T-Retriever: Tree-based Hierarchical Retrieval Augmented Generation for Textual Graphs

Chunyu Wei[1*], **Huaiyu Qin**[1*], **Siyuan He**[1], **Yunhai Wang**[1], **Yueguo Chen**[1†]

[1]Renmin University of China, China
weicy15@icloud.com

Abstract

Retrieval-Augmented Generation (RAG) has significantly enhanced Large Language Models' ability to access external knowledge, yet current graph-based RAG approaches face two critical limitations in managing hierarchical information: they impose rigid layer-specific compression quotas that damage local graph structures, and they prioritize topological structure while neglecting semantic content. We introduce T-Retriever, a novel framework that reformulates attributed graph retrieval as tree-based retrieval using a semantic and structure-guided encoding tree. Our approach features two key innovations: (1) Adaptive Compression Encoding, which replaces artificial compression quotas with a global optimization strategy that preserves the graph's natural hierarchical organization, and (2) Semantic-Structural Entropy (S^2-Entropy), which jointly optimizes for both structural cohesion and semantic consistency when creating hierarchical partitions. Experiments across diverse graph reasoning benchmarks demonstrate that T-Retriever significantly outperforms state-of-the-art RAG methods, providing more coherent and contextually relevant responses to complex queries.

Code — https://github.com/T-Retriever/T-Retriever

Introduction

Large Language Models (LLMs) have revolutionized artificial intelligence (Brown et al. 2020; Touvron et al. 2023), yet they still struggle with complex structured data. A significant portion of real-world information—from scientific knowledge to social networks and enterprise data—naturally exists as **attributed graphs** (Wei et al. 2022a, 2025; Zhang et al. 2025). Enabling LLMs to effectively reason over these structures is crucial for unlocking deeper insights (He et al. 2024; Zhang et al. 2024). In response, Graph-based Retrieval-Augmented Generation (RAG) approaches (Edge et al. 2024; Peng et al. 2024) have emerged to connect LLMs with external knowledge captured in graphs.

A critical limitation in current graph-based RAG approaches is the absence of sophisticated hierarchical knowledge management for effective multi-resolution context retrieval, as illustrated in Figure 1. Effectively managing large attributed graphs requires partitioning them into hierarchies that organize information at different granularity levels. Current systems like GraphRAG (Edge et al. 2024) and HiRAG (Huang et al. 2025) typically rely on conventional community detection algorithms such as Leiden (Traag, Waltman, and Van Eck 2019), which present two fundamental limitations when building *hierarchical* indices:

- **Suboptimal Hierarchical Partitioning:** Conventional algorithms impose rigid, predefined layer-specific compression quotas that damage local graph structures and fail to adapt to the data's intrinsic organization. Their bottom-up construction treats layers in isolation, hindering semantic coherence across hierarchical levels and producing representations misaligned with the graph's natural multi-resolution structure.

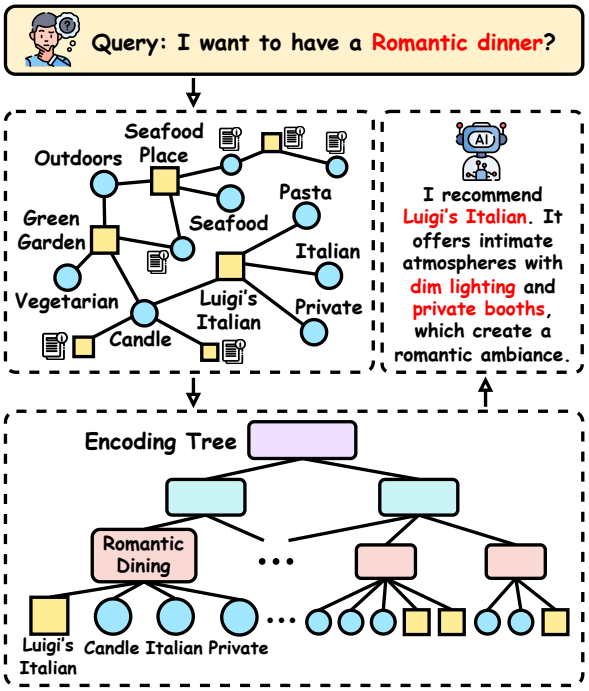


Figure 1: Illustration of `T-Retriever`. Hierarchical organization of attributed graph knowledge enabling effective multi-resolution context retrieval for question answering.

*These authors contributed equally.
†Corresponding author. He works at BRAIN of RUC.

- **Semantic-Structural Disconnect:** These methods predominantly focus on topological structure while neglecting the rich semantic information embedded in node and edge attributes. This oversight results in hierarchical clusters that may be structurally coherent but semantically inconsistent, severely limiting the RAG system's ability to synthesize knowledge requiring both structural and semantic understanding.

To address these limitations, we propose a paradigm shift from graph-based to **tree-based retrieval** with **`T-Retriever`**, a system employing a semantic and structure-guided encoding tree for hierarchical representation and retrieval. `T-Retriever` features two innovations:

First, **Adaptive Compression Encoding** overcomes suboptimal partitioning through a top-down approach inspired by Shannon-Fano coding (Connell 1973). This replaces rigid quotas with a global optimization strategy that recursively divides the graph based on joint entropy. Unlike bottom-up methods that can disrupt semantic coherence, our approach preserves local structural patterns and cross-layer dependencies while maintaining a global perspective that better reflects the graph's intrinsic multi-resolution structure.

Second, **Semantic-Structural Entropy** (S^2-Entropy) addresses the semantic-structural disconnect by quantifying information via the joint distribution of graph topology and attribute semantics. Minimizing S^2-Entropy during encoding ensures the construction of clusters that are both structurally cohesive and semantically consistent, enabling more effective retrieval and reasoning.

Our main contributions are:

- We reformulate graph retrieval as tree-based retrieval, proposing Adaptive Compression Encoding to preserve the graph's natural hierarchical organization.
- We introduce Semantic-Structural Entropy, integrating both structural patterns and semantic content to guide the construction of a unified, semantically and structurally coherent encoding tree.
- Experiments on graph reasoning benchmarks demonstrate that `T-Retriever` significantly outperforms state-of-the-art RAG methods in graph-related scenarios.

Related Work

Graph-based Retrieval for Large Language Models. The synergy between large language models (LLMs) and knowledge graphs (KGs) is crucial for enhancing reasoning and mitigating hallucination (Pan et al. 2024; Clark et al. 2019). While early methods required costly model fine-tuning (Sun et al. 2019; Yasunaga et al. 2021), the advent of Retrieval-Augmented Generation (RAG) (Lewis et al. 2020) has shifted focus to in-context learning.

This paradigm was extended to graphs, creating the field of GraphRAG (Gao et al. 2023; Sen, Mavadia, and Saffari 2023). Unlike standard RAG, GraphRAG retrieves interconnected nodes from a graph, providing structured context to the LLM. Foundational methods like G-Retriever (He et al. 2024) established the efficacy of this approach by textualizing retrieved subgraphs. Our work builds on this paradigm by optimizing the core retrieval step.

Hierarchical Indexing for RAG To handle large-scale data, hierarchical indexing has become critical. In the textual domain, RAPTOR (Sarthi et al. 2024) builds a hierarchy via bottom-up clustering and summarization. This concept has been extended to graphs: ArchRAG (Wang et al. 2025) applies community detection, while HippoRAG (Jimenez Gutierrez et al. 2024) uses a Personalized PageRank (PPR) traversal at query time. These methods represent a significant step forward, but their hierarchy construction often relies on heuristics or costly online computation. In contrast, our approach builds the hierarchy offline using a principled, information-theoretic objective.

Structural Entropy and Our Novelty A key challenge in GraphRAG is graph partitioning. While structural information theory provides a principled framework for this (Li and Pan 2016; Rosvall and Bergstrom 2008), and has been applied in methods like Structural Entropy guided Pooling (SEP) (Wu et al. 2022), these topological approaches are blind to node semantics. Our work addresses this by proposing **S^2-Entropy**, an objective unifying structural and semantic information. T-Retriever's novelty lies in using S^2-Entropy to drive a **top-down partitioning** of the graph, creating a balanced, multi-resolution index that contrasts sharply with prior bottom-up or heuristic approaches.

Preliminaries

Textual Attributed Graphs. We model complex, information-rich data as a **textual attributed graph**, defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \{x_v\}_{v\in\mathcal{V}}, \{x_e\}_{e\in\mathcal{E}})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of n nodes; $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of m edges representing relationships between nodes, defining the graph's topology, typically represented by an adjacency matrix $\mathbf{A}$; $x_v \in \mathcal{D}^{L_v}$ denotes the sequential text attribute of node $v \in \mathcal{V}$, where $\mathcal{D}$ is the vocabulary and L_v is the text length; $x_e \in \mathcal{D}^{L_e}$ denotes the sequential text attribute of edge $e \in \mathcal{E}$. This definition captures attributed graphs where attributes are textual, requiring semantic understanding when analyzed by LLMs.

Retrieval-Augmented Generation (RAG) for Graphs. RAG enables users to interact with and query the textual attributed graph $\mathcal{G}$ using natural language, leveraging the power of Large Language Models (LLMs). A graph-based RAG system $\mathcal{M}$ consists of:

1. **Graph Indexer (φ)**: Processes $\mathcal{G}$ to create an efficient representation for retrieval. This component generates semantic embeddings for nodes (v) and edges (e) by applying a pre-trained Language Model (LM) to their respective text attributes: $z_v = \text{LM}(x_v) \in \mathbb{R}^d$, yielding d-dimensional vectors that capture semantic meaning. In our work, φ specifically creates an optimized hierarchical encoding tree index over $\mathcal{G}$.
2. **Graph Retriever (ψ)**: Given a natural language query q, retrieves the most relevant contextual G^* (e.g., relevant nodes, edges, subgraphs, or information derived from the hierarchical index) from the indexed graph. $\psi(G^*|q, \mathcal{G})$ denotes the process of retrieving context G^*.

3. **Generator** (LLM): An LLM generating the answer a based on the query q and the retrieved graph context G^*.

The objective is to generate the optimal answer a^* by maximizing the likelihood:

$$a^* = \arg\max_a LLM(a|q, G^*) \quad \text{where } G^* \sim \psi(G|q, \mathcal{G}).$$

Encoding Tree. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \{x_v\}_{v\in\mathcal{V}}, \{x_e\}_{e\in\mathcal{E}})$, we define its corresponding Encoding Tree $\mathcal{T}$ as a hierarchical structure with the following properties:

1. Each node α in $\mathcal{T}$ is associated with a subset of graph nodes $\mathcal{V}_\alpha \subseteq \mathcal{V}$. The root node ρ corresponds to the entire node set, $\mathcal{V}_\rho = \mathcal{V}$. For a leaf node α at the maximum tree depth L, $\mathcal{V}_\alpha$ is a singleton set $\{v\}$ containing exactly one graph node $v \in \mathcal{V}$. Leaf nodes at depths less than L may correspond to empty sets.
2. For every non-leaf node α, let $\alpha^{\langle 1\rangle}, \ldots, \alpha^{\langle k_\alpha\rangle}$ be its immediate children, where k_α is the number of children. The parent node of α is denoted α^-. The node set $\mathcal{V}_\alpha$ is the disjoint union of the node sets associated with its children: $\mathcal{V}_\alpha = \bigcup_{i=1}^{k_\alpha} \mathcal{V}_{\alpha^{\langle i\rangle}}$.

Consequently, each level of the encoding tree $\mathcal{T}$ implicitly defines a partition of the graph's node set $\mathcal{V}$, with granularity increasing at deeper levels.

Structural Entropy. Let $d_v = |\mathcal{N}(v)|$ be the degree of node $v \in \mathcal{V}$ (i.e., the number of its neighbors). The volume of a node subset $\mathcal{S} \subseteq \mathcal{V}$ is defined as the sum of the degrees of the nodes within that set: $\mathrm{Vol}(\mathcal{S}) = \sum_{v\in\mathcal{S}} d_v$. The total volume of the graph is $\mathrm{Vol}(\mathcal{G}) = \sum_{v\in\mathcal{V}} d_v = 2|\mathcal{E}|$.

The structural entropy of graph $\mathcal{G}$ with respect to an encoding tree $\mathcal{T}$ is defined as:

$$\begin{aligned} H^{\mathcal{T}}(\mathcal{G}) &= \sum_{\alpha\in\mathcal{T},\alpha\neq\rho} H^{\mathcal{T}}(\mathcal{G};\alpha), \\ \text{where} \quad H^{\mathcal{T}}(\mathcal{G};\alpha) &= -\frac{g_\alpha}{\mathrm{Vol}(\mathcal{G})}\log_2\frac{\mathrm{Vol}(\mathcal{V}_\alpha)}{\mathrm{Vol}(\mathcal{V}_{\alpha^-})}. \end{aligned} \tag{1}$$

Here, g_α represents the number of edges connecting nodes within the set $\mathcal{V}_\alpha$ to nodes outside $\mathcal{V}_\alpha$:

$$g_\alpha = |\{(u, v) \in \mathcal{E} \mid u \in \mathcal{V}_\alpha, v \in \mathcal{V} \setminus \mathcal{V}_\alpha\}|. \tag{2}$$

$\mathrm{Vol}(\mathcal{V}_\alpha)$ is the volume of the node set associated with node α, and α^- denotes the parent node of α.

Methodology

We present T-Retriever, a framework that reformulates attributed graph retrieval through information-theoretic principles. Our approach offers two key innovations: (1) Adaptive Compression Encoding—a globally optimized, learning-free algorithm that constructs hierarchical partitions based on information content rather than predetermined compression quotas; and (2) Semantic-Structural Entropy (S²-Entropy)—a novel metric that unifies topological structure with semantic content to guide partitioning. As shown in Figure 2, T-Retriever constructs an information-theoretically optimal encoding tree, generates hierarchical summaries, and enables efficient multi-resolution retrieval for answering complex graph queries.

Semantic-Structural Entropy

While structural entropy (Eq. 1) effectively captures topological information, it overlooks the rich semantics in node attributes $\{x_v\}$. Pure structural partitioning often yields clusters that lack semantic coherence, hampering the RAG system's ability to synthesize relevant information.

To address this limitation, we introduce semantic information into the hierarchical partitioning objective by leveraging node embeddings $z_v = \mathrm{LM}(x_v) \in \mathbb{R}^d$. Rather than computing pairwise distances (which scales as $O(n_\alpha^2)$), we propose a more efficient **Semantic Density Entropy** that characterizes the embedding distribution in semantic space.

We estimate the probability density function using Kernel Density Estimation:

$$\begin{aligned} p(z) &= \frac{1}{n_\alpha}\sum_{v\in\mathcal{V}_\alpha} K_h(z - z_v), \\ \text{with } K_h(u) &= \frac{1}{(2\pi h^2)^{d/2}}\exp\left(-\frac{\|u\|_2^2}{2h^2}\right). \end{aligned} \tag{3}$$

The semantic density entropy is then defined as:

$$H_{sem}(\mathcal{V}_\alpha) = -\frac{1}{n_\alpha}\sum_{v\in\mathcal{V}_\alpha}\log p(z_v), \tag{4}$$

where lower values indicate higher semantic coherence.

Our **Semantic-Structural Entropy (S²-Entropy)** combines structural and semantic components:

$$H_{S^2}(\mathcal{G};\alpha) = H^{\mathcal{T}}(\mathcal{G};\alpha) + \lambda H_{sem}(\mathcal{V}_\alpha), \tag{5}$$

where the hyperparameter $\lambda \geq 0$ balances their relative importance. The total S²-Entropy for tree $\mathcal{T}$ is: $H_{S^2}^{\mathcal{T}}(\mathcal{G}) = \sum_{\alpha\in\mathcal{T},\alpha\neq\rho} H_{S^2}(\mathcal{G};\alpha)$. Minimizing this metric guides partitioning toward clusters that are both structurally well-defined and semantically meaningful, a crucial advancement for effective graph-based RAG systems.

Adaptive Compression Encoding

The encoding tree organizes the attributed graph hierarchically while preserving essential structural and semantic relationships. By minimizing S²-Entropy, we both reduce structural uncertainty and enhance semantic coherence, creating an optimized knowledge hierarchy. Our goal is to construct an encoding tree $\mathcal{T}^*$ that minimizes S²-Entropy:

$$\mathcal{T}^* = \arg\min_{\forall\mathcal{T}:\text{height}(\mathcal{T})\leq L} H_{S^2}^{\mathcal{T}}(\mathcal{G}). \tag{6}$$

Since this optimization is generally intractable, we develop an efficient approximation algorithm. Unlike the bottom-up approach in Wu et al. (Wu et al. 2022) that iteratively merges nodes, we draw inspiration from Shannon-Fano coding (Connell 1973) to develop a top-down recursive partitioning approach. This method offers superior computational efficiency for large-scale attributed graphs and produces partitions that better align with the semantic organization of textual attributes. We define three key tree transformation operations:

Definition 1 (Partition Operation)**.** *PARTITION*$_{\mathcal{T}}(\alpha)$ *divides node* α *with set* $\mathcal{V}_\alpha$ *into children by solving:*

$$\begin{aligned} \min_{\mathcal{V}_{\alpha_1},\mathcal{V}_{\alpha_2}} \quad & H_{S^2}(\mathcal{G};\alpha_1) + H_{S^2}(\mathcal{G};\alpha_2) \\ \textit{s.t.} \quad & \mathcal{V}_{\alpha_1} \sqcup \mathcal{V}_{\alpha_2} = \mathcal{V}_\alpha, \;\; \mathcal{V}_{\alpha_1}, \mathcal{V}_{\alpha_2} \neq \emptyset. \end{aligned} \tag{7}$$

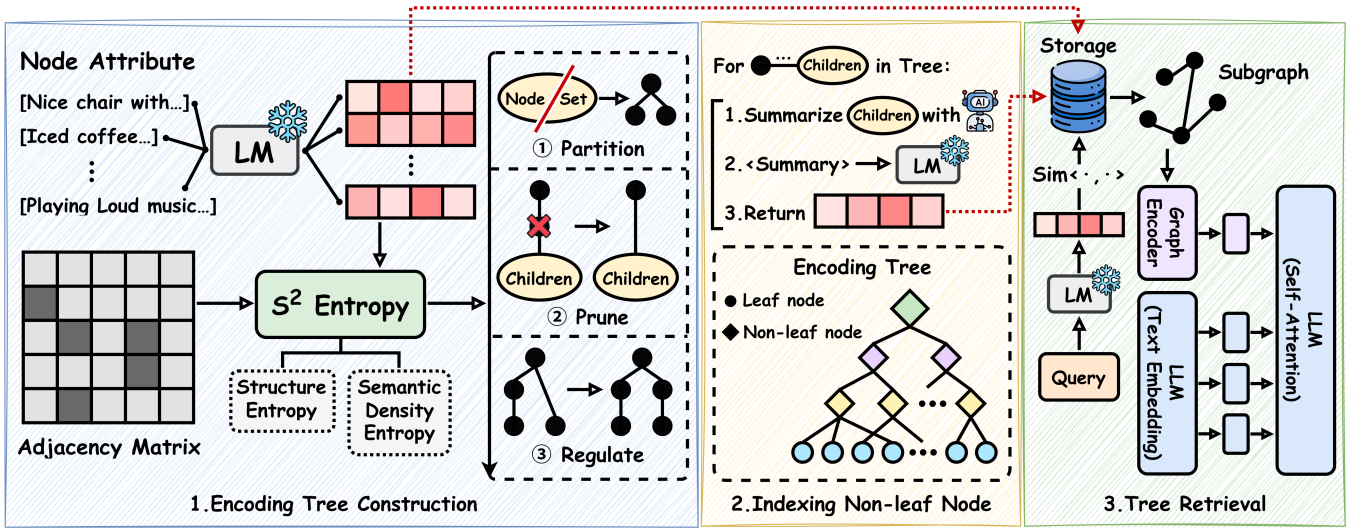


Figure 2: The T-Retriever framework pipeline: (1) Encoding Tree Construction optimizes S²-Entropy (combining structural and semantic information) through partition, prune, and regulate operations; (2) Indexing generates and embeds LLM-based summaries for tree nodes; (3) Tree Retrieval finds relevant nodes, extracts subgraphs, and generates responses using GNN-enhanced LLM prompting.

Definition 2 (Prune Operation)**.** *The $PRUNE_{\mathcal{T}}(\alpha)$ operation removes internal node α, connecting its children to its parent, α^-. The update rule is expressed as:*

$$\alpha^-.children \leftarrow (\alpha^-.children \setminus \{\alpha\}) \cup \alpha.children.$$

Definition 3 (Regulate Operation)**.** *We define the **Regulate Operation**, denoted $REGULATE_{\mathcal{T}}(\alpha, \beta)$, which inserts a node γ between α and its descendant β when their height difference exceeds 1. This process involves two steps:*

1. *Node α adopts γ as a new child, replacing β.*

$$\alpha.children \leftarrow (\alpha.children \setminus \{\beta\}) \cup \{\gamma\}.$$

2. *The new node γ adopts β as its child.*

$$\gamma.children \leftarrow \{\beta\}.$$

Our algorithm proceeds through three stages:

1. **Top-Down Recursive Partitioning**: Starting with the entire graph at the root, we recursively apply the partition operation until reaching either singleton sets or the maximum depth:

$$\text{PARTITION}_{\mathcal{T}}(\alpha) \text{ for all } \alpha \in \mathcal{T} \text{ where } |\mathcal{V}_\alpha| > 1 \quad (8)$$

2. **Height Optimization**: If the tree exceeds height L, we selectively prune nodes that minimize entropy increase:

$$\alpha = \arg \min_{\alpha \in \mathcal{T} \setminus \{\rho, \text{leaves}\}} \left\{ H_{S^2}^{\mathcal{T}_{\text{PRUNE}(\alpha)}}(\mathcal{G}) - H_{S^2}^{\mathcal{T}}(\mathcal{G}) \right\} \quad (9)$$

Unlike conventional methods that impose rigid layer-specific compression quotas, our global optimization approach is guided solely by the desired tree height L, enabling better preservation of local graph structures while effectively capturing cross-layer dependencies.

3. **Structure Regularization**: We ensure proper tree structure by adding intermediate nodes where needed (Regulate Operation), preserving S²-Entropy (Proposition 1).

Proposition 1. *For any encoding tree $\mathcal{T}$ and nodes $\alpha, \beta \in \mathcal{T}$ where α is an ancestor of β with height difference ¿1, the Regulate operation preserves S²-Entropy: $H_{S^2}^{\mathcal{T}}(\mathcal{G}) = H_{S^2}^{\mathcal{T}_{REGULATE(\alpha,\beta)}}(\mathcal{G})$. Proof is provided in Appendix.*

Our Shannon-Fano inspired approach offers several advantages over bottom-up methods like SEP (Wu et al. 2022) when dealing with semantic entropy: (1) better semantic coherence preservation across hierarchical levels, as we maintain the global context during partitioning rather than potentially disrupting it through iterative merging; (2) avoiding the quadratic complexity of pairwise node comparisons inherent in bottom-up methods.

Indexing with Encoding Tree

The encoding tree $\mathcal{T}$ organizes graph nodes hierarchically, with each tree node $\alpha \in \mathcal{T}$ representing a subset of graph nodes $\mathcal{V}_\alpha \subseteq \mathcal{V}$. This structure enables multi-resolution knowledge representation, from specific entities at leaf nodes to broader conceptual groups at higher levels.

Semantic Content Preparation. For leaf nodes representing individual graph nodes, we directly use their original text attributes x_v. For non-leaf nodes α, we generate summaries using an LLM. Let the set of node attributes in the cluster be $\mathcal{X}_v(\alpha) = \{x_v \mid v \in \mathcal{V}_\alpha\}$ and edge attributes be $\mathcal{X}_e(\alpha) = \{x_e \mid e = (u, v) \in \mathcal{E}, u, v \in \mathcal{V}_\alpha\}$. The summary S_α is then:

$$S_\alpha = \begin{cases} x_v, & \text{if } \alpha \text{ is a leaf with } \mathcal{V}_\alpha = \{v\} \\ \text{LLM}(\mathcal{X}_v(\alpha), \mathcal{X}_e(\alpha)), & \text{otherwise} \end{cases} \quad (10)$$

These summaries encapsulate key entities, relationships, and concepts in each subtree.

Embedding and Index Construction. Following Preliminaries section, we use the same language model for encoding both query and node content: $z_\alpha = \text{LM}(S_\alpha) \in \mathbb{R}^d$, where $\text{LM}(\cdot)$ maps text to a d-dimensional embedding space. We organize these embeddings into a multi-level index: $\mathcal{I} = \{(\alpha, z_\alpha, l_\alpha) \mid \alpha \in \mathcal{T}\}$, where l_α denotes the tree level of node α. For efficient similarity search, we deploy an approximate nearest neighbor (ANN) structure, enabling logarithmic-time retrieval of relevant tree nodes.

Retrieving from Encoding Tree

We introduce how to achieve efficient query processing utilizing the encoding tree index. It operates through embedding-based retrieval, subgraph extraction, and GNN-augmented generation.

Given a query q, we compute its embedding and retrieve the most relevant encoding tree nodes:

$$z_q = \text{LM}(q) \in \mathbb{R}^d, \tag{11}$$

$$\mathcal{N}_q = \text{TopK}(\{(\alpha, \text{sim}(z_q, z_\alpha)) \mid \alpha \in \mathcal{T}\}, k). \tag{12}$$

where $\text{sim}(\cdot, \cdot)$ is a similarity function and k controls retrieval volume. This approach treats all encoding tree nodes uniformly regardless of their hierarchical position.

For each retrieved node $\alpha \in \mathcal{N}_q$, we extract its corresponding subgraph:

$$\begin{aligned} G_\alpha &= \big(\mathcal{V}_\alpha, \mathcal{E}_\alpha, \{x_v\}_{v\in\mathcal{V}_\alpha}, \{x_e\}_{e\in\mathcal{E}_\alpha}\big), \\ \text{where} \quad \mathcal{E}_\alpha &= \{(u,v) \in \mathcal{E} \mid u, v \in \mathcal{V}_\alpha\}. \end{aligned} \tag{13}$$

The combined retrieval subgraph is: $G_q = \bigcup_{\alpha\in\mathcal{N}_q} G_\alpha$. Following G-retriever (He et al. 2024) to integrate structure into the language model, we employ a GNN encoder:

$$h_g = \text{POOL}(\text{GNN}_\phi(G_q)) \in \mathbb{R}^{d_g}, \tag{14}$$

$$\hat{h}_g = \text{MLP}_\theta(h_g) \in \mathbb{R}^{d_l}. \tag{15}$$

We textualize the subgraph: $T_q = \text{Textualize}(G_q)$ and generate the final response by:

$$\text{Response} = \text{LLM}(q, T_q, \hat{h}_g) \tag{16}$$

The Catalytic Effect of S²-Entropy

We now analyze how incorporating semantic information enhances the encoding tree's quality through what we term the "catalytic effect." This effect occurs when semantically similar but structurally distant nodes are grouped together, catalyzing improved overall clustering.

For nodes $u, v \in \mathcal{V}$ with high semantic similarity $\text{sim}(z_u, z_v) > 1 - \delta$ but large geodesic distance $d_G(u, v) > \gamma$, pure structural entropy minimization typically separates them into different clusters. However, when S²-Entropy is minimized instead:

Proposition 2 (Catalytic Effect)**.** *There exists a threshold $\lambda_0 > 0$ such that when $\lambda > \lambda_0$ in S²-Entropy, semantically similar but structurally distant nodes will be placed in the same cluster, catalyzing the inclusion of bridging nodes and yielding lower entropy than any partitioning separating them. Proof in Appendix.*

The key insight is that this process occurs in two phases: (1) semantic entropy first brings together distant but semantically similar nodes, and (2) structural optimization then incorporates intermediate nodes that form bridges between them. This creates clusters that better reflect semantic relationships and structural aspects of the graph.

This effect is particularly beneficial for retrieval, as information relevant to a query is often distributed across semantically related but structurally distant parts of the graph. By grouping this information together in the encoding tree, our approach enables more comprehensive retrieval.

Experiments

Experimental Setup

Our framework uses `Sentence-BERT` (Reimers and Gurevych 2019) for encoding and `Llama-2-7b-chat` (Touvron et al. 2023) for generation, with Accuracy as the primary metric.

Baselines. To ensure a comprehensive evaluation, we compare against methods representing different RAG philosophies. **Inference-only**: Standard prompting without retrieval. **Flat Graph-RAG**: Powerful methods like G-Retriever (He et al. 2024) and GRAG (Hu et al. 2024). **Hierarchical Graph-RAG**: We adapt state-of-the-art methods to our attributed graph setting. **RAPTOR** (Sarthi et al. 2024) represents a "semantics-first" approach, adapted by applying its text-based clustering to our node attributes. **ArchRAG** (Wang et al. 2025) represents a "structure-first" approach, adapted by using its community detection method on our graph topology.

Main Results

Table 1 shows that T-Retriever consistently outperforms all baselines. Our key findings are:

- **Value of Hierarchy is Nuanced:** While hierarchical indexing is a promising direction, its effectiveness depends on the strategy. The structure-first ArchRAG shows significant gains over flat methods on complex graphs (WebQSP, BookGraphs). However, the semantics-first RAPTOR, which ignores graph topology, struggles on these tasks and is sometimes outperformed by strong flat baselines like GRAG.
- **Joint Optimization is Key:** T-Retriever achieves the best performance by outperforming all competitors, including the strong structure-aware ArchRAG. This validates our core hypothesis: by jointly optimizing structure and semantics via S²-Entropy, our method creates a more effective knowledge hierarchy than approaches that prioritize one aspect over the other.
- **Gains Scale with Complexity:** The performance advantage of T-Retriever is most pronounced on the largest dataset, BookGraphs (↑6.63%). This pattern confirms that as graph size and complexity increase, the benefits of our principled partitioning become increasingly significant. For smaller graphs like SceneGraphs, the limited tree depth causes T-Retriever to more closely approximate other hierarchical methods, resulting in a smaller performance gap.

Setting	Method	SceneGraphs (Avg. 19 nodes)	WebQSP (Avg. 1,371 nodes)	BookGraphs (Avg. 76,875 nodes)
Inference-only	Zero-shot	$0.4012_{\pm 0.0388}$	$0.4118_{\pm 0.0183}$	$0.3169_{\pm 0.0145}$
	Zero-CoT	$0.5217_{\pm 0.0098}$	$0.5104_{\pm 0.0228}$	$0.3785_{\pm 0.0091}$
	CoT-BAG	$0.5587_{\pm 0.0198}$	$0.4029_{\pm 0.0184}$	$0.3843_{\pm 0.0372}$
	KAPING	$0.4598_{\pm 0.0187}$	$0.5374_{\pm 0.0269}$	$0.4298_{\pm 0.0129}$
Flat Graph-RAG	G-Retriever w/ PT	$0.8102_{\pm 0.0311}$	$0.6772_{\pm 0.0182}$	$0.6603_{\pm 0.0211}$
	G-Retriever w/ LoRA	$\underline{0.8311_{\pm 0.0199}}$	$0.7186_{\pm 0.0394}$	$0.6715_{\pm 0.0385}$
	GRAG w/ PT	$0.7988_{\pm 0.0375}$	$0.7228_{\pm 0.0283}$	$0.6689_{\pm 0.0392}$
	GRAG w/ LoRA	$0.8017_{\pm 0.0486}$	$0.7254_{\pm 0.0477}$	$0.6738_{\pm 0.0573}$
Hierarchical Graph-RAG	RAPTOR (Semantics-first)	$0.7995_{\pm 0.0215}$	$0.7114_{\pm 0.0250}$	$0.6819_{\pm 0.0288}$
	ArchRAG (Structure-first)	$0.8212_{\pm 0.0180}$	$\underline{0.7533_{\pm 0.0291}}$	$\underline{0.7108_{\pm 0.0345}}$
Ours	**T-Retriever**	$\mathbf{0.8507_{\pm 0.0121}}$	$\mathbf{0.7715_{\pm 0.0387}}$	$\mathbf{0.7579_{\pm 0.0183}}$
	Improvement	$\uparrow 2.36\%$	$\uparrow 2.42\%$	$\uparrow 6.63\%$

Table 1: Performance comparison across SceneGraphs, WebQSP, and BookGraphs datasets. **Boldface** denotes the highest score, and underline indicates the best result among baselines.

Hyperparameter Analysis

We conduct sensitivity analysis of three key hyperparameters: (1) the number of encoding tree layers L, (2) the number of retrieved subgraphs k, and (3) the S²-Entropy weighting factor λ and the KDE bandwidth h.

- **Number of Encoding Tree Layers** L: Figure 3(a-c) demonstrates that `T-Retriever`'s accuracy increases with encoding tree depth. Deeper hierarchy better capture the multi-resolution nature of attributed graphs, allowing the encoding tree to represent both fine-grained local structure and broader semantic relationships, creating increasingly optimized information partitioning.
- **Number of Retrieved Subgraphs** k: As shown in Figure 3(a-c), accuracy generally improves with more retrieved subgraphs, providing wider contextual information for the LLM. However, for larger graphs, excessive retrieval may introduce noise and increase LLM processing complexity, necessitating a balance between retrieval effectiveness and efficiency. Our experiments identify $k = 6$ as the typical optimal value.
- **S²-Entropy Hyperparameters (λ and h).** We analyzed the hyperparameters governing S²-Entropy. As shown for WebQSP in Figure 3(d), performance peaks around $\lambda = 1.0$. Our validation across datasets confirms that performance is most robust when λ is in the [1.0, 1.5] range, suggesting a near-equal balance is broadly effective. The critical KDE bandwidth h (Eq. 4) was determined for each dataset via a principled, cross-validated grid search, which prevents performance loss from overfitting or oversmoothing the semantic distribution.

To empirically validate `T-Retriever`'s effectiveness, we conduct analyses on the WebQSP benchmark. As depicted in Figure 4, `T-Retriever` operates in three phases: (1) constructing a hierarchical encoding tree via S²-Entropy optimization, (2) retrieving subgraphs closely aligned with the query in semantic and structural aspects, and (3) converting these subgraphs into standardized formats for LLM-based answer generation. The joint S²-Entropy optimization ensures structural integrity and optimal semantic alignment with the query, boosting accuracy and efficiency.

Configuration	Acc.	F1	Rec.
Semantic-Only	0.6319 (↓18.09%)	0.4055 (↓21.69%)	0.3981 (↓24.67%)
Structural-Only	0.7154 (↓7.27%)	0.4649 (↓10.22%)	0.4780 (↓9.56%)
S²-Entropy	**0.7715**	**0.5178**	**0.5285**

Table 2: Ablation study comparing different entropy configurations on WebQSP.

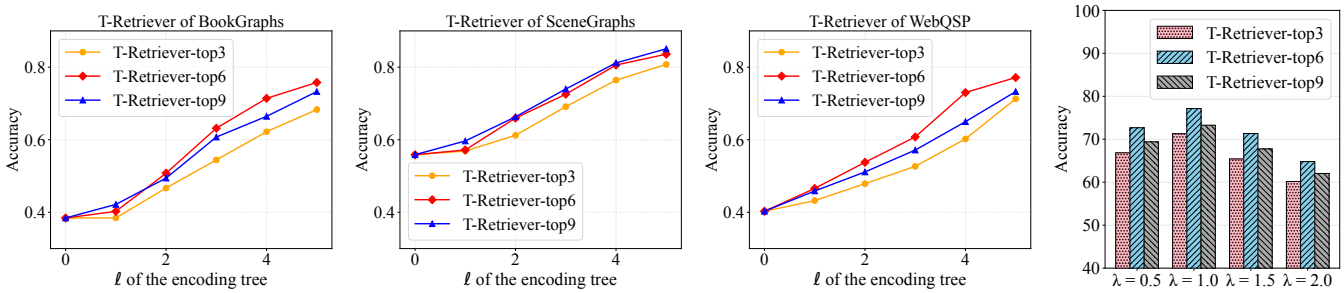
Ablation Study

To rigorously evaluate the contribution of individual components within the Semantic-Structural (S²) Entropy, we conducted a comprehensive ablation study on the WebQSP dataset. We systematically compared three configurations: (1) Semantic-Only entropy optimization, (2) Structural-Only entropy optimization, and (3) our proposed joint S²-Entropy integration.

The results in Table 2 reveal several critical insights. First, structural entropy demonstrates substantially greater influence on retrieval performance than semantic entropy alone, aligning with graph information theory principles suggesting that topological structure provides a fundamental scaffold for information organization. Second, Semantic-Only captures similarity but fails to preserve structural coherence, while Structural-Only lacks semantic relevance due to discarded node attributes. Finally, S²-Entropy achieves balanced performance by simultaneously optimizing both aspects, yielding retrievals that are both structurally cohesive and semantically relevant.

Efficiency Evaluation

Our analysis highlights two key aspects: online retrieval benefits and offline preprocessing costs. First, as shown in Ta-



(a) BookGraphs Analysis (b) SceneGraphs Analysis (c) WebQSP Analysis (d) Impact of weight λ

Figure 3: Sensitivity analysis of key hyperparameters. (a-c) Impact of encoding tree layers L and retrieved subgraphs k across different datasets. (d) Impact of the entropy weighting factor λ on WebQSP.

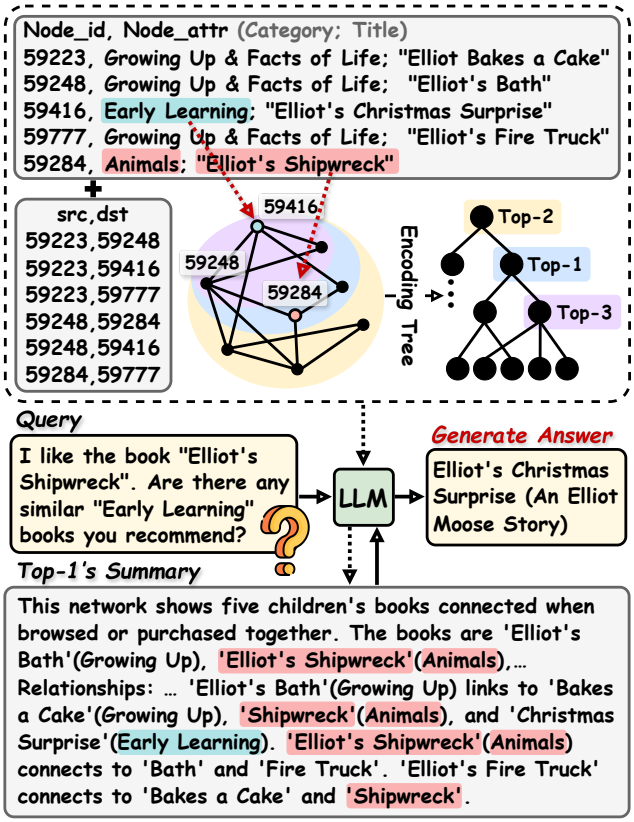


Figure 4: Case study visualization from BookGraphs.

ble 3, T-Retriever significantly reduces the context size of LLMs compared to G-Retriever, cutting token counts by up to 84.16%. This demonstrates its superior online efficiency. Second, to ensure a thorough assessment of practical scalability, we report the one-time offline costs in Table 4. The total indexing time for the large 77k-node BookGraphs dataset is approximately 7.3 hours, a practical one-time investment for a high-quality, reusable index. This demonstrates that the substantial online efficiency gains are achieved with a reasonable and transparent offline computational budget.

Metric	BookGraphs	SceneGraphs	WebQSP
Before Retrieval (Avg.)			
# Tokens	5,377,476	1,396	100,627
# Nodes	76,876	19	1,371
After G-Retriever (Avg.)			
# Tokens	3,973.4	346.0	721.0
# Nodes	213	9	39
After T-Retriever (Avg.)			
# Tokens	**1,487.6**	**123.1**	**114.2**
Reduction	*(↓62.56%)*	*(↓64.42%)*	*(↓84.16%)*
# Nodes	**33**	**7**	**26**
Reduction	*(↓84.51%)*	*(↓22.22%)*	*(↓33.33%)*

Table 3: Efficiency comparison of graph processing methods across three benchmark datasets.

Preprocessing Stage	WebQSP	BookGraphs
Node Embedding	50.3 sec	18.7 min
S^2-Entropy Partitioning	5.7 min	4.2 hours
Summarization & Indexing	9.0 min	3.1 hours
Total Time	**~14.8 min**	**~7.3 hours**

Table 4: Wall-clock time for T-Retriever's one-time, offline preprocessing stages on a single NVIDIA A100 GPU.

Conclusions

We introduced T-Retriever, a framework that reformulates attributed graph retrieval via an information-theoretically optimal encoding tree. By minimizing our proposed S^2-Entropy—a novel metric unifying semantic content and topological structure—T-Retriever builds a multi-resolution knowledge hierarchy that is both structurally and semantically coherent. A key finding is that this superior knowledge organization allows T-Retriever to outperform even fine-tuned baselines without requiring any parameter updates, highlighting the efficiency and power of our approach for complex graph reasoning. Consequently, experiments across diverse benchmarks confirm that T-Retriever sets a new state-of-the-art.

Acknowledgments

This work was supported by The Disciplinary Breakthrough Project of Ministry of Education (MOE, No.00975101), NSFC (No.6250072448, No.62272466, U24A20233), and Big Data and Responsible Artificial Intelligence for National Governance, Renmin University of China.

References

Bianchi, F. M.; Grattarola, D.; and Alippi, C. 2020. Spectral clustering with graph neural networks for graph pooling. In *International conference on machine learning*, 874–883. PMLR.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chen, Z.; Mao, H.; Li, H.; Jin, W.; Wen, H.; Wei, X.; Wang, S.; Yin, D.; Fan, W.; Liu, H.; et al. 2024. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2): 42–61.

Chen, Z.; Mao, H.; Wen, H.; Han, H.; Jin, W.; Zhang, H.; Liu, H.; and Tang, J. 2023. Label-free node classification on graphs with large language models (llms). *arXiv preprint arXiv:2310.04668*.

Clark, K.; Khandelwal, U.; Levy, O.; and Manning, C. D. 2019. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

Connell, J. B. 1973. A huffman-shannon-fano code. *Proceedings of the IEEE*, 61(7): 1046–1047.

Edge, D.; Trinh, H.; Cheng, N.; Bradley, J.; Chao, A.; Mody, A.; Truitt, S.; Metropolitansky, D.; Ness, R. O.; and Larson, J. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Fatemi, B.; Halcrow, J.; and Perozzi, B. 2023. Talk like a graph: Encoding graphs for large language models. *arXiv preprint arXiv:2310.04560*.

Gao, H.; and Ji, S. 2019. Graph u-nets. In *international conference on machine learning*, 2083–2092. PMLR.

Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; Wang, H.; and Wang, H. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2.

He, X.; Bresson, X.; Laurent, T.; Perold, A.; LeCun, Y.; and Hooi, B. 2023. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. *arXiv preprint arXiv:2305.19523*.

He, X.; Tian, Y.; Sun, Y.; Chawla, N.; Laurent, T.; LeCun, Y.; Bresson, X.; and Hooi, B. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37: 132876–132907.

Hu, Y.; Lei, Z.; Zhang, Z.; Pan, B.; Ling, C.; and Zhao, L. 2024. Grag: Graph retrieval-augmented generation. *arXiv preprint arXiv:2405.16506*.

Huang, H.; Huang, Y.; Yang, J.; Pan, Z.; Chen, Y.; Ma, K.; Chen, H.; and Cheng, J. 2025. Retrieval-Augmented Generation with Hierarchical Knowledge. *arXiv preprint arXiv:2503.10150*.

Huang, J.; Zhang, X.; Mei, Q.; and Ma, J. 2023. Can llms effectively leverage graph structural information: when and why.

Hudson, D. A.; and Manning, C. D. 2019. 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*, 6700–6709.

Jia, Z.; Fan, Y.; Zhang, J.; Wei, C.; Yan, R.; and Wu, X. 2023. Improving Next Location Recommendation Services With Spatial-Temporal Multi-Group Contrastive Learning. *IEEE Trans. Serv. Comput.*, 16(5): 3467–3478.

Jimenez Gutierrez, B.; Shu, Y.; Gu, Y.; Yasunaga, M.; and Su, Y. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in Neural Information Processing Systems*, 37: 59532–59569.

Jin, B.; Liu, G.; Han, C.; Jiang, M.; Ji, H.; and Han, J. 2024. Large language models on graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering*.

Lee, J.; Lee, I.; and Kang, J. 2019. Self-attention graph pooling. In *International conference on machine learning*, 3734–3743. pmlr.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474.

Li, A.; and Pan, Y. 2016. Structural information and dynamical complexity of networks. *IEEE Transactions on Information Theory*, 62(6): 3290–3339.

Li, Y.; Li, Z.; Wang, P.; Li, J.; Sun, X.; Cheng, H.; and Yu, J. X. 2023. A survey of graph meets large language model: Progress and future directions. *arXiv preprint arXiv:2311.12399*.

Pan, S.; Luo, L.; Wang, Y.; Chen, C.; Wang, J.; and Wu, X. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7): 3580–3599.

Peng, B.; Zhu, Y.; Liu, Y.; Bo, X.; Shi, H.; Hong, C.; Zhang, Y.; and Tang, S. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*.

Perozzi, B.; Fatemi, B.; Zelle, D.; Tsitsulin, A.; Kazemi, M.; Al-Rfou, R.; and Halcrow, J. 2024. Let your graph do the talking: Encoding structured data for llms. *arXiv preprint arXiv:2402.05862*.

Ranjan, E.; Sanyal, S.; and Talukdar, P. 2020. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 5470–5477.

Reimers, N.; and Gurevych, I. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Rosvall, M.; and Bergstrom, C. T. 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the national academy of sciences*, 105(4): 1118–1123.

Sarthi, P.; Abdullah, S.; Tuli, A.; Khanna, S.; Goldie, A.; and Manning, C. D. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*.

Sen, P.; Mavadia, S.; and Saffari, A. 2023. Knowledge graph-augmented language models for complex question answering. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, 1–8.

Sun, Y.; Wang, S.; Li, Y.; Feng, S.; Chen, X.; Zhang, H.; Tian, X.; Zhu, D.; Tian, H.; and Wu, H. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Tian, Y.; Song, H.; Wang, Z.; Wang, H.; Hu, Z.; Wang, F.; Chawla, N. V.; and Xu, P. 2024. Graph neural prompting with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 19080–19088.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Traag, V. A.; Waltman, L.; and Van Eck, N. J. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1): 1–12.

Wang, S.; Fang, Y.; Zhou, Y.; Liu, X.; and Ma, Y. 2025. ArchRAG: Attributed Community-based Hierarchical Retrieval-Augmented Generation. *arXiv preprint arXiv:2502.09891*.

Wei, C.; Bai, B.; Bai, K.; and Wang, F. 2022a. GSL4Rec: Session-based Recommendations with Collective Graph Structure Learning and Next Interaction Prediction. In Laforest, F.; Troncy, R.; Simperl, E.; Agarwal, D.; Gionis, A.; Herman, I.; and Médini, L., eds., *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, 2120–2130. ACM.

Wei, C.; Fan, Y.; Jia, Z.; and Zhang, J. 2024a. Cross-View Graph Alignment for Mashup Recommendation. *IEEE Trans. Serv. Comput.*, 17(5): 2151–2164.

Wei, C.; Fan, Y.; and Zhang, J. 2022. High-Order Social Graph Neural Network for Service Recommendation. *IEEE Trans. Netw. Serv. Manag.*, 19(4): 4615–4628.

Wei, C.; Fan, Y.; and Zhang, J. 2023. Time-Aware Service Recommendation With Social-Powered Graph Hierarchical Attention Network. *IEEE Trans. Serv. Comput.*, 16(3): 2229–2240.

Wei, C.; Fan, Y.; Zhang, J.; Jia, Z.; and Yan, R. 2024b. Dynamic Relation Graph Learning for Time-Aware Service Recommendation. *IEEE Trans. Netw. Serv. Manag.*, 21(2): 1503–1517.

Wei, C.; Hu, W.; Hao, X.; Wang, Y.; Chen, Y.; Bai, B.; and Wang, F. 2025. Graph Evidential Learning for Anomaly Detection. In Antonie, L.; Pei, J.; Yu, X.; Chierichetti, F.; Lauw, H. W.; Sun, Y.; and Parthasarathy, S., eds., *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.2, KDD 2025, Toronto ON, Canada, August 3-7, 2025*, 3122–3133. ACM.

Wei, C.; Liang, J.; Liu, D.; Dai, Z.; Li, M.; and Wang, F. 2023a. Meta Graph Learning for Long-tail Recommendation. In Singh, A. K.; Sun, Y.; Akoglu, L.; Gunopulos, D.; Yan, X.; Kumar, R.; Ozcan, F.; and Ye, J., eds., *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, 2512–2522. ACM.

Wei, C.; Liang, J.; Liu, D.; and Wang, F. 2022b. Contrastive Graph Structure Learning via Information Bottleneck for Recommendation. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Wei, C.; Wang, Y.; Bai, B.; Ni, K.; Brady, D.; and Fang, L. 2023b. Boosting Graph Contrastive Learning via Graph Contrastive Saliency. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, 36839–36855. PMLR.

Wu, J.; Chen, X.; Xu, K.; and Li, S. 2022. Structural entropy guided graph hierarchical pooling. In *International conference on machine learning*, 24017–24030. PMLR.

Yasunaga, M.; Ren, H.; Bosselut, A.; Liang, P.; and Leskovec, J. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*.

Yih, W.-t.; Richardson, M.; Meek, C.; Chang, M.-W.; and Suh, J. 2016. 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)*, 201–206.

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31.

Zhang, X.; Wei, C.; Yan, R.; Fan, Y.; and Jia, Z. 2024. Large Language Model Ranker with Graph Reasoning for Zero-Shot Recommendation. In Wand, M.; Malinovská, K.; Schmidhuber, J.; and Tetko, I. V., eds., *Artificial Neural Networks and Machine Learning - ICANN 2024 - 33rd International Conference on Artificial Neural Networks, Lugano, Switzerland, September 17-20, 2024, Proceedings, Part V*, volume 15020 of *Lecture Notes in Computer Science*, 356–370. Springer.

Zhang, X.; Yan, R.; Fan, Y.; Zhang, J.; Yuan, H.; and Wei, C. 2025. Cross-domain attention transfer network for recommendation. *Adv. Eng. Informatics*, 68: 103667.