Out-of-Distribution Detection via LLM-Guided Outlier Generation for Text-attributed Graph

Xiangwei Lv[1*] **Mengze Li**[1*] **Jingyuan Chen**[1†]
Zhiang Dong[1] **Sirui Han**[2] **Beishui Liao**[1,3,4†]
[1]Zhejiang University [2]The Hong Kong University of Science and Technology
[3]School of Philosophy, Zhejiang University
[4]The State Key Lab of Brain-Machine Intelligence, Zhejiang University
xiangwei.lv@zju.edu.cn, mengzeli@zju.edu.cn, jingyuanchen@zju.edu.cn,
dong.za@zju.edu.cn, siruihan@ust.hk, baiseliao@zju.edu.cn

Abstract

Text-Attributed Graphs (TAGs), which are characterized with text attributes, are widely used in the real world. When evaluating fully trained models designed for TAG predictions, they may perform significantly unsatisfactory on samples outside the In-Distribution (ID) data, which may raise serious security issues. To tackle it, Out-Of-Distribution (OOD) detection is introduced to the TAGs field, which aims to utilize a detector to classify OOD and ID samples. Recent studies attempt to introduce extra OOD datasets to regularize the detection model. However, due to the vastness of the OOD data space, high-quality OOD samples for training the detector are scarce and difficult to obtain in the real world. Thus, we utilize Large Language Models (LLMs) to generate the OOD training samples with high quality. There are two issues in this process: (1) LLMs tend to generate OOD nodes noticeably distinct from ID nodes, with a limited learning value for OOD and ID relations. (2) Due to the inherent structure of TAGs, obtained OOD nodes need to be integrated with existing nodes by generating edges using LLMs. However, the large number of nodes makes reasoning over each node pair computationally unbearable. Toward these issues, we introduce LLMGuard with challenging OOD-node generation and lightweight edge predictors. Extensive experiments prove the effectiveness of LLMGuard. The source code is available [1].

1 Introduction

Text-Attributed Graphs (**TAGs**), whose graph structures are characterized with text attributes, are commonly applied in various real-world scenarios (Chen et al., 2024; Yang et al., 2021). Within text-attributed graphs, graph nodes capture entities with textual information and graph edges encapsulate relationships between different entities, such as social graphs where each user node is attributed with a textual description (Sharma et al., 2024; Fan et al., 2019) and paper citation graphs in which language content is linked to each paper node (Kipf and Welling, 2016). Despite significant progress, directly transferring fully trained TAG models to samples from unknown distributions may lead to a sharp drop in performance, thereby reducing the prediction reliability and raising security issues (Yang et al., 2024b). To tackle this issue, the Out-of-Distribution (**OOD**) detection (Li et al., 2022) is introduced to the field of text-attributed graphs. It aims to utilize a detector to identify graph OOD data and separate it with In Distribution (**ID**) data to reduce the risk of unreliable predictions.

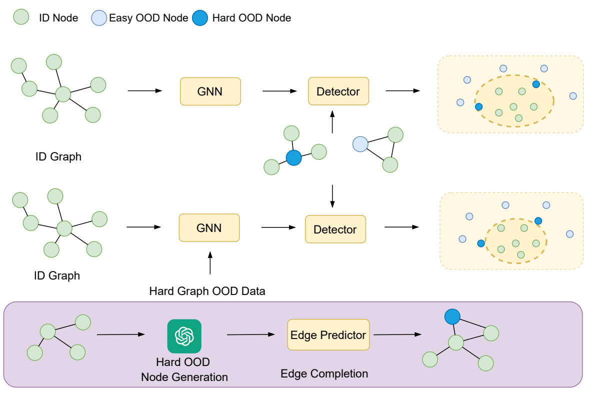


Figure 1: Differences between our method and traditional methods to detect node-level OOD data. Our key idea is to leverage LLMs to generate challenging OOD nodes and efficiently structure them to facilitate the formation of a tighter decision boundary.

The widely used solution (Yang et al., 2024b; Wu et al., 2023) for distinguishing OOD and ID text-attributed graphs predominantly relies on training the detector with ID samples and naturally collected OOD ones. However, due to the vast diversity of the OOD text-attributed graphs (Ming et al., 2022), there may be an extremely low per-

*Equal contribution.
†Corresponding author.
[1]https://github.com/lvXiangwei/LLMGuard.git

centage of the naturally collected graphs exactly located near the classification boundary. The textual node properties and topological graph structures of these collected samples often exhibit substantial semantic differences from existing ID text-attributed graphs. Consequently, these OOD samples are considered low-quality because they are too easy to distinguish and offer limited value in enhancing the graph detector's ability to identify more challenging OOD graphs. To establish a more precise decision boundary, it's crucial to focus on high-quality OOD graphs sharing similar patterns with ID ones yet differ fundamentally.

To acquire more OOD text-attributed graphs with high value, we attempt to utilize the pre-trained Large Language Models (**LLMs**) to generate the qualified samples. In this synthesis process, two major issues need to be considered: (1) **Low-Quality OOD Node Generation.** The OOD nodes directly generated by large language models typically belong to classes that are noticeably distinct from those of the ID nodes. As a result, these OOD nodes are positioned far from the decision boundary separating OOD and ID nodes, thereby diminishing their values for training the detection model. Even when OOD nodes belong to ID-related classes, the specific textual descriptions associated with these directly generated OOD nodes remain disconnected from the existing ID nodes, which is another serious damage to their training value. It is crucial to consider strategies that can effectively prompt large language models to generate OOD nodes of genuinely high quality. (2) **Costly Edge Completion**. Due to the node interdependence of graphs, newly generated OOD nodes need to be associated with existing ID nodes. The inherent differences between OOD and ID nodes make it impractical to directly infer OOD edges using a predictor trained solely on ID nodes. One possible approach is to leverage pre-trained large language models for direct edge prediction between node pairs. However, due to the presence of numerous nodes in text-attributed graphs, this approach often requires an excessive number of calls to large language models for each pair of nodes, leading to an unbearable computational burden.

To address these issues, we propose an effective and efficient framework via LLM-guided Graph oUtlier generAtion foR OOD Detection in text-attributed graphs, called **LLMGuard**. The key idea is to leverage large language models to generate challenging OOD nodes that facilitate learning a clear decision boundary while distilling the edge prediction capabilities of large language models into lightweight modules for efficient edge generation. This consists of two key processes. **Process 1**: We employ pre-trained large language models, guided by potential ID-related classes of OOD nodes and textual content of ID subgraphs, to generate high-quality OOD nodes. These obtained OOD nodes may be highly correlated with existing ID nodes while still preserving OOD characteristics. **Process 2**: We first construct training data for edge prediction using a limited number of LLM calls to identify relationships between generated OOD nodes and ID nodes. To better capture semantic differences between ID and OOD nodes, we fine-tune a local LLM on the node distinction task to obtain informative node representations. Leveraging both the edge training data and node representations from the local LLM, we further train two lightweight edge predictors to distill the LLM's ability for edge prediction.

2 Related Work

2.1 LLM for Text-Attributed Graphs

Text-attributed graphs (TAGs) (Jin et al., 2024; Ren et al., 2024), where both node and edge features are represented as text, have attracted considerable attention for their universal ability to represent graph from diverse domains (Feng et al., 2024). With the emergence of large language models (LLMs) such as GPT-4 (OpenAI, 2023), researchers have increasingly explored their potential across a wide range of applications (Zhao et al., 2024a,b; Dong et al.; Wu et al., 2024), particularly in enhancing tasks related to attributed graphs (TAGs). One paradigm focuses on designing prompts to enable LLMs to better understand graph structures and respond to queries effectively, as seen in InstructGLM (Ye et al., 2023) and NLGraph (Wang et al., 2023). Another line of approaches, exemplified by GraphGPT (Tang et al., 2024) and GraphLLM (Chai et al., 2023), involves encoding graph data as token sequences, allowing LLMs to process and generate graph-related outputs more efficiently.

2.2 Node-level OOD Detection on TAGs

In real-world applications, data samples encountered during test time often exhibit domain shifts (Lv et al., 2025a; Yang et al., 2024a; Lv et al., 2025b; Wang et al., 2025). In this work, we focus on node-level out-of-distribution (OOD) detection on attributed graphs (TAGs), aiming to identify nodes that deviate from the in-distribution data. This enables more reliable and robust predictions. Traditional methods focus on developing scoring metrics to detect OOD nodes, such as MSP (Hendrycks and Gimpel, 2016), ODIN (Liang et al., 2017), and Energy (Liu et al., 2020). For graph OOD detection, GKDE (Zhao et al., 2020) is proposed to utilize a multi-source uncertainty framework to predict node-level Dirichlet distributions and detect OOD nodes. Recently, GNNSAFE (Wu et al., 2023) leverages TAG structures with an energy propagation scheme to improve performance. NodeSAFE (Yang et al., 2024b) further regularize the energy term to mitigate extreme values. GRASP (Ma et al., 2024) proposes a new edge augmentation strategy to better propagate OOD scores among neighboring nodes. Another approach integrates additional OOD samples into training. GNNSAFE++ (Wu et al., 2023) and NodeSAFE++ (Yang et al., 2024b) extend GNNSAFE and NodeSAFE with extra graph OOD data to enhance performance. However, collecting additional OOD datasets in real-world scenarios either impractical or low-quality due to the vast diversity of the OOD space. Our approach harnesses LLMs to generate high-quality OOD graph nodes and strategically structure them, facilitating a more precise decision boundary.

3 Preliminaries

3.1 Notations

In this paper, we study the problem of node-level graph Out-of-Distribution (OOD) detection on Text-Attributed Graphs (TAGs), where unknown classes emerge from an OOD domain. Let $G = (V, E, T)$ be a text-attributed graph, where V denotes the set of nodes, E the set of edges, and T the set of textual attributes. Each node $v_i \in V$ is associated with a text representation $t_i \in T$. The adjacency matrix of G is $A \in \mathbb{R}^{n\times n}$, where $A_{ij} = 1$ if an edge exists between v_i and v_j, and $A_{ij} = 0$ otherwise. The node set V is partitioned into labeled nodes V_l and unlabeled nodes V_u, such that $V = V_l \cup V_u$. The unlabeled nodes further include in-distribution (ID) nodes V_{uid} and out-of-distribution (OOD) nodes V_{uood}, i.e., $V_u = V_{uid} \cup V_{uood}$. The label space of ID nodes is $Y_{id} = \{y_1, y_2, \dots, y_C\}$, where C is the number of ID classes. In contrast, the OOD label space Y_{ood} is typically unknown and highly diverse.

3.2 Node-level Graph OOD Detection

3.2.1 Problem Definition

Let D_{id} denote the distribution of ID nodes $v \in V_{uid}$ with labels $y \in Y_{id}$, and D_{ood} represent the OOD distribution, consisting of OOD nodes $v \in V_{uood}$ with unknown labels $y \in Y_{ood}$. Graph OOD detection aims to distinguish whether a given sample $\mathbf{x} \in V_u$ belongs to an ID or OOD class. Formally, the objective is to design an OOD detector F, defined as:

$$F(\mathbf{x}, G; f_{\text{GNN}}, f_{\text{text}}) = \begin{cases} \text{ID}, & \text{if } S(\mathbf{x}) \geq \gamma \\ \text{OOD}, & \text{if } S(\mathbf{x}) < \gamma \end{cases} \tag{1}$$

Here, f_{GNN} represents a Graph Neural Network (GNN) trained for node classification. More on GNNs is in Appendix D. The text encoder $f_{\text{text}} : T \to \mathbb{R}^d$ maps a node's text attribute to its initial representation. The scoring function $S(\mathbf{x})$ assigns a value to $\mathbf{x}$, with a threshold γ set for high classification accuracy (e.g., 95%). Samples with higher score are classified as ID, while those with lower scores are labeled as OOD.

3.2.2 Energy-based Graph OOD Detection

Motivated by the equivalence between classifiers and energy-based models (Ranzato et al., 2007), recent works (Ma et al., 2024; Wu et al., 2023; Yang et al., 2024b) have leveraged energy values derived from GNN prediction logits to facilitate OOD detection. For a given node v, its energy $\mathbf{E}(v)$ is defined as:

$$\mathbf{E}(v) = -\log \sum_{c=1}^{C} e^{z_c}, \tag{2}$$

where $z_c \in \mathbb{R}$ denotes the predicted logit of the GNN model for class $c \in Y_{id}$, formally expressed as $z_c = f_{\text{GNN}}(v; G)_{[c]}$. In node-level tasks, the GNN model is typically trained using a task-specific loss. Therefore, the supervised loss can be formulated as:

$$\mathcal{L}_{\text{sup}} = \mathbb{E}_{(v,y)\sim D_{id}}(-z_y - \mathbf{E}(v)) \tag{3}$$

Recent works (Wu et al., 2023; Yang et al., 2024b) attempt to introduce additional OOD datasets to better separate the energy values of ID and OOD nodes. Specifically, they incorporate an energy

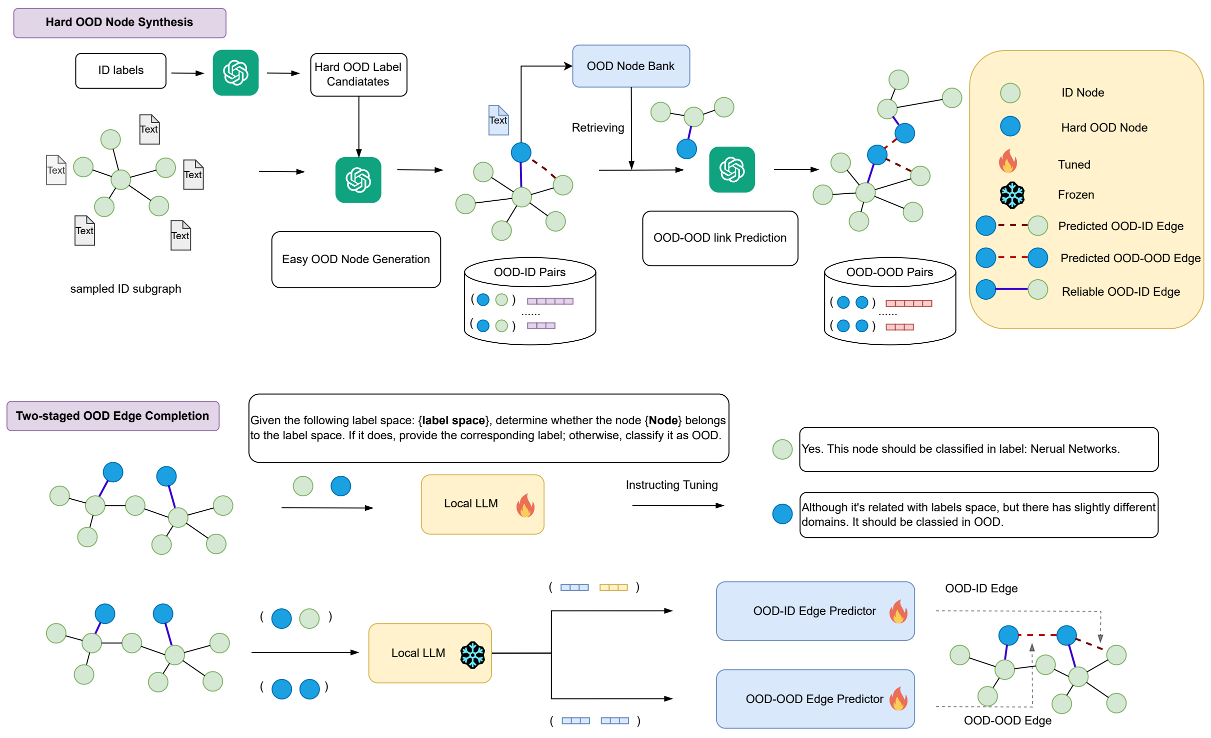

Hard OOD Node Synthesis
ID labels
Hard OOD Label Candiatates
Text
Text
Text
Text
Text
Text
sampled ID subgraph
Easy OOD Node Generation
OOD Node Bank
Retrieving
OOD-ID Pairs
OOD-OOD link Prediction
OOD-OOD Pairs
ID Node
Hard OOD Node
Tuned
Frozen
Predicted OOD-ID Edge
Predicted OOD-OOD Edge
Reliable OOD-ID Edge
Two-staged OOD Edge Completion
Given the following label space: {label space}, determine whether the node {Node} belongs to the label space. If it does, provide the corresponding label; otherwise, classify it as OOD.
Local LLM
Instructing Tuning
Yes. This node should be classified in label: Nerual Networks.
Although it's related with labels space, but there has slightly different domains. It should be classied in OOD.
Local LLM
OOD-ID Edge Predictor
OOD-OOD Edge Predictor
OOD-ID Edge
OOD-OOD Edge

Figure 2: The architecture of LLMGuard consists of two componets: (a) *LLM-Guided Hard OOD Node Synthesis* leverages sampled ID subgraphs and pre-generated challenging OOD labels to synthesize hard OOD nodes. (2) *Two-Staged OOD Edge Completion* distills the edge prediction capabilities of LLMs into two lightweight edge predictors through a two-stage tuning process.

regularization term:

$$\mathcal{L}_{\text{reg}} = \mathbb{E}_{(v,y)\sim D_{id}}\Big(\max(0, \mathbf{E}(v) - a_{\text{in}})^2\Big) + \mathbb{E}_{(v,y)\sim D_{ood}}\Big(\max(0, a_{\text{out}} - \mathbf{E}(v))^2\Big). \tag{4}$$

where a_{in} and a_{out} are thresholds that control the separation margin. The final objective function is given by:

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{sup}} + \lambda \mathcal{L}_{\text{reg}} \tag{5}$$

where λ is a hyperparameter that balances the contribution of the regularization term. Notably, extra OOD datasets are typically unavailable in real-world scenarios. To address it, our proposed method generate high-quality and efficient graph OOD data, enabling more effective detection.

4 Method

4.1 Overview

In this section, we introduce our novel graph OOD data generation framework, LLMGuard, designed to generate high-quality OOD nodes and structure them efficiently into a graph, thereby facilitating exposure to high-quality graph OOD data. Here, high-quality OOD nodes specifically refer to challenging or hard OOD nodes that closely resemble ID nodes, making them difficult to distinguish.

As illustrated in Figure 2, our framework consists of two key components: (1) **LLM-Guided Hard OOD Node Synthesis**. Hard OOD nodes are constructed by first generating challenging OOD labels and then leveraging related ID subgraphs as contextual knowledge to guide LLMs in the generation process. (2) **Two-Staged OOD Edge Completion**. Since TAGs' structured interconnections, newly generated OOD nodes cannot exist in isolation. To mitigate the high cost of extensive LLM calls for each node pair, we distill the edge prediction capabilities of LLMs into lightweight modules through a two-staged training process.

4.2 LLM-Guided Hard OOD Node Synthesis

Leveraging the extensive domain knowledge and strong generative capabilities of LLMs, we aim to generate potential OOD nodes to enhance the training of detection models. However, since OOD nodes are more diverse and scattered than ID nodes in the data space, blindly generating low-quality

OOD nodes offers limited benefits in forming a tighter decision boundary between ID nodes and OOD nodes. Therefore, our key idea is to generate challenging OOD nodes—those that share similarities with ID nodes but have distinct label attributes. This approach enables the detector to learn a more precise and well-defined decision boundary.

To implement it, we frist construct a set of potential challenging OOD labels for each ID label. Then, using the contextual information from the ID subgraph, we generate new OOD nodes that preserve OOD characteristics while maintaining semantic similarities with ID nodes.

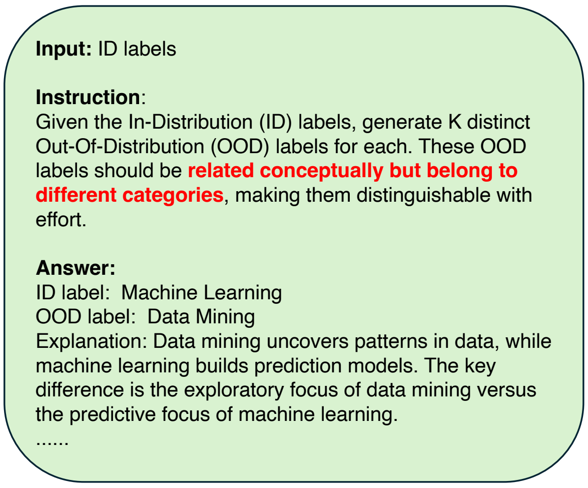
4.2.1 Challenging OOD Label Generation

We classify OOD nodes in D_{ood} as easy or hard based on their distinguishability from ID nodes. Easy OOD nodes are well-separated from ID data, making them relatively straightforward to identify. However, hard OOD nodes lie are closer to the decision boundary, posing greater challenges. For example, in an artificial intelligence citation network, given an ID label *machine learning*, a hard OOD label like *data mining* shares semantic similarities, while an easy OOD label like *optimization algorithm* exhibits a larger semantic shift, offering limited benefit for refining the detector's discrimination ability.

Given the existing ID label space Y_{id}, for each label $y_i \in Y_{id}$, we design prompts to call LLMs to generate K related hard OOD label set, denoted as S_{hood}. In fact, this process constructs a mapping function $\phi : Y_{id} \to S_{hood}$, which maps a specific ID label to a set of candidate hard OOD labels. To streamline the generation of hard OOD labels, we assume that given an ID label y_i, the conditional distribution of its corresponding hard OOD label $y_i^{ood} \in \phi(y_i)$ is independent of the graph structure G, i.e., $P(y_i^{ood}|y_i, G) = P(y_i^{ood}|y_i)$. In this way, we temporarily disregard the impact of the graph structure when generating hard OOD labels. The hard OOD label generation prompt is shown in Figure 3.

4.2.2 ID-Related Hard OOD Node Synthesis

Due to the interdependent nature of graph data, well-structured OOD nodes require essential contextual information to establish meaningful and coherent connections with existing nodes. For example, the same OOD label, such as *Machine Learning*, may exhibit different characteristics depending on the surrounding graph context. In a


Input: ID labels

Instruction:
Given the In-Distribution (ID) labels, generate K distinct Out-Of-Distribution (OOD) labels for each. These OOD labels should be **related conceptually but belong to different categories**, making them distinguishable with effort.

Answer:
ID label: Machine Learning
OOD label: Data Mining
Explanation: Data mining uncovers patterns in data, while machine learning builds prediction models. The key difference is the exploratory focus of data mining versus the predictive focus of machine learning.
......

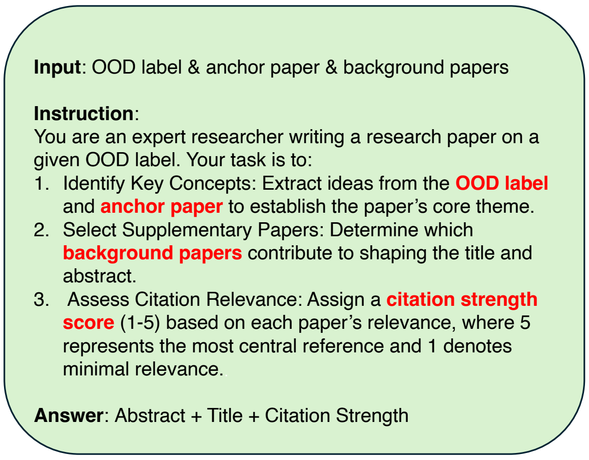
Figure 3: Prompt example for generating hard OOD labels.

citation network, nodes with this label primarily exhibit academic characteristics, whereas in a co-purchase network, they are more likely to reflect consumer-oriented information.

To tackle this challenge, we propose a hard OOD node generation strategy that leverages ID-subgraph information. Specifically, we randomly select an ID node $v_i \in V_l$ with its label $y_i \in Y_{id}$ as an anchor. To provide background knowledge, we extract a subgraph G_i centered on v_i, with its corresponding text attribute set denoted as T_i. The generated OOD node is primarily derived from the anchor node v_i, allowing us to obtain the corresponding hard OOD label $y_i^{ood} \in \phi(y_i)$ based on the original ID label y_i. Following this approach, the probability distribution for generating new text t_i^{ood}, conditioned on the sampled ID subgraph G_i and label y_i, is formulated as:

$$P(t_i^{ood} \mid G_i, y_i) = \sum_{y_i^{ood}} P(y_i^{ood} \mid y_i) P(t_i^{ood} \mid y_i^{ood}, G_i) \tag{6}$$

More derivation details can be found in Appendix B. According to Equ. 6, we can summarize the following generation process for hard OOD nodes. First, we obtain the hard OOD label by sampling $y_i^{ood} \sim P(y_i^{ood} \mid y_i)$. This step is equivalent to randomly selecting an OOD label from the hard OOD label set $\phi(y_i)$. Next, we generate new text attributes by sampling $t_i^{ood} \sim P(t_i^{ood} \mid y_i^{ood}, G_i)$. This process can be implemented by combining the selected subgraph G_i with the hard OOD label y_i^{ood} to prompt the LLM to generate the desired OOD node text attributes. Finally, the process of



Input: OOD label & anchor paper & background papers

Instruction:
You are an expert researcher writing a research paper on a given OOD label. Your task is to:

1. Identify Key Concepts: Extract ideas from the **OOD label** and **anchor paper** to establish the paper's core theme.
2. Select Supplementary Papers: Determine which **background papers** contribute to shaping the title and abstract.
3. Assess Citation Relevance: Assign a **citation strength score** (1-5) based on each paper's relevance, where 5 represents the most central reference and 1 denotes minimal relevance.

Answer: Abstract + Title + Citation Strength

Figure 4: Prompt example for hard OOD node generation.

ID-subgraph related hard OOD node generation can be formalized as:

$$t_i^{ood}, E_i^{\text{O-I}} = \text{LLM}(v_i; y_i^{ood}; T_i) \tag{7}$$

where t_i^{ood} is the newly generated OOD node's attribute information. $E_i^{\text{O-I}}$ records the strength of the relationships between the generated OOD node with other ID nodes in G_i, which will be utilized for subsequent edge prediction training. Figure 4 provides a specific example for hard OOD node generation process in a citation network.

4.2.3 Retrieval-Based OOD-OOD Relation

While above generation process involves the relationship (i.e., $E^{\text{O-I}}$) between OOD nodes and existing ID nodes, the links among OOD nodes themselves remain unexplored. The absence of OOD-OOD links misaligns with real-world relationships, thereby degrading the quality of OOD nodes. Here, we aim to construct an informative OOD-OOD strength dataset $E^{\text{O-O}}$ with minimal LLM calls to support efficient subsequent edge prediction training.

We maintain an OOD node bank B_{ood} to store previously generated OOD nodes v_k^{ood} along with their text attributes t_k^{ood} and OOD labels y_k^{ood}. With a pre-trained language model such as DeBERTa (He et al., 2020), we convert corresponding text attributes into embeddings, i.e., $h_k^{ood} = f_{\text{text}}([t_k^{ood}; y_k^{ood}])$. Then, for a new OOD node v_j^{ood}, we retrieve a few related OOD nodes from B_{ood} and query LLMs to determine potential links. For efficient retrieval, we use cosine similarity for embedding-based matching:

$$V_j^{ood} = \text{Top-L}\left(\{v_k^{ood} \in B_{ood} \mid \cos(h_j^{ood}, h_k^{ood})\}\right) \tag{8}$$

In this way, we query LLMs at most L times to determine whether edges exist between v_j^{ood} and the OOD nodes in V_j^{ood}. This ensures that the total number of LLM queries remains within $O(NL)$, where N is the total number of generated OOD nodes. An example is provided in Appendix E.

4.3 Two-Staged OOD Edge Completion

During the OOD node generation process, we retain only a limited set of link strengths in $E^{\text{O-I}}$ and $E^{\text{O-O}}$, derived from a few LLM calls. The formation of these edges relies on sampling or retrieval strategies, which inevitably result in missing potential links. To fill this gap, we aim to maximizes the utilization of edge datasets from LLM calls to distill the LLM's edge reasoning ability into lightweight modules for efficient edge completion.

4.3.1 Tuning Local LLM for Node Distinction

Since newly generated OOD nodes exhibit fine-grained textual differences from ID nodes, directly using existing language models for representation extraction may overlook these distinctions. To address this, we first fine-tune a local LLM to better differentiate between the two types of nodes, enabling it to generate more precise representations, particularly for ID nodes and hard OOD nodes. As illustrated in Figure 2, we construct an instruction-tuning task based on the generated OOD nodes and their corresponding anchor ID nodes.

4.3.2 Lightweight Edge Predictor Training

After fine-tuning the local LLM, we use it as a feature extractor for informative text representations. With the local LLM frozen, we then train two lightweight edge predictors based on these features to determine two edge types: OOD-ID and OOD-OOD. To achieve this, we initialize two simple multilayer perceptrons (MLPs) $f_{\text{O-I}}$ and $f_{\text{O-O}}$, as edge predictors. For edge datasets $E^{\text{O-O}}$ and $E^{\text{O-I}}$, we introduce a threshold η, where edge strength scores within the two datasets below η are labeled 0 and scores above or equal to η are labeled 1. The loss function of two edge predictors are defined as follows:

$$\begin{aligned}\mathcal{L}_{\text{O-I}} &= \sum_{(v_i,v_j,c_{ij})\in E^{\text{O-I}}} \text{CE}\left(f_{\text{O-I}}([h_i;h_j]); c_{ij}\right)\\ \mathcal{L}_{\text{O-O}} &= \sum_{(v_i,v_j,c_{ij})\in E^{\text{O-O}}} \text{CE}\left(f_{\text{O-O}}([h_i;h_j]); c_{ij}\right)\end{aligned} \tag{9}$$

Table 1: Node-level graph OOD detection performance comparision on various datasets. The best results are **highlighted** and the second-best results are underlined.

Method	Cora		Arxiv		Products-subset		WikiCS	
	AUROC (↑)	FPR95 (↓)	AUROC (↑)	FPR95 (↓)	AUROC (↑)	FPR95 (↓)	AUROC (↑)	FPR95 (↓)
MSP	91.15	37.73	63.91	90.59	75.45	81.79	79.82	81.66
ODIN	49.80	89.98	55.07	95.07	45.94	96.84	57.15	95.27
Energy	91.40	41.08	51.24	91.61	78.31	80.24	77.62	81.25
GKDE	57.23	88.95	46.48	95.62	63.73	91.14	72.19	82.07
GNNSAFE	92.69	31.24	70.91	86.94	79.89	72.25	83.27	64.44
NodeSAFE	85.58	53.35	72.44	84.27	84.38	58.44	84.42	58.22
GRASP	93.50	29.70	73.93	81.24	92.89	38.47	83.25	64.71
GNNSAFE++	92.74	32.96	74.40	78.28	76.31	80.91	85.55	61.57
NodeSAFE++	87.73	72.01	75.49	75.24	81.19	65.87	83.53	71.14
Ours	**96.09**	**23.14**	**76.56**	**74.17**	**94.58**	**31.64**	**88.84**	**51.13**

where h_i and h_j are the embeddings of v_i and v_j from the frozen local LLM, and c_{ij} denotes the edge label (0 or 1). CE refers to the cross-entropy function. After training, the lightweight predictors can be used to complete potential edges.

5 Experiments

To assess the effectiveness of our proposed LLM-Guard model, we conduct extensive experiments aimed at addressing the following key research questions:

- **RQ1:** Can the proposed method achieve significant performance improvements in the node-level graph OOD detection task?
- **RQ2:** How do the various components of the proposed approach influence performance?
- **RQ3:** How do the generated graph OOD data contribute to improving the model's detection capability? What is the impact of their quantity on detection performance?

5.1 Experimental Settings

Datasets. We conduct experiments on four widely used real-world TAG datasets, covering two citation networks: **Cora** (Liu et al., 2023; Chen et al., 2024) and **Arxiv** (Hu et al., 2020; Liu et al., 2023), a co-purchase network: **Products-subset** (He et al., 2023), and a Wikipedia page network: **WikiCS** (Mernyei and Cangea, 2020; Liu et al., 2023). More dataset details are in Appendix A.

Evaluation Metrics. Following the convention in (Yang et al., 2024b; Wu et al., 2023), we evaluate OOD detection performance using two metrics: AUROC and FPR95. Here, AUROC represents the area under the receiver operating characteristic curve, while FPR95 refers to the false positive rate of OOD samples when the true positive rate of ID samples is fixed at 95%.

Baselines. We mainly compare our method with three groups of baselines: (1) **Classical OOD detection methods**: MSP (Hendrycks and Gimpel, 2016), ODIN (Liang et al., 2017), and Energy (Liu et al., 2020). (2) **Node-level graph OOD detection methods**: GKDE, GNNSAFE (Wu et al., 2023), NODESAFE (Yang et al., 2024b), and GRASP (Ma et al., 2024). These methods take into account the interdependence characteristics of graph structures. (3) **Extra Dataset-Based OOD Methods**: GNNSAFE++ (Wu et al., 2023) and NODESAFE++ (Yang et al., 2024b). These two methods are enhanced variants of GNNSAFE and NODESAFE, incorporating additional OOD datasets to improve detection performance. More implementation details are in Appendix C.

5.2 Performance Comparision (RQ1)

We compare our approach with other classicial baselines over four classical datasets. The experiments is summarized in Table 1. As shown, our method consistently outperforms other detection approaches, demonstrating its effectiveness in constructing high-quality OOD graph data to enhance detection performance. Overall, extra dataset-based OOD detection methods achieve relatively superior performance compared to those relying solely on ID data. This result highlights that incorporating extra OOD data enhances the model's ability to better understand the domain shift between ID and OOD data. However, these methods still fall short of ours, as our approach generates more challenging graph data, thereby facilitating the formation of a tighter and more accurate decision boundary.

5.3 Ablation Study (RQ2)

Table 2: Ablation study on node-level OOD detection performance.

	Cora		Arxiv	
	AUROC (↑)	FPR95 (↓)	AUROC (↑)	FPR95 (↓)
w/o HOOD	94.32	29.40	74.82	76.90
w/o IDG	95.13	18.31	75.56	75.67
w/o EC	94.94	23.21	75.09	76.15
w/o FT	95.12	17.29	75.74	75.31
Ours	96.09	23.14	76.56	74.17

To assess the contribution of each component in our method, we conduct extensive experiments comparing the following variants: (i) **w/o HOOD**: Removes the node generation process and instead incorporates randomly generated OOD nodes from LLMs into training. (ii) **w/o IDG**: Generates OOD node information using only a single ID node and its hard OOD label, without leveraging ID subgraph information. (iii) **w/o EC**: Omits the two-stage OOD edge completion process, relying solely on LLM-predicted OOD-ID and OOD-OOD edges, resulting in fewer connections. (iv) **w/o FT**: Removes instruction tuning for the local LLM, directly using the pretrained LLM with two edge predictors.

As shown in Table 2, our method significantly outperforms the **w/o HOOD** variant, highlighting the advantage of our generated hard OOD nodes over randomly generated ones. This improvement stems from the fact that hard OOD nodes often lie near the decision boundary, encouraging the model to learn a more refined and precise separation. Furthermore, the performance drop in the **w/o IDG** variant suggests that isolated OOD nodes provide limited benefits, emphasizing the importance of incorporating subgraph information. The **w/o EC** variant underperforms compared to LLMGuard, as the lack of edges linking OOD nodes to existing nodes results in structurally distinct OOD samples. This makes them easier for the model to differentiate, ultimately reducing their effectiveness as hard OOD examples.

5.4 More Analysis (RQ3)

Energy score distribution visualization. We compare the energy score density on Cora dataset between our method and the baseline GNNSAFE++. As shown in Figure 5, our approach exhibits a

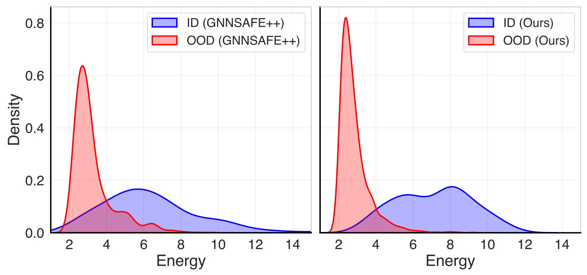


Figure 5: Density of the ID and OOD energy score with baseline GNNSAFE++ (left) and our approach (right).

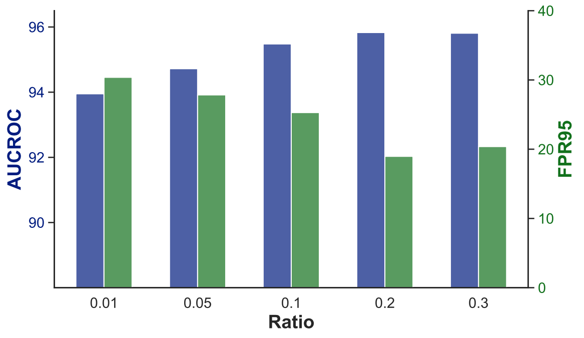


Figure 6: Effect of the proportion of OOD nodes relative to ID nodes on Cora.

greater divergence between ID and OOD data. This suggests that the selected OOD graph data in GNNSAFE++ are of lower quality, as they are often too simple for the detector to differentiate from ID data. In contrast, our method leverages LLMs to generate more challenging OOD graph data, thereby facilitating a more precise decision boundary between ID and OOD data.

The effect of the number of OOD nodes. We fruther examine the impact of the number of generated OOD nodes on OOD detection performance using. As illustrated in Figure 6, we select OOD nodes at proportions of 1%, 5%, 10%, 20%, and 30% relative to the total number of ID nodes. Experiments demonstrate that the AUROC metric initially increases and then stabilizes once the OOD ratio reaches approximately 10%. And FPR95 achieves its best performance at a 20% OOD ratio. These results highlight that the generated OOD data effectively enhance the detector's performance, even when the OOD node ratio is low.

Quality of Generated Nodes. To further demonstrate quality of the generated OOD nodes, we employ sentence-transformers to encode generated OOD node's text content into vector representations. And then we apply t-SNE to project these vectors into a 2D space. Based on this, we compare the distribution of these embeddings with those of randomly generated nodes. As shown in Fig-

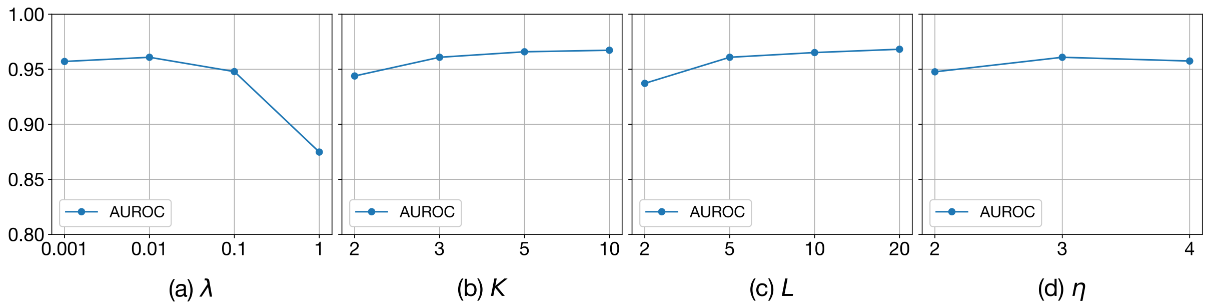


Figure 7: Hyperparameters analysis.

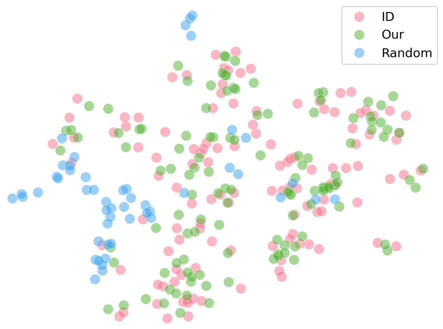


Figure 8: Node content representation visualization.

ure 8, we observe that the generated OOD nodes are more overlapped compared with random generated ones, which confirm that generated hard OOD nodes maintain relevant semantic relationships with ID nodes. Besides, the generated OOD nodes are dispersed across the entire space, which demonstrates their diversity. This can be attributed to the use of hard OOD labels and the subgraph background, which provide diverse information.

Hyperparameter Analysis. We further conduct an extensive hyperparameter analysis, as illustrated in Figure 7. For the weight parameter λ, our approach achieves the best performance when $\lambda = 0.01$. A smaller λ fails to sufficiently leverage OOD information, while a larger value can compromise the main supervised learning objective. Regarding the size of the hard OOD label set K, the performance gradually stabilizes as K increases. This suggests that the most influential OOD nodes are those near the decision boundary, and enlarging K beyond a certain point yields diminishing returns. For the retrieved OOD node set size L, increasing it allows for more OOD-OOD context during LLM calls, but also leads to higher computational costs. When $L > 5$, performance improvements become marginal, so we set the default value to 5. Finally, for the edge strength threshold η, a smaller η favors denser connections, while a larger value promotes a sparser structure. Empirical results indicate that setting $\eta = 3$ yields the best performance.

Table 3: Comparison of edge prediction efficiency on three benchmark datasets (time per edge).

Method	Metric	Cora	WikiCS	Arxiv
Ours	Time (ms)	0.04	0.04	0.04
	AUROC	96.09	88.84	76.56
LLM Pred	Time (s)	10.76	12.65	8.54
	AUROC	95.83	88.23	77.09

Edge Completion Efficiency. To demonstrate our Two-stage Edge Completion strategy can efficient to achieve edge generation process. We compare time cost of our approach and direct LLM calls on datasets of varying sizes. From Table 3, we can conclude our approach achieves performance comparable to direct LLM calls (LLM Pred) while significantly reducing inference time. Since our edge prediction process relies solely on two node embeddings (which can be precalculated) and a 2-layer MLP, the time cost for each node pair remains fixed. In contrast, LLM calls must account for token count and network delays.

6 Conclusion

This paper explores node-level OOD detection on text-attributed graphs. While existing approaches incorporate additional OOD graph data for training, we argue that these collected OOD samples are often of low quality, as their textual attributes and topological structures typically exhibit significant deviations from ID graph data. This discrepancy limits their effectiveness in refining a precise decision boundary. To address this, we propose LLMGuard, a novel framework for generating high-quality OOD graph data. It consists of two key modules: LLM-Guided Hard OOD Node Synthesis and Two-Stage OOD Edge Completion. Extensive experiments demonstrate that our approach significantly enhances detection performance by generating valuable synthetic OOD graph data.

7 Limitations

In this section, we analyze the current limitations of our approach. As an early exploration of leveraging LLMs to generate high-quality OOD graph data, certain modules in our framework can be further refined. In OOD node generation, for efficiency reasons, we primarily use direct node information as background knowledge, which may limit contextual richness. Additionally, in constructing the OOD-OOD strength dataset, exploring more advanced retrieval methods could further improve the quality of OOD-OOD strength dataset.

Acknowledgments

The research reported in this paper was partially supported by the National Key Research and Development Program of China (2022YFC3340900) and the National Natural Science Foundation of China (62037001, 62307032).

References

Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. 2023. Graphllm: Boosting graph reasoning ability of large language model. *arXiv preprint arXiv:2310.05845*.

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

Zhiang Dong, Liya Hu, Jingyuan Chen, Zhihua Wang, and Fei Wu. Comprehend then predict: Prompting large language models for recommendation with semantic and collaborative data. *ACM Transactions on Information Systems*.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426.

Jiarui Feng, Hao Liu, Lecheng Kong, Mingfang Zhu, Yixin Chen, and Muhan Zhang. 2024. Taglas: An atlas of text-attributed graph datasets in the era of large graph and language models. *arXiv preprint arXiv:2406.14683*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2023. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. *arXiv preprint arXiv:2305.19523*.

Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*.

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

Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2024. Large language models on graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering*.

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

Zenan Li, Qitian Wu, Fan Nie, and Junchi Yan. 2022. Graphde: A generative framework for debiased learning and out-of-distribution detection on graphs. *Advances in Neural Information Processing Systems*, 35:30277–30290.

Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. 2017. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*.

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

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. 2020. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 33:21464–21475.

Xiangwei Lv, Jingyuan Chen, Mengze Li, Yongduo Sui, Zemin Liu, and Beishui Liao. 2025a. Grasp the key takeaways from source domain for few shot graph domain adaptation. In *Proceedings of the ACM on Web Conference 2025*, pages 2330–2340.

Xiangwei Lv, Guifeng Wang, Jingyuan Chen, Hejian Su, Zhiang Dong, Yumeng Zhu, Beishui Liao, and Fei Wu. 2025b. Debiased cognition representation learning for knowledge tracing. *ACM Transactions on Information Systems*.

Longfei Ma, Yiyou Sun, Kaize Ding, Zemin Liu, and Fei Wu. 2024. Revisiting score propagation in graph out-of-distribution detection. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Péter Mernyei and Cătălina Cangea. 2020. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*.

Yifei Ming, Ying Fan, and Yixuan Li. 2022. Poem: Out-of-distribution detection with posterior sampling. In *International Conference on Machine Learning*, pages 15650–15665. PMLR.

R OpenAI. 2023. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2(5).

Marc'Aurelio Ranzato, Y-Lan Boureau, Sumit Chopra, and Yann LeCun. 2007. A unified energy-based framework for unsupervised learning. In *Artificial Intelligence and Statistics*, pages 371–379. PMLR.

Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh Chawla, and Chao Huang. 2024. A survey of large language models for graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6616–6626.

Kartik Sharma, Yeon-Chang Lee, Sivagami Nambi, Aditya Salian, Shlok Shah, Sang-Wook Kim, and Srijan Kumar. 2024. A survey of graph neural networks for social recommender systems. *ACM Computing Surveys*, 56(10):1–34.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 491–500.

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

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

Xiaoguang Wang, Chenxu Wang, Huanlong Liu, Mengqin Wang, Tao Qin, and Pinghui Wang. 2025. Figraph: A dynamic heterogeneous graph dataset for financial anomaly detection. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 813–816.

Qitian Wu, Yiting Chen, Chenxiao Yang, and Junchi Yan. 2023. Energy-based out-of-distribution detection for graph neural networks. *arXiv preprint arXiv:2302.02914*.

Tao Wu, Mengze Li, Jingyuan Chen, Wei Ji, Wang Lin, Jinyang Gao, Kun Kuang, Zhou Zhao, and Fei Wu. 2024. Semantic alignment for multimodal large language models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 3489–3498.

Jinluan Yang, Zhengyu Chen, Teng Xiao, Wenqiao Zhang, Yong Lin, and Kun Kuang. 2024a. Leveraging invariant principle for heterophilic graph structure distribution shifts. *arXiv preprint arXiv:2408.09490*.

Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. 2021. Graphformers: Gnn-nested transformers for representation learning on textual graph. *Advances in Neural Information Processing Systems*, 34:28798–28810.

Shenzhi Yang, Bin Liang, An Liu, Lin Gui, Xingkai Yao, and Xiaofang Zhang. 2024b. Bounded and uniform energy-based out-of-distribution detection for graphs. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, Yongfeng Zhang, et al. 2023. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134*, 4(5):7.

Xinping Zhao, Dongfang Li, Yan Zhong, Boren Hu, Yibin Chen, Baotian Hu, and Min Zhang. 2024a. Seer: Self-aligned evidence extraction for retrieval-augmented generation. *arXiv preprint arXiv:2410.11315*.

Xinping Zhao, Yan Zhong, Zetian Sun, Xinshuo Hu, Zhenyu Liu, Dongfang Li, Baotian Hu, and Min Zhang. 2024b. Funnelrag: A coarse-to-fine progressive retrieval paradigm for rag. *arXiv preprint arXiv:2410.10293*.

Xujiang Zhao, Feng Chen, Shu Hu, and Jin-Hee Cho. 2020. Uncertainty aware semi-supervised learning on graph data. *Advances in Neural Information Processing Systems*, 33:12827–12836.

A Dataset Details

Here, we provide additional details about the dataset and summarize its statistics in Table 4.
Cora. The Cora dataset is a widely used citation network in the graph community. Each node represents a paper, while edges model citation relationships between papers. The textual information associated with each paper includes its title and abstract.
Arxiv. The Arxiv dataset is a larger-scale citation network derived from the arXiv platform. Similar to Cora, nodes correspond to research papers, and edges capture citation relationships.
Products-subset. The Products-subset is a copurchase graph where nodes represent product items from Amazon, and edges indicate copurchase relationships between products. Each product is associated with textual attributes such as its title and description.

WikiCS. WikiCS is a Wikipedia page link network, where nodes correspond to Wikipedia pages, and edges represent hyperlinks between them.

B Formulation

In Equ. 6, we assume the following conditional independence properties: (a) $P(t_i \mid y_i^{ood}, y_i, G_i) = P(t_i \mid y_i^{ood}, G_i)$. This assumption implies that the generated node's textual information is independent of the ID label y_i once y_i^{ood} and G_i are given. (b) $P(y_i^{ood} \mid y_i, G_i) = P(y_i^{ood} \mid y_i)$. This suggests that the transformation from y_i to y_i^{ood} does not depend on the graph structure G_i, meaning y_i^{ood} is solely determined by y_i. Based on these assumptions, the derivation proceeds as follows:

$$
\begin{aligned}
&P(t_i^{ood} \mid G_i, y_i) \\
&\overset{(1)}{=} \sum_{y_i^{ood}} P(t_i^{ood}, y_i^{ood} \mid G_i, y_i) \\
&\overset{(2)}{=} \sum_{y_i^{ood}} P(y_i^{ood} \mid G_i, y_i) P(t_i^{ood} \mid G_i, y_i, y_i^{ood}) \\
&\overset{(3)}{=} \sum_{y_i^{ood}} P(y_i^{ood} \mid y_i) P(t_i^{ood} \mid y_i^{ood}, G_i) \\
&\overset{(4)}{=} \sum_{y_i^{ood}} P(y_i^{ood} \mid y_i) P(t_i^{ood} \mid y_i^{ood}, G_i)
\end{aligned}
$$

Each step follows: (1) Law of total probability. (2) Chain rule: $P(A, B) = P(B)P(A \mid B)$. (3) and (4) employ conditional independence assumptions (a) and (b) respectively.

Table 4: Statistics of the datasets.

	Cora	Arxiv	Products-subset	WikiCS
#Nodes	2,708	169,343	54,025	11,701
#Edges	10,556	1,166,243	144,638	216,123
#Labels	7	40	47	10
Domain	citation	citation	co-purchase	wikipedia
ID Class	{0, ···, 3}	{0, ···, 10}	{0, ..., 10}	{0, ..., 3}
OOD Class	{4, 5, 6}	{11, ···, 39}	{11, ..., 46}	{4, ..., 9}

C Implementation Details

We adopt GNNSAFE++ as our backbone and replacing their collected OOD dataset D_{OOD} with our generaed high-quality OOD graph data. After generating high-quality OOD ndoes and structuring them, we utilize the loss function $\mathcal{L}_{final}$ in Equ. 6 to train the detector with $\lambda = 0.01$. We choose $a_{in} = 5$ and $a_{out} = 4$ in Equ. 4. The hard OOD label set size K for each ID label is set to 3,

Input: Node 1 & Node 2

Instruction:
Given information about two nodes: **Node 1** and **Node 2**, determine whether an edge exists between them. Assign a strength score from 1 to 5 based on their relevance, where 5 indicates the highest relevance and 1 represents the lowest relevance.

Answer: strength score

Figure 9: Example prompt for edge prediction using LLMs.

while the retrieved OOD node set size L is set to 5. Strength score threshold η is set to 3. The number of generated OOD nodes is set to 20% of the ID node count. We choose Vicuna-v1.5 as the local LLM, trained using the LoRA method. GPT-4o serves as our default LLM for querying. For a fair comparision, we set the hidden size as 64.

D Graph Neural Networks

Many contemporary graph neural networks (GNNs) utilize the message-passing mechanism to iteratively update node representation by aggregating messages from neighboring nodes (Kipf and Welling, 2016; Hamilton et al., 2017; Veličković et al., 2017) . The update process for a node $v \in V$ at the l-th layer can be expressed as:

$$\mathbf{h}_v^{(l)} = \text{COMB}\left(\mathbf{h}_v^{(l-1)}, \text{AGG}\left(\mathbf{h}_u^{(l-1)} \mid u \in \mathcal{N}(v)\right)\right) \quad (10)$$

Here, $\mathbf{h}_v^{(l)}$ represents the embedding of node v at the l-th layer, and $\mathcal{N}(v)$ denotes the set of neighbors of node v. The functions COMB and AGG correspond to the combination and aggregation operations, respectively, which vary depending on the specific GNN model employed, such as GCN (Kipf and Welling, 2016), GraphSAGE (Hamilton et al., 2017), and GAT (Veličković et al., 2017).

E More Example Prompts

An edge prediction prompt is shown in Figure 9.