AGE: Adaptive-masking for Graph Embedding in Graph Retrieval-Augmented Generation

Bao Long Nguyen Huu[1] and Atsushi Hashimoto[2]

[1] OMRON Corporation
[2] OMRON SINIC X Corporation

Abstract. GraphRAG is an extension of retrieval-augmented generation (RAG) that supports large language models (LLMs) by referring to graph-structured data as external knowledge. While this technique ideally captures intricate relationships, it often struggles with graph representations for LLMs, particularly for frozen LLMs, due to the misalignment between graph-based and text-based latent features. We tackle this issue by introducing the *Adaptive-masking for Graph Embedding (AGE)*. AGE employs a Transformer in a mask-based self-supervised learning (SSL) approach. We designed the architecture similar to text embedding encoders, addressing the latent feature misalignment. In contrast to natural language texts, graphs are concise representations, and there exist *key nodes* that hold dominant contextual information, which are challenging to predict from their surroundings. Masking such key nodes leads to inefficiency in the SSL process. Therefore, AGE focuses on predicting nodes apart from key nodes, utilizing a learnable node sampler. Our experimental results indicate that AGE significantly improves approaches using non-parametric search component in GraphQA tasks, achieving superior accuracy across four benchmark datasets with distinct characteristics.

Keywords: Knowledge Graph Question Answering · Retrieval-Augmented Generation · Reinforcement Learning · Self Supervised Learning.

1 Introduction

Large Language Models (LLMs) such as GPT [58, 59], Claude [2], Gemini [71], Qwen [83], and LLaMA [24] have significantly advanced natural language understanding and generation capabilities. Retriever-Augmented Generation (RAG) [20, 23, 80] integrates query-relevant information into the generation process, enabling LLMs to access and utilize domain-specific knowledge beyond their pretraining corpus. However, although RAG enhances LLMs with external data, it may struggle to capture essential structured relationships, reducing search precision and reasoning effectiveness [84, 89]. Graph Retriever Augment Generation (GraphRAG) [18,55] is a technology that uses graphs to overcome the limitations of RAG. Graph data, represented by nodes (entities) and edges (relationships), clearly presents complex relationships. This provides several benefits, such as

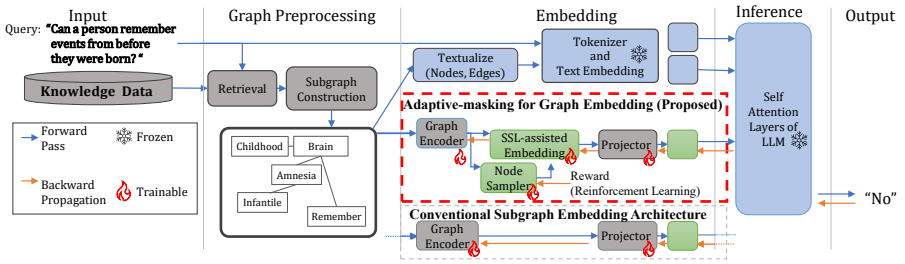


Fig. 1: Overview of GraphRAG with the proposed *Adaptive-masking for Graph Embedding* (AGE) embedding. 1) Retrieval: Find graph elements relevant to the query using a non-parametric process. 2) Subgraph Construction: Extend retrieved graph elements with their adjacencies [32]. 3) Embedding: Use tokenizer and text embedder for textualized graph and query. Apply AGE for structured relationships of the graph. 4) Inference: Input embeddings into LLM to generate an answer.

facilitating data integration [36], improving search accuracy [9, 28], enhancing inference capabilities [27, 29], and reducing hallucinations [32]. By capturing sub-graphs, the broader context and interconnections within the graph structure can be captured, enabling comprehensive information to be accessed for LLMs enhance the performance in domain-specific tasks.

This study investigates GraphRAG methods that operate within practical computational costs. Fine-tuning LLMs can enhance GraphRAG performance, yet it is resource-intensive. Instead, previous methods often focused on the retrieval module as it is a key factor for GraphRAG performance. Trainable retrievers, such as LLM-based retrievers [50,66] realize a higher retrieval accuracy. However, this strategy still requires significant computational overhead. Non-parametric retrievers [32,85] are efficient and low-cost but may contain redundant or missing critical nodes, leading to the lack of explicit structural constraints. To maintain practicality, we base our method on non-parametric retrievers with frozen LLM, and improve performance of structural representation by updating the graph embedding module. Embeddings play a crucial role in bridging the gap between retrieved graph data and the LLM input space. Multiple methods [32, 61] use graph embedding together with textualized graph representation (Fig. 1), indicating that the graph encoder should embed relationships between elements, rather than their individual contents. What is the optimal strategy to achieve such encoding for a frozen LLM? We considered two factors: similarity of the embedding space to the LLM's text encoder and its relationship embedding capability. Since the LLM's text encoder uses mask-based SSL [48, 57, 62], which learns to embed relationships between elements into the embedding space by optimize the reconstruction of masked elements. These factors aim to embed relationships between nodes within the retrieved subgraph into the embedding space by adapting the mask-based SSL with minimal modification.

To realize this intention, we propose a novel embedding strategy, the *Adaptive-masking for Graph Embedding (AGE)*. The architecture of AGE is designed to

imitate the general self-supervised text embedding process, while incorporating the Joint-Embedding Predictive Architecture (JEPA) [45], which improves representations in the embedding space by eliminating unnecessary detail reconstruction. Although generative SSL shows promise, the quality of reconstruction relies on the discriminability of input nodes [14, 79]. Random masking fails on non-discriminative nodes, leading to poor representations [8, 64]. To avoid this, the only major modification is adding a node sampler trained via reinforcement learning (RL) to selectively mask nodes, replacing traditional random node masking with an adaptive approach. The motivation for this approach stems from the fact that, graphs are concise logical structures with minimal redundancy. Hence, some nodes are crucial for maintaining graph integrity; we refer to them as *key nodes*. Our RL-based strategy aims to guide SSL to distinguish the representations of key and auxiliary nodes, encouraging LLMs to identify redundant information within the retrieved graph. The contributions of this paper are outlined as follows:

1. We propose AGE, a novel method that represents retrieved subgraphs via key-node and auxiliary-node embedded by RL-guided mask-based SSL.
2. Our study reveals adaptive masking approach's notable effectiveness over random masking within GraphRAG.
3. AGE uses a non-parametric retriever and open LLMs, while also achieving SOTA on three other benchmarks.

2 Related Work

2.1 Graph Representation for LLMs

In the context of representing graphs as input to LLMs, it is necessary to first convert the retrieved graph data into specific formats. We summarize two distinct formats: textualization and graph embeddings. Textualization [21, 41, 47] is a text-based formalization method designed to characterize and represent graph data. Node sequences are a popular form of textualization [13,50,66]. Some methods [12, 50, 66] propose LLM-based retrievers to extract reasoning paths. A node sequence ordered along the path aids LLM's reasoning. However, many studies report negative conclusions in interpreting text-encoded graphs with concurrent LLMs [26, 37, 76], suggesting a need for solutions beyond textualization.
The other format, graph embeddings, have recently been adopted in GraphToken [61]. Following this, G-Retriever [32] proposed a retrieval framework for graph embeddings. This occurs when graph embeddings are added as tunable prompts to the LLM in addition to their textualized representations. In this work, we improve the quality of LLM responses on the G-Retriever framework through enhancing the representation of graph embeddings. Some methods [36, 82] build self-alignment and cross-question module among retrieved entities, relations, and subgraph embedding elements. Some methods enhance embeddings through a two-stage training process [40, 75]. The first stage trains the embedding module on SSL alone; in the second stage, prompt tuning aligns the

structured relationships embedded for LLM input by the pretrained module. Since each LLM has its own domain embeddings and input spaces, two-stage training process prioritize maximizing performance. Instead, focusing on practical, we propose a one-stage training process SSL that integrates with prompt tuning.

2.2 Self-Supervised Learning

Many existing self-supervised learning architectures focus on learning representations that effectively capture relationships between input data. Joint-Embedding Architecture (JEA) [6, 10, 25] has shown considerable promise in advancing SSL methodologies. Joint-Embedding SSL for GNNs, such as GraphCL [87], GCA [91] and JOAO [88], learn node representations by contrasting positive and negative samples. Subsequent studies identified areas for enhancement in JEA [15, 42, 46], particularly the issue of mapping all inputs to a single constant vector, known as the collapsing problem. Generative Architecture (GA) [4, 17, 31] focuses on reconstructing masked portions of the input at either the pixel or token level. GraphMAE [33, 34] learns representations by reconstructing masked samples. These methods encourage the model to learn more robust and diverse representations, potentially reducing the risk of representation collapse [3, 15, 42]. Joint-Embedding Predictive Architectur (JEPA) [45] eliminate reconstruction of pixel or token-level details and enhances the semantic level of self-supervised representations [3, 5]. In this work, we first demonstrate JEPA's effectiveness in the GraphRAG framework. JEPA originates from cognitive neuroscience, suggesting that humans have an ability of top-down schema reasoning, aiding planning, decision-making, and problem-solving on complex tasks [56, 70, 72]. In GraphRAG, the tokens fed into the LLM's hidden layer should capture the ability. We implement it as JEPA in the graph embedding module. Cognitive science has revealed that a brain region called the temporal lobe plays a role in bottom-up associative learning, selecting key knowledge and linking it to related data [16, 19, 39]. These selection processes can also occur with new information. Aiming to reproduce this functionality in GraphRAG, AGE has the novel node sampler module.

3 Preliminaries

GQA with LLM. For a query q on a textual graph G, there is an optimal subgraph $\overline{S^*} \in S(G)$ and query relevant text-modal knowledge T^* that guides the LLM to produce expected answers, where $S(G)$ is the set of all subgraphs of G. The challenge of GraphRAG is to efficiently search for the relevant subgraph S^* and represent it to $\overline{S^*}$ for an LLM p_Φ improve generation. The probability distribution of the output sequence Y is given by:

$$p_\Phi(Y \mid [q, G]) = \prod_{i=1}^{n} p_\Phi(y_i \mid y_{<i}, [q, T^*, \overline{S^*}]), \tag{1}$$

where $y_{<i}$ represents the prefix tokens, and $[q, \overline{S^*}]$ indicates the concatenation of the query, relevant text-modal knowledge and optimal subgraph information,

respectively.

Joint-Embedding Predictive Architecture. Mask-based SSL methods such as MAE are well suited to handle corrupted input, as they learn to reconstruct missing or corrupted input parts. JEPA improves upon MAE by eliminating the reconstruction of unnecessary input feature details, focusing instead on learning more abstract representations. JEPA consists of an encoder $E_\theta(\cdot)$, predictor $P_\phi(\cdot)$ and target encoder $T_\theta(\cdot)$. The stop-gradient operation sg is employed to prevent representation collapse in the target encoder $T_\theta(\cdot)$. The predictor generates y from visible input x and masked input Δ_x. The encoder and predictor are trained simultaneously with the objective:

$$\min ||P_\phi\big(\Delta_x, E_\theta(x)\big) - \text{sg}\left(T_\theta(y)\right)||_2, \tag{2}$$

The loss is applied only to the predictions of the masked input Δ_x.

Reinforcement Learning. To estimate the key and auxiliary nodes on retrieved graph for mask-based SSL discriminative embedding. We adopt REINFORCE [67], a basic policy gradient method in RL. Let $\mathcal{D} = (q, Y^*)$ denote a corpus of training data, where Y^* is the complete reference label for query q. REINFORCE optimizes a policy π_θ parameterized by θ, to maximize reconstruction quality R_a for each masking action a. The policy gradient is given by:

$$\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_{q\sim\mathcal{D}, S^*\sim q}\left[\sum_{v\in S^*} \nabla \log \pi_\theta(a|v) \cdot R_a\right] \tag{3}$$

where where $\mathcal{J}(\theta)$ is the expected return. $\pi_\theta(a|v)$ denotes the probability distribution estimated by policy π_θ for taking action a (masking or not) on node v. The SSL framework with masked node reconstruction serves as the RL environment.

4 Approach

Our framework, illustrated in Figure 1, consists of four main steps: *input*, *graph preprocessing*, *embedding*, and *inference*. We adopt the previous method [32] that applies SentenceBert [62] to indexed knowledge data at (*input*) step and employ a static k-nearest neighbors [44] retrieval approach combined with Prize-Collecting Steiner Tree [7] subgraph construction during *graph preprocessing*. For *inference*, we can use arbitrary LLMs, as usual RAG methods. Therefore, this section focuses on the details of *embedding* step.

4.1 Text Embedding of Query and Text Graph

We transform the retrieved subgraph S^* into a textual format, following [32] as in the first two steps. The converted text is then concatenated with the input query q. The concatenated texts are embedded into h_{text} using a pretrained function for the frozen LLM, TextEmbedding, where $[;]$ denotes concatenation and L is the output token sequence length, as follows:

$$h_{\text{text}} = \text{TextEmbedding}([\text{textualize}(S^*); q]) \in \mathbb{R}^{L\times d_l}, \tag{4}$$

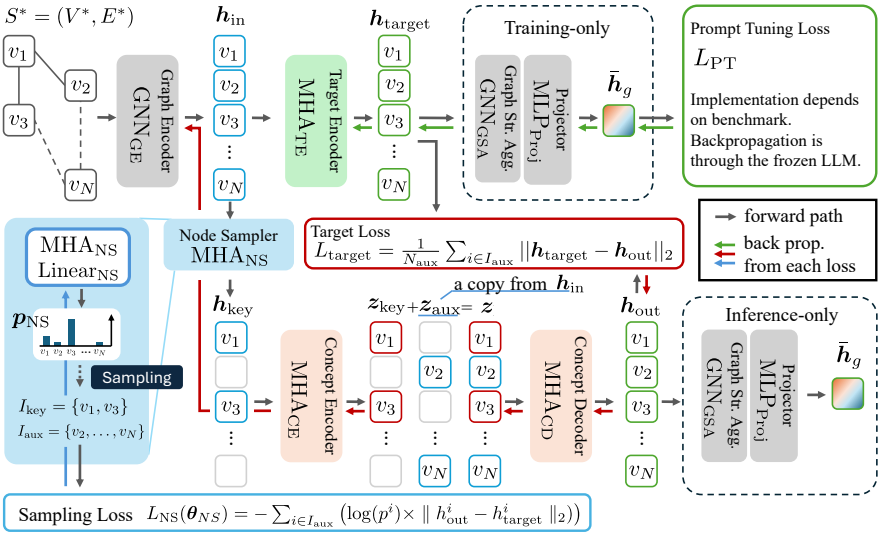


Fig. 2: Architecture for Adaptive-masking for Graph Embedding: During training, $\boldsymbol{h}_{\text{target}}$ is connected to the downstream for the target encoder training, while $\boldsymbol{h}_{\text{out}}$ is used during inference. The node sampler explores the optimal distribution for mask-based SSL for graphs. The loss functions train distinct sets of modules without overlap.

4.2 Adaptive-masking for Graph Embedding

AGE comprises a *node sampler*, *concept encoder-decoder*, *target encoder*, and *graph-structure-based aggregator* modules, as overviewed in Fig. 2. The AGE's input, $\boldsymbol{h}_{\text{in}}$, is encoded from the retrieved graph S^* with a conventional graph encoder. $\boldsymbol{h}_{\text{in}}$ is then passed to *node sampler* and *target encoder*. The node sampler categorizes the nodes into *key nodes* and the remaining *auxiliary nodes*. The selected *key nodes* are fed to *concept encoder-decoder*. The output $\boldsymbol{h}_{\text{out}}$ is trained to predict $\boldsymbol{h}_{\text{target}}$, the output of *target encoder*, forming a JEPA. The embedding $\boldsymbol{h}_{\text{out}}$ is then aggregated into a token to be fed to the LLM. The rest of this subsection explains each module in Fig. 2 individually.

Graph Encoder prepares input for AGE. The retrieved subgraph $S^* = (V^*, E^*)$ consists of query-relevant nodes V^* and edges E^*. $\boldsymbol{h}_{\text{in}}$ is obtained from S^* by the graph encoder GNN_{GE} as follows:

$$\boldsymbol{h}_{\text{in}} = \text{GNN}_{\text{GE}}(S^*; \boldsymbol{\theta}_{\text{GE}}) \in \mathbb{R}^{N \times d_g}, \tag{5}$$

where $\boldsymbol{\theta}_{\text{GE}}$ is parameter of GNN_{GE}, d_g is dimension of each output node feature, and $N = |V^*|$.

Node sampler estimates key nodes using $\boldsymbol{h}_{\text{in}}$ from the graph encoder for adapting masks on auxiliary nodes. The node sampler processes $\boldsymbol{h}_{\text{in}}$ through a Multi-Head Attention (MHA) network, a linear layer, and a softmax activation, to

obtain nodes probability scores $\boldsymbol{p}_{\mathrm{NS}}$ as follows:

$$\boldsymbol{z}_{\mathrm{NS}} = \mathrm{MHA}_{\mathrm{NS}}(\boldsymbol{h}_{\mathrm{in}}; \boldsymbol{\theta}_{\mathrm{NS}}^{\mathrm{MHA}}) \in \mathbb{R}^{N \times d_g}, \tag{6}$$

$$\boldsymbol{p}_{\mathrm{NS}} = \mathrm{Softmax}(\mathrm{Linear}(\boldsymbol{z}_{\mathrm{NS}}; \boldsymbol{\theta}_{\mathrm{NS}}^{\mathrm{Linear}})) \in [0,1]^{N \times 1}, \tag{7}$$

where $\boldsymbol{\theta}_{\mathrm{NS}} = \{\boldsymbol{\theta}_{\mathrm{NS}}^{\mathrm{MHA}}, \boldsymbol{\theta}_{\mathrm{NS}}^{\mathrm{Linear}}\}$ is the parameter of this module. We sample N_{key} nodes based on a categorical distribution defined by $\boldsymbol{p}_{\mathrm{NS}}$, where we decide N_{key} by the sampling rate ρ as $N_{\mathrm{key}} = \lceil \rho N \rceil$. Hereafter, we denote the sampled key nodes as I_{key} and auxiliary nodes as $I_{\mathrm{aux}} (= V^* \backslash I_{\mathrm{key}})$. Based on I_{key}, we extract key node features $\boldsymbol{h}_{\mathrm{key}} \in \mathbb{R}^{k \times d_g}$ from $\boldsymbol{h}_{\mathrm{in}}$ and input it to our concept encoder-decoder module.

Concept Encoder-Decoder consists of a concept encoder $\mathrm{MHA}_{\mathrm{CE}}$ and a concept decoder $\mathrm{MHA}_{\mathrm{CD}}$. $\mathrm{MHA}_{\mathrm{CE}}$ encode the input $\boldsymbol{h}_{\mathrm{key}}$ into the latent representation $\boldsymbol{z}_{\mathrm{key}}$ as follows:

$$\boldsymbol{z}_{\mathrm{key}} = \mathrm{MHA}_{\mathrm{CE}}(\boldsymbol{h}_{\mathrm{key}} + \mathrm{PE}(\boldsymbol{h}_{\mathrm{key}}); \boldsymbol{\theta}_{\mathrm{CE}}) \in \mathbb{R}^{k \times d_g}, \tag{8}$$

where $\mathrm{PE}(\cdot)$ represents positional encoding as defined by [52], and $\boldsymbol{\theta}_{\mathrm{CE}}$ is the parameter of $\mathrm{MHA}_{\mathrm{CE}}$. $\boldsymbol{z}_{\mathrm{key}}$ is combined with $\boldsymbol{z}_{\mathrm{aux}}$, placeholder vectors for unsampled auxiliary nodes with values copied from $\boldsymbol{h}_{\mathrm{in}}$, as in [90]. Let $\boldsymbol{z} \in \mathbb{R}^{N \times d_g}$ be the combined node features, which maintains $\boldsymbol{h}_{\mathrm{in}}$'s original node position. $\mathrm{MHA}_{\mathrm{CD}}$ decodes $\boldsymbol{h}_{\mathrm{out}}$ with the positional encoding PE as follows:

$$\boldsymbol{h}_{\mathrm{out}} = \mathrm{MHA}_{\mathrm{CD}}(\boldsymbol{z} + \mathrm{PE}(\boldsymbol{z}); \boldsymbol{\theta}_{\mathrm{CD}}) \in \mathbb{R}^{N \times d_g}, \tag{9}$$

where $\boldsymbol{\theta}_{\mathrm{CD}}$ is the parameter of $\mathrm{MHA}_{\mathrm{CD}}$. $\boldsymbol{h}_{\mathrm{out}}$ is the output of AGE, which we train to predict $\boldsymbol{h}_{\mathrm{target}}$, an embedding obtained from all nodes through the target encoder.

Target Encoder is applied to obtain a prediction target for the previous module in a semantic space (JEPA), as it works more robustly than in the input space (GA) [3,5,11,22]. The target encoder $\mathrm{MHA}_{\mathrm{TE}}$ projects $\boldsymbol{h}_{\mathrm{in}}$ to a target embedding $\boldsymbol{h}_{\mathrm{target}}$ as follows:

$$\boldsymbol{h}_{\mathrm{target}} = \mathrm{MHA}_{\mathrm{TE}}(\boldsymbol{h}_{\mathrm{in}} + \mathrm{PE}(\boldsymbol{h}_{\mathrm{in}}); \boldsymbol{\theta}_{\mathrm{TE}}) \in \mathbb{R}^{N \times d_g}, \tag{10}$$

where $\boldsymbol{\theta}_{\mathrm{TE}}$ is the parameter of $\mathrm{MHA}_{\mathrm{TE}}$. We train $\mathrm{MHA}_{\mathrm{TE}}$ with downstream tasks, optimizing it to produce embeddings that directly contribute to the task. $\mathrm{MHA}_{\mathrm{CE}}$ and $\mathrm{MHA}_{\mathrm{CD}}$ are trained in parallel with $\mathrm{MHA}_{\mathrm{TE}}$, with the target encoder learning graph representations for LLMs and the concept encoder-decoder learn to exploit key-node representations for mimic the target encoder's auxiliary node representations. Inferring masked auxiliary nodes condenses relational concepts between key and auxiliary nodes into $\boldsymbol{z}_{\mathrm{key}}$ (and thus $\boldsymbol{h}_{\mathrm{out}}$ in the downstream). This JEPA-derived mechanism would be synergetic with GraphRAG as the previous studies suffers from embedding graph's structured relationships efficiently [37].

Graph-structure-based Aggregator $\mathrm{GNN}_{\mathrm{GSA}}$ projects $\boldsymbol{h}_{\mathrm{out}}$ to a single token $\bar{\boldsymbol{h}}_g$. As the graph encoder module, we followed [32] for this module. It aggregates

$\boldsymbol{h}_{\text{out}}$ referring E^*, the edge connections of the original subgraph S^*. Where POOL is mean pooling and d_l is the input dimension of the target layer. The projector MLP_{Proj} adjusts the aggregated embeddings to fit the LLM's input dimension.

$$\boldsymbol{h}_g = \text{POOL}(\text{GNN}_{\text{GSA}}(\boldsymbol{h}_{\text{out}}; \boldsymbol{\theta}_{\text{GSA}})) \in \mathbb{R}^{d_g}, \quad \bar{\boldsymbol{h}}_g = \text{MLP}_{\text{Proj}}(\boldsymbol{h}_g; \boldsymbol{\theta}_{\text{Proj}}) \in \mathbb{R}^{d_l}, \tag{11}$$

4.3 Optimization of Adaptive-masking for Graph Embedding

This subsection describes three loss functions used in our method. In the training phase, we connect $\boldsymbol{h}_{\text{target}}$ in the target encoder stream to the LLM and optimize $\boldsymbol{\theta}_{\text{TE}}$ with the prompt tuning loss L_{PT}. During training the target encoder with L_{PT}, the concept encoder-decoder module is optimized exclusively with L_{target}, in a JEPA approach. Once entire network has been trained, we connect $\boldsymbol{h}_{\text{out}}$ to the downstream LLM rather than $\boldsymbol{h}_{\text{target}}$ at the inference phase. One challenge of AGE lies in optimizing $\boldsymbol{\theta}_{\text{NS}}$. Optimizing $\boldsymbol{\theta}_{\text{NS}}$ using L_{target} is difficult due to the non-differentiability of the sampling operation. Therefore, we propose an additional loss function L_{NS} for optimizing $\boldsymbol{\theta}_{\text{NS}}$.

Prompt tuning loss L_{PT} maximizes accuracy of a downstream task. It was originally introduced in [32], and we use the definition as is. We optimize $\boldsymbol{\theta}_{\text{TE}}$, $\boldsymbol{\theta}_{\text{GSA}}$, and $\boldsymbol{\theta}_{\text{Proj}}$ with L_{PT}. The concrete implementation depends on the benchmark tasks; refer to the original papers for details. Note that we train $\boldsymbol{\theta}_{\text{GE}}$ with L_{target} rather than L_{PT}, as the concept encoder-decoder is used in the inference phase and the upstream network should be optimized to that module.

Target loss $L_{\textbf{target}}$ optimizes parameters $\boldsymbol{\theta}_{\text{GE}}$, $\boldsymbol{\theta}_{\text{CE}}$, and $\boldsymbol{\theta}_{\text{CD}}$ to maximize embedding reconstruction by minimizing the distance between $\boldsymbol{h}_{\text{out}}$ and $\boldsymbol{h}_{\text{target}}$ for each auxiliary node indexed by I_{aux} as follows:

$$L_{\text{target}}(\boldsymbol{\theta}_{\text{GE}}, \boldsymbol{\theta}_{\text{CE}}, \boldsymbol{\theta}_{\text{CD}}) = \frac{1}{N_{\text{aux}}} \sum_{i \in I_{\text{aux}}} \parallel h^i_{\text{out}} - sg(h^i_{\text{target}}) \parallel_2, \tag{12}$$

with N_{aux} defined as $N - N_{\text{key}}$. The objective is to apply knowledge distillation to effectively represent key nodes for reconstructing auxiliary nodes. Furthermore, we apply normalization to enhance stability during the learning process, details are provided in the Appendix Table B.8.

Sampling loss $L_{\textbf{NS}}$ optimizes $\boldsymbol{\theta}_{\text{NS}}$ using RL-inspired supervision. Regarding the operation as an action, the node sampler as a policy network, and $\boldsymbol{h}_{\text{out}}$ as a state, we design L_{NS} on $\boldsymbol{p}_{\text{NS}} = \{p^1, \ldots, p^N\}$ in Eq. 7 as follows:

$$L_{\text{NS}}(\boldsymbol{\theta}_{\text{NS}}) = -\frac{1}{N_{\text{aux}}} \sum_{i \in I_{\text{aux}}} \left(\log(p^i) \times sg(\parallel h^i_{\text{out}} - h^i_{\text{target}} \parallel_2) \right). \tag{13}$$

The loss is back-propagated only to $\boldsymbol{\theta}_{\text{NS}}$. Here, for each node assigned to I_{aux}, larger $\parallel h^i_{\text{out}} - h^i_{\text{target}} \parallel_2$ increases p^i more, resulting in pushing such node into I_{key}. This reflects our aim to classify nodes that are difficult to predict from their surrounding as *key nodes*. Additional strategies for key node selections and

sampling optimization for RL are discussed in the Appendix Table B.6.
Total loss is given as $L_{\text{PT}} + L_{\text{target}} + L_{\text{NS}}$. Since we designed the optimization process so that each loss optimizes different modules without overlap, our methods does not require weight adjustment between these losses.

4.4 Analysis of Learning Objectives

To explain the motivation for architecture design and applying a distributed loss to each module, we analyze the learning objective of the $R\omega$ module, which represents expected $\overline{S^*}$ for LLM π_θ on LoRA finetuning as:

$$\begin{aligned}\mathcal{L} = \quad & \underbrace{-\mathbb{E}_{(S^*)}\big[\log R\omega(\overline{S^*} \mid S^*)\big]}_{\text{Loss of Graph Representation Module}} \\ & \underbrace{-\mathbb{E}_{(q,T^*,\overline{S^*})}\big[\log \pi_\theta(i \mid q, T^*, \overline{S^*})\, \pi_\theta(r \mid q, T^*, \overline{S^*}, i)\big]}_{\text{Loss of LLM}}\end{aligned} \tag{14}$$

According to Bayes' Theorem, given an input X, a target Y, and latent rationales Z, we can sample these latent rationales Z from the posterior distribution $P(Z|X,Y)$. This posterior represents the probability of latent Z given both the input X and the target Y. To compute the marginal likelihood of obtaining answer Y given input X, we marginalize over all possible rationales Z:

$$P(Y|X) = \sum_{Z \sim P(Z|X,Y)} P(Z,Y|X) = \sum_{Z \sim P(Z|X,Y)} P(Z|X) \cdot P(Y|X,Z) \tag{15}$$

The equations above show how to compute the marginal likelihood $P(Y|X)$. Equation (15) with first line makes explicit that Z is sampled from the posterior distribution $P(Z|X,Y)$. Second line extend applies the chain rule of probability to decompose $P(Z,Y|X)$ into two components: $P(Z|X)$ and $P(Y|X,Z)$. Following this analysis, we apply it to the learning objective for target representation $\overline{S^*}$ given input S, latent $\mathcal{Z}$ from a posterior $R\omega(\mathcal{Z}|S,\overline{S})$ that bridges S and $\overline{S}$. The marginal likelihood of $\overline{S}$ given S is:

$$\begin{aligned} R\omega(\overline{S^*}|S^*) &= \sum_{Z \sim R\omega(Z|S^*,\overline{S^*})} R\omega(Z,\overline{S^*}|S^*) \\ &= \sum_{Z \sim R\omega(Z|S^*,\overline{S^*})} R\omega(Z|S^*) \cdot R\omega(\overline{S^*}|S^*,Z) \end{aligned} \tag{16}$$

Above analysis shows that learning objective Graph Representation implicitly learns to identify the latent Z and map it to the expected $\overline{S^*}$ for LLM:

$$\begin{aligned} -\mathbb{E}\big[\log_{R\omega}(\overline{S^*} \mid S^*)\big] = \; & \underbrace{\mathbb{E}\big[\log R\omega(\mathcal{Z} \mid S^*, \overline{S^*})\big]}_{\text{Loss of Latent Identification}} \\ & \underbrace{-\mathbb{E}\big[\log R\omega(\mathcal{Z} \mid S^*) \cdot R\omega(\overline{S^*} \mid S^*, \mathcal{Z}))\big]}_{\text{Loss of Representation}} \end{aligned} \tag{17}$$

Instead of using a single model for both latent $\mathcal{Z}$ identification and $\overline{S^*}$ representation learning. We separate the learning into $Sampler_\theta$ for **latent identification by minimizing reconstruction loss** and $Encoder_\theta$-$Decoder_\theta$ for representation as:

$$-\mathbb{E}\big[\log_{R\omega}(\overline{S^*} \mid S^*)\big] \approx \underbrace{-\mathbb{E}_{(S^*,\overline{S^*})}\big[V_{key} \in \mathcal{Z} \sim \log\ Sampler_\theta(\mathcal{Z} \mid S^*, \overline{S^*})\big]}_{\text{Loss of Node Sampling}} \underbrace{-\mathbb{E}_{(S^*)}\big[\log Encoder_\theta(\mathcal{Z} \mid V_{key}) \cdot Decoder_\theta(\overline{S^*} \mid \Delta_{V_{masked}}, \mathcal{Z})\big]}_{\text{Loss of Encoder-Decoder}} \tag{18}$$

By separating the learning processes, the target encoder learns the representation directly, while the encoder-decoder learns to reconstruct this representation through **Evidence Lower Bound (ELBO) optimization**. Specifically, the node sampler learns to extrapolate $V_{key} \in \mathcal{Z}$, making static sampling illogical. Therefore, our node sampler with encoder-decoder architecture and explicit loss distribution yields efficient learning signals, faster convergence, and improved graph representations. To support our analysis, we provide empirical comparisons of sampling strategies in Appendix Figure B.3, B.4 and analyze the stability of the target encoder teacher module for the encoder-decoder in Appendix Table B.8. In the prompt tuning setting, given r as the reasoning trajectory, the learning objective for frozen LLMs with graph representation model R_ω is:

$$\mathcal{L} = \underbrace{\underbrace{\mathbb{E}\big[\log R_\omega(\mathcal{Z} \mid S^*, \overline{S^*})\big]}_{\text{Loss of Latent Identification}} \underbrace{-\mathbb{E}\big[\log R_\omega(\mathcal{Z} \mid S^*) \cdot R_\omega(\overline{S^*} \mid S^*, \mathcal{Z}))\big]}_{\text{Loss of Representation}}}_{\text{Frozen}} \underbrace{\underbrace{-\mathbb{E}\big[\log \pi_\theta(i \mid q, T^*, \overline{S^*})\big]}_{\text{Loss of Knowledge Recalling}} \underbrace{-\mathbb{E}\big[\log \pi_\theta(r \mid q, T^*, \overline{S^*}, i)\big]}_{\text{Loss of Contextualized Reasoning}}}_{\text{Frozen}} \tag{19}$$

We observe that R_ω explicitly learns to identify the latent $\mathcal{Z}$ from S^* for LLM-expected $\overline{S^*}$. During training with frozen LLM parameters, R_ω implicitly captures latent identification i by satisfying LLM's expectations: $Retriever(S^* \mid q, T^*)$ $R_\omega(\overline{S^*} \mid S^*, \mathcal{Z}) \approx \pi_\theta(\overline{S^*} \mid q, T^*, i)$, yielding $\mathcal{Z} \subseteq i$. Therefore, R_ω able to learns a subspace of the frozen LLM's complete latent space through this objective. Throughout this, we argue that leveraging a learned latent space $\mathcal{Z}$, robustly restructured into the LLM-expected representation $\overline{S^*}$, can directly improve knowledge recall and indirectly enhance reasoning.

5 Experiments

Datasets and Evaluation Metrics. Following previous work [32, 36, 40], we conduct experiments on ExplaGraphs [63] is a generative commonsense reasoning dataset, SceneGraphs [38] is a visual question answering dataset. And WebQSP [86], ComplexWebQuestions (CWQ) [68] is a large question-answering dataset derived from Web questions, where all queries can be answered using Freebase, a large collaborative knowledge graph database. We use accuracy as

the primary metric for ExplaGraphs and SceneGraphs, datasets focusing on reasoning, following [32,36,40]. For WebQSP and CWQ, a dataset with extra-large graphs, we use the Hit@1 metric, as in [50].
Implementation Details. We employed the open-source Llama3.2 (1B and 3B) [1], Llama 2 (7b and 13B) [73] and Llama3.1 (8B) [24] as frozen LLM components. Based on the analysis in 5.2, we set the sampling rate $\rho = 0.3$ (see Appendix 2.1 and 2.5 for cost–benefit trade-offs).

5.1 Main Results

Table 1 illustrates our main results, comparing the methods in three settings: **Frozen LLM with Graph Embedding**: Use a graph embedding technique to tune tokens with given prompt. **Frozen LLM with Graph Embedding + PEFT**: Apply LoRA [35], a PEFT technique, in combination with the graph embedding technique. **LLM with LLM-Retriever**: Use LLM for retrieval in addition to the one for inference. Any methods train either LLM. These are reference scores with a larger computational cost. Among **Frozen LLM with Graph Embedding** settings (with and without PEFT), **AGE consistently improved performance of G-Retriever and AMAR regardless of the backbone LLM models**. Without PEFT, Llama3.2-1B with AGE showed the most notable gain against G-Retiever: 26.72 percent points increase on ExplaGraphs, while the least gain was observed with Llama3.2-3B on WebQSP, which was 2.02 points. This might be due to the extra-large size of knowledge graphs in WebQSP datasets, which include textual knowledge absent at pretraining, and non-parameter retriever struggling to provide critical information for representation. AGE maintains consistent superiority against G-Retriever and shows more gains on retrieval from smaller graphs. By employing a cross-question approach enriched with retrieved elements, GRAG improved performance by 2.8 points, AMAR achieved an improvement of 4.2 points with its baseline. This suggests that improving embedding module is beneficial, it alone may not be enough to boost performance significantly, and relying on a non-parameter retriever could be limiting. Despite that challenge, when integrated with AMAR, AGE continues to achieve further enhancements the performance. Direct comparison with current **LLM with LLM Retriever** methods on WebQSP and the larger CWQ shows that AGE AMAR, which applies non-parametric retriever-based approaches, outperforms the proprietary LLM-based retriever ReKnoS [77] on both datasets. AGE AMAR underperforms compared with Paths-over-Graph [69], Plan-on-Graph [81] and DoG [51] on WebQSP, but outperforms them on CWQ, suggesting that AGE AMAR is advantageous on larger datasets, showing substantial potential for future work (as reported in the Appendix Table B.11, further including the more baseline methods.)

5.2 Ablation study

Performance Comparison of Self-Supervised Learning Architectures. Table 2 presents the performance of concept encoder-decoder with some varia-

Table 1: Performance comparison across ExplaGraphs, SceneGraphs, and WebQSP datasets under the five settings. The best and second-best scores are highlighted in **bold** and underline, respectively.

Setting	Method	LLM	Expla Graphs	Scene Graphs	WebQSP	CWQ
Frozen LLM w/ Graph Embedding	G-Retriever	Llama3.2-1B	0.5595	0.7540	60.1	–
	G-Retriever	Llama3.2-3B	0.7761	0.8229	71.3	–
	G-Retriever	Llama2-7B	0.8516	0.8131	68.1	–
	AGE G-Retriever	Llama3.2-1B	0.8267	0.8184	62.5	–
	AGE G-Retriever	Llama3.2-3B	0.9260	0.8930	73.5	–
	AGE G-Retriever	Llama3.1-8B	0.9350	0.9276	78.3	–
Frozen LLM w/ Graph Embedding + PEFT(LoRA)	G-Retriever	Llama3.2-1B	0.7328	0.8689	65.3	–
	G-Retriever	Llama3.2-3B	0.8339	0.9074	71.4	–
	G-Retriever	Llama2-7B	0.8705	0.8683	70.2	–
	AGE G-Retriever	Llama3.2-1B	0.8501	0.9056	69.1	–
	AGE G-Retriever	Llama3.2-3B	0.9134	**0.9486**	77.3	–
	AGE G-Retriever	Llama3.1-8B	**0.9612**	0.9325	80.3	–
	AMAR	Llama2-7B	–	–	84.3	82.9
	AMAR	Llama2-13B	–	–	83.3	83.1
	AGE AMAR	Llama2-7B	–	–	86.5	**85.2**
	AGE AMAR	Llama2-13B	–	–	86.2	85.1
LLM w/ LLM Retriever	ToG	GPT-4	–	–	82.6	67.6
	ReKnoS	GPT-4	–	–	84.9	68.2
	KG-Agent	Llama2-7B	–	–	83.3	72.2
	Paths-over-Graph	GPT-4	–	–	**96.7**	81.4
	Plan-on-Graph	GPT-4	–	–	87.3	75.0
	DoG	GPT-4	–	–	91.0	56.0

tions and the baseline of G-Retriever, which demonstrates contribution of proposed technique independently. As a SSL variation, we prepared AGE based on generative architecture (GA) against our choice of JEPA. We also compared AGE with random mask against the proposed learnable node sampler. All the results used Llama3.2 1B as its LLM. Using a GA with a random mask, AGE achieves a performance of 0.6532, which is a 9.37% improvement over the baseline. Next, AGE with a random mask, improving performance by 15.46% over the baseline. Finally, AGE using JEPA with the learnable node sampler achieves a 26.72% improvement over the baseline. Based on these experiments, we confirmed that JEPA works better than GA as expected, while node sampler further improves performance with a notable margin. Furthermore, we include an additional study on architectural design choices in the Appendix Table B.7, such as reason we choose different input features for LLMs during training and inference in GraphRAG.

Analysis on the Sampling Rate ρ. Figure 3 illustrates how the sampling rate impacts AGE G-Retriever performance on ExplaGraphs and WebQSP, guiding

Table 2: Performance improvements (Llama3.2 1B, ExplaGraphs; % points).

	G-Retriever	GA w Random mask	JEPA w Random mask	GA w Node sampler	JEPA w Node sampler
Loss	L_{PT}	$L_{\text{PT}} + L_{\text{target}}$	$L_{\text{PT}} + L_{\text{target}}$	$L_{\text{PT}} + L_{\text{target}} + L_{\text{NS}}$	$L_{\text{PT}} + L_{\text{target}} + L_{\text{NS}}$
Acc	0.5595	0.6532 (↑ 9.37%)	0.7141 (↑ 15.46%)	0.7870 (↑ 22.75%)	0.8267 (↑ 26.72%)

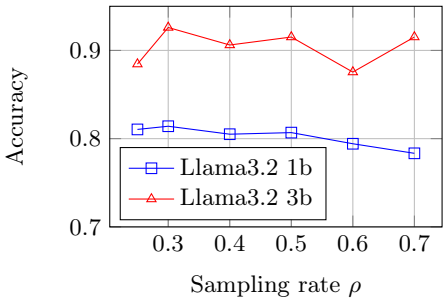


Fig. 3: Performance of against sampling rate (ExplaGraphs)

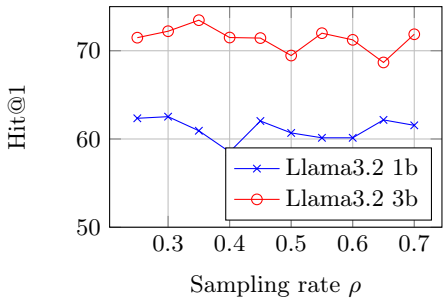


Fig. 4: Performance against sampling rate (WebQSP)

our hyper-parameter setting. Average retrieved nodes are 18.21 and 5.17 on WebQSP and ExplaGraphs, respectively. A sampling rate $\rho = 0.3$ gives the best performance on ExplaGraphs with both LLM settings (81.4% for Llma3.2-1B and 92.6% for Llama3.2-3B). The same setting also achieves the best performance on WebQSP for Llama3.2-1B (62.5%) and the second best for Llam3.2-3B (72.2%), compared to the best score of 73.5% at $\rho = 0.35$. From these observations, we decided to use $\rho = 0.3$ through the experiments.

Other LLM backbones. A Qwen3.5 result (Table R1) confirms the same trend. We will clarify the LoRA setting. AGE consistently outperforms G-Retriever across all scales and tasks, with especially large gains on ExplaGraphs. The improvements remain strong even at the smaller 0.8B scale, indicating efficiency and scalability. Gains on WebQSP further demonstrate the robustness of the proposed method across different tasks.

Weighting Loss. The three losses optimize disjoint parameters, so weighting is unnecessary by design; confirmed also in Table R2. Equal weighting (1:1:1) consistently yields the best performance across datasets and model sizes. Deviating from equal weights degrades performance, with no consistent benefit from emphasizing any single loss. This supports that the three losses operate on disjoint parameters, making explicit weighting unnecessary.

Qualitative Evaluation. To analyze node sampling results, we visualized the node sampling results on two samples from the ExplaGraphs test set in Figure 5. Nodes are colored based on clustered text embeddings to track node-wise feature restructuring through graph relationships. The graph encoder process maintains

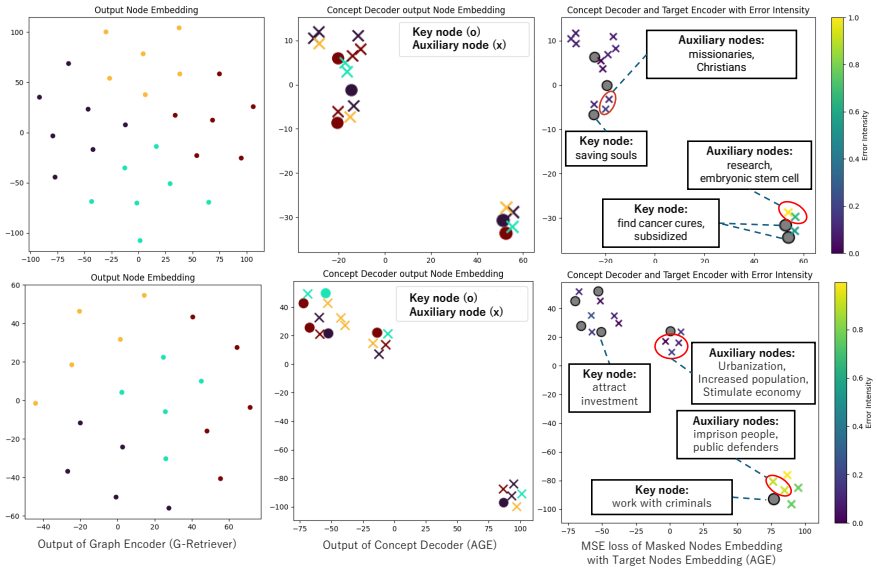


Fig. 5: Node embedding of G-Retriever and AGE G-Retriever with sampling rate $\rho = 0.3$ using t-SNE [53]: Nodes are colored by clustering node's text (left two), or target error (the rightest).

Table R1: The performance on Qwen3.5 family.

Method	Size	Frozen		LoRA	
		Expla Graphs	WebQSP	Expla Graphs	WebQSP
G-Retriever	0.8B	0.4832	59.5	0.7178	65.0
	2B	0.7477	72.8	0.8298	69.9
AGE (ours)	**0.8B**	**0.8089**	**61.7**	**0.8231**	**68.4**
	2B	**0.9101**	**73.6**	**0.9097**	**77.3**

the clustering structure of text graph embeddings in both G-Retriever (first column) and AGE (second column). In contrast, the concept encoder-decoder module (second column) shuffles the colored nodes, indicating a reorganization of the node-wise embeddings.

The last column displays text entities of some key and auxiliary nodes. Our node sampler is designed to sample entities from specific domains as key nodes. As "saving souls" is sampled as a key node, inferring "missionaries" and "Christians" from it seems easier than the reverse. We observe the same tendency with the key node "work with criminals" and the auxiliary nodes "imprison people" and "public defenders." The last column also shows the target loss for each auxiliary node using the color bar, where 1.0 represents the maximum error in the test set, and 0.0 the least. From the color visualization, we observe that non-isolated key nodes achieve lower errors in auxiliary node prediction, suggesting that relations between key nodes support the prediction. We provides additional qualitative results, including failure cases, in Appendix B.9.

Table R2: Loss-weight ablation (w_i on $L_{NS}, L_{target}, L_{PT}$).

$w_1 : w_2 : w_3$	ExplaGraphs		WebQSP	
	1B	**3B**	**1B**	**3B**
1.3 : 0.7 : 1	0.8087	0.8863	59.4	72.0
0.7 : 1.3 : 1	0.7834	0.8971	59.6	72.4
1 : 1 : 1	**0.8267**	**0.9260**	**62.5**	**73.5**

6 Limitation and Conclusion

Although our method delivers substantial improvements in GraphRAG, several limitations remain. First, we used a fixed sampling rate despite variations in key node density between the graphs. Second, we tested AGE only on GraphRAG tasks, even though it is applicable to other modalities. These limitations suggest areas for future improvement and the potential for broader applications. Third, our approach primarily targets small-scale models, and its effectiveness for large-scale LLMs remains unexplored due to computational constraints. Finally, our method focuses on representing retrieved structured data for LLMs rather than directly addressing graph learning tasks (e.g., node classification or link prediction). In the absence of theoretical guarantees on the benefits of node and link integration, our current scope is mainly limited to KGQA scenarios.

We proposed Adaptive-masking for Graph Embedding (AGE) to improve structured graph embeddings and enhance LLM performance on GraphQA tasks. The method introduced JEPA, a self-supervised learning architecture which enhanced the graph-structure embedding for downstream reasoning tasks. Our node sampler demonstrated its effectiveness in the ablation study, successfully identified key nodes within given graphs. The quantitative results confirmed AGE's consistent performance gain in GraphRAG tasks while maintaining computational cost. We hope this work contributes to structured knowledge representation for intelligent agents and facilitates cross-modal reasoning through structured perceptual representations.

References

1. AI, M.: Llama 3.2 connect 2024: Vision and edge for mobile devices (2024)
2. Anthropic: Model card and evaluations for claude models. Technical Report (2023), `https://www-cdn.anthropic.com/files/4zrzovbb/website/bd2a28d2535bfb0494cc8e2a3bf135d2e7523226.pdf`
3. Assran, M., Duval, Q., Misra, I., Bojanowski, P., Vincent, P., Rabbat, M., LeCun, Y., Ballas, N.: Self-supervised learning from images with a joint-embedding predictive architecture. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 15619–15629 (June 2023)
4. Baevski, A., Hsu, W.N., Xu, Q., Babu, A., Gu, J., Auli, M.: data2vec: A general framework for self-supervised learning in speech, vision and language. arXiv preprint arXiv:2202.03555 (2022)
5. Bardes, A., Garrido, Q., Ponce, J., Chen, X., Rabbat, M., LeCun, Y., Assran, M., Ballas, N.: Revisiting feature prediction for learning visual representations from video (2024)

6. Bardes, A., Ponce, J., LeCun, Y.: Vicreg: Variance-invariance-covariance regularization for self-supervised learning. arXiv preprint arXiv:2105.04906 (2021)
7. Bienstock, D., Goemans, M.X., Simchi-Levi, D., Williamson, D.: A note on the prize collecting traveling salesman problem. Mathematical Programming pp. 413–420 (1993)
8. Bizeul, A.: Masking Principal Components for Discriminative Self-Supervised Representation Learning. Master's thesis, ETH Zurich (2024), `https://mds.inf.ethz.ch/fileadmin/user_upload/principal_component_masking_for_self_supervised_learning.pdf`
9. Cao, Y., Gao, Z., Li, Z., Xie, X., Zhou, K., Xu, J.: Lego-graphrag: Modularizing graph-based retrieval-augmented generation for design space exploration. arXiv preprint arXiv:2411.05844 (2025)
10. Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. arXiv preprint arXiv:2006.09882 (2020)
11. Chen, D., Hu, J., Wei, X., Wu, E.: Denoising with a joint-embedding predictive architecture. In: The Thirteenth International Conference on Learning Representations (2025)
12. Chen, L., Tong, P., Jin, Z., Sun, Y., Ye, J., Xiong, H.: Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs. arXiv preprint arXiv:2410.23875 (2024)
13. Chen, R., Zhao, T., Jaiswal, A., Shah, N., Wang, Z.: Llaga: Large language and graph assistant. arXiv preprint arXiv:2402.08170 (2024)
14. Chien, E., Chang, W.C., Hsieh, C.J., Yu, H.F., Zhang, J., Milenkovic, O., Dhillon, I.S.: Node feature extraction by self-supervised multi-scale neighborhood prediction. In: International Conference on Learning Representations (ICLR) (2022)
15. Chin, Z.Y., Jiang, C.M., Huang, C.C., Chen, P.Y., Chiu, W.C.: Masking improves contrastive self-supervised learning for convnets, and saliency tells you where. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 2761–2770 (2024)
16. Cox, C.R., Rogers, T.T., Shimotake, A., Kikuchi: Representational similarity learning reveals a graded multidimensional semantic space in the human anterior temporal cortex. Imaging Neuroscience **2**, 1–22 (2024)
17. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
18. Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Metropolitansky, D., Ness, R.O., Larson, J.: From local to global: A graph rag approach to query-focused summarization. arXiv preprint arXiv:2404.16130 (2024)
19. Edmonds, M., Qi, S., Zhu, Y., Kubricht, J., Zhu, S.C., Lu, H.: Decomposing human causal learning: Bottom-up associative learning and top-down schema reasoning. In: Proceedings of the Annual Meeting of the Cognitive Science Society (2019)
20. Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.S., Li, Q.: A survey on rag meeting llms: Towards retrieval-augmented large language models. arXiv preprint arXiv:2405.06211 (2024)
21. Fatemi, B., Halcrow, J., Perozzi, B.: Talk like a graph: Encoding graphs for large language models. In: The Twelfth International Conference on Learning Representations (2024)
22. Fei, Z., Fan, M., Huang, J.: A-jepa: Joint-embedding predictive architecture can listen. arXiv preprint arXiv:2311.15830 (2023)

23. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., Wang, H.: Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997 (2024)
24. Grattafiori, A., Dubey, A., Abhinav Jauhri, .: The llama 3 herd of models (2024)
25. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P.H., Buchatskaya, E., Doersch, C., Pires, B.A., Guo, Z.D., Azar, M.G., et al.: Bootstrap your own latent: A new approach to self-supervised learning. arXiv preprint arXiv:2006.07733 (2020)
26. Guo, J., Du, L., Liu, H., Zhou, M., He, X., Han, S.: Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. arXiv preprint arXiv:2305.15066 (2023)
27. Guo, K., Shomer, H., Zeng, S., Han, H., Wang, Y., Tang, J.: Empowering graphrag with knowledge filtering and integration. arXiv preprint arXiv:2503.13804 (2025)
28. Han, H., Shomer, H., Wang, Y., Lei, Y., Guo, K., Hua, Z., Long, B., Liu, H., Tang, J.: Rag vs. graphrag: A systematic evaluation and key insights. arXiv preprint arXiv:2502.11371 (2025)
29. Han, H., Wang, Y., Shomer, H., Guo, K., Ding, J., Lei, Y., Halappanavar, M., Rossi, R.A., Mukherjee, S., Tang, X., He, Q., Hua, Z., Long, B., Zhao, T., Shah, N., Javari, A., Xia, Y., Tang, J.: Retrieval-augmented generation with graphs (graphrag). arXiv preprint arXiv:2501.00309 (2025)
30. He, G., Lan, Y., Jiang, J., Zhao, W.X., Wen, J.R.: Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining. p. 553 561. WSDM '21, Association for Computing Machinery (2021)
31. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. arXiv:2111.06377 (2021)
32. He, X., Tian, Y., Sun, Y., Chawla, N.V., Laurent, T., LeCun, Y., Bresson, X., Hooi, B.: G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. CoRR (2024)
33. Hou, Z., He, Y., Cen, Y., Liu, X., Dong, Y., Kharlamov, E., Tang, J.: Graphmae2: A decoding-enhanced masked self-supervised graph learner. In: Proceedings of the ACM Web Conference 2023. pp. 737–746 (2023)
34. Hou, Z., Liu, X., Cen, Y., Dong, Y., Yang, H., Wang, C., Tang, J.: Graphmae: Self-supervised masked graph autoencoders. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 594–604 (2022)
35. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
36. Hu, Y., Lei, Z., Zhang, Z., Pan, B., Ling, C., Zhao, L.: Grag: Graph retrieval-augmented generation. arXiv preprint arXiv:2405.16506 (2024)
37. Huang, J., Zhang, X., Mei, Q., Ma, J.: Can llms effectively leverage graph structural information: when and why. arXiv preprint arXiv:2309.16595 (2023)
38. Hudson, D.A., Manning, C.D.: Gqa: A new dataset for real-world visual reasoning and compositional question answering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6700–6709 (2019)
39. Jackson, R.L., Bajada, C.J., Rice, G.E., Cloutman, L.L., Lambon Ralph, M.A.: An emergent functional parcellation of the temporal cortex. NeuroImage **170**, 385–399 (2018)
40. Ji, Y., Liu, C., Chen, X., Ding, Y., Luo, D., Li, M., Lin, W., Lu, H.: Nt-llm: A novel node tokenizer for integrating graph structure into large language models. arXiv preprint arXiv:2410.10743 (2024)

41. Jin, B., Xie, C., Zhang, J., Roy, K.K., Zhang, Y., Li, Z., Li, R., Tang, X., Wang, S., Meng, Y., Han, J.: Graph chain-of-thought: Augmenting large language models by reasoning on graphs. arXiv preprint arXiv:2404.07103 (2024)
42. Jing, L., Vincent, P., LeCun, Y., Tian, Y.: Understanding dimensional collapse in contrastive self-supervised learning. arXiv preprint arXiv:2110.09348 (2021)
43. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks (2016)
44. Kramer, O.: K-nearest neighbors. In: Dimensionality Reduction with Unsupervised Nearest Neighbors, Intelligent Systems Reference Library, vol. 51, pp. 13–23. Springer, Berlin, Heidelberg (2013)
45. LeCun, Y., Courant: A path towards autonomous machine intelligence version 0.9.2, 2022-06-27 (2022)
46. Lee, C., Oh, J., Lee, K., yong Sohn, J.: A theoretical framework for preventing class collapse in supervised contrastive learning. arXiv preprint arXiv:2503.08203 (2025)
47. Li, Y., Zhang, R., Liu, J.: An enhanced prompt-based llm reasoning scheme via knowledge graph-integrated collaboration. arXiv preprint arXiv:2402.04978 (2024)
48. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
49. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization (2017)
50. Luo, L., Li, Y.F., Haffari, G., Pan, S.: Reasoning on graphs: Faithful and interpretable large language model reasoning. arXiv preprint arXiv:2310.01061 (2024)
51. Ma, J., Gao, Z., Chai, Q., Sun, W., Wang, P., Pei, H., Tao, J., Song, L., Liu, J., Zhang, C., Cui, L.: Debate on graph: A flexible and reliable reasoning framework for large language models. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 24768–24776 (2025)
52. Ma, L., Rabbany, R., Romero-Soriano, A.: Graph attention networks with positional embeddings (2021)
53. van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**, 2579–2605 (2008)
54. Mavromatis, C., Karypis, G.: ReaRev: Adaptive reasoning for question answering over knowledge graphs. In: Findings of the Association for Computational Linguistics: EMNLP 2022. pp. 2447–2458. Association for Computational Linguistics (2022)
55. Mavromatis, C., Karypis, G.: Gnn-rag: Graph neural retrieval for large language model reasoning. arXiv preprint arXiv:2405.20139 (2024)
56. Mittal, S., Lamb, A., Goyal, A., Voleti, V., Shanahan, M., Lajoie, G., Mozer, M., Bengio, Y.: Learning to combine top-down and bottom-up signals in recurrent neural networks with attention over modules. arXiv preprint arXiv:2006.16981 (2020)
57. OpenAI: New embedding models and api updates. `https://openai.com/blog/new-embedding-models/` (2022)
58. OpenAI: Gpt-4 technical report (2024)
59. OpenAI: Openai gpt-5 system card. arXiv preprint arXiv:2601.03267 (2025), `https://arxiv.org/abs/2601.03267`
60. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical Report SIDL-WP-1999-0120, Stanford Digital Library Technologies Project (1998)
61. Perozzi, B., Fatemi, B., Zelle, D., Tsitsulin, A., Kazemi, M., Al-Rfou, R., Halcrow, J.: Let your graph do the talking: Encoding structured data for llms (2024)

62. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. pp. 3982–3992. Association for Computational Linguistics (2019)
63. Saha, S., Yadav, P., Bauer, L., Bansal, M.: Explagraphs: An explanation graph generation task for structured commonsense reasoning. arXiv preprint arXiv:2104.07644 (2021)
64. Seong, J., Han, H.: Rethinking random masking in self-distillation on vit. arXiv preprint arXiv:2506.10582 (2025)
65. Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., Sun, Y.: Masked label prediction: Unified message passing model for semi-supervised classification (2020)
66. Sun, J., Xu, C., Tang, L., Wang, S., Lin, C., Gong, Y., Ni, L.M., Shum, H.Y., Guo, J.: Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. arXiv preprint arXiv:2307.07697 (2024)
67. Sutton, R.S., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 1057–1063 (2000)
68. Talmor, A., Berant, J.: The web as a knowledge base for answering complex questions. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 641–651. Association for Computational Linguistics (2018)
69. Tan, X., Wang, X., Liu, Q., Xu, X., Yuan, X., Zhang, W.: Paths-over-graph: Knowledge graph empowered large language model reasoning. In: Proceedings of the ACM Web Conference 2025. pp. 3505–3522 (2025)
70. Tang, A.C., Sutherland, M.T., Sun, P., Zhang, Y., Nakazawa, M., Korzekwa, A., Yang, Z., Ding, M.: Top-down versus bottom-up processing in the human brain: Distinct directional influences revealed by integrating sobi and granger causality. In: Independent Component Analysis and Signal Separation. pp. 802–809. Springer (2007)
71. Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A.M., Hauth, A., et al.: Gemini: a family of highly capable multimodal models (2023)
72. Theves, S., Neville, D.A., Fernández, G., Doeller, C.F.: Learning and representation of hierarchical concepts in hippocampus and prefrontal cortex. Journal of Neuroscience pp. 7675–7686 (2021)
73. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
74. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks (2017)
75. Wang, D., Zuo, Y., Li, F., Wu, J.: Llms as zero-shot graph learners: Alignment of gnn representations with llm token embeddings. arXiv preprint arXiv:2408.14512 (2024)
76. Wang, H., Feng, S., He, T., Tan, Z., Han, X., Tsvetkov, Y.: Can language models solve graph problems in natural language? arXiv preprint arXiv:2305.10037 (2023)
77. Wang, S., Lin, J., Guo, X., Shun, J., Li, J., Zhu, Y.: Reasoning of large language models over knowledge graphs with super-relations. In: The Thirteenth International Conference on Learning Representations (2025)

78. Wang, Z., Yu, J., Ma, D., Chen, Z., Wang, Y., Li, Z., Xiong, F., Wang, Y., E, W., Tang, L., Zhang, W.: Rare: Retrieval-augmented reasoning modeling. arXiv preprint arXiv:2503.23513 (2025)
79. Wei, C., Fan, H., Xie, S., Wu, C.Y., Yuille, A., Feichtenhofer, C.: Masked feature prediction for self-supervised visual pre-training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 14668–14678 (2022)
80. Wu, S., Xiong, Y., Cui, Y., Wu, H., Chen, C., Yuan, Y., Huang, L., Liu, X., Kuo, T.W., Guan, N., Xue, C.J.: Retrieval-augmented generation for natural language processing: A survey. arXiv preprint arXiv:2407.13193 (2024)
81. Wu, X., Shen, Y., Shan, C., Song, K., Wang, S., Zhang, B., Feng, J., Cheng, H., Chen, W., Xiong, Y., Li, D.: Can graph learning improve planning in llm-based agents? In: Advances in Neural Information Processing Systems 38 (NeurIPS 2024) (2024)
82. Xu, D., Li, X., Zhang, Z., Lin, Z., Zhu, Z., Zheng, Z., Wu, X., Zhao, X., Xu, T., Chen, E.: Harnessing large language models for knowledge graph question answering via adaptive multi-aspect retrieval-augmentation. In: Proceedings of the AAAI Conference on Artificial Intelligence (2025), arXiv preprint arXiv:2412.18537
83. Yang, A., Li, A., Yang, B., Zhang, B., et al.: Qwen3 technical report. arXiv preprint arXiv:2505.09388 (2025), `https://arxiv.org/abs/2505.09388`
84. Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., Narasimhan, K.: Tree of thoughts: Deliberate problem solving with large language models. Advances in Neural Information Processing Systems **36** (2024)
85. Yasunaga, M., Ren, H., Bosselut, A., Liang, P., Leskovec, J.: Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2021). pp. 535–546 (2021)
86. tau Yih, W., Richardson, M., Meek, C., Chang, M.W., Suh, J.: The value of semantic parse labeling for knowledge base question answering. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 201–206 (2016)
87. Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., Liu, T.Y.: Do transformers really perform badly for graph representation? In: Advances in Neural Information Processing Systems. pp. 28877–28888 (2021)
88. You, Y., Chen, T., Shen, Y., Wang, Z.: Graph contrastive learning automated. In: Proceedings of the International Conference on Machine Learning (ICML) (2021)
89. Zeng, Q., Yang, Q., Dong, S., Du, H., Zheng, L., Xu, F., Li, Y.: Perceive, reflect, and plan: Designing llm agent for goal-directed city navigation without instructions. arXiv preprint arXiv:2408.04168 (2024)
90. Zheng, K., Yang, J., Liang, S., Feng, B., Liu, Z., Ju, W., Xiao, Z., Zhang, M.: Exlm: Rethinking the impact of [mask] tokens in masked language models. arXiv preprint arXiv:2501.13397 (2025)
91. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Graph contrastive learning with adaptive augmentation. In: Proceedings of the Web Conference 2021. pp. 2069–2080 (2021)

Appendix
AGE: Adaptive-masking for Graph Embedding in Graph Retrieval-Augmented Generation

Bao Long Nguyen Huu[1] and Atsushi Hashimoto[2]

[1] OMRON Corporation
[2] OMRON SINIC X Corporation

1 Proof of Concept

In this section, we motivate AGE's design for representing retrieved subgraphs for higher-order reasoning skills. We assume this requires on the most common static search single-turn retrieval and LLM for academic simplicity.
Problem definition: Given input query q, we generate chain-of-thoughts response $y = i \oplus r$ (interleaved domain knowledge i and reasoning steps r). We use a static retrieval engine that returns T^* for text knowledge and S^* for subgraph. The learning objective $\min_\theta \mathbb{E}[\mathcal{L}]$ optimizes representation S^* to prioritize reasoning r over knowledge i.
For the chain-of-thoughts response is defined as $y = i \oplus r$ where y is the concatenation of knowledge i and reasoning r through three discrete generation processes below.

- **Graph Knowledge Retrieval:** Given the query q on a textual graph G and $S(G)$ is the set of all subgraphs of G. Retrieval system extracts relevant subgraph $S^* \in S(G)$ and text-modal knowledge T^*. Popular retrieval systems select top-k elements by cosine similarity, yielding nodes V_k^* and edges E_k^* considered relevant to the query. A non-optimized retriever may yield corrupted subgraphs $S^* = (V_k^*, E_k^*)$, as they may be contain redundant or lack suggestive elements.
- **Graph Knowledge Representation:** Graph embedding module is trained to represent graph that guides the LLM to produce expected answers. Embedding module $R_\omega(\cdot)$ learn to represent corrupted subgraph S^* to $\overline{S^*}$ for an LLMΦ improve generation.
- **Contextualized Reasoning:** Given q, T^* and $\overline{S^*}$, LLM synthesizes domain knowledge t by recall their internal parametric knowledge with external inputs, following the conditional distribution $i \sim \pi_\theta(i|q, T^*, \overline{S^*})$. Then LLM generates reasoning steps r conditioned on q, T^*, $\overline{S^*}$ with the recalled internal knowledge i, adhering to the reasoning distribution $r \sim \pi_\theta(r|q, T^*, \overline{S^*}, i)$.

Here we formally analyze and discuss the subgraph representation learning objectives of both vanilla embedding module and AGE embedding module with LLM generation distribution.

– Subgraph embedding module that employ GNN or Transformer:

$$\overline{S^*} = \mathrm{R}_{\omega}^{\mathrm{GNN}}(S^*) = \left\{ \overline{V}_i \right\}_{i \in V^*} \tag{1}$$

$$\overline{V}_i = \sum_{j \in \mathcal{N}(i)} \alpha_{ij} \, W_V V_j, \tag{2}$$

Here, $\mathcal{N}(i)$ denotes the local neighborhood of node i (e.g., $\mathcal{N}(i) = \{j \mid (i,j) \in E^*\}$).

$$\overline{S^*} = \mathrm{R}_{\omega}^{\mathrm{Transformer}}(S^*) = \left\{ \overline{V}_i \right\}_{i \in V^*} \tag{3}$$

$$\overline{V}_i = \sum_{j \in V^*} \alpha_{ij} \, W_V V_j, \qquad \alpha_{ij} = \mathrm{softmax}_{j \in V^*}\Big(\tfrac{(W_Q V_i)^\top (W_K V_j)}{\sqrt{d_k}} \Big), \tag{4}$$

where W_Q, W_K, W_V are learnable matrices, d_k is the key dimension. These formulations use attention-weighted aggregation as node embeddings. Static retrieval from graphs with high-order structural patterns (motifs/role patterns) produces corrupted subgraphs that contain redundant nodes or miss critical elements. Without explicit structural constraints, training weighted-sum aggregation through supervised or semi-supervised learning separates signal from noise, may lead to diluted node representations within the embedding space. In a frozen state, LLMs may fail to capture relationships in the embedding space due to their struggle to handle diluted node representations.

– Subgraph embedding module that employ RL-guide mask-based SSL:

$$\overline{S^*} = R\omega^{\mathrm{AGE}}(S^*) = \left\{ \overline{V}_i \right\}_{i \in V^*} = Decoder_{\phi}\big(\Delta_{V_{masked}}, Encoder_{\theta}(V_{key})\big) \tag{5}$$

$$= V_{key} \sim \mathrm{Sampler}_{\theta}(V_{key} \mid V^*) \Bigg[\sum_{j \in V^*} \alpha_{ij} W_V \Big[\Big[\sum_{j \in V^{key}} \alpha_{ij} \, W_V V_j \Big]; \Delta_{V_{masked}} \Big] \Bigg] \tag{6}$$

$\Delta_{V_{masked}}$ denotes the auxiliary masked node features used as decoder input, while V_{key} denotes the key node features used as encoder input and $[;]$ is the concatenation operation.

$$\begin{cases} \Delta_{V_{masked}} = Mask(V_j) & \text{if } a_j = 1 \\ V_{key} = Visible(V_j) & \text{if } a_j = 0 \end{cases} \quad \sum_{j \in V^*} \mathrm{Sampler}_{\theta}(a_j \mid V_j), \tag{7}$$

where V_{key} represents key node features for encoder input, where $\mathrm{Mask}(\cdot)$ masks features, $\mathrm{Visible}(\cdot)$ preserves them, and $a_j \in \{0, 1\}$ denotes the binary RL action (1=mask, 0=keep) for node j.

By employing mask-based SSL alongside a reinforcement learning framework, AGE learns structural dependency node representations in the embedding space through reconstruction objectives. The reinforcement learning framework is used to estimate which nodes are critical for preserving

graph structure and semantic information. Then, the estimated nodes are applied to guide mask-based SSL to reconstruct that provide structural constraints in the embedding space, enabling LLMs to better separate signals and capture relationships within it.

– The joint generation distribution of LLMs is:

$$\begin{aligned}\pi_\theta(y \mid q, T^*, \overline{S^*}) &= \pi_\theta(i \oplus r \mid q, T^*, \overline{S^*}) \\ &= \underbrace{\pi_\theta(i \mid q, T^*, \overline{S^*})}_{\text{Knowledge Recalling}} \cdot \underbrace{\pi_\theta(r \mid q, T^*, \overline{S^*}, i)}_{\text{Contextualized Reasoning}} ,\end{aligned} \tag{8}$$

– The loss function optimizes both knowledge integration and contextualized reasoning:

$$\begin{aligned}\mathcal{L} &= -\mathbb{E}_{(q,T^*,\overline{S^*})}\left[\log \pi_\theta(i \mid q, T^*, \overline{S^*})\, \pi_\theta(r \mid q, T^*, \overline{S^*}, i)\right] \\ &= -\mathbb{E}_{(q,T^*,\overline{S^*})}\left[\log \pi_\theta(i \mid q, T^*, \overline{S^*})\right] \; - \mathbb{E}_{(q,T^*,\overline{S^*})}\left[\log \pi_\theta(r \mid q, T^*, \overline{S^*}, i)\right] ,\end{aligned} \tag{9}$$

Through the lens of multi-task learning, we compare above equations and from two perspectives:

– **Retrieved Subgraph Representation on LLMs' generation distribution.** Based on equation equation 8, LLMs decompose response generation into knowledge recall and contextualized reasoning. In the frozen state, diluted subgraph representations trigger a cascade—weak knowledge recall causes flawed reasoning. This limitation creates a bottleneck that degrades response quality.

– **Retrieved Subgraph Representation with Parameter-Efficient Fine-Tuning.**
Following previous research [78], we assume the retrieved representation is **explicitly**, we have equation 9 as:

$$\mathcal{L} = \underbrace{-\mathbb{E}_{(q,T^*,\overline{S^*})}\left[\log \pi_\theta(i \mid q, T^*, \overline{S^*})\right]}_{\textbf{Loss of Integration}} \downarrow \; \underbrace{-\mathbb{E}_{(q,T^*,\overline{S^*})}\left[\log \pi_\theta(r \mid q, T^*, \overline{S^*}, i)\right]}_{\text{Loss of Reasoning}} \uparrow , \tag{10}$$

The arrows indicate the loss function shifts to reasoning. That means the loss term shifts from knowledge identification to integration, that $\pi_\theta(i \mid q, T^*, \overline{S^*})$ has already reached "application" levels of retrieved graph knowledge. Therefore, explicit subgraph representation aids knowledge integration beyond mere identification during fine-tuning.
Conversely, the retrieved subgraph representation is **diluted**, we have equation 9 as:

$$\mathcal{L} = \underbrace{-\mathbb{E}_{(q,T^*,\overline{S^*})}\left[\log \pi_\theta(i \mid q, T^*, \overline{S^*})\right]}_{\textbf{Loss of Identification}} \uparrow \; \underbrace{-\mathbb{E}_{(q,T^*,\overline{S^*})}\left[\log \pi_\theta(r \mid q, T^*, \overline{S^*}, i)\right]}_{\text{Loss of Reasoning}} \downarrow , \tag{11}$$

The arrows indicate that the loss function prioritizes identification. This reveals inefficient knowledge use: the model identifies patterns directly, treating retrieved subgraphs as training data. This creates a trade-off: instead of learning to apply retrieved subgraphs in reasoning, $\pi_\theta(i \mid q, T^*, \overline{S^*})$ prioritizes identifying graph structures over reasoning. Poor retrieved subgraph representation implicitly hinders reasoning capability development, forcing resources into identification tasks that better representations would cover.

2 Additional Experimental Details

2.1 Implementation Settings (AGE G-Retriever)

When integrated with G-Retriever, we consistently use the AdamW [49] optimizer and set the initial learning rate at $1e-4$, with a weight decay of 0.05. Following the baseline work [32], we set learning rate decays with a half-cycle cosine decay after the warm-up period. To avoid overfitting, we implement early stopping with a patience of 3 epochs. The experiments used 2 NVIDIA 2080Ti-11G or 2 NVIDIA A100-80G GPUs.

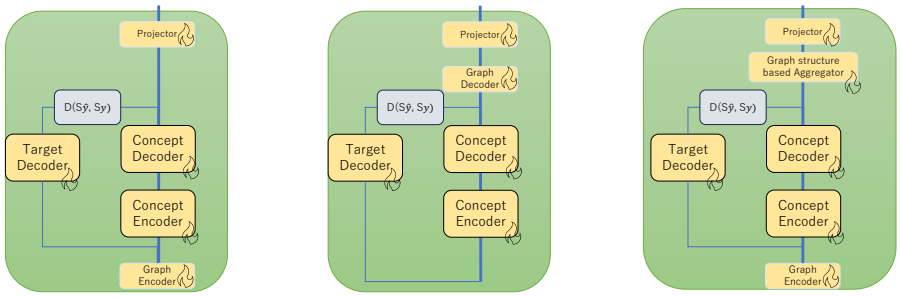
GNN. We use Graph Transformer as the GNN backbone applied in the Graph Encoder and Graph Structure Based Aggregator. Similar to previous approaches [32], our settings for its employ 2 layers, each with 4 attention heads, and a hidden dimension size of 1024.

LLM. We use the open-source Llama3.2 1B, 3B [1], and Llama3.1 8B as the LLM backbone. When LoRA [35] is applied with the LLM, the LoRA scaling factor hyperparameter is set to 16. Following previous work [32], we configure the LLM with a maximum input text length of 512 and a maximum number of new tokens to generate of 32.

Subgraph Construction. We follow previous approaches [32] that select the top k nodes and edges through subgraph construction by setting k to 3 for SceneGraphs dataset. For WebQSP dataset, $k = 3$ for nodes and $k = 5$ for edges. For the ExplaGraphs dataset, the entire graph fits within the LLM's context window. Thus, setting k to 0 for retrieves the original graph without modification.

2.2 Implementation Settings (AGE AMAR)

When integrated with AMAR [82], to fairly compare we keep the training settings of AMAR, setting the retrieved data to 100 on WebQSP, Soft prompt length to 7, Beam search number to 8, and Max new tokens to 256 for the WebQSP dataset. For the CWQ dataset, we set the retrieved data to 4, Soft prompt length to 16, Beam search number to 15, and Max new tokens to 256. With the Llama2 [73] is trained with LoRA learning rate $5e-5$ scaling factor hyperparameter is set to 32.



(a) w/o Graph structure based Aggregator

(b) w/o Graph Encoder

(c) w Graph Encoder + Graph-structure-based Aggregator

Fig. B.1: Investigation of core component arrangement: We tested our JEPA [45] architecture with three different GNN arrangements, including (a) graph encoder only, (b) graph-structure-based aggregator only, and (c) both of them.

2.3 The choice of AGE architecture

The choice of graph structure extractor architecture on AGE G-Retriever Figure B.1 shows the architectures with graph encoder only, graph-structure-based aggregator only, and both combined. The best-performing architecture is the combination of graph encoder and graph-structure-based aggregator. It achieves a Hit@1 score of 73.46% on the WebQSP dataset, improving upon 71.12% with Graph Encoder and 72.44% with graph-structure-based aggregator.

GNNs	WebQSP	ExplaGraphs
GCN	56.75	0.8321
GAT	61.42	0.8212
Graph Transformer	62.53	0.8501

Table B.1: Performance comparison of different GNNs on Llama3.2 1B.

The Choice of GNN on AGE G-Retriever In Table B.1, our investigation extends to existing popular GNNs employed as both graph encoders and graph-structure-based Aggregator, including the Graph Convolutional Network (GCN) [43], Graph Attention Network (GAT) [74] and Graph Transformer [65]. This illustrates the performance comparison of these GNNs on the WebQSP and ExplaGraphs datasets. On the WebQSP dataset, the GCN, GAT, and Graph Transformer achieve Hit@1 scores of 56.75, 61.42, and 62.53, respectively. In the

ExplaGraphs dataset, the Graph Transformer achieves the highest accuracy of 0.8501, followed by the GCN with an accuracy of 0.8321, and the GAT trailing slightly at 0.8212.
These findings emphasize the critical role of selecting an appropriate GNN architecture tailored to the unique properties and demands of each dataset. To maintain performance across various datasets, we choose the Graph Transformer for all experiments.

The design of AGE with Generation Architecture As shown in Figure B.2, we provide more details of AGE with the randomly masked generative architecture (GA) introduced in Section 4.3. Designing AGE with GA aims to complement the input node to enhance the embedding of the graph-structure-based aggregator during the inference stage. To do this, we train the encoder-decoder with input nodes masked by a random mask at a masking ratio of 70%.Then, the encoder is trained to embed unmasked nodes, and the decoder reconstructs masked nodes through the target loss. On the other hand, prompt tuning loss is used to train the graph-structure-based aggregator and MLP for referring to edges E^*, and the MLP adjusts the aggregated embeddings to align with the LLM input dimension. During the inference stage, random masking is disabled. All input nodes are fed into the encoder-decoder to reconstruct the input for the graph-structure-based aggregator.

Table B.2: Analysis on the number of GNN_{ge}' layers with LLaMA 3.2 3B on WebQSP. GE refers to Graph Embedding.

	PT w/o GE	G-Retriever		AGE		
# of layers	-	2	4	1	2	4
Hit@1 (↑)	48.3	64.9	71.3	73.5	70.5	69.7
Training time (Min./Epoch) (↓)	4.5	4.4	4.5	4.5	4.6	4.9
Inference speed (Tokens/sec) (↑)	88.9	86.0	84.4	87.6	84.9	81.1

Analysis on the Layer Number of the Graph Encoder GNN_{ge} on AGE G-Retriever Compared to G-Retriever, AGE's inference path has additional modules. While this might increase processing time, Table B.2 indicates otherwise. For G-Retrievers, a deeper GNN_{ge} performs better. In contrast, AGE performs better with fewer layers, as the added modules effectively substitute for reduced GNN layers. As a result, AGE achieves superior performance while maintaining the training time of the baseline method. We provide further analysis on computational complexity in Appendix.

	LLM	GNN Layer	Parameter	FLOPs (G)	Acc
G-Retriever	Llama 3.2 1B	4	3.9 M	0.2 G	0.5595
G-Retriever	Llama 3.2 1B	20	11.3M	1.2 G	0.7238
AGE G-Retriever	Llama 3.2 1B	2	7.8 M	1.1 G	0.8501
G-Retriever	Llama 3.2 1B	4	3.9 M	0.2 G	0.7761
G-Retriever	Llama 3.2 1B	20	11.3M	1.2 G	0.8682
AGE G-Retriever	Llama 3.2 1B	2	7.8 M	1.1 G	0.9260

Table B.3: Compare AGE with DeeperGNN in ExplaGraphs test set

Decoder Depth	Para. (M)	Expla (Acc)	WebQSP (Hit@1)
1	81	0.8501	62.5
2	85	0.7978	61.2
4	106	0.8123	57.4

(a) Concept Decoder

Depth	d	FLOPs (G)	Expla (Acc)	WebQSP (Hit@1)
1	1024	1.1	0.8501	62.5
1	2048	1.6	0.8213	59.3
2	1024	1.4	0.7906	63.1

(b) Node Sampler

Table B.4: Ablation studies for network architecture design.

Comparison with Deeper GNNs Table B.3 compares the number of GNN layers and the performance of G-Retriever and Adaptive-masking for Graph Embedding models. G-Retriever with 20 layers is prepared as a model whose computational cost (GFLOPs) is similar to AGE with 2 layers.
Applying the G-Retriever with 20 layers largely improves performance. However, AGE still outperforms G-Retriever by approximately 10 points when using Llama 3.2 1B and 6 points when using Llama 3.2 3B, demonstrating AGE's superior performance.

The choice of concept decoder and node sampler architecture. Table B.4a illustrates our analysis of model performance across various concept decoder depths. We increased the decoder depth from 1 block to 4 blocks, thereby increasing the parameters from 81M to 106M. Despite this increase, the performance decreased, with scores dropping from 62.5 to 57.4 on WebQSP. The best performance is achieved with a decoder depth of 1 on both ExplaGraphs and WebQSP. Therefore, we choose a single transformer block to maintain the performance of the concept decoder in this work.
As shown in Table B.4b, we investigate different network architectures for the node sampler design. Increasing the number of transformer blocks leads to marginal gains in performance on the WebQSP dataset, although it requires more memory. To maintain computational and performance efficiency, we selected a single transformer block with a hidden dimension of 1024 for the node sampler in all subsequent experiments.

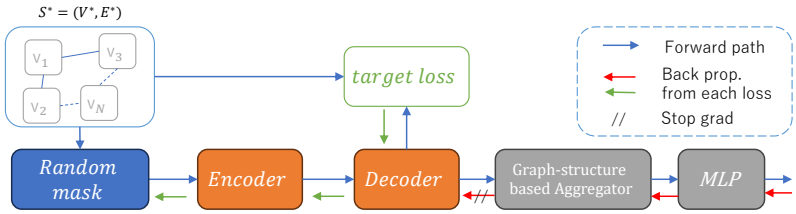


Fig. B.2: Adaptive-masking for Graph Embedding in Generation Architecture: The node embedding module is trained on both prompt tuning loss and target loss.

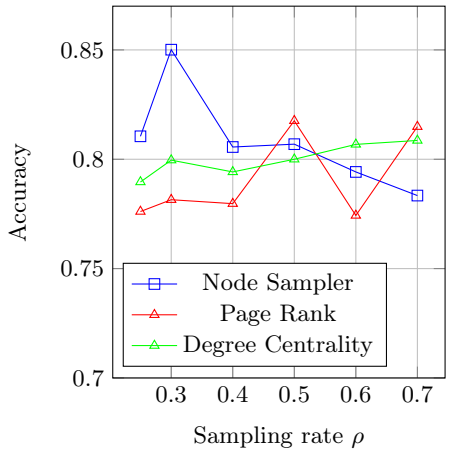


Fig. B.3: Relationship of sampling rate with key node sampling strategy (on ExplaGraphs)

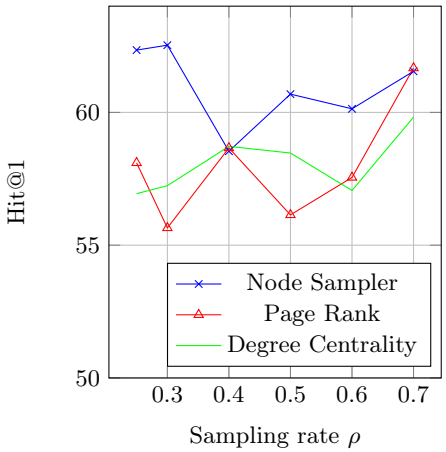


Fig. B.4: Relationship of sampling rate with key node sampling strategy (on WebQSP)

The study on key nodes sampling Figure B.3 and B.4 illustrates how the static strategy node sampler and RL-based node sampler performance in various sampling rate on ExplaGraphs and WebQSP, guiding our node sampler architecture setting.

When using static PageRank [60] and Degree Centrality strategies for node sampling, higher sampling rates tend to better performance. However, this suggests that the key nodes that identified by these static methods are not sufficiently impactful. Lead to the Concept Encoder-Decoder needs a larger set of key nodes to effectively embed the graph, which then helps guide the LLM to produce the desired answers.

In contrast, RL-based node samplers can achieve high performance with lower sampling rates. This indicates that the key nodes chosen by RL-based methods more effectively support the Concept Encoder-Decoder, boosting the quality of

the graph embedding. As a result, the LLM can produce the expected answers with fewer key nodes involved in the guidance process.

Method	**WebQSP → Expla**	**Expla → WebQSP**	
G-Retriever	Llama 3.2 1B	0.5106	36.48
AGE G-Retriever	Llama 3.2 1B	0.5685	39.25
G-Retriever	Llama 3.2 3B	0.4404	50.35
AGE G-Retriever	Llama 3.2 3B	0.6021	53.53

Table B.5: Cross-Dataset Transfer Learning Performance.

The study on transferability Table B.5 show the transferability of AGE when interacted with G-Retriever. AGE support G-Retriever to strong transferability to transfer learned graph embedding encoding capabilities across datasets. When trained on a large dataset, AGE can enhance generation on a smaller dataset using the trained model. Notably, AGE trained on WebQSP on ExplaGraphs with Llama 3.2 3B outperforms transferability of GRAG.

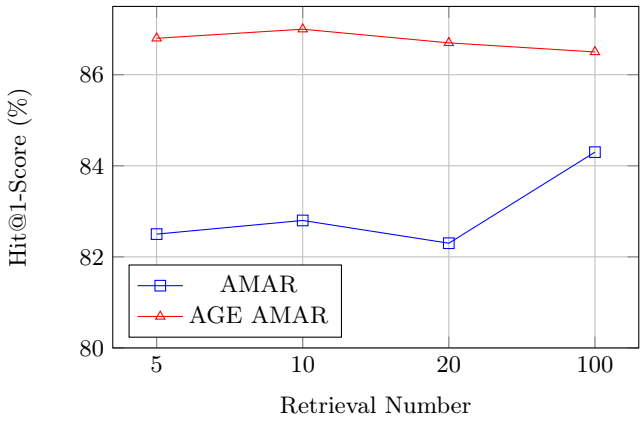


Fig. B.5: Performance impart on vary number of retrieval on WebQSP

The study on number of retrieval As illustrated in FigureB.5, AMAR was designed to address the challenge posed by excessively long retrieved data inputs and to leverage rich information more effectively. However, when the volume of

retrieved data is relatively small, AMAR's performance shows minimal improvement, indicating that the recalled information is insufficient. In such cases, AGE is capable of mapping retrieved data to useful embeddings, eading to significant performance improvements, with scores of 86.8 for 5 retrievals and 87.0 for 10 retrievals.
Conversely, when large amounts of data are retrieved, the accompanying noise complicates the ability of LLMs to identify and prioritize the most relevant information. AGE consistently maintains its performance with minimal variation, highlighting the robustness as 86.5. Moreover,to fairly compare with AMAR, we choose 100 retrievals with an Hit1@ of 86.5.

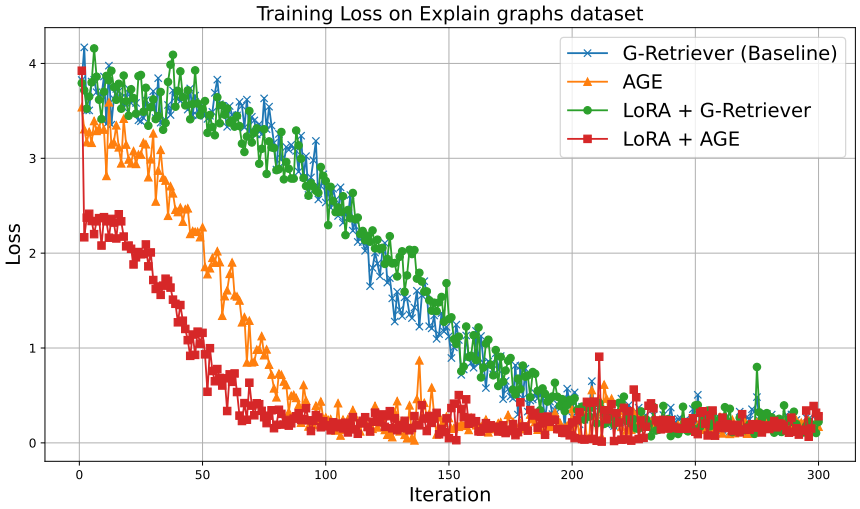


Fig. B.6: Training loss on Explain Graphs.

Impart of AGE with LoRA Finetuning performance We conduct extensive experiments to compare the trend of training loss with G-Retriever baselines, as illustrated in FigureB.6 and B.7. AGE provides structural constraints in the embedding space through mask-based SSL with reconstruction objective, enabling LLMs to better separate signals and capture relationships within it, leading to increased convergence rate and lower loss observed in the initial training stage compared with the G-Retriever.

The choice of RL method for Node Sampler Table B.6 show Both Gumbel-Softmax and REINFORCE demonstrate strong performance on the ExplaGraphs dataset, outperforming the Straight-Through Estimator. On the WebQSP dataset,

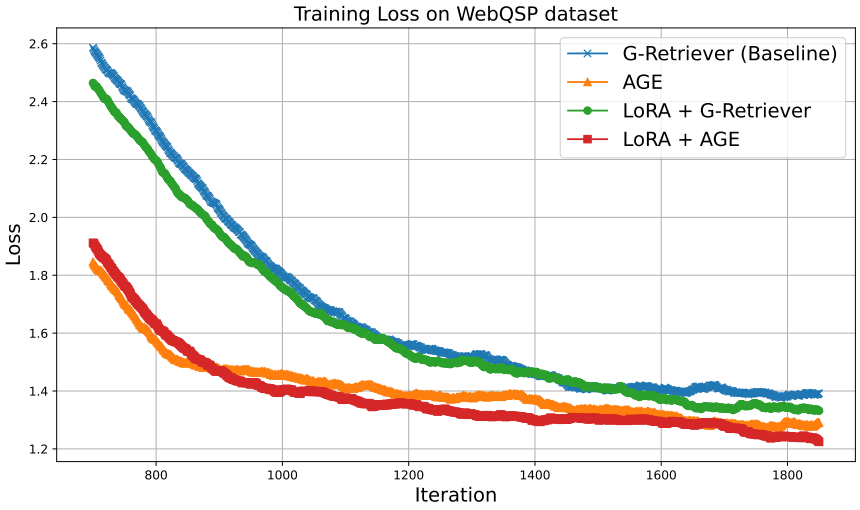


Fig. B.7: Training loss on WebQSP

Table B.6: The study of sampling strategy on Node Sampler (with Llama3.2 1b on PEFT, on ExplaGraphs and WebQSP).

	Gumbel-Softmax	Straight-Through (ST) Estimator	REINFROCE
ExplaGraphs	0.8378	0.8241	0.8501
WebQSP	68.5	66.7	69.1

REINFORCE leads slightly, indicating it may be the most effective method among the three tested for this task. From these observations, we decided to use REINFROCE through the experiments.

The chose different input features for LLMs during training and inference Table B.7 show AGE with h_{out} as LLMs input in inference state outperforms AGE with h_{target} as input on both datasets. This indicates that representing nodes using a concept encoder-decoder h_{out} is more effective than the target encoder in downstream LLMs input tasks. Based on these observations, we concluded that designing the connection of h_{target} during training and h_{out} during inference to the downstream LLM not internalizes the learning-inference mismatch. Instead, it allows the student model has already surpassed the performance of the teacher model, allowing for a more robust representation.

The stability of Target encoder Due to differences in node representations during the training inference stage, using identical parameters for both the concept encoder and the target encoder helps prevent distributional shifts. We apply

Table B.7: Performance of AGE in PEFT (with Llama3.2 1b , on ExplaGraphs and WebQSP).

	PEFT G-Retriever	AGE w h_{target} as LLM input	AGE w h_{out} as LLM input
ExplaGraphs	0.7328	0.8212	0.8501
WebQSP	65.3	66.7	69.1

Table B.8: Performance of Target Encoder on ExplaGraphs, trained with PEFT using Llama3.2 1b.

Loss type	w/o norm + w/o EMA	norm	EMA	norm + EMA
L1	0.7978	0.8375	0.8194	0.8303
MSE	0.8357	0.8501	0.8375	0.8501

two popular techniques to enhance the stability provided by the target encoder for the concept encoder-decoder during the training stage. EMA weights are defined as an exponential moving average of the encoder weights, and normalization is applied to enhance stability during the learning process. Normalization ensures consistent activation distributions and reduces internal covariate shift. Table B.8 shows that MSE performs better with L1, and normalization alone achieved the highest score. These results indicate that normalization consistently improves encoder stability and performance, and adding EMA offers further enhancements.

The landscape of existing KGQA methods Figure B.8 illustrates the spectrum of current Knowledge Graph Question Answering (KGQA) approaches regarding KG retrieval and reasoning capabilities. Graph Neural Network (GNN)-based methods, including NSM [30], ReaRev [54], and G-Retriever [32], perform reasoning on retrieved dense subgraphs by utilizing the GNN to embed graph structures.

Recent LLM-based methods leverage the power of LLMs for both retrieval and reasoning. ToG [66] uses the LLM to retrieve relevant facts hop-by-hop. RoG [50] uses the LLM to generate plausible relation paths which are then mapped on the KG to retrieve the relevant information. However, the frequent calls to the LLM significantly increase the training and inference costs.

In this work, we improve LLM reasoning by enhancing the graph embedding of the GNN method with RL-inspired supervision integrated into the SSL framework. This improves the performance of the non-parametric retriever to levels comparable to those of LLM-based retrievers.

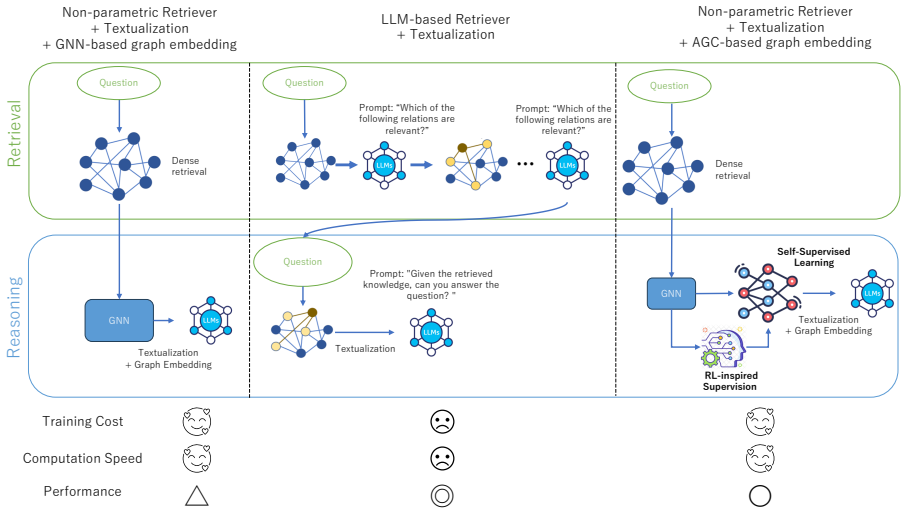


Fig. B.8: The landscape of existing KGQA methods. GNN-based methods reason on dense subgraphs as they can handle complex and graph information. LLM-based methods employ the same LLM for both retrieval and reasoning due to its ability to understand natural language.

2.4 Additional Qualitative Evaluation

We provide additional visualizations in Figures B.10, B.11 on the ExplaGraphss dataset and Figure B.12 on WebQSP dataset.
In the first row of Figure B.10, we consider the addition of node text information and its visualization. It is easier to infer the auxiliary node embeddings "Payday loans" and "For the disadvantaged" from an key node embeddings "Provide assistance". Conversely, it is more challenging to infer the key node embeddings "Provide assistance" from the auxiliary node embeddings "help society" and "available".
Similarly, it is easier to infer "Bullying, However they like, Banned" mask node embeddings from a "Expensive clothes, Students" key node embeddings. Conversely, it is more challenging to infer a "Expensive clothes, Students" mask node embeddings from a "Bullying, However they like, Banned" key node embeddings.
In the second row of Figure B.10, when considering the addition of node text information and its visualization, it is easier to infer the auxiliary node embeddings "Motivation" and "Students work harder" from the key node embedding "Student loans". Conversely, it is more challenging to infer a "Student loans" auxiliary node embeddings from "Motivation" and "Students work harder" key node embeddings. Similar things are shown in Figure B.11 and Figure B.12.
Failure Case Analysis: Furthermore, we provide a failure case on the WebQSP dataset where AGE was trained with LLaMA 3.2 1B using a sampling

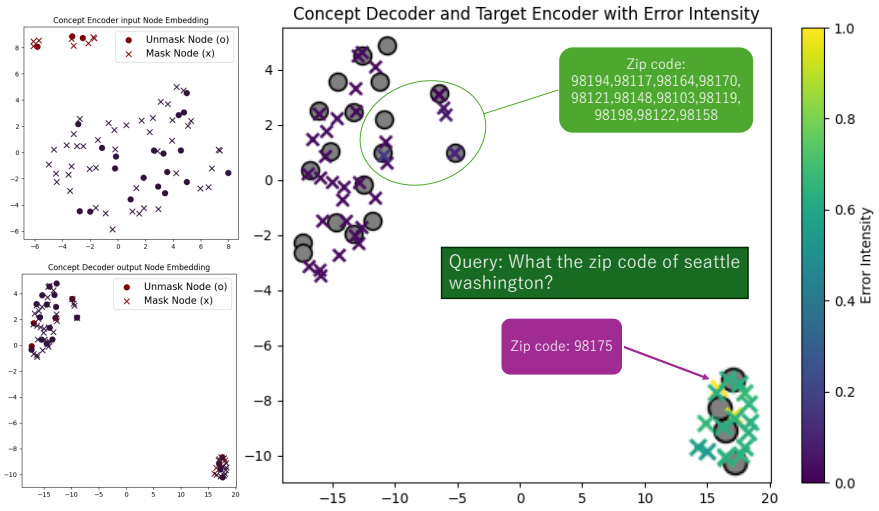


Fig. B.9: Failure case visualization of AGE on WebQSP dataset.

rate of 0.3. In this case, the query is "What is the zip code of Seattle, Washington?". Based on the query, the retriever is provided with the node "98175," which is one of the true answers. However, the response of the LLM lacks this node. To analyze this, we visualized the node sampling results from the concept encoder's input and the concept decoder's output, as shown in Figure B.9. In the first column (top row), similar to the above visualization, the graph encoder process maintains the clustering structure of text graph embeddings to provide input to the concept encoder. Additionally, the output of the concept encoder-decoder module (bottom row) shuffles the colored nodes. In the second column, the target loss of each auxiliary node is represented using a color bar, where 1.0 indicates the maximum error in the test set and 0.0 the minimum error. In this column, the top left side nodes "98194, 98117, 98164,..." include low target loss auxiliary nodes, and the key nodes are the true answers. In parallel, the bottom right side node "98175" is an auxiliary node with a high target loss. This may be why the LLM omits this node in its response, and adjusting the trainable sampling rate could be a solution.

2.5 Discussion on the Complexity

Training Computational Resources on AGE G-Retriever Following the previous G-Retriever [32] method, we utilized the same two A100 GPUs, each with 80GB of memory, and conducted tests on the Llama3-8b, Llama3.1-1B, and Llama3.1-3B on WebQSP datasets. Our experiments had a training batch size of 16 and an evaluation batch size of 32, yielding the following results in Table B.9 for training cost and Table B.10 for validation speed.

Method	LLM	Hit@1	Training Time (min/epoch)
G-Retriever	Llama 2 7B	70.5	6.2
AGE G-Retriever	Llama 3.2 1B	62.5	2.0
AGE G-Retriever	Llama 3.2 3B	73.5	4.5
AGE G-Retriever	Llama 3.2 8B	78.3	6.4
G-Retriever	Llama 2 7B LoRA	73.8	6.9
AGE G-Retriever	Llama 3.2 1B LoRA	69.1	2.4
AGE G-Retriever	Llama 3.2 3B LoRA	77.3	5.9
AGE G-Retriever	Llama 3.2 8B LoRA	80.3	7.3
AGE AMAR	Llama 2 7B LoRA	86.5	8.7

Table B.9: Training cost of AGE G-Retriever on the WebQSP dataset.

	G-Retriver	**AGE**	**AGE**	**AGE**
LLM size	**Llama 2 7b**	**Llama 3.1 8b**	**Llama 3.2 3b**	**Llama 3.2 1b**
All Parameters (B)	6.8	8.1	3.3	1.3
Trainable Para (B)	0.041	0.087	0.078	0.072
Inference speed (Tokens/sec.)	97.0	81.4	87.6	148.5
Hit@1	70.49	78.25	73.46	62.53

Table B.10: Inference speed of AGE on the WebQSP dataset.

The Table B.9 shows the training speed and performance of AGE on the WebQSP dataset. The PEFT setting, without the graph RAG component, takes 18.7 min/epoch through prompt tuning and 19.0 min/epoch when applied with LoRA. Subsequently, the G-Retriever approach via graph RAG reduces graph size and speeds up training time.
By enhancing the embedding module on the graph RAG component, AGE with Llama3.1 8B achieves a higher Hit@1 of 78.25 in 6.4 minutes per epoch. In the tuned LLM setting, AGE with Llama3.1 8B and LoRA achieves a Hit@1 of 80.34 in 7.3 minutes per epoch. These results highlight that AGE with Llama3.2 3B outperforms G-Retriever with Llama2 7B, achieving better performance without longer training time.

Inference Computational Resources on AGE G-Retriever Table B.10 presents the validation speed and performance of various AGE configurations on the WebQSP dataset. Among the AGE models, Llama 3.2 3B model offers a balanced performance with a Hit@1 and an inference speed of 87.6 tokens per second. The AGE with Llama 3.2 1B achieves a significantly higher inference

	LLM	Non-parameter Retriever	Trainable Retriever		WebQSP	CWQ
			GNN	LLM	Hit@1	Hit@1
ToG	Llama2-70B			✓	68.9	57.6
RoG	Llama2-7B			✓	74.2	56.4
ReKnoS	Llama3.1-8B		✓	✓	67.9	56.7
DualR	Llama2-13B		✓	✓	78.3	58.0
StructGPT	ChatGPT			✓	72.6	55.3
ToG	ChatGPT			✓	-	76.2
ToG-2	ChatGPT			✓	81.1	-
RoG	ChatGPT			✓	-	80.0
ReKnoS	ChatGPT		✓	✓	81.1	58.5
GNN-RAG	ChatGPT		✓		85.7	66.8
PoG	ChatGPT			✓	-	82.0
DualR	ChatGPT		✓	✓	-	82.8
KBQA	GPT-4			✓	72.5	-
ReKnoS	GPT-4		✓	✓	84.9	68.2
ToG	GPT-4			✓	82.6	69.5
PoG	GPT-4			✓	87.3	75.0
DualR	GPT-4		✓	✓	87.6	73.6
GraphToken	Llama2-7B	✓			57.1	-
G-Retriever	Llama2-7B-LoRA	✓			70.2	-
AGE G-Retriever	Llama3.1 8B-LoRA	✓			80.3	-
AMAR	Llama2-7B-LoRA	✓			84.3	82.9
AMAR	Llama2-13B-LoRA	✓			83.3	83.1
AGE AMAR	Llama2-7B-LoRA	✓			86.5	85.2
AGE AMAR	Llama2-13B-LoRA	✓			86.2	85.1

Table B.11: Performance comparison of trainable retriever with AGE.

speed of 148.5 tokens per second while maintaining a lower Hit@1. This increased speed can be attributed to the reduced number of parameters in the 1B model, which allows for faster computation and more efficient processing, albeit at the expense of some accuracy.
These results indicate that while higher parameter models like AGE+Llama 3.1 8B provide superior accuracy, lower parameter models such as AGE+Llama 3.2 1B offer significantly increased processing speeds, supporting diverse application requirements.

Comparison with trainable retriever methods AGE, utilizing a non-parametric retriever, achieves accuracy levels comparable to state-of-the-art models that employ trainable parametric retrievers. As shown in Table B.11, AGE (Llama3.1 8B-LoRA with a non-parametric retriever) attains a Hit@1 score of 80.3% on the WebQSP dataset, closely approaching DualR (ChatGPT with a parametric retriever), which achieves 82.8%. This demonstrates that AGE effectively bridges the performance gap between non-parametric and parametric retriever models, achieving high accuracy without the additional complexity and training overhead associated with parametric retrievers. This performance

notably surpasses other models employing non-parametric retrievers, such as GraphToken (Llama2-7B) with 57.1% and G-Retriever (Llama2-7B-LoRA) with 70.2%.

The substantial increase in accuracy demonstrates that AGE enhances reasoning capabilities without relying on trainable parametric retrievers. This positions AGE as a leading approach within non-parametric retriever frameworks, closing the performance gap with models that utilize more complex and resource-intensive trainable retrievers. AGE can be deployed to train and perform inference on two RTX 2080Ti 11GB GPUs or one A100 80GB GPU.

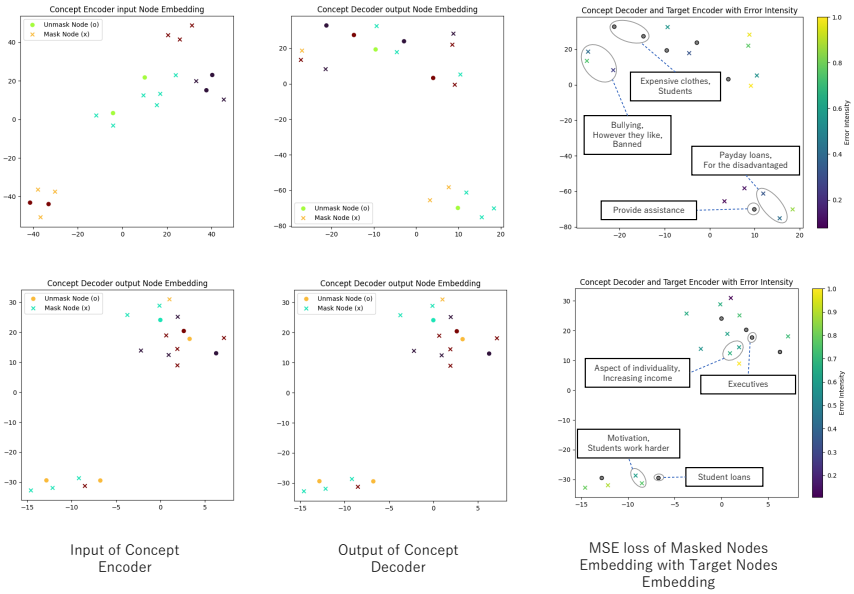


Fig. B.10: An example visualization of AGE on the ExplaGraphs dataset.

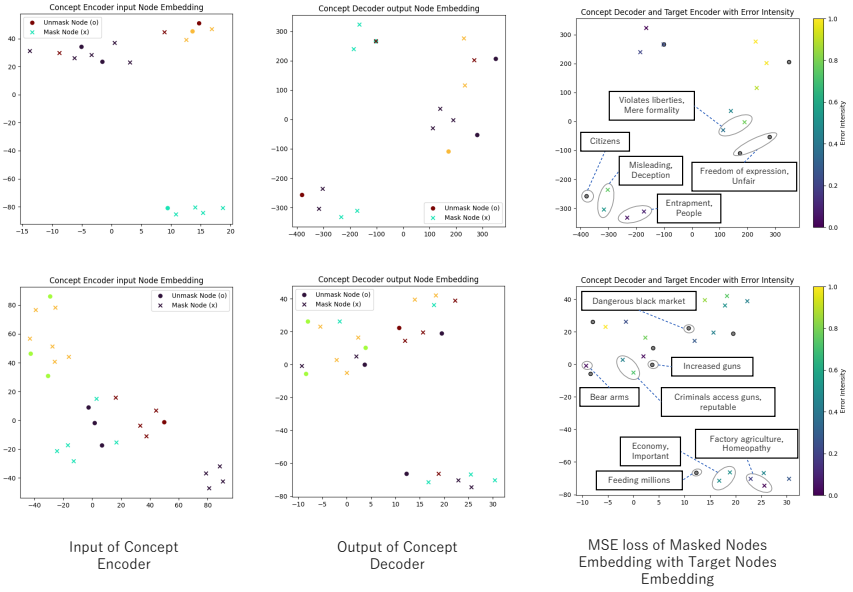


Fig. B.11: Another example visualization of AGE on the ExplaGraphs dataset.

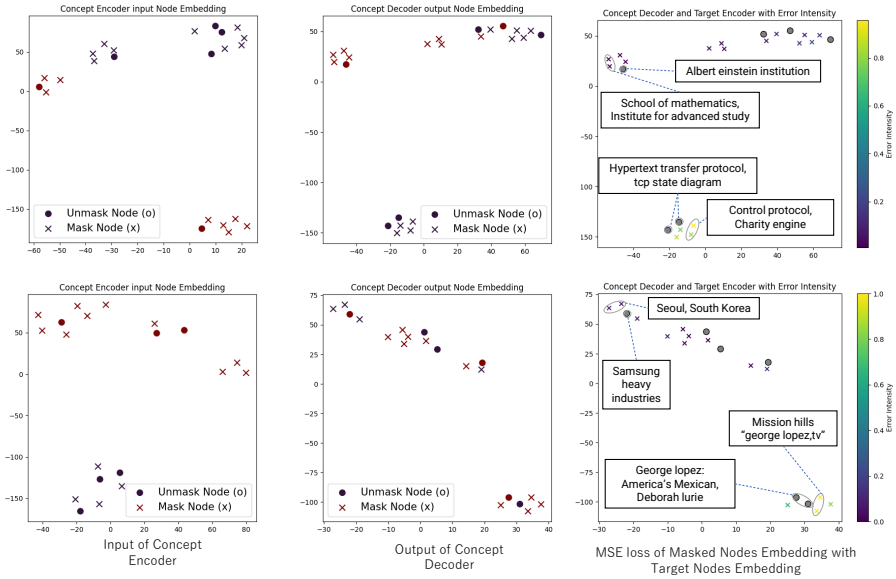


Fig. B.12: An example visualization of AGE on the WebQSP dataset.