REGRAPH: ITERATIVE GRAPH READING FOR OPEN-ENDED REASONING WITH LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Open-ended graph reasoning should allow Large Language Models (LLMs) to decide what to read from a graph as their reasoning unfolds. Yet existing approaches typically expose graph information either as retrieved text or as a fixed representation formed before reasoning begins, limiting how graph evidence can adapt to the evolving language-model state. We introduce REGRAPH (**RE**ad **GRAPH**), an iterative graph-reading framework that interleaves graph access with intermediate LLM computation. REGRAPH maintains a persistent graph memory and repeatedly queries it with graph-query states refined by preceding Transformer layers, enabling each read to depend on both the instruction and evidence acquired so far. A query-conditioned topology diffusion mechanism further couples semantic relevance with graph structure, while keeping the graph outside the LLM context and the pretrained LLM frozen. Experiments on six benchmarks, including two introduced in this work, demonstrate the effectiveness of REGRAPH across open-ended graph reasoning settings. The source code is available here.

1 INTRODUCTION

Large language models (LLMs) are broadening graph analysis beyond predefined prediction tasks toward open-ended graph reasoning (Zhang et al., 2024; He et al., 2024b; Kong et al., 2025), where graph-conditioned behaviors can be specified through natural-language instructions and expressed as flexible outputs, including answers, explanations, paths, and labels. Realizing this flexibility requires the LLM to access the graph evidence relevant to each question while preserving its structural relationships. This raises a fundamental question: *how should graph information be exposed to an LLM for flexible yet structurally grounded reasoning?*

Existing graph–LLM interfaces make graph information available in three broad ways. *Graph-as-Text* methods (Wang et al., 2023; Fatemi et al., 2024) serialize graphs into textual sequences. They preserve the generative flexibility of LLMs, but scale poorly and entangle graph understanding with arbitrary serialization choices. *Graph representation alignment* methods (Zhang et al., 2024; Chen et al., 2024b) encode the graph and map the resulting representation into the LLM token space. This avoids lengthy graph strings, yet a fixed graph token must compress many potentially relevant structural facts before the question-specific reasoning trajectory is known. *Graph retrieval-augmented generation* (GraphRAG) (Zhu et al., 2026; Dong et al., 2026) retrieves query-relevant graph evidence to support generation, improving grounding and scalability. However, open-ended reasoning is not necessarily reducible to a retrieval problem. Questions may express abstract reasoning goals that do not readily translate into explicit retrieval queries, requiring the model to develop a task-specific understanding of the graph to determine what information matters and how it should be interpreted. Xiang et al. (2026) show that GraphRAG is most effective when reasoning requires connected evidence, while indiscriminate expansion adds redundancy and cost. More related work is provided in Appendix A.

These challenges motivate a closer coupling between data access and the LLM's evolving understanding of the task. For an open-ended question, deciding what to read from the graph is itself part of reasoning, since the relevance of graph information may become apparent only as the model interprets previously acquired evidence. Each task-relevant read can refine this understanding and reveal further information needs. We therefore seek an interface in which LLM reasoning and graph reading refine one another, with intermediate LLM states guiding evidence selection and the acquired evidence refining those states to guide subsequent reads. This motivates a different abstraction:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

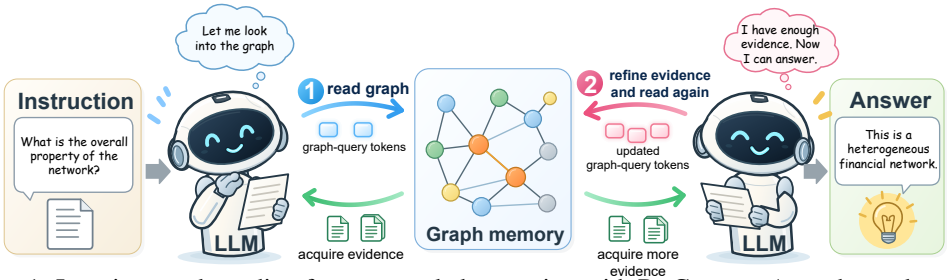

Figure 1: Iterative graph reading for open-ended reasoning with REGRAPH. At each graph-reading round, the task instruction and current graph-query tokens guide the LLM agent to read task-relevant evidence from the persistent graph memory (blue arrow). The acquired evidence is then encoded back into the graph-query tokens, which represent the agent's current knowledge about the task. In the next graph-reading round, the updated graph-query tokens are used to retrieve further evidence from the graph memory (red arrow), progressively refining the model's understanding of the task. Repeating this process allows REGRAPH to gather the evidence needed to generate the final answer.

Read the graph, reason in language space, form a new graph query, and read again.

We propose REGRAPH, which realizes this abstraction by interleaving graph reading with intermediate LLM computation. REGRAPH maintains a persistent graph memory accessed through graph-query tokens. After the initial Transformer layers encode the instruction, graph-query tokens issue the first graph read and absorb the returned evidence into their hidden states. Subsequent LLM layers reason over the evidence acquired in the preceding graph-reading round and refine the graph-query tokens into new task-conditioned states, which in turn guide the next graph-reading round. This creates a reciprocal interaction in which graph evidence shapes subsequent language computation, while the evolving language states determine what to read from the graph next. Fig. 1 illustrates this process with two successive reads as an example. Through this interaction, REGRAPH turns the graph from a static source of input information into an adaptive source of evidence that evolves with reasoning.

To make each graph read both query-aware and structurally grounded, we introduce a *Topology-Diffused Graph Reader*. Given the current graph-query tokens, the reader first identifies semantically relevant nodes and then propagates their relevance over the graph topology. A query-conditioned mixture over diffusion depths allows different graph-query tokens to gather evidence at the structural scale required by their current reasoning state, after which the retrieved evidence is written back into the tokens for subsequent LLM computation. REGRAPH as a graph interface is trained end-to-end from answer supervision while the pretrained LLM remains frozen for efficient training. We evaluate REGRAPH on six benchmarks spanning node-level inference, graph-level question answering, and structural localization, including two benchmarks introduced in this work. Extensive experiments demonstrate consistent gains over representative graph–LLM baselines.

2 PROBLEM FORMULATION

Let $G = (V, E, X)$ be an attributed graph with $N_V = |V|$ nodes, edge set $E \subseteq V \times V$, and node attributes $X = \{x_v \mid v \in V\}$. Attributes may be textual, numerical, categorical, visual, or multimodal. An attribute encoder f_{attr} maps each x_v to a vector representation. Given a natural-language instruction tokenized as $q = (q_1, \ldots, q_{N_q})$, the objective is to generate an open-ended response $Y = (y_1, \ldots, y_M)$ autoregressively (Brown et al., 2020):

$$p_\Theta(Y \mid q, G) = \prod_{s=1}^{M} p_\Theta(y_s \mid y_{<s}, q, G), \tag{1}$$

where $\Theta = (\theta_{\text{attr}}, \theta_{\text{LLM}}, \Psi)$ denotes the complete parameterization of the graph-conditioned generator: θ_{attr} and θ_{LLM} are the parameters of the attribute encoder and the pretrained LLM, respectively, while Ψ collects the REGRAPH-specific parameters. In our setting, θ_{attr} and θ_{LLM} are frozen and only Ψ is optimized, following the broad frozen backbone paradigm in which trainable continuous interfaces condition a fixed LLM (Tsimpoukelli et al., 2021; Alayrac et al., 2022; Li et al., 2023).

We use no task-specific prediction head. Instead, every task is expressed through the shared language-model vocabulary, so the response may be a class name, one or more entities, a path, an explanation,

or free-form natural language (Raffel et al., 2020). We use *open-ended* to describe this generative interface, and an individual benchmark may still have a finite answer vocabulary. A supervised example therefore takes the general form (G, q, Y).

Graph Memory. Each node is initialized as $c_v = f_{\text{attr}}(x_v) \in \mathbb{R}^{d_0}$. A graph encoder ENC_ϕ produces a persistent node memory:

$$H^{\text{base}} = \text{ENC}_\phi(G, C) = [h_1^{\text{base}}; \ldots; h_{N_V}^{\text{base}}] \in \mathbb{R}^{N_V \times d_g}, \tag{2}$$

where $C = [c_1; \ldots; c_{N_V}] \in \mathbb{R}^{N_V \times d_0}$. Unlike a pooled graph token, H^{base} preserves one addressable representation per node. Some questions explicitly identify nodes and assign them functional roles, as in "Find a path from node 7 to node 19." When such references can be resolved unambiguously, we define $r_v(q) \in \{\text{NONE}, \text{MENTIONED}, \text{SOURCE}, \text{TARGET}\}$, and construct the instruction-conditioned node representation $h_v(q) = h_v^{\text{base}} + e_{\text{role}}(r_v(q))$. The resulting memory is $H(q) = [h_1(q); \ldots; h_{N_V}(q)]$. For anchor-free questions, $r_v(q) = \text{NONE}$ for $v \in V$, and the reader must discover relevant nodes or regions from the instruction-conditioned query state and graph topology rather than from a predefined anchor.

Topology operator. Let A be the adjacency matrix. We add self-loops, $\widetilde{A} = A + I$, and define the row-normalized transition matrix $P = D^{-1}\widetilde{A}$, where $D_{vv} = \sum_u \widetilde{A}_{vu}$. For directed graphs, P follows the directed edges. If the benchmark protocol requires bidirectional propagation, reverse edges with distinct relation types are inserted before computing both H and P.

3 REGRAPH

Instead of compressing H once before reasoning (Tang et al., 2024; Perozzi et al., 2024; He et al., 2024b; Ning et al., 2025; Zhang et al., 2024), REGRAPH maintains graph-query tokens whose information needs evolve inside the LLM. Each round reads query-specific graph evidence, writes it into these tokens, and lets subsequent LLM layers transform them before the next read.

3.1 INTERLEAVING GRAPH ACCESS WITH LLM COMPUTATION

Let the pretrained LLM have hidden width d, vocabulary $\mathcal{V}_{\text{LM}}$, and input embedding lookup $e = e_{\theta_{\text{LLM}}} : \mathcal{V}_{\text{LM}} \to \mathbb{R}^d$, thus $e(w)$ is the LLM embedding of token w. Positional encoding and causal masking are retained but omitted from notation. We can partition the Transformer layers into $T + 1$ consecutive groups $F_0, F_1, \ldots, F_T$ and create T graph reading locations. Then we introduce N_B learnable *graph-query tokens* $B_{\text{base}} = [b_1; \ldots; b_{N_B}] \in \mathbb{R}^{N_B \times d}$ and place them after the instruction but before an answer-start token y_0. The prefill sequence is:

$$Z_{\text{in}} = [e(q_1); \ldots; e(q_{N_q}); B_{\text{base}}; e(y_0)] \in \mathbb{R}^{(N_q + N_B + 1) \times d}. \tag{3}$$

This ordering is deliberate. Under the ordinary causal mask, graph-query tokens can attend to the complete instruction and preceding graph-query tokens, while all answer tokens can later attend to graph-query tokens.

The first Transformer-layer group maps the input sequence to $Z_{\text{pre}}^0 = F_0(Z_{\text{in}})$. Let $\mathcal{I}_B$ denote the ordered set of sequence positions occupied by the graph-query tokens. For each graph reading round $t \in \{0, \ldots, T-1\}$, their current hidden states are extracted as

$$B_{\text{pre}}^t = Z_{\text{pre}}^t[\mathcal{I}_B] \in \mathbb{R}^{N_B \times d}, \tag{4}$$

where $Z[\mathcal{I}_B]$ denotes the selected rows of Z indexed by $\mathcal{I}_B$. Thus, B_{pre}^t contains the graph-query token representations produced by the Transformer-layer group immediately preceding round t and serves as the language-side query state before the graph is accessed at that round. The complete interleaved update at graph-reading round t is:

$$R^t = \texttt{Read}(B_{\text{pre}}^t, H, P), \tag{5}$$

$$B_{\text{post}}^t = \texttt{Fuse}(B_{\text{pre}}^t, R^t), \tag{6}$$

$$Z_{\text{post}}^t = \texttt{Replace}(Z_{\text{pre}}^t, \mathcal{I}_B, B_{\text{post}}^t), \tag{7}$$

$$Z_{\text{pre}}^{t+1} = F_{t+1}(Z_{\text{post}}^t). \tag{8}$$

Here we explain Eq. (5)-Eq. (8) step by step. The initial graph query tokens $B^0_{\text{pre}} = Z^0_{\text{pre}}[\mathcal{I}_B] = F_0(Z_{\text{in}})[\mathcal{I}_B]$ are conditioned only on the instruction, without receiving any information from the graph. More generally, B^t_{pre} represents the model's current estimate of what information should be read from the graph at the graph reading round t. For $t > 0$, it additionally reflects the graph evidence acquired in earlier rounds and subsequently transformed by the intervening Transformer-layer groups. At round t, `Read` operator in Eq. (5) uses B^t_{pre} as a set of instruction-conditioned query tokens over the persistent node memory H, while using the transition matrix P to account for topology. It returns $R^t \in \mathbb{R}^{N_B \times d}$, whose i-th row R^t_i contains the graph evidence retrieved for the i-th graph-query token under its current reasoning state. Intuitively, the operation uses the current reasoning knowledge encoded in B^t_{pre} to query the graph for relevant information and return the corresponding evidence R^t by retrieving that information from the graph encoded by H while respecting the topology P.

Then the `Fuse` operator defined in Eq. (6) encodes the returned evidence R^t into the current query state B^t_{pre} to form B^t_{post}, which is conditioned on the instruction and on all graph evidence acquired through round t.

Next, `Replace` operator in Eq. (7) writes B^t_{post} back to the graph-query token positions, yielding $Z^t_{\text{post}}[\mathcal{I}_B] = B^t_{\text{post}}$, while leaving all other token positions in Z^t_{post} unchanged. The modified sequence is then processed by the next Transformer-layer group F_{t+1}, producing Z^{t+1}_{pre} as Eq. (8). If another graph reading round remains, its query state is extracted as $B^{t+1}_{\text{pre}} = Z^{t+1}_{\text{pre}}[\mathcal{I}_B]$. Importantly, B^{t+1}_{pre} is not merely a repeated query over the same graph memory. It is a refined information need formed after the LLM has processed the evidence returned at round t. Accordingly, successive graph reads are conditioned on progressively evolving language-side reasoning states.

The overall computation therefore alternates between graph access and LLM computation:

$$F_0 \to \underbrace{[\text{Read} \to \text{Fuse} \to \text{Replace}]}_{\Gamma_0} \to F_1 \to \cdots \to \underbrace{[\text{Read} \to \text{Fuse} \to \text{Replace}]}_{\Gamma_{T-1}} \to F_T. \tag{9}$$

For rounds $t < T - 1$, the output of F_{t+1} supplies the query state for the next graph read. After the final round, F_T produces the fully graph-conditioned hidden sequence used for answer generation. In this way, REGRAPH alternates between graph access and LLM computation: the current graph-query states retrieve relevant evidence from the graph, the subsequent Transformer layers reason over the injected evidence, and the resulting hidden states refine what information should be queried from the graph next. Repeating this process allows graph access to adapt progressively to the evolving reasoning state of the LLM.

We next instantiate two operations. `Read`$(\cdot)$ retrieves graph evidence that is relevant to the evolving states of graph-query tokens while accounting for topology, and `Fuse`$(\cdot)$ adaptively injects the retrieved evidence back into the graph-query tokens.

3.2 TOPOLOGY-DIFFUSED GRAPH READER

The reader `Read`$(\cdot)$ must make two decisions: *where* in the graph should each graph-query token begin reading, and *how far* should relevance propagate? Given the current states of graph-query tokens B^t_{pre}, graph memory H and topology P as the input (Eq. (5)), we model these two decisions separately while learning them jointly. Intuitively, each graph-query token uses its current LLM state to identify relevant seed nodes, then follows the graph topology to gather evidence from their multi-hop neighborhoods. The retrieved evidence is written back into the graph-query token, turning it into a graph-informed state for subsequent LLM computation. Since different graph-query tokens may require different structural scopes that evolve across reasoning rounds, REGRAPH learns an adaptive mixture over propagation depths instead of using a fixed neighborhood size.

We first identify where each graph-query token should read. Let d_r denote the hidden dimension of the graph reader. We project the current graph-query states into the reader query space:

$$U^t = \text{LN}\left(B^t_{\text{pre}}\right) W_Q \in \mathbb{R}^{N_B \times d_r}, \tag{10}$$

and project the persistent node memory into key and value spaces,

$$K_H = \text{LN}(H) W_K \in \mathbb{R}^{N_V \times d_r}, \quad V_H = \text{LN}(H) W_V \in \mathbb{R}^{N_V \times d_r}, \tag{11}$$

where $W_Q \in \mathbb{R}^{d\times d_r}$ and $W_K, W_V \in \mathbb{R}^{d_g\times d_r}$ are weight matrices. $\mathrm{LN}(\cdot)$ denotes layer normalization (Ba et al., 2016). Since H is persistent across graph-reading rounds, K_H and V_H can be computed once and reused, whereas U^t changes with the evolving graph-query token states. For the i-th graph-query token and node v, their semantic compatibility is $\frac{\langle U_i^t, K_{H,v}\rangle}{\sqrt{d_r}}$. Normalizing over graph nodes gives the initial graph reading distribution $S^{t,(0)} = \mathrm{softmax}_V\left(\frac{U^t K_H^\top}{\sqrt{d_r}}\right) \in \mathbb{R}^{N_B\times N_V}$. The quantity $S_{iv}^{t,(0)}$ measures how strongly the i-th graph-query token, under its current reasoning state at round t, regards node v as a useful semantic starting point. Importantly, this distribution is query-conditioned, because different graph-query tokens may select different regions, and the same token may shift its attention across rounds as the LLM refines what information it needs.

Semantic similarity alone, however, does not determine which surrounding graph context should be read. We therefore propagate the query-conditioned seed distribution through P as $S^{t,(k)} = S^{t,(k-1)}P = S^{t,(0)}P^k$ for $k = 1, \ldots, K$, where K is the maximum topology-diffusion depth and $P^0 = I$. For the i-th graph-query token, $S_{i,:}^{t,(k)}$ describes the node relevance obtained by propagating its initial semantic seed distribution through k graph transitions. Thus, $k = 0$ preserves the semantic seeds themselves, while larger k progressively incorporates structurally related nodes. Moreover, different graph-query tokens $\{B_{\mathrm{pre},i}^t\}_{i=1}^{N_B}$ within the same round t can require different structural scales, so each token predicts a hop mixture:

$$\alpha^t = \mathrm{softmax}_{\mathrm{hop}}\big(\mathrm{LN}(B_{\mathrm{pre}}^t)W_\alpha + b_\alpha\big) \in \mathbb{R}^{N_B\times(K+1)}, \tag{12}$$

where $W_\alpha \in \mathbb{R}^{d\times(K+1)}$ and $b_\alpha \in \mathbb{R}^{K+1}$, with $\sum_{k=0}^K \alpha_{ik}^t = 1$. α_{ik}^t specifies how much the i-th graph-query token relies on evidence propagated through k graph transitions at round t. Since α^t is predicted from B_{pre}^t, the preferred reading radius can change as language-side reasoning evolves. We combine the different structural scales to obtain the final topology-aware reading distribution $\widetilde{S}_{i,:}^t = \sum_{k=0}^K \alpha_{ik}^t S_{i,:}^{t,(k)}$. Equivalently, in matrix form,

$$\widetilde{S}^t = \sum_{k=0}^K \mathrm{Diag}(\alpha_{:,k}^t)S^{t,(k)} = \sum_{k=0}^K \mathrm{Diag}(\alpha_{:,k}^t)S^{t,(0)}P^k. \tag{13}$$

Thus, $S^{t,(0)}$ determines *where* to begin reading, while α^t determines *how far* relevance should propagate. This yields a query-conditioned polynomial graph filter over semantic relevance. Finally, node values are aggregated and projected back to the LLM hidden space:

$$R^t = \widetilde{S}^t V_H W_O \in \mathbb{R}^{N_B\times d}, \tag{14}$$

where $W_O \in \mathbb{R}^{d_r\times d}$. R_i^t is not a global graph summary, but the evidence requested by the i-th graph-query token at round t. In the multi-head implementation, $S^{t,(k)}$ and Eq. (13) are computed separately for each head. The resulting evidence vectors are concatenated and passed through a shared output projection (Vaswani et al., 2017).

3.3 Gated Residual Evidence Fusion

The reader determines what evidence in the graph is relevant to the task, whereas the fusion module $\texttt{Fuse}(\cdot)$ in Eq. (6) determines how strongly the retrieved evidence R^t updates the current graph-query states B_{pre}^t to produce B_{post}^t. We therefore instantiate $\texttt{Fuse}$ as a gated residual update that preserves the current language-side state while controlling the contribution of the retrieved evidence according to its relevance to the current reasoning state (Alayrac et al., 2022). Since B_{pre}^t and R^t originate from different encoders, we normalize them separately by $\widehat{B}^t = \mathrm{LN}(B_{\mathrm{pre}}^t)$ and $\widehat{R}^t = \mathrm{LN}(R^t)$. A scalar evidence-acceptance gate is predicted for each graph-query token as:

$$g^t = \sigma\Big([\widehat{B}^t \| \widehat{R}^t]w_g + b_g\Big) \in (0,1)^{N_B}, \tag{15}$$

where $\|$ denotes concatenation along the hidden dimension, $w_g \in \mathbb{R}^{2d}$, $b_g \in \mathbb{R}$, and σ is the logistic sigmoid. The i-th gate g_i^t measures how strongly the evidence retrieved for the i-th graph-query token influences its current state. The graph evidence is then injected through a gated residual connection:

$$B_{\mathrm{post}}^t = B_{\mathrm{pre}}^t + g^t \odot R^t, \tag{16}$$

where $\odot$ denotes row-wise scalar multiplication. When $g_i^t \approx 0$, the graph-query token largely retains its current state, and when $g_i^t \approx 1$, the retrieved graph evidence is fully incorporated. Since the gate is recomputed at every reading round from both the current query state and the retrieved evidence, REGRAPH can dynamically adjust its reliance on the graph as reasoning progresses.

3.4 ANSWER GENERATION

Given a decoder-only LLM containing L Transformer layers, we partition it into $T+1$ non-overlapping consecutive groups, denoted by $F_0, \ldots, F_T$. With these $T+1$ LLM blocks, there are naturally T graph-reading rounds, inserted between adjacent LLM layer groups as Eq. (9). We use $\Gamma_t(Z_{\text{pre}}^t, H, P) = \texttt{Replace}(Z_{\text{pre}}^t, \mathcal{I}_B, \texttt{Fuse}(B_{\text{pre}}^t, \texttt{Read}(B_{\text{pre}}^t, H, P)))$ to represent a complete Read-Fuse-Replace operation for graph reading, which produces Z_{post}^t for the next Transformer-layer group F_{t+1}. Conceptually, when generating the s-th answer token, the model conditions on the previously generated prefix $y_{<s} = (y_1, \ldots, y_{s-1})$. The corresponding input sequence is $Z_{\text{in}}^{(s)} = \left[e(q_1); \ldots; e\left(q_{N_q}\right); B_{\text{base}}; e(y_0); e(y_1); \ldots; e\left(y_{s-1}\right)\right]$. Applying the complete interleaved computation gives:

$$Z_{\text{out}}^{(s)} = F_T \circ \Gamma_{T-1} \circ F_{T-1} \circ \cdots \circ \Gamma_0 \circ F_0 \left(Z_{\text{in}}^{(s)}\right), \tag{17}$$

where $\circ$ denotes function composition along the sequence of interleaved LLM layer groups and graph-reading operations. Let $j_s = N_q + N_B + s$ denote the final position of $Z_{\text{in}}^{(s)}$ under one-based indexing. In particular, $j_1 = N_q + N_B + 1$ corresponds to the answer-boundary token y_0. The final hidden state at position j_s is passed through the pretrained LLM's frozen output head to obtain the next-token distribution:

$$p_\Theta(y_s \mid y_{<s}, q, G) = \left[\text{softmax}\left(W_{\text{vocab}}\ \text{LN}_{\text{final}}\left(Z_{\text{out}}^{(s)}[j_s]\right) + b_{\text{vocab}}\right)\right]_{y_s}, \tag{18}$$

where $[\cdot]_{y_s}$ denotes the probability assigned to token y_s. The predicted token is appended to the response prefix, and the process continues autoregressively until $\langle\text{EOS}\rangle$ is generated or the maximum generation length is reached.

Efficient inference with KV caching. Since the graph-query tokens precede all answer tokens, the causal mask prevents their hidden states from depending on generated outputs. Thus, for every graph reading round t, we have $B_{\text{pre}}^{t,(s)} = B_{\text{pre}}^{t,(1)}$ for $s \geq 1$ and the Read-Fuse-Replace operations $[\Gamma]_0^{T-1}$ are executed only once during prefill over $[q; B_{\text{base}}; y_0]$. The resulting graph-conditioned keys and values are stored in the standard KV cache (Pope et al., 2023; Kwon et al., 2023). Subsequent answer tokens are then generated by ordinary incremental decoding, attending to the cached graph-query representations without recomputing any Γ_t. Hence, the graph-reading cost is independent of the generated answer length.

Training. For a graph G, let $\mathcal{D}_G = \{(q^{(n)}, Y^{(n)})\}_{n=1}^{N_G}$ denote its associated training set, consisting of N_G instruction–answer pairs. During teacher-forced training, we optimize the answer-only autoregressive negative log-likelihood:

$$\mathcal{L}_{\text{gen}} = -\sum_{n=1}^{N_G} \sum_{s=1}^{M_n+1} \log p_\Theta\left(y_s^{(n)} \mid y_{<s}^{(n)}, q^{(n)}, G\right), \tag{19}$$

where $y_{M_n+1}^{(n)} = \langle\text{EOS}\rangle$ and M_n is the number of tokens in the gold response corresponding to instruction $q^{(n)}$. The loss is computed only over the gold response tokens and the terminal $\langle\text{EOS}\rangle$, and positions corresponding to the instruction, graph-query tokens, and answer boundary are masked. The primary setting freezes the attribute encoder θ_{attr} and pretrained LLM θ_{LLM}, and optimizes only the REGRAPH-specific parameters Ψ, including the graph encoder, graph-query tokens, Topology-Diffused Graph Reader, and Gated Residual Evidence Fusion module.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTING

We evaluate REGRAPH on six text-attributed graph benchmarks that collectively cover complementary forms of graph-conditioned language reasoning. All tasks are formulated as (G, q, Y) and solved by

Table 1: Results (%) on node-centric benchmarks.

Method	ogbn-arxiv			Cora	NeighborhoodQA	
	Top-1	Top-3	Top-5	Accuracy	Set-F1	Exact
BERT	1.63	7.63	28.00	23.56	32.18	3.72
RoBERTa	3.55	11.98	22.93	27.91	35.64	4.81
BERT (fine-tuned)	14.53	29.60	38.30	83.12	66.84	19.26
RoBERTa (fine-tuned)	6.95	16.53	23.75	84.37	68.73	22.14
LLM + s_v	14.07	26.98	42.46	70.11	64.21	16.83
LLM + $s_v + s_{N(v)}$	17.90	28.43	37.99	76.84	76.92	34.76
GraphTranslator	28.48	37.62	39.87	62.65	74.83	31.28
GPT-3.5-Turbo	69.50	89.37	92.42	71.75	78.36	37.05
LLaGA-ND-7B	71.98	88.68	93.87	88.86	80.87	40.96
LLaGA-HO-7B	71.66	90.12	93.31	**89.22**	81.62	42.18
REGRAPH	**72.28**	**93.23**	**96.90**	88.99	**83.83**	**44.93**

Table 2: Results (%) on graph-level QA and structural localization.

Method	Expla Graphs	Scene Graphs	Structural Anomaly
Zero-shot	56.50	39.74	41.85
Zero-CoT	57.04	52.60	49.30
CoT-BAG	57.94	56.80	57.65
LoRA	85.38	78.62	86.75
KAPING	62.27	43.75	-
Prompt tuning	57.63	63.41	63.40
GraphToken	85.08	49.03	96.35
G-Retriever	85.16	81.31	97.05
G-Retriever (w/ LoRA)	87.05	**86.83**	98.42
GRAG	88.94	68.37	97.18
REGRAPH	**90.43**	53.24	**99.15**

autoregressive generation. Among the four established datasets, **ogbn-arxiv** (Hu et al., 2020) follows GraphTranslator (Zhang et al., 2024) protocol; **Cora** (Sen et al., 2008) follows the LLaGA (Chen et al., 2024b) single-focus setting; **ExplaGraphs** (Saha et al., 2021) and **SceneGraphs** derived from GQA (Hudson & Manning, 2019) use the GraphQA formulation of G-Retriever (He et al., 2024b). Together, they cover node-topic generation, neighborhood-conditioned inference, relational-polarity reasoning, and grounded compositional question answering.

We additionally propose **NeighborhoodQA** and **StructuralAnomaly**. NeighborhoodQA is derived from ogbn-arxiv and requires generating the complete set of distinct subject areas represented among a paper's cited neighbors, directly testing multi-node aggregation and deduplication. StructuralAnomaly uses planted-partition graphs (Holland et al., 1983). Without a named anchor, the model must locate the uniquely dense community from topology and identify its semantic theme from node text, while a matched density-free control is unanswerable by construction. These benchmarks target two capabilities largely absent from the established suite: exact multi-node synthesis and anchor-free structure–semantics binding. More details of the benchmarks are provided in Appendix B. Note that graph information is injected into the LLM through graph-query tokens at intermediate Transformer layers, while the graph itself is never serialized into the input context. Further details on the comparison protocol and implementation are provided in Appendix C.

4.2 RESULTS

We group the six benchmarks by prediction interface. Table 1 reports node-centric tasks and comparisons with text-based and graph–language models. Table 2 reports graph-level QA and structural localization, with comparisons against graph-conditioned generative methods. The details of the baselines are provided in Appendix D. REGRAPH supports both task groups within a single architecture: role markers identify a focal node when one is specified, while the same query-conditioned reader operates without an anchor for graph-level reasoning, with one autoregressive output interface throughout. More model analysis and ablation studies are provided in Appendix E.

As shown in Tables 1 and 2, REGRAPH performs strongly across both node-centric and graph-level open-ended reasoning tasks. The results suggest that graph information is more effectively exploited through adaptive graph reading across LLM layers, as in REGRAPH, than by compressing it into a single static representation (e.g., GraphTranslator and LLaGA) or serialized textual context (e.g., KAPING and CoT-BAG). For the two datasets proposed in this work, the strong results on NeighborhoodQA indicate that iterative reading can aggregate and reconcile information distributed across multiple neighboring nodes, while the near-saturated performance on StructuralAnomaly provides particularly clear evidence that REGRAPH can identify structurally salient regions even when the query provides no explicit anchor node. These results show that topology should not merely be encoded once and aligned to the LLM, but should remain directly accessible throughout reasoning so that the model can dynamically determine *where* and *at what structural scale* to read. The relatively weaker result on SceneGraphs also exposes a complementary limitation: highly fine-grained object–attribute and relation bindings remain difficult to preserve through a compact vector interface, indicating an important direction for further improving graph–language interaction. Table 3 illustrates REGRAPH's graph-conditioned responses to free-form SceneGraphs questions, with additional examples and discussion in Appendix F.

Table 3: Open-ended questions on SceneGraphs test scenes, answered by REGRAPH and by the same frozen LLM without access to the graph. Object and attribute mentions are marked green when they match the scene graph and red when they are absent from it or contradict it. Answers are verbatim, and long baseline answers are truncated at [. . .].

	Scene description	**Place and reason**	**Objects and colors**	**Main subject**
SCENE GRAPH encoder input	giraffe (playing) · neck (long) · grass (green) · trees · sky (gray) · . . .	bus (red) · bus (double decker) · building (brick) · tree (bare) · sidewalk · . . .	towel (white) · sink (white) · sink (porcelain) · floor (black) · mirror · . . .	train · man · driver · platform · sidewalk · . . .
QUESTION	*Describe this scene in one or two sentences.*	*Where was this photo most likely taken, and what makes you think so?*	*What objects stand out in this scene, and what colors are they?*	*What is the main subject of this picture, and what is it doing?*
LLM W/O GRAPH	I'm happy to help, but I don't see a scene described. [. . .]	I can provide some clues that might help you narrow down the possibilities. The photo appears to be a scenic landscape with a large body of water in the foreground, and there are trees and hills in the background. [. . .]	The objects that stand out in this scene are the red and blue objects. The red object is a fire truck, and the blue object is a police car. [. . .]	The main subject of this picture is a tree, and it is standing upright. [. . .]
REGRAPH (OURS)	The scene depicts a serene and peaceful atmosphere, with a tall giraffe standing in a lush green grassy area, surrounded by tall trees and a bright blue sky. The giraffe appears to be gazing calmly into the distance, its long neck stretched out in a relaxed pose.	this photo was most likely taken at "downtown" and what makes me think so is that the bus is parked in front of a large building.	The objects that stand out in this scene are the towel and the sink. The towel is white and the sink is black.	main subject is train, and it is standing

Table 4: Graph dependence of REGRAPH. PairAcc requires correct predictions on both the original and perturbed graphs. For NeighborhoodQA, complete correctness is measured by exact-set match.

Benchmark	**Original Acc.** ↑	**Answer-changing PairAcc** ↑	**Answer-preserving PairAcc** ↑
SceneGraphs	53.24	44.18	50.37
NeighborhoodQA	44.93	36.72	42.11
StructuralAnomaly	99.15	96.84	98.52

Table 5: Adaptive-reading comparison. Single-read injects evidence once, and static replay supplies the first evidence tensor at all three reading boundaries.

	SceneGraphs	**NeighborhoodQA**	
Schedule	Acc. ↑	Set-F1 ↑	Exact ↑
Single-read	47.73	80.25	39.19
Static replay	51.06	81.71	40.86
REGRAPH	53.24	83.83	44.93

4.3 GRAPH DEPENDENCE

To investigate whether REGRAPH's predictions are genuinely grounded in the graph information incorporated across intermediate Transformer layer groups, we compare each original graph with two perturbations. **Answer-changing perturbations** modify graph evidence that determines the correct answer. **Answer-preserving perturbations** alter graph content that is irrelevant to the question. A graph-grounded model should change its answer appropriately under answer-changing perturbations and remain correct under answer-preserving perturbations. We apply this protocol to SceneGraphs, NeighborhoodQA, and StructuralAnomaly, covering relational question answering, multi-node synthesis, and anchor-free localization. We recompute graph representations and perform a fresh prefill for each input. Table 4 presents the comparison using paired accuracy, which requires correct answers on both the original and perturbed graphs. The results show that REGRAPH exhibits consistent selective dependence on graph evidence across all three task types. When answer-determining evidence is changed, the model correctly follows the revised graph on most originally correct instances, while performance remains substantially closer to the original level when only irrelevant graph content is

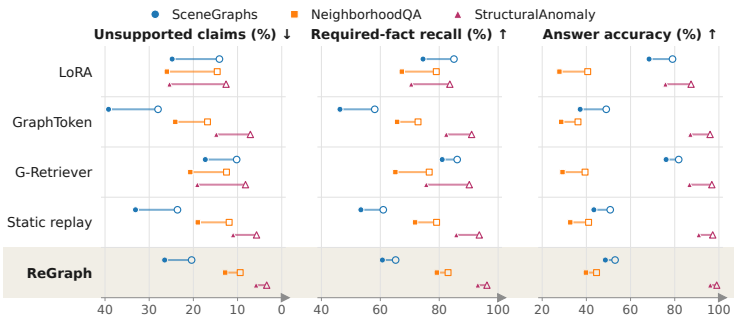

Figure 3: Faithfulness under plausible graph distractors. Hollow markers denote clean graphs, while filled markers denote graphs augmented with 50% distractor nodes. Connecting lines indicate the performance change after adding distractors. Moving right indicates better performance.

perturbed. They indicate that REGRAPH's predictions are conditioned on question-relevant graph evidence. Detailed perturbation construction and metric definitions are provided in Appendix E.1.

4.4 ADAPTIVE READING

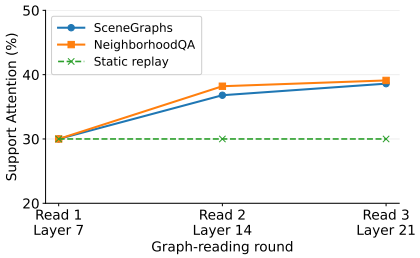

Figure 2: Evolution of graph-reading focus across intermediate LLM layers.

REGRAPH creates a feedback loop before generating answers: read the graph, reason over the returned evidence, and then decide what to read next. We evaluate whether this feedback is useful by comparing REGRAPH with Single-read, which accesses the graph only once, and Static replay, which repeatedly provides the evidence retrieved by the first read without allowing the requested graph information to change. Table 5 shows that repeatedly exposing the LLM to the same graph evidence is insufficient. Static replay improves over Single-read, and REGRAPH yields further gains by allowing later graph queries to depend on previously acquired evidence. This suggests that the benefit of iterative graph reading comes not only from accessing the graph multiple times, but from using previous evidence to refine what the model looks for next. Fig. 2 further shows that the attention (definition in Appendix E.2) assigned to question-relevant supporting nodes increases after the first read and remains higher in later rounds, while static replay necessarily preserves the initial reading focus. It indicates that, after incorporating earlier graph evidence and reasoning through subsequent LLM layers, REGRAPH forms increasingly targeted requests for the graph.

4.5 FAITHFULNESS

We evaluate unsupported-claim rate, required-fact recall, and answer accuracy on clean graphs and graphs augmented with 50% distractor nodes. As shown in Fig. 3, REGRAPH is comparatively robust across these measures. On NeighborhoodQA and StructuralAnomaly, it achieves the best faithfulness and answer quality, with only a small degradation after adding 50% distractor nodes. The static-replay variant degrades more, indicating that adaptive re-reading helps filter irrelevant evidence. On SceneGraphs, although G-Retriever attains higher absolute accuracy, REGRAPH remains comparatively stable under distraction.

5 CONCLUSION

We introduce REGRAPH, a graph–LLM interface that interleaves graph reading with intermediate LLM computation, allowing evidence acquisition to adapt as reasoning progresses. REGRAPH maintains a persistent graph memory and uses evolving graph-query tokens to repeatedly identify task-relevant evidence. This reciprocal interaction allows retrieved graph evidence to shape subsequent language computation, while the resulting hidden states refine later graph queries. A query-conditioned topology-diffused reader further combines semantic relevance with graph structure at each reading round, while the graph remains outside the textual input and the pretrained LLM remains frozen. Experiments on six benchmarks support the effectiveness of adaptive graph reading and the broader principle that graph access should evolve with the LLM's reasoning state.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pp. 78–106, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. Graphwiz: An instruction-following language model for graph computational problems. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 353–364, 2024a.

Runjin Chen, Tong Zhao, Ajay Kumar Jaiswal, Neil Shah, and Zhangyang Wang. LLaGA: Large language and graph assistant. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 7809–7823. PMLR, 21–27 Jul 2024b. URL https://proceedings.mlr.press/v235/chen24bh.html.

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Junnan Dong, Siyu An, Yifei Yu, QIANWEN ZHANG, Linhao Luo, Xiao Huang, Yunsheng Wu, Xing Sun, et al. Youtu-graphrag: Vertically unified agents for graph retrieval-augmented complex reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 143735–143755, 2026.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *International conference on learning representations*, volume 2024, pp. 43909–43934, 2024.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. Chatglm: A family of large language models from glm-130b to glm-4 all tools, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066*, 2023.

Xiaojun Guo, Ang Li, Yifei Wang, Stefanie Jegelka, and Yisen Wang. G1: Teaching LLMs to reason on graphs with reinforcement learning. In *1st ICML Workshop on Foundation Models for Structured Data*, 2025a. URL https://openreview.net/forum?id=nCcZgxQD2P.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tian Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation. In *EMNLP (Findings)*, pp. 10746–10761, 2025b.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569, 2024.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: LLM-to-LM interpreter for enhanced text-attributed graph representation learning. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=RXFVcynVe1.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37:132876–132907, 2024b.

Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. Grag: Graph retrieval-augmented generation. In *Findings of the association for computational linguistics: NAACL 2025*, pp. 4145–4157, 2025.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6693–6702. IEEE, 2019.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Lecheng Kong, Jiarui Feng, Hao Liu, Chengsong Huang, Jiaxin Huang, Yixin Chen, and Muhan Zhang. GOFA: A generative one-for-all model for joint graph language modeling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mIjblC9hfm.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pp. 19730–19742. PmLR, 2023.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Linhao Luo, Yuan-Fang Li, Reza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*, volume 2024, pp. 14400–14423, 2024.

Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Yuan-Fang Li, Chen Gong, and Shirui Pan. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Fr7kH2SFq7.

Xuying Ning, Dongqi Fu, Tianxin Wei, Wujiang Xu, and Jingrui He. Graph4MM: Weaving multimodal learning with structural information. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=FB2e8PV6qg.

Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. Let your graph do the talking: Encoding structured data for llms. *arXiv preprint arXiv:2402.05862*, 2024.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624, 2023.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Swarnadeep Saha, Prateek Yadav, Lisa Bauer, and Mohit Bansal. Explagraphs: An explanation graph generation task for structured commonsense reasoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7716–7740, 2021.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–106, 2008.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *International Conference on Learning Representations*, volume 2024, pp. 3868–3898, 2024.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao 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.

Jianheng Tang, Qifan Zhang, Yuhan Li, Nuo Chen, and Jia Li. Grapharena: Evaluating and exploring large language models on graph computation. In *International Conference on Learning Representations*, volume 2025, pp. 48118–48145, 2025.

Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36:30840–30861, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Zhishang Xiang, Chuanjie Wu, Qinggang Zhang, Shengyuan Chen, Zijin Hong, Xiao Huang, and Jinsong Su. When to use graphs in RAG: A comprehensive analysis for graph retrieval-augmented generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=i9q9xDMjG7.

Minji Yoon, Jing Yu Koh, Bryan Hooi, and Russ Salakhutdinov. Multimodal graph learning for generative tasks. In *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*, 2023. URL https://openreview.net/forum?id=YILik4gFBk.

Mengmei Zhang, Mingwei Sun, Peng Wang, Shen Fan, Yanhu Mo, Xiaoxiao Xu, Hong Liu, Cheng Yang, and Chuan Shi. Graphtranslator: Aligning graph model to large language model for open-ended tasks. In *Proceedings of the ACM Web Conference 2024*, pp. 1003–1014, 2024.

Zulun Zhu, Tiancheng Huang, Kai Wang, Junda Ye, Xinghe Chen, and Siqiang Luo. Graph-based approaches and functionalities in retrieval-augmented generation: A comprehensive survey. *ACM Computing Surveys*, 58(10):1–38, 2026.

A Related Work

We categorize LLM-based approaches by how graph information enters language-model computation: as text, retrieved evidence, or learned representations. These categories may overlap and are distinguished by their graph-access mechanisms.

A.1 Graph Reasoning through LLM Prompts and Post-training

Early studies investigate whether LLMs can solve graph problems directly from textual descriptions. NLGraph (Wang et al., 2023) and GPT4Graph (Guo et al., 2023) evaluate structural understanding beyond ordinary question answering, exposing difficulties in inferring graph properties from language inputs. Talk Like a Graph (Fatemi et al., 2024) further demonstrates that graph encoding and node naming materially affect performance, making the input representation an important part of the reasoning problem. Chain-of-thought prompting (Wei et al., 2022) provides a general mechanism for eliciting intermediate textual reasoning, while graph-specific post-training seeks to make such computation more reliable. GraphWiz (Chen et al., 2024a) combines instruction tuning on graph problems with preference optimization over reasoning responses, and G1 (Guo et al., 2025a) applies

reinforcement learning to graph reasoning. These approaches primarily improve how the LLM interprets and computes over the graph information supplied in its token sequence.

The resulting capabilities require evaluation beyond favorable prompts or small examples. GraphArena (Tang et al., 2025) examines graph computation across diverse tasks and difficulty levels, while long-context analysis shows that placing relevant evidence within the context window does not ensure its reliable use (Liu et al., 2024). REGRAPH investigates a complementary interface: the graph remains outside the textual prompt, and a frozen LLM requests evidence through graph-query tokens whose states evolve across intermediate layers. This avoids requiring the LLM to reconstruct the entire topology from a serialization, but does not by itself establish unrestricted algorithmic reasoning. Our neighborhood-visibility and dependency-depth protocols therefore distinguish access to useful graph information from the ability to compute arbitrary graph properties.

A.2 GRAPH-RETRIEVAL-AUGMENTED LLMS

Retrieval-augmented generation supplements an LLM's parametric knowledge with external evidence (Lewis et al., 2020). For knowledge-graph QA, RoG (Luo et al., 2024) generates relation-based plans and retrieves corresponding graph paths, whereas ToG (Sun et al., 2024) interleaves LLM reasoning with explicit graph exploration. GCR (Luo et al., 2025) uses a KG-Trie to constrain path generation to valid graph continuations. These methods ground language reasoning in identifiable entities, relations, and paths, with the graph serving as an external source of structured evidence.

Graph-based retrieval also extends beyond entity-centered path search. GraphRAG (Edge et al., 2024) uses community summaries for corpus-level, query-focused synthesis, HippoRAG (Gutiérrez et al., 2024) propagates relevance through a knowledge graph using personalized PageRank, and LightRAG (Guo et al., 2025b) combines low- and high-level retrieval over graph-indexed knowledge. Youtu-GraphRAG (Dong et al., 2026) coordinates schema-guided construction, hierarchical knowledge organization, and agentic query decomposition with iterative reflection. Accordingly, neither iterative access nor support for broad, anchor-free questions is unique to REGRAPH. For textual-graph QA, G-Retriever (He et al., 2024b) selects a subgraph through prize-collecting Steiner tree optimization and combines its textual description with a learned graph prompt. GRAG (Hu et al., 2025) similarly integrates complementary textual and graph representations, with soft pruning to suppress irrelevant subgraph content. The key distinction is *where adaptation occurs*. Existing GraphRAG methods retrieve graph evidence as external context, whereas REGRAPH repeatedly queries persistent graph memory from intermediate LLM states, coupling graph reading with depth-wise language reasoning without serializing graph content.

A.3 GRAPH–LLM ALIGNMENT

In multimodal language modeling, Frozen (Tsimpoukelli et al., 2021) conditions a frozen language model on learned visual prefixes, BLIP-2 (Li et al., 2023) uses a querying transformer to bridge visual representations and LLMs, and Flamingo (Alayrac et al., 2022) inserts gated cross-attention modules between frozen language-model layers. These works establish precedents for compact query-based interfaces and intermediate-layer evidence injection. In the graph domain, GraphTranslator (Zhang et al., 2024) uses generated graph–text alignment data to train a translator from node representations to soft prompt tokens. GraphGPT (Tang et al., 2024) aligns graph representations with LLMs through graph instruction tuning. LLaGA (Chen et al., 2024b) maps structure-aware neighborhood sequences into the LLM embedding space. Their graph interfaces prepare representations for the language model, rather than repeatedly querying node memory with the hidden states produced by its successive layer groups. MMGL (Yoon et al., 2023) extends language generation to graph-linked textual and visual context. Graph4MM (Ning et al., 2025) further introduces Hop-Diffused Attention and MM-QFormer to incorporate multi-hop structure into multimodal representations before supplying them to an LLM. This is particularly relevant to our reader: both exploit topology to organize attention, but they apply it at different stages and to different quantities. Graph4MM performs structure-guided fusion of modality-specific node representations. In REGRAPH, the Topology-Diffused Graph Reader propagates query-to-node relevance through P and predicts the hop mixture from the current graph-query state B^t_{pre}. The stored memory H remains fixed between reads.

B BENCHMARK DETAILS

This section provides additional details for the benchmarks introduced in Section 4.1. Since the central claim of REGRAPH is that the graph is never serialized into the context, for all benchmarks, the language model prompt contains only the natural language question. The graph reaches the model through the encoder pathway: node attributes are embedded, the relation-aware graph transformer produces the node memory, and the reader injects retrieved evidence into the graph-query token hidden states between frozen layer groups. There is no node text, no edge list, and no adjacency description appears among the input tokens.

B.1 ESTABLISHED BENCHMARKS

ogbn-arxiv is the large citation network released with the Open Graph Benchmark (Hu et al., 2020). Following GraphTranslator (Zhang et al., 2024), each example contains a sampled two-hop citation ego-graph, and the model generates the queried paper's human-readable arXiv computer-science area. The task evaluates whether title, abstract, and citation context can be combined for node-centric semantic inference, but its output remains restricted to a known 40-area ontology. **Cora** is the seven-class citation network of Sen et al. (2008). We adopt the text-attributed two-hop single-focus protocol used by LLaGA (Chen et al., 2024b). The model names the queried paper's topic from its own text and a compact citation neighborhood. Because the center text and citation homophily are already informative, this benchmark primarily measures whether graph context improves a bounded topic-generation task.

ExplaGraphs originates from the structured commonsense benchmark of Saha et al. (2021). Under G-Retriever's GraphQA adaptation (He et al., 2024b), the model receives a compact directed explanation graph and generates `support` or `counter` to indicate the relation between the two arguments. It probes relational-polarity reasoning, although the arguments themselves provide substantial text-only signal. **SceneGraphs** is derived from GQA (Hudson & Manning, 2019) and converted to graph question answering by G-Retriever. Object and attribute descriptions form nodes, typed spatial or action relations form edges, and the output is a short natural-language answer. This task stresses fine-grained object–attribute–relation binding and compositional reasoning rather than only recovering a global graph summary.

B.2 OUR PROPOSED BENCHMARKS

NeighborhoodQA is constructed from ogbn-arxiv to move beyond predicting one label for one center node. Given the exact sampled citation ego-graph released to the model, the target is every distinct CS area represented among the sampled papers cited by the center paper. The answer therefore has variable cardinality: the model must identify the relevant one-hop neighbors, aggregate and deduplicate their labels, infer how many outputs are required, and terminate only after producing the complete set. Gold answers are computed programmatically from the displayed graph and evaluated by macro set-F1 and exact-set match. Matched center-node-only and hop-restricted controls preserve the question and answer while removing required neighborhood evidence, making the contribution of graph access directly measurable.

StructuralAnomaly targets anchor-free structure–semantics binding using planted-partition stochastic block models (Holland et al., 1983). Every graph contains five equally sized thematic communities and all five themes occur in every example. In the answerable arm, exactly one community is substantially denser, but the question names no node or community. Topology is needed to localize the anomalous region, while node text is needed to name its theme. Each theme is selected uniformly, fixing chance accuracy at 20%. The matched control removes the density contrast and assigns the target uniformly, so it is unanswerable by construction. Above-chance control accuracy would reveal leakage. The released task deliberately isolates local density-based localization and should not be interpreted as evidence of unrestricted long-range symbolic graph reasoning.

B.3 DATASET-SPECIFIC EXPERIMENTAL DETAILS

Table 6 summarizes the key dataset statistics, prediction targets, and evaluation metrics. We further introduce the source graph and task, example construction and splits, QA format, and scoring protocol

Table 6: Benchmark statistics and reasoning coverage. Split entries are training/validation/test unless noted otherwise. NeighborhoodQA and StructuralAnomaly are introduced in this work.

Benchmark	Split	Encoded graph input	Generated target / metric	Principal capability or control
Established benchmarks				
ogbn-arxiv	20,000 / 2,000 / 4,000	Fanout-10 two-hop citation ego-graph; 45.5 nodes and 56.1 edges on average.	One of 40 CS areas; Top-1/3/5 accuracy.	Node-centric topic inference from paper text and citation context.
Cora	1,624 / 542 / 542	Fanout-10 two-hop citation ego-graph; 17.0 nodes and 23.6 edges on average.	One of seven topics; accuracy.	Local neighborhood-conditioned topic inference.
ExplaGraphs	1,659 / 553 / 554	Directed explanation graph; 5.17 nodes and 4.25 edges on average.	`support` or `counter`; accuracy.	Commonsense relational-polarity reasoning; text-only signal is an important boundary.
SceneGraphs	Official 60/20/20% by image ID	Object–attribute graph with typed relations; 19.13 nodes and 68.44 edges on average.	Short natural-language answer; normalized accuracy.	Grounded object–attribute–relation binding and compositional reasoning.
Benchmarks proposed in this work				
NeighborhoodQA	19,902 / 1,999 / 3,993	Two-hop citation ego-graph; 56.9 nodes, 71.1 edges, and 7.39 sampled one-hop cited papers on average.	Variable-length set of CS areas; macro set-F1 and exact-set match.	Multi-node aggregation and variable-cardinality generation; center-only and hop-restricted controls.
StructuralAnomaly	8,000 / 1,000 / 2,000 per arm	60 nodes in five 12-node communities; 128.8 undirected edges on average in the answerable arm and 102.4 in the control.	Community theme; answer accuracy. Expected chance accuracy: 20%.	Anchor-free structural localization and semantic identification; density-free control.

for each benchmark. The QA examples are concrete instances drawn from the corresponding test sets. Importantly, REGRAPH never serializes the graph into the LLM input sequence. Instead, graph information is iteratively read and injected into the graph-query hidden states between Transformer layer groups.

B.3.1 OGBN-ARXIV

Source graph and task. ogbn-arxiv is the citation network in the Open Graph Benchmark (Hu et al., 2020). It contains 169,343 papers and 1,166,243 citation edges. Each node has a paper title and abstract and is assigned to one of 40 arXiv computer-science areas. Following GraphTranslator (Zhang et al., 2024), the prediction target is the human-readable area name rather than an integer class identifier.

Example construction and splits. We draw 20,000 training centers from the official OGB training partition and 2,000 validation centers from the official validation partition. The test set is the label-proportional 4,000 paper subset used by GraphTranslator. For every center paper, we sample a fanout-10 two-hop citation ego-graph, yielding 45.5 nodes per example on average. Node attributes

concatenate the title and abstract, citation edges use the relation `cites`, and the queried paper is marked as the source node.

QA example.

> **Context.**
> `Center paper: Interpretable MTL from heterogeneous domains using boosted tree.`
> **Question.**
> `This is a paper from the arXiv computer science collection, shown together with the papers it cites. Which arXiv CS subject area does it belong to?`
> **Gold answer.** `Machine Learning`

Scoring. The generated answer is normalized and matched against the 40 canonical area names. Candidate labels are ranked using likelihood ranking. Top-k accuracy measures whether the gold area appears among the first k candidates, for $k \in \{1, 3, 5\}$.

B.3.2 CORA

Source graph and task. Cora (Sen et al., 2008) contains 2,708 machine-learning papers connected by 5,429 citation links. We use the text-attributed single-focus setting adopted by LLaGA (Chen et al., 2024b). The model generates one of seven topic names: Case Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning, or Theory.

Example construction and splits. Title and abstract attributes are aligned with the original Cora citation graph, and the data are divided into 1,624/542/542 training, validation, and test examples. Each queried paper is represented by a fanout-10 two-hop ego-graph with approximately 17 nodes on average. The center paper is marked as the source node, and every edge carries the relation `cites`. The split and node ordering are fixed before ego-graph sampling so that paper text, labels, and citation neighborhoods remain aligned.

QA example.

> **Context.**
> `Center paper: Applications of machine learning: a medical follow-up study.`
> **Question.**
> `This is a paper from the Cora citation network, shown with the papers it cites. Which machine-learning topic does it belong to?`
> **Gold answer.** `Neural Networks`

Scoring. Predictions are normalized and matched against the seven canonical topic names, and accuracy is reported over the test examples. Because the output vocabulary is small and a paper's own text is often highly predictive, the score measures bounded neighborhood-conditioned topic generation rather than unrestricted open-vocabulary reasoning.

B.3.3 EXPLAGRAPHS

Source graph and task. ExplaGraphs was introduced for structured commonsense reasoning (Saha et al., 2021). We use the GraphQA conversion of G-Retriever (He et al., 2024b). Each instance contains two arguments and a compact directed explanation graph whose nodes are commonsense concepts and whose typed edges express relations between them. The model must generate either `support` or `counter`.

Example construction and splits. The benchmark contains 2,766 examples divided into 1,659/553/554 training, validation, and test instances. Graphs have 5.17 nodes and 4.25 directed edges on average. Native triples of the form `(source; relation; target)` are converted into relation-labeled edges and passed to the encoder, while the two arguments and the polarity instruction form the language question.

QA example. A `counter` instance from the released test split.

> **Context.**
> `Concepts: computers, glitches, delay progress, allowance, cause problems, loss of resources, should make allowance; 7 nodes and 6 relation-labeled edges.`
> **Arguments.**
> `Argument 1: We should make allowance for glitches.`
> `Argument 2: Glitches delay progress and cause problems.`
> **Question.**
> `Do Argument 1 and Argument 2 support or counter each other? Answer with one word: 'support' or 'counter'.`
> **Gold answer.** `counter`

Scoring. The generated response is lowercased and matched to the first legal polarity label, and test accuracy is reported. Since the arguments themselves often reveal whether the relation is supportive or opposing, high accuracy should be interpreted together with graph-use diagnostics.

B.3.4 SCENEGRAPHS

Source graph and task. SceneGraphs is derived from GQA (Hudson & Manning, 2019) and follows the GraphQA conversion of G-Retriever (He et al., 2024b). Each graph represents one image. Nodes encode object names, attributes, and bounding boxes, while directed edges encode spatial or action relations such as `left of`, `on`, `holding`, and `wearing`. The model receives a natural-language question and generates the associated short answer.

Example construction and splits. We retain 100,000 question–graph pairs. Scene graphs contain 19.13 nodes and 68.44 typed edges on average. The 60/20/20 split is performed over image identifiers rather than individual question rows, ensuring that all questions associated with the same scene graph remain in one subset. The test split contains 20,025 examples. Canonical REGRAPH uses the complete scene graph for each image without a serialized text channel.

QA example.

> **Context.**
> `Node 7: name: calculator; attribute: gray, plastic; (x,y,w,h): (348,224,71,55)`
> **Question.**
> `What object is to the left of the gray plastic calculator?`
> **Gold answer.** `charger`

Scoring. The generated short answer is normalized and compared with the gold answer using exact answer accuracy under the G-Retriever protocol. The principal difficulty is preserving the binding among an object, its attributes, and its typed relations. Full-graph results are therefore distinguished from baselines that first retrieve a much smaller subgraph.

B.3.5 NEIGHBORHOODQA

Source graph and task. NeighborhoodQA is constructed from ogbn-arxiv (Hu et al., 2020). Instead of predicting the center paper's own area, it asks for the set of distinct CS areas represented among the sampled one-hop papers cited by that center. The target is a variable-length free-text list, so the model must aggregate multiple neighbors, remove duplicate areas, and decide when the answer set is complete.

Example construction and splits. Each instance uses a fanout-10 two-hop citation ego-graph and contains at least three sampled one-hop cited papers. Gold areas are computed from the exact sampled one-hop neighborhood presented to the model, rather than from the center paper's full neighborhood in the source graph. The benchmark contains 25,894 examples divided into 19,902/1,999/3,993 training, validation, and test instances. An example has 56.9 nodes, 71.1 edges, and 7.39 one-hop cited papers on average. Gold answers contain 2.28 distinct areas on average and range from one to eight.

QA example. A test instance with nine sampled cited papers spanning three areas.

> **Context.**
> `Center paper: Self-supervised dialogue learning. Its nine sampled cited papers include Deep contextualized word representations, Effective approaches to attention-based neural machine translation, ...`
> **Question.**
> `This is a paper from the arXiv computer science collection, shown with the papers it cites. Which arXiv CS subject areas appear among those cited papers? List every area that appears.`
> **Gold answer.** `Computation and Language, Computer Vision, Machine Learning`

Scoring. Prediction and target strings are normalized and parsed into sets by matching the 40 canonical area names. Ordering, separators, and surrounding prose do not affect the result. Per-example set-F1 is averaged over the test set, and exact-set match is reported alongside it. The center-node-only and hop-restricted controls retain the question and target while removing the neighborhood evidence needed to recover the complete set. The corresponding performance gap measures the value of graph access.

B.3.6 StructuralAnomaly

Source graph and task. StructuralAnomaly is generated from planted-partition stochastic block models (Holland et al., 1983). Every graph contains five communities of 12 nodes, for 60 nodes in total. The communities correspond to cryptography, jazz music, marine biology, textile manufacturing, and volcanology, with each node carrying a short phrase from its community theme (for instance `basaltic lava flow rheology` for volcanology or `elliptic curve key exchange` for cryptography). The question names no node or community, and all nodes receive the anchor-free role. The model must first localize the anomalously dense community from topology and then identify its theme from node text.

Example construction and splits. In the answerable arm, ordinary within-community edges are sampled with probability $p_{\text{in}} = 0.12$, edges inside one uniformly selected anomalous community with $p_{\text{hot}} = 0.60$, and cross-community edges with $p_{\text{out}} = 0.01$. Each theme is equally likely to be anomalous. The matched control retains the same graph size, theme set, question, split sizes, and target distribution but removes the density contrast and samples the target uniformly, making the question unanswerable from the input. Each arm contains 8,000/1,000/2,000 training, validation, and test examples. Because all five themes occur in every graph and each is selected uniformly, chance accuracy is exactly 20%.

QA example. The example below illustrates the question and answer format. Community labels are not supplied to the model.

> **Context.**
> `60 nodes in five 12-node communities.`
> **Question.**
> `This graph shows a collection of items and the links between them. One group of closely related items is far more densely interconnected than any other group. What is that group about?`
> **Gold answer.** `textile manufacturing`

Scoring. The response is normalized and matched against the five legal theme names. We report answer accuracy by matching the generated response against the five canonical theme names. Above-chance control accuracy would indicate leakage and invalidate the structural interpretation of the main score. The released setting isolates an anchor-free local density cue.

C Comparison Protocols and Implementation

The GraphQA datasets (ExplaGraphs and SceneGraphs) follow the preprocessing and answer normalization of G-Retriever (He et al., 2024b). ogbn-arxiv uses GraphTranslator's released 4,000-node test subset and two-hop sampling protocol (Zhang et al., 2024). Cora follows the per-dataset TAPE (He et al., 2024a) and LLaGA (Chen et al., 2024b) single-focus setting. For NeighborhoodQA and StructuralAnomaly, matched controls isolate the contribution of the graph evidence required by each task.

The default configuration freezes Llama-3.2-3B-Instruct (Grattafiori et al., 2024), with a hidden dimension of 3,072 and 28 Transformer layers, and updates only the graph-side components. ReGraph uses eight graph-query tokens, three graph-reading rounds, and a maximum topology-diffusion depth of $K = 2$. Node and relation attributes are initialized with `all-roberta-large-v1` (Reimers & Gurevych, 2019) and processed by a four-layer relation-aware Graph Transformer with a hidden dimension of 1,024 and four attention heads. The Topology-Diffused Graph Reader uses a 1,024-dimensional hidden space and eight attention heads, and the `Reader` and `Fuse` parameters are shared across reading rounds. We optimize the answer-only autoregressive objective with AdamW (Loshchilov & Hutter, 2019), using a learning rate of 10^{-5}, weight decay of 0.05, a 5% warm-up followed by cosine decay, a training batch size of four, gradient clipping at 1.0, and at most ten epochs. We select the checkpoint with the lowest validation loss and apply early stopping with a patience of two epochs. At inference, all graph-reading rounds are executed once during prefill, followed by greedy autoregressive decoding. The graph is represented exclusively through the graph encoder and iterative reader and is never serialized into the LLM context.

Environment. All experiments are run on an NVIDIA RTX PRO 6000 Blackwell GPU. The frozen Llama-3.2-3B-Instruct backbone runs in bfloat16 with gradient checkpointing. Table 7 reports training and evaluation times for each dataset.

Table 7: Evaluation and training time across datasets.

Dataset	Evaluation	Training
ogbn-arxiv	9 min	45 min
Cora	1 min	9 min
ExplaGraphs	$<$ 1 min	3 min
SceneGraphs	21 min	2.3 h
NeighborhoodQA	4 min	48 min
StructuralAnomaly	2 min	20 min

D Baselines

This section describes the 20 baseline configurations reported in Tables 1 and 2. The node-centric baselines are adopted from GraphTranslator and LLaGA, whereas the graph-level baselines follow the experimental protocols of G-Retriever.

D.1 Node-Centric Baselines

BERT (Devlin et al., 2019). BERT is an encoder-only Transformer pretrained with bidirectional masked-language modeling. Following the zero-shot protocol of GraphTranslator, the model embeds the queried paper description and the candidate category descriptions and predicts the category with the highest semantic similarity.

RoBERTa (Liu et al., 2019). RoBERTa strengthens BERT through a more extensively optimized pretraining procedure. In the GraphTranslator protocol, the pretrained encoder is used without downstream fine-tuning to compare the paper description against textual descriptions of the candidate labels, and citation edges are not explicitly modeled.

BERT (fine-tuned). This baseline initializes the text encoder from pretrained BERT and further optimizes it on labeled examples from the target benchmark. It provides a supervised text-only reference that can adapt to the target label distribution but does not explicitly incorporate the citation topology.

RoBERTa (fine-tuned). The fine-tuned RoBERTa baseline adapts the pretrained RoBERTa encoder to the labeled training split of the target task. As with fine-tuned BERT, its prediction is based on textual information and therefore does not directly model relations between the queried paper and its neighbors.

LLM + s_v. This text-only LLM baseline appends the queried node's textual attribute s_v to the task instruction and asks the language model to generate the answer. In the original GraphTranslator experiments, ChatGLM2-6B (GLM et al., 2024) is used as the language model, and no neighboring node information or learned graph representation is provided (Zhang et al., 2024).

LLM + $s_v + s_{N(v)}$. This baseline extends the preceding configuration by concatenating the raw text of the queried node and its sampled neighbors, $s_v + s_{N(v)}$, into the language model prompt. It exposes neighborhood content to the LLM, but topology is represented only implicitly through textual concatenation, without a graph encoder or a learned graph–language interface.

GraphTranslator (Zhang et al., 2024). GraphTranslator first uses a pretrained graph model to encode the queried node and its neighborhood. A Producer constructs textual descriptions of the node, its neighbors, and their commonalities, and a query-token-based Translator is trained in two stages to map the resulting graph representation into the frozen LLM's token space. The translated representation acts as a soft graph prompt for answer generation.

GPT-3.5-Turbo. GPT-3.5-Turbo is the general-purpose LLM reference adopted in the LLaGA (Chen et al., 2024b) study. It solves graph tasks through natural-language descriptions and prompting, without a learned graph encoder, structure-aware node template, or graph-to-token projector.

LLaGA-ND-7B (Chen et al., 2024b). LLaGA-ND uses the Neighborhood Detail template to convert a node-centered subgraph into a fixed-shape sequence. It samples a two-hop computational tree with ten neighbors per hop, retains separate representations for the sampled nodes, and projects the resulting sequence into the token space of a frozen Vicuna-7B (Chiang et al., 2023) model. Only the graph-to-language projector is optimized.

LLaGA-HO-7B (Chen et al., 2024b). LLaGA-HO uses the Hop-Field Overview template, which aggregates all nodes at each hop into one hop-level representation. Four hop representations summarize the center node and its progressively wider receptive fields before being mapped into the frozen Vicuna-7B token space. Compared with LLaGA-ND, this variant emphasizes a compact neighborhood overview instead of preserving individual neighbor details.

D.2 Graph-Level QA and Structural Localization Baselines

Zero-shot. The zero-shot baseline serializes the textual graph and combines it with the question and task instruction. A frozen LLM model is then asked to generate the answer directly, without demonstrations, graph-specific training, or trainable graph components.

Zero-CoT (Wei et al., 2022). Zero-CoT uses the same textualized graph and frozen LLM as the zero-shot baseline but appends the instruction "Let's think step by step." This generic reasoning cue encourages the model to produce an intermediate reasoning process without requiring demonstrations or parameter updates (Kojima et al., 2022).

CoT-BAG (Fatemi et al., 2024). CoT-BAG applies Build-a-Graph prompting to the serialized graph. It instructs the LLM to first construct a graph with nodes and edges before answering, thereby encouraging the model to explicitly organize the structural information already present in the textual input. The model itself remains frozen.

KAPING (Baek et al., 2023). KAPING is a zero-shot knowledge-augmented prompting method originally designed for knowledge-graph question answering. It retrieves the triples most relevant to the question, verbalizes them, and prepends them to the LLM input. Its retrieval operates over textual facts and does not learn a graph encoder or graph-conditioned soft prompt.

Prompt tuning. This configuration prepends a small set of trainable continuous prompt tokens to the textualized graph and question while keeping all LLM parameters fixed. In the G-Retriever protocol, ten virtual tokens are optimized from the task supervision, and no graph encoder or explicit graph retrieval module is used.

GraphToken (Perozzi et al., 2024). GraphToken represents graph information as a learned soft prompt for a frozen LLM. A graph encoder compresses the input graph into a pooled graph representation, which is projected into the LLM embedding space. Unlike G-Retriever, it does not first retrieve a query-specific subgraph and must summarize the graph through a compact latent interface.

G-Retriever (He et al., 2024b). G-Retriever first embeds the question and the textual attributes of graph nodes and edges, retrieves semantically relevant elements, and constructs a compact connected subgraph through Prize-Collecting Steiner Tree optimization. The retrieved subgraph is both encoded into a soft graph token and serialized as text for the generator.

LoRA (Hu et al., 2022). This baseline applies low-rank adaptation directly to Llama-3.2-3B-Instruct using the graph-question-answer training data. The graph is supplied through its textual serialization, while no graph encoder, query-specific subgraph retrieval, or graph soft prompt is introduced. Only the inserted low-rank parameter matrices are optimized.

G-Retriever with LoRA. This configuration combines G-Retriever's node-and-edge retrieval, PCST subgraph construction, graph encoder, and textualized retrieved evidence with LoRA adaptation of the LLM. It therefore differs from the frozen G-Retriever variant by allowing the language model to adapt to the downstream GraphQA supervision through low-rank parameter updates.

GRAG (Hu et al., 2025). GRAG retrieves query-relevant textual subgraphs using a graph-specific divide-and-conquer procedure. The retrieved structure is encoded through two complementary channels: a graph encoder produces a topology-aware soft prompt, while hierarchical textual descriptions preserve node and edge semantics as a hard prompt. The LLM generates its response from both forms of retrieved evidence.

E Analytical Protocols and More Experiments

E.1 Graph Dependence Metrics

Given an original example (G, q, Y), we construct a perturbed graph G' while keeping the question q unchanged. The reference answer Y' is recomputed from the perturbed graph. We consider two types of perturbations. Answer-changing perturbations modify the evidence required by q, such that $Y' \neq Y$. Answer-preserving perturbations modify graph content irrelevant to q, such that $Y' = Y$. Perturbations are generated independently of model predictions, and examples for which the edited graph yields an ambiguous or invalid answer are discarded.

We instantiate this principle according to the semantics of each benchmark. For SceneGraphs, answer-changing perturbations modify an object attribute or relation required by the question, whereas answer-preserving perturbations edit an unrelated object or relation. We retain only edits for which the resulting scene graph admits an unambiguous reference answer. For NeighborhoodQA, we modify citation edges or neighboring papers and recompute the complete set of subject areas in the queried neighborhood. An answer-changing edit must alter this target set, whereas an answer-preserving edit leaves it unchanged. For StructuralAnomaly, we modify graph connectivity so that the uniquely anomalous region changes for answer-changing perturbations, while answer-preserving perturbations alter other edges without changing the anomalous region. In all cases, G' is re-encoded and evaluated with a fresh prefill so that predictions depend only on the perturbed graph.

Let $\hat{Y}$ and $\hat{Y}'$ denote the responses generated from (G, q) and (G', q), respectively. For a collection $\mathcal{D}$ of perturbation pairs, we compute:

$$\mathrm{PairAcc}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(G,G',q,Y,Y') \in \mathcal{D}} \mathbb{I}[\hat{Y} \equiv Y] \mathbb{I}\left[\hat{Y}' \equiv Y'\right], \tag{20}$$

where $\equiv$ follows the benchmark-specific answer criterion: normalized exact-answer correctness for SceneGraphs and StructuralAnomaly, and exact set equality for NeighborhoodQA, ignoring the order of generated subject areas. PairAcc therefore counts a pair as correct only when the model answers both the original and perturbed graph correctly. We report it separately for answer-changing and answer-preserving perturbations. The former measures whether the prediction correctly follows changes in relevant graph evidence, while the latter measures whether it remains correct under irrelevant graph changes.

E.2 Support Attention

For each graph–question pair, let $\mathcal{V}_\star \subseteq V$ denote its supporting nodes, determined independently of model predictions and attention. For SceneGraphs, these are the nodes involved in a verified question-execution trace, including intermediate relation endpoints. For NeighborhoodQA, they are the papers satisfying the question's citation criterion within the sampled input graph. Support annotations are used only for evaluation and remain fixed across reading rounds. Using the topology-diffused relevance distribution $\widetilde{S}^t$ defined in the method, we compute:

$$M_t = \frac{1}{N_B} \sum_{i=1}^{N_B} \sum_{v \in \mathcal{V}_\star} \widetilde{S}^t_{iv}. \tag{21}$$

Because each row of $\widetilde{S}^t$ sums to one, $M_t \in [0, 1]$ measures the fraction of reader attention assigned to supporting nodes, averaged across graph-query tokens. For the multi-head implementation, this quantity is computed separately for each head and then averaged.

E.3 Backbone Analysis

Table 8: Backbone comparison for ReGraph.

Benchmark	Metric (%)	3B	8B
ogbn-arxiv	Top-1	72.28	71.75
Cora	Accuracy	88.99	86.72
ExplaGraphs	Accuracy	90.43	92.42
SceneGraphs	Accuracy	53.24	51.83
NeighborhoodQA	Set-F1 / exact	83.83 / 44.93	83.87 / 45.28
StructuralAnomaly	Accuracy	99.15	99.05

We study backbone sensitivity by replacing Llama-3.2-3B-Instruct with Llama-3.1-8B-Instruct. As shown in Table 8, the larger backbone improves ExplaGraphs, yields nearly unchanged performance on NeighborhoodQA and StructuralAnomaly, and underperforms the 3B variant on the remaining benchmarks. These observations suggest that switching to a larger LLM does not automatically resolve graph reasoning errors, consistent with graph-evidence extraction and integration remaining potential bottlenecks. Meanwhile, comparable neighborhood synthesis and anchor-free localization performance show that ReGraph can support these capabilities without relying on the larger backbone.

E.4 Visibility Controls

We further investigate whether ReGraph benefits from graph evidence beyond the focal node and whether this advantage persists when the required information lies farther away in the graph. We construct two controlled variants of NeighborhoodQA while keeping the question, target answer, and data split fixed. For one-hop area synthesis, the restricted model observes only the focal paper,

Table 9: Visibility controls. We restrict how much of the same sampled graph is visible to REGRAPH while keeping questions, answers, and splits unchanged. Exact denotes exact-set match.

Task	Visible graph	Set-F1	Exact
One-hop area synthesis	Focal node only	72.61	26.67
	Full neighborhood	**83.83**	**44.93**
Second-hop exclusive areas	Up to 1 hop	48.53	1.36
	Up to 2 hops	**56.55**	**2.74**

whereas the full model accesses its sampled citation neighborhood. For second-hop exclusive areas, the answer contains subject areas that appear only at the second citation step and excludes those already present among direct neighbors; we therefore compare a one-hop view against the complete two-hop graph. Graph restriction is applied before graph encoding, so withheld nodes cannot influence either the node memory or the topology operator.

As shown in Table 9, expanding graph visibility consistently improves performance in both settings. The effect is especially pronounced for complete answer recovery in one-hop synthesis, suggesting that the focal paper often provides enough semantic signal to guess some neighboring topics, but the graph is necessary to recover the full set reliably. The second-hop task provides a stronger control: restricting the graph to one hop removes the nodes from which the target areas are defined, yet the model still achieves non-trivial set-F1, reflecting predictable topical correlations in citation graphs. Exposing the actual second-hop evidence yields a clear additional gain, showing that REGRAPH can exploit evidence beyond the focal node's immediate neighbors. Nevertheless, the low exact-match scores indicate that exhaustively synthesizing all distant evidence remains substantially harder than recovering part of the answer.

E.5 GENERALIZATION

We study whether iterative graph reading transfers beyond the graph sizes and reasoning demands encountered during training. The protocol separates graph size, dependency depth, and evidence breadth. For size extrapolation, models are trained on graphs with 16–64 nodes and evaluated on held-out graphs with up to 512 nodes, keeping the question, supporting evidence, and answer fixed as irrelevant regions are added. We apply this test to SceneGraphs, NeighborhoodQA, and StructuralAnomaly. For depth extrapolation, SceneGraphs and NeighborhoodQA checkpoints are trained on one- and two-step relational dependencies and tested on deeper compositions at a fixed graph size of 128 nodes. A complementary NeighborhoodQA study holds graph size and citation depth fixed while increasing the number of distinct subject areas required in the answer, from one or two during training to four or eight at test time. REGRAPH retains $N_B = 8$, $T = 3$, and $K = 2$ without test-time parameter updates. The controls are LoRA, GraphToken, G-Retriever, and static replay.

Fig. 4 presents the results of this protocol. The size curves distinguish robustness to a growing search space from the ability to produce a correct answer: REGRAPH loses less performance as irrelevant regions accumulate, but G-Retriever remains more accurate on SceneGraphs. The increasing advantage over static replay beyond the training-depth range illustrates the complementary benefit that renewed evidence selection could provide when information must be composed across successive dependencies. Nevertheless, all methods deteriorate on the deepest questions, and complete answer recovery becomes difficult as the number of required facts grows.

F MORE OPEN-ENDED QA EXAMPLES

We complement Table 3 with free-form SceneGraphs prompts and reworded instructions on four benchmarks trained with fixed question templates. Responses are reproduced verbatim, including errors and incomplete sentences. [. . .] marks truncation. Graph excerpts and reference answers are displayed for interpretation only: REGRAPH accesses the encoded graph through its reader, without appending these summaries or references to the language input.

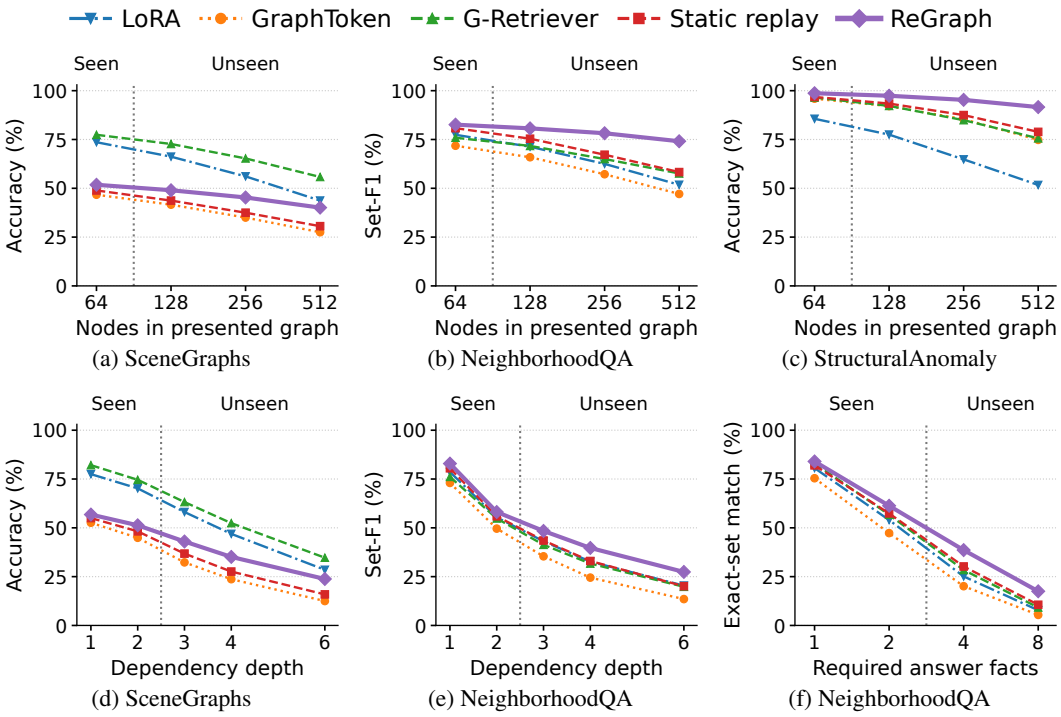

Figure 4: Generalization beyond the training regime. (a–c) Graph-size extrapolation with fixed supporting evidence. (d–e) Dependency-depth extrapolation at 128 nodes. (f) Increasing the number of distinct required subject areas at fixed graph size and citation depth. Dotted vertical lines separate training-covered (seen) and extrapolation (unseen) test regimes. Metrics are indicated per panel, with higher values better.

F.1 Free-Form Scene Questions

Table 10 groups additional scenes by four prompts covering description, short-story generation, object attributes, and likely location. Each prompt is applied to every scene. The graph-free baseline uses the same frozen LLM and receives the same question without scene information. Its response is therefore shown once per prompt. The examples are drawn from 300 randomly sampled test scenes, excluding those in Table 3. Within each prompt, we select four responses of 6–30 words with the highest number of supported mentions minus unsupported mentions, breaking ties by more supported mentions and then randomized order, while skipping verbatim repeats. The displayed cases therefore illustrate successful content use and remaining errors.

Green highlights denote object or attribute mentions supported by the scene graph, and red wavy underlines denote mentions absent from or contradicted by it. Basic color words are checked only when the corresponding color is explicitly annotated. A shared baseline mention is marked only when its status agrees across the four displayed scenes. Unmarked text is not certified as correct, and a highlighted object does not validate an accompanying action, emotion, or inferred location.

Table 10: Free-form SceneGraphs responses. Green marks graph-supported mentions; red wavy underlines mark unsupported mentions.

Case	Graph excerpt	REGRAPH response
(a) *Describe this scene in one or two sentences.*		
LLM w/o graph: I'm happy to help, but I don't see a scene described. [. . .]		
a1	street · city (white) · person (walking) · sidewalk · traffic light · . . .	the scene depicts a busy street with the city skyline in the background, and a person standing on the sidewalk, looking up at the traffic light.
a2	horse (brown, white) · field (green) · tree (large) · sky (cloudy) · . . .	The scene depicts a serene and peaceful landscape with a lone horse standing in a field, surrounded by tall trees and a bright blue sky.
a3	man (crouching) · street · motorcycle · helmet (white) · . . .	the man is standing in the middle of the street, surrounded by a halo of light, with the motorcycle parked next to him.
a4	table (white) · plate (white) · cheese (melted) · fork (metal) · . . .	The foreground is a large, dark-colored table, and in the background, there is a large, white plate.
(b) *Write one sentence that tells a short story about this scene.*		
LLM w/o graph: As she walked through the old mansion, Emily stumbled upon a mysterious room hidden behind a tattered tapestry, [. . .]		
b1	computer monitor (silver) · desk · man (small) · screen (computer) · . . .	The computer monitor on the desk in front of the man was displaying a blank screen, and he was staring at it with a mixture of frustration and disappointment.
b2	television · coffee table · couch (black) · room · . . .	The television was sitting on the coffee table in front of the couch, its screen glowing softly in the darkened room.
b3	man (old) · eye (small) · clock (gold) · wall (metal) · . . .	The man stood alone in the empty room, his eyes fixed on the old clock on the wall.
b4	man · camera · elephant (large) · car (red) · . . .	The man stood patiently behind the camera, waiting for the perfect shot of the elephant.
(c) *What objects stand out in this scene, and what colors are they?*		
LLM w/o graph: The objects that stand out in this scene are the red and blue objects. The red object is a fire truck, and the blue object is a police car. [. . .]		
c1	sheep (white) · hill (grassy) · bush (green) · fence (small) · . . .	The objects that stand out in this scene are the sheep and the hill. The colors of the sheep are white and the hill is green.
c2	couch (brown) · coffee table · chair (red) · cord (white) · . . .	The objects that stand out in this scene are the couch and the coffee table. The couch is brown and the coffee table is wood.
c3	coffee table (wood) · television (black) · chair (blue) · floor (brown) · . . .	The objects that stand out in this scene are the coffee table and the television. The coffee table is brown and the television is black.
c4	train · man · shirt (gray) · bandana (red) · . . .	The objects that stand out in this scene are the train and the man. The colors of the train are red and the man is wearing a blue shirt.

Continued on the next page.

Table 10: Free-form SceneGraphs questions (continued).

Case	Graph excerpt	REGRAPH response
(d) *Where was this photo most likely taken, and what makes you think so?*		
LLM w/o graph: Unfortunately, I can't identify the location of this photo. However, I can provide some clues that might help you narrow down the possibilities. The photo appears to be a scenic landscape with a large body of water in the foreground, and there are trees and hills in the background. [. . .]		
d1	grass (green) · trees (bare) · building (white) · zebra (white) · . . .	zoo, I think so because the grass is green and there are trees in the background, which is typical of a zoo.
d2	road · truck (white) · power lines (overhead) · telephone pole (large) · . . .	road, and I think so because of the truck
d3	computer · screen (computer) · keyboard (gray) · box (gray) · . . .	computer lab Reason: photo is of a computer screen
d4	shelf · books · cat (black) · bookshelf (handmade) · . . .	store, I think so because the store is in the background and there is a shelf of books in the foreground.

The responses change with the supplied graph, identifying different objects under the same prompt, while the graph-free baseline cannot distinguish these scenes. However, relevant object mentions coexist with unsupported elaboration: the story responses introduce emotional states, and the location responses infer settings from a few objects. The train example also assigns a blue shirt where the graph specifies gray. These cases distinguish graph-conditioned content from fully faithful generation and complement the quantitative faithfulness analysis in Section 4.5.

F.2 REWORDED BENCHMARK QUESTIONS

Table 11 replaces the fixed training instructions on ExplaGraphs, Cora, NeighborhoodQA, and StructuralAnomaly with free-form requests concerning the same stance, topic, topic set, or community theme. ExplaGraphs retains the two arguments in the language input, as in the native task. Each block reports the benchmark's own metric over all probed test inputs, including the graph-free and graph-as-text comparisons. These scores concern the reworded prompts and are separate from the native benchmark results in Tables 1 and 2. They do not measure the correctness of every sentence in an explanation.

For qualitative inspection, each block shows two inputs with different reference answers on which REGRAPH predicts the correct target. Selection follows a fixed random order, preferring sentence responses of at least eight words without repetition loops when available. Green and red annotations evaluate the indicated target labels, not the entire response. Reference answers occupy a separate row to distinguish evaluation information from the graph supplied to the model.

Table 11: Reworded questions on fixed-template benchmarks. Scores (%) summarize all probed inputs; highlighting evaluates target labels.

	Example 1	Example 2
ExplaGraphs Stance accuracy; 554 test inputs. Scores (%): REGRAPH **82.3**; LLM w/o graph 44.6; graph as text 45.1.		
Question	*In one sentence, explain why argument 2 supports or counters argument 1.*	
Arguments	*Arg. 1:* There should not be parades that celebrate homosexuality. *Arg. 2:* Nobody should be restricted from expressing their point of view.	*Arg. 1:* Television is fun and part of us already *Arg. 2:* Almost everyone has a television in their house
Graph	parades · expressing · point of view · sexuality · homosexuality · free speech	television · everyone · fun · house
Reference	counter	support
LLM w/o graph	Argument 2 counters argument 1 because it asserts that individuals should not be restricted from expressing their point of view, implying that freedom of expression should not be limited based [. . .]	Argument 2 counters argument 1 because if almost everyone has a television in their house, it implies that television is not a unique or special part of our lives, but [. . .]
REGRAPH	Argument 2 counters argument 1 because it states that nobody should be restricted from expressing their point of view, implying that everyone, including those who support parades celebrating homosexuality, should [. . .]	Argument 2 supports argument 1 because almost everyone has a television, which means it's a common and widespread part of people's lives, making it fun and enjoyable.
Cora Topic accuracy; 542 test inputs. Scores (%): REGRAPH **55.2**; LLM w/o graph 28.2; graph as text 30.1.		
Question	*What method or technique does this paper study?*	
Graph	"Asking Questions to Minimize Errors" + 23 cited papers	"Landscapes, Learning Costs and Genetic Assimilation." + 13 cited papers
Reference	Theory	Genetic Algorithms
LLM w/o graph	This paper studies the method of "Deep Learning" for image classification tasks. [. . .]	
REGRAPH	Theory Explanation: This paper studies the theory of the game.	Genetic Algorithms Explanation: The paper studies Genetic Algorithms. [. . .]
NeighborhoodQA Set-F1; 200 test inputs. Scores (%): REGRAPH **66.3**; LLM w/o graph 28.7; graph as text 37.8.		
Question	*What research topics do the papers cited here cover?*	
Graph	"duration robust weakly supervised sound event detection" + 60 papers in its citation neighbourhood	"list decoding random euclidean codes and infinite constellations" + 53 papers in its citation neighbourhood
Reference	Computer Vision, Machine Learning, Sound	Computational Complexity, Information Theory
LLM w/o graph	The papers cited here cover research topics in the field of computer science, specifically in the areas of artificial intelligence, machine learning, and natural language processing. [. . .]	
REGRAPH	Computer vision, Machine learning, Neural networks, Sound engineering, Speech recognition	The papers cited here cover the following research topics: Computational Complexity, Data Structures, Information Theory, and Algorithms.

Continued on the next page.

Table 11: Reworded questions on fixed-template benchmarks (continued).

	Example 1	**Example 2**
StructuralAnomaly Accuracy; 200 test inputs. Scores (%): REGRAPH **92.0**; LLM w/o graph 0.0; graph as text 5.5.		
Question	*Describe the most densely connected group of items.*	
Graph	60 items in 5 unnamed groups; densest group: walking bass line phrasing, free jazz collective improvisation, . . .	60 items in 5 unnamed groups; densest group: secret sharing threshold scheme, side channel timing attack, . . .
Reference	jazz music	cryptography
LLM w/o graph	The most densely connected group of items is a single item. A single item is connected to itself, and there are no other items to connect to. [. . .]	
REGRAPH	jazz music	cryptography

REGRAPH scores above both comparisons across the four reworded-prompt probes, indicating that graph-conditioned predictions need not depend on the exact training instruction. Nevertheless, the responses remain strongly tied to the trained answer space. The Cora examples often repeat a topic label without explaining the method, NeighborhoodQA responses can add topics beyond the reference set, and StructuralAnomaly produces a theme label even when the prompt requests a description.

G AI USE STATEMENT

In this work, we used generative AI tools to edit a research paper to improve readability. We have not used generative AI tools to help develop theoretical models or conceptual frameworks. We have reviewed all AI-assisted work, and every sentence polished by LLMs have been checked. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.