Towards a Learnable Retriever for Textual Graph Question Answering

Anonymous ACL submission

Abstract

Textual graph question answering (GraphQA) and graph-based retrieval-augmented generation (GraphRAG) have gained increasing attention as a way to ground large language models (LLMs) on structured knowledge. Despite extensive efforts, existing methods suffer from two fundamental limitations. First (*coverage*), retrievers may fail to recall the evidence nodes required for reasoning, resulting in insufficient coverage of the gold answer set. Second (*compactness*), naive attempts to boost recall typically lead to an explosion of retrieved subgraph size, introducing excessive irrelevant information that compromises compactness and overwhelms the LLM. The gold answer set provides critical supervision in retrieving relevant information while filtering out irrelevant ones, which however, is rarely exploited by existing methods. In this work, we propose TALENT, a learnable retriever that utilizes gold answer signals to enhance both retrieval coverage and compactness. Specifically, we stratify graph nodes into three distinct levels: gold answers, answer-related nodes, and irrelevant noise. Afterwards, we adopt a weighted pseudo-label loss to prioritize gold answers and preserve answer-related nodes for coverage, while discarding irrelevant noises for compactness. Empirical results across two GraphQA datasets demonstrate that TALENT consistently improves downstream performance.

1 Introduction

Large language models (LLMs) have achieved impressive progress in general-purpose generation and reasoning (Thoppilan et al., 2022; Bang, 2023), but their outputs are ultimately bounded by what is encoded in the model parameters. Particularly, trained on large-scale open-domain corpora collected before a fixed cutoff date, LLMs may hallucinate or omit critical evidence when a query requires precise, up-to-date, or domain-specific knowledge (Zhao et al., 2023b; Wang et al., 2023).

To mitigate these limitations, Retrieval-Augmented Generation (RAG) has emerged as a dominant paradigm that grounds LLM outputs in external evidence by dynamically retrieving relevant knowledge snippets for context-aware generation (Lewis et al., 2020; Shi et al., 2024; Gao et al., 2023). While conventional RAG methods (Lewis et al., 2020; Shi et al., 2024) mainly retrieve evidence from unstructured text corpora, recent work has increasingly shifted toward textual graphs as a more expressive and structured knowledge source. This trend has spurred the development of Graph-based RAG (GraphRAG) and Graph Question Answering (GraphQA). By exploiting the explicit semantics and relational structure of textual graphs, graph-based RAG methods support more grounded reasoning and complex information synthesis (Logan IV et al., 2019; Luo et al., 2025; He et al., 2024).

Conventional retrieval methods in GraphRAG can be categorized into non-parametric, Graph neural network (GNN)-based, and LLM-based approaches. Non-parametric retrievers construct subgraphs using classical graph search algorithms or predefined rules (Taunk et al., 2023; Yasunaga et al., 2021). GNN-based methods encode graph structure with GNNs and identify task-relevant components to form the retrieved subgraph (Mavromatis and Karypis, 2024). LM-based retrievers utilized LM's ability in their retrieval process (Jiang et al., 2023).

However, these three categories rely primarily on query similarity, but largely ignored the critical signals provided by the gold answers. Hence, existing retrievers have two fundamental limitations that hinder their effectiveness, including *coverage* and *compactness*. An illustrative example is shown in Figure 1, where we partition the nodes into three groups: (A) *Gold answers* which refer to all nodes appear in the gold answer list provided by the dataset, (B) *answer-related nodes* that are

semantically close to the gold answers but are not themselves gold answers, and (C) *irrelevant noise* that is similar to the query, e.g., containing "Newton" or "Isaac Newton", yet is not answer-related.

First, *coverage* refers to the limitation that an existing query-based retriever may fail to cover the gold answer set in the retrieved subgraph (blue circle). As shown in Figure 1, the retriever only recovers two gold professions, *physicist* and *scientist*, while missing other correct professions such as *mathematician*, *astronomer*, *philosopher*, and *chemist*. This happens because similarity-based ranking tends to prioritize nodes that tightly match dominant query tokens ("Isaac Newton"), rather than nodes that directly express the professions-type answers. As a result, the retrieved nodes have low coverage of the gold answer set.

Second, *compactness* refers to the limitation that query-based retriever can introduce excessive irrelevant noise. This happens because query similarity is a weak proxy for *answer-related* evidence: especially in large graphs, many nodes can exhibit strong lexical or semantic overlap with the query. As shown in Figure 1, the retriever additionally pulls in a substantial amount of Newton-named but non-profession nodes, e.g., "Isaac Newton: The Last Sorcerer", "Sir Isaac Newton", and other Newton-centric mentions. These nodes inflate the retrieved subgraph, consume limited context budget, and can distract or mislead the downstream generator from the few signals that actually determine the correct professions.

A key observation is that the *gold answer set* provides precisely the missing supervision for retrieval. Unlike query-relevent retrieval methods, gold answers specify what should be covered, and can therefore help alleviate the two limitations. For *coverage*, gold answers identify the evidence nodes that are truly required for correct reasoning. Training the retriever with gold-answer signals encourages it to consistently include these answer-related nodes, rather than relying solely on naive query resemblance that may miss them. For *compactness*, gold answers also help distinguish *answer-related evidence* from other weakly related or even irrelevant nodes. Specifically, gold supervision can guide the model to prioritize nodes that explain or connect to the correct answers, while suppressing those that are unlikely to contribute to the correct answer. As a result, the retriever can remain recall-oriented on the right signals without inflating the retrieved subgraph. To date, such gold signals are rarely exploited to guide or train the retriever.

In this work, we propose TALENT, a learnable retriever that leverages gold answer signals to improve both retrieval coverage and compactness. We start from a rule-based candidate subgraph that provides a high-recall pool of nodes. To improve retrieval coverage, we train a retriever based on a weighted pseudo-label objective, incorporating gold answer signals to guide the retrieval process. To promote answer-critical evidence, we prioritize gold answer and down-weight other rest signals to improve retrieval compactness. Intuitively, the loss assigns the highest weight to gold answers to ensure coverage, assigns a much smaller weight to answer-related nodes to retain useful context, and ignores irrelevant noise to enforce compactness. This design injects gold signal supervision into retrieval process. After retriever training, the generator is fine-tuned on contexts constructed from the retriever outputs, translating improved retrieval into better downstream QA performance. Our contributions are summarized as follows:

- **Idea.** Gold answer signals effectively improve both coverage and compactness, outperforming retrieval based solely on query similarity.
- **Method.** We propose a learnable retriever trained with a weighted pseudo-label objective that leverages gold answers to guide retrieval. By prioritizing gold answer nodes and down-weighting weakly related nodes, the retriever leads to better downstream QA performance.
- **Evaluation.** We conduct extensive experiments on GraphQA benchmarks to validate both retrieval effectiveness and end-task QA performance. Our method outperform best competitor by at least 1.13%.

2 Preliminaries

Notation. We use calligraphic letters to denote sets, e.g., $\mathcal{V}$ and $\mathcal{E}$ for node and edge sets. A textual graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node $v \in \mathcal{V}$ has a textual attribute $\text{text}(v)$, e.g., an entity name or concept, and each edge $e = (u, r, v) \in \mathcal{E}$ denotes a relation r connecting the source node u to the destination node v. The textual attribute of an edge is denoted as $\text{text}(e)$. We use $\mathcal{S}$ to denote the subgraph of $\mathcal{G}$, where $\mathcal{S} \subseteq \mathcal{G}$ indicates that its node set $\mathcal{V}_\mathcal{S} \subseteq \mathcal{V}$ and its edge set $\mathcal{E}_\mathcal{S} \subseteq \mathcal{E}$. We use q for the query and a for the final answer from the generator. We use $a^\star$ to denotes the token sequence of the gold answer in the dataset and use

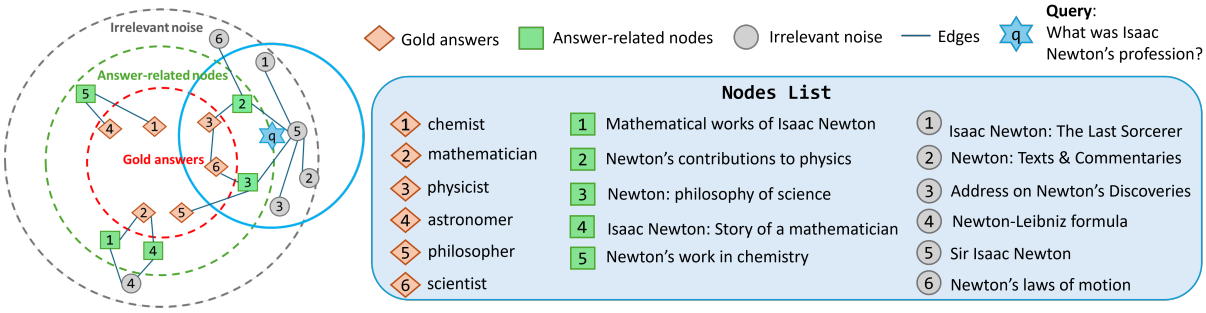

Figure 1: **An illustrative example of two limitations of query-based graph retrieval.** For the query "What was Isaac Newton's profession?", a query-similarity retriever returns a Newton-centric subgraph (blue circle). We partition the nodes into three groups: *Gold answers* (only a small subset of the gold answer set is retrieved by the subgraph in the blue circle, showing limited *coverage*), *Answer-related nodes* (topically relevant but not exact gold answers), and *Irrelevant noise* (nodes that match the query tokens such as "Newton" yet are not answer-related). The presence of substantial noise alongside partial gold coverage highlights the *compactness* issue: improving recall by expanding query-similar candidates can inflate the subgraph with distractors.

$\mathcal{V}^\star$ as the set of nodes appearing in the gold answer, i.e., the gold answer set. We denote the retriever and generator by R_θ and G_ϕ, respectively.

2.1 Problem Setup

We consider a general retrieve-then-generate setup (He et al., 2024) for textual graph question answering (GraphQA). Each instance in the dataset consists of a natural-language query q and a textual graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as the knowledge base. Due to the large scale of $\mathcal{G}$, it is impractical to directly serialized $\mathcal{G}$ into an LLM prompt, hence the system first retrieves a relevant *subgraph*, conditioned on which, an answer is further generated.

Formally, a retriever R_θ extracts a subgraph $\mathcal{S}$ from the textual graph $\mathcal{G}$ based on the query q:

$$\mathcal{S} = R_\theta(q, \mathcal{G}). \tag{1}$$

A generator G_ϕ then predicts the answer a based on the query q and the retrieved subgraph $\mathcal{S}$:

$$a = G_\phi(q, \mathcal{S}). \tag{2}$$

2.2 Candidate Connected Subgraph

For large textual graphs, directly serializing the full graph often exceeds the LLM context budget. Following G-Retriever (He et al., 2024), we first construct a *candidate connected subgraph* $\mathcal{S}$ using the Prize-Collecting Steiner Tree (PCST) algorithm (Bienstock et al., 1993), and use $\mathcal{S}$ as the input context for downstream modules.

Indexing and similarity retrieval. Given a query q and a textual graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we embed the query and textual attributes of node $\mathcal{V}$ and edges $\mathcal{E}$ using a sentence encoder such as Sentence-BERT (Reimers and Gurevych, 2019):

$$\mathbf{z}_q = \mathrm{LM}(q), \tag{3}$$
$$\mathbf{z}_v = \mathrm{LM}(\mathrm{text}(v)), \quad v \in \mathcal{V}, \tag{4}$$
$$\mathbf{z}_e = \mathrm{LM}(\mathrm{text}(e)), \quad e \in \mathcal{E}. \tag{5}$$

We compute cosine similarity scores and retrieve the top-k_v nodes and top-k_e edges:

$$\mathcal{V}_{\text{sim}} = \underset{v \in \mathcal{V},\, k=k_v}{\arg\operatorname{topk}} \cos(\mathbf{z}_q, \mathbf{z}_v), \tag{6}$$
$$\mathcal{E}_{\text{sim}} = \underset{e \in \mathcal{E},\, k=k_e}{\arg\operatorname{topk}} \cos(\mathbf{z}_q, \mathbf{z}_e), \tag{7}$$

where k_v and k_e control the number of retrieved nodes and edges, respectively.

Subgraph construction. We assign *prizes* to retrieved candidates based on their ranks, so higher-ranked items receive larger prizes. Concretely, for $v \in \mathcal{V}_{\text{sim}}$ with rank r_v we set $p(v) = k_v - r_v$ to prioritize those with higher similarity, and analogously define $p(e) = k_e - r_e$ for $e \in \mathcal{E}_{\text{sim}}$; items outside these retrieved lists receive zero prize. PCST then selects a connected subgraph $\mathcal{S} = (\mathcal{V}_\mathcal{S}, \mathcal{E}_\mathcal{S})$ that trades off prize coverage and compactness:

$$\mathcal{S} = \arg\max_{\substack{\mathcal{S} \subseteq \mathcal{G} \\ \text{connected}}} \Big(\textstyle\sum_{v \in \mathcal{V}_\mathcal{S}} p(v) + \sum_{e \in \mathcal{E}_\mathcal{S}} p(e) - c\,|\mathcal{E}_\mathcal{S}| \Big). \tag{8}$$

where c controls the size-coverage trade-off.

3 Method

Overview. Our proposed TALENT mainly consists of two parts: (i) a learnable nodes retriever R_θ trained with weighted pseudo-label supervision (Section 3.1), and (ii) an answer generator G_ϕ fine-tuned with standard Supervised Fine-Tuning (SFT) (Section 3.2). The framework for TALENT is shown in Figure 2. Given a query q and a full textual graph $\mathcal{G}$, we first apply a rule-based graph

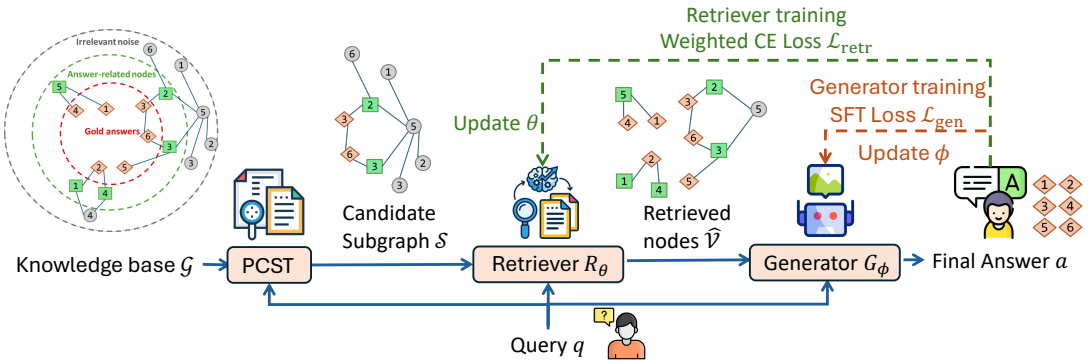

Figure 2: Overview of TALENT. The proposed TALENT includes the following steps: (1) Given Knowledge base $\mathcal{G}$ and query q, PCST retrieves a candidate subgraph $\mathcal{S}$. (2) Given the candidate subgraph $\mathcal{S}$ and query q, we train a learnable retriever R_θ with weighted pseudo-label supervision, producing $\hat{\mathcal{V}}$. (3) Given retrieved node set $\hat{\mathcal{V}}$ and q, we train a generator G_ϕ with standard SFT. (4) G_ϕ outputs the final answer a.

builder (PCST) to obtain an initial candidate connected subgraph $\mathcal{S}$ that fits the context budget. We then train a learnable retriever R_θ with weighted pseudo-label supervision to select answer-relevant nodes $\hat{\mathcal{V}}$ from $\mathcal{S}$ (Section 3.1). Using answer-relevant nodes $\hat{\mathcal{V}}$, we construct textual contexts and fine-tune a generator G_ϕ with standard supervised learning (Section 3.2).

3.1 Retriever Training

As discussed earlier, PCST and other query-similarity based retrievers often fail to cover the answer-related nodes and include irrelevant noise, resulting in limited *coverage* and poor *compactness*. We therefore use gold answer signals to guide retriever training, encouraging prioritization of gold answers while down-weighting other signals. We first describe TALENT's retriever prompting and then introduce the weighted pseudo-label objective.

Retriever prompting. We implement our learnable retriever R_θ as an instruction-following LLM (with LoRA) that performs node selection based on a provided candidate subgraph. Given a query q and the PCST subgraph $\mathcal{S}$, we prompt the LLM to select the candidate nodes that are most helpful for answering q. The LLM outputs should be exactly one line in the format of "`name1|name2|...`". Each `name` such as `name1` and `name2` here represents the names of the nodes R_θ finally selects. This constrained format makes the output directly interpretable as a retrieved nodes set, which we denote by $\hat{\mathcal{V}} = R_\theta(q, \mathcal{S})$. It is worth noting that since our retriever R_θ is an LLM, it leverages its inherent parametric knowledge during the selection process. Consequently, the output node set $\hat{\mathcal{V}}$ may include nodes that were not present in the initial candidate subgraph $\mathcal{S}$ (i.e., $\hat{\mathcal{V}} \not\subseteq \mathcal{V}_\mathcal{S}$). In our method, $\hat{\mathcal{V}}$ encompasses the complete set of nodes actually generated by R_θ, reflecting both the information from $\mathcal{S}$ and the LLM's internal knowledge.

Fine-tuning with weighted pseudo-label supervision. Recall that existing works in GraphRAG use retrievers that are largely query-similarity driven, which leads to two limitations: coverage and compactness. Our goal is to train the retriever to output the node set $\hat{\mathcal{V}}$ that achieves both high *coverage* and *compactness*: it should reliably include nodes that appears in the gold answer set $\mathcal{V}^\star$ (coverage) while avoiding redundant or weakly related candidates (compactness). The gold answer provides precisely the missing supervision, i.e., the gold signal, for retrieval. To inject gold signals into retrieval, we leverage the dataset-provided gold answers and stratify candidate nodes into three levels: (i) *gold answers*, which are candidate nodes that are overlapped with the gold answer set $\mathcal{V}^\star$, and provides evidence that the retriever should prioritize to support correct reasoning, (ii) *answer-related nodes*, which are candidate nodes within $\mathcal{S}$ but are not gold answers, and may still helpful for providing context, though they are less directly tied to the final answer, and (iii) *irrelevant noise*, which are nodes not contained in $\mathcal{S}$, and are completely irrelevant and should be suppressed to maintain compactness. Given the candidate subgraph $\mathcal{S}$ with node set $\mathcal{V}_\mathcal{S}$, and the gold answer set $\mathcal{V}^\star$, we utilize these nodes as weighted pseudo label supervision by defining

$$\mathcal{P} = \mathcal{V}^\star \cap \mathcal{V}_\mathcal{S} \quad \text{(strong positives; level-1)}, \quad (9)$$

$$\mathcal{W} = \mathcal{V}_\mathcal{S} \setminus \mathcal{V}^\star \quad \text{(weak positives; level-2)}. \quad (10)$$

All nodes outside $\mathcal{V}_\mathcal{S}$ are treated as irrelevant noise (level-3) during retriever training.

We train R_θ as conditional generation of a target string y that concatenates nodes in $\mathcal{P}$ and $\mathcal{W}$ (separated by the pipe symbol "|"; strong positives first,

then weak positives)

$$y = p_1 \mid \cdots \mid p_{|\mathcal{P}|} \mid w_1 \mid \cdots \mid w_{|\mathcal{W}|}, \quad (11)$$

where $p_i, \forall i = 1, \ldots |\mathcal{P}|$ are nodes in $\mathcal{P}$ and $w_j, \forall j = 1 \ldots |\mathcal{W}|$ are nodes in $\mathcal{W}$. Let x denote the retriever input prompt, which includes the query q and the textualized candidate subgraph $\mathcal{S}$, and let $y_{1:T}$ be the tokenized target sequence, where T is its token length. To prioritize level-1 (gold answer) nodes while still preserving level-2 (answer-related) candidates as a weak learning signal, we minimize the token-weighted negative log-likelihood:

$$\mathcal{L}_{\text{retr}}(\theta) = -\sum_{t=1}^{T} \omega_t \log p_\theta(y_t \mid x, y_{<t}). \quad (12)$$

Since we want to encourage R_θ to prioritize gold answer nodes while still allowing potentially useful answer-related nodes to contribute as a weak learning signal, we adopt the following token weights to achieve this

$$\omega_t = \begin{cases} 1, & \text{if } y_t \text{ belongs to an node in } \mathcal{P}, \\ \lambda, & \text{if } y_t \text{ belongs to an node in } \mathcal{W}, \\ 0, & \text{otherwise,} \end{cases} \quad (13)$$

where λ is a hyperparamters that controls the contribution of weak positives relative to strong positives.

3.2 Generator Training

Generator Prompting Given the predicted node set $\hat{\mathcal{V}} = R_\theta(q, \mathcal{S})$ output by the trained R_θ, we form the generator input as a *prompt* that concatenates the query q with subgraphs derived from $\hat{\mathcal{V}}$. These subgraphs contains (i) the retrieved nodes in $\hat{\mathcal{V}}$ and (ii) their associated edges $\hat{\mathcal{E}}$ that appear in $\mathcal{S}$. Concretely, we include edges for which at least one endpoint is in $\hat{\mathcal{V}}$:

$$\hat{\mathcal{E}} = \{(u, r, v) \in \mathcal{E} : u \in \hat{\mathcal{V}} \text{ or } v \in \hat{\mathcal{V}}\}. \quad (14)$$

We then serialize the retrieved nodes $\hat{\mathcal{V}}$ together with the selected edges $\hat{\mathcal{E}}$ into a *serialized graph text* and prepend the query q to form the generator prompt. Importantly, this *serialized graph text* is not required to form one whole connected subgraph. This design avoids forcing unnecessary "bridge" nodes or edges solely to maintain connectivity, which often increases irrelevant context and hurts compactness. Instead, we allow the retrieved evidence to appear as multiple disconnected components, as long as it covers the nodes in $\hat{\mathcal{V}}$. Moreover, as we mentioned in Section 3.1, $\hat{\mathcal{V}}$ may include nodes that come from R_θ's parametric knowledge and do not appear in the initial candidate subgraph $\mathcal{S}$, we retained these nodes as additional isolated nodes in the *serialized graph text*.

Fine-tuning with standard SFT. We fine-tune an llm-based generator G_ϕ using standard supervised fine-tuning (SFT). For each training instance, we first run the trained retriever R_θ to obtain the generator prompt constructed in the previous paragraph (i.e., the *serialized graph text* formed from $\hat{\mathcal{V}}$ and the associated $\hat{\mathcal{E}}$, together with the query q). We then train G_ϕ to generate the gold answer $a^\star$ conditioned on this prompt:

$$\mathcal{L}_{\text{gen}}(\phi) = -\sum_{t=1}^{T_a} \log p_\phi(a_t^\star \mid q, \hat{\mathcal{V}}, \hat{\mathcal{E}}, a_{<t}^\star), \quad (15)$$

where $a^\star = (a_1^\star, \ldots, a_{T_a}^\star)$ denotes the *token* sequence of the gold answer, and T_a is its length. This SFT step transfers the improved evidence selection from retrieval into downstream QA.

3.3 Inference and Evaluation

At inference time, we apply the full retrieve-then-generate pipeline using the retriever and generator fine-tuned on the training (and validation) splits. For each test instance $(q, \mathcal{G})$, we first construct the candidate subgraph $\mathcal{S}$ by PCST, then run the learned retriever to obtain $\hat{\mathcal{V}} = R_\theta(q, \mathcal{S})$. We materialize $(\hat{\mathcal{V}}, \hat{\mathcal{E}})$ into a textual graph context and feed it to the generator to produce the final answer $a = G_\phi(q, \hat{\mathcal{V}}, \hat{\mathcal{E}})$. We report task metrics computed from the generator outputs on the test split.

4 Experiments

4.1 Experiment Setup

Datasets and Metrics. To evaluate the effectiveness of our method on textual QA tasks, we conduct experiments on two textual graph question answering datasets, ExplaGraphs and WebQSP from the GraphQA benchmark (He et al., 2024). Specifically, (1) ExplaGraphs's primary task is to determine whether the arguments are supportive or contradictory. (2) For the knowledge graph question answering dataset WebQSP, the primary task is to answer questions that necessitate multi-hop reasoning. For evaluation, we adopt Accuracy for ExplaGraphs and Hit@1 for WebQSP. Dataset statistics are summarized in Table 5, and detailed experimental configurations are provided in Appendix A.

Baselines. We compare against two groups of baselines. (1) Inference-Only methods that include Zero-shot, Zero-CoT (Kojima et al., 2022), CoT-BAG (Wang et al., 2024), KAPING (Baek et al.,

2023), and Graph-based Inference. (2) Prompt-Tuning methods that include GraphToken (Perozzi et al., 2024), G-Retriever (He et al., 2024), and TAONA-A (Yan et al., 2025). For G-Retriever, we include one variant that fine-tunes the LLM-based generator with LoRA (G-Retriever w/ LoRA).

Experiment pipeline. We adopt the open-source LLaMA2-Chat-7B model (Touvron et al., 2023) as both the retriever and generator, and conduct all experiments on a single NVIDIA A100 GPU. We follow the same train/validation/test splits as G-Retriever and report results averaged over four random seeds $0, 1, 2, 3$. For WebQSP, since the full KG exceeds the context budget, we first construct a PCST candidate subgraph with $k_v{=}k_e{=}10$ and then apply our learnable retriever. During retriever training, we use the weighted pseudo-label objective with strong-positive weight 1 and weak-positive weight $\lambda = 0.01$. Additional hyperparameter details are provided in Appendix B.

4.2 Benchmark results

Table 1 reports the benchmark results on WebQSP (Hit@1) and ExplaGraphs (Accuracy). Overall, TALENT achieves the best performance on both datasets, improving over the strongest competitors. We draw the following three observations.

(1) TALENT achieves state-of-the-art performance. On WebQSP, TALENT attains a Hit@1 of 74.92%, ranking first among all methods and outperforming the strongest prior baselines, G-Retriever w/ LoRA and TAONA-A by 1.13% and 3.69%, respectively. On ExplaGraphs, TALENT similarly achieves the highest accuracy of 91.79%, improving over G-Retriever w/ LoRA and TAONA-A by 4.74% and 4.78%, respectively. These consistent gains across WebQSP and ExplaGraphs indicate that TALENT generalizes well under different graph QA objectives and metrics.

(2) Comparison to inference-only methods. On WebQSP, inference-only baselines (Zero-shot, Zero-CoT, CoT-BAG, KAPING, and Graph-based Inference) remain substantially behind TALENT, with the strongest baseline achieving a Hit@1 of only 52.64% compared to 74.92% for TALENT. This performance gap stems from their reliance on test-time prompting and rule-based subgraph selection, without task-specific fine-tuning of either the retriever or generator, leaving the LLM poorly adapted to textual-graph reasoning. In contrast, TALENT fine-tunes both components, enabling more reliable evidence selection and more accurate answers. Notably, Graph-based Inference performs particularly poorly, with a Hit@1 of 47.22% on WebQSP and an accuracy of 33.93% on ExplaGraphs, indicating that merely exposing graph information can even hinder reasoning when the context is noisy or misaligned. It further show that learning to select answer-relevant nodes is essential beyond simply providing the graph.

(3) Comparison to prompt-tuning methods. On WebQSP, prompt-tuning approaches form substantially stronger baselines than inference-only prompting methods. In particular, G-Retriever and its LoRA variant demonstrate that fine-tuning with a PCST candidate subgraph yields large gains. Nevertheless, TALENT further improves over G-Retriever w/ LoRA from 73.79% to 74.92%, indicating remaining headroom beyond rule-based, query-driven retrieval. Overall, TALENT's superior performance supports our central claim: training retrieval with gold signals via weighted pseudo-label supervision better balances answer-node coverage and compactness, reducing redundancy and benefiting generation. On ExplaGraphs, prompt-tuning methods again outperform inference-only baselines (e.g., GraphToken 85.08%, G-Retriever w/ LoRA 87.05%, TAONA-A 87.01%), yet TALENT remains best with an accuracy of 91.79%.

4.3 Coverage-Compactness Tradeoff

In this section, we study how TALENT's retriever achieves a better coverage-compactness tradeoff for its output node set $\hat{\mathcal{V}}$. We conduct experiments on WebQSP, with results shown in Figure 3, and make the following observations.

Compactness: similar coverage with substantially fewer nodes. We first compare the compactness of nodes retrieved by the initial PCST candidate subgraph ($k_v{=}k_e{=}10$) and TALENT, as shown in Figure 3. Under similar gold-node coverage, TALENT's retriever output $\hat{\mathcal{V}}$ contains significantly fewer nodes. Both methods achieve comparable Hit@1 and Recall on WebQSP, yet $\hat{\mathcal{V}}$ includes only about half as many nodes: on average 16.77 nodes for TALENT versus 31.75 nodes for the PCST subgraph. Meanwhile, $\hat{\mathcal{V}}$ attains notably higher precision and F1, improving precision from 9.26% to 19.98% and F1 from 12.22% to 21.66%. These results suggest that the weighted pseudo-label supervision effectively incorporates gold signals, enabling the retriever to filter redundant or weakly related nodes and produce a more compact context without sacrificing coverage.

Table 1: Benchmark results (%).

Dataset (Metrics)	ExplaGraphs (ACC)	WebQSP (Hit@1)
Zero-shot	56.50	41.06
Zero-CoT (Kojima et al., 2022)	57.04	51.30
CoT-BAG (Wang et al., 2024)	57.94	39.60
KAPING (Baek et al., 2023)	62.27	52.64
Graph-based Inference	33.93	47.22
Frozen LLM + Prompt Tuning (PT)	57.63	48.34
GraphToken (Perozzi et al., 2024)	85.08	57.05
G-Retriever	85.16	70.49
G-Retriever w/ LoRA	87.05	73.79
TAONA-A (Yan et al., 2025)	87.01	71.23
TALENT	**91.79**	**74.92**

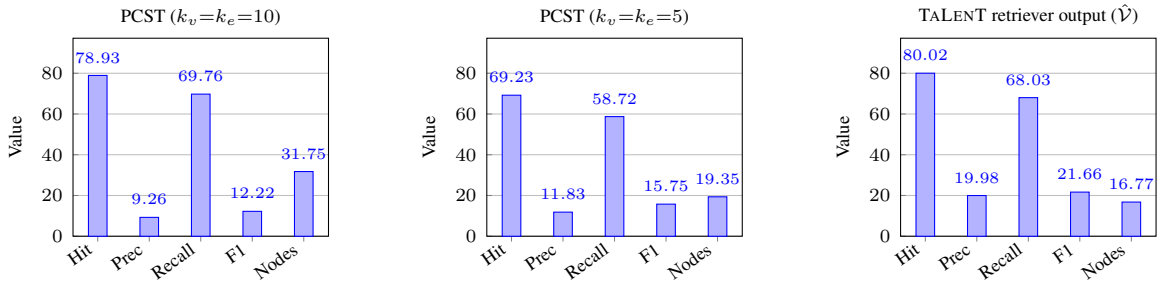

Figure 3: WebQSP test-set retrieval results (macro-averaged). We measure how well PCST subgraphs and TALENT's learned retriever output $\hat{\mathcal{V}}$ cover the gold answer set $\mathcal{V}^\star$. We compare: (1) the PCST candidate subgraph with k_v=k_e=10 (this is the retriever input), (2) a smaller PCST subgraph with k_v=k_e=5 (a size-matched reference), and (3) the nodes set $\hat{\mathcal{V}}$ selected by TALENT's learned retriever. Our retriever keeps coverage close to PCST(k_v=k_e=10) while using far fewer nodes, and it also covers more gold nodes than the size-matched PCST(k_v=k_e=5).

Table 2: Breakdown of gold answer coverage contributed by different parts of the learned retriever output on WebQSP test. All metrics are macro-averaged over examples and reported as percentages.

Output subset	**Hit@1**	**Recall**
All retrieved nodes	80.02	68.03
Nodes within PCST	75.58	64.76
Nodes out of PCST	10.14	5.21

Coverage: better gold answer coverage under comparable node number. To further compare coverage, we report results for a smaller PCST subgraph with k_v=k_e=5, whose size is comparable to that of TALENT. As shown in Figure 3, under a similar node budget, TALENT's retriever achieves substantially higher coverage. Specifically, TALENT improves Hit@1 from 69.23% to 80.02% and Recall from 58.72% to 68.03% compared to PCST. This demonstrates that, given a similar number of nodes, our gold-signal-involved weighted pseudo-label supervision enables the retriever to achieve significantly higher coverage than the rule-based, query-similarity-driven PCST algorithm.

4.4 Retrieval Beyond PCST

Fine-tuning on an LLM-based retriever may leverage parametric knowledge to retrieve nodes beyond the PCST candidate subgraph. We therefore examine whether TALENT's retriever can retrieve such beyond-PCST nodes and how they contribute to performance. Specifically, we split the retriever output $\hat{\mathcal{V}}$ into nodes inside and outside the PCST subgraph. Shown in Table 2, nodes outside PCST alone recover a non-trivial portion of gold answer nodes, with a Hit@1 of 10.14% and a Recall of 5.21%. Moreover, incorporating beyond-PCST nodes improves Hit@1 from 75.58 to 80.02% and Recall from 64.76% to 68.03%. Results indicate that the trained LLM retriever effectively complements the PCST candidate subgraph with parametric-knowledge nodes that enhance gold-node coverage.

5 Ablation Study

We conduct two ablation studies on WebQSP to illustrate (i) the effect of pseudo-label retriever training (ii) the effect of learnable retriever.

5.1 On the Effect of Pseudo-label Training

Retrieval quality. Table 3 compares TALENT's retriever against an ablated variant, which uses the same backbone LLM LLaMA2-Chat-7B model but keeps it *frozen* (no training) and directly prompts it to select helpful nodes from the PCST candi-

Table 3: Retriever ablation on WebQSP test set (macro-averaged over examples). "Ablate retriever training" replace TALENT's learnable retrieved by a frozen retriever without LoRA fine-tuning.

Retriever setting	Hit@1	Prec	Recall	F1
TALENT	80.02	19.98	68.03	21.66
Ablate retriever training	36.61	14.97	26.26	14.97

Table 4: QA ablations on WebQSP (Hit@1, %). All systems follow the same pipeline as TALENT except for the ablated component stated in each row.

System	Hit@1 (%)
TALENT w/o retriever training	68.37
TALENT w/o retriever	72.50
TALENT	74.92

date subgraph. Results show that the frozen retriever fails to recover gold nodes required for answering, with Hit@1, Precision, and Recall of 36.61%, 14.97%, and 26.26%, respectively. In contrast, TALENT's retriever explicitly prioritizes gold answer nodes during retriever fine-tuning using the pseudo label training, substantially improving coverage with a Hit@1 of 80.02%, a precision of 19.98% and a recall of 68.03%, indicating that injecting gold signals into retriever training is crucial for selecting answer-relevant nodes.

Downstream QA performance. We next examine whether the retrieval improvement translates into better end-task QA. Table 4 summarizes QA Hit@1 under three systems: (i) TALENT with a *frozen* retriever (ablate training on retriever), (ii) TALENT *without* the retriever module (directly feeding the PCST candidate subgraph to the generator), and (iii) the full TALENT system. Switching from a frozen retriever to our pseudo-label trained retriever improves QA Hit@1 from 68.37% to 74.92%. This gain is consistent with Table 3: pseudo-label training recovers substantially more gold nodes, providing the generator with more reliable evidence.

5.2 On the Effect of Learnable Retriever

We further isolate the contribution of the retriever by removing it from the pipeline and directly feeding the PCST candidate subgraph into the generator, i.e., without R_θ or $\hat{\mathcal{V}}$ selection. As shown in Table 4, removing the learnable retriever reduces QA performance from 74.92% to 72.50%, indicating that an explicit node-selection retriever provides clear benefits for downstream QA. Notably, using a frozen retriever performs even worse (Hit@1 of 68.37%) than removing the retriever entirely, suggesting that retrieval is not inherently beneficial when gold answer node coverage is low, as noisy context can mislead the generator. This result further underscores the importance of our gold-signal-based pseudo-label training, which substantially improves retrieval quality for end-task QA.

6 Related Work

Retrieval Augmented Generation. RAG grounds LLMs with external evidence to reduce hallucinations and improve faithfulness (Sun et al., 2024; Gao et al., 2023). Classic RAG systems follow a straightforward index-retrieve-generate workflow (Ma et al., 2023), while many recent variants strengthen retrieval with additional processing before and after retrieval. For example, pre-retrieval techniques such as query transformation, expansion, and rewriting are often adopted to better match the search space (Zheng et al., 2023; Gao et al., 2023), and post-retrieval reranking is commonly used to refine the retrieved evidence set (Blagojevi, 2023). More broadly, modular RAG systems incorporate heterogeneous sources and use LLMs to iteratively refine retrieval queries or evidence selection (Yu et al., 2022).

RAG over Graphs. Extensive works study how to combine graph-structured information with LLMs (Li et al., 2023b; Wang et al., 2024). Prior efforts span general graph modeling (Liu et al., 2023; Lei et al., 2023), multimodal architectures that incorporate graph-like structures (Yoon et al., 2023; Li et al., 2023a). Representative applications include fundamental graph reasoning (Zhao et al., 2023a), node prediction (Sun et al., 2023; Chen et al., 2024), graph-level prediction (Qian et al., 2023). A related line of work discusses GraphRAG / GraphQA, where the system selects graph evidence and conditions an LLM on graph context to answer questions (Logan IV et al., 2019; Luo et al., 2025).

7 Conclusion

We address two fundamental limitations in existing GraphRAG methods: *coverage*, where retrievers fail to recall required evidence nodes, and *compactness*, where excessive irrelevant information overwhelms the LLM. Both issues may stem from the absence of gold answer signals to guide evidence selection. To address this, we propose TALENT, a learnable retriever trained with a weighted pseudo-label loss that prioritizes answer-related signals to improve coverage while suppressing noise for compactness. Experimental results on two GraphQA benchmarks demonstrate that TALENT consistently improves downstream performance.

Limitations

While TALENT shows strong performance, we acknowledge several limitations. First, retriever training focuses on selecting nodes only. Edges are added by a deterministic rule based on the selected node set, instead of being directly optimized with learnable edge selection. Second, our weighted pseudo-label supervision reduces node selection to sequence generation by serializing the target nodes into a pipe-separated string in a fixed order, where strong positives precede weak positives, but we do not study whether alternative orderings, would further affect or improve learning. Future work could explore joint node-and-edge selection and order-robust training objectives.

References

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *arXiv preprint arXiv:2306.04136*.

Yejin Bang. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Daniel Bienstock, Michel X Goemans, David Simchi-Levi, and David Williamson. 1993. A note on the prize collecting traveling salesman problem. *Mathematical programming*, 59(1):413–420.

Vladimir Blagojevi. 2023. Enhancing rag pipelines in haystack: Introducing diversityranker and lostinthemiddleranker.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiang Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. 2025. Graphllm: Boosting graph reasoning ability of large language model. *IEEE Transactions on Big Data*.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and 1 others. 2024. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

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

Bin Lei, Chunhua Liao, Caiwen Ding, and 1 others. 2023. Boosting logical reasoning in large language models through a new framework: The graph of thought. *arXiv preprint arXiv:2308.08614*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Xin Li, Dongze Lian, Zhihe Lu, Jiawang Bai, Zhibo Chen, and Xinchao Wang. 2023a. Graphadapter: Tuning vision-language models with dual knowledge graph. *Advances in Neural Information Processing Systems*, 36:13448–13466.

Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. 2023b. A survey of graph meets large language model: Progress and future directions. *arXiv preprint arXiv:2311.12399*.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. 2023. One for all: Towards training one graph model for all classification tasks. *arXiv preprint arXiv:2310.00149*.

Robert L Logan IV, Nelson F Liu, Matthew E Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge-graphs for fact-aware language modeling. *arXiv preprint arXiv:1906.07241*.

Linhao Luo, Jiaxin Ju, Bo Xiong, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2025. Chatrule: Mining logical rules with large language models for knowledge graph reasoning. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 314–325. Springer.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315.

Costas Mavromatis and George Karypis. 2024. Gnn-rag: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*.

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

Chen Qian, Huayi Tang, Zhirui Yang, Hong Liang, and Yong Liu. 2023. Can large language models empower molecular property prediction? *arXiv preprint arXiv:2307.07443*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8371–8384.

Andy Sun, Tianqi Zheng, Aakash Kolekar, Rohit Patki, Hossein Khazaei, Xuan Guo, George Cai, David Liu, Ruirui Li, Yupin Huang, and 1 others. 2024. A product-aware query auto-completion framework for e-commerce search via retrieval-augmented generation method.

Shengyin Sun, Yuxiang Ren, Jiehao Chen, and Chen Ma. 2023. Large language models as topological structure enhancers for text-attributed graphs. *arXiv preprint arXiv:2311.14324*.

Dhaval Taunk, Lakshya Khanna, Siri Venkata Pavan Kumar Kandru, Vasudeva Varma, Charu Sharma, and Makarand Tapaswi. 2023. Grapeqa: Graph augmentation and pruning to enhance question-answering. In *Companion Proceedings of the ACM Web Conference 2023*, pages 1138–1144.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, and 1 others. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V Chawla, and Panpan Xu. 2024. Graph neural prompting with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19080–19088.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2024. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36.

Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering. *arXiv preprint arXiv:2308.13259*.

Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. 2024. Next-gpt: Any-to-any multimodal llm. In *Forty-first International Conference on Machine Learning*.

Yuchen Yan, Aakash Kolekar, Sahika Genc, Wenju Xu, Edward W Huang, Anirudh Srinivasan, Mukesh Jain, Qi He, and Hanghang Tong. 2025. To answer or not to answer (taona): A robust textual graph understanding and question answering approach. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 6360–6376.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*.

Minji Yoon, Jing Yu Koh, Bryan Hooi, and Ruslan Salakhutdinov. 2023. Multimodal graph learning for generative tasks. *arXiv preprint arXiv:2310.07478*.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2022. Generate rather than retrieve: Large language models are strong context generators. *arXiv preprint arXiv:2209.10063*.

Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. 2023. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*.

Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. 2023a. Graphtext: Graph reasoning in text space. *arXiv preprint arXiv:2310.01089*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023b. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H Chi, Quoc V Le, and Denny Zhou. 2023. Take a step back: Evoking reasoning via abstraction in large language models. *arXiv preprint arXiv:2310.06117*.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*.

A Additional Dataset Details

A.1 Datasets

Datasets and Metrics. To evaluate the effectiveness of our method on textual QA tasks, we conduct experiments on two textual graph question answering datasets, ExplaGraphs and WebQSP from the GraphQA benchmark (He et al., 2024). Specifically, (1) ExplaGraphs is developed for generative commonsense reasoning, with a specific focus on the construction of explanation graphs for stance prediction in debates. By providing detailed and unambiguous commonsense-augmented graphs, it enables an evaluation of whether arguments either support or refute a specific belief. ExplaGraphs's primary task is to determine whether the arguments are supportive or contradictory. Its node denotes commonsense concepts and the edge denotes Commonsense relations. In this dataset, the gold answer is a binary label (i.e., 'support' or 'counter'), which does not provide explicit supervision for the retriever. Therefore, we leverage the LLaMA2-Chat-7B model (Touvron et al., 2023) to generate pseudo gold reasoning node sets, and treat these annotated nodes as pseudo-label supervision $\mathcal{V}^\star$ for training the learnable retriever. Due to the small graph size of the graphs in ExplaGraphs, we directly use the entire graph as the candidate context for retrieval. Following GraphQA (He et al., 2024), we evaluate the end task with **Accuracy**. (2) WebQSP is a large-scale multi-hop knowledge graph QA dataset that consists of 4,737 questions. The primary task is to answer questions that necessitate multi-hop reasoning. It employs a subset of Freebase (Bollacker et al., 2008), with a specific focus on facts within 2 hops of the entities identified in the questions. The nodes denote Entities in Freebase and the edges denote Relations in Freebase. The gold answers are lists of Freebase entities, which naturally correspond to node lists and can directly serve as $\mathcal{V}^\star$ without requiring additional annotation. For metrics, we report Hit@1, counting a prediction as correct if the generated answer matches any of the answers in the gold answer list. Dataset statistics are summarized in Table 5. See Appendix A.2 for Pseudo-Gold Nodes Annotation for ExplaGraphs.

A.2 Pseudo-Gold Nodes Annotation for ExplaGraphs

ExplaGraphs provides gold supervision as a stance label (support/counter), rather than node-level evidence annotations. To make our learnable retriever pseudo label supervision applicable, we introduce a lightweight preprocessing step on the training and validation splits: we use `Llama-2-7b-chat` to annotate a set of pseudo-gold reasoning nodes for each example. Concretely, we provide the model with (i) the full textual explanation graph, (ii) the arguments, and (iii) the gold label, and prompt it to select a set of nodes from the graph that form a critical reasoning path leading to the correct stance decision. The selected nodes are used as $\mathcal{V}^\star$ in our weighted pseudo-label objective for training the retriever. This annotation is used only to train our retriever: the generator is still fine-tuned with the original task supervision (predicting support vs. counter), and at inference time the pipeline does not require access to any node annotations.

B Hyperparameter Configuration

Retriever fine-tuning. We fine-tune the retriever with LoRA under our token-weighted cross-entropy objective. Given a question and a candidate subgraph, we construct weighted pseudo labels over candidate nodes: nodes overlapping with gold answer nodes are treated as strong positives with weight 1.0, while the remaining candidate nodes are treated as weak positives with weight $\lambda = 0.01$. The training target is a single-line, pipe-separated node list, formed by concatenating strong positives and weak positives using the delimiter "|" (strong positives first). We deduplicate node names in both retriever outputs and the contexts later used for generator training.

We use LoRA with rank r=8, scaling factor α=16, and dropout 0.05. Retriever training uses 3 epochs, learning rate 5×10^{-5}, warmup ratio 0.03, per-device batch size 1 with gradient accumulation 8.

Generator fine-tuning. We fine-tune the generator with LoRA using standard supervised fine-tuning on retrieved textual graph contexts and gold targets. We use the same LoRA configuration (r=8, α=16, dropout 0.05) for generator fine-tuning.

C More Related Works

Graphs and Large Language Models A growing body of work studies how to combine graph-structured information with LLMs (Li et al., 2023b; Wang et al., 2024). Prior efforts span general graph modeling (Liu et al., 2023; Lei et al., 2023), multimodal architectures that incorporate graph-like structures (Yoon et al., 2023; Li et al., 2023a). Rep-

Table 5: Statistics of datasets.

Dataset	ExplaGraphs	WebQSP
#Graphs	2,766	4,737
Average #Nodes	5.17	1370.89
Average #Edges	4.25	4252.37
Node Attribute	Commonsense concepts	Entities in Freebase
Edge Attribute	Commonsense relations	Relations in Freebase
Task	Commonsense reasoning	KGQA
Evaluation metrics	Accuracy	Hit@1

resentative applications include fundamental graph reasoning (Zhao et al., 2023a), node prediction (Sun et al., 2023; Chen et al., 2024), graph-level prediction (Qian et al., 2023). A related line of work discusses GraphRAG / GraphQA, where the system selects graph evidence and conditions an LLM on graph context to answer questions (Logan IV et al., 2019; Luo et al., 2025).

Parameter-Efficient Fine-Tuning Parameter-efficient fine-tuning (PEFT) methods enable adapting large pretrained models with limited trainable parameters. Representative techniques include prompt tuning (Lester et al., 2021), prefix tuning (Li and Liang, 2021), and low-rank adaptation such as LoRA (Hu et al., 2022). Related adapter-style methods (e.g., LLaMA-adapter (Zhang et al., 2023)) have also supported rapid adaptation and inspired multimodal instruction-following systems such as MiniGPT-4 (Zhu et al., 2023) and NExT-Chat (Wu et al., 2024). PEFT has recently been explored in graph+LLM settings as well, including GraphLLM (Chai et al., 2025) and GraphToken (Perozzi et al., 2024) for graph reasoning, and GNP (Tian et al., 2024) for knowledge-graph QA.

D Potential Risks

The proposed TALENT improves textual GraphQA/GraphRAG by training an LLM-based retriever with gold-answer signals. While effective, it introduces several potential risks and limitations.

Dependence on candidate-subgraph quality. For large graphs (e.g., WebQSP), our pipeline first constructs a candidate subgraph via a rule-based method (PCST) before applying the learnable retriever. If this candidate subgraph fails to include sufficient answer-related evidence, downstream retrieval and generation may be constrained accordingly.

Reliance on pseudo-gold node annotations for ExplaGraphs. ExplaGraphs provides only stance labels (support/counter) and does not include node-level evidence. To enable retriever training under our weighted pseudo-label objective, we use an LLM (LLaMA2-Chat-7B) to annotate pseudo-gold reasoning node sets on the training/validation splits. This preprocessing may introduce annotation noise or bias, which can propagate into retriever behavior, although inference does not require node annotations.

Potential ungrounded evidence from parametric knowledge. Because the retriever is an instruction-following LLM, it may leverage parametric knowledge and output nodes not present in the initial candidate subgraph. We retain such nodes as isolated nodes in the serialized context; however, these nodes may not be verifiable within the provided graph evidence, potentially affecting faithfulness.

Structured-output and optimization limitations. As discussed in our paper limitations, retriever training focuses on selecting nodes (edges are added deterministically), and the weighted pseudo-label supervision reduces node selection to sequence generation with a fixed serialization order (strong positives precede weak positives). We do not study whether alternative orderings or permutation-invariant objectives could further improve learning.

Black-box constraints. Our approach assumes access to model weights (for LoRA fine-tuning) and to intermediate text/graph representations during retrieval and prompting. This may limit applicability in strictly black-box API scenarios where model adaptation or constrained structured decoding is not feasible.

E Use Or Create Scientific Artifacts

Our work builds on established public benchmarks and pre-trained language models for textual graph understanding. We evaluate on GraphQA benchmark datasets, including `ExplaGraphs` and `WebQSP`, and use a subset of Freebase as the underlying knowledge graph for WebQSP. For both retriever and generator, we use `LLaMA2-Chat-7B` as the backbone LLM, and follow prior work in first constructing a candidate subgraph using PCST.

We do not modify the original dataset content. For ExplaGraphs only, we create *pseudo-gold* node annotations on the training/validation splits to train the retriever; these annotations are not used at inference time.

E.1 Cite Creators Of Artifacts

All external artifacts are credited to their original publications and repositories. Specifically, we cite: (i) the GraphQA benchmark and baselines (e.g., G-Retriever), (ii) `ExplaGraphs` and `WebQSP`, (iii) the backbone LLM (`LLaMA2-Chat-7B`), and (iv) PCST / sentence-encoder based indexing components used for candidate subgraph construction.

E.2 Discuss The License For Artifacts

We comply with the licenses and terms of use of all artifacts used in this work.

- **ExplaGraphs.** Used under its dataset license/terms as provided by the dataset authors.
- **WebQSP.** Used for research purposes under their respective licenses/terms as distributed by the original providers.
- **Backbone LLM.** `LLaMA2-Chat-7B` is used under the corresponding Meta Llama 2 license and acceptable use policy.

E.3 Artifact Use Consistent With Intended Use

We confirm that our use of datasets and pre-trained models is consistent with their intended research purposes. ExplaGraphs is designed for commonsense reasoning with explanation graphs (stance prediction), and WebQSP is designed for multi-hop KGQA; both align directly with our goal of evaluating retrieval-augmented reasoning over textual graphs. The backbone LLM is used for instruction-following retrieval and generation, consistent with common research usage.

E.4 Data Contains Personally Identifying Info Or Offensive Content

The datasets we use are standard public benchmarks. WebQSP is derived from Freebase and may contain entities corresponding to real people (as part of a public knowledge graph). We do not add new personally identifying information beyond what exists in the original datasets, and we use the data solely for research evaluation. ExplaGraphs consists of debate-style claims and explanations; while not intended to contain Personally Identifiable Information, it may include sensitive topics typical of argumentation datasets.

E.5 Documentation Of Artifacts

We provide dataset descriptions and usage details in Appendix A, including task definitions, graph semantics (node/edge attributes), and evaluation metrics. We also provide detailed hyperparameter configurations for retriever and generator fine-tuning (including LoRA settings and optimization hyperparameters) in Appendix B.

F Statistics For Data

Dataset statistics are summarized below (see Table 5 and Appendix A for details).

F.1 ExplaGraphs

- #Graphs: 2,766.
- Average #Nodes: 5.17; Average #Edges: 4.25.
- Node attributes: commonsense concepts; edge attributes: commonsense relations.
- Task: commonsense reasoning for stance prediction; metric: Accuracy.

F.2 WebQSP

- #Questions/Graphs: 4,737.
- Average #Nodes: 1370.89; Average #Edges: 4252.37.
- Node attributes: entities in Freebase; edge attributes: relations in Freebase.
- Task: multi-hop KGQA; metric: Hit@1.

G Computational Experiments

All computational experiments are reproducible given the specifications in the main paper and Appendix B.

G.1 Model Size And Budget

We use `LLaMA2-Chat-7B` as the backbone for both retriever and generator. All experiments are run on a single NVIDIA A100 GPU, and we repeat experiments over four random seeds, reporting the average.

G.2 Experimental Setup And Hyper-params

We follow the GraphQA benchmark protocol and use PCST to build a candidate subgraph for large graphs (WebQSP), then apply TALENT on top. Key hyperparameters include:

- PCST candidate-subgraph settings: top-k_v nodes and top-k_e edges for prize assignment (e.g., $k_v = k_e = 10$ for WebQSP).
- Weighted pseudo-label supervision: strong-positive weight 1.0 and weak-positive weight λ (e.g., $\lambda = 0.01$).
- Structured retriever targets: a single-line pipe-separated node list; strong positives precede weak positives.
- LoRA configurations and optimizer schedules for retriever/generator fine-tuning (see Appendix B).

G.3 Descriptive Statistics

We report downstream task performance on the test split: Accuracy for ExplaGraphs and Hit@1 for WebQSP, following the benchmark definitions.

G.4 Parameters For Packages

All implementation and dependency details (libraries and versions) will be documented in the released codebase and configuration files.

H Ai Assistants In Research Or Writing

We use `Llama-2-7b-chat` as a preprocessing tool to produce pseudo-gold reasoning node sets for ExplaGraphs training/validation, enabling retriever supervision. This annotation is used only for retriever training; the generator is fine-tuned with original task supervision, and inference does not require node annotations.

In addition, AI assistant tools are used for language editing (spelling/grammar/clarity) of the manuscript. They are not used to generate experimental results or to replace human scientific judgment.