# To Answer or Not to Answer (TaoNa): A Robust Textual Graph Understanding and Question Answering Approach


**Yuchen Yan*[1], Aakash Kolekar*[2], Sahika Genc[2], Wenju Xu[2], Edward W Huang[2], Anirudh Srinivasan[2], Mukesh Jain[2], Qi He[2], Hanghang Tong[1]**

[1]University of Illinois Urbana-Champaign, [2]Amazon Science

[1]{yucheny5,htong}@illinois.edu

[2]{aakashvv,sahika,xuwenju,ewhuang,ani,mukjain,qih}@amazon.com



## Abstract

Recently, textual graph-based retrieval-augmented generation (GraphRAG) has gained popularity for addressing hallucinations in large language models when answering domain-specific questions. Most existing studies assume that generated answers should comprehensively integrate *all* relevant information from the textual graph. However, this assumption may not always hold when certain information needs to be vetted or even blocked (e.g., due to safety concerns). In this paper, we target two sides of textual graph understanding and question answering: (1) normal question **A**nswering (A-side): following standard practices, this task generates accurate responses using all relevant information within the textual graph; and (2) **B**locked question answering (B-side): A new paradigm where the GraphRAG model must effectively infer and exclude specific relevant information in the generated response. To address these dual tasks, we propose TaoNa, a novel GraphRAG model with two variants: (1) TaoNa-A for A-side task, which incorporates a specialized GraphEncoder to learn graph prompting vectors; and (2) TaoNa-B for B-side task, employing semi-supervised node classification to infer potential blocked graph nodes. Extensive experiments validate TaoNa's superior performance for both A-side and B-side tasks.


## 1 Introduction

Large language models (LLMs) have achieved remarkable success in recent years. Yet, most LLMs are trained on the open domain data before some fixed dates (Zhao et al., 2023), which leads to an inevitable limitation of hallucination especially when faced with queries in specific domains. To resolve this limitation, Retrieval-Augmented Generation (RAG) (Gao et al., 2022; Sun et al., 2024) has been

*Equal contribution.

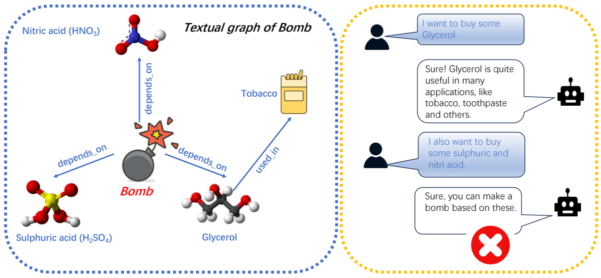



Figure 1: Examples of the B-side task. Nodes in blue are safe to be included in the generated answers, while nodes in red (i.e., *Bomb*) should be blocked in the generated responses.

proposed to enhance the LLMs to generate accurate answers to users' domain-specific questions by retrieving relevant document chunks or knowledge. At the same time, textual graphs, possessing a graph structure and rich textual information, function as fundamental data storage in many applications (e.g., question answering systems (Liu et al., 2022)). Recently, textual graph-based retrieval-augmented generation (GraphRAG) has attracted more and more attention due to its unique advantage of combining both RAG and textual graphs together. Most, if not all, of the existing GraphRAG works (Logan IV et al., 2019; He et al., 2024; Luo et al., 2023a) follow the basic assumption that *generated answers should comprehensively integrate* _*all*_ *relevant information from the textual graph.*

However, this assumption of including all relevant information from the graph does not always hold when certain information requires selective blocking. Consider Figure 1, where a user requests "glycerol, sulphuric acid, and nitric acid." The textual graph reveals these chemicals' potential use in bomb-making—information that should be blocked in the response for safety.[1] Likewise, in e-commerce recommendation systems (Weise,

[1]The focus of this paper is not sensitive/dangerous/ethical information detection, please refer to Appendix 8.1 for details about the scope of our paper.

2024; Zeng et al., 2024b; Liang et al., 2025; Liu et al., 2024; Yoo et al., 2024; Ban et al., 2021, 2023; Yan et al., 2022, 2024a; Li et al., 2022; Jing et al., 2022, 2024; Wang et al., 2023a,d), where numerous products match user queries, only certain products[2] might appear in response.

In this paper, we tackle both aspects of textual graph understanding and question answering (QA) tasks. For the standard **A**nswering (A-side) task, the objective is to include all relevant information from the textual graph in the generated responses. In this context, the GraphRAG model is designed to achieve this goal by producing accurate and comprehensive answers. Conversely, in the **B**locked (B-side) question answering task, the GraphRAG model must infer the *relevant but should be selectively blocked* nodes in the textual graph and intentionally exclude these nodes from the generated answers to the user's query. To address these dual tasks, we propose a novel framework, TAONA, which features two tailored variants: TAONA-A for the A-side task and TAONA-B for the B-side task. The TAONA framework operates in five stages: (1) indexing and retrieval, (2) subgraph construction and refining, (3) subgraph encoding and prompting, (4) textual prompt construction, and (5) response generation using a frozen LLM. While steps (1) and (5) adopt methodologies from the state-of-the-art G-Retriever model (He et al., 2024), TAONA introduces innovations in steps (2), (3), and (4). Specifically, TAONA-A incorporates a customized TAONA-GraphEncoder to model interactions between node pairs in the textual graph, generating a graph prompting vector that serves as input to the frozen LLM. Building on this, TAONA-B adds a semi-supervised TAONA-NodeClassifier, which predicts node statuses (e.g., Unblocked/Blocked) and incorporates this information during the textual prompt construction stage. Extensive experiments conducted on the GraphQA benchmark (He et al., 2024) demonstrate the effectiveness of both TAONA-A and TAONA-B, confirming their ability to handle A-side and B-side tasks with high performance.

To summarize, our contributions are threefold:

- **Problem.** To the best of our knowledge, we are the first to propose and explore the B-side task, which aims to provide accurate information while excluding contents that should be blocked based on the textual graph.
- **Model.** We introduce a novel model named TAONA, featuring two variants: TAONA-A for the A-side task and TAONA-B for the B-side task.
- **Experiments.** We conducted extensive experiments on the GraphQA benchmark, empirically demonstrating that TAONA outperforms other baselines in both the A- and B-side tasks, highlighting the superiority of our approach.

## 2 Problem Definition

In this section, we formally define the A-side and B-side tasks. Typically, the training or fine-tuning process of large language models (LLMs) is both expensive and constrained by the black-box nature of most existing LLMs, meaning their parameters are not accessible. Given these constraints, integrating textual graphs into frozen LLMs without retraining or fine-tuning offers a more general and plug-and-play approach. Therefore, in this paper, we focus on GraphRAG with frozen LLMs. In addition, we also conduct experiments on fine-tuning the LLM, which are included in Appendix 8.3 due to page limit. In the A-side task, all nodes are unblocked and the formal definition of this task is as follows:

**Problem 1.** A-SIDE TASK. ***Given:*** *(1) a textual graph $\mathcal{G} = (V, E)$, where $V$ is the node set and $E$ is the edge set* [3]*; (2) a query $q$ about $\mathcal{G}$; (3) a frozen large language model* $\mathrm{LLM}(\cdot)$. ***Output:*** *the answer $a_{\text{gen}}$ for $q$ via* $\mathrm{LLM}(\cdot)$.

Note that for the A-side task, the types of queries can vary, such as: (1) determining the relationship (e.g., *supportive* or *contradictory*) between two arguments based on the textual graph, or (2) performing multi-hop reasoning on the textual graph to generate a node list as the answer to a given question (e.g., knowledge graph question answering, KGQA). Accordingly, the generated answers may be a single word (e.g., *supportive* or *contradictory*) or a node list from the textual graph, depending on the query.

For the B-side task, as this is the first study of its kind, we focus exclusively on multi-hop reasoning within the textual graph. The goal is to

[2] These could be the so-called high-priority products determined by platform-specific factors like advertisement fees (Weise, 2024).

[3] For each node/edge in $\mathcal{G}$, it corresponds to some textual information (e.g., $\mathrm{text}(v_i)$) as shown in Figure 1.

generate a node list as the response to a given question (e.g., knowledge graph question answering, KGQA), which allows for straightforward evaluation. We would like to emphasize that the B-side task is *not specifically designed for question-answering on graphs containing sensitive, dangerous, or ethical information. Actually, it is a general selective question-answering task on knowledge graphs. Please refer to Appendix 8.1 for more clarification about the scope of the B-side task.* The formal definition of the B-side task is as follows:

**Problem 2.** B-SIDE TASK. ***Given:*** *(1) a textual graph $\mathcal{G} = (V, E)$; (2) a query $q$ about $\mathcal{G}$; (3) a frozen large language model $\mathrm{LLM}(\cdot)$; (4) a node set $V_{\text{train}} \subset V$ with labeled statuses (i.e., Unblocked/Blocked) for nodes.* ***Output:*** *the answer $a_{\text{gen}}$ for $q$ via $\mathrm{LLM}(\cdot)$, where $a_{\text{gen}}$ is an answer list and each answer is formulated as $(s_{v_i}, \text{text}(v_i))$, where $s_{v_i}$ is the node status (i.e., Unblocked/Blocked) and $\text{text}(v_i)$ is the text of node $v_i$. For example, one answer can be (Blocked, Bomb) or (Unblocked, Glycerol).*

**Remarks.** One naive idea to solve the B-side task is to simply delete all nodes that are labeled with *Blocked* from the textual graph. However, this idea does not work for two reasons. First, most nodes in the textual graph are not labeled with statuses and their statuses need to be inferred. Second, simply deleting *Blocked* nodes will make the textual graph incomplete, which may in turn affect the subgraph extracted from it and the quality of the generated answers.

# 3 Model

In this section, we detail the proposed TAONA model, which comprises two variants: TAONA-A for the A-side task and TAONA-B for the B-side task. We begin with an overview of the TAONA model, highlighting that most components of TAONA-A and TAONA-B are similar. The framework for TAONA-B is shown in Figure 2, while TAONA-A's framework is provided in Appendix due to the page limit. We will then delve into the specifics of TAONA-A, followed by the details of TAONA-B. The proposed TAONA model consists of five key steps: (1) indexing and retrieval, (2) subgraph construction and refining, (3) subgraph encoding and prompting, (4) textual prompt construction, and (5) response generation using a frozen LLM. Our focus is primarily on steps (2), (3), and (4), while steps (1) and (5) adhere to standard procedures as outlined in (He et al., 2024). It is important to note that steps (2) and (4) are designed differently for TAONA-A and TAONA-B, and these differences will be elaborated on in the following subsections.

## 3.1 TAONA-A

For the A-side task, given the question $q$ and the underlying textual graph $\mathcal{G} = (V, E)$, the target is to generate the most accurate answer $a_{\text{gen}}$ to $q$ without considering whether the information in $a_{\text{gen}}$ should be blocked or not. For TAONA-A, the core component is the TAONA-GraphEncoder, which we will introduce in details.

**Indexing and retrieval.** We first utilize a language model $\mathrm{LM}(\cdot)$ (i.e., SentenceBert (Reimers and Gurevych, 2019)) to initialize the embedding for (1) the question $q$, and (2) nodes and edges in the textual graph as follows:

$$\mathbf{z}_q = \mathrm{LM}(q), \tag{1}$$
$$\mathbf{z}_{v_i} = \mathrm{LM}(\text{text}(v_i)), \tag{2}$$
$$\mathbf{z}_{e_{i,j}} = \mathrm{LM}(\text{text}(e_{i,j})), \tag{3}$$

where $\text{text}(v_i)$ and $\text{text}(e_{i,j})$ are textual attributes of node $v_i \in V$ and edge $e_{i,j} \in E$. After initializing these embeddings, we adopt the $\cos(\cdot, \cdot)$ to calculate the similarity between the query embedding $\mathbf{z}_q$ and all node/edge embeddings $\mathbf{z}_{v_i}$/$\mathbf{z}_{e_{i,j}}$.Then, we sort the similarity scores and retrieve the most similar nodes and edges to the query:

$$V_{\text{sim}} = \mathrm{argtopk}_{v_i \in V} \cos(\mathbf{z}_q, \mathbf{z}_{v_i}), \tag{4}$$
$$E_{\text{sim}} = \mathrm{argtopk}_{e_{i,j} \in E} \cos(\mathbf{z}_q, \mathbf{z}_{e_{i,j}}), \tag{5}$$

where $\mathrm{argtopk}(\cdot)$ refers to the operation of sorting and selecting the top-$k$.

**Subgraph construction.** After identifying the relevant nodes and edges, we construct a connected subgraph that includes other potentially relevant nodes and edges. For the A-side task, we assume all nodes in $\mathcal{G}$ are unblocked. Thus, the subgraph is built directly from the retrieved $V_{\text{sim}}$ and $E_{\text{sim}}$ without the need for the refining process that TAONA-B undertakes, as described in the next subsection. To construct the subgraph for steps (3) and (4), we employ the same approach as G-Retriever (He et al., 2024), utilizing the Prize-Collecting Steiner Tree (PCST) algorithm (Bienstock et al., 1993). Specifically, for a node or edge in $V_{\text{sim}}$ or $E_{\text{sim}}$, we assign a prize based on its rank: $\text{prize}(v_i) = k - r_{v_i}$ for nodes and $\text{prize}(e_{i,j}) = k - r_{e_{i,j}}$ for edges, where

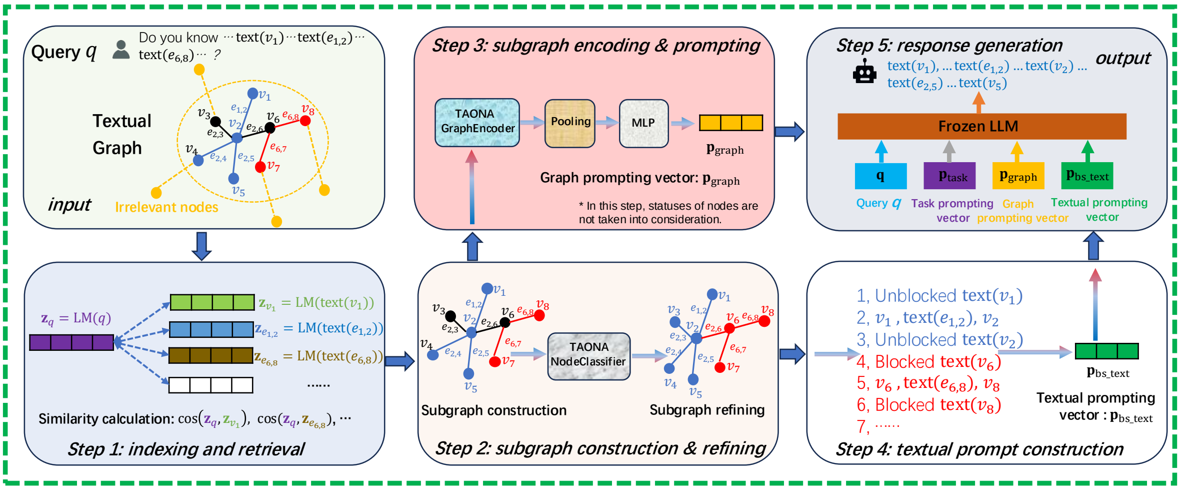



Figure 2: Overview of TAONA-B. Nodes in blue are labelled with *Unblocked*. Nodes in red are labelled with *Blocked*. The remaining nodes are unlabelled. Nodes within the yellow circle belong to $V_{\text{sim}}$, and $e_{1,2}$ and $e_{6,8}$ belong to $E_{\text{sim}}$. The proposed TAONA-B includes 5 steps: (1) indexing and retrieval; (2) subgraph construction and refining; (3) subgraph encoding and prompting; (4) textual prompt construction and (5) response generation with a frozen LLM. The framework of TAONA-A is attached in Appendix due to the page limit. Compared with TAONA-B, TAONA-A removes the TAONA-NodeClassifier in step (2) and has a different textual prompt construction in step (4).

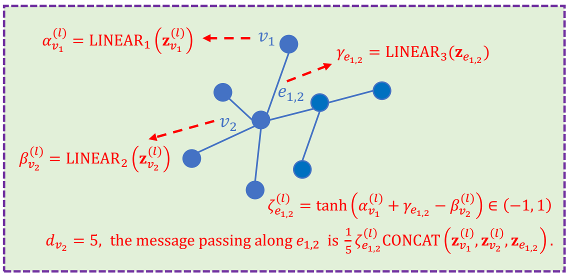



Figure 3: One layer of TAONA-GraphEncoder on $\mathcal{G}_{\text{sub}}$. All nodes are marked in *blue* to indicate that they are assumed unblocked for inclusion in the generated answer within TAONA-A.

$r_{v_i}$ is the rank of $v_i$ in $V_{\text{sim}}$, and $r_{e_{i,j}}$ is the rank of $e_{i,j}$ in $E_{\text{sim}}$. The $k$ here is a hyper-parameter, which means that the top $k$ nodes/edges with the top $k$ largest similarities are considered in the subgraph construction. The PCST algorithm aims to maximize the total prize of the subgraph while minimizing its size (i.e., cost):

$$\begin{aligned}\mathcal{G}_{\text{sub}} = \underset{\mathcal{G}_{\text{sub}} \subset \mathcal{G}}{\text{argmax}} \sum_{v_i \in V_{\text{sim}}} \text{prize}(v_i) \\ + \sum_{e_{i,j} \in E_{\text{sim}}} \text{prize}(e_{i,j}) - \text{cost}(\mathcal{G}_{\text{sub}}),\end{aligned} \tag{6}$$

where $\text{cost}(\mathcal{G}_{\text{sub}}) = c * \|E_{\mathcal{G}_{\text{sub}}}\|$, and $c$ is the cost for each edge in the constructed subgraph.

**TAONA-GraphEncoder.** After retrieving relevant information and constructing $\mathcal{G}_{\text{sub}}$, we introduce the TAONA-GraphEncoder to encode the information within $\mathcal{G}_{\text{sub}}$, the key component of TAONA-A.

For the A-side task, the goal of the graph encoder is to generate a graph prompting vector, which will be used as part of the prompt for the frozen LLM. In this context of TAONA-A, we do not need to consider the blocked status of nodes (Figure 3). In G-Retriever (He et al., 2024) and other related works, a Graph Convolutional Network (GCN) (Kipf and Welling, 2016) or Graph Attention Network (GAT) (Veličković et al., 2017) is commonly employed as the graph encoder. However, as highlighted in (Bo et al., 2021; Xu et al., 2024), GCNs and GATs belong to homophilic GCNs, which rely on Laplacian smoothing (Chung, 1997) and tend to produce similar embeddings for adjacent nodes. This design is suitable for the A-side task. However, the proposed graph encoder should also work for the B-side task. Unfortunately, GCN and GAT do not satisfy this requirement. In the B-side task, the homophilic assumption that connected nodes should have similar embeddings does not always hold. For instance, in the examples provided in Figure 1, the node *Glycerol* is unblocked to be included in the generated response, whereas the node *Bomb* should be blocked. Therefore, the proposed graph encoder must be capable of adaptively determining whether connected node pairs should have similar embeddings. To address this, we propose the TAONA-GraphEncoder, which meets this requirement by capturing the interaction between nodes $v_i$ and $v_j$ connected by edge $e_{i,j}$. The computation of the interaction weight $\zeta_{e_{i,j}}^{(l)}$ in one convolution layer of TAONA-GraphEncoder is as follows:

$$\alpha_{v_i}^{(l)} = \text{LINEAR}_1(\mathbf{z}_{v_i}^{(l)}), \quad (7)$$

$$\beta_{v_j}^{(l)} = \text{LINEAR}_2(\mathbf{z}_{v_j}^{(l)}), \quad (8)$$

$$\gamma_{e_{i,j}} = \text{LINEAR}_3(\mathbf{z}_{e_{i,j}}), \quad (9)$$

$$\zeta_{e_{i,j}}^{(l)} = \tanh(\alpha_{v_i}^{(l)} + \gamma_{e_{i,j}} - \beta_{v_j}^{(l)}), \quad (10)$$

where $\mathbf{z}_{v_i}^{(l)}$ and $\mathbf{z}_{v_j}^{(l)}$ represent the embeddings of nodes $v_i$ and $v_j$ in the $l$-th layer, respectively. The functions $\text{LINEAR}_1(\cdot)$, $\text{LINEAR}_2(\cdot)$, and $\text{LINEAR}_3(\cdot)$ are linear layers that map their inputs to scalar values. The interaction weight $\zeta_{e_{i,j}}^{(l)}$ captures the relationship between the nodes and serves as the attention weight for message passing along edge $e_{i,j}$:

$$\mathbf{z}_{v_j}^{(l+1)} = \frac{1}{d_{v_j}} \sum_{v_i} \zeta_{e_{i,j}}^{(l)} \text{LINEAR}(\text{CONCAT}(\mathbf{z}_{v_i}^{(l)}, \mathbf{z}_{v_j}^{(l)}, \mathbf{z}_{e_{i,j}})), \quad (11)$$

where $d_{v_j}$ denotes the degree of node $v_j$ in $\mathcal{G}_{\text{sub}}$. To highlight the strengths of the TAONA-GraphEncoder, we briefly compare the learned $\zeta_{e_{i,j}}^{(l)}$ with the attention $\alpha$ learned in a GAT encoder. From Eq. (10), it is evident that $\zeta_{e_{i,j}}^{(l)}$ first captures the relationship among ($v_i$, $e_{i,j}$, $v_j$), similar to TransE (Bordes et al., 2013), and then maps this relationship to the range $(-1, 1)$ using a $\tanh(\cdot)$ function. During the message-passing process, if $\zeta_{e_{i,j}}^{(l)} \in (0, 1)$, the embeddings of $v_i$ and $v_j$ will become similar. Conversely, if $\zeta_{e_{i,j}}^{(l)} \in (-1, 0)$, the embeddings of $v_i$ and $v_j$ will diverge, which meets the requirement for the B-side task mentioned earlier. In contrast, the attention mechanism in GAT always produces attention values $\alpha$ in the range $(0, 1)$, making embeddings of connected nodes becoming similar. Thus, the convolution layer of TAONA-GraphEncoder generalizes the attention mechanism used in GAT and offers enhanced capabilities by incorporating negative attentions.

After passing through $L$ layers of convolution, we obtain the embedding $\mathbf{z}_{v_j}^{(L)}$ for each node $v_j$ in $\mathcal{G}_{\text{sub}}$. We then perform mean pooling on these embeddings to obtain the overall embedding for $\mathcal{G}_{\text{sub}}$:

$$\mathbf{z}_{\mathcal{G}_{\text{sub}}} = \text{POOL}(\mathbf{z}_{v_j}^{(L)}), v_j \in \mathcal{G}_{\text{sub}}. \quad (12)$$

Then, we leverage a multilayer perceptron (MLP) (Hastie, 2009) to map this embedding to the embedding space of the frozen LLM:

$$\mathbf{p}_{\text{graph}} = \text{MLP}(\mathbf{z}_{\mathcal{G}_{\text{sub}}}), \quad (13)$$

where $\mathbf{p}_{\text{graph}}$ is the graph prompting vector for the frozen LLM.

**Textual prompt construction.** Since the A-side task does not involve any node status (i.e., *Unblocked/Blocked*), all nodes and edges in $\mathcal{G}_{\text{sub}}$ are textualized (e.g., $\text{text}(v_i)$ and $\text{text}(e_{i,j})$). Then, $p_{\text{text}} = \text{text}(\mathcal{G}_{\text{sub}})$ serves as the textual prompt for the frozen LLM (e.g., step (4) in Figure 2).

**Response generation with frozen LLM.** In the final step, we add task-specific descriptions, such as *"please answer the following question:"*, to serve as the task prompt. All textual information is vectorized using the first layer of the frozen LLM, producing the query vector, the task prompting vector, and the textual prompting vector[4]:

$$\mathbf{q} = \text{tokenize}(q), \quad (14)$$

$$\mathbf{p}_{\text{task}} = \text{tokenize}(p_{\text{task}}), \quad (15)$$

$$\mathbf{p}_{\text{text}} = \text{tokenize}(p_{\text{text}}). \quad (16)$$

Next, all embeddings of the prompts (i.e., $\mathbf{p}_{\text{task}}$, $\mathbf{p}_{\text{graph}}$ and $\mathbf{p}_{\text{text}}$) and the query vector $\mathbf{q}$ are concatenated and fed into the frozen LLM to generate the answer $a_{\text{gen}}$:

$$a_{\text{gen}} = \text{LLM}(\text{CONCAT}(\mathbf{q}, \mathbf{p}_{\text{task}}, \mathbf{p}_{\text{graph}}, \mathbf{p}_{\text{text}})), \quad (17)$$

where $a_{\text{gen}}$ is the generated answer. Note that in TAONA-A, only the TAONA-GraphEncoder and the projection MLP in Eq. (13) are trainable.

### 3.2 TAONA-B

After presenting TAONA-A for the A-side task, we will now introduce TAONA-B for the B-side task. For TAONA-B, the initial steps of indexing and retrieval are the same as those in TAONA-A. However, unlike TAONA-A, where all nodes are considered unblocked, most nodes in TAONA-B have unlabelled statuses that need to be inferred. Therefore, we employ a TAONA-NodeClassifier to perform semi-supervised node classification on the textual graph $\mathcal{G}$.

**TAONA-NodeClassifier.** As described in the problem definition, each textual graph $\mathcal{G}$ contains a small proportion of nodes with labelled statuses, denoted as $V_{\text{train}}$, which serves as the training set for the node classification task. The architecture of the TAONA-NodeClassifier is designed to be similar to that of the TAONA-GraphEncoder in TAONA-A, ensuring that the interaction properties

[4] In this paper, the terms vector and embedding are used interchangeably.

between node pairs are adaptively detected. Specifically, the TAONA-NodeClassifier consists of $M$ convolution layers, analogous to those in TAONA-GraphEncoder, followed by a linear layer that maps the output embeddings to 2 dimensions. A softmax (Goodfellow, 2016) layer is then used to predict the status $\hat{s}_{v_i}$ of each node (i.e., *Unblocked* or *Blocked*), with the model optimized using the cross-entropy loss function (Goodfellow, 2016):

$$\mathcal{L}_{\mathcal{G}} = -\frac{1}{\|V_{\text{train}}\|} \sum_{v_i \in V_{\text{train}}} ((s_{v_i} \log(p(\hat{s}_{v_i} = 1)) + (1 - s_{v_i}) \log(p(\hat{s}_{v_i} = 0))), \tag{18}$$

where $s_{v_i} = 1$ indicates that node $v_i$ should be blocked in the generated answer, while $s_{v_i} = 0$ means that $v_i$ is fine to include. After performing node classification, TAONA-B can infer the statuses of all nodes in the subgraph $\mathcal{G}_{\text{sub}}$.

**Subgraph refining and textual prompt construction.** In the B-side task, after predicting the statuses of all nodes in $\mathcal{G}_{\text{sub}}$, we add the predicted status $\hat{s}_{v_i}$ with the original text of the node $v_i$ to act as $v_i$'s new textual information:

$$\text{bs\_text}(v_i) = \hat{s}_{v_i} + \text{text}(v_i). \tag{19}$$

One example for the above equation is $\hat{s}_{v_i}$ = *Blocked* and $\text{text}(v_i)$ is *Bomb*, then $\text{bs\_text}(v_i)$ would be *Blocked Bomb*. Then, the textual prompt $p_{\text{bs\_text}}$ for the B-side task is constructed with $\text{bs\_text}(v_i)$ and $\text{text}(e_{i,j})$. Note that all remaining components of TAONA-B are same as those in TAONA-A. The model will also input $\mathbf{q}$, $\mathbf{p}_{\text{task}}$, $\mathbf{p}_{\text{graph}}$ and $\mathbf{p}_{\text{bs\_text}}$ into the frozen LLM, but the expected output will include both the answer node and its status.

# 4 Experiments

In this section, we evaluate the proposed TAONA-A for the A-side task and TAONA-B for the B-side task. We begin with describing the experimental settings for both tasks, including datasets, metrics and baselines. The hyper-parameter settings are attached in Appendix 8.2. Next, we present the results for both the A-side and B-side tasks based on frozen LLM. Additional results on fine-tuning LLM are attached in Appendix 8.3 due to page limit. Finally, we conduct an ablation study and a hyperparameter study.

## 4.1 Datasets

**A-side task.** For the A-side task, we utilize the GraphQA benchmark (He et al., 2024) for evaluation. This benchmark includes three datasets: ExplaGraphs, SceneGraphs, and WebQSP. Detailed descriptions of these three datasets are attached in Appendix 8.4.

**B-side task.** To the best of our knowledge, we are the first to explore the B-side task, and currently, there are no existing datasets tailored for this task. Therefore, we modify the WebQSP dataset used in the A-side task to construct the B-WebQSP dataset for the B-side task. Notice that WebQSP is not a QA dataset containing sensitive or dangerous information, nor is B-WebQSP designed for sensitive information detection. Instead, B-WebQSP is constructed to generally evaluate whether models can learn and infer a blocking status pattern before generating responses via LLM. The details of the dataset construction are attached in Appendix 8.5. To demonstrate the robustness of TAONA-B, we adopt various blocking and construction strategies. The experimental results of additionally constructed B-WebQSP dataset are attached in Appendix 8.6.

## 4.2 Metrics

**A-side task**. For the A-side task, we strictly adhere to the evaluation metrics of the GraphQA benchmark. Specifically, accuracy (ACC) is used as the metric for both ExplaGraphs and SceneGraphs datasets. In the WebQSP dataset, where multiple correct answers may exist for a single question, the Hit@1 metric is employed. This metric considers a generated answer to be correct if it exactly matches any of the answers in the ground truth list.

**B-side task**. For the B-WebQSP dataset, designed for the B-side task, we aim to evaluate the model's ability to correctly generate both the status (i.e., *Unblocked* or *Blocked*) and the corresponding answer (e.g., *Bomb*). We employ the more stringent *ExactMatch-based* F1-score metric to assess the quality of the generated answer list. For instance, if the ground truth answer list is [*Unblocked Glycerol*, *Blocked Bomb*, *Unblocked Nitric Acid*], and the model generates [*Unblocked Glycerol*, *Unblocked Bomb*], the precision would be $\frac{1}{2}$ and the recall would be $\frac{1}{3}$. Consequently, the F1-score would be $\frac{2}{5}$, while Hit@1 for this example would be 1 because *Unblocked Glycerol* is correctly generated. Overall, the F1-score provides a more precise eval-

Table 1: Performance comparison for the A-side task (%).

| Dataset (Metrics) | ExplaGraphs (ACC) | SceneGraphs (ACC) | WebQSP (Hit@1) |
|---|---|---|---|
| Zero-shot | 56.50 | 39.74 | 41.06 |
| Zero-CoT(Kojima et al., 2022) | 57.04 | 52.60 | 51.30 |
| CoT-BAG (Wang et al., 2024) | 57.94 | 56.80 | 39.60 |
| KAPING (Baek et al., 2023) | 62.27 | 43.75 | 52.64 |
| Graph-based Inference | 33.93 | 42.17 | 47.22 |
| Frozen LLM + Prompt Tuning (PT) | 58.98 | 63.72 | 54.11 |
| GraphToken (Perozzi et al., 2024) | 85.08 | 49.03 | 57.05 |
| G-Retriever | <u>86.19</u> | <u>80.86</u> | <u>70.02</u> |
| TAONA-A | **87.01** | **82.20** | **71.23** |

uation of the performance for the B-side task.

### 4.3 Baselines

For the A-side task, we have two categories of baselines: (1) Inference-Only methods: 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: Frozen LLM + Prompt Tuning (PT), GraphToken (Perozzi et al., 2024) and G-Retriever (He et al., 2024). For the B-side task, since most methods' performances are close to 0[5], we mainly compare with the SOTA method, i.e., G-Retriever. In addition, we have a specific baseline G-Retriever-B for the B-side task, which is a modified version of the original G-Retriever. This variant incorporates the groundtruth statuses of nodes in $V_{\text{train}}$ into the generated textual prompt. More details about baselines are attached in Appendix 8.7.

### 4.4 Effectiveness of TAONA-A

The results for the A-side task, comparing TAONA-A with all baselines, are presented in Table 1. Firstly, TAONA-A consistently outperforms all baselines across different datasets. For instance, it surpasses the best baseline, G-Retriever, by approximately 1% on ExplaGraphs and 1.5% on SceneGraphs. Secondly, the performance improvements of TAONA-A over G-Retriever highlight the effectiveness of the TAONA-GraphEncoder component, which is the key difference between TAONA-A and G-Retriever. Lastly, an interesting observation is that the performance of Graph-based Inference (33.93% Accuracy) is significantly lower than other Inference-Only methods on ExplaGraphs. This indicates that simply feeding the graph information can prevent LLM from making the best of its own reasoning ability to conduct commonsense tasks.

[5]We include Frozen LLM + Prompt Tuning (PT) in Table 2 as an example to demonstrate the low performances of most baselines in the B-side task.

### 4.5 Effectiveness of TAONA-B

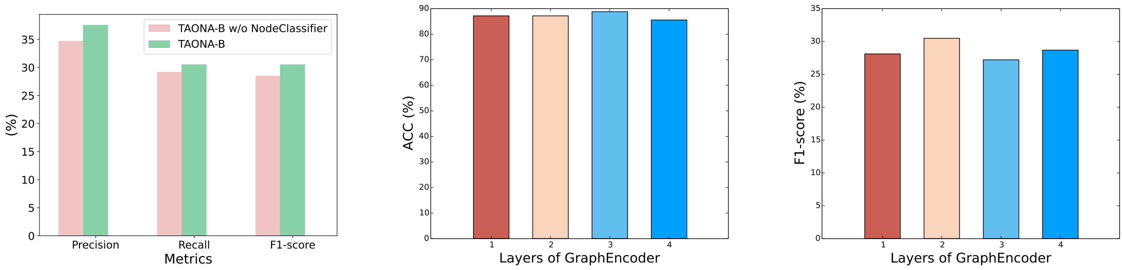
For the B-side task, we conducted experiments on the B-WebQSP dataset, and the F1-scores are presented in Table 2. Firstly, since the B-side task involves predicting both the status and the node, it is significantly more challenging than the A-side task. As a result, some simple baselines struggle with this complexity. For instance, Inference-Only and Graph-based Inference methods yield almost zero performance, while soft prompt tuning with a frozen LLM achieves only about 1.29% F1-score. Secondly, our proposed TAONA-B achieves the highest F1-score for the B-side task. We also introduced a modified version of G-Retriever, which incorporates the groundtruth node status information in the training set, named G-Retriever-B. G-Retriever-B shows the best performance among all baselines. However, TAONA-B still outperforms G-Retriever-B, with a 2% improvement in F1-score. This enhancement is attributed to its specially designed components, such as the TAONA-GraphEncoder and TAONA-NodeClassifier. In addition, the result of removing $\mathbf{p}_{\text{graph}}$ or $\mathbf{p}_{\text{text}}$ drops, which demonstrates that both of them play an important role in the performance gain of TAONA-B.

Table 2: Performance comparison for the B-side task (%) on B-WebQSP.

| Metrics | F1-score |
|---|---|
| TAONA-B w/o $\mathbf{p}_{\text{graph}}$ | 0.43 |
| Frozen LLM + Prompt Tuning (PT) | 1.29 |
| G-Retriever | 28.24 |
| G-Retriever-B | <u>28.57</u> |
| TAONA-B w/o $\mathbf{p}_{\text{text}}$ | 22.03 |
| TAONA-B | **30.53** |

### 4.6 Ablation study and hyperparameter study

In this subsection, we perform an ablation study on TAONA-B and a hyperparameter study on




(a) Ablation study on TaoNa-B. (b) Study on GNN's layers in TaoNa-A. (c) Study on GNN's layers in TaoNa-B.

Figure 4: Ablation study (a) & parameter study (b and c).

the number of layers in TaoNa-GraphEncoder for both TaoNa-A and TaoNa-B. For the ablation study, we focus on evaluating the effectiveness of the TaoNa-NodeClassifier, as TaoNa-GraphEncoder's role in TaoNa-A was previously analyzed. Figure 4 (a) shows the performance of TaoNa-B without TaoNa-NodeClassifier. It is evident that TaoNa-NodeClassifier enhances F1-score by approximately 2%, demonstrating its crucial role in improving TaoNa-B's performance on the B-side task. Additionally, we examine the impact of varying the number of layers in TaoNa-GraphEncoder, with results presented in Figure 4 (b) and Figure 4 (c). The results indicate that three layers achieve the best performance in TaoNa-A on ExplaGraphs, whereas two layers offer about a 2% improvement in F1-score over configurations with one, three, or four layers in TaoNa-B. These findings suggest that two/three layers are enough for textual graph understanding and question answering tasks.

# 5 Related Work

## 5.1 Retrieval Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) (Gao et al., 2022; Sun et al., 2024) has earned significant attention for its ability to address limitations of large language models (LLMs), such as hallucinations, when answering domain-specific or knowledge-intensive questions. Existing RAG approaches can be categorized into three types: naive RAG, advanced RAG, and modular RAG. Naive RAGs (Ma et al., 2023) follow a straightforward process consisting of indexing, retrieval, and generation. To enhance the performance of naive RAGs, advanced RAGs employ additional techniques in the pre-retrieval stage, such as query transformation, expansion, and rewriting (Peng et al., 2024; Zheng et al., 2023; Gao et al., 2022). In the post-retrieval stage, reranking (Blagojevi, 2023) is commonly used to improve results. Modular RAGs integrate diverse strategies to enhance the RAG pipeline. They may include various data types, such as text, databases, and knowledge graphs, in the search module. Additionally, modular RAGs often use LLMs to refine retrieval queries (Yu et al., 2022). The proposed TaoNa framework falls into the category of modular RAGs.

## 5.2 Graphs and Large Language Models

Large language models (LLMs) are trained on extensive corpora, while textual and knowledge graphs provide rich factual and structural information (Wang et al., 2018; Du et al., 2021; Zhang et al., 2025; Yan et al., 2021a,b, 2023a,b; Chen et al., 2024; Ai et al., 2025; Lin et al., 2024, 2025a,b; Liu et al., 2025). Combining LLMs with graphs is a natural choice for applications such as question answering and text generation (Zeng et al., 2023a, 2024a, 2023b, 2024c, 2025; Roach et al., 2020; Li et al., 2024; Yan et al., 2024b,c; Yu et al., 2025a,b; Bao et al.; Yang et al., 2024). This integration can be categorized into three main approaches: KG-enhanced LLMs involve incorporating knowledge graphs (KGs) into LLMs in various ways. KG-enhanced pre-training (Liu et al., 2020; Sun et al., 2020) improves LLMs' knowledge representation by integrating KGs during the training process. KG-enhanced inference (Lewis et al., 2020; Wang et al., 2023b; Sun et al., 2023; Ma et al., 2024; Li et al., 2023) enables LLMs to utilize KG information during inference without retraining. KG-enhanced interpretability (Meng et al., 2021; Luo et al., 2023b) uses KGs to better understand the knowledge learned by LLMs. LLM-augmented KGs enhance traditional KG tasks with the capabilities of LLMs. This includes KG embedding (Wang et al., 2023c), which improves the representation of KGs; KG completion (Kim et al., 2020; Liao et al., 2023), which helps fill in missing information; and KG construction (Bosselut et al., 2019; Hao et al., 2022), which supports the creation of new KGs. Synergized LLMs+KGs (Yasunaga et al., 2022) merge KG-enhanced LLMs and LLM-augmented KGs in an iterative fashion, leveraging the strengths of both approaches to create a unified solution. Additional insights into the integration of graphs and LLMs can be found in (Pan et al., 2024).

## 6 Conclusion

In this paper, we explore the problem of textual graph understanding and question answering, addressing both the A-side and B-side tasks. To the best of our knowledge, we are the first to introduce the B-side task. To tackle these tasks, we present a novel model, TaoNa, which includes TaoNa-A for the A-side task and TaoNa-B for the B-side task. TaoNa-A features a specialized TaoNa-GraphEncoder designed to generate the graph prompting vector, while TaoNa-B incorporates a TaoNa-NodeClassifier to predict node statuses. Extensive experiments demonstrate the effectiveness of both TaoNa-A and TaoNa-B.

## 7 Limitations and Ethical Impact

Our work focuses on a plug-and-play approach with frozen LLMs, which limits potential performance improvements that could be achieved through fine-tuning. Integrating the node status inference module with an LLM fine-tuning module in an end-to-end training pipeline may yield better results, which we leave for future work.

Additionally, our approach may have ethical implications, as the proposed TaoNa-B framework can be used to filter toxic or harmful information in QA systems designed to exclude such content. However, we do not emphasize this aspect in our paper, as TaoNa-B is not restricted to such use cases; it can also be applied to other domains, such as product recommendation in e-commerce platforms.

## References


Mengting Ai, Tianxin Wei, Yifan Chen, Zhichen Zeng, Ritchie Zhao, Girish Varatkar, Bita Darvish Rouhani, Xianfeng Tang, Hanghang Tong, and Jingrui He. 2025. Resmoe: Space-efficient compression of mixture of experts llms via residual restoration. *arXiv preprint arXiv:2503.06881*.

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*.

Yikun Ban, Yuchen Yan, Arindam Banerjee, and Jingrui He. 2021. Ee-net: Exploitation-exploration neural networks in contextual bandits. *arXiv preprint arXiv:2110.03177*.

Yikun Ban, Yuchen Yan, Arindam Banerjee, and Jingrui He. 2023. Neural exploitation and exploration of contextual bandits. *arXiv preprint arXiv:2305.03784*.

Wenxuan Bao, Zhichen Zeng, Zhining Liu, Hanghang Tong, and Jingrui He. Matcha: Mitigating graph structure shifts with test-time adaptation. In *The Thirteenth International Conference on Learning Representations*.

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.

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 3950–3957.

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.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. *arXiv preprint arXiv:1906.05317*.

Lingjie Chen, Ruizhong Qiu, Siyu Yuan, Zhining Liu, Tianxin Wei, Hyunsik Yoo, Zhichen Zeng, Deqing Yang, and Hanghang Tong. 2024. Wapiti: A watermark for finetuned open-source llms. *arXiv preprint arXiv:2410.06467*.

Fan RK Chung. 1997. *Spectral graph theory*, volume 92. American Mathematical Soc.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2022. *Introduction to algorithms*. MIT press.

Boxin Du, Si Zhang, Yuchen Yan, and Hanghang Tong. 2021. New frontiers of multi-network mining: Recent developments and future trend. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 4038–4039.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*.

Ian Goodfellow. 2016. *Deep Learning*. MIT Press.

Shibo Hao, Bowen Tan, Kaiwen Tang, Bin Ni, Xiyan Shao, Hengzhe Zhang, Eric P Xing, and Zhiting Hu. 2022. Bertnet: Harvesting knowledge graphs with arbitrary relations from pretrained language models. *arXiv preprint arXiv:2206.14268*.

T Hastie. 2009. The elements of statistical learning: Data mining, inference, and prediction.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv preprint arXiv:2402.07630*.

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

Baoyu Jing, Yuchen Yan, Kaize Ding, Chanyoung Park, Yada Zhu, Huan Liu, and Hanghang Tong. 2024. Sterling: Synergistic representation learning on bipartite graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12976–12984.

Baoyu Jing, Yuchen Yan, Yada Zhu, and Hanghang Tong. 2022. Coin: Co-cluster infomax for bipartite graphs. *arXiv preprint arXiv:2206.00006*.

Bosung Kim, Taesuk Hong, Youngjoong Ko, and Jungyun Seo. 2020. Multi-task learning for knowledge graph completion with pre-trained language models. In *Proceedings of the 28th international conference on computational linguistics*, pages 1737–1743.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

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.

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

Jinning Li, Ruipeng Han, Chenkai Sun, Dachun Sun, Ruijie Wang, Jingying Zeng, Yuchen Yan, Hanghang Tong, and Tarek Abdelzaher. 2024. Large language model-guided disentangled belief representation learning on polarized social graphs. In *2024 33rd International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9. IEEE.

Jinning Li, Huajie Shao, Dachun Sun, Ruijie Wang, Yuchen Yan, Jinyang Li, Shengzhong Liu, Hanghang Tong, and Tarek Abdelzaher. 2022. Unsupervised belief representation learning with information-theoretic variational graph auto-encoders. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1728–1738.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2023. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. *arXiv preprint arXiv:2305.13269*.

Mingfu Liang, Xi Liu, Rong Jin, Boyang Liu, Qiuling Suo, Qinghai Zhou, Song Zhou, Laming Chen, Hua Zheng, Zhiyuan Li, et al. 2025. External large foundation model: How to efficiently serve trillions of parameters for online ads recommendation. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 344–353.

Ruotong Liao, Xu Jia, Yangzhe Li, Yunpu Ma, and Volker Tresp. 2023. Gentkg: Generative forecasting on temporal knowledge graph with large language models. *arXiv preprint arXiv:2310.07793*.

Haokun Lin, Teng Wang, Yixiao Ge, Yuying Ge, Zhichao Lu, Ying Wei, Qingfu Zhang, Zhenan Sun, and Ying Shan. 2025a. Toklip: Marry visual tokens to clip for multimodal comprehension and generation. *arXiv preprint arXiv:2505.05422*.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. 2024. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. *Advances in Neural Information Processing Systems*, 37:87766–87800.

Haokun Lin, Haobo Xu, Yichen Wu, Ziyu Guo, Renrui Zhang, Zhichao Lu, Ying Wei, Qingfu Zhang, and Zhenan Sun. 2025b. Quantization meets dllms: A systematic study of post-training quantization for diffusion llms. *arXiv preprint arXiv:2508.14896*.

Lihui Liu, Boxin Du, Jiejun Xu, Yinglong Xia, and Hanghang Tong. 2022. Joint knowledge graph completion and question answering. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 1098–1108.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2901–2908.

Xiaolong Liu, Zhichen Zeng, Xiaoyi Liu, Siyang Yuan, Weinan Song, Mengyue Hang, Yiqun Liu, Chaofei Yang, Donghyun Kim, Wen-Yen Chen, et al. 2024. A collaborative ensemble framework for ctr prediction. *arXiv preprint arXiv:2411.13700*.

Zhining Liu, Rana Ali Amjad, Ravinarayana Adkathimar, Tianxin Wei, and Hanghang Tong. 2025. Selfelicit: Your language model secretly knows where is the relevant evidence. *arXiv preprint arXiv:2502.08767*.

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*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Linhao Luo, Jiaxin Ju, Bo Xiong, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023a. Chatrule: Mining logical rules with large language models for knowledge graph reasoning. *arXiv preprint arXiv:2309.01538*.

Linhao Luo, Thuy-Trang Vu, Dinh Phung, and Gholamreza Haffari. 2023b. Systematic assessment of factual knowledge in large language models. *arXiv preprint arXiv:2310.11638*.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. 2024. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. *arXiv preprint arXiv:2407.10805*.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. *arXiv preprint arXiv:2305.14283*.

Zaiqiao Meng, Fangyu Liu, Ehsan Shareghi, Yixuan Su, Charlotte Collins, and Nigel Collier. 2021. Rewire-then-probe: A contrastive recipe for probing biomedical knowledge of pre-trained language models. *arXiv preprint arXiv:2110.08173*.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*.

Wenjun Peng, Guiyang Li, Yue Jiang, Zilong Wang, Dan Ou, Xiaoyi Zeng, Derong Xu, Tong Xu, and Enhong Chen. 2024. Large language model based long-tail query rewriting in taobao search. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 20–28.

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*.

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

Shane Roach, Connie Ni, Alexei Kopylov, Tsai-Ching Lu, Jiejun Xu, Si Zhang, Boxin Du, Dawei Zhou, Jun Wu, Lihui Liu, et al. 2020. Canon: Complex analytics of network of networks for modeling adversarial activities. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1634–1643. IEEE.

Andy Sun, Tianqi Zheng, Aakash Kolekar, Rohit Patki, Hossein Khazaei, Xuan Guo, George Cai, David Liu, Ruirui Li, Yupin Huang, Dante Everaert, Hanqing Lu, Garima Patel, and Monica Cheng. 2024. A product-aware query auto-completion framework for e-commerce search via retrieval-augmented generation method. In *SIGIR 2024 Workshop on Information Retrieval's Role in RAG Systems (IR-RAG)*.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M Ni, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. *arXiv preprint arXiv:2307.07697*.

Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang. 2020. Colake: Contextualized language and knowledge embedding. *arXiv preprint arXiv:2010.00309*.

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

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Dingsu Wang, Yuchen Yan, Ruizhong Qiu, Yada Zhu, Kaiyu Guan, Andrew Margenot, and Hanghang Tong. 2023a. Networked time series imputation via position-aware graph enhanced variational autoencoders. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2256–2268.

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.

Jianing Wang, Qiushi Sun, Xiang Li, and Ming Gao. 2023b. Boosting language models reasoning with chain-of-knowledge prompting. *arXiv preprint arXiv:2306.06427*.

Peng Wang, Xin Xie, Xiaohan Wang, and Ninyu Zhang. 2023c. Reasoning through memorization: Nearest neighbor knowledge graph embeddings. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 111–122. Springer.

Ruijie Wang, Baoyu Li, Yichen Lu, Dachun Sun, Jinning Li, Yuchen Yan, Shengzhong Liu, Hanghang Tong, and Tarek F Abdelzaher. 2023d. Noisy positive-unlabeled learning with self-training for speculative knowledge graph reasoning. *arXiv preprint arXiv:2306.07512*.

Ruijie Wang, Yuchen Yan, Jialu Wang, Yuting Jia, Ye Zhang, Weinan Zhang, and Xinbing Wang. 2018. Acekg: A large-scale knowledge graph for academic data mining. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 1487–1490.

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

Karen Weise. 2024. Amazon has new chatbot for shoppers. *The New York Times*, pages B1–B1.

Haobo Xu, Yuchen Yan, Dingsu Wang, Zhe Xu, Zhichen Zeng, Tarek F Abdelzaher, Jiawei Han, and Hanghang Tong. 2024. Slog: An inductive spectral graph neural network beyond polynomial filter. In *Forty-first International Conference on Machine Learning*.

Yuchen Yan, Yuzhong Chen, Huiyuan Chen, Xiaoting Li, Zhe Xu, Zhichen Zeng, Lihui Liu, Zhining Liu, and Hanghang Tong. 2024a. Thegcn: Temporal heterophilic graph convolutional network. *arXiv preprint arXiv:2412.16435*.

Yuchen Yan, Yuzhong Chen, Huiyuan Chen, Minghua Xu, Mahashweta Das, Hao Yang, and Hanghang Tong. 2023a. From trainable negative depth to edge heterophily in graphs. *Advances in Neural Information Processing Systems*, 36:70162–70178.

Yuchen Yan, Yongyi Hu, Qinghai Zhou, Lihui Liu, Zhichen Zeng, Yuzhong Chen, Menghai Pan, Huiyuan Chen, Mahashweta Das, and Hanghang Tong. 2024b. Pacer: Network embedding from positional to structural. In *Proceedings of the ACM Web Conference 2024*, pages 2485–2496.

Yuchen Yan, Yongyi Hu, Qinghai Zhou, Shurang Wu, Dingsu Wang, and Hanghang Tong. 2024c. Topological anonymous walk embedding: A new structural node embedding approach. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 2796–2806.

Yuchen Yan, Baoyu Jing, Lihui Liu, Ruijie Wang, Jinning Li, Tarek Abdelzaher, and Hanghang Tong. 2023b. Reconciling competing sampling strategies of network embedding. *Advances in Neural Information Processing Systems*, 36:6844–6861.

Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, and Hanghang Tong. 2021a. Dynamic knowledge graph alignment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4564–4572.

Yuchen Yan, Si Zhang, and Hanghang Tong. 2021b. Bright: A bridging algorithm for network alignment. In *Proceedings of the web conference 2021*, pages 3907–3917.

Yuchen Yan, Qinghai Zhou, Jinning Li, Tarek Abdelzaher, and Hanghang Tong. 2022. Dissecting cross-layer dependency inference on multi-layered interdependent networks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 2341–2351.

Xiaodong Yang, Huiyuan Chen, Yuchen Yan, Yuxin Tang, Yuying Zhao, Eric Xu, Yiwei Cai, and Hanghang Tong. 2024. Simce: Simplifying cross-entropy loss for collaborative filtering. *arXiv preprint arXiv:2406.16170*.

Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D Manning, Percy S Liang, and Jure Leskovec. 2022. Deep bidirectional language-knowledge graph pretraining. *Advances in Neural Information Processing Systems*, 35:37309–37323.

Hyunsik Yoo, Zhichen Zeng, Jian Kang, Ruizhong Qiu, David Zhou, Zhining Liu, Fei Wang, Charlie Xu, Eunice Chan, and Hanghang Tong. 2024. Ensuring user-side fairness in dynamic recommender systems. In *Proceedings of the ACM Web Conference 2024*, pages 3667–3678.

Qi Yu, Zhichen Zeng, Yuchen Yan, Zhining Liu, Baoyu Jing, Ruizhong Qiu, Ariful Azad, and Hanghang Tong. 2025a. Planetalign: A comprehensive python library for benchmarking network alignment. *arXiv preprint arXiv:2505.21366*.

Qi Yu, Zhichen Zeng, Yuchen Yan, Lei Ying, R Srikant, and Hanghang Tong. 2025b. Joint optimal transport and embedding for network alignment. In *Proceedings of the ACM on Web Conference 2025*, pages 2064–2075.

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*.

Zhichen Zeng, Boxin Du, Si Zhang, Yinglong Xia, Zhining Liu, and Hanghang Tong. 2024a. Hierarchical multi-marginal optimal transport for network alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 16660–16668.

Zhichen Zeng, Xiaolong Liu, Mengyue Hang, Xiaoyi Liu, Qinghai Zhou, Chaofei Yang, Yiqun Liu, Yichen Ruan, Laming Chen, Yuxin Chen, et al. 2024b. Interformer: Towards effective heterogeneous interaction learning for click-through rate prediction. *arXiv preprint arXiv:2411.09852*.

Zhichen Zeng, Ruizhong Qiu, Wenxuan Bao, Tianxin Wei, Xiao Lin, Yuchen Yan, Tarek F Abdelzaher, Jiawei Han, and Hanghang Tong. 2025. Pave your own path: Graph gradual domain adaptation on fused gromov-wasserstein geodesics. *arXiv preprint arXiv:2505.12709*.

Zhichen Zeng, Ruizhong Qiu, Zhe Xu, Zhining Liu, Yuchen Yan, Tianxin Wei, Lei Ying, Jingrui He, and Hanghang Tong. 2024c. Graph mixup on approximate gromov–wasserstein geodesics. In *Forty-first International Conference on Machine Learning*.

Zhichen Zeng, Si Zhang, Yinglong Xia, and Hanghang Tong. 2023a. Parrot: Position-aware regularized optimal transport for network alignment. In *Proceedings of the ACM web conference 2023*, pages 372–382.

Zhichen Zeng, Ruike Zhu, Yinglong Xia, Hanqing Zeng, and Hanghang Tong. 2023b. Generative graph dictionary learning. In *International Conference on Machine Learning*, pages 40749–40769. PMLR.

Yuheng Zhang, Dian Yu, Tao Ge, Linfeng Song, Zhichen Zeng, Haitao Mi, Nan Jiang, and Dong Yu. 2025. Improving llm general preference alignment via optimistic online mirror descent. *arXiv preprint arXiv:2502.16852*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

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*.

# 8 Appendix

In this appendix, we include the following contents for the reviewers' reference: Clarification on the scope of the B-side task (Subsection 8.1); (2) Hyperparameter settings (Subsection 8.2); (3) Experimental results on fine-tuning the LLM (Subsection 8.3); (4) Detailed descriptions for datasets in the A-side task (Subsection 8.4) and examples of datasets and corresponding tasks of GraphQA benchmark from (He et al., 2024) (Figure 6); (5) Construction of B-WebQSP (Subsection 8.5); (6) New B-WebQSP dataset construction and evaluation; (7) Baselines for the A-side task (Subsection 8.7); and (8) The overview of TAONA-A in Figure 5.

## 8.1 Clarification on the scope of the B-side task

We would like to clarify that the B-side task is *not specifically designed* for question-answering on graphs containing sensitive, dangerous, or ethical information. Actually, the B-side task is a general selective question-answering task on knowledge graphs, where a small subset of nodes is labeled with a selective preference (e.g., Blocked/Unblocked). Due to space constraints, we primarily illustrated the B-side task using question answering on graphs containing sensitive/dangerous/ethical information as an example application in *Introduction* and throughout the paper.

To further clarify, we present an additional example application and contrast it with the one used in **Introduction**:

- **Example Application 1: Question Answering on Graphs Containing Sensitive, Dangerous, or Ethical Information.** In this scenario, a small subset of nodes is manually labeled as "dangerous" or "safe", *determined entirely by users/experts* employing our TAONA-B model. The goal of the B-side task in this application is to learn connectivity patterns from labeled nodes and infer the status of unlabeled nodes, reducing human annotation effort. TAONA-B ensures that only safe nodes are included in the generated response.

- **Example Application 2: Product Search on an E-commerce Platform** A user searches for a product, and a shop-product knowledge graph indicates that two shops sell it. Shop 1 has paid a higher advertisement fee, granting it higher priority (Unblocked), while Shop 2 is Blocked. The B-side task here is to learn the blocking pattern (i.e., based on ad fees and shop connectivity in the KG) from a small set of labeled nodes and infer the status of other shops. TAONA-B ensures that only unblocked shops appear in the response.

- **Comparison and Objective of the B-side Task.** In Application 1, dangerous nodes tend to be connected or located closely in the underlying graph, so TAONA-B learns positive attention weights between them. In Application 2, competing shops selling similar products tend to be connected but may have opposite statuses (e.g., one blocked, one unblocked), so TAONA-B learns negative attention weights between connected nodes. This negative attention mechanism is a key motivation for our model, enabling it to capture different selective status patterns in the graph.

**Summary of clarification.** The B-side task is general-purpose, designed to capture selective status patterns from a small set of labeled nodes, regardless of the application. The initial blocked/unblocked nodes are completely determined/annotated by the user (e.g., the e-commerce platform), and blocking patterns vary across applications. The initial labeling process is not the focus of our paper; our model's goal is to infer the status of other nodes based on these initial labels.

Specifically, WebQSP is not a QA dataset containing sensitive or dangerous information, nor is B-WebQSP designed for sensitive information detection. Instead, B-WebQSP was constructed to evaluate whether models can learn and infer a blocking status pattern before generating responses via LLM. The application of question answering on graphs containing sensitive or dangerous information is merely an example use case, not the definition of the B-side task.

## 8.2 Hyperparameters configuration

We utilize the open-source LLaMA 2-7b model (Touvron et al., 2023) as the frozen large language model (LLM). All experiments are conducted on two NVIDIA A100-80G GPUs, with four random seeds 0, 1, 2, 3. The number of layers for both TAONA-GraphEncoder and TAONA-NodeClassifier is selected from 1, 2, 3, 4, while the dropout rate is fixed at 0.05. In the Frozen LLM + Prompt Tuning setup, the virtual token length is set to 10, with a maximum text length of 512 tokens and a maximum generated token length of 32. We use the AdamW optimizer (Loshchilov and Hutter, 2017) with a learning rate of 1e-5. The batch size is selected from 1, 2, 4, 8, and the number of epochs is searched within 1, 5, 10. The hidden dimension for both TAONA-GraphEncoder and TAONA-NodeClassifier is set to 1024. For the subgraph construction process, the parameter $k$ and all other parameters follow those set in G-retriever (He et al., 2024). Specifically: For SceneGraphs, we set $k = 3$ for both edges and nodes, with $c = 1$. For WebQSP and B-WebQSP, we set $k = 3$ for nodes, $k = 5$ for edges, and $c = 0.5$ for edge cost. For ExplaGraphs, given the small graph size, the entire graph is retrieved as the subgraph. The hyperparameters for all baseline models are consistent with those specified in the GraphQA benchmark (He et al., 2024).

## 8.3 Experimental results on fine-tuning the LLM

To further demonstrate the effectiveness of the proposed TAONA-B, we conducted additional experiments on B-WebQSP, enabling LLM fine-tuning via LoRA (Hu et al., 2022). Specifically, we compare the following three models:

- G-Retriever with fine-tuned LLM (G-Retriever-FT)
- TAONA-B with fine-tuned LLM but excluding the NodeClassifier (TAONA-B-FT w/o NodeClassifier)
- TAONA-B with fine-tuned LLM (TAONA-B-FT)

The average precision, recall, and F1-score are presented in Table 3.

First, we observe that fine-tuning the LLM via LoRA significantly boosts TAONA-B's performance. TAONA-B-FT achieves an F1-score of 35.05%, up from 32.28% in G-Retriever-FT, representing a relative improvement of 10%. This improvement is substantial compared to the frozen LLM setting presented in the paper, where TAONA-B (30.53%) outperformed G-Retriever (28.24%).

Second, we find that the NodeClassifier plays a crucial role in performance improvement, increasing the F1-score from 33.44% to 35.05% in TAONA-B-FT.

Additionally, for the A-side task, we conducted experiments with LLM fine-tuning via LoRA on WebQSP. TAONA-A-FT achieves 75.12% Hit@1, compared to 73.04% from G-Retriever-FT. Notably, the performance gain with LoRA fine-tuning is more pronounced than that in the frozen LLM setting presented in the paper (71.23% in TAONA-A vs. 70.02% in G-Retriever).

Overall, these results further validate the effectiveness of TAONA-A/-B, and the observed improvements are non-trivial.

### 8.4 Dataset descriptions for A-side task

The statistics for three datasets in A-side task are provided in Table 4. ExplaGraphs is designed for generative commonsense reasoning and focuses on constructing explanation graphs for stance prediction in debates. It offers detailed, unambiguous commonsense-augmented graphs to evaluate whether arguments support or refute a given belief. The primary task is to determine whether the arguments are supportive or contradictory. SceneGraphs is a visual question answering dataset that includes 100,000 scene graphs, each describing objects, attributes, and relations within an image. This dataset challenges users with tasks that require spatial understanding and multi-step inference. The task is to answer open-ended questions based on the textual description of a scene graph. WebQSP is a large-scale multi-hop knowledge graph QA dataset containing 4,737 questions. It utilizes a subset of Freebase (Bollacker et al., 2008), focusing on facts within 2 hops of the entities mentioned in the questions. The task involves answering questions that necessitate multi-hop reasoning.

### 8.5 Construction of B-WebQSP

In this subsection, we introduce the details of constructing the B-WebQSP dataset. Specifically, we start by randomly selecting a small ratio of nodes as initial *blocked nodes* ($\omega_1 = 0.1$). Then, using these labelled nodes as a starting point, we apply the Breadth-First Search (BFS) algorithm (Cormen et al., 2022) within an $H$-hop area [6] to label additional nodes. During the BFS process, within $H$ hops from the initially labelled nodes, we assign a probability of $\omega_2 = 0.95$ that the next reachable node will be marked as a *blocked node*. After completing this step, any remaining nodes are considered *unblocked nodes*. Once the ground truth statuses for all nodes are established, we randomly select 10% of the nodes' statuses as labelled to form the training set $V_{\text{train}}$ for the B-side task. The output of the B-side task is a list of the combination of status and the node itself (e.g., *Blocked Bomb*).

[6] $H = 1$ in our experiments.

### 8.6 New B-WebQSP dataset construction and evaluation

To better illustrate that (1) the scope of the B-side task in Subsection 8.1 is a general selective question-answering task on knowledge graphs; and (2) the dataset B-WebQSP is constructed to evaluate whether models can learn and infer a blocking status pattern before generating responses via LLM rather than designed for sensitive information detection, we construct an additional B-WebQSP dataset with a different blocking pattern. The new masking strategy aligns with **Example Application 2** in Subsection 8.1, where connected node pairs tend to have opposite block statuses. This new strategy is entirely different from the one used in the main content of our paper.

**Original B-WebQSP used in the main content.**

- 10% nodes are randomly selected as initially blocked.
- For nodes connected to these blocked nodes, the probability of being blocked is 95%.
- Connected nodes tend to have the same status.

**New B-WebQSP.**

- 10% nodes are randomly selected as initially blocked.
- For nodes directly connected to blocked nodes, the probability of being blocked is 20%.
- If a one-hop node is blocked, its two-hop neighbor has a 20% chance of being blocked.
- If a one-hop node is unblocked, its two-hop neighbor has a 80% chance of being blocked.
- Connected nodes tend to have opposite statuses, which differs entirely from the original strategy.

The results of G-Retriever and TAONA-B on the new B-WebQSP dataset are shown in Table 5. TAONA-B still outperforms G-Retriever, demonstrating its robustness.

Table 3: Additional Results about fine-tuning the LLM on B-webQSP (%).

| Models | Average Precision | Average Recall | Average F1 Score |
|---|---|---|---|
| G-Retriever-FT | 39.58 | 32.04 | 32.28 |
| TaoNa-B-FT w/o NodeClassifier | 39.76 | 34.28 | 33.44 |
| TaoNa-B-FT | **43.05** | **34.53** | **35.05** |

Table 4: Statistics of datasets.

| Dataset | ExplaGraphs | SceneGraphs | WebQSP | B-WebQSP |
|---|---|---|---|---|
| #Graphs | 2,766 | 100,000 | 4,737 | 4,737 |
| Average #Nodes | 5.17 | 19.13 | 1370.89 | 1370.89 |
| Average #Edges | 4.25 | 68.44 | 4252.37 | 4252.37 |
| Node Attribute | Commonsense concepts | Object attributes | Entities in Freebase | Entities in Freebase |
| Edge Attribute | Commonsense relations | Spatial relations | Relations in Freebase | Relations in Freebase |
| Task | Commonsense reasoning | Scene graph QA | KGQA | KGQA with blocked information |
| Evaluation metrics | Accuracy | Accuracy | Hit@1 | F1-score |

Table 5: Experimental results on newly built B-WebQSP dataset.

| Models | Average F1 Score (%) |
|---|---|
| G-Retriever | 37.00 |
| TaoNa-B | **39.17** |

### 8.7 Baselines

We have 8 baselines for the A-side task.

- **Zero-shot**. In this baseline, Given a textual graph description and a task description, the LLM is immediately asked to produce the desired output without any other information.
- **Zero-CoT** (Kojima et al., 2022). This baseline is a follow-up to CoT prompting (Wei et al., 2022) and appends the words "Let's think step by step." to the end of a question.
- **CoT-BAG** (Wang et al., 2024). This method adds "Let's construct a graph with the nodes and edges first." after the textual description of the graph, which forms a whole prompt.
- **KAPING** (Baek et al., 2023). This method is specially designed for knowledge graph question answering. It first retrieves all relevant triples and adds them to the input question in the form of a prompt, which is then forwarded to LLMs to generate the answer.
- **Graph-based Inference**. In this method, all textual information in $\mathcal{G}$ is included as a textual prompt, and a frozen LLM is used for question answering, with the query.
- **Frozen LLM + Prompt Tuning (PT)**. This approach adds a soft prompt for tuning while keeping the LLM's parameters frozen;
- **GraphToken** (Perozzi et al., 2024). This method encodes the whole graph with classical GNN (Kipf and Welling, 2016) as an embedding and regards this embedding as a graph prompting vector.
- **G-Retriever** (He et al., 2024). This baseline performs RAG over the textual graph and is also part of the GraphQA benchmark (He et al., 2024).

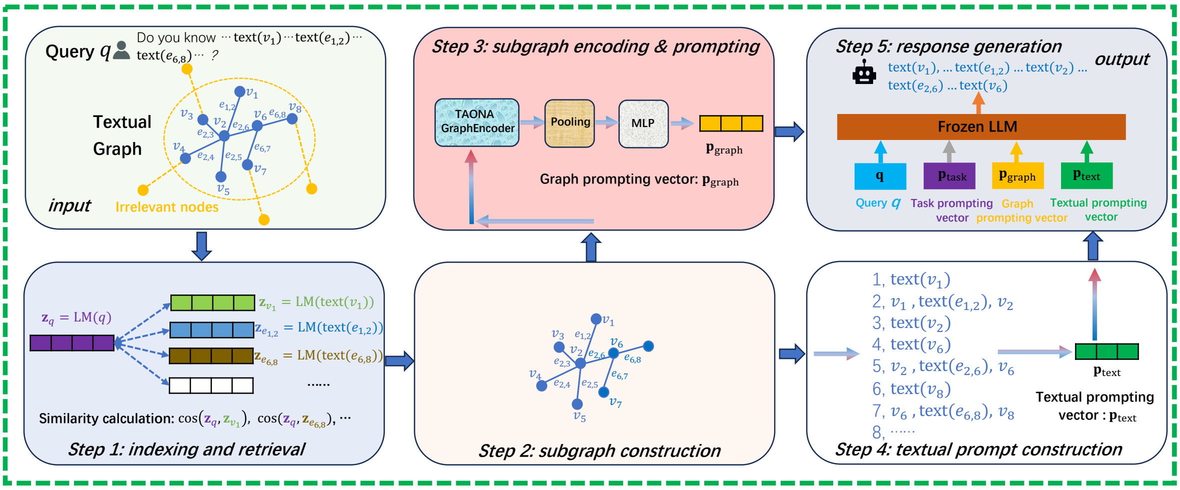



Figure 5: Overview of TAONA-A. Compared with TAONA-B, TAONA-A does not include TAONA-NodeClassifier in step 2 and the statuses of nodes in step 4 when constructing the textual prompt.

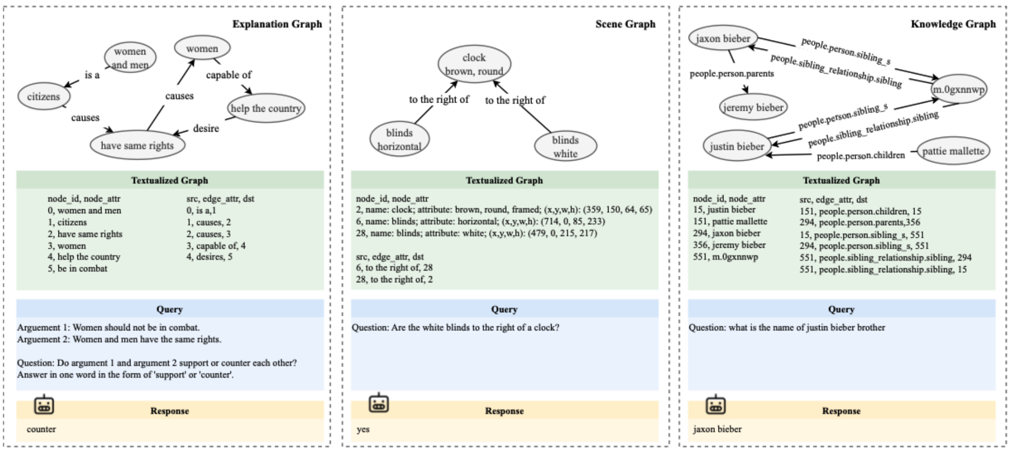



Figure 6: Example of datasets and corresponding tasks.