci, E. Review of automatic and semi-automatic creation of knowledge graphs from structured and unstructured data. In Proceedings of the RTA-CSIT, Tirana, Albania, 26–27 April 2023; pp. 72–79.
46. Li, B.; Fan, S.; Zhu, S.; Wen, L. CoLE: A collaborative legal expert prompting framework for large language models in law. *Knowl.-Based Syst.* **2025**, *311*, 113052. [CrossRef]
47. Xiao, Y.; Zheng, S.; Shi, J.; Du, X.; Hong, J. Knowledge graph-based manufacturing process planning: A state-of-the-art review. *J. Manuf. Syst.* **2023**, *70*, 417–435. [CrossRef]
48. Buchgeher, G.; Gabauer, D.; Martinez-Gil, J.; Ehrlinger, L. Knowledge Graphs in Manufacturing and Production: A Systematic Literature Review. *IEEE Access* **2021**, *9*, 55537–55554. [CrossRef]
49. Wang, M.; Chen, L.; Fu, C.; Liao, S.; Zhang, X.; Wu, B.; Yu, H.; Xu, N.; Zhang, L.; Luo, R.; et al. Leave No Document Behind: Benchmarking Long-Context LLMs with Extended Multi-Doc QA. *arXiv* **2024**. [CrossRef]
50. Choi, H.; Jeong, J. Domain-Specific Manufacturing Analytics Framework: An Integrated Architecture with Retrieval-Augmented Generation and Ollama-Based Models for Manufacturing Execution Systems Environments. *Processes* **2025**, *13*, 670. [CrossRef]
51. Chen, T.; Wang, H.; Chen, S.; Yu, W.; Ma, K.; Zhao, X.; Zhang, H.; Yu, D. Dense X Retrieval: What Retrieval Granularity Should We Use? *arXiv* **2023**. [CrossRef]
52. Heredia, J.A.; Barreda, J.G. An Advanced Retrieval-Augmented Generation System for Manufacturing Quality Control. *SSRN Electron. J.* **2024**, *64*, 103007. [CrossRef]
53. Besta, M.; Memedi, F.; Zhang, Z.; Gerstenberger, R.; Piao, G.; Blach, N.; Nyczyk, P.; Copik, M.; Kwaśniewski, G.; Müller, J.; et al. Demystifying Chains, Trees, and Graphs of Thoughts. *arXiv* **2024**. [CrossRef]
54. Tuunanen, T.; Winter, R.; vom Brocke, J. Dealing with Complexity in Design Science Research: A Methodology Using Design Echelons. *MIS Q.* **2024**, *48*, 427–458. [CrossRef]
55. Gregor, S.; Hevner, A.R. Positioning and Presenting Design Science Research for Maximum Impact. *MIS Q.* **2013**, *37*, 337–355. [CrossRef]
56. Hevner, A.R.; March, S.T.; Park, J.; Ram, S. Design Science in Information Systems Research. *MIS Q.* **2004**, *28*, 75. [CrossRef]
57. Peffers, K.; Tuunanen, T.; Rothenberger, M.A.; Chatterjee, S. A Design Science Research Methodology for Information Systems Research. *J. Manag. Inf. Syst.* **2007**, *24*, 45–77. [CrossRef]
58. Sonnenberg, C.; vom Brocke, J. Evaluations in the Science of the Artificial–Reconsidering the Build-Evaluate Pattern in Design Science Research. In *Design Science Research in Information Systems. Advances in Theory and Practice*; Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J.M., Mattern, F., Mitchell, J.C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., et al., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7286, pp. 381–397. [CrossRef]
59. Agrawal, G.; Kumarage, T.; Alghamdi, Z.; Liu, H. Can Knowledge Graphs Reduce Hallucinations in LLMs?: A Survey. *arXiv* **2023**. [CrossRef]
60. Li, Z.; Guo, Q.; Shao, J.; Song, L.; Bian, J.; Zhang, J.; Wang, R. Graph Neural Network Enhanced Retrieval for Question Answering of LLMs. *arXiv* **2024**. [CrossRef]
61. Poliakov, M.; Shvai, N. Multi-Meta-RAG: Improving RAG for Multi-Hop Queries using Database Filtering with LLM-Extracted Metadata. *arXiv* **2024**. [CrossRef]
62. Fang, J.; Meng, Z.; Macdonald, C. TRACE the Evidence: Constructing Knowledge-Grounded Reasoning Chains for Retrieval-Augmented Generation. *arXiv* **2024**. [CrossRef]
63. Ru, D.; Qiu, L.; Hu, X.; Zhang, T.; Shi, P.; Chang, S.; Jiayang, C.; Wang, C.; Sun, S.; Li, H.; et al. Ragchecker: A fine-grained framework for diagnosing retrieval-augmented generation. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 10–15 December 2024; Volume 37, pp. 21999–22027.
64. Amazon Web Services. Amazon Textract. 2025. Available online: https://docs.aws.amazon.com/textract (accessed on 15 February 2025).
65. Huang, Y.; Lv, T.; Cui, L.; Lu, Y.; Wei, F. Layoutlmv3: Pre-training for document ai with unified text and image masking. In Proceedings of the 30th ACM International Conference on Multimedia, Lisboa, Portugal, 10–14 October 2022; pp. 4083–4091.
66. Garncarek, Ł.; Powalski, R.; Stanisławek, T.; Topolski, B.; Halama, P.; Turski, M.; Graliński, F. LAMBERT: Layout-aware language modeling for information extraction. In Proceedings of the International Conference on Document Analysis and Recognition, Lausanne, Switzerland, 5–10 September 2021; Springer: Cham, Switzerland, 2021; pp. 532–547.
67. LangChain. LangChain Document Transformers—Data Connection. 2025. Available online: https://python.langchain.com/v0.1/docs/modules/data_connection/document_transformers/ (accessed on 15 February 2025).
68. Amazon Web Services, Inc. Amazon Titan Models. 2025. Available online: https://docs.aws.amazon.com/bedrock/latest/userguide/titan-models.html (accessed on 15 February 2025).

69. Chroma Inc. Chroma. 2025. Available online: https://docs.trychroma.com/ (accessed on 15 February 2025).
70. Cyganiak, R.; Wood, D.; Lanthaler, M. RDF 1.1 Concepts and Abstract Syntax. 2014. Available online: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/ (accessed on 15 February 2025).
71. Ontotext. What Is GraphDB?—GraphDB 10.6 Documentation. 2025. Available online: https://graphdb.ontotext.com/documentation/10.6/ (accessed on 15 February 2025).
72. Campos, R.; Mangaravite, V.; Pasquali, A. YAKE! Keyword Extraction from Single Documents using Multiple Local Features. *Inf. Sci.* **2020**, *509*, 257–289. [CrossRef]
73. Anthropic. Claude 3 Haiku. 2025. Available online: https://www.anthropic.com/news/claude-3-haiku (accessed on 15 February 2025).
74. Amazon Bedrock. 2025. Available online: https://aws.amazon.com/de/bedrock/ (accessed on 15 February 2025).
75. Anthropic. Claude 3.5 Sonnet. 2025. Available online: https://www.anthropic.com/news/claude-3-5-sonnet (accessed on 15 February 2025).
76. Topsakal, O.; Akinci, T.C. Creating Large Language Model Applications Utilizing Langchain: A Primer on Developing LLM Apps Fast. In Proceedings of the International Conference on Applied Engineering and Natural Sciences, Konya, Turkey, 10–12 July 2023; Volume 1, pp. 1050–1056.
77. Knollmeyer, S.; Caymazer, O.; Koval, L.; Akmal, M.; Asif, S.; Mathias, S.; Großmann, D. Benchmarking of Retrieval Augmented Generation: A Comprehensive Systematic Literature Review on Evaluation Dimensions, Evaluation Metrics and Datasets. In Proceedings of the 16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management. SCITEPRESS—Science and Technology Publications, Porto, Portugal, 17–19 November 2024; pp. 137–148. [CrossRef]
78. Rajpurkar, P.; Jia, R.; Liang, P. Know What You Don't Know: Unanswerable Questions for SQuAD. *arXiv* **2018**. [CrossRef]
79. Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W.W.; Salakhutdinov, R.; Manning, C.D. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. *arXiv* **2018**. [CrossRef]
80. Trivedi, H.; Balasubramanian, N.; Khot, T.; Sabharwal, A. MuSiQue: Multihop Questions via Single-hop Question Composition. *arXiv* **2021**. [CrossRef]
81. Zhou, Y.; Liu, Z.; Jin, J.; Nie, J.Y.; Dou, Z. Metacognitive retrieval-augmented large language models. In Proceedings of the ACM Web Conference 2024, Singapore, 13–17 May 2024; pp. 1453–1463. [CrossRef]
82. Feng, X.; Chen, Z.Y.; Qin, Y.; Lin, Y.; Chen, X.; Liu, Z.; Wen, J.R. Large Language Model-based Human-Agent Collaboration for Complex Task Solving. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, FL, USA, 12–16 November 2024; pp. 1336–1357. [CrossRef]
83. Es, S.; James, J.; Espinosa-Anke, L.; Schockaert, S. RAGAS: Automated Evaluation of Retrieval Augmented Generation. *arXiv* **2023**. [CrossRef]
84. RAGAS. Context Precision. Available online: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/context_precision/ (accessed on 12 March 2025).
85. RAGAS. Context Recall. Available online: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/context_recall/ (accessed on 12 March 2025).
86. Adlakha, V.; BehnamGhader, P.; Lu, X.H.; Meade, N.; Reddy, S. Evaluating Correctness and Faithfulness of Instruction-Following Models for Question Answering. *arXiv* **2023**. [CrossRef]
87. Rackauckas, Z.; Câmara, A.; Zavrel, J. Evaluating RAG-Fusion with RAGElo: An Automated Elo-based Framework. *arXiv* **2024**. [CrossRef]
88. Xu, C.; Wang, M.; Ren, Y.; Zhu, S. Enhancing Aspect-based Sentiment Analysis in Tourism Using Large Language Models and Positional Information. *arXiv* **2024**. [CrossRef]
89. Jaech, A.; Kalai, A.; Lerer, A.; Richardson, A.; El-Kishky, A.; Low, A.; Helyar, A.; Madry, A.; Beutel, A.; Carney, A.; et al. Openai o1 system card. *arXiv* **2024**. [CrossRef]
90. Sun, H.; Jin, R.; Xu, S.; Pan, L.; Cui, M.; Du, J.; Lei, Y.; Yang, L.; Shi, L.; Xiao, J.; et al. FuxiTranyu: A Multilingual Large Language Model Trained with Balanced Data. *arXiv* **2024**. [CrossRef]