Graph RAG in the Wild: Insights and Best Practices from Real-World Applications [1]

Diego Collarana [a,b,*], Christopher Ingo Pack [a], Yan-Ying Liao [a], Marlena Flüh [a], Jonathan Lehmkuhl [a], Abhishek Nageri [a], Alexander Graß [a,c], Moritz Busch [a], Prinon Das [c], Lara Dingels [c], Stefan Decker [a,c] and Christian Beecks [a,d]

[a] *Data Science and Artificial Intelligence, Fraunhofer FIT, Sankt Augustin, Germany*
E-mails: diego.collarana.vargas@fit.fraunhofer.de, christopher.ingo.pack@fit.fraunhofer.de, yan-ying.liao@fit.fraunhofer.de, marlena.flueh@fit.fraunhofer.de, jonathan.lehmkuhl@fit.fraunhofer.de, abhishek.nadgeri@fit.fraunhofer.de, moritz.busch@fit.fraunhofer.de, christian.beecks@fit.fraunhofer.de, stefan.decker@fit.fraunhofer.de
[b] *Institute for Computational Intelligence, Universidad Privada Boliviana, Cochabamba, Bolivia*
[c] *Information Systems & Databases, RWTH Aachen University, Aachen, Germany*
E-mails: prinon.das@rwth-aachen.de, alexander.grass@rwth-aachen.de, lara.dingels@rwth-aachen.de
[d] *Chair of Data Science, FernUniversität in Hagen, Hagen, Germany*

Abstract. Retrieval-Augmented Generation (RAG) enhances Large Language Models (LLMs) by enabling access to external knowledge without retraining. While effective, traditional RAG methods—typically reliant on vector-based retrieval—face limitations in understanding complex semantics, connecting dispersed information, and supporting user-centric search workflows. Graph Retrieval-Augmented Generation (Graph RAG) addresses these challenges by incorporating knowledge graphs into the retrieval process, enabling semantically enriched and structured query handling. This paper presents the application of Graph RAG across seven real-world applications, including legal compliance, customer support, enterprise knowledge management, finance, education, data protection enforcement, and time series analytics. For each application, we outline the distinct challenges, solutions, and design decisions made. In addition, we introduce a modular Graph RAG Engine, named FRAG-KEDA, to support ingestion, graph construction, hybrid retrieval, and LLM orchestration. We present empirical evidence demonstrating improvements in accuracy, latency, and user trust. Additionally, we address cross-domain challenges, including graph drift and evaluation strategies.

Keywords: Graph RAG, Retrieval-Augmented Generation, Knowledge Graph, Large Language Model

1. Introduction

Retrieval-Augmented Generation (RAG) is an established technique [23] for enhancing the capabilities of Large Language Models (LLMs). RAG enables LLMs to access external knowledge without requiring costly retraining. By using an external knowledge base, RAG refines LLM outputs, helping mitigate issues such as hallucinations, limited domain-specific knowledge, and outdated information. However, traditional RAG approaches [46], which rely on retrieving text fragments based on vector similarity, struggle to connect scattered information, understand complex semantic concepts in large datasets, and answer questions that require a holistic understanding of the information [1, 104]. Furthermore, contemporary RAG architectures are often limited to document-centric applications and do not fully cover a user's typical search process.

Graph Retrieval-Augmented Generation (Graph RAG) [32] has emerged as a novel solution to overcome the limitations of traditional RAG. Graph RAG improves on RAG by integrating knowledge graphs into the retrieval

[1] D. Collarana, C.I. Pack, and Y-Y. Liao contributed equally.
[*] Corresponding author E-mail: diego.collarana.vargas@fit.fraunhofer.de.

1570-0844/$35.00 © 0 – IOS Press. All rights reserved.

process. Instead of relying solely on semantic similarity between queries and text fragments, Graph RAG leverages the structured nature of knowledge graphs, where data are organized as nodes (entities) and relationships between them. By constructing knowledge graphs from the data and querying them, Graph RAG can retrieve relevant information more efficiently and accurately [32, 78], thereby providing LLM-based systems with a semantically enriched context for query processing. Graphs are well-suited to connecting different pieces of information through their relationships, making them ideal for performing complex analytical tasks, unlike flat, unconnected data [32]. This paper reports the application of Graph RAG in seven real-world scenarios, which span legal compliance, consumer support, enterprise knowledge management, education, data protection enforcement, finance, and time series analytics. Table 1 presents a summary of the applications.

Application Name	**Domain(s)**	**Type / Schema**	**Why Graph RAG?**
EU Regulation QA Agent (4.1)	Legal/ Regulatory compliance	Schemaless	**Multi-hop Reasoning**: Traverses links between disparate regulatory articles that are not semantically similar.
Car Owners Manual Assistant (4.2	Automotive / Technical documentation	Schemaless	**Precise Grounding**: Uses graph edges to map specific user intents to exact page coordinates.
Wenn Fraunhofer Wüsste Was Fraunhofer Weiß (4.3)	Enterprise knowledge / Research management	Schemaless	**Hidden Link Discovery**: Identifies non-obvious relationships between researchers across 72 institutes.
Artificial Buddy for Effective Learning (5.1)	Education / Training	Schema First / Self-made vocabulary	**Causal Pedagogy:** Structures learning materials by "prerequisite" relations.
GDPR Compliance Verifier (5.2)	Legal / Data protection / Compliance	Schema First / DPV vocabulary	**Legal Provenance**: Maps raw data to the DPV-GDPR vocabulary for standardized reasoning.
Financial Report Generator (5.3)	Finance / Financial analysis	Schema First / Self-made vocabulary	**Context Preservation:** Prevents "lost-in-the-middle" errors by pinning KPIs to a structured schema.
Data Science Agents (5.4)	Data Science / Time series analysis	Schema First / SemTS ontology	**Numeric-to-Symbolic Bridge**: Converts raw time-series data into a semantic graph, allowing natural language querying of numerical trends.

Table 1

Application Overview: In this paper, we present the application of Graph RAG in seven applications from different domains, from Finance to Data Analytics, explaining the important role of Graph RAG. We classify applications into two types: Schemaless and Schema-First.

The aforementioned applications were developed in collaboration with industry and research partners, and each presents unique technical requirements that dictate specific Graph RAG architectures. For legal compliance, the system must support high-dimensional reasoning across dense regulatory frameworks such as DORA. The automotive application requires a hierarchical graph structure for document grounding, while enterprise knowledge management requires strong cross-domain entity linking to bridge institutional silos. The educational and data protection applications leverage schemas to enforce pedagogical and legal constraints. In finance, the focus shifts to preserving numerical context through KPI-centric modeling. Lastly, the data science agents use symbolic-numeric translation via the Semantic Time Series Ontology [29] to enable natural-language interaction with raw time-series datasets.

To efficiently implement and evaluate these diverse applications, we develop a unified Graph RAG Engine, named FRAG-KEDA[2]. This engine serves as the standardized technological baseline for all applications, ensuring that advancements in graph-based retrieval remain reusable and scalable. FRAG-KEDA implements a modular reference architecture that manages the core lifecycle of a Graph RAG system: ingestion, automated graph construction, and hybrid retrieval orchestration.

[2]FRAG derived from the German imperative of *Fragen* and KEDA being the acronym of the research group *Knowledge-enhanced Data Analytics*

2. The rise of Graph RAG

Graph RAG represents the synergy of several key advancements in natural language processing and knowledge representation. In the following sections, we trace its evolution from the foundation of Large Language Models, through traditional Retrieval-Augmented Generation, to the integration of graph-based retrieval mechanisms.

2.1. Large Language Models

Large Language Models (LLMs) have led to a paradigm shift in Natural Language Processing [2, 110]. These models are based on the transformer architecture [91], which is characterized by a large number of parameters and extensive training on large datasets. LLMs are capable of learning generalized text representations, facilitating their adaptation to diverse downstream tasks [110]. The **attention mechanism** within the transformer architecture enables the model to capture relationships among input words. There are three primary types of transformers: (i) *Encoder models* use bi-directional attention to model the relation of each input token to all tokens. (ii) *Decoder models* or autoregressive models use masked attention, so that each token can only be modeled in relation to all previous tokens up to itself. (iii) *Encoder-Decoder models* use both of these types, combining them by feeding the encoder's result into the decoder. Since the transformer model lacks recurrent or convolutional capabilities, it employs positional encoding to represent the order of tokens in the input sequence. The transformer model processes data as tokens, which are numerical representations of letter sequences. A tokenizer generates a finite vocabulary of recognized letter sequences, allowing the transformer to identify each as a single token.

LLMs were first developed by OpenAI, with GPT-2 [68] and later scaled to GPT-3 [7], followed by the release of ChatGPT [59] and GPT-4 [60]. Due to their model sizes, LLMs have several capabilities beyond mastering text processing, known as emergent capabilities. This includes general pattern recognition, tool use, and reasoning [8, 54, 67, 95]. Additionally, LLMs are adaptable to multiple specific use cases via fine-tuning [38]. LLMs can be divided into two types: Dense models use all their parameters for both training and inference. This is how transformer models perform inference. This architecture is primarily used for open weights and smaller models, such as the LLama series of models [88, 89], or smaller DeepSeek-R1 [15] models. Mixture-of-Experts [14], or sparse models, train all their weights but only use a subset of these weights during inference. As the name suggests, experts are activated in response to requests, reducing inference time without significant loss of quality.

Reasoning models [15] are fine-tuned LLM models that focus on generating long, thoughtful responses or chain-of-thought outputs to analyze requests better and accomplish tasks. Reasoning models can be both dense and sparse. These models have the most expensive inference, as they tend to 'think' about any given input for a long time before reaching a definitive answer, producing lengthy outputs. These models have achieved good results on many difficult benchmarks but can be overqualified for simpler tasks, where 'normal' LLMs often outperform them.

Limitations of LLMs: Despite their remarkable emergent capabilities, LLMs still have inherent limitations. Because LLMs are never retrained, their knowledge is limited to the information on which they were initially trained [31, 44]. LLMs also suffer from a language barrier. Most datasets for pretraining large language models (LLMs) are in English. This impairs LLMs' ability to process other languages [61, 79, 97]. In addition, LLM tokenizers optimize their tokens by quantity, meaning that non-English languages also need more tokens to represent texts [43, 77]. The most significant limitation of LLMs is hallucination. Because pretraining only predicts the next token in the text, LLMs have no inherent connection to real-world truth beyond the fact that the correct answer is often the most likely next token. As a result, LLMs typically generate text that reads well and sounds accurate, but can sometimes contain factual errors [47, 100]. In addition, the inherent nature of LLMs' knowledge makes it more challenging to explain their outputs, which is related to the black-box issue in deep learning models [50, 106].

2.2. Retrieval-Augmented Generation (Traditional RAG)

To address the limitations of LLMs discussed in Section 2.1, Retrieval-Augmented Generation (RAG) has been proposed [46]. In RAG, the retrievable component functions as a non-parametric memory (source) that is regularly updated, allowing it to store extensive information and incorporate confidential data [20, 23, 108]. A typical RAG

process consists of two steps (i) *Retrieval* and (ii) *Generation*. In the retrieval step, given an input, the retrieval mechanism scouts and identifies the most relevant source, assuming the source is indexed into smaller units. The retrieval step has become increasingly complex due to factors such as retriever type, retriever granularity, and the nature of enhancements (pre-retrieval and post-retrieval) [20]. For simplicity, we classify RAG into two types: (i) *Naive RAG* and (ii) *Advanced RAG*.

Naive RAG employs a straightforward process that begins with indexing sources, followed by retrieval and generation conditioned on the input question [46]. Each chunk is encoded into a vector representation using an embedding model and stored in a vector database. When a user query is received, it is also converted into a vector representation using the same embedding model. Retrieval is performed by calculating similarity scores between the query vector and the stored chunk embeddings. The top K chunks with the highest similarity scores are selected and used as expanded context for generation. In the generation stage, the retrieved chunks and the original user query are combined into a coherent prompt, which is then used as input to an LLM to produce an answer. While effective, naive RAG approaches have limitations in retrieval accuracy and generation quality [23], which advanced RAG methodologies address.

Although the overall structure of *Advanced RAG* is similar to that of Naive RAG, it introduces enhancements to each core component. To overcome indexing limitations, *Advanced RAG* employs refined techniques, including sliding-window approaches, fine-grained segmentation, and metadata integration [11, 23]. Retrieval quality, a primary bottleneck in Naive RAG, is improved through both pre-retrieval and post-retrieval strategies. The **pre-retrieval** stage focuses on optimizing indexing structures and query formulation by refining data granularity, optimizing indices, augmenting metadata, improving alignment, and utilizing mixed retrieval methods [75, 109, 111]. Since the initial ranking of retrieved chunks may be suboptimal, the **post-retrieval** stage emphasizes re-ranking methods to ensure the most relevant chunks are prioritized within the context prompt [94, 103]. Furthermore, because retrieved chunks may exceed LLMs' context-length constraints, *Advanced RAG* emphasizes selecting essential information, highlighting critical sections, and reducing context length [23, 92].

Limitations of Traditional RAG: Despite the success of RAG in various applications [20, 80, 108] and its significant role in enhancing LLM capabilities, as discussed in Section 2.2, traditional RAG systems continue to face the following critical limitations.

1. *Multi-hop Query*: Traditional RAG systems predominantly utilize dense vector-based retrieval [46]. Although this method is effective for simple queries that require extracting information from a single passage, its performance declines when synthesizing responses from multiple sources is necessary. For example, in the legal compliance scenario described in Section 4.1, a typical user query requires retrieving multiple relevant sections and inferring incomplete information and causal relationships among documents.
2. *Contextual Misalignment*: Retrieval in RAG often acts as a bottleneck, restricting the information available to the LLM and leading to disjointed, redundant responses [1, 23]. For example, in the application discussed in Section 5.1, where a query seeks to understand the economic concept of *Effect of trade policies in Country A for Country B*, traditional RAG may retrieve isolated passages about each country's trade policies but fail to capture causal relationships or intermediary factors. The inherent limitation stems from standard dense retrieval methods, which prioritize dot-product similarity over conceptual coherence, leading the LLM to generate incomplete responses and increasing the risk of hallucinations.
3. *Reliability Challenges*: In real-world applications, user interactions with data are frequently complex and difficult to replicate accurately in controlled evaluation environments. For instance, as discussed in Section 4.2, a customer service chatbot trained on car owners' manuals may perform well during evaluation but fail to retrieve accurate troubleshooting information when confronted with non-standard user queries. A user may describe an issue as *The car is making a weird noise when I turn the wheel*, rather than using the owner's manual phrasing of *Problem due to low power steering fluid*.

Graph Retrieval-Augmented Generation (Graph RAG) [32] has emerged as a novel solution to overcome the limitations of traditional RAG. Graph RAG has been shown to outperform traditional RAG approaches across many applications and domains. In Xu et al.'s work [99], the customer support question answering task employed a two-level graph architecture. It integrates Intra-Issue Trees for modeling ticket content in fine detail and Inter-Issue Graphs for

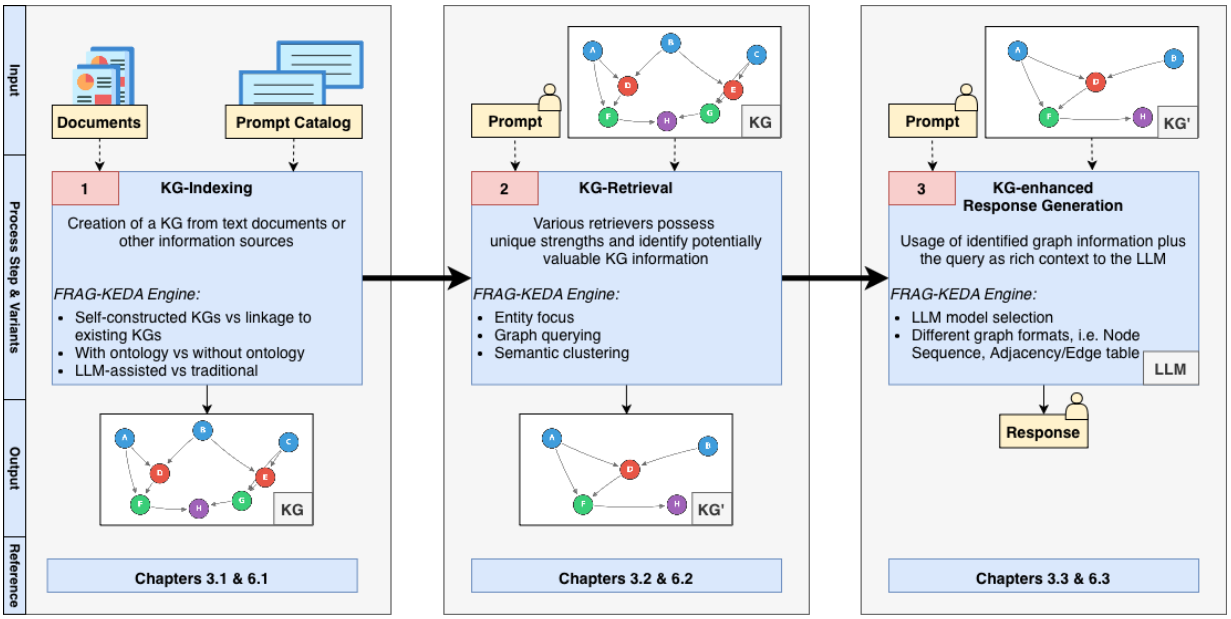


Fig. 1. We divide our FRAG-KEDA engine into three main stages. First, KG-Indexing prepares the graph database by using existing KGs or building one from text corpora, thereby creating indexes for efficient search. Next, using various retrievers, the KG-Retrieval stage extracts relevant context from the graph, such as triplets or subgraphs, in response to the user's query. After refining the retrieved results, this data is transformed into formats compatible with LLM generation, which is integrated into the KG-Generation stage to produce the final response.

connecting tickets based on both explicit and implicit relationships. By combining semantic similarity-based search with LLM-based subgraph extraction, the system effectively retrieves the correct subgraphs. Notably, the method has been adopted by LinkedIn's customer support team and has yielded substantial reductions in issue-resolution time. In the biomedical domain, rather than constructing knowledge graphs from scratch, some applications directly use open-source medical knowledge graphs (KGs) such as CMeKG, CPubMed-KG, and Disease-KG in Graph RAG systems. For instance, HyKGE [42] leverages existing graphs as high-quality fact sources within a graph-based generation and retrieval pipeline to enhance output accuracy and credibility. In this paper, we report our experience applying Graph RAG across seven domains and discuss its benefits and impact.

3. FRAG-KEDA Engine

Knowledge Graphs (KGs) serve as an alternative to commonly used vector databases [62] for storing factual knowledge. Building on the framework presented by Peng et al. [64], this work introduces a modular and extendable Graph RAG engine, FRAG-KEDA, which consists of three stages: (i) *Graph-based Indexing*, (ii) *Graph-guided Retrieval*, and (iii) *Graph-enhanced Generation*, (cf. Figure 1). The engine interfaces with existing libraries and tools, standardizes KG formats, and integrates inputs and outputs into a coherent set of configurable components. The upcoming subsections describe each stage, relevant design decisions, and adaptations.

3.1. Graph-based Indexing (KG-Indexing):

The primary objective of the initial stage is to construct a knowledge graph and index its embedding representation to facilitate efficient search. The engine incorporates modules that can either generate the KG from scratch (self-construction) or connect to an existing KG. Self-construction entails populating the KG with entities and relations extracted from a text corpus, such as EU regulations. Alternatively, an existing knowledge graph, such as one based on the Data Protection vocabulary, may be used. The selection between these approaches depends on factors such as the availability of a suitable KG for Graph RAG adaptation. Another important consideration is whether to

implement a fixed schema for entities and relations using transformation scripts or to permit dynamic extraction of entity and relation types (schemaless), typically performed by a large language model (LLM). Data quality is essential in this context. KGs constructed from reliable, structured metadata sources offer a robust foundation for knowledge-based text generation. In contrast, KGs automatically generated from text using LLMs are more error-prone, though they remain valuable for providing context during generation.

Constructing an effective KG from a text corpus traditionally relies on methods such as information extraction (IE) [18], entity linking [13], and relation extraction [53]. These techniques extract structured data from unstructured text, thereby facilitating the development of KGs. The advent of LLMs has enabled end-to-end approaches for building KGs. FRAG-KEDA, current implementation, employs the Microsoft GraphRAG library to construct scenarios. As Microsoft GraphRAG does not generate an RDF KG, the resulting artifacts are transformed into an RDF graph while preserving provenance. For integration with existing KGs, the engine connects to domain-specific tools or scripts that create the RDF graph.

The next design decision concerns whether to guide the LLM-based KG creation process using an ontology. For applications where reliability, precision, structural coherence, and explainable reasoning are essential, employing a well-defined, human-validated ontology as the KG's foundational schema is strongly recommended. Advanced LLMs can assist in KG construction by automatically extracting the semantics of entities and relations. Therefore, this decision involves balancing the initial effort and cost of developing and maintaining an ontology against the long-term benefits of improved quality and trustworthiness.

In contrast to traditional RAG, which stores embedded text chunks, Graph RAG embeds structured knowledge as triples in the subject-predicate-object (SPO) format. The third design decision concerns the preparation of node properties to support the LLM, specifically the embedding representation of entities and relations from the KG. During the KG-Indexing process, node properties are prepared to enrich the LLM's context. The following assisted fields are defined: (1) Entity Summary, (2) Relations Summary, and (3) Community Summary.

3.2. Graph-guided Retrieval (KG-Retrieval):

After the graph is constructed, the next step is to retrieve pertinent information from the knowledge graph to address the input query. This step is pivotal in Graph RAG systems, as it determines the quality and relevance of the generated responses. Since the introduction of Graph RAG, numerous retrieval methods have been proposed. Among them, our engine provides the following approaches:

1. **Entity Focus retrieval**: This method retrieves relevant nodes or subgraphs based on lexical or semantic similarity between the input query and components of the knowledge graph. It commonly leverages embedding models (e.g., sentence transformers) and similarity metrics (e.g., cosine similarity) to align query intents with semantically related entities or triples [36, 39, 98]. Alternatively, some approaches adopt keyword-based matching [51, 84], a simple yet effective technique for identifying anchor entities within the graph. Once these anchor points are identified, traditional graph traversal algorithms, such as depth-first search (DFS), can be employed to explore local neighborhoods and generate candidate subgraphs for downstream processing.
2. **Graph Querying retrieval:** A unique technique to Graph RAG involves the automatic translation of user queries, initially posed in natural language, into formal, structured graph query languages such as SPARQL or Cypher. An example is Fact Finder [81], which uses LLMs to translate natural language questions into executable Cypher queries. These queries are run against a graph database to retrieve precise subgraph information, combining LLMs reasoning with the power of structured graph querying. It is also possible to use a set of predefined advanced queries linked to the KG. The goal is then to execute them against the underlying KG to retrieve relevant subgraphs, entities, relationships, or aggregated data.
3. **Semantic Clustering retrieval:** This approach leverages LLMs' robust understanding, interpretation, and processing capabilities in natural language to dynamically guide the retrieval process. For example, Graph RAG [16] uses an LLM to generate intermediate responses from prebuilt community summaries, with each point rated for importance. Subsequently, the top-rated points are aggregated to inform the development of the final response. A requirement for this style of retrieval is a preexisting knowledge graph that contains at least some of the entities relevant to the task at hand [56, 66, 96]. Based on these graph-entities, subgraphs are

compiled, for example, through k-hop-neighborhood retrieval [56, 96], or the retrieval of relevant ancestor-nodes [66]. These subgraphs serve as the basis for the additional information provided to the LLM. The exact characteristics of this additional input depend on the specific approaches, as intermediate processing steps might be applied to the subgraphs before they are passed to the LLM as a whole or iteratively, part by part.

3.3. Graph-enhanced Generation (KG-Generation)

After gathering relevant information from the KG, this stage integrates the retrieved data into the generation process. A general approach is to convert the retrieved graph data and the input query into text, and then feed the resulting text into the LLM for response generation. As prompt format can significantly affect LLM performance [35, 76], it is crucial to design prompts carefully to guide the model toward generating faithful, coherent, and context-sensitive outputs. This includes keeping textual representations concise to avoid overloading the input and mitigating the "lost in the middle" phenomenon [48], in which models tend to overlook information in the middle of long contexts. Peng et al. [64] classify graph formats into five categories: (i) Adjacency/Edge Table, (ii) Natural Language, (iii) Node Sequence, (iv) Code-like Forms, and (v) Syntax Tree. Several studies exemplify the use of these formats in graph-based generation tasks. For instance, Fatemi et al. [21] use adjacency tables and edge tags to represent entity connections, which facilitates structural reasoning. Ye et al. [101] convert nodes, edges, and their relational structure into human-readable natural language descriptions. Guo et al. [30] explore how LLMs interpret code-like graph formats, such as Graph Markup Language (GraphML) [6]. These formats vary in their expressiveness, which affects how well large language models (LLMs) can understand and generate information from graph-structured data. This underscores the importance of format selection in graph-enhanced language modeling.

Effective KG generation hinges on three interlocking design decisions. First, the model selection and prompt engineering must align with the application's goal, e.g., multiple-choice QA versus open-ended text production and language dependency. Second, graph serialization balances completeness, brevity, and LLM interpretability. For example, natural-language renderings preserve semantic nuance, whereas code-like formats leverage the model's code-comprehension capabilities. The chosen representation directly affects context-window usage and the risk of information loss. Finally, generation quality can be improved through staged enhancements: pre-generation tactics (e.g., deduplication, salience-based pruning) refine the input; mid-generation controls (e.g., constrained decoding or multi-step prompting) steer reasoning paths; and post-generation aggregation reconciles diverse outputs. These implementation choices establish a principled workflow for reliable, graph-aware natural-language generation.

4. Schemaless Applications

In some applications, either no ontology exists, or building one is constrained by time. Thus, classes and relations are extracted during KG indexing using Prompt Engineering. We present applications in which the FRAG-KEDA engine operates without an available ontology. Table 2 presents the summary of schemaless applications.

4.1. EU Regulation Question and Answering Agent (CRR)

Motivation: The complex regulatory landscape in the EU poses significant challenges for financial institutions. For instance, all banks are required to implement the EU Digital Operational Resilience Act (DORA) by 2025[3]. Traditional question-answering systems often struggle to retain structured information and fail to address the logical complexities of these regulations [12]. As a result, institutions face complex, multifaceted issues that require both legal and technical expertise. Consider the following question: *"Can the risk weight of 65% also be applied to companies in the financial sector that are to be reported in the exposure class 'corporates'?"*. To answer this question, QA systems require an understanding of how different components of a regulation fit together. The answer depends on how companies are classified and whether the regulation includes or excludes them from this specific rule. All the necessary information is available in the legal text, but it is distributed across different sections rather

[3] https://www.eiopa.europa.eu/digital-operational-resilience-act-dora_en, Apr, 2025

Application	Example Question	Main Entity Types	Impact
CRR	What are the ICT incident reporting obligations for financial entities under DORA, and how do they specifically impact the capital requirements defined in the CRR?	Regulation, Article, Requirement, Financial Entity, Obligation	Reduction of Regulatory Risk: Potential to automate large-scale compliance audits across interconnected legal frameworks.
CarAssist	If the ID.3 displays a yellow battery warning, what specific safety steps are required in the engine compartment before I can check the fuses?	Vehicle Model, Component, Warning Light, Procedure, Manual Section	Enhanced Safety: Zero-hallucination support in high-stakes maintenance environments.
WFW	Which researchers in the Fraunhofer network have worked on 'Graph RAG' and also have a shared project history with the 'SustainAI' initiative?	Researcher, Institute, Project, Publication, Technology, Finding	Accelerated Innovation: Breaking down silos and preventing redundant research.

Table 2

Schemaless Applications: In the paper, we present the use of Graph RAG in seven applications from different domains, from Finance to Data Analytics, explaining the important role of Graph RAG. We classify applications into two types: Schemaless and Schema-First.

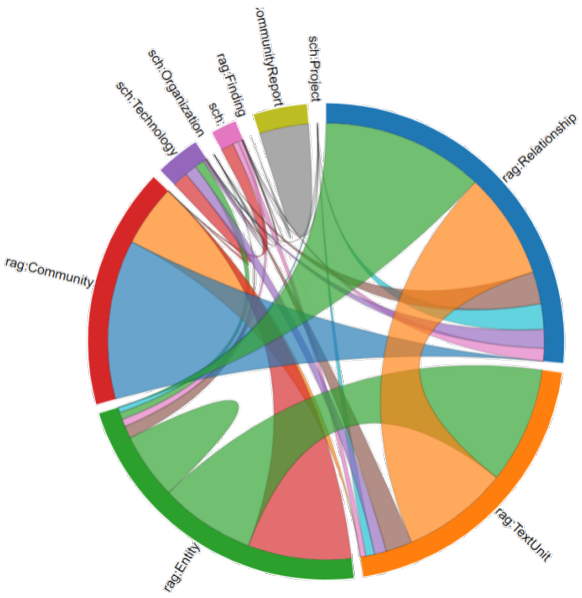
than stated in a single paragraph. Traditional RAG fails to answer such questions because, rather than connecting the dots, it simply retrieves risk weights from the regulation. In collaboration with a major financial consulting firm, we evaluate FRAG-KEDA's capability to connect relevant concepts and implicit rules to generate complete answers.

Engine Configuration: To construct the knowledge graph, we selected four key regulatory texts critical to banking compliance in the EU: CRR, CRR II, CRD IV, and CRD V. We apply a self-constructed, without ontology, LLM-assisted method to build the graph. We processed these documents using the open-source, instruction-tuned LLM Gemma-2-27B, guided by detailed prompts to extract structured knowledge. The hierarchical chunking technique was employed to segment the regulatory text into contiguous, consistent pieces. The method was applied to enhance naive token-based chunking, which is known to yield disjoined information. Once chunked, an LLM served as the knowledge extractor, generating triplets in the form of (subject, predicate, object). The result is an attributed knowledge graph in which each entity and relation is enriched with a concise description generated by the LLM, based on their unique attributes and semantic roles within the regulation. After populating the KG with entities and relations, we also identify communities that create summaries from them.

For **knowledge retrieval**, we operate on the constructed knowledge graph and support two specialized retrieval modes: (1) Similarity-based retrieval, designed for entity-specific queries. This mode navigates the graph to retrieve information about particular entities and their direct relationships. (2) Semantic Clustering aimed at addressing broader, more complex queries; this mode synthesizes information from pre-generated community reports. In contrast, the traditional RAG module focuses on straightforward queries, which can be addressed effectively via dense retrieval to retrieve semantically relevant chunks from the vector store.

For **answer generation**, the EU Regulation Agent first classifies each query and routes it to the appropriate Graph RAG retrieval mode. Once relevant subgraphs or community reports are retrieved, the system generates an answer by composing information from the underlying graph, explicitly referencing the involved entities and relations. A validation module then checks the result against thresholds for faithfulness and relevance. If the answer does not meet quality standards, the system automatically reroutes the query to an alternative retrieval path. This iterative, self-correcting process continues until a satisfactory answer is produced or all retrieval options are exhausted.

Evaluation and Impact: To evaluate the performance of our Graph RAG solution, we use the LLM-As-A-Judge method. We use RAGAS for automatically generating a test dataset. We evaluate the Graph RAG versus Traditional RAG using three metrics: (1) **Comprehensiveness**, i.e, how much detail does the answer provide to cover all aspects and details of the question. (2) **Diversity**, i.e., how varied and rich is the answer in providing different perspectives and insights on the question. (3) **Empowerment**, i.e., how well does the answer help the reader understand and make informed judgments about the topic.



(a) Entities extracted in the WFW application

(b) CRR application user interface

Fig. 2. (a) In the schemaless applications, LLMs create the KG using prompts. (b) UI is standard across all applications that show the generated answer, and the subgraph used for generation (provenance) is also shown.

Table 3 shows the results. Graph RAG outperforms Traditional RAG in all metrics. However, these results should be interpreted with caution, as the LLM-As-A-Judge is sensitive to various configurations, including prompting. We discuss this limitation in Subsection 7.1. ***Impact:*** The results of this application were extended into a product by our partner as an Agentic Hybrid RAG solution. The EU Regulation Agent intelligently controls the retrieval process and compensates for the limitations of traditional vector-based RAG systems. The agent is tasked with classifying user queries and selecting the appropriate retrieval strategy—either vector-based RAG or one of the Graph RAG modes (entity focus or semantic clustering). It also includes an iterative self-correction mechanism based on two evaluation metrics: answer faithfulness and relevance. If the generated response does not meet the desired thresholds for these metrics, the agent will automatically rework the question via alternative retrieval paths until a satisfactory answer is obtained. This approach ensures an optimal balance among cost, performance, and the high precision required for regulatory compliance.

4.2. Car Owners Manual Assistant (CarAssist)

Motivation: Modern owner manuals typically exceed 300 pages. Survey data indicate that drivers require approximately seven minutes to locate a single maintenance instruction [24]. Critical safety warnings, such as airbag deployment, jack points, and high-voltage warnings, are distributed across multiple chapters, creating “information fragmentation” that hampers rapid retrieval in roadside or workshop scenarios. Keyword search fails when the terminology varies, such as between "central locking" and "power door lock". It also fails when users ask procedural questions rather than exact phrase matches, like "How do I reset the TPMS?" Traditional RAG systems designed for manual documentation struggle to handle highly complex and ambiguous queries [70]. In this application, we develop an assistant that answers natural-language questions via a voice interface. Graph RAG grounds questions in structured entities, including Vehicle Parts, Warnings, and Procedures. This capability enables the assistant to reason over part hierarchies (e.g., wheel → tire → TPMS sensor) and manage prerequisite relationships, such as the requirement to remove the battery cover before disconnecting the negative terminal.

	EU Regulation Question and Answering Agent						Car Owners Manual Assistant						Wenn Fraunhofer Wüsste - Was Fraunhofer Weiß					
	Entity Retrieval			Cluster Retrieval			EF Retrieval			Clus			EF Retrieval			Global		
Metric	**RAG**	**G-RAG**	**Tie**	**RAG**	**G-RAG**	**Tie**	**RAG**	**G-RAG**	**Tie**	**RAG**	**G-RAG**	**Tie**	**RAG**	**G-RAG**	**Tie**	**RAG**	**G-RAG**	**Tie**
Compre.	34.83	**61.69**	3.48	35.82	**60.70**	2.99	15.84	**79.21**	4.95	20.79	**77.72**	1.49	12.87	**84.16**	2.97	33.17	**65.35**	1.49
Diversity	18.91	**77.61**	3.48	16.42	**78.11**	5.47	6.93	**88.12**	4.95	6.44	**91.09**	2.48	4.95	**91.58**	3.47	25.74	**71.78**	2.48
Empower.	29.85	**62.19**	7.96	28.86	**67.11**	3.98	14.85	**78.71**	6.44	16.83	**81.19**	1.98	14.36	**68.32**	17.33	33.17	**58.42**	8.42
Winner	31.34	**65.17**	3.48	28.36	**67.16**	4.48	12.87	**82.18**	4.95	15.85	**83.17**	0.99	11.39	**85.64**	2.97	30.20	**67.33**	2.48

Table 3

LLM-As-A-Judge (GPT4-o) voting percentages compare Graph RAG (with entity focus and semantic clustering) against traditional RAG across three qualitative dimensions. In all applications, GPT-4 serves as the LLM, and text-embedding-3-large is used as the embedding model. Graph RAG demonstrates superior performance across all metrics. However, these results should be interpreted with caution, as the LLM-As-A-Judge is sensitive to various configurations, including prompting.

Engine Configuration: We built a Proof of Concept (PoC) by instantiating our Graph RAG Engine using the Volkswagen VW-ID 3 manual, which has over 600 pages. A notable challenge in this application is the inclusion of technical images across multiple pages, the use of concise bullet points, and the presentation of the content in German. We used LlamaParser[4] to accurately extract text from the PDF. LlamaParser can accurately extract images and short texts, providing complete metadata. We exported the text as Markdown during the knowledge graph creation stage. We apply a self-constructed, without ontology, LLM-assisted method to build the graph. We processed the manual page by page using GPT-4, guided by detailed prompts to extract structured knowledge, i.e., identifying entities and relationships. We keep the page number as a reference for each entity and relation. After populating the KG with entities and relations, we also identify communities and create summaries.

As in the previous application, we employ two knowledge-retrieval modes: similarity-based retrieval and Semantic Clustering. Similarity-based retrieval addresses entity-focused questions, such as "Does this car have a CO2 emission sensor?" The semantic clustering retrieval method covers questions that require a global understanding of all car characteristics, for example, "What are the car's safety measures?"

Based on the retrieved subgraph or community summary, the assistant formulates a response by synthesizing information from the knowledge graph. Each answer is grounded in specific entities and relations and includes direct references to the original manual pages where the information was found. This ensures factual accuracy and allows users to verify the content within its original context. Especially for safety-critical topics, this page-level citation enhances reliability and transparency, making the assistant a trustworthy companion in maintenance scenarios.

Evaluation and Impact: To evaluate this application, we use the same settings as in Section 4.1. We use RAGAS to automatically generate two types of test datasets: local and global. We use GPT-4o as a judge to evaluate three metrics: Comprehensiveness, Diversity, and Empowerment. The experimental results favor Graph RAG, which outperforms the alternatives across all metrics, as shown in Table 3. ***Impact***: This application integrates with Fraunhofer IDMT audio components, enabling interaction with the system via voice commands. This application was developed in collaboration with an aerospace manufacturing company, further disseminating the advantages of Graph RAG methods in industries.

4.3. Wenn Fraunhofer Wüsste - Was Fraunhofer Weiß (WFW)

Motivation: Efficient enterprise knowledge management is a pressing issue for fostering innovation and gaining a competitive advantage. In Fraunhofer's case, there are 76 institutes located throughout Germany. Each institute manages its web portal, PDF repository, and press release, resulting in a large, decentralized information landscape. Enterprise surveys report that 67% of organizations struggle with knowledge silos, and researchers spend 20-30% of their time on project scoping, identifying prior work, and potential partners. Conventional keyword searches cannot resolve synonym drift ("graph analysis" versus "network mining") or complex questions involving multiple entities, such as "Which institutes collaborate on AR for maintenance?" Missed connections have tangible costs. For example, Fraunhofer cites duplicate proposals on AI/AR worth €1.2 M due to unawareness of parallel efforts.

[4] https://www.llamaindex.ai/llamaparse

Engine Configuration: Similar to the previous applications, we applied our Graph RAG methodology to extract structured knowledge from the websites of three Fraunhofer institutes: FIT, IAO, and IDMT. The content—such as team pages, project descriptions, and research areas—was parsed using a custom web crawler and processed page by page with GPT-4o, guided by detailed prompts to extract structured knowledge in the form of (subject, predicate, object) triples. We retained the source page URL for each extracted entity and relation to ensure traceability. After populating the graph, we identified semantic communities and generated summaries for each cluster to support higher-level understanding and reasoning.

To retrieve information from the graph, we support both entity-level and thematic query handling. Specific lookups, such as "Who works on edge computing at Fraunhofer IAO?", are handled via similarity-based retrieval, which traverses the graph to retrieve relevant nodes and their direct relationships. For broader, exploratory queries—such as "Which institutes collaborate on AI for smart factories?"—we initially applied semantic clustering over community summaries. However, due to the high computational cost of traversing large clusters, we introduced a lightweight alternative using Hypothetical Document Embeddings (HyDE). This method generates a synthetic query representation and uses it to quickly rank and select the most relevant summaries.

In the generation step, responses are constructed from either the retrieved subgraph or the selected community summary. As in the car manual assistant, all generated answers are explicitly grounded in the graph and include references to the source web pages from which entities and relations were extracted. This ensures transparency, enables verification, and makes the assistant reliable for both factual lookup and higher-level knowledge discovery.

Evaluation and Impact: We evaluated the WFW application using the same LLM-As-A-Judge setup and RAGAS-based test generation process described in Section 4.1, covering both local (entity-level) and global (multi-hop) queries. As shown in Table 3, Graph RAG consistently outperforms traditional RAG across all qualitative metrics: Comprehensiveness, Diversity, and Empowerment. The most significant gains were observed in exploratory questions involving multi-entity reasoning, where traditional methods often fell short. One limitation is the reliance on public website data, which may omit internal or unpublished collaborations. ***Impact***: The application was demonstrated internally at Fraunhofer and received positive feedback. Future work will integrate additional data sources and scale the graph to include all 76 Fraunhofer institutes. In summary, the WFW assistant showcases how Graph RAG can streamline knowledge discovery in large organizations by surfacing connections across institutional silos.

5. Schema First Applications

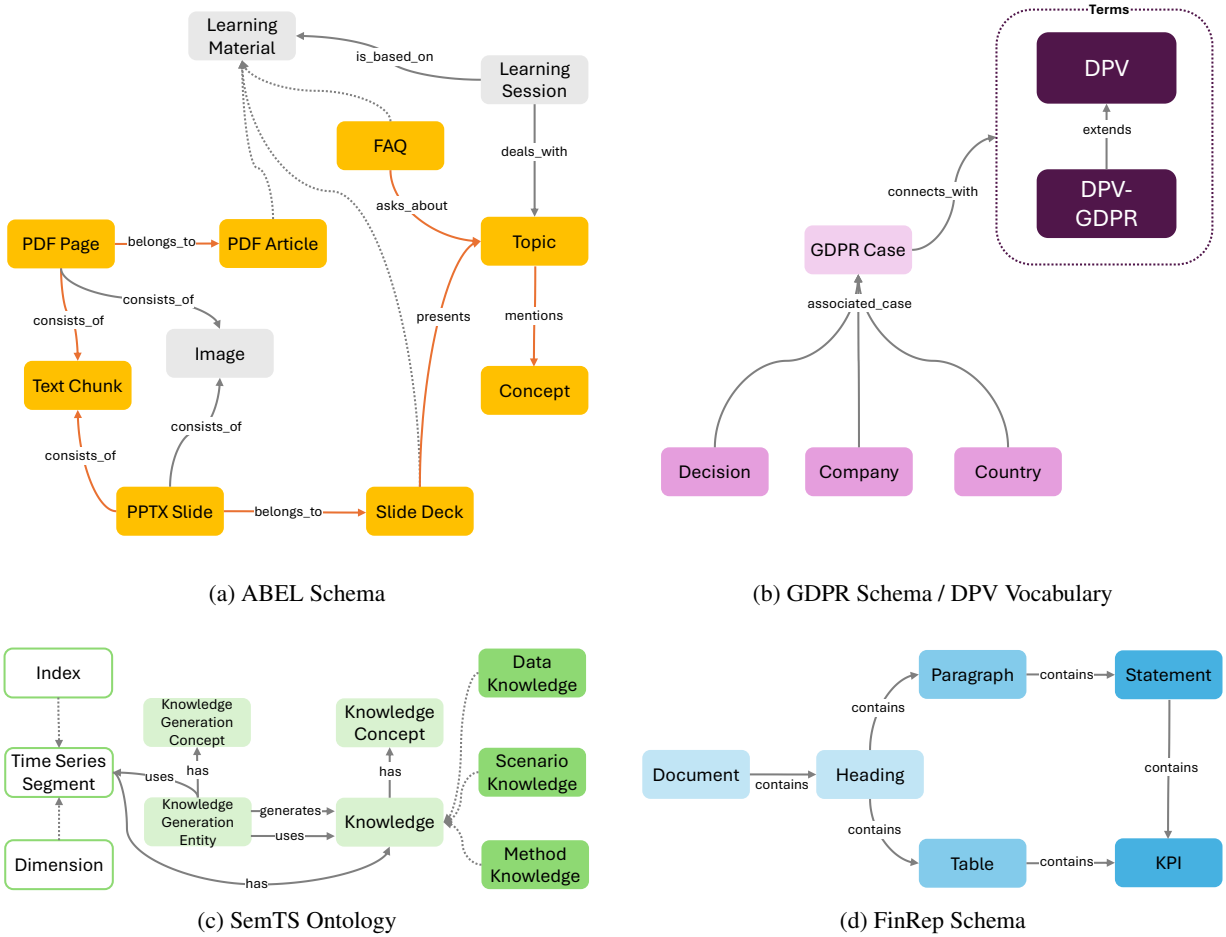
In several applications, a formal ontology exists or can be built. In this section, we present applications in which an ontology is a core element of the FRAG-KEDA engine setup.

5.1. Artificial Buddy for Effective Learning (ABEL)

Motivation: The relevance of technical topics such as Artificial Intelligence (AI) and Data Science is rapidly expanding beyond traditional technical fields, becoming essential for professionals from a wide range of backgrounds. In particular, training is reaching a more diverse target group, necessitating scalable solutions that can accommodate participants with varying levels of domain knowledge. Modern educational settings require more individualized and asynchronous learning, with a strong emphasis on personalized learning experiences where expert knowledge must be conveyed effectively [26, 41]. The growing demand for micro-credentials reflects a preference for learning in small units and the ability to demonstrate achievements in a modular way [86]. Micro-credentials are short, targeted learning programs that certify specific skills or knowledge.

Technologies—especially AI—have the potential to meet these requirements and transform traditional learning. However, in the context of knowledge transfer, the accuracy of information remains critical. Technologies such as chatbots based on large language models (LLMs) often exhibit hallucination, and interactions with them frequently lead to information overload due to verbose, overly detailed responses. To address hallucination issues, Retrieval-Augmented Generation (RAG) offers a promising solution by allowing an LLM to access external domain-specific knowledge. However, traditional RAG approaches are limited by their text-understanding capabilities [64], underscoring the need for improved methods in this area.

Engine Configuration: To address these challenges, we propose ABEL—the Artificial Buddy for Effective Learning—a prototype for a personalized learning app. ABEL is a joint project of the *Fraunhofer Institute for Applied Information Technology FIT* and the *Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS* with the overall goal of contributing to the further development of modern educational technologies. The ABEL app offers a hybrid learning concept that combines instructor-led training with personalized and self-guided learning. ABEL is based on a Graph Retrieval-Augmented Generation (Graph RAG) approach, realized with an LLM augmented by a knowledge graph. Graph RAG outperforms traditional RAG in many text understanding tasks [64]. The LLM augmented with knowledge acts as an interactive learning assistant, and the graph-based representation of the learning material makes the interrelations between the subject areas accessible.



(a) ABEL Schema

(b) GDPR Schema / DPV Vocabulary

(c) SemTS Ontology

(d) FinRep Schema

Fig. 3. Overview of schema abstractions used for schema-first applications.

The knowledge graph is built from existing Fraunhofer internal learning materials. The material comprises various types of documents, including PowerPoint presentations, PDF files, and manually generated pairs of Frequently Asked Questions (FAQs) with their corresponding answers. The documents contain text, tables, and pictures. The graph follows the predefined conceptual model in Figure 3(a), adapted to the structure of the provided data to accommodate different document and data types. To enable efficient semantic search within the knowledge graph, each node representing a text entity is augmented with its corresponding vector embedding. These embeddings are then stored directly in a graph database. This indexing approach enables rapid retrieval of semantically similar nodes using cosine similarity. The graph data is accessible via a REST API, accessed by an LLM chatbot, which answers a user query with one of the following RAG mechanisms:

– **FAQ-Based Retrieval:** The system identifies the top n FAQs that most closely match the query. These FAQs are used as context for the LLM, which generates an answer informed by them.
– **Concept-Centric Retrieval:** The system selects the concept most relevant to the query and retrieves the corresponding learning material, including the most pertinent text chunks. This context is provided to the LLM to produce a context-aware response.

These RAG mechanisms leverage the knowledge graph's structure to improve the accuracy and relevance of the generated responses. Currently, the chatbot responds only based on the external knowledge provided to it. If no match is found for a query, the chatbot will not respond. We achieve this behavior with the following prompt: *"Use the given pieces of context to answer the question. If you don't know the answer, just say that you don't know, don't try to make up an answer."*

Evaluation and Impact: The performance was assessed with the RAGAS [5] open-source evaluation framework. As an LLM scorer, we selected OpenAI's GPT-4o with a temperature of 0. To evaluate our retrieval quality, we focus on a collection of evaluation queries and define the following LLM-based metrics: (1) **Context Recall:** How many of the relevant documents were successfully retrieved?; (2) **Faithfulness:** How factually consistent is a response with the retrieved context? Using the RAGAS evaluation framework, we obtain a context–recall of **0.8056**, and a faithfulness score of **0.6763**. A context recall score > 0.8 indicates that the retriever retrieves most of the key facts. In contrast, the faithfulness is only moderate; tighter grounding — such as enforcing inline citations or adding a post-generation verification pass — should mitigate this shortcoming.

Impact: This application resulted in a scientific publication (In Pre-Print) highlighting the potential of Graph RAG systems in our learning context. With a knowledge graph constructed from accurate and relevant learning materials, learning efficiency can be increased. LLMs often suffer from hallucinations, and asking questions to an LLM usually results in an information overload due to long and detailed explanations. In contrast, graph-driven retrieval narrows the context, thereby reducing noise and improving answer relevance. Moreover, the ABEL concept can be applied to other thematic content by replacing the knowledge graph. The approach is scalable and enables rapid adaptation to market demands in educational training. With ABEL, we aim to advance the broader goal of improving modern educational technologies. We aim to demonstrate that modern technologies, such as knowledge graphs and LLMs, can make a significant contribution to the education sector.

5.2. GDPR Compliance Verifier (GdprVerifier)

Motivation: Since the General Data Protection Regulation (GDPR) came into effect in 2018, European data protection authorities have issued over 2,000 monetary penalties, totaling €4.48 billion as of March 2024 [19]. Providing an AI-assisted solution to help companies avoid common GDPR fines is crucial. Ideally, knowledge from prior cases should be used to enrich the AI-assisted solution, specifically by identifying companies with similar cases. Prior work has explored knowledge-graph-based compliance tooling, but these systems either offer static SPARQL queries [5] or ad-hoc dashboards that require expert users. RAG promises natural-language access, yet vanilla vector similarity falls short in legal settings where answers depend on specific articles, recitals, and cross-references [4, 112]. Graph RAG integrates knowledge graphs to bridge this gap by enabling entity-level grounding, constraint reasoning, and citation generation—capabilities highlighted by recent surveys [105].

Engine Configuration: In this application, we build the knowledge graph around the W3C Data Protection Vocabulary (DPV) [63] and its GDPR extension (version 2.1) [63]. All 2,864 concepts and 196 object properties retain their canonical IRIs and multilingual labels, providing the graph with legal provenance and out-of-the-box interoperability with DPV-aware tools. We use the documents stored at the GDPRxiv [83] repository to create a knowledge graph. This repository contains cases from the official government websites of EU countries. Each case is linked to texts in multiple languages. We perform a cleaning process to extract the case text files for each case, using only the English version, from the GitHub repository to ensure conformance.

[5] https://docs.ragas.io

To create the knowledge graph, we identify all DPV-GDPR terms in the text. We then perform entity linking using complex regular expressions (regex) to match only terms that are similar to the provided text. Subsequently, we filter out irrelevant terms by revalidating them against the quoted article columns in the enforcement database. During this process, we also extract subsections of select regulations to ensure precise results, which we then add to the connected terms column. Following this, we construct a knowledge graph using the DPV-GDPR vocabulary. Based on the information in the vocabulary CSV file, we define superclass-subclass relationships in ontology terms. Each node is categorized as a "Term" and is unique. We create a separate node for each case in the case.csv file, ensuring that each node has a distinct ID. Using the 'Connected terms' column, we link the case-type nodes to the term-type nodes, thus completing the knowledge graph. Additionally, we create distinct nodes for countries, company names, and legal decisions, and connect them through relevant relationships. Each country is assigned a URI from DBpedia. We also include a property-assistive field within the Case and Term nodes, both of which are textual. Furthermore, we create a structural assistive field that captures information about immediate neighbors and the types of relationships in text form. The property and structural assistive fields together are referred to as the LLM assistive field (LAF). In summary, the knowledge graph comprises five types of nodes: 1) Term, 2) Case, 3) Company, 4) Decision, and 5) Country.

Evaluation and Impact: For evaluation, we used DeepEval, an open-source framework for measuring and improving the performance of RAG systems and LLM applications. We set up four pipelines: (a) Baseline RAG, (b) LLM Assistive Field (LAF), (c) LAF with summary, and (d) LAF with summary and advanced queries.

The first pipeline is a vanilla RAG system for reference, which includes all legal text and case decisions. The second pipeline, LAF, as described earlier, includes the text of LAF Case and Term nodes. For the third pipeline, LAF with summary, we summarized the case text field from each unique case node using an LLM. The prompt instructed the LLM to preserve references to the GDPR, maintain key legal concepts and outcomes, and use neutral, professional language. The summarized results replaced the full case text in the structural assistive field, which, when combined with the property assistive field, formed an LAF summary. The fourth pipeline added advanced query results from the knowledge graph. The results encompass unique insights and analytics about the knowledge graph. These results, initially in JSON format, were converted into a textual file with sentences to provide context for the LLM. The text file was then integrated into LAF, along with a summary.

The results show that LLM Assistive Fields (LAF) alone does not perform significantly better than the Baseline RAG in terms of faithfulness and answer relevance. However, LAF achieves higher contextual relevance (0.474), indicating the richness and depth of the knowledge graph. When combined with summarization, LAF outperforms Baseline RAG in answer relevancy (0.981) and contextual relevancy (0.455). This improvement suggests that LAF with summary retains key legal concepts essential for generating accurate answers. LAF with summary and advanced queries achieves the highest contextual relevancy among all pipelines, demonstrating the added value of query-derived context for the LLM. While LAF with a summary excels in answer relevancy, LAF with a summary and advanced queries performs best in contextual relevancy. A strategic combination of both approaches could further improve performance beyond Baseline RAG.

Metrics	**Baseline RAG**	**LLM Assistive Field (LAF)**	**LAF with summary**	**LAF with summary and advanced queries**
Faithfulness	0.97	0.94	0.96	**0.97**
Answer Relevancy	0.96	0.93	**0.98**	0.95
Contextual Relevancy	0.38	0.47	0.45	**0.85**

Table 4

GdprVerifier evaluation: DeepEval metrics comparing different RAG approaches. Bold values indicate the best performance for each metric.

Impact: The system can be improved by adopting newer DPV releases, thereby yielding a more comprehensive ontology. The system will benefit not only companies but also Data Protection Officers and Government officials by enabling them to track companies and their finances.

5.3. Financial Report Generation (FinRep)

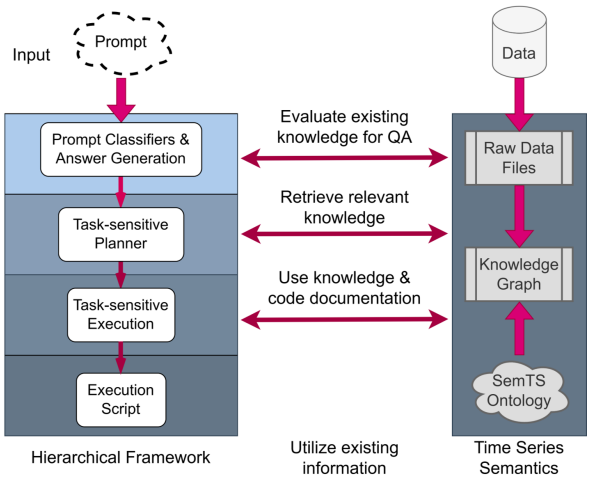
Motivation: The complex and highly specialized nature of finance—with its vast volumes of data and intricate quantitative analyses—often poses challenges for financial analysts working in this domain. LLM's ability to ingest and analyze large volumes of text makes it a highly promising tool in finance, already transforming how data-driven decisions are made within the financial sector [58, 107]. One compelling application is automated financial report generation, in which LLMs are tasked with transforming raw filings (e.g., annual reports) into coherent, KPI-driven narratives in response to a user prompt. In our application, an agentic pipeline autonomously retrieves the most relevant documents from a shared corpus. However, individual financial documents—such as SEC 10-K filings—often span hundreds of pages, and naively feeding them into an LLM's context window often yields factual errors and buries essential information [49]. This motivates the use of RAG for information extraction; however, traditional RAG often overlooks or misinterprets critical KPIs required to answer specific queries. For example, answering the query *"Compare Apple's revenue from 2020 to 2024 by segment"* requires the LLM to:

1. Locate the revenue by segment section in five different annual report documents (which may have different layouts).
2. Extract total and segment-level revenues from each section.
3. Align these values into a coherent time series.
4. Analyze year-to-year changes and find explanatory notes that may appear elsewhere in the filings.

As traditional RAG depends on vector similarity, it can miss semantically subtle but KPI-reach or explanation-dense passages and often struggles to reconcile dense numerical information when faced with long concatenations of text chunks [104].

Engine Configuration: To address these extraction and interpretation challenges, we construct a KPI-centric knowledge graph that (1) extracts each metric into a normalized schema and (2) preserves the context in which each figure appears. Our graph-construction pipeline leverages an LLM to ingest documents on-the-fly and automatically populate the graph. We guide the LLM with a prompt-encoded schema, yet still permit it to introduce novel entity and relation types when the source document requires concepts beyond the base ontology. This approach represents a schema-guided yet open-extraction hybrid strategy.

(a) User interface of the financial report generator



(b) Data Science Agents architecture

Fig. 4. (a) User interface of the financial report generator (b) Prompt-encoded schema used for constructing our KPI-centric knowledge graph from financial documents. The schema defines nodes for document components (nested headings, paragraphs, tables) alongside KPI and statement nodes—thereby preserving the full context of each financial figure.

The base schema used in our prompts is shown in Figure 4(b). By modeling document instances, including arbitrarily nested headings, paragraphs, and tables, we can trace each KPI to its precise location and narrative context, while dedicated KPI and statement nodes capture nuanced financial figures. We customized prompts to drive the graph-extraction process. For example, the first line of the prompt guiding the extraction reads: *"Given a markdown-parsed financial document (e.g., a 10-K) and the following ontology of entity types and relation types, identify all entities and all relationships among them."*

Although Graph RAG effectively captures KPIs, preserves their context, and extracts additional entities and relations from the text, it can sometimes fall short when the knowledge graph lacks specific entities or the extraction pipeline hasn't yet caught up with newly ingested documents. To address these gaps, we employ a Hybrid RAG approach, similar to [73], that combines Vector RAG (previously referred to as "traditional RAG") and Graph RAG. To streamline and shorten the context retrieved by Vector RAG, we employ a map-reduce summarization pipeline. It begins by summarizing individual text chunks, then consolidates them into document-level overviews, and finally merges all document summaries into a single unified summary. We then concatenate this summary with the results of both local, entity-focused, and global, community-based graph searches. Because user queries often require synthesizing information from multiple documents, we adopt a multi-hop strategy [87]: for each relevant document, we generate one or more targeted sub-queries and apply our Hybrid RAG pipeline to each. Figure 4(a) shows the user interface of our financial report generator, including the initial part of a generated report.

Evaluation and Impact: This application was presented at Hannover Messe 2025, showing the potential of automated financial report generation that: (1) **Selects** relevant documents via an LLM-based agent and formulates targeted sub-queries for each. (2) **Employs** a Graph RAG strategy to retrieve the most relevant context. (3) **Constructs** a KPI-centric knowledge graph using a schema-guided prompt that still permits open extraction of additional entities and relations. (4) **Streamlines** RAG's retrieved context through a map-reduce summarization pipeline. (5) **Concatenates** the assembled context and feeds it into an LLM to generate the final report. While a comprehensive evaluation of our approach remains future work, our preliminary results are promising. The addition of Graph RAG to our Vector RAG pipeline yields notable gains in both report completeness and accuracy. These gains emerged only after we employed a schema to guide KG construction, thereby ensuring a clear focus on KPIs and financial statements. The schema proves especially valuable for reliably extracting KPI values from complex Markdown tables, where Vector RAG alone often misses numbers. Finally, our map-reduce summarization pipeline also improves accuracy by providing cleaner and more concise inputs an effect we observed most strongly when generating reports with smaller models such as LLaMA 3.3 70B.

5.4. Data Science Agents (DSA)

Motivation: Using data science across various fields remains challenging due to a lack of expertise and in-house specialists to apply it effectively. Consequently, introducing the advantages of data science and analytics to these industries requires a more automated approach that can effectively analyze complex data without relying on data science professionals. This is where Data Science Agents become valuable, enabling users to interact with, inquire about, and derive insights from their data through natural language.

The Data Science Agents framework [28] comprises a suite of specialized agents that utilize LLMs and Graph RAG to analyze data autonomously. The current version of this framework focuses on time-series analysis. Its goal is to convert time series data into structured information and integrate it with an ontology to form a time-series knowledge graph. This enables LLMs to reason over the data, respond to inquiries, and generate new insights, such as forecasts and anomaly detection. Data Science Agents enable users to interact with and extract information from their data using natural language. For experts, these agents build a high-quality knowledge graph of time-series data that can be used in subsequent tasks.

Large Language Models enhanced with Graph RAG have already been used for various data preprocessing tasks. Owing to their proficiency in language processing, LLMs, when supplemented with task-specific knowledge graphs, provide a practical foundation for automated data preprocessing [66, 96]. Data preprocessing comprises various subtasks aimed at ensuring data quality and completeness [33]. Employing Graph RAG for data pre-processing

consistently results in higher accuracy compared to standard LLM methods [56, 66, 96]. The integration of knowledge graphs aids LLMs in selection or classification tasks by providing type information about candidate entities in entity-disambiguation tasks [66], or by offering clearer representations for potentially abbreviated entity mentions within the input data [96]. Furthermore, Graph RAG enhances the adaptability of these approaches across various data domains more effectively than other methods [66]. Given that real-world data, especially in large quantities, often exhibit incompleteness, inaccuracies, and other quality issues, data preprocessing is crucial for the success of more complex processing tasks [33]. Graph RAG approaches, with their superior accuracy and adaptability, are well-suited to these essential data preprocessing tasks. By moving beyond data preprocessing to data analysis, Data Science Agents can provide new insights into the data and enhance its reusability.

Engine Configuration: The knowledge graph is generated using the Semantic Time Series (SemTS) Ontology[6] [29], which describes how to model analytical and scenario-specific time-series information in a knowledge graph. This schema-based approach enables an LLM to automatically generate a structured knowledge graph from raw time-series data. A simplified illustration of how time-series knowledge can be structured using SemTS is shown in Figure 3. Knowledge is structured to function as an index, associating meaningful time series data with categorized insights. Generally, time series knowledge is divided into three types: DataKnowledge, ScenarioKnowledge, and MethodKnowledge. DataKnowledge encompasses insights derived directly from data or through analytics, such as class membership obtained via time-series clustering. ScenarioKnowledge covers verified contexts, including data annotations and domain-specific knowledge, often provided as expert information, and may include inferred facts. MethodKnowledge involves effective analytical method presets and the mathematical or logical counterparts of established process information. Any knowledge instance, as well as associated generation methods, can be further refined by additional concepts. Before any user requests the framework, knowledge is automatically generated by the agents when they gain access to time-series data. This enables the framework to begin with a valuable time-series knowledge graph before interacting with the user. Graph retrieval is achieved using semantic similarity search, where knowledge graph triples are embedded using Text-Embedding-3-Large[7] and retrieved based on the user request. In addition to retrieving from the knowledge graph, the framework can retrieve code documentation to improve code-generation performance and examine the actual data to infer additional information, such as which columns in a dataset require processing.

The agentic framework assigns specific tasks to various specialized agents. These include task-sensitive agents such as (i) a forecasting agent that utilizes graph information to deploy an optimized forecasting algorithm, (ii) a code generation agent for handling tasks beyond the usual scope to interact with the data, (iii) the central Graph RAG agent that can query and engage with graph information, and (iv) a conversational agent that uses data gathered or generated by other agents to craft natural language responses to user inquiries. Furthermore, the agents can identify when information is missing in the graph and initiate processes to create new insights. This new information is then stored back in the knowledge graph for reuse in other tasks.

The LLM's capabilities in data preprocessing using Graph RAG also generalize to this broader application. By utilizing a specialized graph for time series, the agentic network can interpret real-world data, respond to queries, and produce new insights for unique requests, with the creation process optimized using existing graph information. When working with time series, accuracy is key to creating valuable insights and can only be achieved using Graph RAG on our time series knowledge graph. The graph also allows the use of multiple different LLMs, so the framework is not limited to the most powerful LLMs on the market.

Evaluation and Impact: Essential to the operation of Data Science Agents is accurately delegating incoming user requests to their corresponding task-specific agents. This requires classifying the current user's conversation to analyze their intent for existing tasks. In the presented solution, this is achieved by utilizing an LLM classification component. Large Language Models have proven remarkable few-shot and zero-shot classification capabilities in multiple fields [52], [37], [85], [25]. An incoming request is sent to the classification component, alongside the currently available tasks with explanations, which returns a classification designating it to one of the subtasks, or a

[6] https://w3id.org/semts

[7] https://platform.openai.com/docs/models/text-embedding-3-large

simple chat answer if none apply. Using the ontology and knowledge graph, the LLM can retrieve class descriptions dynamically, as well as retrieve context information from the knowledge graph, which results in better accuracy. For example, a request such as *'Did our sensors indicate a maintenance today?'* might be either an anomaly detection or a classification task, depending on the context.

LLM Model	Accuracy	Failure Rate	Avg Response Time
Deepseek-R1 1.5b	0.67	0.06	2.86s
Deepseek-R1 7b	0.75	0.04	5.02s
Deepseek-R1 32b	0.83	**0.0**	30.78s
Mixtral	0.7	0.17	0.30s
Gemma2 27b	0.87	**0.0**	**0.16s**
Llama-3 70b	**0.88**	0.01	18.42s
GPT4o	0.83	**0.0**	0.59s

Table 5

Classification Test Results: Performance metrics on curated classification dataset. The failure rate indicates how often the model did not respond with a class at all. Bold values indicate the best performance for each metric.

Equally important to the framework is the precise identification of information already present in the knowledge graph and determining when to generate new information. To properly assess the framework's proficiency, we performed a test specifically designed for this task. In the case of our framework, reducing the number of false negatives is crucial, as a negative response will lead to the deployment of insight generation. In the case of a false positive, the user can simply rephrase their request to deploy insight generation as a follow-up.

LLM Model	Accuracy	False Positives	False Negatives	Avg Response Time
Deepseek-R1 1.5b	0.81	0.26	0.10	3.10s
Deepseek-R1 7b	0.92	0.13	0.02	6.97s
Deepseek-R1 32b	0.95	0.06	0.03	34.19s
Mixtral	**0.98**	**0.02**	**0.01**	1.28s
Gemma2 27b	0.92	0.12	0.04	**0.48s**
Llama-3 70b	0.97	0.03	0.02	0.67s
GPT4o	0.96	0.06	**0.01**	0.59s

Table 6

Graph RAG Information Check Test: Performance metrics on curated dataset. Bold values indicate the best performance for each metric.

Impact: The first version was demonstrated at Hannover Messe 2025 and published in a generative AI report [22]. Ongoing improvements to the engine are planned. We aim to test and incorporate additional graph-retrieval mechanisms, such as retrieving concepts from both the ontology and the knowledge graph, extracting information about communities and clusters of graph nodes, and directly connecting graph retrieval to raw data. Furthermore, in its current version, the framework enables to retrieve any information from the graph, while the generation of new information focuses on anomaly detection and forecasting. Looking ahead, we plan to extend the framework past its current emphasis on time series. While enhancing functionality in this area remains a priority, the goal is to broaden its scope to encompass general data analysis, aligning with the application's name.

6. Insight and Best Practices

As Graph RAG evolves, different paradigms emerge for building and indexing KGs, retrieving relevant information, and synthesizing responses. Given the variety of methods and the complexity of other domains and datasets, it is hard to define a single best practice. However, some common principles remain worth discussing.

6.1. Graph Indexing

In graph indexing, the primary focus should be on making the knowledge graph as rich and high-quality as possible. When creating a graph with an LLM, this means tuning the prompts for graph construction to your specific use case. For more complex use cases, we recommend using an ontology or schema to guide the creation of the knowledge graph. Although Large Language Models (LLMs) can function as powerful tools to assist in the KG construction process, for example, by extracting entities and relations from textual sources according to the ontology-defined structure, they should not serve as the exclusive basis for defining the knowledge structure or validating its content. Relying solely on LLMs for these use cases risks yielding knowledge graphs (KGs) that are low-quality, inconsistent, and error-prone. Consequently, the decision requires weighing the initial effort and cost of developing and maintaining a formal ontology against the long-term benefits in terms of quality, trustworthiness, and enhanced capabilities of the resulting Graph RAG application. Once the initial graph has been indexed, it is essential to enrich it as much as possible. Adding extra nodes for specialized retrieval, such as FAQ nodes for question-and-answer systems, is very effective. Additionally, post-processing can remove duplicate or semantically similar nodes to improve graph conciseness.

The best approach depends on the specific use cases, but here are some specific examples that improved Graph RAG indexing in our applications: ABEL (Section 5.1) stores the embeddings as a list of properties on nodes in the graph, which can be used to perform fast similarity searches over the entire graph. Financial Report Generation (Section 5.3) provides a schema for the LLM and includes a prompt that indicates how to extract financial figures from source documents. Data Science Agents (Section 5.4) use a comprehensive ontology to describe time series information in a knowledge graph. This ontology is also used during indexing to optimize the embedding of knowledge graph nodes. In addition to this, model selection matters: Bigger models do not always perform better for KG extraction. For example, Gemma 2:27B consistently outperformed Llama 3.1:70B on entity and relation extraction tasks in our experiments, highlighting that model architecture and training data alignment matter more than size.

6.2. Graph Retrieval

To enable effective graph retrieval, using a high-quality knowledge graph is the first and most crucial step. A sparse or noisy knowledge graph may hinder the retrieval of relevant information and the generation of coherent responses. Conversely, an overly dense knowledge graph can lead to information overload, resulting in the retrieval of redundant or irrelevant details. To improve precision and reduce redundancy, a dedicated reranking mechanism should be employed after initial retrieval. When handling a query, instead of directly embedding the entire query, first extract named entities and combine them with the original query during retrieval. This often results in higher contextual recall. Additionally, when a query has no information in the graph, incentivizing the LLM not to answer in such cases increases retrieval accuracy, as the LLM will significantly reduce hallucinations. When retrieving information, understanding the structure of the underlying knowledge graph is vital for more effective retrieval. However, no retrieval strategy is optimal for all use cases. It is essential to evaluate multiple retrieval mechanisms, including vector-based, multi-hop, and hybrid approaches.

The EU Regulations QA Agent (Section 4.1) employs reranking to filter noisy matches in local search, thereby improving precision. ABEL (Section 5.1) employs a specific retrieval mechanism that first locates the concept, then moves to the topic, and finally to the text chunks within the document, thereby accelerating retrieval. Data Science Agents (Section 5.4) use the ontology to define SPARQL queries to retrieve triples from the knowledge graph. It uses the ontology's definitions to determine whether nodes require deeper search.

6.2.1. Query Classification

When using hybrid retrieval strategies, during the graph-guided retrieval step, a suitable search strategy must be selected, as different approaches are better suited to various scenarios. For instance, if a query targets a specific entity, e.g., *What is the scope of the Capital Requirements Regulation?*, then semantic similarity may prove to be an efficient means to map the query to stored entities. In contrast, when the question requires understanding across the entire knowledge base (e.g., *What are the five fundamental goals of the EU's Digital Services Act for regulating online platforms?*), a global search mode that accesses broader, pre-generated community reports becomes more

	CRR			VW			WFW		
Metric	**RAG**	**G-RAG**	**Tie**	**RAG**	**G-RAG**	**Tie**	**RAG**	**G-RAG**	**Tie**
Comprehensiveness	**71.39**	22.39	6.22	**66.58**	19.30	14.11	**58.41**	31.68	9.90
Diversity	33.58	**51.49**	14.92	32.17	**51.98**	15.84	30.94	**60.39**	8.66
Empowerment	**51.99**	35.57	12.43	**51.73**	31.43	16.83	**49.01**	33.41	17.57
Overall Winner	**61.94**	25.62	**12.44**	**60.64**	27.97	**11.38**	**46.28**	38.61	**15.10**

Table 7

After prompt engineering of traditional RAG, the results favor it. This result demonstrates the sensitivity of LLM-As-A-Judge's approach to settings such as prompting, underscoring the need for more robust evaluation methods.

suitable. When retrieval speed is not relevant, multiple strategies can be combined to provide richer contextual information to the LLM. For context-intensive use cases, using existing context information from the graph yields higher classification accuracy.

6.3. Graph Generation

When generating responses using the retrieved context, it is essential to clearly define the output format and the evaluation criteria for the LLM. When generating answers from the graph, code-like formats are most effective, striking a good balance between efficiency and efficacy. They achieve better faithfulness and answer relevance, due to LLMs' strong code understanding.

ABEL (Section 5.1) defines that responses be provided as natural language sentences in markdown with the corresponding source and the document name. That way, the response references the source documents. Data Science Agents (Section 5.4) provide task descriptions and output definitions for the following tasks: graph RAG-guided classification, graph information evaluation, and graph question answering. The retrieved triples can also be used to directly reference existing information in the graph.

7. Open Challenges

While Graph RAG presents a promising framework for structured retrieval and reasoning, its practical deployment introduces several challenges.

7.1. Graph RAG Evaluation

Evaluating natural language that accounts for both the request and the underlying situation remains difficult, as no purely algorithmic metrics can fully capture these nuances. As a result, novel evaluation metrics that use LLMs to assess complex natural-language responses introduce new challenges. Popular evaluation frameworks, such as Ragas [17], utilize RAG to create test datasets and LLMs to evaluate responses. The potential issue here lies in using RAG to evaluate new RAG applications. If the underlying retrieval mechanisms in the new applications require specific adaptations to function, using Ragas, with its more generic RAG implementation, will not yield usable test datasets and will have a detrimental effect on evaluation performance [71, 72, 102]. Ragas requires references to text chunks that include the node properties of its own knowledge graph. For Data Science Agents, that is not possible, since we have no text documents, but we use a time series as the basis of our knowledge graph. In addition, the Ragas graph has no concept of semantics or RDF and cannot properly handle such information to generate a dataset. The lack of support for RDF and JSON-LD in these evaluation frameworks poses a challenge when evaluating more complex Graph RAG applications. Ragas [17] employs a proprietary knowledge graph implementation for its RAG system, which is not compatible with RDF graphs. Furthermore, it is important to be critical when evaluating other LLMs using LLMs. Table 7 shows that adjusting the prompts can substantially affect LLM-based judgments of output quality, thereby reducing the reliability of the metric.

7.2. KG Efficiency and Scalability

In Graph RAG, scalability challenges arise in both the knowledge graph construction and retrieval phases. Constructing a knowledge graph using popular frameworks, such as GraphRAG [16], often requires prior summarization of source data, which can incur significant upfront indexing costs that are prohibitive in large-scale settings. Moreover, generating summary nodes typically relies on large language models (LLMs), thereby increasing the cost of data creation and storage with increasing dataset size.

Similarly, during the retrieval phase, as the number of entities and relations in the knowledge graph grows, the set of candidate subgraphs increases exponentially [65], complicating efficient retrieval. For instance, in the legal compliance application discussed in Section 4.1, the continuous addition of regulatory documents expands the graph, thereby degrading retrieval quality. To address this, an agent-based approach was implemented to retrieve relevant subgraphs, thereby incrementally improving retrieval performance.

7.3. Maintenance of Dynamic Knowledge Graphs

Dynamic knowledge graph maintenance involves continuously updating a knowledge graph with newly discovered facts, relationships, and semantic structures in response to evolving data sources. This process is critical for Graph RAG applications, where the underlying graph provides real-time context for large language models. In particular, LLM-based ontology population [57] can automate the discovery of new classes, properties, and instances by extracting relevant information from text, thereby reducing manual effort to expand or refine the ontology. This approach can operate within defined schemas or adopt a schemaless strategy, offering flexibility when handling diverse and unpredictable data. However, challenges arise in implementing efficient graph updates, particularly because global community-detection or indexing strategies often require regeneration. Incrementally updating only affected subgraphs, rather than recreating the entire network, mitigates these issues and significantly reduces computational overhead. Techniques like versioned graph storage, streaming ingestion, and node-level event-driven triggers can help maintain accurate, dynamically populated knowledge graphs with minimal downtime. Establishing robust pipelines for extracting and validating new information is also key. Graph embeddings, combined with LLM-driven entity linkage and disambiguation, help ensure that newly introduced nodes and edges remain consistent with the existing knowledge base. Additionally, leveraging incremental indexing strategies preserves quick access patterns for global search without rebuilding the entire index.

7.4. Temporal Knowledge Graphs

Temporal updates to knowledge graphs are becoming increasingly relevant, particularly in light of updated EU regulations that require consistent tracking and documentation of historical data. Unlike standard KGs, temporal graphs face unique challenges because they must capture not just relationships among entities, but also how these relationships change over time. Various strategies exist for representing time in graphs, including snapshot-based approaches [40], time-stamped triples, and specialized temporal data models [27]. However, Graph RAG applications currently do not employ temporal KGs. This raises an updateability issue, where older information is often overwritten, resulting in the loss of valuable historical context. For instance, if an organization consults its KG for regulatory compliance, missing or outdated information may compromise the accuracy of the final analysis. The question, therefore, becomes: What kind of temporal KG can effectively integrate with Graph RAG methods for richer, time-aware responses? Another critical concern is the impact of historical data on question-answering (Q&A) performance. On the one hand, retaining older versions of facts can significantly enrich context and perspective, strengthen adherence to factual timelines, and improve compliance tracking. On the other hand, there may be a trade-off: an LLM might misinterpret or retrieve historical data, thereby inadvertently generating responses that no longer reflect current regulations or conditions. In some scenarios, it may be more effective to restrict the LLM's access to time-limited data to prevent confusion. Adopting temporal graphs requires a careful balance between maintaining comprehensive historical records and providing accurate, current answers. This balance is crucial for advancing Graph RAG and ensuring effective integration of changing regulations and historical data without compromising QA performance.

7.5. Local and Small LMs

As large-scale LLMs with parameters in the double or even triple digits of billions require substantial computational resources, they are most often hosted and operated by third-party providers. The reliance on third-party providers raises concerns, depending on the field of application, as well as general concerns about data privacy and protection. These include the use of edge devices (e.g., car assistants), the creation of financial or medical reports, and ensuring compliance with GDPR or similar laws in other jurisdictions. To address these challenges, using small, local LLMs presents an option that adheres to legal and operational constraints.

As indicated by the evaluation on Data Science Agents in Section 5.4, smaller LLMs can achieve good results, comparable to large LLMs, using Graph RAG. However, an important topic when handling smaller LLMs is their higher tendency to produce malformed output. This is especially disadvantageous when, for example, function calls are incorrectly formed in an agent-based workflow. Smaller LLMs require more sophisticated output handling to remain effective. They also become more confused, are more likely to hallucinate, and produce output that is less related to the prompt more often. Another concern is that small LLMs also have a limited ability to handle semantically challenging inputs that require additional context. In the Data Science Agents classification tests, the dataset requests are rated by complexity, and smaller LLMs more frequently misclassify more complex requests. Applying smaller LLMs in our applications has also shown that they lack the reasoning capabilities of their larger counterparts, resulting in poorer performance in entity and relationship extraction tasks.

A way to overcome these hurdles is fine-tuning, which is more manageable due to the small LLMs' decreased size. Fine-tuning, as discussed in Section 2.1, enables the model to adapt to a specific task. To apply fine-tuning, two key issues must be addressed: dataset preparation and selection of the fine-tuning algorithm. To the best of the authors' knowledge, there are only a few, if any, open datasets that support general graph-aware fine-tuning for LLMs. If no suitable dataset exists, a task-specific dataset must be created, which is time-consuming and labor-intensive. Regarding the fine-tuning algorithm, there is no consensus on which performs best for adapting small LLMs to Graph RAG tasks. Two notable fine-tuning mechanisms for low-resource settings are LoRA [38] and GRPO [15]. The former adds weights to the model and trains these instead of the model's parameters, reducing the total number of training parameters. GRPO is based on reinforcement learning without human feedback and trains the model to use reasoning without relying on expected outputs, thereby easing the requirements for the fine-tuning dataset. Given the lack of established guidelines, fine-tuning typically involves several iterations of adjusting both the process and the dataset as needed. We are also developing such guidelines.

8. Conclusion

In this work, we demonstrate the potential of Graph Retrieval-Augmented Generation to address key limitations of traditional RAG methods by leveraging structured knowledge graphs for improved retrieval and reasoning. Seven diverse real-world applications were developed across domains such as legal compliance, finance, education, and customer support, in collaboration with industry and applied research partners. These efforts resulted in commercial products, demonstrations at international fairs, publications, and project acquisitions. The applications disseminate the benefits of Graph RAG in terms of retrieval accuracy, contextual relevance, and user trust, to a broader audience outside the Semantic Web community.

The FRAG-KEDA Engine serves as a modular reference implementation that enables graph-based ingestion, retrieval, and LLM orchestration, supporting the application of Graph RAG across heterogeneous domains. Our experiments distil best practices for (i) schema selection versus schemaless graph induction, (ii) retrieval-strategy switching under tight context budgets, and (iii) prompt-template optimization for LLMs. These resources lower the entry barrier for practitioners aiming to replicate or extend Graph RAG in production environments, offering practical guidance for constructing effective applications. The study also identifies cross-cutting challenges, such as graph drift, data freshness, and evaluation complexity, and proposes pragmatic solutions for deploying Graph RAG in production. These insights emphasize the ongoing need for innovation at the intersection of generative AI and structured knowledge representation.

Future work includes devising continual-learning strategies for dynamic graphs, developing reproducible, task-aligned benchmark suites for Graph RAG evaluation, investigating cross-modal graph representations, and exploring reinforcement-learning techniques that jointly optimize retrieval and generation. Addressing these areas can help the community further bridge the gap between symbolic knowledge integration and neural text generation, enabling reliable AI assistants in domains where factual precision and auditability are essential. Experiences and tools are shared with the community through a Graph RAG tutorial, supporting broader adoption and encouraging future research that advances the integration of symbolic knowledge and neural generation.

References

[1] G. Agrawal, T. Kumarage, Z. Alghamdi and H. Liu, Mindful-rag: A study of points of failure in retrieval augmented generation, in: *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, IEEE, 2024, pp. 607–611.

[2] W. Ansar, S. Goswami and A. Chakrabarti, A Survey on Transformers in NLP with Focus on Efficiency, *arXiv preprint arXiv:2406.16893* (2024).

[3] M. Awais, M. Naseer, S. Khan, R.M. Anwer, H. Cholakkal, M. Shah, M.-H. Yang and F.S. Khan, Foundational models defining a new era in vision: A survey and outlook, *arXiv preprint arXiv:2307.13721* (2023).

[4] S. Barnett, S. Kurniawan, S. Thudumu, Z. Brannelly and M. Abdelrazek, Seven Failure Points When Engineering a Retrieval Augmented Generation System, 2024. https://arxiv.org/abs/2401.05856.

[5] H. Bast, J. Kalmbach, T. Klumpp, F. Kramer and N. Schnelle, Efficient and Effective SPARQL Autocompletion on Very Large Knowledge Graphs, in: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 2893–2902–. ISBN 9781450392365. doi:10.1145/3511808.3557093.

[6] U. Brandes, M. Eiglsperger, J. Lerner and C. Pich, Graph markup language (GraphML) (2013).

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

[8] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y.T. Lee, Y. Li, S. Lundberg et al., Sparks of artificial general intelligence: Early experiments with gpt-4, *arXiv preprint arXiv:2303.12712* (2023).

[9] M.J. Buehler, Accelerating Scientific Discovery with Generative Knowledge Extraction, Graph-Based Representation, and Multimodal Intelligent Graph Reasoning, arXiv, 2024, arXiv:2403.11996 [cond-mat]. http://arxiv.org/abs/2403.11996.

[10] S. Carta, A. Giuliani, L. Piano, A.S. Podda, L. Pompianu and S.G. Tiddia, Iterative Zero-Shot LLM Prompting for Knowledge Graph Construction, arXiv, 2023, arXiv:2307.01128 [cs]. doi:10.48550/arXiv.2307.01128. http://arxiv.org/abs/2307.01128.

[11] W. Chen, T. Bai, J. Su, J. Luan, W. Liu and C. Shi, Kg-retriever: Efficient knowledge indexing for retrieval-augmented large language models, *arXiv preprint arXiv:2412.05547* (2024).

[12] D. Collarana, T. Heuss, J. Lehmann, I. Lytra, G. Maheshwari, R. Nedelchev, T. Schmidt and P. Trivedi, A Question Answering System on Regulatory Documents, M. Palmirani, ed., 2018. doi:10.3233/978-1-61499-935-5-41.

[13] S. Cucerzan, Large-Scale Named Entity Disambiguation Based on Wikipedia Data, in: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, J. Eisner, ed., Association for Computational Linguistics, Prague, Czech Republic, 2007, pp. 708–716. https://aclanthology.org/D07-1074/.

[14] DeepSeek-AI, A. Liu, B. Feng, B. Xue, B.-L. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Guo, D. Yang, D. Chen, D.-L. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Zhang, H. Ding, H. Xin, H. Gao, H. Li, H. Qu, J.L. Cai, J. Liang, J. Guo, J. Ni, J. Li, J. Wang, J. Chen, J. Chen, J. Yuan, J. Qiu, J. Li, J.-M. Song, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Xu, L. Xia, L. Zhao, L. Wang, L. Zhang, M. Li, M. Wang, M. Zhang, M. Zhang, M. Tang, M. Li, N. Tian, P. Huang, P. Wang, P. Zhang, Q. Wang, Q. Zhu, Q. Chen, Q. Du, R.J. Chen, R.L. Jin, R. Ge, R. Zhang, R. Pan, R. Wang, R. Xu, R. Zhang, R. Chen, S.S. Li, S. Lu, S. Zhou, S. Chen, S.-P. Wu, S. Ye, S. Ma, S. Wang, S. Zhou, S. Yu, S. Zhou, S. Pan, T. Wang, T. Yun, T. Pei, T. Sun, W.L. Xiao, W. Zeng, W. Zhao, W. An, W. Liu, W. Liang, W. Gao, W.-X. Yu, W. Zhang, X.Q. Li, X. Jin, X. Wang, X. Bi, X. Liu, X. Wang, X.-C. Shen, X. Chen, X. Zhang, X. Chen, X. Nie, X. Sun, X. Wang, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yu, X. Song, X. Shan, X. Zhou, X. Yang, X. Li, X. Su, X. Lin, Y.K. Li, Y.Q. Wang, Y.X. Wei, Y.X. Zhu, Y. Zhang, Y. Xu, Y. Huang, Y. Li, Y. Zhao, Y. Sun, Y. Li, Y. Wang, Y. Yu, Y. Zheng, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Tang, Y. Piao, Y. Wang, Y. Tan, Y.-B. Ma, Y. Liu, Y. Guo, Y. Wu, Y. Ou, Y. Zhu, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Zha, Y. Xiong, Y. Ma, Y. Yan, Y.-W. Luo, Y.-m. You, Y. Liu, Y. Zhou, Z.F. Wu, Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Huang, Z. Zhang, Z. Xie, Z.-g. Zhang, Z. Hao, Z. Gou, Z. Ma, Z. Yan, Z. Shao, Z. Xu, Z. Wu, Z. Zhang, Z. Li, Z. Gu, Z. Zhu, Z. Liu, Z.-A. Li, Z. Xie, Z. Song, Z. Gao and Z. Pan, DeepSeek-V3 Technical Report, *ArXiv* **abs/2412.19437** (2024). https://api.semanticscholar.org/CorpusID:275118643.

[15] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J.-M. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Zhang, X. Yu, Y. Wu, Z.F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B.-L. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Chen, D.-L. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Ding, H. Xin, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Wang, J. Chen, J. Yuan, J. Qiu, J. Li, J. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R.J. Chen, R.L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S.S. Li, S. Zhou, S.-K. Wu, T. Yun, T. Pei, T. Sun, T. Wang, W. Zeng, W. Zhao, W. Liu, W. Liang, W. Gao,

W.-X. Yu, W. Zhang, W.L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X.Q. Li, X. Jin, X.-C. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y.K. Li, Y.Q. Wang, Y.X. Wei, Y. Zhang, Y. Xu, Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y.-J. Zou, Y. He, Y. Xiong, Y.-W. Luo, Y.-m. You, Y. Liu, Y. Zhou, Y.X. Zhu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z.-g. Zhang, Z. Hao, Z. Ma, Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z.-A. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang and Z. Zhang, DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, *ArXiv* **abs/2501.12948** (2025). https://api.semanticscholar.org/CorpusID:275789950.

[16] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt and J. Larson, From Local to Global: A Graph RAG Approach to Query-Focused Summarization, arXiv, 2024, arXiv:2404.16130 [cs]. http://arxiv.org/abs/2404.16130.

[17] S. Es, J. James, L.E. Anke and S. Schockaert, Ragas: Automated evaluation of retrieval augmented generation, in: *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, 2024, pp. 150–158.

[18] O. Etzioni, M. Banko, S. Soderland and D.S. Weld, Open information extraction from the web, *Communications of the ACM* **51**(12) (2008), 68–74. doi:10.1145/1409360.1409378.

[19] European Data Protection Board (EDPB) or relevant authority, GDPR Enforcement and Penalties Statistics, 2024, Accessed: March 2024. https://www.edpb.europa.eu/.

[20] W. Fan, Y. Ding, L. Ning, S. Wang, H. Li, D. Yin, T.-S. Chua and Q. Li, A survey on rag meeting llms: Towards retrieval-augmented large language models, in: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, pp. 6491–6501.

[21] B. Fatemi, J. Halcrow and B. Perozzi, Talk like a Graph: Encoding Graphs for Large Language Models, arXiv, 2023, arXiv:2310.04560 [cs]. doi:10.48550/arXiv.2310.04560. http://arxiv.org/abs/2310.04560.

[22] Fraunhofer Big Data and Artificial Intelligence Alliance, Jenseits von Text und Bild – Generative KI für vielfältige Datenwelten, Whitepaper, Fraunhofer-Gesellschaft, 2025. https://www.bigdata-ai.fraunhofer.de/de/publikationen/whitepaper-genai-fuer-verschiedene-datentypen.html.

[23] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang and H. Wang, Retrieval-augmented generation for large language models: A survey, *arXiv preprint arXiv:2312.10997* **2** (2023).

[24] M. Gauthier, The Average Owner's Manual Takes Over Six Hours To Read, The Audi A3's Is Nearly Twice That, 2020. https://www.carscoops.com/2020/10/the-average-owners-manual-takes-takes-over-six-hours-to-read-the-audi-a3s-is-nearly-twice-that/.

[25] A. Gera, A. Halfon, E. Shnarch, Y. Perlitz, L.E. Dor and N. Slonim, Zero-Shot Text Classification with Self-Training, in: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022, pp. 1107–1119.

[26] I. Gligorea, M. Cioca, R. Oancea, A.-T. Gorski, H. Gorski and P. Tudorache, Adaptive Learning Using Artificial Intelligence in e-Learning: A Literature Review, *Education Sciences* **13**(12) (2023), 1216, Number: 12 Publisher: Multidisciplinary Digital Publishing Institute. doi:10.3390/educsci13121216. https://www.mdpi.com/2227-7102/13/12/1216.

[27] A. Graß, C. Beecks, S.A. Chala, C. Lange and S. Decker, A knowledge graph for query-induced analyses of hierarchically structured time series information, in: *European Conference on Advances in Databases and Information Systems*, Springer, 2023, pp. 174–184.

[28] A. Graß, C.I. Pack, D. Collarana, S. Decker and C. Beecks, Semantic Intelligence: Graph RAG-Driven Agents for Time Series Analytics, in: *International Conference on Intelligent Data Engineering and Automated Learning*, Springer, 2025, pp. 226–231.

[29] A. Graß, R.A. Deshmukh, C. Beecks and S. Decker, Towards an ontology for representing time series knowledge: motivation, requirements and concept, in: *International Conference on Advanced Information Systems Engineering*, Springer, 2025, pp. 103–110.

[30] J. Guo, L. Du, H. Liu, M. Zhou, X. He and S. Han, GPT4Graph: Can Large Language Models Understand Graph Structured Data ? An Empirical Evaluation and Benchmarking, arXiv, 2023, arXiv:2305.15066 [cs]. doi:10.48550/arXiv.2305.15066. http://arxiv.org/abs/2305.15066.

[31] M.U. Hadi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M.B. Shaikh, N. Akhtar, J. Wu, S. Mirjalili et al., Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects, *Authorea Preprints* **1** (2023), 1–26.

[32] H. Han, Y. Wang, H. Shomer, K. Guo, J. Ding, Y. Lei, M. Halappanavar, R.A. Rossi, S. Mukherjee, X. Tang et al., Retrieval-augmented generation with graphs (graphrag), *arXiv preprint arXiv:2501.00309* (2024).

[33] J. Han, J. Pei and H. Tong, *Data mining: concepts and techniques*, Morgan kaufmann, 2022.

[34] Y. Han, C. Zhang, X. Chen, X. Yang, Z. Wang, G. Yu, B. Fu and H. Zhang, ChartLlama: A Multimodal LLM for Chart Understanding and Generation, arXiv, 2023, arXiv:2311.16483. doi:10.48550/arXiv.2311.16483. http://arxiv.org/abs/2311.16483.

[35] J. He, M. Rungta, D. Koleczek, A. Sekhon, F.X. Wang and S. Hasan, Does Prompt Formatting Have Any Impact on LLM Performance?, arXiv, 2024, arXiv:2411.10541 [cs]. doi:10.48550/arXiv.2411.10541. http://arxiv.org/abs/2411.10541.

[36] X. He, Y. Tian, Y. Sun, N.V. Chawla, T. Laurent, Y. LeCun, X. Bresson and B. Hooi, G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering, arXiv, 2024, arXiv:2402.07630 [cs]. doi:10.48550/arXiv.2402.07630. http://arxiv.org/abs/2402.07630.

[37] S. Hegselmann, A. Buendia, H. Lang, M. Agrawal, X. Jiang and D. Sontag, Tabllm: Few-shot classification of tabular data with large language models, in: *International Conference on Artificial Intelligence and Statistics*, PMLR, 2023, pp. 5549–5581.

[38] J.E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang and W. Chen, LoRA: Low-Rank Adaptation of Large Language Models, *ArXiv* **abs/2106.09685** (2021). https://api.semanticscholar.org/CorpusID:235458009.

[39] Y. Hu, Z. Lei, Z. Zhang, B. Pan, C. Ling and L. Zhao, GRAG: Graph Retrieval-Augmented Generation, arXiv, 2024, arXiv:2405.16506 [cs]. http://arxiv.org/abs/2405.16506.

[40] S. Huang, F. Poursafaei, R. Rabbany, G. Rabusseau and E. Rossi, UTG: Towards a Unified View of Snapshot and Event Based Models for Temporal Graphs, *arXiv preprint arXiv:2407.12269* (2024).

[41] C.D. Jaldi, E. Ilkou, N. Schroeder and C. Shimizu, Education in the Era of Neurosymbolic AI, arXiv, 2024, arXiv:2411.12763 [cs]. doi:10.48550/arXiv.2411.12763. http://arxiv.org/abs/2411.12763.

[42] X. Jiang, R. Zhang, Y. Xu, R. Qiu, Y. Fang, Z. Wang, J. Tang, H. Ding, X. Chu, J. Zhao and Y. Wang, HyKGE: A Hypothesis Knowledge Graph Enhanced Framework for Accurate and Reliable Medical LLMs Responses, arXiv, 2024, arXiv:2312.15883 [cs]. doi:10.48550/arXiv.2312.15883. http://arxiv.org/abs/2312.15883.

[43] A. Kiulian, A. Polishko, M. Khandoga, Y. Kostiuk, G. Gabrielli, Ł. Gagała, F. Zaraket, Q.A. Obaida, H. Garud, W.W.Y. Mak et al., From English-Centric to Effective Bilingual: LLMs with Custom Tokenizers for Underrepresented Languages, *arXiv preprint arXiv:2410.18836* (2024).

[44] M.T.R. Laskar, S. Alqahtani, M.S. Bari, M. Rahman, M.A.M. Khan, H. Khan, I. Jahan, A. Bhuiyan, C.W. Tan, M.R. Parvez et al., A systematic survey and critical review on evaluating large language models: Challenges, limitations, and recommendations, in: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 13785–13816.

[45] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer and C. Bizer, DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia, *Semantic Web* **6**(2) (2015), 167–195. doi:10.3233/SW-140134.

[46] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, *Advances in neural information processing systems* **33** (2020), 9459–9474.

[47] J. Li, J. Chen, R. Ren, X. Cheng, W.X. Zhao, J.-Y. Nie and J.-R. Wen, The dawn after the dark: An empirical study on factuality hallucination in large language models, *arXiv preprint arXiv:2401.03205* (2024).

[48] N.F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni and P. Liang, Lost in the Middle: How Language Models Use Long Contexts, arXiv, 2023, arXiv:2307.03172 [cs]. doi:10.48550/arXiv.2307.03172. http://arxiv.org/abs/2307.03172.

[49] N.F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni and P. Liang, Lost in the middle: How language models use long contexts, *Transactions of the Association for Computational Linguistics* **12** (2024), 157–173.

[50] H. Luo and L. Specia, From understanding to utilization: A survey on explainability for large language models, *arXiv preprint arXiv:2401.12874* (2024).

[51] S. Ma, C. Xu, X. Jiang, M. Li, H. Qu, C. Yang, J. Mao and J. Guo, Think-on-Graph 2.0: Deep and Faithful Large Language Model Reasoning with Knowledge-guided Retrieval Augmented Generation, arXiv, 2025, arXiv:2407.10805 [cs]. doi:10.48550/arXiv.2407.10805. http://arxiv.org/abs/2407.10805.

[52] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu and J. Gao, Deep learning–based text classification: a comprehensive review, *ACM computing surveys (CSUR)* **54**(3) (2021), 1–40.

[53] M. Mintz, S. Bills, R. Snow and D. Jurafsky, Distant supervision for relation extraction without labeled data, in: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, K.-Y. Su, J. Su, J. Wiebe and H. Li, eds, Association for Computational Linguistics, Suntec, Singapore, 2009, pp. 1003–1011. https://aclanthology.org/P09-1113/.

[54] S. Mirchandani, F. Xia, P. Florence, B. Ichter, D. Driess, M.G. Arenas, K. Rao, D. Sadigh and A. Zeng, Large language models as general pattern machines, *arXiv preprint arXiv:2307.04721* (2023).

[55] B. Mo, K. Yu, J. Kazdan, P. Mpala, L. Yu, C. Cundy, C. Kanatsoulis and S. Koyejo, KGGen: Extracting Knowledge Graphs from Plain Text with Language Models, arXiv, 2025, arXiv:2502.09956 [cs]. doi:10.48550/arXiv.2502.09956. http://arxiv.org/abs/2502.09956.

[56] I.O. Mulang', K. Singh, C. Prabhu, A. Nadgeri, J. Hoffart and J. Lehmann, Evaluating the impact of knowledge graph context on entity disambiguation models, in: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2157–2160.

[57] D.M. Mustafa, A. Nadgeri, D. Collarana, B.T. Arnold, C. Quix, C. Lange and S. Decker, From Instructions to ODRL Usage Policies: An Ontology Guided Approach, *Proceedings of the VLDB Endowment. ISSN* **2150** (2024), 8097.

[58] Y. Nie, Y. Kong, X. Dong, J.M. Mulvey, H.V. Poor, Q. Wen and S. Zohren, A survey of large language models for financial applications: Progress, prospects and challenges, *arXiv preprint arXiv:2406.11903* (2024).

[59] OpenAI, Introducing ChatGPT, 2022, Accessed: October 04, 2023. https://openai.com/blog/chatgpt.

[60] OpenAI, GPT-4 Technical Report, 2023.

[61] M. Ostendorff, P.O. Suarez, L.F. Lage and G. Rehm, LLM-Datasets: An Open Framework for Pretraining Datasets of Large Language Models, in: *First Conference on Language Modeling*, 2024.

[62] J.J. Pan, J. Wang and G. Li, Survey of vector database management systems, *The VLDB Journal* **33**(5) (2024), 1591–1615.

[63] H.J. Pandit, B. Esteves, G.P. Krog, P. Ryan, D. Golpayegani and J. Flake, Data Privacy Vocabulary (DPV) – Version 2, 2024. https://arxiv.org/abs/2404.13426.

[64] B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, Y. Zhang and S. Tang, Graph Retrieval-Augmented Generation: A Survey, *CoRR* **abs/2408.08921** (2024). doi:10.48550/ARXIV.2408.08921.

[65] B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, Y. Zhang and S. Tang, Graph retrieval-augmented generation: A survey, *arXiv preprint arXiv:2408.08921* (2024).

[66] G. Pons, B. Bilalli and A. Queralt, Knowledge Graphs for Enhancing Large Language Models in Entity Disambiguation, in: *International Semantic Web Conference*, Springer, 2024, pp. 162–179.

[67] Y. Qin, S. Hu, Y. Lin, W. Chen, N. Ding, G. Cui, Z. Zeng, Y. Huang, C. Xiao, C. Han et al., Tool learning with foundation models, *arXiv preprint arXiv:2304.08354* (2023).

[68] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever et al., Language models are unsupervised multitask learners, *OpenAI blog* **1**(8) (2019), 9.

[69] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson et al., Sam 2: Segment anything in images and videos, *arXiv preprint arXiv:2408.00714* (2024).

[70] M.R.A.H. Rony, C. Suess, S.R. Bhat, V. Sudhi, J. Schneider, M. Vogel, R. Teucher, K.E. Friedl and S.R. Sahoo, CarExpert: Leveraging Large Language Models for In-Car Conversational Question Answering, in: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: EMNLP 2023 - Industry Track, Singapore, December 6-10, 2023*, Association for Computational Linguistics, 2023, pp. 586–604. doi:10.18653/V1/2023.EMNLP-INDUSTRY.56.

[71] S. Roychowdhury, S. Soman, H. Ranjani, N. Gunda, V. Chhabra and S.K. Bala, Evaluation of rag metrics for question answering in the telecom domain, *arXiv preprint arXiv:2407.12873* (2024).

[72] D. Ru, L. Qiu, X. Hu, T. Zhang, P. Shi, S. Chang, C. Jiayang, C. Wang, S. Sun, H. Li et al., Ragchecker: A fine-grained framework for diagnosing retrieval-augmented generation, *Advances in Neural Information Processing Systems* **37** (2024), 21999–22027.

[73] B. Sarmah, D. Mehta, B. Hall, R. Rao, S. Patel and S. Pasquali, Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction, in: *Proceedings of the 5th ACM International Conference on AI in Finance*, 2024, pp. 608–616.

[74] B. Sarmah, B. Hall, R. Rao, S. Patel, S. Pasquali and D. Mehta, HybridRAG: Integrating Knowledge Graphs and Vector Retrieval Augmented Generation for Efficient Information Extraction, arXiv, 2024, arXiv:2408.04948 [cs, q-fin, stat]. http://arxiv.org/abs/2408.04948.

[75] K. Sawarkar, A. Mangal and S.R. Solanki, Blended rag: Improving rag (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers, in: *2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, IEEE, 2024, pp. 155–161.

[76] M. Sclar, Y. Choi, Y. Tsvetkov and A. Suhr, Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting, arXiv, 2024, arXiv:2310.11324 [cs]. doi:10.48550/arXiv.2310.11324. http://arxiv.org/abs/2310.11324.

[77] J. Seo, J. Kim, S. Byun and H. Shin, How does a Language-Specific Tokenizer affect LLMs?, *arXiv preprint arXiv:2502.12560* (2025).

[78] Z. Sepasdar, S. Gautam, C. Midoglu, M.A. Riegler and P. Halvorsen, Enhancing structured-data retrieval with graphrag: Soccer data case study, *arXiv preprint arXiv:2409.17580* (2024).

[79] L. Shen, W. Tan, S. Chen, Y. Chen, J. Zhang, H. Xu, B. Zheng, P. Koehn and D. Khashabi, The language barrier: Dissecting safety challenges of llms in multilingual contexts, *arXiv preprint arXiv:2401.13136* (2024).

[80] A. Singh, A. Ehtesham, S. Kumar and T.T. Khoei, Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG, *arXiv preprint arXiv:2501.09136* (2025).

[81] D. Steinigen, R. Teucher, T.H. Ruland, M. Rudat, N. Flores-Herr, P. Fischer, N. Milosevic, C. Schymura and A. Ziletti, Fact Finder – Enhancing Domain Expertise of Large Language Models by Incorporating Knowledge Graphs, arXiv, 2024, arXiv:2408.03010 [cs]. doi:10.48550/arXiv.2408.03010. http://arxiv.org/abs/2408.03010.

[82] F.M. Suchanek, G. Kasneci and G. Weikum, Yago: a core of semantic knowledge, in: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, ACM, 2007, pp. 697–706. doi:10.1145/1242572.1242667.

[83] C. Sun, E. Jacobs, D. Lehmann, A. Crouse and S. Shastri, GDPRxiv: Establishing the State of the Art in GDPR Enforcement, *Proceedings on Privacy Enhancing Technologies (PoPETs)* **2023** (2023), 484–499. doi:10.56553/popets-2023-0121.

[84] J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, L.M. Ni, H.-Y. Shum and J. Guo, Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph, arXiv, 2024, arXiv:2307.07697 [cs]. doi:10.48550/arXiv.2307.07697. http://arxiv.org/abs/2307.07697.

[85] X. Sun, X. Li, J. Li, F. Wu, S. Guo, T. Zhang and G. Wang, Text classification via large language models, *arXiv preprint arXiv:2305.08377* (2023).

[86] G. Tamoliune, R. Greenspon, M. Tereseviciene, A. Volungeviciene, E. Trepule and E. Dauksiene, Exploring the potential of microcredentials: A systematic literature review, *Frontiers in Education* **7** (2023), Publisher: Frontiers. doi:10.3389/feduc.2022.1006811.

[87] Y. Tang and Y. Yang, Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries, *arXiv preprint arXiv:2401.15391* (2024).

[88] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale et al., Llama 2: Open foundation and fine-tuned chat models, *arXiv preprint arXiv:2307.09288* (2023).

[89] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar et al., Llama: Open and efficient foundation language models, *arXiv preprint arXiv:2302.13971* (2023).

[90] M. Trajanoska, R. Stojanov and D. Trajanov, Enhancing Knowledge Graph Construction Using Large Language Models, arXiv, 2023, arXiv:2305.04676 [cs]. http://arxiv.org/abs/2305.04676.

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

[92] S. Verma, Contextual Compression in Retrieval-Augmented Generation for Large Language Models: A Survey, *arXiv preprint arXiv:2409.13385* (2024).

[93] D. Vrandecic and M. Krötzsch, Wikidata: a free collaborative knowledgebase, *Commun. ACM* **57**(10) (2014), 78–85. doi:10.1145/2629489.

[94] X. Wang, Z. Wang, X. Gao, F. Zhang, Y. Wu, Z. Xu, T. Shi, Z. Wang, S. Li, Q. Qian et al., Searching for best practices in retrieval-augmented generation, *arXiv preprint arXiv:2407.01219* (2024).

[95] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E.H.-h. Chi, F. Xia, Q. Le and D. Zhou, Chain of Thought Prompting Elicits Reasoning in Large Language Models, *ArXiv* **abs/2201.11903** (2022). https://api.semanticscholar.org/CorpusID:246411621.

[96] L.L. Wei, G. Xiao and M. Balazinska, RACOON: An LLM-based Framework for Retrieval-Augmented Column Type Annotation with a Knowledge Graph, *arXiv preprint arXiv:2409.14556* (2024).
[97] B. Weng, Navigating the landscape of large language models: A comprehensive review and analysis of paradigms and fine-tuning strategies, *arXiv preprint arXiv:2404.09022* (2024).
[98] J. Wu, J. Zhu, Y. Qi, J. Chen, M. Xu, F. Menolascina and V. Grau, Medical Graph RAG: Towards Safe Medical Large Language Model via Graph Retrieval-Augmented Generation, arXiv, 2024, arXiv:2408.04187 [cs]. doi:10.48550/arXiv.2408.04187. http://arxiv.org/abs/2408.04187.
[99] Z. Xu, M.J. Cruz, M. Guevara, T. Wang, M. Deshpande, X. Wang and Z. Li, Retrieval-Augmented Generation with Knowledge Graphs for Customer Service Question Answering, 2024. doi:10.1145/3626772.3661370.
[100] Z. Xu, S. Jain and M. Kankanhalli, Hallucination is inevitable: An innate limitation of large language models, *arXiv preprint arXiv:2401.11817* (2024).
[101] R. Ye, C. Zhang, R. Wang, S. Xu and Y. Zhang, Language is All a Graph Needs, arXiv, 2024, arXiv:2308.07134 [cs]. doi:10.48550/arXiv.2308.07134. http://arxiv.org/abs/2308.07134.
[102] H. Yu, A. Gan, K. Zhang, S. Tong, Q. Liu and Z. Liu, Evaluation of retrieval-augmented generation: A survey, in: *CCF Conference on Big Data*, Springer, 2024, pp. 102–120.
[103] Y. Yu, W. Ping, Z. Liu, B. Wang, J. You, C. Zhang, M. Shoeybi and B. Catanzaro, Rankrag: Unifying context ranking with retrieval-augmented generation in llms, *Advances in Neural Information Processing Systems* **37** (2024), 121156–121184.
[104] Q. Zhang, S. Chen, Y. Bei, Z. Yuan, H. Zhou, Z. Hong, J. Dong, H. Chen, Y. Chang and X. Huang, A Survey of Graph Retrieval-Augmented Generation for Customized Large Language Models, *arXiv preprint arXiv:2501.13958* (2025).
[105] Q. Zhang, S. Chen, Y. Bei, Z. Yuan, H. Zhou, Z. Hong, J. Dong, H. Chen, Y. Chang and X. Huang, A Survey of Graph Retrieval-Augmented Generation for Customized Large Language Models, 2025. https://arxiv.org/abs/2501.13958.
[106] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin and M. Du, Explainability for large language models: A survey (2023), *URL: https://arxiv. org/abs/2309.01029* (2023).
[107] H. Zhao, Z. Liu, Z. Wu, Y. Li, T. Yang, P. Shu, S. Xu, H. Dai, L. Zhao, G. Mai et al., Revolutionizing finance with llms: An overview of applications and insights, *arXiv preprint arXiv:2401.11641* (2024).
[108] P. Zhao, H. Zhang, Q. Yu, Z. Wang, Y. Geng, F. Fu, L. Yang, W. Zhang, J. Jiang and B. Cui, Retrieval-augmented generation for ai-generated content: A survey, *arXiv preprint arXiv:2402.19473* (2024).
[109] Z. Zhong, H. Liu, X. Cui, X. Zhang and Z. Qin, Mix-of-granularity: Optimize the chunking granularity for retrieval-augmented generation, *arXiv preprint arXiv:2406.00456* (2024).
[110] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He et al., A comprehensive survey on pretrained foundation models: A history from bert to chatgpt.(2023), *arXiv preprint arXiv:2302.09419* (2023).
[111] J. Zhou and L. Chen, OpenRAG: Optimizing RAG End-to-End via In-Context Retrieval Learning, *arXiv preprint arXiv:2503.08398* (2025).
[112] Y. Zhou, Y. Liu, X. Li, J. Jin, H. Qian, Z. Liu, C. Li, Z. Dou, T.-Y. Ho and P.S. Yu, Trustworthiness in Retrieval-Augmented Generation Systems: A Survey, 2024. https://arxiv.org/abs/2409.10102.