Highlights

Integrating Graphs, Large Language Models, and Agents: Reasoning and Retrieval

Hamed Jelodar, Samita Bai, Mohammad Meymani, Parisa Hamedi, Roozbeh Razavi-Far, Ali Ghorbani

- A unified taxonomy of graph-LLM integration strategies across graph and integration mechanisms.
- A cross-domain analysis of representative graph-LLM methods, outlining strengths and limitations.
- Identification of key challenges, including scalability, hallucination mitigation, and reasoning depth.
- Practical guidelines for selecting appropriate graph-LLM approaches based on data constraints.

Integrating Graphs, Large Language Models, and Agents: Reasoning and Retrieval[⋆,⋆⋆]

Hamed Jelodar[a], Samita Bai[a], Mohammad Meymani[a], Parisa Hamedi[a], Roozbeh Razavi-Far[a] and Ali Ghorbani[a]

[a]Canadian Institute for Cybersecurity, Faculty of Computer Science, University of New Brunswick, 46 Dineen Drive, Fredericton, NB E3B 9W4, Canada

ARTICLE INFO

Keywords:
Large Language Models
Graph–LLM
Integration Knowledge Graphs
Graph-Augmented Reasoning
Retrieval-Augmented Generation
Graph Neural Network
Knowledge-Enhanced Pretrained
Agent-Based LLM Systems

ABSTRACT

Generative AI particularly Large Language Models, increasingly integrates graph-based representations to enhance reasoning, retrieval, and structured decision-making. Despite rapid advances, there remains limited clarity regarding when, why, where, and what types of graph–LLM integrations are most appropriate across applications. This survey provides a concise, structured overview of the design choices underlying the integration of graphs with LLMs. We categorize existing methods based on their purpose (e.g., reasoning, retrieval, generation, recommendation), graph modality (knowledge graphs, scene graphs, interaction graphs, causal graphs, dependency graphs), and integration strategies (prompting, augmentation, training, or agent-based use). By mapping representative works across domains such as cybersecurity, healthcare, materials science, finance, robotics, and multimodal environments, we highlight the strengths, limitations, and best-fit scenarios for each technique. This survey aims to offer researchers a practical guide for selecting the most suitable graph–LLM approach depending on task requirements, data characteristics, and reasoning complexity.

1. Introduction

Generative Artificial Intelligence (Gen-AI) has rapidly transformed the landscape of intelligent systems, with large language models (LLMs) demonstrating remarkable capabilities in understanding, generating, and reasoning over unstructured text. Despite these advances, LLMs primarily operate on sequential token representations and often lack explicit mechanisms for modeling structured relational knowledge [1, 2, 3]. In parallel, graph representations provide a powerful framework for capturing entities, relationships, and topological dependencies in structured data [4]. Consequently, integrating LLMs with graph-based learning has emerged as a promising direction to enhance reasoning, retrieval, and decision-making.

Graph-enhanced LLMs combine the semantic reasoning power of language models with the structured inductive bias of graphs. While LLMs excel at extracting contextual meaning from text, they often struggle to explicitly model complex relationships and multi-hop dependencies [5]. Graph structures address this limitation by encoding entities and their interactions in an organized and interpretable manner. Based on previous works [6, 7, 8], graph-LLM integration enables more reliable reasoning over structured knowledge that may be difficult to infer from text alone.

Recent studies further demonstrate the effectiveness of this combination across multiple domains. For instance, applications span software engineering [9], recommendation systems [10], healthcare [11] and clinical decision support [12], traffic prediction [13], and cybersecurity [14, 15]. For example, [16] introduces a graph-augmented LLM framework that incorporates relational structures to improve reasoning and predictive performance on structured datasets. Similarly, [17] proposes a hybrid LLM–GNN architecture in which semantic features generated by LLMs enhance graph-based classification and inference, highlighting the synergy between language and graph representations. Figure 1 illustrates the recent methods for integrating graphs and LLMs.

Motivated by these advances, this paper investigates recent strategies for integrating large language models with graph-based methods. Furthermore, we provide a structured overview of existing approaches and present a unified perspective on their design choices, capabilities, and application scenarios.

Motivation Since 2018, the importance of transformer-based models and large language models (LLMs) has increased rapidly and continues to be highlighted across diverse real-world applications. More recently, integrating LLMs with graph-based representations has shown strong potential to produce more informative and structured reasoning outcomes. Although a limited number of studies have explored the use of LLMs in conjunction with graphs, we have not identified any systematic research that comprehensively investigates the different aspects of graph-LLM integration. In particular, there is a lack of studies addressing the fundamental questions (when, why, where, and what), regarding the effective use of LLMs with graph structures. In this research, we aim to provide a structured and systematic analysis of graph-LLM integration, clarifying when such integration is most beneficial and why graph representations enhance LLM reasoning. We further examine where these approaches are most effective across application domains, and identify the key design choices and architectural patterns critical for their successful deployment.

*Corresponding author
h.jelodar@unb.ca (H. Jelodar); samita.bai@unb.ca (S. Bai); mohammad.meymani79@unb.ca (M. Meymani); parisa.hamedi@unb.ca (P. Hamedi); roozbeh.razavi-far@unb.ca (R. Razavi-Far); ghorbani@unb.ca (A. Ghorbani)
ORCID(s): 0000-0002-0713-3143 (H. Jelodar)

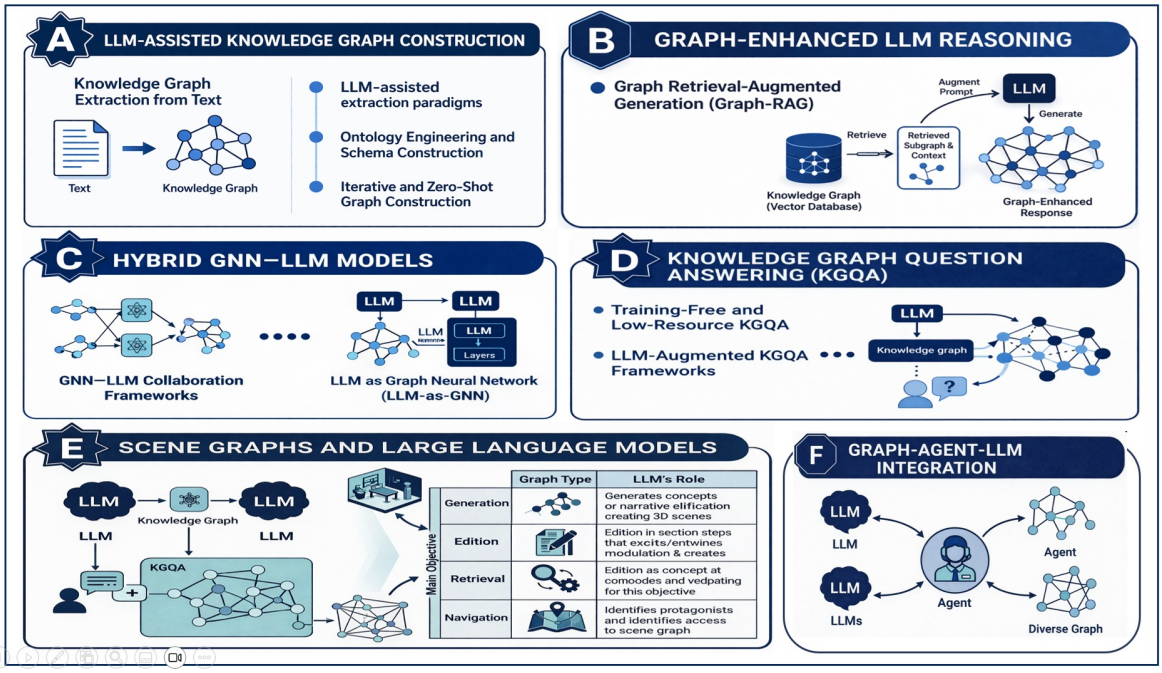

Figure 1: Recent methods for integrating Graph and LLMs

Why Combine Graphs and LLMs? To address the question of why graphs and large language models should be combined, the rationale largely depends on the specific use case and application requirements. LLMs are highly effective at understanding unstructured text, while graph-based models are effective at capturing structured relationships and dependencies.

For example, in software engineering [9], LLMs can analyze source code to infer semantic intent, whereas graphs can represent program structures such as call graphs and control-flow graphs. Together, they enable more accurate tasks such as code localization, vulnerability analysis, and program understanding.

Similarly, in healthcare and clinical decision support systems [11], LLMs can interpret clinical notes, while graphs encode structured medical knowledge, including disease-symptom relationships and drug interactions. Their integration supports more reliable clinical reasoning, improved decision support, and reduces the risk of hallucination in high-stakes medical applications.

Survey Scope and Contributions This survey focuses on the systematic analysis of recent efforts that integrate large language models with graph-based representations. We cover a broad range of graph modalities, including knowledge graphs, program and dependency graphs, interaction graphs, causal graphs, and multi-modal graph structures, and examine how they are combined with LLMs across diverse tasks and domains. The main contributions of this survey are summarized as follows:

- We present a unified taxonomy of graph-LLM integration strategies, categorizing existing approaches by their functional role, graph modality, and integration mechanism.
- We investigate representative methods across multiple application domains, highlighting their strengths, limitations, and suitability for different task settings.
- We identify common design patterns and recurring challenges, including scalability, hallucination mitigation, interpretability, and reasoning depth.
- We provide practical guidelines to help researchers and practitioners select an appropriate graph-LLM integration approach based on the task complexity, data availability, and application constraints.

The rest of the paper is organized as follows. Section 2 presents the foundational concepts underlying Large Language Models (LLMs) and Graph Neural Networks (GNNs), highlighting their complementary strengths and limitations. Section 3 reviews LLM-assisted graph construction techniques, including knowledge graph extraction, ontology engineering, and iterative graph building methods. Section 4 introduces graph-enhanced LLM reasoning paradigms, with a particular focus on Graph Retrieval-Augmented Generation (GraphRAG) and related reasoning frameworks. Section 5 discusses hybrid GNN–LLM models, categorizing them into collaboration frameworks, directional integrations, explainability-enhanced methods, and pre-trained architectures.

Section 6 explores knowledge graph question answering (KGQA) approaches, covering both training-free and LLM-augmented methods. Section 7 examines the integration of scene graphs with LLMs, including applications in generation, editing, retrieval, and navigation tasks. Section 8 presents graph–agent–LLM integration frameworks, emphasizing agentic reasoning and multi-step workflows. Section

9 highlights real-world applications across domains such as cybersecurity, healthcare, recommendation systems, and governance. Finally, the paper concludes with a summary of key insights, challenges, and future research directions.

2. Foundations

Add details of the subsections below:

2.1. Large Language Models

Large Language Models are deep neural architectures trained on large-scale textual corpora to learn general-purpose representations of language. They capture statistical patterns, semantic relationships, and contextual dependencies across tokens, enabling a wide range of tasks including text generation, question answering, and reasoning. Most modern LLMs are built upon the Transformer architecture, which leverages self-attention mechanisms to model long-range dependencies without relying on recurrence. Due to large-scale pretraining, LLMs exhibit strong generalization and in-context learning capabilities [18, 19], allowing them to adapt to new tasks with minimal or no parameter updates.

Formally, let a tokenized sequence be denoted as

$$\mathbf{x} = (x_1, x_2, \ldots, x_T),$$

where each $x_t \in \mathcal{V}$ and $\mathcal{V}$ is the vocabulary. An LLM parameterized by θ models the joint probability of the sequence using the autoregressive factorization:

$$P_\theta(\mathbf{x}) = \prod_{t=1}^{T} P_\theta(x_t \mid x_{<t}),$$

where $x_{<t} = (x_1, \ldots, x_{t-1})$. The model is trained by maximizing the likelihood over a dataset $\mathcal{D}$:

$$\theta^* = \arg\max_\theta \; \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[\sum_{t=1}^{T} \log P_\theta(x_t \mid x_{<t}) \right].$$

The core computation in Transformer-based LLMs is the self-attention mechanism. Given an input representation $X \in \mathbb{R}^{T\times d}$, attention is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V,$$

where

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V,$$

and W_Q, W_K, W_V are learnable projection matrices. Multiple attention heads are combined to capture diverse contextual interactions.

At inference time, given a context or prompt C, the model generates an output sequence $\mathbf{y}$ as:

$$P_\theta(\mathbf{y} \mid C) = \prod_{t=1}^{|\mathbf{y}|} P_\theta(y_t \mid C, y_{<t}),$$

enabling in-context learning without modifying model parameters.

Despite their strong representational power, LLMs encode knowledge implicitly within parameters θ, lacking explicit structured representations such as graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. This limits their effectiveness in tasks requiring explicit relational reasoning, multi-hop inference, and verifiable factual consistency.

2.2. Graph Neural Networks

Graph Neural Networks (GNNs) are designed to operate on graph-structured data, where entities are represented as nodes and relationships as edges. Through message passing and neighborhood aggregation, GNNs capture structural dependencies and relational patterns that are difficult to model using sequential representations. GNNs have been widely applied in domains such as social networks, recommendation systems, biology, and program analysis. Despite their effectiveness in structured reasoning, GNNs typically lack the rich semantic understanding and generalization capabilities of LLMs when dealing with unstructured language.

Graph neural networks were first introduced in [31] to learn representations of nodes in an unordered set of nodes connected by graph connectivity, aggregating integrative features, and disseminating information among local neighbors. GNNs became a hot topic after the emergence of GCNs [7]. GCNs use a variant of convolutional neural networks for node feature propagation. Simple Graph Convolution (SGC) [32] simplifies the vanilla GCN by removing nonlinear activations and compressing the weight matrix. Graph-SAGE [33] more space-efficiently aggregates and updates node features from local neighbors for aggregation and updating.

These methods often have issues such as over-smoothing. In response to these challenges, researchers have explored Transformer-based methods, such as Graphormer [34], for graph-learning tasks. These methods conceptualize graph nodes and edges as tokens, leveraging attention mechanisms to learn their relations for effective graph representation. Although these methods alleviate the over-smoothing issue and capture long-range dependencies between nodes, inadequate generalization significantly impedes progress in graph learning. GNNs have shown outstanding performance in graph reasoning tasks, however, the majority of GNNs are graph-specific. They are tailored for a particular graph type with consistent features and structures, thus posing challenges for generalization to different graphs.

3. LLM-Assisted Graph Construction

Write the text here about the sub-sections....

3.1. Knowledge Graph Extraction from Text

Knowledge graph (KG) extraction from text, often referred to as Text2KG, aims to transform unstructured or semi-structured documents into structured knowledge representations, typically in the form of entity-relation-entity triples enriched with types, attributes, and provenance [20].

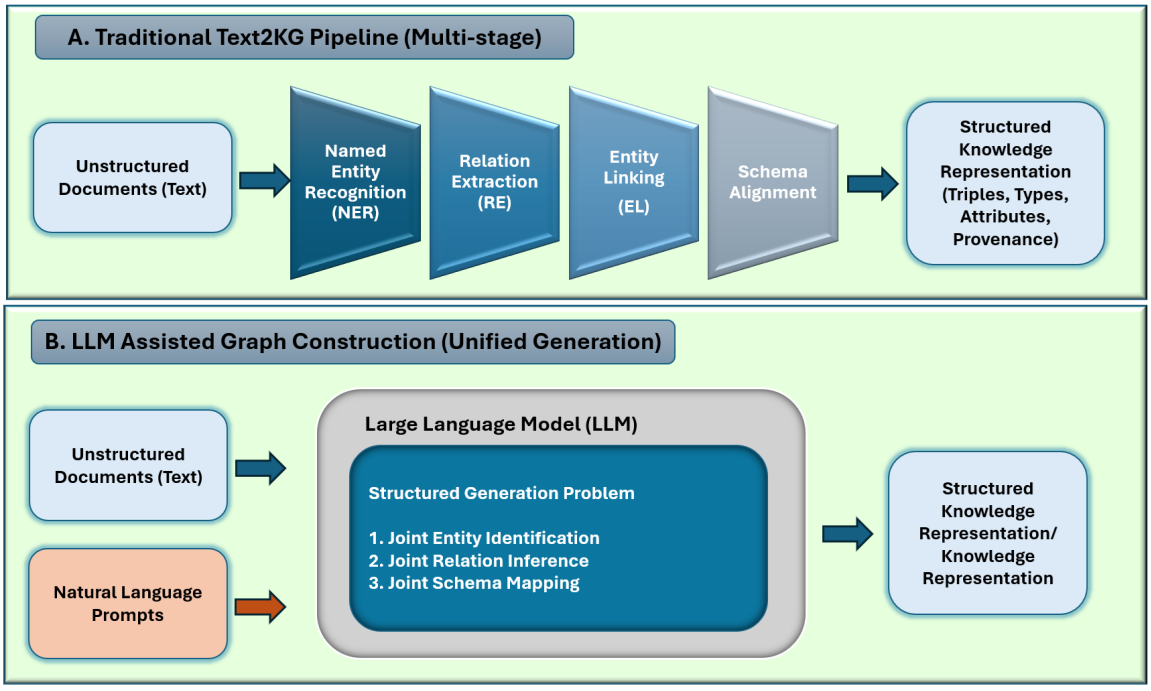

Figure 2: LLM-assisted graph construction vs traditional text2KG

Traditional Text2KG pipelines decompose this task into multiple components, including named entity recognition, relation extraction, entity linking, and schema alignment. While effective, such pipelines require task-specific models, annotated datasets, and substantial engineering effort, limiting their scalability and adaptability to new domains.

Recent advances in large language models (LLMs) have significantly reshaped the Text2KG landscape by enabling *LLM-assisted graph construction*, where extraction is formulated as a structured generation problem [21]. Instead of relying on separate extraction modules, LLMs can jointly identify entities, infer relations, and map outputs to a predefined or dynamically induced schema using natural language prompts. This paradigm substantially reduces pipeline complexity while improving semantic coverage, particularly for implicit relations and higher-level abstractions expressed in text [22, 23, 24].

3.2. LLM-assisted extraction paradigms.

Current research converges on several dominant paradigms for LLM-based KG construction. Prompt-based extraction instructs the LLM to output entities and relations in a constrained format such as JSON or RDF-like triples, often conditioned on a fixed ontology or relation inventory to control relation drift. Hybrid pipelines combine LLM-generated candidates with deterministic validation steps, including schema constraints, duplicate removal, and entity canonicalization, to improve precision and consistency. Retrieval-augmented approaches further ground extraction by incorporating relevant schema definitions, exemplar triples, or external knowledge retrieved at inference time, thereby mitigating hallucinations. More advanced systems adopt agentic or multi-step workflows, decomposing KG construction into extraction, verification, normalization, and consolidation stages with iterative refinement.

3.3. Ontology Engineering and Schema Construction

Ontology engineering is another method for connecting LLMs and graphs. For example, in [25], the authors focused on automating ontology and knowledge graph construction using large language models (LLMs) to reduce reliance on human experts. Their goal was to design a semi-automated pipeline that generates competency questions, builds an ontology schema, extracts entities and relationships from scientific texts, and constructs a knowledge graph with minimal manual intervention. They applied retrieval-augmented generation (RAG) and tested multiple open-source LLMs on a dataset of scientific publications related to deep learning. Their results showed that LLMs can effectively support ontology and knowledge graph creation with reasonable accuracy.

In [26], the authors of the paper focused on exploring how large language models like ChatGPT can assist with knowledge graph engineering (KGE), especially to support tasks that normally require deep expertise in graph structures, vocabularies, logic, and web technologies. They conducted comprehensive experiments to test whether ChatGPT (including GPT-3.5 and GPT-4) can help with the development and management of knowledge graphs by performing specific engineering tasks and evaluated the results of these experiments to demonstrate the model's potential and limitations in supporting KGE processes.

In [27], the authors focused on applying large language models (LLMs) to the challenging task of complex ontology alignment, which means finding semantic correspondences between detailed, multi-entity structures in different ontologies rather than just simple one-to-one matches. Their goal was to replicate and evaluate a previous approach that uses modular information about ontologies to guide LLMs in generating alignment rules, testing whether this method remains effective on a new dataset with more varied, complex alignments. They used a dataset based on the Enslaved ontology to propose alignment rules.

3.4. Iterative and Zero-Shot Graph Construction

Also, other researchers focused on zero-shot graph construction. In [28], the authors focused on automatically constructing knowledge graphs using large language models (LLMs) in a zero-shot setting, without relying on annotated data or predefined ontologies. Their goal was to design a scalable and general pipeline that extracts entities, relations, and triplets directly from unstructured text through iterative prompting. Using models like GPT-3.5, they demonstrated that LLMs can generate structured knowledge graphs effectively, showing the potential of prompt-based approaches for automated knowledge graph construction.

In [29], the authors focused on creating a new method to evaluate and detect hallucinations (inaccurate or inconsistent outputs) from large language models by converting their responses into knowledge graph (KG) structures and checking each part against known context. Their goal was to provide a more explainable and systematic way to pinpoint exactly where a model's answer diverges from factual grounding than previous metrics, while also improving detection accuracy by combining their KG representation with natural language inference models. They also introduced a follow-up method called GraphCorrect that leverages the same KG structure to attempt to correct hallucinated information.

In [30], the authors focused on creating a scalable benchmark called LLM-KG-Bench to assess how well large language models perform on tasks related to knowledge graph engineering (KGE). Their goal was to design a framework with multiple challenge tasks, including syntax/error correction, fact extraction, and dataset generation, that can automatically evaluate LLM responses, track prompt engineering, and visualize performance, because existing benchmarks didn't adequately measure LLMs' abilities in KGE. They found that while LLMs can be useful tools, they are not yet effective enough for knowledge graph generation using zero-shot prompting, and their benchmark helps quantify and compare model strengths and weaknesses in this area.

4. Graph-Enhanced LLM Reasoning

This section surveys graph-enhanced LLM reasoning paradigms, where graph structures are incorporated directly into the inference pipeline to impose relational constraints, enable multi-hop reasoning, and improve factual consistency. We review Graph Retrieval-Augmented Generation (Graph-RAG) methods that perform structure-aware retrieval over graphs (Section 4.1), graph-based prompting and tokenization techniques that encode topology and relational semantics into LLM inputs or representations (Section 4.2), and graph-guided reasoning frameworks in which explicit graph traversal or path selection steers LLM inference (Section 4.3). Figure 2 shows a general view of LLM-graph reasoning methods.

4.1. Graph Retrieval-Augmented Generation (RAG)

In a Direct LLM workflow, the model receives a user query and produces an answer solely based on its internal parametric knowledge. All factual recall, reasoning, and multi-step inference are implicitly encoded in the model's learned weights. While this approach offers a simple and efficient inference pipeline, it exhibits fundamental limitations. The model's knowledge is static and frozen at training time, performance degrades on domain-specific or long-tail queries, and multi-hop reasoning remains unreliable because intermediate relational constraints are not explicitly represented or enforced [31]. Consequently, Direct LLMs are prone to hallucination, brittle reasoning chains, and limited transparency, particularly when answering questions that require integrating multiple interdependent facts.

Retrieval-Augmented Generation (RAG) addresses the knowledge limitation of Direct LLMs by introducing an external retrieval step. Given a query, relevant documents or text chunks are retrieved from a corpus—typically using vector similarity—and supplied to the LLM as contextual grounding before generation. This design reduces hallucination by anchoring responses in external evidence and enables access to up-to-date or domain-specific information [32]. However, classical RAG treats retrieved content as a flat collection of independent passages, largely ignoring structural relationships among entities, facts, or documents. As a result, while RAG improves factual recall, it continues to struggle with multi-hop reasoning, compositional queries, and global logical consistency. In complex reasoning scenarios, relevant pieces of information may be retrieved, yet the model lacks an explicit mechanism to coherently chain them.

Graph Retrieval-Augmented Generation (GraphRAG) extends the RAG paradigm by explicitly modeling retrieved knowledge as a graph, where nodes represent entities, documents, or concepts, and edges encode semantic, temporal, causal, or relational dependencies. Rather than retrieving isolated text chunks, GraphRAG retrieves subgraphs or reasoning paths, preserving the relational structure required for multi-step inference [33, 34, 35]. This enables controlled traversal, neighborhood expansion, and constraint-aware reasoning, allowing the LLM to follow explicit relational pathways instead of implicitly inferring them from unstructured text.

In summary, GraphRAG can be viewed as a natural evolution of retrieval-augmented generation, where grounding

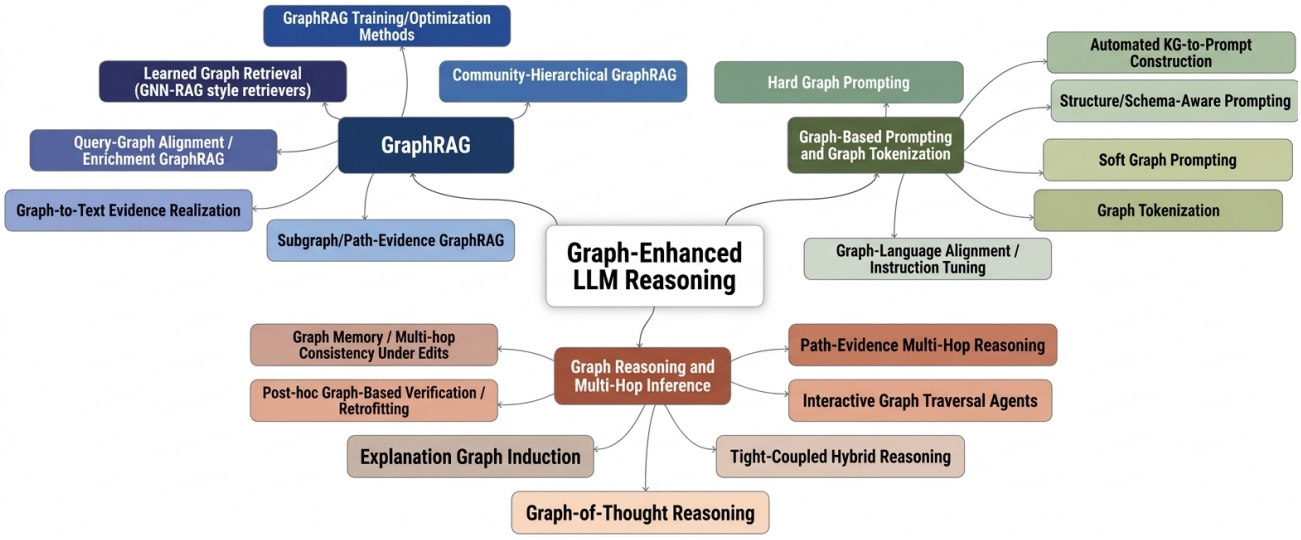

Figure 3: Graph-enhanced LLM reasoning and related graph categories

Table 1
Comparison of Direct LLM, RAG, and GraphRAG.

Paradigm	Core Workflow	Strengths	Weaknesses
Direct LLM	Query processed directly by LLM for answer generation	Static inference, low latency, strong language fluency	Static knowledge, hallucinations, weak multi-hop reasoning
RAG	Retrieve relevant documents and inject contextual evidence before LLM generation	External grounding, reduced hallucination, updatable corpus	Flat retrieval, no relational structure, weak compositional reasoning
GraphRAG	Graph-aware retrieval followed by subgraph reasoning with LLM	Structure-aware retrieval, explicit multi-hop reasoning, interpretability	Graph construction cost, higher system complexity, scalability challenges

alone is insufficient and explicit relational structure is required to support reliable, interpretable, and generalizable reasoning by large language models. The key differences in workflow design, strengths, and limitations among Direct LLMs, RAG, and GraphRAG are summarized in Table 1.

5. Hybrid GNN-LLM Models

Recent advances in integrating Large Language Models (LLMs) with Graph Neural Networks (GNNs) have led to a rapidly evolving research landscape. Existing works can be broadly categorized into four paradigms based on their interaction mechanisms and architectural designs: (1) collaboration frameworks, (2) directional integrations, (3) explainability-enhanced hybrids, and (4) pre-trained hybrid architectures. This taxonomy is consistent with recent surveys that categorize LLM-graph integration based on coupling strategies between language and graph models [36].

Figure 6 provides a conceptual overview of these paradigms by contrasting hybrid GNN-LLM collaboration frameworks with the emerging LLM-as-GNN paradigm. Hybrid approaches explicitly combine structural inductive bias with semantic reasoning, whereas LLM-as-GNN methods implicitly encode graph structure into textual sequences and rely on attention mechanisms for reasoning.

A unified comparison of representative methods across all paradigms is provided in Table 2.

5.1. Collaboration Frameworks

Collaboration frameworks enable bidirectional interaction between LLMs and GNNs through iterative or cooperative mechanisms. These approaches can be categorized into: (1) iterative co-training, (2) LLM-guided supervision, and (3) graph structure refinement.

Iterative Co-training: Iterative co-training frameworks establish feedback loops between LLMs and GNNs. Representative methods such as GLEM [37] and LOGIN [38] alternately refine graph representations and predictions using both semantic and structural signals.

LLM-Guided Supervision : In this setting, LLMs provide pseudo-labels, rationales, or soft supervision signals. Distillation-based approaches [39] enable knowledge transfer from LLMs to GNNs, improving performance in low-resource settings.

Graph Structure Refinement : These approaches leverage LLMs to modify graph topology. GraphEdit [40] is a representative example that refines graph connectivity to improve downstream learning.

5.2. Directional Integrations: LLM4GNN and GNN4LLM

Directional integration represents unidirectional interaction, where either LLMs enhance graph learning (LLM4GNN) or graph structures enhance LLM reasoning (GNN4LLM).

5.2.1. LLM4GNN

LLM4GNN approaches can be divided into: (1) feature augmentation, (2) pseudo-labeling and supervision, and (3) graph structure enhancement.

Feature augmentation methods (e.g., GLEM [37], LLaGA [41]) improve node representations using LLM embeddings. Pseudo-labeling approaches (e.g., LOGIN [38], distillation [39]) provide supervision signals. Structure enhancement methods (e.g., [42], SaVe-TAG [43]) refine graph topology or generate data.

5.2.2. GNN4LLM

GNN4LLM approaches can be categorized into: (1) graph retrieval, (2) graph-guided prompting, and (3) instruction-tuned reasoning. GNN-RAG [44] enables graph-based retrieval for reasoning, while LLaGA [41] supports graph-to-text encoding. InstructGraph [45] enables graph-aware instruction tuning for zero-shot reasoning.

5.3. Explainability-Enhanced Hybrid Models

These models aim to improve interpretability by combining graph reasoning with natural language explanations.

They can be categorized into: (1) natural language explanation generation, (2) rationale-guided learning, and (3) structure-aware explanation modeling.

Gspell [46] generates both textual explanations and supporting subgraphs. Distillation approaches [39] use rationales as supervision signals. Structure-aware models such as LLaGA [41] ensure explanations are grounded in graph topology.

5.4. Pre-trained Hybrid Architectures

Pre-trained hybrid architectures aim to learn general-purpose representations across graph and text modalities.

These approaches can be categorized into: (1) instruction-tuned graph LLMs, (2) graph-language co-pretraining, and (3) graph-native foundation models.

Instruction-tuned models (GraphGPT [47], InstructGraph [45], HiGPT [48]) adapt LLMs for graph reasoning tasks. Co-pretraining approaches (LLaGA [41]) align graph and text representations. Graph-native models (GOFA [49], GDL4LLM [50]) integrate graph computation directly into LLM architectures.

6. Knowledge Graph Question Answering (KGQA)

6.1. Training-Free and Low-Resource KGQA

Training-free and low-resource KGQA methods aim to reduce reliance on large annotated datasets and costly fine-tuning procedures. These approaches leverage prompt-based reasoning, symbolic graph traversal, or lightweight adaptation techniques to enable question answering over knowledge graphs with minimal supervision, making them suitable for domain-specific or rapidly evolving knowledge graphs.

Representative systems demonstrate that LLMs can perform dynamic reasoning, link prediction, and multi-hop inference without full retraining. Although their performance may not always match fully supervised models, these methods significantly lower deployment barriers and highlight the practicality of LLM-based KGQA in data-scarce settings.

In [51], the authors focus on building a training-free question answering system over a dynamic knowledge graph tailored to the financial domain. They propose FinQA, which constructs a dynamic finance knowledge graph partitioned by update frequency, designs a training-free pipeline that parses natural language questions into graph query language (NL2GQL), and integrates an open-source large language model for revision to improve parsing accuracy. The method targets the challenge of frequent data updates and scarce annotated training data in financial KGQA and evaluated on real-world finance KGQA tasks, demonstrating strong performance in dynamic finance question answering.

In [52], the authors focus on improving knowledge graph question answering (KGQA) in low-resource settings by enhancing large language models (LLMs) through fine-tuning with generative adversarial imitation learning (GAIL). They propose a framework that uses GAIL to guide the LLM's reasoning behavior, aiming to generate more accurate answers over a knowledge graph when annotated training data is scarce. The method is evaluated on benchmark KGQA datasets to demonstrate its effectiveness in low-resource QA scenarios.

In [53], the authors focus on improving multi-hop link prediction in knowledge graphs by introducing a framework called KG-LLM, which leverages large language models (LLMs) to convert structured graph data into natural language prompts and fine-tune the LLMs to reason over graph connections. They transform graph paths into chain-of-thought-style natural language prompts and use instruction fine-tuning (and optionally in-context learning) to train models such as Flan-T5, LLaMA2, and Gemma to predict whether distant entities are connected. The method is evaluated on standard multi-hop link prediction benchmark datasets such as WN18RR and NELL-995, demonstrating improved generalization and prediction accuracy in unfamiliar scenarios

Table 2
Global comparison of representative GNN-LLM hybrid models across paradigms.

Method	Year	Paradigm	Subtype	Interaction	Key Mechanism	Strength
GLEM [37]	2022	Collaboration / LLM4GNN	Co-training	Bidirectional	EM-style optimization	Strong graph-text alignment
LOGIN [38]	2024	Collaboration / LLM4GNN	Co-training	Iterative	LLM as consultant	Selective knowledge injection
Distillation [39]	2024	Collaboration / Explainability	Supervision	One-way	Knowledge transfer	Efficient learning
GraphEdit [40]	2024	Collaboration	Structure refinement	Indirect	Graph editing via LLM	Improved graph quality
Robustness [42]	2025	LLM4GNN	Structure	One-way	Edge refinement	Robust learning
SaVe-TAG [43]	2024	LLM4GNN	Augmentation	One-way	Data generation	Handles long-tail data
GNN-RAG [44]	2024	GNN4LLM	Retrieval	One-way	Graph retrieval	Multi-hop reasoning
LLaGA [41]	2024	GNN4LLM / Explainability	Prompting	One-way	Graph-to-text encoding	Structured reasoning
InstructGraph [45]	2024	GNN4LLM / Pre-trained	Instruction	One-way	Instruction tuning	Zero-shot ability
Gspell [46]	2025	Explainability	NL + Structure	Hybrid	Text + subgraph explanation	Interpretability
GraphGPT [47]	2023	Pre-trained	Instruction	Unified	Graph instruction tuning	Generalization
HiGPT [48]	2024	Pre-trained	Instruction	Unified	Graph tokenization	Heterogeneous graphs
GOFA [49]	2024	Pre-trained	Graph-native	Unified	GNN inside LLM	Deep integration
GDL4LLM [50]	2025	Pre-trained	Graph-native	Unified	Graph-as-language	Efficient encoding

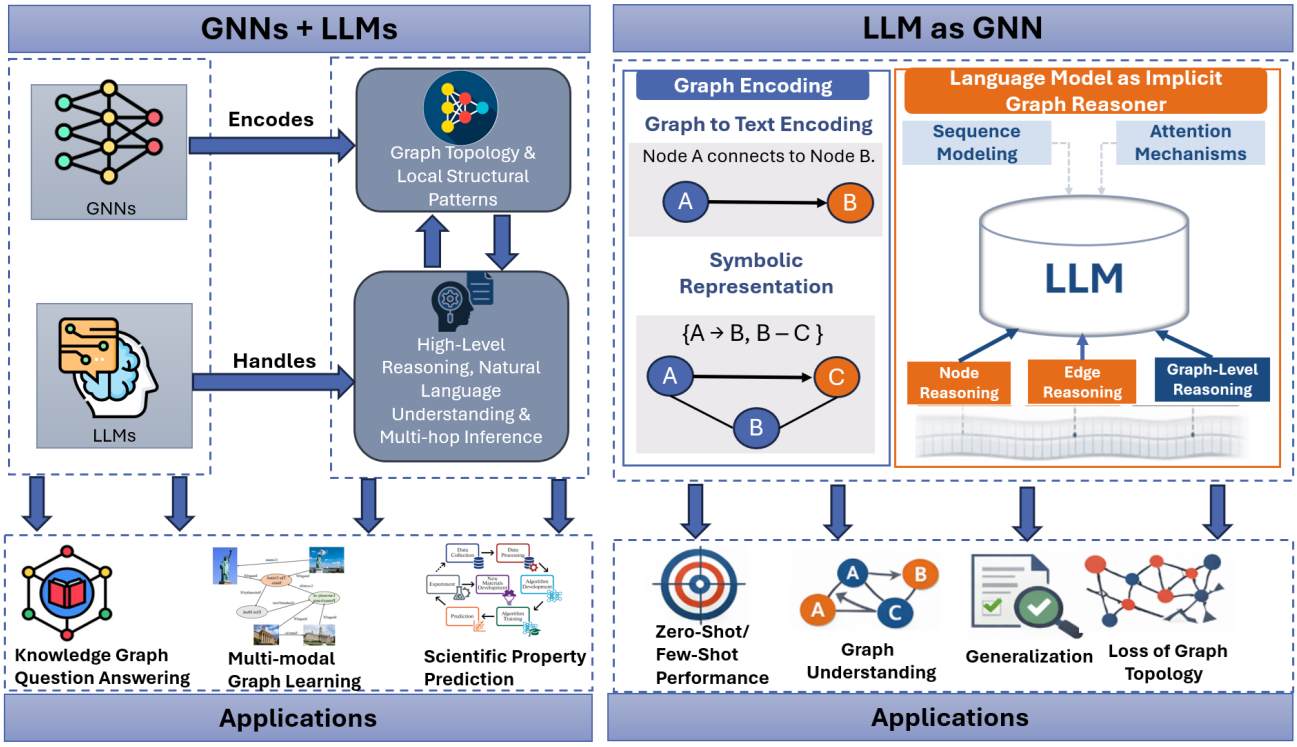

Figure 4: Conceptual comparison between hybrid GNN-LLM collaboration frameworks and the LLM-as-GNN paradigm. Hybrid approaches explicitly combine structural inductive bias with semantic reasoning, whereas LLM-as-GNN methods implicitly encode graph structures into textual representations and rely on attention mechanisms for reasoning.

6.2. LLM-Augmented KGQA Frameworks

LLM-augmented KGQA frameworks integrate large language models into the reasoning pipeline to enhance semantic parsing, candidate answer generation, and logical inference. In these systems, LLMs often act as reasoning agents that guide graph traversal, generate executable queries, or compensate for incompleteness and noise in knowledge graphs.

Experimental results indicate that LLM augmentation improves robustness and reasoning accuracy, particularly for complex, multi-hop, and domain-specific queries. Nevertheless, challenges remain in mitigating hallucination, ensuring faithfulness to graph evidence, and balancing generative flexibility with symbolic correctness, motivating ongoing research on tighter integration and verification mechanisms.

In this [54], the authors focus on designing a knowledge-graph question answering system tailored to materials science. They introduce KGQA4MAT, a framework that leverages knowledge graphs constructed for material entities and properties, coupled with pre-trained large language models to interpret and answer natural language questions related to materials data. The method combines structured graph retrieval with semantic language understanding to support complex queries over materials knowledge graphs, and the approach is evaluated on domain-specific KGQA tasks involving materials databases and related benchmarks.

In this [55], the authors focus on addressing question answering over incomplete knowledge graphs, where the knowledge graph does not contain all the triples needed to answer a question. They propose a training-free method called Generate-on-Graph (GoG) that treats a large language model (LLM) both as an agent that searches the graph and as a knowledge source that generates new factual triples to augment the graph. GoG uses a Thinking-Searching-Generating a framework, which the model iteratively explores the graph, identifies missing information, and synthesizes additional facts to enable correct answers. They evaluate their approach on two IKGQA (incomplete KG Question Answering) datasets constructed from existing KGQA benchmarks and show that GoG outperforms previous methods in this setting

In this [56], the authors focus on improving large language model (LLM) performance on knowledge graph question answering (KGQA) by reducing hallucinated or ungrounded reasoning results produced by generative LLMs. They propose a framework called READS that reformulates KGQA into a series of discriminative subtasks (including subgraph search, subgraph pruning, and answer inference) and designs a corresponding discriminative inference strategy to better guide reasoning over the knowledge graph. The approach is evaluated on widely used benchmark KGQA datasets, such as WebQSP and Complex Web Questions (CWQ), and achieves state-of-the-art performance compared with strong baselines

7. Scene Graphs and Large Language Models

Scene graphs (SGs) represent objects, relationships, and attributes existing in a scene [57, 58, 59]. LLMs can be used as advanced tools to facilitate scene-graph-centric tasks such as generation, editing, and navigation by playing different roles across the pipeline. In the following sections, we explain the different dimensions of the integration of LLM and SGs as shown in Figure 5.

7.1. Graph Type

The graph type is SGs can be viewed from spatial dimensionality and temporal setting. Based on spatial dimensionality, the graph could either be 2D or 3D. A 2D scene graph models relations in the image plane, while a 3D scene graph additionally encodes metric depth and object geometry, enabling physically grounded spatial reasoning [57, 58, 60]. Temporal setting describes whether or not the graph evolves over time. So, based on the temporal setting, the graph could either be static or dynamic. A static graph is constructed from a single observation and represents one fixed scene or time instant, although it may be internally refined during optimization. In contrast, a dynamic graph evolves as new or sequential observations update the scene representation [57, 58, 61, 62].

7.2. LLM's Role

LLMs play important roles in SG tasks. These roles include reasoning, parsing/translation, planning, and evaluation/verification.

In reasoning LLMs are used to infer semantics, relationships, or latent attributes, which can't be directly observed [5, 63]. In parsing/translation, LLMs are used to directly convert unstructured inputs (text or instructions) into structured presentations such as scene graphs, triplets, logic, or commands [64, 65, 61]. In planning, LLMs are used to make decisions, organize actions, select goals, or guide search [61, 66, 62]. In evaluation, LLMs are used to assess correctness, consistency, or alignment between representations [67, 68, 69, 70].

7.3. Main Objective

The main objectives of the integration of LLMs and SGs are divided into generation, editing, retrieval, and navigation.

7.3.1. Generation

In generation, the primary objective is to create a new structured representation or scene, typically a scene graph or a scene itself, from raw or unstructured inputs.

In [5], the authors propose SceneLLM for dynamic scene graph generation (SGG). The framework consists of three phases: video-to-language (v2l) mapping, reasoning with an LLM, and SG prediction. First, V2L maps the video frames to an implicit linguistic signal (a sequence of scene tokens) by discretizing object features and embedding spatial-temporal information. Then, in the second phase, they embed the linguistic tokens into a prompt and feed it to a LoRA fine-tuned LLM to generate reasoned hidden representations. Finally, in SG prediction, a transformer-based SGG predictor generates the final SG based on the hidden states from the previous step. The results demonstrate the state-of-the-art performance of SceneLLM in terms of `recall@K` for the Action Genome (AG) dataset across various tasks.

In [71], the authors propose SceneCraft for static-3D SGG by transforming a text query into a 3D scene. This pipeline consists of four phases: asset retrieval and scene decomposition, SG construction, scene layout optimization, and library learning. In the first phase, an LLM makes a list of assets (3D objects) and their descriptions from a text query. Then for each asset the top-10 3D models are retrieved from a large depository and the one with the highest text-to-image score is kept as final. Then, SceneCraft breaks crowded scenes into smaller sub-scenes to solve the problem more efficiently. In the second phase, a layout matrix is constructed for every asset to put each of them in the correct location and orientation. This done by constructing a relational scene graph with help of LLM-Planner. In the third phase, SceneCraft uses a set of scoring functions to iteratively optimize the scene layout using a Multimodal LLM's feedback to obtain the final optimized scene.

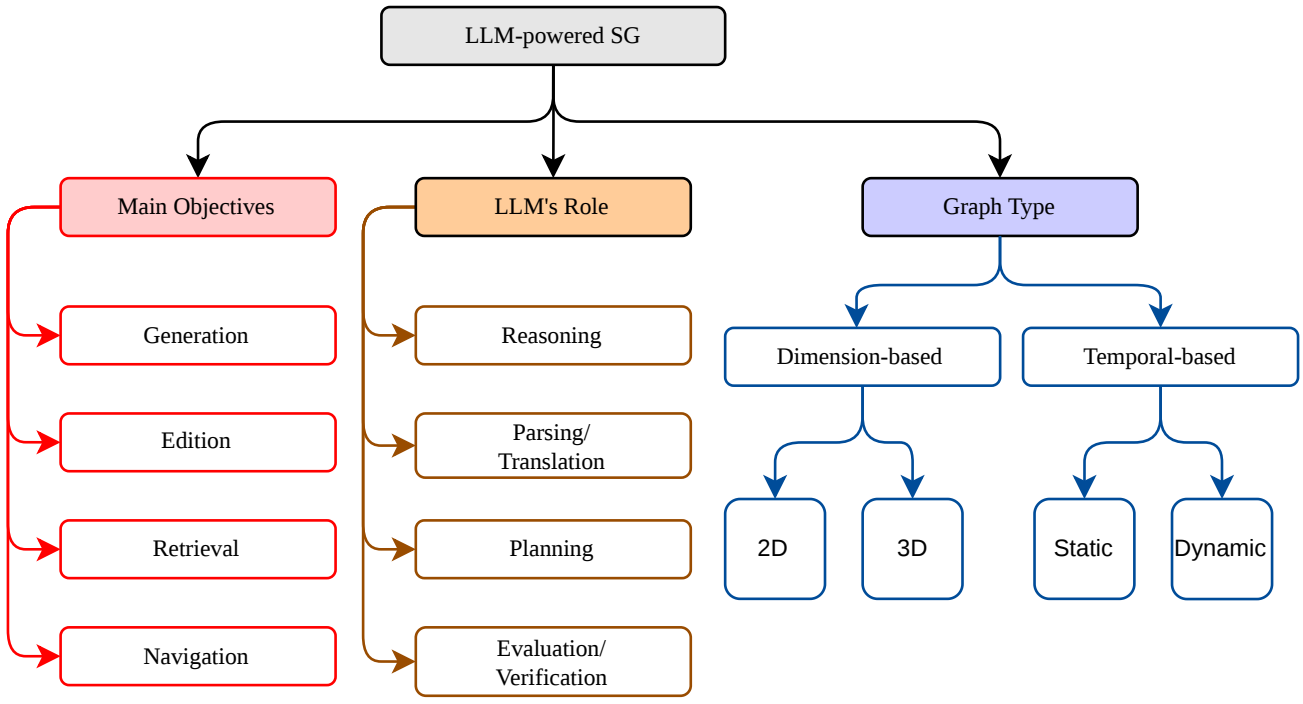

Figure 5: Taxonomy of LLM-enhanced scene graphs.

7.3.2. Editing

In addition, the goal of LLM is to modify an existing scene or scene graph to obtain a new configuration.

EditRoom [72] is a static-3D scene graph edition (SGE) framework, which is capable of six types of edits: rotate, translate, scale, replace, add, and remove. EditRoom receives a natural language text and the original scene as input to make the final scene; this includes two main modules: command parameterizer and scene editor. In the command parameterizer, the natural language command is converted into a set of edit types. After that, EditRoom uses graph diffusion models to find a target scene with the highest probability.

Scene graph edit (SGEdit) uses parsing and planning capabilities of LLMs to perform static-2D SGE tasks. SGEdit consists of two main stages: scene parsing and image editing. Given an input image, SGEdit generates a scene graph, masks, and descriptions using an LLM-driven scene parser. After that, a diffusion model is fine-tuned to learn each object's identity. In the next stage, the LLM acts as a planner to decide the changes and how to apply them, and a diffusion model performs editing tasks.

7.3.3. Retrieval

In retrieval, scene graphs are used as structured indices to search, match, rank, or query information, rather than to generate or modify content.

In [73], the authors propose Visual Triplet-based Graph Transformation (VTGT) for static-2D object retrieval from input images. The pipeline consists of two phases: graph transformation and graph representation learning. In phase one, VTGT transforms a scene graph into a triplet graph, where nodes include subject-relation-object triplets and edges represent similarity between the nodes; this is achieved by introducing local and global features for each triplet. Local features are used to keep the ordered sequence, while global features are attained by help of an LLM. The encodings for these two feature types are then merged to be used in the next phase. In the second phase, a graph attention network is used to update the representation of each triplet based on its top-k similar triplets. Finally, a single semantic embedding for the input image is generated, which is used for downstream retrieval tasks.

3DGraphLLM [60] is a static-3D method that takes point clouds of scene objects and builds a new scene representation to support object retrieval from natural language queries. It assigns each object both 2D features (appearance cues such as texture and color) and 3D features (shape and geometry). Then, semantic relations between object pairs are extracted using a vision-language spatial attention transformer and encoded as latent relation vectors. Finally, the objects and their relations are projected into the LLM token space and fed to an LLM (via object identifier tokens and a compact k-nearest-neighbor subgraph representation) to retrieve the target object from the scene.

7.3.4. Navigation

In navigation, scene graphs are used to guide an embodied agent's movement or task execution within an environment.

In [66], the authors propose Time is in my Sight (TioMS), which is an LLM-driven robot architecture for dynamic environments using continuously updated scene graphs. The

Table 3
Comprehensive comparison between state-of-the-art frameworks - reasoning (R1), parsing/translation (R2), planning (R3), and evaluation/verification (R4).

Research work	Year	Objective	Input	LLM's Role				Graph Type
				R1	R2	R3	R4	
SceneLLM [5]	2026	Generation	Video	✓	✓	✗	✗	Dynamic-2D
SceneCraft [71]	2024	Generation	Text	✓	✓	✓	✓	Static-3D
LLM4SGG [64]	2024	Generation	Image+Text	✓	✓	✗	✗	Static-2D
SDSGG [74]	2024	Generation	Image	✓	✗	✗	✗	Static-2D
PASGG-LM [75]	2024	Generation	Image	✓	✗	✗	✗	Static-2D
Planner3D [76]	2025	Generation	Graph/Text	✓	✗	✗	✗	Static-3D
LLaVA-SpaceSGG [77]	2025	Generation	Image	✓	✓	✗	✗	Static-2D
GraLa3D [65]	2024	Generation	Text	✗	✓	✓	✗	Static-3D
IntegraPSG [78]	2025	Generation	Image	✓	✗	✗	✗	Static-2D
SGFormer [63]	2024	Generation	Point cloud	✓	✗	✗	✗	Static-3D
ELEGANT [69]	2023	Generation	Image	✓	✓	✗	✓	Static-2D
HRSGL [79]	2024	Editing	3D scene	✓	✗	✓	✓	Static-3D
SGEdit [80]	2024	Editing	Image	✓	✓	✓	✗	Static-2D
EditRoom [72]	2025	Editing	3D Scene+Text	✗	✓	✓	✗	Static-3D
ScanEdit [81]	2025	Editing	3D Scene+Text	✓	✗	✓	✗	Static-3D
SAKR-Edit [70]	2025	Editing	Image+Text	✓	✗	✗	✗	Static-2D
VTGT [73]	2025	Retrieval	Image	✓	✗	✗	✗	Static-2D
3DGraphLLM [60]	2025	Retrieval	3D Scene	✓	✗	✗	✗	Static-3D
TioMS [66]	2024	Navigation	Video	✗	✓	✓	✗	Dynamic-3D
OSGP [61]	2024	Navigation	Graph	✗	✓	✓	✗	Static-3D
SG-Nav [62]	2024	Navigation	Video	✓	✗	✓	✗	Dynamic-3D

pipeline has two main modules: perception and planning. In the perception module, RGB-D frames are processed by a scene graph generator to detect objects and relationships, then object positions are estimated in 3D using robot/camera pose, and a particle filter refines tracking and localization across frames before updating the semantic map. In the planning module, the LLM uses the updated semantic map to translate natural-language commands into robot skills and replans when execution fails.

In [62], the authors propose scene graph navigation (SG-Nav) for zero-shot object-goal navigation by using an online 3D scene graph as the main representation for LLM-based planning. This pipeline consists of three main steps: online scene graph construction, LLM-guided exploration, and re-perception. In the first step, the agent incrementally builds a hierarchical 3D scene graph from RGB-D observations and prunes unreliable edges using geometric constraints and VLM-based checks. In the second step, the LLM scores graph regions to select promising exploration frontiers for searching the target object. In the third step, when the target object is detected, SG-Nav re-observes it from multiple views to confirm the detection and reduce false positives before finalizing the navigation result.

Table 3 illustrates a comprehensive comparison between state-of-the-art research works on LLM-powered SGs by providing details such as objective, LLM's role, and graph type.

8. Graph-Agent-LLM Integration

The authors in [82] focused on developing LAFA (LLM-Agentic Federated Analytics), a hierarchical multi-agent framework that enables complex natural-language analytics over decentralized data while preserving privacy. Instead of centralizing raw data, their system uses a structured agent pipeline—consisting of a coarse-grained planner, fine-grained planner, DAG optimizer, and answerer agent—to translate user queries into optimized federated analytics execution plans. The framework leverages Large Language Models (LLMs) to decompose and structure analytical tasks, while federated analytics backends handle secure aggregation and privacy mechanisms such as encryption and differential privacy. For evaluation, the authors used the AdultPii dataset (32,563 records with 18 features) and a benchmark of 20 complex natural-language analytics queries generated using GPT-4o based on realistic privacy scenarios (including Apple privacy reports). The results demonstrate improved semantic parsing accuracy, higher execution success rates than baseline prompting strategies, and a significant reduction in redundant federated operations through DAG optimization, leading to greater efficiency while maintaining

strict privacy guarantees.

The authors in [83] focused on developing X-GridAgent. This LLM-powered agentic AI system automates complex power grid analysis using natural-language queries while integrating domain-specific tools and structured databases for rigorous engineering computations. Their model uses a three-layer hierarchical architecture (planning, coordination, and action layers) in which the planning layer interprets user intent and generates structured workflows, the coordination layer manages task execution and memory, and the action layer interfaces with professional tools (e.g., Pandapower modules) to perform analyses like power flow, contingency analysis, optimal power flow, short-circuit calculations, and topology searches—all grounded in engineering rather than pure text generation. To enhance performance, they introduced two novel algorithms: LLM-driven prompt refinement with human feedback and schema-adaptive hybrid retrieval-augmented generation (RAG) for accurate retrieval from large structured grid datasets. While the paper doesn't use a traditional machine-learning dataset, evaluations were conducted using various power grid cases and user query scenarios to demonstrate that X-GridAgent can interpret diverse natural-language requests and generate interpretable, tool-invoked analytical results with high reliability and flexibility. Results show the system effectively automates interpretable power system analysis and handles previously unseen tasks in a modular, extensible way, significantly reducing manual effort and domain expertise requirements.

The authors in [84] focused on improving GraphRAG (graph-based retrieval-augmented generation) by introducing GraphSearch, an agentic deep searching workflow that integrates dual-channel retrieval over both semantic text chunks and structural graph knowledge, with iterative, multi-turn reasoning and modular stages to better surface relevant evidence and leverage graph structure. GraphSearch enhances how large language models interact with and reason over structured graph data, addressing limitations of shallow retrieval and inefficient use of graph information. They evaluated GraphSearch across six multi-hop RAG benchmarks, demonstrating consistently higher answer accuracy and generation quality compared to traditional RAG strategies, confirming the effectiveness of their agentic approach to graph retrieval-augmented generation. In more detail, we provided recent integration agentic models with LLM and graphs in Table 4

9. Applications

9.1. Cybersecurity and Malware Analysis

The integration of large language models (LLMs) with graph-based representations has emerged as a powerful paradigm in cybersecurity and malware analysis. Graph structures such as control-flow graphs, call graphs, and dependency graphs capture rich structural relationships that are difficult to obfuscate, while LLMs provide high-level semantic understanding and reasoning capabilities. By grounding language-based inference in explicit program and interaction structures, these hybrid approaches enhance robustness against code obfuscation, polymorphism, and rapidly evolving attack strategies.

Recent studies demonstrate that graph-enhanced LLM frameworks can uncover latent malicious behaviors by reasoning over structural patterns rather than relying solely on surface-level code features. For example, cluster-aware graph modeling combined with LLM-assisted recovery improves the detection of obfuscated JavaScript malware, while graph-guided vulnerability detection systems focus LLM reasoning on security-critical execution paths. These methods consistently improve detection accuracy and reduce false positives, while also offering interpretable insights into the underlying causes of security alerts.

Beyond malware detection, graph–LLM integration has been successfully applied to identifying adversarial and automated behaviors in online ecosystems. Graph-based representations of user interactions and network behaviors provide contextual signals that complement LLM-based semantic analysis, enabling the detection of coordinated or LLM-driven bot activity. Collectively, these advances highlight the effectiveness of graph-augmented LLMs as a scalable, explainable, and resilient foundation for next-generation cybersecurity solutions.

In [99], the authors focus on detecting malicious JavaScript code that has been intentionally obfuscated. They propose a hybrid method called DeCoda that combines large language model (LLM)-based deobfuscation with a cluster-aware code graph learning approach. The LLM is used to progressively reconstruct and normalize obfuscated JavaScript into Abstract Syntax Trees (ASTs), and then a hierarchical graph model with node-to-cluster attention captures both local and global structural relationships in the code graphs. The framework is evaluated on two benchmark malicious JavaScript datasets, where it achieves significantly higher detection performance (e.g., F1-scores around 94–97%) compared with state-of-the-art baselines.

In [100], the authors focus on improving software vulnerability detection by enhancing large language models (LLMs) with structural code information and in-context learning. They propose a framework called GRACE, which integrates graph structure representations (including code graph information such as abstract syntax trees, program dependence graphs, and control flow graphs) into LLM prompting and uses a demonstration selection module that retrieves similar code examples based on semantic, syntactic, and lexical similarity. The enhanced vulnerability detection module combines domain knowledge, graph-structure prompts, and in-context examples to guide the LLM in identifying vulnerabilities more accurately. The method is evaluated on three widely studied vulnerability detection benchmark datasets (Reveal, FFmpeg+Qemu, and Big-Vul), and GRACE significantly outperforms existing token-based and graph-based baselines in terms of F1 score.

Table 4
Comparison of Graph, Agent, and LLM-based frameworks across architecture, evaluation, and contributions.

Paper / Model	Category	Core Idea	Architecture / Method	Data / Evaluation	Key Results / Contribution
Agent-as-a-Judge [85]	Evaluation Framework	Multi-agent evaluation paradigm	Planning + tool verification + debate mechanisms	Cross-domain survey	Roadmap for robust and verifiable LLM evaluation
MAGMA [86]	Agent Memory Architecture	Multi-graph structured memory	Semantic, temporal, causal graph views + policy-guided traversal	Scenario-based evaluation	Improved interpretability and long-context reasoning
Graph-S3 [87]	Graph Retrieval Training	Stepwise supervision for graph reasoning	Synthetic supervision pipeline for iterative retrieval	WebQSP, CWQ, MetaQA	Improved accuracy and F1 over baselines
AgenticMath [88]	Data-Centric Reasoning	Multi-agent math data generation	Filtering + paraphrasing + CoT augmentation	AgenticMathQA (30K–90K)	Matches/surpasses larger baselines via high-quality data
AriGraph [89]	Memory Graph Agent	Episodic + semantic world model	Dynamic KG construction + structured memory updates	TextWorld + multi-hop QA	Outperforms full-history and RAG baselines
AgentBench [90]	Agent Benchmark	Standardized agent evaluation suite	Multi-task benchmark (planning, reasoning, tool use)	Synthetic + real-world tasks	Reveals weaknesses in long-horizon reasoning
Self-Refine [91]	Iterative Refinement	Self-critique loop for improvement	Draft → Critique → Refine pipeline	CNN/DM, HumanEval	Improved generation quality vs single-pass
ToolStorm [92]	Tool Ecosystem	Standardized multimodal tool integration	Unified APIs + tool chaining	Simulated + real tasks	Higher task success and composability
COT-PLUS [93]	Structured Reasoning	Explicit planning + subgoal reasoning	Plan → Decompose → Recursive solve	Math, logic, QA	Outperforms vanilla CoT in deep reasoning
GraphAgents [94]	Scientific Design Agent	KG-guided cross-domain reasoning	Specialized agents + graph traversal	PFAS-free materials case	Multi-agent > single-shot prompting
ReaGAN [95]	Graph Learning Agent	Node-as-agent reasoning	LLM-guided node planning + RAG + GNN	Cora, Citeseer, Chameleon	84.95 / 60.25 / 43.80 accuracy
GraphCodeAgent [96]	Repo-Level Code Agent	Dual-graph guided code generation	Requirement Graph + Structural Code Graph	DevEval, CoderEval	Outperforms repo-level RAG baselines
AgentVNE [97]	Graph RL + LLM	Affinity-aware agent placement	LLM semantic inference + Graph RL	Simulation-based	<40% latency, 5–10% acceptance gain
AgentEdge [98]	Edge Multi-Agent System	Agentic orchestration in edge-cloud	PARES framework + ActSimCrit validation	Real-world orchestration	3.6× success, 2.8× API reduction

In [101], the authors focus on evaluating and enhancing the use of large language models (LLMs) for software vulnerability detection in source code. They systematically analyze how state-of-the-art LLMs can detect security flaws in C/C++ and other languages by comparing different prompting strategies, such as basic vulnerability queries, CWE-specific prompts, and data-flow-analysis-based prompts, to guide the models' reasoning on code semantics. The evaluation is conducted on multiple popular code vulnerability datasets (including Juliet, OWASP Juliet, CVEFixes, and others) to measure detection performance and F1-scores across vulnerability types. The results highlight both the strengths and limitations of current LLM methods for vulnerability detection and suggest directions for improving prompt design and context-aware reasoning.

9.2. Healthcare and Biomedical Knowledge Graphs

In healthcare and biomedical domains, the integration of LLMs with knowledge graphs (KGs) addresses key challenges related to data heterogeneity, domain specificity, and interpretability. Biomedical knowledge is inherently relational, involving complex interactions among genes, diseases, drugs, phenotypes, and clinical outcomes. Knowledge graphs provide a structured representation of these relationships, allowing LLMs to perform reasoning that is grounded in curated and evidence-based medical knowledge.

Recent systems illustrate how KG-enhanced LLMs can support advanced clinical and research tasks. Web-scale biomedical knowledge graphs enable interactive exploration of treatment pathways, molecular mechanisms, and clinical evidence, supporting decision-making in precision medicine. Similarly, LLM-powered biomedical assistants leverage phenotypic and genomic graphs to facilitate structured querying, hypothesis generation, and knowledge discovery across large and diverse biomedical datasets.

Graph-empowered LLMs have also shown promise in mental health assessment and disease monitoring. By modeling symptoms, behavioral signals, and clinical indicators as interconnected graph elements, these approaches improve the reliability and contextual awareness of LLM-based predictions. Overall, the synergy between knowledge graphs and LLMs enhances transparency, factual consistency, and

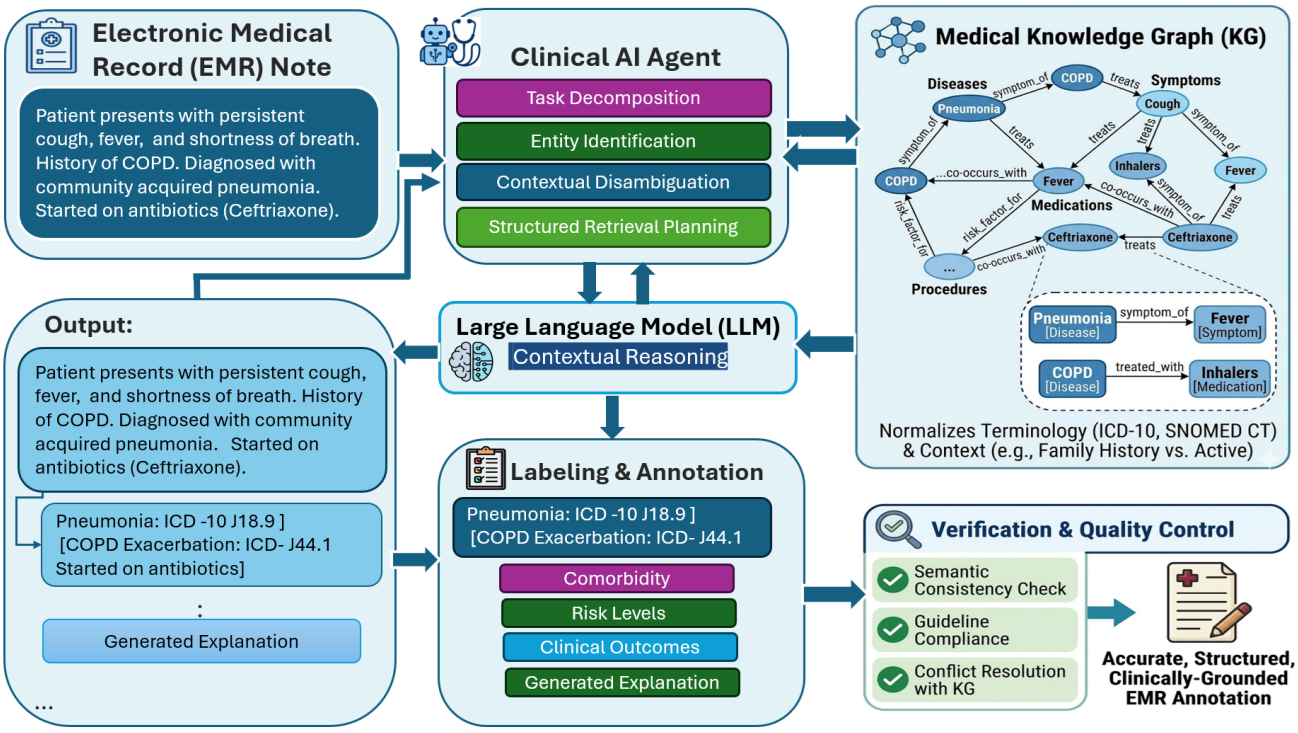

Figure 6: Graph-Agent-LLM integration for Electronic Medical Record (EMR) labeling

trustworthiness, which are essential for real-world healthcare and biomedical applications.

In [102], the authors focus on building a web-scale hybrid knowledge graph and large language model (KG-LLM) system to support interactive assistance with optimal cancer treatment and care. They propose CancerKG.ORG, an integrated platform that automatically ingests and organizes the latest peer-reviewed medical knowledge into a verifiable knowledge graph and combines it with an LLM for retrieval and reasoning. The knowledge graph acts as a guardrail to reduce hallucinations and ensure that responses are grounded in verified medical facts. The system is evaluated on real-world cancer treatment and clinical information tasks, demonstrating its ability to improve information retrieval quality compared with standalone LLM or KG methods

In [103], the authors focus on enabling non-expert users to interactively query and explore a large biomedical knowledge graph using natural language and large language models (LLMs). They introduce Phenomics Assistant, a prototype chat-based interface that translates user questions into calls to the Monarch Knowledge Graph (a comprehensive biomedical KG integrating gene, disease, and phenotype data), uses LLMs to interpret and refine queries, and returns summarized, factual responses grounded in the KG. The system is evaluated on benchmark gene-disease association and gene alias query tasks, showing that LLMs augmented with KG access produce significantly more accurate answers compared with standalone LLM responses

In [104], the authors focus on automatic depression detection from clinical interview recordings by modeling the full multi-modal interview data (questions, answer transcripts, audio, and video) as a structural element graph (SEGA). They introduce SEGA, a directed acyclic graph representation aligned with human expertise about interview elements, and combine it with principle-guided LLM-based data augmentation and graph contrastive learning to address data scarcity. The method is evaluated on two real-world depression corpora (in English and Chinese), and results show that SEGA significantly outperforms strong baselines and powerful LLMs like GPT-3.5 and GPT-4 on depression classification tasks.

9.3. Recommendation Systems

Recommendation systems increasingly adopt graph-enhanced LLMs to overcome the limitations of traditional collaborative filtering and neural recommendation approaches. User–item interactions, social relationships, and contextual dependencies are naturally represented as graphs, capturing both direct and higher-order relational information. When combined with the reasoning and conversational capabilities of LLMs, these graph structures enable more personalized, adaptive, and context-aware recommendations.

Recent research demonstrates that graph-augmented LLMs effectively model long-range dependencies and evolving user intent, particularly in conversational recommendation scenarios. Knowledge graphs and interaction graphs guide LLM reasoning by providing structured representations of user preferences, item attributes, and historical

behaviors. This integration improves performance in sparse-data settings and enhances robustness to noisy or incomplete interaction histories.

Moreover, graph–LLM integration significantly improves explainability in recommendation systems. By tracing paths and relationships within knowledge graphs, LLM-based recommenders can generate transparent and user-aligned explanations for their suggestions. These capabilities position graph-enhanced LLMs as a key enabler for trustworthy, interactive, and human-centered recommendation platforms.

In [105], the authors focus on improving recommender systems under sparse feedback by leveraging large language models (LLMs) to augment user-item interaction graphs. They propose LLMRec, a framework that uses three simple LLM-based graph augmentation strategies reinforcing interaction edges, enhancing item attribute understanding, and generating user profile information together with a denoising and robustification mechanism to refine augmented data. The approach is evaluated on benchmark recommendation datasets (e.g., MovieLens and Netflix), showing that LLM-augmented graphs lead to better recommendation performance compared with state-of-the-art baselines

In [106], the authors focus on reviewing the capabilities and current limitations of graph neural networks (GNNs) when applied to knowledge graph representation and reasoning. They systematically analyze why standard GNN architectures struggle with tasks like long-range reasoning, multi-hop inference, and compositional generalization over symbolic relational structures. The paper discusses theoretical limitations (e.g., expressiveness bounds) and practical challenges (e.g., scalability and heterogeneity) and surveys existing extensions and alternatives aimed at bridging the gap between GNNs and symbolic relational models. This work does not introduce a new dataset or a new algorithm but synthesizes empirical and theoretical findings from prior studies to outline future research directions.

9.4. Education, Law, and Governance

In education, law, and governance, graph-enhanced LLMs address the need for structured reasoning over complex, rule-driven, and highly interconnected knowledge sources. Legal frameworks, educational curricula, and policy documents are naturally suited to graph representations, where entities and regulations are linked through formal relationships. Integrating these graphs with LLMs enables accurate question answering and decision support grounded in authoritative and structured knowledge.

Recent applications show that graph-augmented LLM systems can effectively support regulatory compliance and legal analysis. For instance, graph-enhanced question-answering systems for AI governance leverage structured representations of legal texts to guide LLM reasoning during regulatory transitions. Anchoring LLM outputs to graph-based legal knowledge reduces hallucinations and improves factual reliability in high-stakes decision-making environments.

In education and public administration, LLM-enabled knowledge graph construction facilitates scalable knowledge access and intelligent assistance. Cross-domain educational knowledge graphs allow LLMs to align student queries with structured learning objectives, while service-domain graphs support efficient governance-oriented information retrieval. These applications underscore the value of graph–LLM integration in promoting transparency, accountability, and informed decision-making across societal institutions.

In [107], the authors focus on creating a structured knowledge base that represents the legal concepts, obligations, risk categories, and regulatory requirements introduced by the European Union's AI Act. They design an ontology-driven model to capture the semantics of the Act's clauses, duties for different stakeholder roles (e.g., providers, deployers, users), and relationships between risk levels and compliance obligations. The method combines legal analysis, ontology engineering, and knowledge graph construction to formalize legal norms into a machine-readable representation. They build and publish the resulting AI Act Knowledge Base (AI-KB) and demonstrate its use with example queries that show how structured representations can support compliance checking, automated reasoning, and stakeholder guidance. The report does not use "datasets" in the machine-learning sense but uses the text of the European AI Act and related legal documents as its primary source to extract and encode concepts into a knowledge graph.

In [108], the authors focus on building a knowledge graph from diverse educational data sources to support a large language model (LLM)-driven question-answering system in the context of higher education. They propose an automatic cross-data knowledge graph construction method that integrates structured and unstructured educational content (e.g., relational databases, text, APIs) into a unified KG and demonstrate how this KG can be used in conjunction with an LLM (such as ChatGPT) to answer educational queries. The approach is evaluated through a case study at Ho Chi Minh City University of Technology (HCMUT) on educational QA tasks, showing the effectiveness of the constructed knowledge graph for improving answer relevance and organization.

In this work, the authors focus on constructing a large-scale, open service domain knowledge graph to support service computing research. They propose BEAR, a service domain KG built using a comprehensive, manually designed service ontology and a zero-shot LLM-based knowledge extraction framework. The method automatically generates ontology-guided prompts and leverages large language models to extract high-quality entities, relations, and attributes from unannotated and heterogeneous data sources without supervised training. The resulting knowledge graph contains over 133k entities, 169k relations, and 424k factual attributes, and its feasibility and richness are demonstrated through empirical analysis rather than traditional ML benchmark datasets.

10. Open Challenges and Future Directions

Based on the surveyed literature and the integration patterns analyzed in this work, several open challenges

Table 5
When, why, where, and what to use for different graph-LLM integration paradigms.

Paradigm	What is it	When to Use	Why Use It	Where It Is Applied
LLM-Assisted Knowledge Graph Construction	LLMs extract entities and relations from text to automatically build knowledge graphs.	When large amounts of unstructured documents need to be converted into structured knowledge.	Reduces manual annotation and simplifies traditional multi-step extraction pipelines.	Scientific literature mining, enterprise knowledge bases, biomedical knowledge graphs.
Graph-Enhanced LLM Reasoning (GraphRAG)	LLMs reason over retrieved graph structures instead of flat text documents.	When tasks require multi-hop reasoning across related facts.	Graphs preserve relationships between facts and improve reasoning consistency.	Question answering, fact verification, domain-specific search systems.
Hybrid GNN–LLM Models	Combines Graph Neural Networks for structural learning with LLMs for semantic reasoning.	When both structural graph patterns and textual semantics are important.	Leverages strengths of both models: GNNs capture topology while LLMs understand language.	Molecular property prediction, recommender systems, knowledge graph learning.
KGQA with LLM	LLMs interpret questions and retrieve answers from knowledge graphs.	When users need natural language access to structured knowledge.	Provides interpretable reasoning over structured knowledge bases.	Search engines, enterprise assistants, legal and financial knowledge systems.
Scene Graph + LLM	LLMs reason over scene graphs representing objects and spatial relationships.	When tasks require visual scene understanding and spatial reasoning.	Scene graphs provide structured representations of visual environments.	Robotics, autonomous navigation, visual question answering, AR/VR systems.
Graph–Agent–LLM Integration	LLMs act as agents that plan tasks and interact with graph-based memory or workflows.	When solving complex multi-step tasks requiring planning and tool use.	Agents enable autonomous reasoning, coordination, and workflow execution.	Scientific discovery systems, automated analytics, engineering decision systems.

emerge that limit the practical deployment and theoretical understanding of graph-LLM systems. These challenges are closely tied to the design choices discussed throughout this survey, including graph construction, retrieval, reasoning, and hybrid GNN-LLM architectures.

10.1. Scalability

Scalability is a recurring challenge across nearly all graph-LLM integration paradigms reviewed in this survey. LLM-assisted knowledge graph construction methods (Section 3) often rely on document chunking, iterative prompting, and multi-stage validation pipelines, which incur high computational costs when applied to large corpora or continuously evolving data sources. Similarly, GraphRAG systems (Section 4.1) require graph-aware retrieval, subgraph extraction, or path expansion, which can become prohibitively expensive as graph size and reasoning depth increase.

Hybrid GNN-LLM models (Section 5) further amplify scalability concerns by combining graph neural message passing with LLM inference, leading to increased memory usage and inference latency. These limitations are particularly evident in large-scale knowledge graphs, program graphs in cybersecurity, and dynamic scene graphs in multimodal environments. Future research should focus on hierarchical graph abstraction, approximate or adaptive subgraph retrieval, and incremental graph updates to enable scalable reasoning without sacrificing structural fidelity.

10.2. Graph-LLM Alignment

A central challenge highlighted throughout this survey is the alignment between graph structure and LLM reasoning behavior. While graphs encode explicit relational constraints, LLMs operate in a latent language space and may not consistently respect graph topology, edge semantics, or logical dependencies. This misalignment is evident in several surveyed settings, including LLM-assisted KG construction (Section 3), where hallucinated relations or schema drift may occur, and LLM-augmented KGQA (Section 6), where generative reasoning can deviate from graph-grounded evidence.

Existing approaches address alignment through prompt engineering, graph serialization, or post-hoc verification; however, these solutions remain fragile and task-specific. Ensuring that LLM reasoning faithfully follows graph constraints, particularly in multi-hop and incomplete-graph scenarios, remains an open problem. Future directions include structure-aware tokenization, graph-constrained decoding, tighter neuro-symbolic coupling, and joint objectives that explicitly penalize graph-inconsistent generations.

10.3. Benchmark Gaps

Although several benchmarks for graph-LLM systems have been introduced (Section 10), the current evaluation landscape remains fragmented. Many benchmarks focus on isolated tasks such as node classification, link prediction, or factual QA, while under-representing complex reasoning scenarios emphasized in this survey, including multi-hop GraphRAG reasoning, dynamic knowledge graph updates, causal inference, and cross-modal scene graph understanding.

Moreover, most evaluations emphasize task accuracy while overlooking critical dimensions such as hallucination rates, faithfulness to graph evidence, reasoning path correctness, and robustness to graph incompleteness or noise. This gap is particularly evident in application-driven domains such as cybersecurity and healthcare, where reliability and interpretability are as important as predictive performance. Developing unified, multi-dimensional benchmarks that reflect real-world graph-LLM workloads remains an important direction for future research.

10.4. Trustworthy and Explainable Graph-LLM Models

Trustworthiness and explainability emerge as cross-cutting concerns across all surveyed applications. While graphs provide a natural substrate for interpretable reasoning through paths, subgraphs, and relational chains, LLM-based reasoning may still produce opaque or ungrounded explanations if not explicitly constrained. This issue is particularly critical in high-stakes domains such as cybersecurity, healthcare, law, and governance (Section 9).

Several works reviewed in this survey leverage graphs as guardrails to mitigate hallucinations, yet ensuring that generated answers are both correct and verifiably grounded in graph evidence remains challenging. Future research should emphasize faithful explanation generation, explicit reasoning trace extraction, uncertainty-aware inference, and graph-grounded verification mechanisms. Strengthening the trustworthiness of graph-LLM systems will be essential for their safe deployment in real-world decision-making pipelines.

11. Conclusion

This survey systematically examined the emerging landscape of graph-LLM integration, addressing the fundamental questions of when, why, where, and how graphs and large language models should be combined. By organizing prior work according to graph modality, integration strategy, and application objective, we provided a unified view of methods spanning LLM-assisted graph construction, graph-enhanced reasoning, hybrid GNN-LLM models, KGQA, scene graph understanding, and domain-specific applications.

Our analysis shows that graph-LLM integration offers clear advantages for tasks requiring structured reasoning, multi-hop inference, and factual grounding, but also introduces new challenges related to scalability, alignment, evaluation, and trustworthiness. Importantly, the effectiveness of graph-LLM methods depends on careful design choices rather than generic integration.

We believe that future progress in this field will be driven by tighter neuro-symbolic alignment, scalable graph-aware reasoning mechanisms, realistic evaluation benchmarks, and a strong emphasis on explainability and reliability. This survey aims to serve as both a reference and a design guide for researchers and practitioners seeking to build robust, structure-aware, and trustworthy graph-LLM systems.

CRediT authorship contribution statement

Hamed Jelodar: Conceptualization, Methodology, Writing. **Samita Bai:** Editing, Writing – Original Draft Preparation. **Mohammad Meymani:** Editing, Writing – Original Draft Preparation. **Parisa Hamedi:** Editing, Writing – Original Draft Preparation. **Roozbeh Razavi-Far:** Project Management, Conceptualization, Methodology, Writing – Original Draft Preparation. **Ali Ghorbani:** Project Management, Writing – Original Draft Preparation.

References

[1] C. Yang, C. Zhou, Y. Xiao, S. Dong, L. Zhuang, Y. Zhang, Z. Wang, Z. Hong, Z. Yuan, Z. Xiang, *et al.*, "Graph-based agent memory: Taxonomy, techniques, and applications," *arXiv preprint arXiv:2602.05665*, 2026.

[2] Q. He, J. Yu, and W. Wang, "Large language model-enhanced symbolic reasoning for knowledge base completion," *IEEE Transactions on Audio, Speech and Language Processing*, 2026.

[3] J. Zhu, Z. Chen, P. D. Meo, J. Guan, Z. Han, and W. Shi, "Knowpath: An llm-supported knowledge graph construction and path finding framework to explainable mooc recommendations," *ACM Transactions on Information Systems*, vol. 44, no. 2, pp. 1–28, 2026.

[4] X. Yang, R. Zhong, Y. Chen, G. Peng, D. Yao, C. Chen, C. Wang, D. Zhang, Y. Zhou, and Z. Yang, "Cti-thinker: an llm-driven system for cti knowledge graph construction and attack reasoning," *Cybersecurity*, vol. 9, no. 1, p. 106, 2026.

[5] H. Zhang, Z. Li, and J. Liu, "Scenellm: Implicit language reasoning in llm for dynamic scene graph generation," *Pattern Recognition*, vol. 170, p. 111992, 2026.

[6] M. Yu, S. Luo, and X. Chen, "Graphpilot: Gui task automation with one-step llm reasoning powered by knowledge graph," *arXiv preprint arXiv:2601.17418*, 2026.

[7] Y. Xiao, C. Zhou, Q. Zhang, B. Li, Q. Li, and X. Huang, "Reliable reasoning path: Distilling effective guidance for llm reasoning with knowledge graphs," *IEEE Transactions on Knowledge and Data Engineering*, 2026.

[8] X. Tong, K. Li, and J. Bao, "Gnn-llm hybrid cognitive architectures for generative task adaptation in multi-human multi-robot collaborative disassembly," *Robotics and Computer-Integrated Manufacturing*, vol. 98, p. 103169, 2026.

[9] H. Tao, Y. Zhang, Z. Tang, H. Peng, X. Zhu, B. Liu, Y. Yang, Z. Zhang, Z. Xu, H. Zhang, *et al.*, "Code graph model (cgm): A graph-integrated large language model for repository-level software engineering tasks," *arXiv preprint arXiv:2505.16901*, 2025.

[10] S. Ahn, S. Shin, and Y.-D. Seo, "Enriching semantic profiles into knowledge graph for recommender systems using large language models," *arXiv preprint arXiv:2601.08148*, 2026.

[11] S. Li, J. Gong, and A. A. Ramirez Molina, "Approximate knowledge graphs: Privacy-preserving healthcare data synthesis via llm-driven approximation," in *Proceedings of the ACM/IEEE International Conference on Connected Health: Applications, Systems and Engineering Technologies*, pp. 458–463, 2025.

[12] C. Park, H. Lee, S. Lee, and O. Jeong, "Synergistic joint model of knowledge graph and llm for enhancing xai-based clinical decision support systems," *Mathematics*, vol. 13, no. 6, p. 949, 2025.

[13] C. Liu, K. H. Hettige, Q. Xu, C. Long, S. Xiang, G. Cong, Z. Li, and R. Zhao, "St-llm+: Graph enhanced spatio-temporal large language models for traffic prediction," *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[14] R. Erlemann, C. C. Morris, and S. Sathe, "Full-stack knowledge graph and llm framework for post-quantum cyber readiness," *arXiv preprint arXiv:2601.03504*, 2026.

[15] G. Faye, W. Ouerdane, G. Gadek, S. Gatepaille, and C. Hudelot, "Tegra: Text encoding with graph and retrieval augmentation for misinformation detection," *arXiv preprint arXiv:2602.11106*, 2026.

[16] Y. Xu, A. Li, X. Guo, J. C. Chan, X. Gong, H. H. Kwok, and J. C. Cheng, "Automated carbon-aware assessment of openbim-based ductwork design using knowledge graph–augmented llm multi-agent framework," *Automation in Construction*, vol. 181, p. 106611, 2026.

[17] Z. Chen, Y. Liu, J. Liu, and W. Gao, "Combining llm semantic reasoning with gnn structural modeling for multi-view multi-label feature selection," *arXiv preprint arXiv:2511.08008*, 2025.

[18] V. H. G. Moia, I. J. Sanz, G. A. F. Rebello, R. D. de Meneses, B. Hitaj, and U. Lindqvist, "Llm in the middle: A systematic review of threats and mitigations to real-world llm-based systems," *Computer Science Review*, vol. 61, p. 100916, 2026.

[19] D. Li, B. Jiang, L. Huang, A. Beigi, C. Zhao, Z. Tan, A. Bhattacharjee, Y. Jiang, C. Chen, T. Wu, *et al.*, "From generation to judgment: Opportunities and challenges of llm-as-a-judge," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2757–2791, 2025.

[20] N. Mihindukulasooriya, S. Tiwari, C. F. Enguix, and K. Lata, "Text2kgbench: A benchmark for ontology-driven knowledge graph generation from text," in *International semantic web conference*, pp. 247–265, Springer, 2023.

[21] Q. Sun, Y. Luo, W. Zhang, S. Li, J. Li, K. Niu, X. Kong, and W. Liu, "Docs2kg: A human-llm collaborative approach to unified knowledge graph construction from heterogeneous documents," in *Companion Proceedings of the ACM on Web Conference 2025*, pp. 801–804, 2025.

[22] K. Gillani, E. Novak, K. Kenda, and D. Mladenić, "Knowledge graph extraction from textual data using llm," in *Information Society Conferences*, 2024.

[23] M. Trajanoska, R. Stojanov, and D. Trajanov, "Enhancing knowledge graph construction using large language models," *arXiv preprint arXiv:2305.04676*, 2023.

[24] L. Huang and X. Xiao, "Ctikg: Llm-powered knowledge graph construction from cyber threat intelligence," in *First Conference on Language Modeling*, 2024.

[25] V. K. Kommineni, B. König-Ries, and S. Samuel, "From human experts to machines: An llm supported approach to ontology and knowledge graph construction," *arXiv preprint arXiv:2403.08345*, 2024.

[26] L.-P. Meyer, C. Stadler, J. Frey, N. Radtke, K. Junghanns, R. Meissner, G. Dziwis, K. Bulert, and M. Martin, "Llm-assisted knowledge graph engineering: Experiments with chatgpt," in *Working conference on artificial intelligence development for a resilient and sustainable tomorrow*, pp. 103–115, Springer, 2023.

[27] A. Barua, R. Amini, S. S. Norouzi, R. Amini, and P. Hitzler, "Complex ontology alignment using llms: A case study," 2022.

[28] 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 preprint arXiv:2307.01128*, 2023.

[29] H. Sansford, N. Richardson, H. P. Maretic, and J. N. Saada, "Grapheval: A knowledge-graph based llm hallucination evaluation framework," *arXiv preprint arXiv:2407.10793*, 2024.

[30] L.-P. Meyer, J. Frey, K. Junghanns, F. Brei, K. Bulert, S. Gründer-Fahrer, and M. Martin, "Developing a scalable benchmark for assessing large language models in knowledge graph engineering," *arXiv preprint arXiv:2308.16622*, 2023.

[31] J. Wu, J. Zhu, Y. Qi, J. Chen, M. Xu, F. Menolascina, Y. Jin, and V. Grau, "Medical graph rag: Evidence-based medical large language model via graph retrieval-augmented generation," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28443–28467, 2025.

[32] Y. Hu, W. Xuan, Q. Zhou, Z. Li, Y. Li, J. Hu, and F. Fang, "A self-correcting agentic graph rag for clinical decision support in hepatology," *Frontiers in medicine*, vol. 12, p. 1716327, 2025.

[33] B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, Y. Zhang, and S. Tang, "Graph retrieval-augmented generation: A survey," *ACM Transactions on Information Systems*, vol. 44, no. 2, pp. 1–52, 2025.

[34] C. Mavromatis and G. Karypis, "Gnn-rag: Graph neural retrieval for efficient large language model reasoning on knowledge graphs," in *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 16682–16699, 2025.

[35] S. Li, Z. Liu, Z. Gui, H. Chen, and W. Zhang, "Enrich-on-graph: Query-graph alignment for complex reasoning with llm enriching," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 7683–7703, 2025.

[36] X. Ren, J. Tang, D. Yin, N. Chawla, and C. Huang, "A survey of large language models for graphs," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 6616–6626, 2024.

[37] J. Zhao, M. Qu, C. Li, H. Yan, Q. Liu, R. Li, X. Xie, and J. Tang, "Learning on large-scale text-attributed graphs via variational inference," *arXiv preprint arXiv:2210.14709*, 2022.

[38] Y. Qiao, X. Ao, Y. Liu, J. Xu, X. Sun, and Q. He, "Login: A large language model consulted graph neural network training framework," in *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pp. 232–241, 2025.

[39] B. Pan, Z. Zhang, Y. Zhang, Y. Hu, and L. Zhao, "Distilling large language models for text-attributed graph learning," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 1836–1845, 2024.

[40] Z. Guo, L. Xia, Y. Yu, Y. Wang, K. Lu, Z. Huang, and C. Huang, "Graphedit: Large language models for graph structure learning," *arXiv preprint arXiv:2402.15183*, 2024.

[41] R. Chen, T. Zhao, A. Jaiswal, N. Shah, and Z. Wang, "Llaga: Large language and graph assistant," *arXiv preprint arXiv:2402.08170*, 2024.

[42] Z. Zhang, X. Wang, H. Zhou, Y. Yu, M. Zhang, C. Yang, and C. Shi, "Can large language models improve the adversarial robustness of graph neural networks?," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 2008–2019, 2025.

[43] L. Wang, Y. Wang, B. Ni, Y. Zhao, H. Wang, Y. Ma, and T. Derr, "Save-tag: Llm-based interpolation for long-tailed text-attributed graphs," *arXiv preprint arXiv:2410.16882*, 2024.

[44] C. Mavromatis and G. Karypis, "Gnn-rag: Graph neural retrieval for large language model reasoning," *arXiv preprint arXiv:2405.20139*, 2024.

[45] J. Wang, J. Wu, Y. Hou, Y. Liu, M. Gao, and J. McAuley, "Instructgraph: Boosting large language models via graph-centric instruction tuning and preference alignment," in *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13492–13510, 2024.

[46] P. Baghershahi, G. Fournier, P. Nyati, and S. Medya, "From nodes to narratives: Explaining graph neural networks with llms and graph context," 2025.

[47] J. Tang, Y. Yang, W. Wei, L. Shi, L. Su, S. Cheng, D. Yin, and C. Huang, "Graphgpt: Graph instruction tuning for large language models," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 491–500, 2024.

[48] J. Tang, Y. Yang, W. Wei, L. Shi, L. Xia, D. Yin, and C. Huang, "Higpt: Heterogeneous graph language model," in *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 2842–2853, 2024.

[49] L. Kong, J. Feng, H. Liu, C. Huang, J. Huang, Y. Chen, and M. Zhang, "Gofa: A generative one-for-all model for joint graph language modeling," *arXiv preprint arXiv:2407.09709*, 2024.

[50] H. Zhou, J. Du, C. Zhou, C. Yang, Y. Xiao, Y. Xie, and X. Huang, "Each graph is a new language: Graph learning with llms," in *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 17548–17559, 2025.

[51] W. Tao, H. Zhu, K. Tan, J. Wang, Y. Liang, H. Jiang, P. Yuan, and Y. Lan, "Finqa: A training-free dynamic knowledge graph question answering system in finance with llm-based revision," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 418–423, Springer, 2024.

[52] Z. Zhang, L. Wen, and W. Zhao, "A gail fine-tuned llm enhanced framework for low-resource knowledge graph question answering," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 3300–3309, 2024.

[53] D. Shu, T. Chen, M. Jin, C. Zhang, M. Du, and Y. Zhang, "Knowledge graph large language model (kg-llm) for link prediction," *arXiv preprint arXiv:2403.07311*, 2024.

[54] Y. An, J. Greenberg, F. J. Uribe-Romo, D. A. Gómez-Gualdrón, K. Langlois, J. Furst, A. Kalinowski, X. Zhao, and X. Hu, "Knowledge graph question answering for materials science (kgqa4mat)," in *Research Conference on Metadata and Semantics Research*, pp. 18–29, Springer, 2023.

[55] Y. Xu, S. He, J. Chen, Z. Wang, Y. Song, H. Tong, G. Liu, K. Liu, and J. Zhao, "Generate-on-graph: Treat llm as both agent and kg in incomplete knowledge graph question answering," *arXiv preprint arXiv:2404.14741*, 2024.

[56] J. Xu, Z. Wu, M. Lin, X. Zhang, and S. Wang, "Llm and gnn are complementary: Distilling llm for multimodal graph learning," *arXiv preprint arXiv:2406.01032*, 2024.

[57] H. Li, G. Zhu, L. Zhang, Y. Jiang, Y. Dang, H. Hou, P. Shen, X. Zhao, S. A. A. Shah, and M. Bennamoun, "Scene graph generation: A comprehensive survey," *Neurocomputing*, vol. 566, p. 127052, 2024.

[58] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. Hauptmann, "A comprehensive survey of scene graphs: Generation and application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 1–26, 2021.

[59] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. Hauptmann, "Scene graphs: A survey of generations and applications," *arXiv preprint arXiv:2104.01111*, vol. 2, 2021.

[60] T. Zemskova and D. Yudin, "3dgraphllm: Combining semantic graphs and large language models for 3d scene understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8885–8895, 2025.

[61] Z. Dai, A. Asgharivaskasi, T. Duong, S. Lin, M.-E. Tzes, G. Pappas, and N. Atanasov, "Optimal scene graph planning with large language model guidance," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14062–14069, IEEE, 2024.

[62] H. Yin, X. Xu, Z. Wu, J. Zhou, and J. Lu, "Sg-nav: Online 3d scene graph prompting for llm-based zero-shot object navigation," *Advances in neural information processing systems*, vol. 37, pp. 5285–5307, 2024.

[63] C. Lv, M. Qi, X. Li, Z. Yang, and H. Ma, "Sgformer: Semantic graph transformer for point cloud-based 3d scene graph generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 4035–4043, 2024.

[64] K. Kim, K. Yoon, J. Jeon, Y. In, J. Moon, D. Kim, and C. Park, "Llm4sgg: Large language models for weakly supervised scene graph generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28306–28316, 2024.

[65] Y.-H. Huang, W. Wang, S.-Y. Huang, and Y.-C. F. Wang, "Toward scene graph and layout guided complex 3d scene generation," *arXiv preprint arXiv:2412.20473*, 2024.

[66] S. Colombani, L. Brini, D. Ognibene, and G. Boccignone, "Time is on my sight: scene graph filtering for dynamic environment perception in an llm-driven robot," *arXiv preprint arXiv:2411.15027*, 2024.

[67] Z. Chen, J. Wu, Z. Lei, and C. W. Chen, "What makes a scene? scene graph-based evaluation and feedback for controllable generation," *arXiv preprint arXiv:2411.15435*, 2024.

[68] D. Yang, M. Kim, S. Mac Kim, B.-w. Kwak, M. Park, J. Hong, W. Woo, and J. Yeo, "Llm meets scene graph: Can large language models understand and generate scene graphs? a benchmark and empirical study," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21335–21360, 2025.

[69] S. Zhao and H. Xu, "Less is more: Toward zero-shot local scene graph generation via foundation models," *arXiv preprint arXiv:2310.01356*, 2023.

[70] J. Wang, J. Li, Z. Ma, and R. Bai, "Sakr-edit: Scene-aware knowledge reasoning for text-to-image editing," in *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 10457–10466, 2025.

[71] Z. Hu, A. Iscen, A. Jain, T. Kipf, Y. Yue, D. A. Ross, C. Schmid, and A. Fathi, "Scenecraft: An llm agent for synthesizing 3d scenes as blender code," in *Forty-first International Conference on Machine Learning*, 2024.

[72] K. Zheng, X. Chen, X. He, J. Gu, L. Li, Z. Yang, K. Lin, J. Wang, L. Wang, and X. E. Wang, "Editroom: Llm-parameterized graph diffusion for composable 3d room layout editing," *arXiv preprint arXiv:2410.12836*, 2024.

[73] S. Jeong, J. Park, M. Choi, Y. Kwon, and S. Lim, "Llm-powered scene graph representation learning for image retrieval via visual triplet-based graph transformation," *Expert Systems with Applications*, vol. 286, p. 127926, 2025.

[74] G. Chen, J. Li, and W. Wang, "Scene graph generation with role-playing large language models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 132238–132266, 2024.

[75] D. Platnick, M. Alirezaie, and H. Rahnama, "Enabling perspective-aware ai with contextual scene graph generation," *Information*, vol. 15, no. 12, p. 766, 2024.

[76] Y. Wei, M. R. Min, G. Vosselman, L. E. Li, and M. Y. Yang, "Planner3d: Llm-enhanced graph prior meets 3d indoor scene explicit regularization," *IEEE transactions on pattern analysis and machine intelligence*, 2025.

[77] M. Xu, M. Wu, Y. Zhao, J. C. L. Li, and W. Ou, "Llava-spacesgg: Visual instruct tuning for open-vocabulary scene graph generation with enhanced spatial relations," in *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 6362–6372, IEEE, 2025.

[78] Y. Zhao, Q. Zhang, X. Sun, and G. Liu, "Integrapsg: Integrating llm guidance with multimodal feature fusion for single-stage panoptic scene graph generation," *Electronics*, vol. 14, no. 17, p. 3428, 2025.

[79] W. Li, Z. Yu, Q. She, Z. Yu, Y. Lan, C. Zhu, R. Hu, and K. Xu, "Llm-enhanced scene graph learning for household rearrangement," in *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024.

[80] Z. Zhang, D. Chen, and J. Liao, "Sgedit: Bridging llm with text2image generative model for scene graph-based image editing," *arXiv preprint arXiv:2410.11815*, 2024.

[81] M. El Amine Boudjoghra, I. Laptev, and A. Dai, "Scanedit: Hierarchically-guided functional 3d scan editing," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 27105–27115, 2025.

[82] H. Ji, Z. Wang, C. Pan, M. Han, Y. Zhu, D. Wang, and Z. Han, "Lafa: Agentic llm-driven federated analytics over decentralized data sources," in *2025 lEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 1–8, IEEE, 2025.

[83] X. Chen *et al.*, "X-gridagent: An llm-powered agentic ai system for assisting power grid analysis," *arXiv preprint arXiv:2512.20789*, 2025.

[84] C. Yang, X. Wu, X. Lin, C. Xu, X. Jiang, Y. Sun, J. Li, H. Xiong, and J. Guo, "Graphsearch: An agentic deep searching workflow for graph retrieval-augmented generation," *arXiv preprint arXiv:2509.22009*, 2025.

[85] R. You, H. Cai, C. Zhang, Q. Xu, M. Liu, T. Yu, Y. Li, and W. Li, "Agent-as-a-judge," *arXiv preprint arXiv:2601.05111*, 2026.

[86] D. Jiang, Y. Li, G. Li, and B. Li, "Magma: A multi-graph based agentic memory architecture for ai agents," *arXiv preprint arXiv:2601.03236*, 2026.

[87] G. Chang, J. Su, J. Liu, P. Yang, Y. Shang, H. Zheng, H. Ma, Y. Liang, Y. Li, and Y. Liu, "Graph-s3: Enhancing agentic textual graph retrieval with synthetic stepwise supervision," *arXiv preprint arXiv:2510.03323*, 2025.

[88] X. Liu, Y. Liu, S. Wang, H. Cheng, A. Estornell, Y. Zhao, J. Shu, and J. Wei, "Agenticmath: Enhancing llm reasoning via agentic-based math data generation," *arXiv preprint arXiv:2510.19361*, 2025.

[89] P. Anokhin, N. Semenov, A. Sorokin, D. Evseev, A. Kravchenko, M. Burtsev, and E. Burnaev, "Arigraph: Learning knowledge graph world models with episodic memory for llm agents," *arXiv preprint arXiv:2407.04363*, 2024.

[90] S. Xia, Z. Xu, J. Chai, W. Fan, Y. Song, X. Wang, G. Yin, W. Lin, H. Zhang, and J. Wang, "From experience to strategy: Empowering llm agents with trainable graph memory," *arXiv preprint arXiv:2511.07800*, 2025.

[91] R. Zhao, S. Conia, E. Peng, M. Li, and S. Potdar, "Agree: Agentic reasoning for knowledge graph completion on emerging entities," *arXiv preprint arXiv:2508.04118*, 2025.

[92] Z. Zhang, X. Li, Y. Zuo, Y. Wen, Z. Fan, Z. Li, B. Zhou, R.-H. Li, and G. Wang, "When llm agents meet graph optimization: An automated data quality improvement approach," *arXiv preprint arXiv:2510.08952*, 2025.

[93] J. Dong, Z. Lin, W. Lin, and M. Zhang, "S-dag: A subject-based directed acyclic graph for multi-agent heterogeneous reasoning," *arXiv preprint arXiv:2511.06727*, 2025.

[94] I. A. Stewart, T. P. Hage, Y.-C. Hsu, and M. J. Buehler, "Graphagents: Knowledge graph-guided agentic ai for cross-domain materials design," *arXiv preprint arXiv:2602.07491*, 2026.

[95] M. Guo, X. Zhu, H. Xue, C. Zhang, S. Lin, J. Huang, Z. Ye, and Y. Zhang, "Reagan: Node-as-agent-reasoning graph agentic network," *arXiv preprint arXiv:2508.00429*, 2025.

[96] J. Li, X. Shi, K. Zhang, G. Li, Z. Jin, L. Li, H. Zhang, F. Liu, Y. Zhang, Z. Tao, *et al.*, "Graphcodeagent: Dual graph-guided llm agent for retrieval-augmented repo-level code generation," *arXiv preprint arXiv:2504.10046*, 2025.

[97] R. Zheng, Y. Zheng, Z. Cheng, L. Luo, H. Luo, G. Sun, H. Yu, and D. Niyato, "Agentvne: Llm-augmented graph reinforcement learning for affinity-aware multi-agent placement in edge agentic ai," *arXiv preprint arXiv:2601.02021*, 2026.

[98] Y. Lu, S. Zhang, C. Liu, R. Zhang, B. Ai, D. Niyato, W. Ni, X. Wang, and A. Jamalipour, "Agentic graph neural networks for wireless communications and networking towards edge general intelligence: A survey," *IEEE Communications Surveys & Tutorials*, 2026.

[99] Z. Liang, X. Wang, Z. Hu, L. Song, L. Chen, J. Guo, Y. Wang, and Y. Tian, "Breaking obfuscation: Cluster-aware graph with llm-aided recovery for malicious javascript detection," *arXiv preprint arXiv:2507.22447*, 2025.

[100] G. Lu, X. Ju, X. Chen, W. Pei, and Z. Cai, "Grace: Empowering llm-based software vulnerability detection with graph structure and in-context learning," *Journal of Systems and Software*, vol. 212, p. 112031, 2024.

[101] J. Duan, W. Li, Q. Bai, M. Nguyen, X. Wang, and J. Jiang, "Llm-botguard: A novel framework for detecting llm-driven bots with mixture of experts and graph neural networks," *IEEE Transactions on Computational Social Systems*, 2025.

[102] M. Gubanov, A. Pyayt, and A. Karolak, "Cancerkg. org-a web-scale, interactive, verifiable knowledge graph-llm hybrid for assisting with optimal cancer treatment and care," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 4497–4505, 2024.

[103] S. T. O'Neil, K. Schaper, G. Elsarboukh, J. T. Reese, S. A. Moxon, N. L. Harris, M. C. Munoz-Torres, P. N. Robinson, M. A. Haendel, and C. J. Mungall, "Phenomics assistant: An interface for llm-based biomedical knowledge graph exploration," *bioRxiv*, pp. 2024–01, 2024.

[104] Z. Chen, J. Deng, J. Zhou, J. Wu, T. Qian, and M. Huang, "Depression detection in clinical interviews with llm-empowered structural element graph," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8181–8194, 2024.

[105] W. Wei, X. Ren, J. Tang, Q. Wang, L. Su, S. Cheng, J. Wang, D. Yin, and C. Huang, "Llmrec: Large language models with graph augmentation for recommendation," in *Proceedings of the 17th ACM international conference on web search and data mining*, pp. 806–815, 2024.

[106] Z. Chen, Z. Jiang, F. Yang, E. Cho, X. Fan, X. Huang, Y. Lu, and A. Galstyan, "Graph meets llm: A novel approach to collaborative filtering for robust conversational understanding," *arXiv preprint arXiv:2305.14449*, 2023.

[107] N. Aggio, "A graph-enhanced llm-based question answering system for the ai act," 2023.

[108] T. Bui, O. Tran, P. Nguyen, B. Ho, L. Nguyen, T. Bui, and T. Quan, "Cross-data knowledge graph construction for llm-enabled educational question-answering system: A case study at hcmut," in *Proceedings of the 1st ACM Workshop on AI-Powered Q&A Systems for Multimedia*, pp. 36–43, 2024.