How to Make LMs Strong Node Classifiers?

Anonymous ACL submission

Abstract

Language Models (LMs) are increasingly challenging the dominance of domain-specific models, such as Graph Neural Networks (GNNs) and Graph Transformers (GTs), in graph learning tasks. Following this trend, we propose a novel approach that empowers off-the-shelf LMs to achieve performance comparable to state-of-the-art (SOTA) GNNs on node classification tasks, without any architectural modification. By preserving the LM's original architecture, our approach retains a key benefit of LM instruction tuning: the ability to jointly train on diverse datasets, fostering greater flexibility and efficiency. To achieve this, we introduce two key augmentation strategies: (1) enriching LMs' input using topological and semantic retrieval methods, providing richer contextual information, and (2) guiding the LMs' classification process through a lightweight GNN classifier that effectively prunes class candidates. Experiments on real-world datasets show that backbone Flan-T5 LMs equipped with these augmentation strategies outperform SOTA text-output node classifiers and are comparable to top-performing vector-output node classifiers. By bridging the gap between specialized node classifiers and general LMs, this work paves the way for more versatile and widely applicable graph learning models. We will open-source the code upon publication.

1 Introduction

There is a growing trend of utilizing Language Models (LMs) for machine learning tasks across diverse domains. This approach has shown tremendous promise in areas such as vision (Desai and Johnson, 2021), audio (Mittal et al., 2021), and multimodal learning (Alayrac et al., 2022). In graph learning, recent efforts have begun to explore the capabilities of LMs in understanding and processing graph structures. (Wang et al., 2023a) showed that LMs can detect node connectivity and identify cycles, while (Fatemi et al., 2024) explored LMs' ability to evaluate graph scale and identify connected components. Furthermore, InstructGLM (Ye et al., 2023) and LLaGA (Chen et al., 2024b) achieved state-of-the-art (SOTA) performance in text-output node classifiers on Text-Attributed Graphs (TAG) (Zhang et al., 2024a), whose nodes have textual features.

However, both InstructGLM and LLaGA suffer from a fundamental limitation that *compromises the generality of the backbone LM*. Specifically, InstructGLM expands the LM's vocabulary by creating *a unique token for each node*, whose token embeddings are topology-aware node embeddings. It comes at the cost of incompatibility with two important use cases: (1) multi-task learning on diverse datasets, a common strategy for training Foundational Models (Wei et al., 2022; Chung et al., 2024), and (2) certain personalized LM fine-tuning services (Li et al., 2024c) that restrict access to the backbone model architecture/code[1]. LLaGA uses a shared text encoder and a projector to overcome the first limitation but still bears inflexibility when *deploying different LMs* and cannot be applied to LMs without code/architecture access. The above discussion raises a crucial question: *How can off-the-shelf, **text-to-text** instruction-tuned LMs (Raffel et al., 2020) achieve competitive performance in node classification tasks without architectural modifications?*

In stark contrast to (Huang et al., 2023), which suggests that LMs may only interpret graph structures in prompts as contextual paragraphs, our work presents a more optimistic outlook. We aim to overcome this inherent limitation by augmenting the LMs' input while preserving their original architecture. Our proposed model, AUGLM (Aumented Graph Language Model), leverages two key aug-

[1] https://platform.openai.com/docs/guides/fine-tuning

mentation strategies to enhance the LM's ability to process graph data:

- **Relevant Node Retrieval**: In contrast to InstructGLM, which relies on multi-hop ego networks akin to message-passing GNNs for structure-aware contextualization, AUGLM draws inspiration from Graph Transformers (GTs) (Min et al., 2022) and Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Guu et al., 2020). This enables the LM to access long-range structural and semantic information about the target node. We propose two complementary approaches to achieve this: (1) topological retrieval, and (2) prototypical semantic retrieval.
- **Candidate Label Pruning**: To improve LMs' understanding of graph data while maintaining their text-to-text architecture, we convey the guidance from a specialist model, a pretrained lightweight GNN, to the input of LMs via narrowing down the candidate labels. This allows LMs to focus on discerning between closely related candidates, ultimately enhancing the performance.

We extensively evaluate our approach on four real-world TAGs, showing the effectiveness of AUGLM. The results indicate that (1) backbone LMs augmented with AUGLM consistently outperform SOTA text-output classifiers while also matching or surpassing the performance of SOTA vector-output classifiers, and (2) AUGLM can be jointly trained on multiple TAGs without performance degradation. These findings represent a crucial step towards bridging the gap between task-specific node classifiers and more general, fine-tuned LMs, highlighting the potential for a unified model excelling in multiple tasks.

2 Related Work

Due to the space limitation, the most relevant work is briefly introduced as follows. A more comprehensive literature review is in Section H.

LMs for graphs. Recent studies have explored the ability of LMs to understand graph topology by investigating problems such as graph substructure recall (Wang et al., 2024), connectivity (Wang et al., 2023a; Perozzi et al., 2024), node/edge counting (Perozzi et al., 2024), spatial-temporal graphs (Zhang et al., 2024b). Building on these findings, several studies have explored tasks on TAGs, including node classification (Ye et al., 2023; Zhao et al., 2023b; Li et al., 2024b; Qin et al., 2023; Zhou et al., 2025; Sun et al., 2023), link prediction (Brannon et al., 2023; Tan et al., 2024; Liu et al., 2024a,b), transfer/zero-shot learning (Tang et al., 2024; Sun et al., 2025b; Pan et al., 2024; He et al., 2025; Xia and Huang, 2024; Li et al., 2024e), and graph reasoning (Jin et al., 2024b). GIANT (Chien et al., 2022) and GLEM (Zhao et al., 2023a) encode textual features in a graph-aware way. TAPE (He et al., 2024) uses LMs to augment the textual features.

Retrieval-augmented generation (RAG) (Lewis et al., 2020; Karpukhin et al., 2020) enhances LMs by querying external knowledge (Hashimoto et al., 2018). This technique retrieves relevant documents from a large corpus and conditioning the LM on the retrieved documents. Building on this, REALM (Guu et al., 2020) pretrains the retriever and generator end-to-end. Subsequently, RETRO (Borgeaud et al., 2022) scales RAG-enhanced models to large datasets. A crucial component of RAG's success is the objective function proposed by (Shi et al., 2023), which enables the training of retriever even the LMs is a block box. HyDE (Yu et al., 2023) uses hypothetical document generation to improve retrieval in RAG systems. Furthermore, RAG has extended to multimodal settings (Yasunaga et al., 2023). Recently, GraphRAG (Edge et al., 2024) has garnered significant attention; it constructs a Knowledge Graph and then generates responses based on the summaries of communities derived from KG. These advancements have significantly improved generated text's accuracy and contextual relevance.

3 Preliminaries

We use the following notation conventions: bold lower-case letters (e.g,. $\mathbf{x}$) denote column vectors, bold upper-case letters (e.g., $\mathbf{X}$) denote matrices, and calligraphic upper-case letters (e.g., $\mathcal{X}$) denote sets. We use $[\cdot]$ and $[\cdot,\cdot]$ to index vectors and matrices, respectively.

We study the node classification problem on TAGs where each node is associated with textual attributes. A TAG with n nodes is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$, where $\mathcal{V} = \{v_i\}_{i=1}^n$ denotes a set of nodes, and $\mathcal{E} = \{e_{ij}\}_{i,j=1}^n$ is a set of edges where $e_{ij} = 1$ indicates that nodes v_i and v_j are connected; otherwise, $e_{ij} = 0$. $\mathcal{T} = \{t_i\}_{i=1}^n$ indicates the set of node textual attributes. The edges can also be represented by an adjacency matrix

$\mathbf{A} \in \{0,1\}^{n\times n}$, where $\mathbf{A}[i,j] = 1$ if and only if $e_{ij} = 1$. The training and test node labels are denoted by $\mathcal{Y} = \mathcal{Y}_{\text{train}} \cup \mathcal{Y}_{\text{test}} = \{y_i\}_{i=1}^n$, where each label y_i belongs to one of the C classes, i.e., $y_i \in \{1,\dots,C\}, \forall i$. In the semi-supervised setting studied in this paper, the graph structure and training labels $\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{Y}_{\text{train}}$ are accessible, and the goal is to predict the labels of test nodes $\mathcal{Y}_{\text{test}}$.

Personalized PageRank (PPR) (Page, 1999; Jeh and Widom, 2003) ranks all the nodes according to their relevance to a given query node. Specifically, given the adjacency matrix $\mathbf{A}$, the PPR scores $\mathbf{r}_i \in \mathbb{R}^n$ for all nodes concerning the query node v_i are computed iteratively as:

$$\mathbf{r}_i \leftarrow (1-\alpha)\tilde{\mathbf{A}}\mathbf{r}_i + \alpha \mathbf{q}_i \tag{1}$$

where $\alpha \in (0,1)$ is the teleport probability, $\mathbf{q}_i \in \{0,1\}^n$ is a one-hot vector whose i-th entry is 1, $\tilde{\mathbf{A}} = \mathbf{A}\mathbf{D}^{-1}$ is the normalized adjacency matrix, and $\mathbf{D}$ is the degree matrix. Once $\mathbf{r}_i$ converges, the top-K relevant nodes concerning the query node v_i can be identified as follows:

$$\texttt{PPR}(v_i, K) = \{v_j : \mathbf{r}_i[j] \in \text{topK}(\mathbf{r}_i)\} \tag{2}$$

Language models (LMs). We employ autoregressive LMs that predict the next token z_i based on the input sequence t and the context of previously generated tokens $z_{1:i-1}$. The probability of generating a sequence z given the input t is:

$$p_{\text{LM}}(z|t) = \prod_{i=1}^{|z|} p_{\text{LM}}(z_i|t, z_{1:i-1}) \tag{3}$$

Retrieval-augmented generation (RAG) (Lewis et al., 2020; Guu et al., 2020) first retrieves a query t-relevant text d^* from an external corpus $\mathcal{D}$ via a similarity function s_ϕ:

$$d^* = \arg\max_{d\in\mathcal{D}} s_\phi(d,t) \tag{4}$$

s_ϕ is typically implemented as a dual-encoder architecture (Bromley et al., 1993):

$$s_\phi(d,t) = \langle \texttt{Encoder}_\phi(d), \texttt{Encoder}_\phi(t) \rangle \tag{5}$$

Once d^* is retrieved, it is fed into the LM together with the query t: $p_{\text{LM}}(z|d^*, t)$ for generation.

4 Method

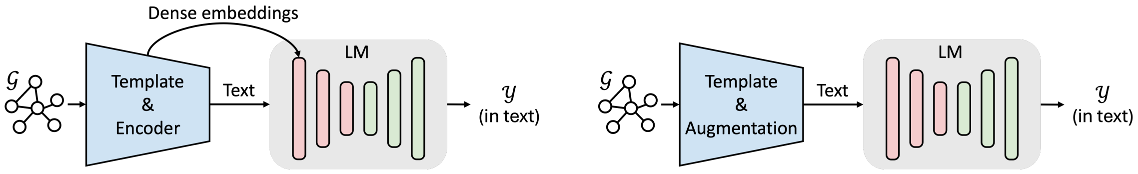
We explore the application of LMs to node classification by reformulating it as a text-to-text task (Raffel et al., 2020). Our method employs carefully designed prompt templates and augmentation techniques to transform graph and ground truth labels into text pairs, enabling LMs to process and be fine-tuned *without* modifying their underlying architecture.

As shown in Figure 1, AUGLM fundamentally differs from InstructGLM (Ye et al., 2023) and LLaGA (Chen et al., 2024b), the current SOTA LM node classifiers. While all three methods utilize prompt templates to transform input graphs into text, InstructGLM and LLaGA explicitly **encode node features into the LM's token embeddings** which can be categorized as **soft prompting** (Lester et al., 2021). In contrast, our approach provides a **data augmentation**-based framework without modifying LM's text-to-text architecture, enabling our model to retain the versatility of the original LM. The following section first details the augmentation techniques developed by this paper and then introduces the templates to incorporate all the augmented textual features.

4.1 Retrieval-based aggregation

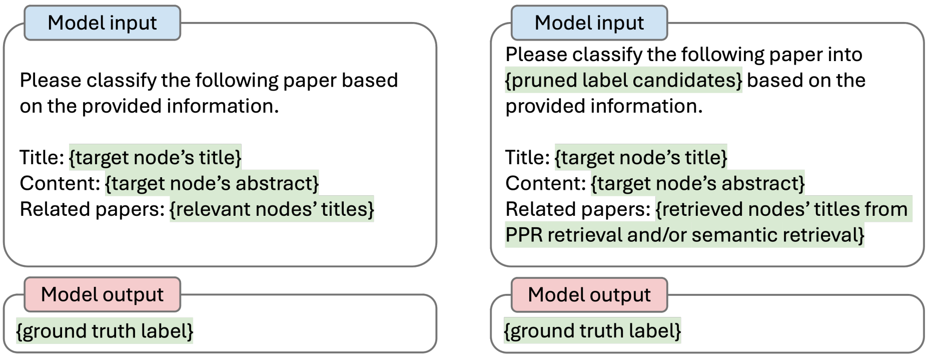
General LMs are not designed to process graph data directly. To overcome this, a common approach is to employ prompt templates to transform graphs and associated tasks into text that LMs can understand. For instance, for the Cora (Sen et al., 2008) literature citation graph, a typical template (Huang et al., 2023; Ye et al., 2023) for node classification, as shown in Figure 2a, consists of three main components: (1) a short task description, (2) the target node's textual features, e.g., its title and abstract, and (3) textual features from relevant nodes.

The success of the message-passing GNNs highlights the importance of the **aggregation** operation, whose typical example is the mean pooling of intermediate node embeddings. A similar spirit is followed for the LM-based classifiers, whose key design is the **selection of relevant nodes**. Existing works (Huang et al., 2023; Ye et al., 2023) select one/multi-hop neighbors as relevant nodes, but we posit that this approach is suboptimal for two reasons. Firstly, not all immediate or extended neighbors are relevant to the target node, which can introduce noise and degrade model performance. Secondly, incorporating multi-hop neighbors can lead to "neighbor explosion" (Hamilton et al., 2017; Chen et al., 2018; Fey et al., 2021), i.e., an exponentially-growing set of "relevant" nodes, resulting in increased computational costs and even leading to the out-of-memory issue. As a response,



(a) InstructGLM and LLaGA. (b) AUGLM (ours).

Figure 1: Comparison of pipelines between the existing LM-based node classifiers and our approach, AUGLM. Unlike InstructGLM and LLaGA, which explicitly encodes graph information into token embeddings as a form of soft prompting, AUGLM maintains the original text-to-text framework of the off-the-shelf LM, offering greater generality and flexibility.



(a) A typical graph-to-text template. (b) Our template with augmented text.

Figure 2: Comparison of a typical graph-to-text template (a) and our template with augmented text features (b).

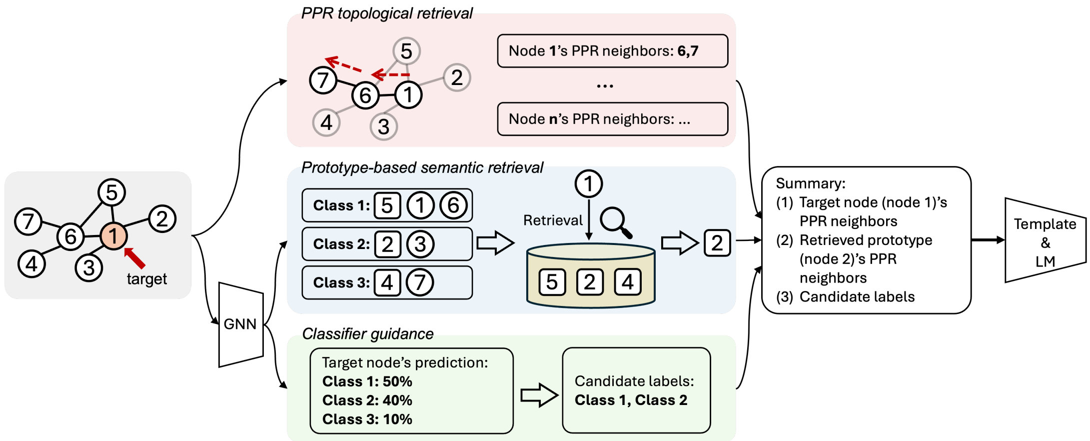


Figure 3: A detailed pipeline of AUGLM. In the semantic retrieval module, rectangles denote the class prototypes.

two novel solutions, *topological retrieval* and *prototypical semantic retrieval*, are proposed to efficiently identify the most informative nodes for the classification tasks.

Topological retrieval. PPR (Page, 1999; Jeh and Widom, 2003) is leveraged for topological retrieval, which has shown great effectiveness in conjunction with GNNs (Klicpera et al., 2019). The success of PPR suggests that its retrieved neighbors may provide more informative context than generic one/multi-hop neighbors. Specifically, for a target node v_i, we select its top-K neighbors $\texttt{PPR}(v_i, K)$ based on their PPR scores (Eqs. (1) and (2), Section 3). Then, the text features from the PPR neighbors are concatenated as the PPR-retrieved text $t_{\text{PPR}} = \bigoplus_{j; v_j \in \texttt{PPR}(v_i, K)} t_j$, where $\oplus$ denotes text concatenation.

It is worth noting that the classic PPR algorithm is computationally expensive for large graphs due to the matrix multiplication (Eq. (1)). However, efficient approximate solutions such as ApproximatePR (Andersen et al., 2006), can be applied to mitigate this issue. Nevertheless, PPR is a topology-based heuristic that inherently cannot leverage textual features or supervision from downstream LMs. To enhance our framework's semantic awareness, we propose a complementary strategy called prototypical semantic retrieval as follows.

Prototypical semantic retrieval. Our semantic retrieval module draws inspiration from two popular techniques: (1) RAG (Lewis et al., 2020; Guu et al., 2020), which retrieves external corpora, and (2) Graph Transformers (Min et al., 2022), which aggregate messages from distant nodes via inner product-based attention weights. For node classification, we treat **the textual features of all nodes except the target node** as a surrogate "external corpus." However, unlike typical question-answering tasks, retrieving textual features from **a single node is often insufficient** for accurate node classification. To address this, we enhance the semantic retrieval by retrieving prototypes, which capture the essence of each class (Snell et al., 2017).

Prototypes (Biehl et al., 2016) are defined as representative examples in classification problems. To obtain prototypes, a lightweight GNN ψ is pretrained and generates a prediction vector for each node: $\tilde{\mathbf{y}}_i = \texttt{GNN}_\psi(v_i, \mathcal{G}) \in \mathbb{R}^C, \forall v_i$. The prediction confidence for each node v_i is defined as: $\texttt{Conf}(v_i) = \max_j \tilde{\mathbf{y}}_i[j]$.

The predicted class-c examples are $\tilde{\mathcal{Y}}_c = \{v_i : \arg\max_j \tilde{\mathbf{y}}_i[j] = c\}$ and their confidence is $\texttt{Conf}_c = \{\texttt{Conf}(v_i) : v_i \in \tilde{\mathcal{Y}}_c\}$. For each class c, the top-N confident examples are selected as prototypes:

$$\mathcal{P}_c = \left\{v_i : v_i \in \tilde{\mathcal{Y}}_c \wedge \texttt{Conf}(v_i) \in \mathrm{topN}\left(\texttt{Conf}_c\right)\right\} \tag{6}$$

For all the classes, there are $N \times C$ prototypes: $\mathcal{P} = \bigcup_{c\in\{1,\dots,C\}} \mathcal{P}_c$. To ensure every document in the corpus $\mathcal{D}$ includes text features from **multiple nodes**, $\mathcal{D}$ is constructed by concatenating the text of **each prototype's PPR neighbors**:

$$\mathcal{D} = \left\{ \bigoplus_{j; v_j \in \texttt{PPR}(v_i, K)} t_j : v_i \in \mathcal{P} \right\} \tag{7}$$

Next, for each target node with its associated text features t_{target}, we compute the prototypically retrieved text using Eq. (4): $t_{\text{proto}} = \arg\max_{d\in\mathcal{D}} s_\phi(d, t_{\text{target}})$. In our experiments, we may use t_{PPR} (from topological retrieval), or t_{proto}, or both by concatenation $t_{\text{PPR}} \oplus t_{\text{proto}}$. For simplicity, we denote the final retrieved text as t_{retri}.

4.2 Classifier guidance

Recent studies (Huang et al., 2023; Fatemi et al., 2024; Chen et al., 2024a) highlighted mainstream LMs' limited understanding of graph topology. While InstructGLM (Ye et al., 2023) and LLaGA (Chen et al., 2024b) address this limitation by incorporating topology-aware node embeddings (e.g., from a pretrained or parameter-free GNN) into the LM's token embeddings, this approach necessitates **modifications to the LM's architecture**. We propose an alternative method that conveys guidance from a pretrained GNN into the **input text of LMs**, thereby preserving the LM's original architecture. Concretely, such guidance is to **prune the classification candidates**.

We repurpose the pretrained $\texttt{GNN}_\psi$ from the prototypical semantic retrieval module. For each node v_i, we identify and save the top-I predicted labels:

$$\mathcal{L}_i = \{j : \tilde{\mathbf{y}}_i[j] \in \mathrm{topI}\left(\tilde{\mathbf{y}}_i\right)\} \in \{1, \dots, C\}^I \tag{8}$$

where $I < C$. For datasets in the experiments, the `IndexToLabel` maps are available, which map numerical labels to their corresponding text. The pruned label candidates for node v_i can be presented as concatenated text: $t_{\text{candidates}} = \oplus_{i\in\mathcal{L}_i} \texttt{IndexToLabel}(i)$. The integration of pruned candidates into the template is detailed in Section 4.3.

By focusing on a more relevant set of candidate labels, valuable topology-aware inductive bias from the GNN is incorporated into the LM's input, thereby enhancing its node classification performance without altering its architecture.

4.3 Overall template

Our augmented training samples include three key elements: (1) the target node's text t_{target}, (2) the retrieved nodes' text t_{retri}, and (3) the pruned label candidates $t_{\text{candidates}}$. We collectively denote these elements as $t_{\text{in}} = (t_{\text{target}}, t_{\text{retri}}, t_{\text{candidates}})$. LM's prediction probability for the target sequence y_{target} is based on Eq. (3) whose input text t is t_{in}. Figure 2b presents an exemplar template for the Cora dataset, showcasing the integration of t_{target}, t_{retri}, and $t_{\text{candidates}}$. The selection of the backbone LM will be detailed in Section 5. Appendix C contains a full list of templates. Note that we exclude the "abstracts" of the retrieved nodes to prevent exceeding the maximum input length of most LMs. We utilize only this template's "model input" part for evaluation.

4.4 Training

Our framework includes three parameterized modules that require training or fine-tuning: (1) GNNs for generating prototypes and candidate label pruning, as described in Sections 4.1 and 4.2, (2) the encoder ϕ from the semantic retriever, defined in Eq. 5, and (3) the backbone LM, utilized in Eq. 3. The GNNs from Sections 4.1 and 4.2 can be shared, and their training is independent of the other modules, which are supervised by ground truth labels. This process is detailed in Appendix A.

One of the standard LM's losses, the average token-wise negative log-likelihood (NLL), is used. For a target node, the loss is:

$$\mathcal{L}_{\text{NLL}}\left(p_{\text{LM}}(y_{\text{target}}|t_{\text{in}}), y_{\text{target}}\right) \tag{9}$$

To train the semantic retriever, we employ a distribution-matching loss. For a given target node's text t_{target}, its retrieval probability for a prototype text $t \in \mathcal{D}$ is:

$$p_\phi(t|t_{\text{target}}) = \frac{e^{s_\phi(t,t_{\text{target}})}}{\sum_{t'\in\mathcal{D}} e^{s_\phi(t',t_{\text{target}})}} \tag{10}$$

Next, an LM-supervised empirical distribution is:

$$\tilde{p}_{\text{LM}}(t|t_{\text{target}}, y_{\text{target}}) = \frac{e^{p_{\text{LM}}(y_{\text{target}}|t_{\text{in}})}}{\sum_{t'\in\mathcal{D}} e^{p_{\text{LM}}\left(y_{\text{target}}|t'_{\text{in}}\right)}} \tag{11}$$

where $t_{\text{in}} = (t_{\text{target}}, t, t_{\text{candidates}})$ and $t'_{\text{in}} = (t_{\text{target}}, t', t_{\text{candidates}})$. This distribution represents the **normalized importance of each prototype text** $t \in \mathcal{D}$ based on the **LM's likelihood of generating the ground truth text** y_{target}. We use $\tilde{p}_{\text{LM}}$ to distinguish this distribution from the generation probability in Eq. (3). The distribution matching loss is the Kullback-Leibler (KL) divergence between the retrieval and the LM-supervised distributions:

$$\text{KL}\left(\mathtt{sg}\left(\tilde{p}_{\text{LM}}\left(\cdot|t_{\text{target}}, y_{\text{target}}\right)\right) \| p_\phi(\cdot|t_{\text{target}})\right) \tag{12}$$

This loss aims to align the retrieval probability of prototype text $t \in \mathcal{D}$ with its importance in facilitating the LM's generation of the label text y_{target} for the target node. The stop gradient operator sg ensures that the loss Eq. (12) only updates the semantic retriever ϕ but keeps the LM's parameters θ frozen. This objective has been used by previous works (Shi et al., 2023; Izacard et al., 2023) without a thorough analysis. We provide an in-depth examination of its implications in Appendix B.

Notably, Eq. (11) requires $|\mathcal{D}|$ inferences of the LM due to the denominator. However, the LM is fine-tuned via the NLL loss, Eq. (9), only for the most relevant prototype, $\arg\max_{d\in\mathcal{D}} s_\phi(d, t_{\text{target}})$. Consequently, each update step involves $|\mathcal{D}|$ forward passes but only one backpropagation. To reduce the computational overhead associated with $|\mathcal{D}|$ inferences, we can use a sampling strategy: selecting the top-M samples to form a retrieval mini-batch $\mathcal{D}_M = \{t : t \in \text{topM}_{t'\in\mathcal{D}} s_\phi(t', t_{\text{target}})\}$. By replacing $\mathcal{D}$ with $\mathcal{D}_M$ in Eqs. (10) and (11), the retrieval and the LM-supervised distributions can be computed "in-batch", reducing the inference times from $|\mathcal{D}|$ to M.

Algorithm 1 (Appendix D) outlines a step-by-step process for fine-tuning AuGLM, processing one training node per step. This procedure can be readily extended to mini-batch settings.

4.5 Model complexity

AuGLM consists of three parameterized modules: (1) a GNN ψ, (2) the semantic retriever ϕ, and (3) the backbone LM θ. Notably, ψ and ϕ are lightweight, whose number of parameters is only $1/30$ to $1/3$ of the number of LM θ parameters. Compared to the SOTA InstructGLM (Ye et al., 2023), AuGLM has an additional module ϕ, resulting in a slightly increased number of parameters. For training, the GNN ψ can be pretrained, and the PPR scores can be precomputed. The training of θ relies on the retrieved text from ϕ, while the training of ϕ requires $\tilde{p}_{\text{LM}}(\cdot|t_{\text{target}}, y_{\text{target}})$, which is obtained through forward inference of θ. Importantly, computational graphs (used for gradient computation) of θ and ϕ are **independent**. When training ϕ, the stop gradient operator sg ensures θ has no gradient. As a result, the cost of backpropagation is close to the sum of the cost for updating the LM θ and the semantic encoder ϕ, separately.

4.6 Discussion on the flexibility of AuGLM

We noticed that LLaGA (Chen et al., 2024b) improves the generality of soft prompting-based solutions by a shared text encoder and a projector. Here, we compare the flexibility and generality of LLaGA and our proposed AuGLM to illustrate our unique contribution better.

- LlaGA **can achieve** 0**-shot transfer learning**, e.g., training on the Cora dataset and testing on the Amazon dataset. However, LLaGA **cannot switch the LM seamlessly** because it requires modifying the LM's code and architecture, e.g., including its graph encoder's output into the LM's token embeddings. The application of LLaGA is also limited if there is no access to the LM's architecture or code.

- Our AUGLM **cannot achieve 0-shot transfer learning** because we need a trained GNN to provide reliable label candidates for text input augmentation. However, thanks to such a data augmentation paradigm, AUGLM **can switch the LM seamlessly** as long as the LM works in a text-to-text manner.

5 Experiments

This section introduces the experimental setups, baseline methods, effectiveness studies, ablation studies, and multi-task training. Efficiency and hyperparameter studies are in the Appendix.

5.1 Setup

Following (He et al., 2024; Ye et al., 2023), we evaluate our approach on four benchmark datasets: Cora (Sen et al., 2008), Pubmed (Sen et al., 2008), ogbn-arxiv (Hu et al., 2020), and a subset of ogbn-products (Hu et al., 2020; He et al., 2024). The dataset statistics are in Table 5.

Our implementation employs two pretrained all-MiniLM-L6-v2 models (Wang et al., 2020) as the the semantic retriever ϕ (Eq. (5)) and the text encoder for GNN ψ (Eq. (13)). We set the PPR teleport probability $\alpha = 0.1$. We employ a 3-layer GraphSAGE (Hamilton et al., 2017) with a hidden dimension of 256 as ψ. Our hyperparameters include $K = 5$ PPR neighbors, $N = 10$ prototypes, and $M = 8$ samples for LM inference. The number of label candidates I is searched from $\{2, 3\}$. Flan-T5-small/base/large (Chung et al., 2022) are used as the backbone LM θ with templates detailed in Section C.

5.2 Comparison with state-of-the-arts

This section presents the comparison between AUGLM and SOTA baselines. We categorize models into two groups: (1) vector-output models which output a vector with dimension equal to the number of classes, and (2) text-output models, whose output is text. Specifically, results from GCN (Kipf and Welling, 2017), GraphSAGE (Hamilton et al., 2017), GLEM (Zhao et al., 2023a)+RevGAT (Li et al., 2021), InstructGLM (Ye et al., 2023), LLaGA (Chen et al., 2024b), ENGINE (Zhu et al., 2024), SimTeG (Duan et al., 2023), GraphAdapter (Huang et al., 2024), OFA (Liu et al., 2024a), LLM4GraphTopology (short as LLM4GT) (Sun et al., 2023), GraphPrompter (Liu et al., 2024b), and GraphICL (Sun et al., 2025b) are reported according to the leaderboards (detailed in Appendix) and their papers. The results for TAPE+RevGAT, GIANT (Chien et al., 2022)+RevGAT, GIANT+GCN, and DeBERTa (He et al., 2021) are reported from (He et al., 2024). Note that all the selected models are **fine-tuned** on the training set; methods focusing on transfer learning (e.g., UniGraph (He et al., 2025) and ZeroG (Li et al., 2024e)) are not included for fairness but are introduced in the related work. Mean and standard deviation over 5 runs are reported. For text-output models, accuracy is evaluated by checking the exact matching between model's output and ground truth text.

Table 1 presents a comparison between AUGLM and SOTAs. AUGLM consistently outperforms InstructGLM and LLaGA, **achieving new SOTA performance** among text-output node classifiers. Notably, this superior performance is achieved without modifying any LMs' architecture, demonstrating the effectiveness of our approach. Furthermore, AUGLM exhibits **competitive performance compared to the best vector-output models**. Specifically, on Cora, Pubmed, and ogbn-arxiv datasets, AUGLM performs closely to that of the SOTA vector-output models. Furthermore, on the ogbn-products dataset, AUGLM surpasses the performance of the best vector-output model, TAPE.

5.3 Ablation study

To evaluate the contribution of each key component in AUGLM, we conducted an ablation study on three crucial modules: (1) topological retrieval, (2) semantic retrieval, and (3) candidate label pruning. In this subsection, Flan-T5-small is used. The results in Table 2 demonstrate that each module consistently improves performance across all datasets. Notably, our analysis reveals that the relative importance of each component varies across different datasets. For instance, candidate label pruning greatly impacts performance for the Cora dataset, whereas its effect is less pronounced for the ogbn-products dataset. This variation in component importance underscores the adaptability of our approach, which can effectively accommodate diverse datasets with different characteristics.

5.4 Multi-task training

One of the key advantages of pure text-to-text instruction tuning is that a single model can be trained

Table 1: Accuracy (%) comparison between AUGLM and existing SOTA models. The best-performing **vector-output** and **text-output** models on each dataset are highlighted in **blue** and **red**, respectively.

	Method	Cora	Pubmed	ogbn-arxiv	ogbn-products
Vector-output	GCN	87.78±0.96	88.90±0.32	73.60±0.18	75.64±0.21
	GraphSAGE	86.51±2.36	89.08±0.28	73.88±0.33	76.04±0.25
	GLEM + RevGAT	88.56±0.60	94.71±0.20	76.97±0.19	–
	GIANT + RevGAT	83.53±0.38	85.02±0.48	75.90±0.19	71.89±0.30
	GIANT + GCN	84.23±0.53	84.19±0.50	73.29±0.10	69.77±0.42
	TAPE + RevGAT	**92.90±3.07**	**96.18±0.53**	77.50±0.12	**82.34±0.36**
	ENGINE	91.48±0.32	–	76.02±0.29	–
	SimTeG+RevGAT	–	–	77.04±0.13	–
	GraphAdapter	–	–	77.07±0.15	–
	OFA	74.76±1.12	78.21±0.71	**77.51±0.17**	–
	LLM4GT	89.16±0.96	94.72±0.56	–	–
Text-output	DeBERTa	76.06±3.78	94.94±0.46	73.61±0.04	72.97±0.23
	GraphPrompter	82.26	94.80	75.61	79.54
	InstructGLM	90.77±0.52	94.62±0.13	75.70±0.12	–
	LLaGA	89.85	95.06	76.66	–
	GraphICL	83.58	93.18	73.68	81.48
	AUGLM (T5-small)	91.14±0.55	94.80±0.15	75.39±0.21	81.73±0.08
	AUGLM (T5-base)	91.24±0.46	95.03±0.35	**76.80±0.14**	81.91±0.11
	AUGLM (T5-large)	**91.51±0.26**	**95.16±0.18**	76.00±0.23	**82.90±0.10**

Table 2: Ablation study results (accuracy %). T, S, and L denote topological retrieval, semantic retrieval, and label pruning, respectively. ↓ indicates accuracy drop compared to the full model (T+S+L).

Model	Cora	Pubmed	ogbn-arxiv	ogbn-products
T+S	85.52 (↓5.62)	94.40 (↓0.40)	72.91 (↓2.48)	79.83 (↓1.90)
T+L	87.27 (↓3.87)	94.32 (↓0.48)	73.79 (↓1.60)	81.05 (↓0.68)
S+L	90.25 (↓0.89)	94.26 (↓0.54)	73.46 (↓1.93)	79.06 (↓2.67)
T+S+L	**91.14**	**94.80**	**75.39**	**81.73**

Table 3: Joint vs. separate training (accuracy %).

Training	Cora	Pubmed	ogbn-arxiv	ogbn-products
Joint	91.52	94.52	74.87	82.29
Separate	91.14	94.80	75.39	81.73

on multiple tasks with the same input-output format. To verify this, AUGLM with Flan-T5-small is jointly trained on diverse datasets: Cora, Pubmed, ogbn-arxiv, and ogbn-products. The results in Table 3 show that the jointly trained model achieves performance comparable to models trained separately on each individual dataset. We observe that on some datasets, such as Cora and ogbn-products, the jointly trained model even outperforms its dataset-specific counterparts.

These findings suggest that our approach can effectively **handle multiple graph datasets using a single model**, without incurring great performance losses compared to models trained individually. This capability is crucial for efficient model deployment when dealing with diverse graphs. In contrast, other approaches, such as InstructGLM, require the addition of a large token dictionary to accommodate all nodes in the joint dataset, which hinders their ability to achieve similar generality. Moreover, most vector-output models, including TAPE, are limited by their predefined input-output dimensions, making them inflexible and unable to handle multiple datasets.

6 Conclusion

We introduce a novel framework AUGLM for node classification on TAGs via text-to-text instruction-tuning. Our approach is built upon two key innovations: (1) topological and semantic retrieval of relevant nodes and (2) a lightweight GNN textual guidance. Extensive experimental results demonstrated (1) the effectiveness of our framework, which consistently outperformed the best text-output node classifiers while achieving performance comparable to SOTA vector-output node classifiers, and (2) the flexibility of AUGLM , which can be jointly trained over multiple datasets without performance drop. These findings suggest a promising direction for harnessing the power of LMs in graph learning tasks.

7 Limitations

One limitation of this work is the need for manual definition of prompt templates in Table 4. A promising direction for future research is to develop methods for automatically searching for optimal templates in a data-driven manner. Another limitation is the requirement for pretraining a GNN ψ on each dataset, which stems from the inherent challenges of language models in understanding graph data and limits our model to be adapted to zero-shot learning scenarios. Addressing this limitation by developing more powerful language models capable of handling graph data is a challenging yet impactful area of future work, which can lead to instruction-tuning only, highly generalizable graph foundation models.

8 Broader Impact

This paper presents work that aims to advance the fields of language models (LMs) and graph machine learning (GML). Thus, the broader societal impact of this research aligns with prior work in both LMs and GML.

While our proposed approach does not target any specific high-stakes application domain, it may be adopted in settings such as recommendation systems and social network analysis. As such, downstream uses could inherit ethical considerations, including privacy, fairness, and potential misuse. Mitigating such issues is an open challenge shared by the broader community, and we encourage future work to evaluate fairness, robustness, and interpretability as these models are adopted for critical tasks.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L. Menick, Sebastian Borgeaud, and 8 others. 2022. Flamingo: a visual language model for few-shot learning. In *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*.

Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. 2006. Local graph partitioning using pagerank vectors. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 475–486. IEEE Computer Society.

Michael Biehl, Barbara Hammer, and Thomas Villmann. 2016. Prototype-based models in machine learning. *Wiley Interdisciplinary Reviews: Cognitive Science*, 7(2):92–111.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, and 9 others. 2022. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

William Brannon, Suyash Fulay, Hang Jiang, Wonjune Kang, Brandon Roy, Jad Kabbara, and Deb Roy. 2023. Congrat: Self-supervised contrastive pretraining for joint graph and text embeddings. *CoRR*, abs/2305.14321.

Jane Bromley, James W. Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using A "siamese" time delay neural network. *Int. J. Pattern Recognit. Artif. Intell.*, 7(4):669–688.

Jianfei Chen, Jun Zhu, and Le Song. 2018. Stochastic training of graph convolutional networks with variance reduction. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 941–949. PMLR.

Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. 2024a. Graphwiz: An instruction-following language model for graph computational problems. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 353–364. ACM.

Runjin Chen, Tong Zhao, Ajay Kumar Jaiswal, Neil Shah, and Zhangyang Wang. 2024b. Llaga: Large language and graph assistant. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S. Dhillon. 2022. Node feature extraction by self-supervised multi-scale neighborhood prediction. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, and 12 others. 2022. Scaling instruction-finetuned language models. *CoRR*, abs/2210.11416.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2024. Scaling instruction-finetuned language models. *J. Mach. Learn. Res.*, 25:70:1–70:53.

Julien Delile, Srayanta Mukherjee, Anton Van Pamel, and Leonid Zhukov. 2024. Graph-based retriever captures the long tail of biomedical knowledge. *arXiv preprint arXiv:2402.12352*.

Karan Desai and Justin Johnson. 2021. Virtex: Learning visual representations from textual annotations. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 11162–11173. Computer Vision Foundation / IEEE.

Junnan Dong, Qinggang Zhang, Xiao Huang, Keyu Duan, Qiaoyu Tan, and Zhimeng Jiang. 2023. Hierarchy-aware multi-hop question answering over knowledge graphs. In *Proceedings of the ACM web conference 2023*, pages 2519–2527.

Keyu Duan, Qian Liu, Tat-Seng Chua, Shuicheng Yan, Wei Tsang Ooi, Qizhe Xie, and Junxian He. 2023. Simteg: A frustratingly simple approach improves textual graph learning. *arXiv preprint arXiv:2308.02565*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2024. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Jure Leskovec. 2021. Gnnautoscale: Scalable and expressive graph neural networks via historical embeddings. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 3294–3304. PMLR.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034.

Tatsunori B. Hashimoto, Kelvin Guu, Yonatan Oren, and Percy Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 10073–10083.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2024. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yufei He, Yuan Sui, Xiaoxin He, and Bryan Hooi. 2025. Unigraph: Learning a unified cross-domain foundation model for text-attributed graphs. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.1, KDD 2025, Toronto, ON, Canada, August 3-7, 2025*, pages 448–459. ACM.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

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. *CoRR*, abs/2005.00687.

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2025. GRAG: graph retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pages 4145–4157. Association for Computational Linguistics.

Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. 2023. Can llms effectively leverage graph structural information: When and why. *CoRR*, abs/2309.16595.

Xuanwen Huang, Kaiqiao Han, Yang Yang, Dezheng Bao, Quanjin Tao, Ziwei Chai, and Qi Zhu. 2024. Can GNN be good adapter for llms? In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, pages 893–904. ACM.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 874–880. Association for Computational Linguistics.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

Glen Jeh and Jennifer Widom. 2003. Scaling personalized web search. In *Proceedings of the Twelfth International World Wide Web Conference, WWW 2003, Budapest, Hungary, May 20-24, 2003*, pages 271–279. ACM.

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

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

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and Jiawei Han. 2024b. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 163–184. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Thomas N. Kipf and Max Welling. 2016. Variational graph auto-encoders. *CoRR*, abs/1611.07308.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then propagate: Graph neural networks meet personalized pagerank. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Dawei Li, Shu Yang, Zhen Tan, Jae Young Baik, Sukwon Yun, Joseph Lee, Aaron Chacko, Bojian Hou, Duy Duong-Tran, Ying Ding, and 1 others. 2024a. Dalk: Dynamic co-augmentation of llms and kg to answer alzheimer's disease questions with scientific literature. *arXiv preprint arXiv:2405.04819*.

Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. 2021. Training graph neural networks with 1000 layers. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6437–6449. PMLR.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3538–3545. AAAI Press.

Rui Li, Jiwei Li, Jiawei Han, and Guoyin Wang. 2024b. Similarity-based neighbor selection for graph llms. *CoRR*, abs/2402.03720.

Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, Rui Kong, Yile Wang, Hanfei Geng, Jian Luan, Xuefeng Jin, Zilong Ye, Guanjing Xiong, Fan Zhang, Xiang Li, and 6 others. 2024c. Personal LLM agents: Insights and survey about the capability, efficiency and security. *CoRR*, abs/2401.05459.

Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. 2024d. A survey of graph meets large language model: progress and future directions. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 8123–8131.

Yuhan Li, Peisong Wang, Zhixun Li, Jeffrey Xu Yu, and Jia Li. 2024e. Zerog: Investigating cross-dataset zero-shot transferability in graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 1725–1735. ACM.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. 2024a. One for all: Towards training one graph model for all classification tasks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S Yu, and 1 others. 2025. Graph foundation models: Concepts, opportunities and challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zheyuan Liu, Xiaoxin He, Yijun Tian, and Nitesh V. Chawla. 2024b. Can we soft prompt llms for graph learning tasks? In *Companion Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, Singapore, May 13-17, 2024*, pages 481–484. ACM.

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

Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. 2022. Transformer for graphs: An overview from architecture perspective. *CoRR*, abs/2202.08455.

Gautam Mittal, Jesse H. Engel, Curtis Hawthorne, and Ian Simon. 2021. Symbolic music generation with diffusion models. In *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, pages 468–475.

Sai Munikoti, Anurag Acharya, Sridevi Wagle, and Sameera Horawalavithana. 2023. Atlantic: Structure-aware retrieval-augmented language model for interdisciplinary science. *arXiv preprint arXiv:2311.12289*.

Lawrence Page. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Technical Report.

Bo Pan, Zheng Zhang, Yifei Zhang, Yuntong Hu, and Liang Zhao. 2024. Distilling large language models for text-attributed graph learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024, Boise, ID, USA, October 21-25, 2024*, pages 1836–1845. ACM.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*.

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

Yijian Qin, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2023. Disentangled representation learning with large language models for text-attributed graphs. *CoRR*, abs/2310.18152.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

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.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Mag.*, 29(3):93–106.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. REPLUG: retrieval-augmented black-box language models. *CoRR*, abs/2301.12652.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.

Li Sun, Zhenhao Huang, Suyang Zhou, Qiqi Wan, Hao Peng, and Philip Yu. 2025a. Riemanngfm: Learning a graph foundation model from riemannian geometry. In *Proceedings of the ACM on Web Conference 2025*, pages 1154–1165.

Shengyin Sun, Yuxiang Ren, Chen Ma, and Xuecang Zhang. 2023. Large language models as topological structure enhancers for text-attributed graphs. *CoRR*, abs/2311.14324.

Yuanfu Sun, Zhengnan Ma, Yi Fang, Jing Ma, and Qiaoyu Tan. 2025b. Graphicl: Unlocking graph learning potential in llms through structured prompt design. In *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New*

Mexico, USA, April 29 - May 4, 2025, pages 2440–2459. Association for Computational Linguistics.

Yanchao Tan, Hang Lv, Xinyi Huang, Jiawei Zhang, Shiping Wang, and Carl Yang. 2024. Musegraph: Graph-oriented instruction tuning of large language models for generic graph mining. *CoRR*, abs/2403.04780.

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, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 491–500. ACM.

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

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023a. Can language models solve graph problems in natural language? In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788.

Xintao Wang, Qianwen Yang, Yongting Qiu, Jiaqing Liang, Qianyu He, Zhouhong Gu, Yanghua Xiao, and Wei Wang. 2023b. Knowledgpt: Enhancing large language models with retrieval and storage access on knowledge bases. *arXiv preprint arXiv:2308.11761*.

Yanbang Wang, Hejie Cui, and Jon M. Kleinberg. 2024. Microstructures and accuracy of graph recall by large language models. *CoRR*, abs/2402.11821.

Zehong Wang, Zheyuan Liu, Tianyi Ma, Jiazheng Li, Zheyuan Zhang, Xingbo Fu, Yiyang Li, Zhengqing Yuan, Wei Song, Yijun Ma, and 1 others. 2025. Graph foundation models: A comprehensive survey. *arXiv preprint arXiv:2505.15116*.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Lianghao Xia and Chao Huang. 2024. Anygraph: Graph foundation model in the wild. *CoRR*, abs/2408.10700.

Michihiro Yasunaga, Armen Aghajanyan, Weijia Shi, Richard James, Jure Leskovec, Percy Liang, Mike Lewis, Luke Zettlemoyer, and Wen-Tau Yih. 2023. Retrieval-augmented multimodal language modeling. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 39755–39769. PMLR.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2023. Natural language is all a graph needs. *CoRR*, abs/2308.07134.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate rather than retrieve: Large language models are strong context generators. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Delvin Ce Zhang, Menglin Yang, Rex Ying, and Hady W. Lauw. 2024a. Text-attributed graph representation learning: Methods, applications, and challenges. In *Companion Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, Singapore, May 13-17, 2024*, pages 1298–1301. ACM.

Jiawei Zhang. 2023. Graph-toolformer: To empower llms with graph reasoning ability via prompt augmented by chatgpt. *arXiv preprint arXiv:2304.11116*.

Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Junnan Dong, Hao Chen, Yi Chang, and Xiao Huang. 2025. A survey of graph retrieval-augmented generation for customized large language models. *arXiv preprint arXiv:2501.13958*.

Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Yijian Qin, and Wenwu Zhu. 2024b. Llm4dyg: Can large language models solve spatial-temporal problems on dynamic graphs? In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 4350–4361. ACM.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2023a. Learning on large-scale text-attributed graphs via variational inference. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael M. Bronstein, Zhaocheng Zhu, and Jian Tang. 2023b. Graphtext: Graph reasoning in text space. *CoRR*, abs/2310.01089.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*.

Ruochen Zhao, Hailin Chen, Weishi Wang, Fangkai Jiao, Xuan Long Do, Chengwei Qin, Bosheng Ding, Xiaobao Guo, Minzhi Li, Xingxuan Li, and 1 others. 2023c. Retrieving multimodal information for augmented generation: A survey. *arXiv preprint arXiv:2303.10868*.

Chuang Zhou, Zhu Wang, Shengyuan Chen, Jiahe Du, Qiyuan Zheng, Zhaozhuo Xu, and Xiao Huang. 2025. Taming language models for text-attributed graph learning with decoupled aggregation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 3463–3474. Association for Computational Linguistics.

Xi Zhu, Haochen Xue, Ziwei Zhao, Wujiang Xu, Jingyuan Huang, Minghao Guo, Qifan Wang, Kaixiong Zhou, and Yongfeng Zhang. 2025a. Llm as gnn: Graph vocabulary learning for text-attributed graph foundation models. *arXiv preprint arXiv:2503.03313*.

Yun Zhu, Haizhou Shi, Xiaotang Wang, Yongchao Liu, Yaoke Wang, Boci Peng, Chuntao Hong, and Siliang Tang. 2025b. Graphclip: Enhancing transferability in graph foundation models for text-attributed graphs. In *Proceedings of the ACM on Web Conference 2025*, pages 2183–2197.

Yun Zhu, Yaoke Wang, Haizhou Shi, and Siliang Tang. 2024. Efficient tuning and inference for large language models on textual graphs. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pages 5734–5742. ijcai.org.

Zulun Zhu, Tiancheng Huang, Kai Wang, Junda Ye, Xinghe Chen, and Siqiang Luo. 2025c. Graph-based approaches and functionalities in retrieval-augmented generation: A comprehensive survey. *arXiv preprint arXiv:2504.10499*.

Appendix

This appendix is organized as follows

- Section A: introduction of the architecture and training of GNNs used in this paper.
- Section B: interpretation of the distribution matching loss.
- Section C: templates used in this paper for node classification.
- Section D: the training algorithm of AuGLM.
- Section E: detailed dataset statistics and their leaderboards.
- Section F: hyperparameters and pretrained backbone models.
- Section G: additional experiments.
  - Section G.1: additional efficiency studies.
  - Section G.2: additional experiments on the backbone GNN selections, topological and semantic retrievers, and the PPR steps.
  - Section G.3 additional experiments on the link prediction tasks.
- Section H: extended related work

A Architecture and Training of the Graph Neural Network ψ

In our setting, semi-supervised node classification problem, $\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{Y}_{\text{train}}$ are accessible during training. Since Graph Neural Networks (GNNs) are not inherently capable of processing textual features, a pretrained text encoder is used to generate d-dimensional dense embeddings for each node

$$\texttt{Encoder}_{\psi_1}(t_i) = \mathbf{h}_i^{(0)} \in \mathbb{R}^d, \forall i \in 1, \ldots, n \quad (13)$$

In our implementation, the text encoder is all-MiniLM-L6-v2, a member of the Sentence Transformers. Subsequently, we apply a standard graph neural network, GraphSAGE (Hamilton et al., 2017), whose iterative architecture is

$$\mathbf{h}_i^{(l)} = \sigma^{(l)}\Big(\texttt{MEAN}\Big(\texttt{Message}_i\Big) \cdot \mathbf{W}^{(l)}\Big) \quad (14)$$

$$\texttt{Message}_i = \{\mathbf{h}_i^{(l-1)}\} \cup \{\mathbf{h}_j^{(l-1)} : (v_i, v_j) \in \mathcal{E}\} \quad (15)$$

where $\sigma^{(l)}$ is the activation function and $\mathbf{W}^{(l)} \in \mathbb{R}^{d\times d}$ is the learnable parameter of each layer. For an L-layer network, in the last layer, $\sigma^{(L)}$ is $\texttt{Softmax}$ and $\mathbf{W}^{(L)} \in \mathbb{R}^{d\times c}$ so that $\mathbf{h}_i^{(L)} \in \mathbb{R}^c$ is the prediction vector. The typical loss used for training the GNN is negative log-likelihood $\mathcal{L}_{\text{NLL}}(\mathbf{h}_i^{(L)}, y_i)$ for all the nodes in the training set $\mathcal{Y}_{\text{train}}$. The complete set of trainable parameters is denoted as $\psi = \{\psi_1\} \cup \{\mathbf{W}^{(l)}\}_{l=1}^{L}$.

B Interpretation of the distribution matching loss

We recap the objective function. For notation brevity, we use t_i to denote the input target node with its pruned candidates: $(t_{\text{target}}, t_{\text{candidates}})$:

$$\mathrm{KL}(\tilde{p}_{\mathrm{LM}}(\cdot|t_i, y_i)\|p_\phi(\cdot|t_i)) \tag{16}$$

where the stop gradient operator is removed if we only compute the gradient with respect to ϕ and

$$p_\phi(t_j|t_i) = \frac{e^{s_\phi(t_i,t_j)}}{\sum_{t_k\in\mathcal{D}} e^{s_\phi(t_i,t_k)}} \tag{17}$$

$$\tilde{p}_{\mathrm{LM}}(t_j|t_i, y_i) = \frac{e^{p_{\mathrm{LM}}(y_i|t_i,t_j)}}{\sum_{k\in\mathcal{N}_i} e^{p_{\mathrm{LM}}(y_i|t_i,t_k)}} \tag{18}$$

For notation brevity, we replace $\sum_{t_k\in\mathcal{D}}$ with $\sum_z$ if there is no ambiguity. Then

$$\min_\phi \ \mathrm{KL}\big(\tilde{p}_{\texttt{LM}}(\cdot|t_i, y_i)\|p_\phi(\cdot|t_i)\big) \tag{19}$$

$$\Leftrightarrow \min_\phi -\sum_z \tilde{p}_{\texttt{LM}}(z|t_i, y_i)\log[p_\phi(z|t_i)] \tag{20}$$

$$= -\sum_z \tilde{p}_{\texttt{LM}}(z|t_i, y_i)\log\left(\frac{e^{s_\phi(z,t_i)}}{\sum_{z'} e^{s_\phi(z',t_i)}}\right) \tag{21}$$

$$= \sum_z \tilde{p}_{\texttt{LM}}(z|t_i, y_i)\log\left(\sum_{z'} e^{s_\phi(z',t_i)}\right)$$
$$- \sum_z \tilde{p}_{\texttt{LM}}(z|t_i, y_i)s_\phi(z, t_i) \tag{22}$$

$$= \log\left(\sum_z e^{s_\phi(z,t_i)}\right)$$
$$- \sum_z \tilde{p}_{\texttt{LM}}(z|t_i, y_i)s_\phi(z, t_i) \tag{23}$$

Hence,

$$\nabla\mathrm{KL} = \frac{\sum_z e^{s_\phi(z,t_i)}\nabla s_\phi(z, t_i)}{\sum_{z'} e^{s_\phi(z',t_i)}})$$
$$- \sum_z \tilde{p}_{\texttt{LM}}(z|t_i, y_i)\nabla s_\phi(z, t_i) \tag{24}$$
$$= \sum_z (p_\phi(z|t_i) - \tilde{p}_{\texttt{LM}}(z|t_i, y_i))$$
$$\cdot \nabla s_\phi(z, t_i) \tag{25}$$
$$= \sum_z \left(1 - \frac{\tilde{p}_{\texttt{LM}}(z|t_i, y_i)}{p_\phi(z|t_i)}\right)$$
$$\cdot p_\phi(z|t_i)\nabla s_\phi(z, t_i) \tag{26}$$

After changing the notation back from $\sum_z$ to $\sum_{t_k\in\mathcal{D}}$, we have

$$\nabla\mathrm{KL} = \sum_{t_k\in\mathcal{D}} \left(1 - \frac{\tilde{p}_{\texttt{LM}}(t_j|t_i, y_i)}{p_\phi(t_j|t_i)}\right)$$
$$\cdot p_\phi(t_j|t_i)\nabla s_\phi(t_j, t_i) \tag{27}$$

whose rationale is that **if the LM's feedback greatly prefers the neighbor v_j (and its associated text t_j), larger than its probability to be retrieved by the retriever (i.e., $\frac{\tilde{p}_{\texttt{LM}}(t_j|t_i,y_i)}{p_\phi(t_j|t_i)} > 1$), then the similarity score between t_i and t_j will increase**, i.e., improve the probability of t_j to be retrieved.

C Templates

Table 4 presents templates used in this paper. We design the "Citation" template for the Cora, Pubmed, and ogbn-arxiv datasets and the "Amazon" template for the ogbn-products dataset.

Drawing inspiration from the findings of (He et al., 2024), who demonstrated the efficacy of positioning the title after the main content for certain datasets, we have also introduced two additional template variations: "Citation, Title Last" and "Amazon, Title Last."

D Algorithm

A step-by-step process for fine-tuning AUGLM, processing one training node per step, is presented in Algorithm 1. This procedure can be readily extended to mini-batch settings.

E Dataset Statistics

We present the detailed statistics of datasets used in this paper in Table 5.

Table 4: Templates used for all datasets.

Template Name	Prompt Text
Citation (Cora, Pubmed, ogbn-arxiv)	`Please classify the following paper into {pruned label candidates} based on the provided information\nTitle: {target node's title}\nContent: {target node's abstract}\nRelated papers: {retrieved nodes' titles}`
Citation, Title Last (Cora, Pubmed, ogbn-arxiv)	`Please classify the following paper into {pruned label candidates} based on the provided information\nContent: {target node's abstract}\nRelated papers: {retrieved nodes' titles}\nTitle: {target node's title}`
Amazon (ogbn-products)	`Please classify the following Amazon product into {pruned label candidates} based on the provided information\nProduct name: {target node's title}\nDescription: {target node's description}\nRelated products: {retrieved nodes' titles}`
Amazon, Title Last (ogbn-products)	`Please classify the following Amazon product into {pruned label candidates} based on the provided information\nDescription: {target node's description}\nRelated products: {retrieved nodes' titles}\nProduct name: {target node's title}`

All the baseline methods' performance on the Cora, Pubmed, and ogbn-arxiv is reported from the public leaderboards [2,3,4] and their published papers.

The ogbn-products dataset used in this paper is a subset of the original ogbn-products dataset (Hu et al., 2020) from TAPE (He et al., 2024). We follow the settings in TAPE and report baseline methods' performance from the TAPE (He et al., 2024) paper.

F Selected Hyperparameters

We report the hyperparameters used for every dataset in Table 6. As mentioned in the main content, we use two pretrained all-MiniLM-L6-v2 models as the dual encoder and the Flan-T5-small/base/large models as the backbone; they are all publicly available[5,6]. More detailed hyperparameters will be released with the code upon publication.

G Additional Experiments

G.1 Additional efficiency study

Memory usage. Memory usage is linear concerning batch size. We report the memory usage of AuGLM with different backbone LMs in Table 7, where we set the batch size to 1 and we found the experimental results reasonable because more powerful backbone LMs require more GPU memory.

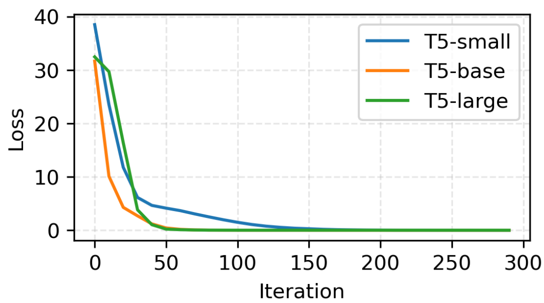


Figure 4: Convergence curve of AuGLM.

Convergence curve. We train AuGLM with different backbone LMs: FLAN-T5-small/base/large on the Cora dataset and plot their loss curves regarding updating steps in Figure 4. In this experiment, the batch size is 16. It shows that our proposed AuGLM converges smoothly and quickly when equipped with various LMs of different scales.

FLOPs. The floating point operations (FLOPs) of AuGLM are studied. Specifically, the computation of our AuGLM includes (1) precomputing PPR neighbors for every node, (2) training and inference of the semantic retriever ϕ, and (3) training and inference of the LM θ. Hence, the extra on-the-fly computation cost is from the semantic retriever ϕ (all-MiniLM-L6-v2 in our experiments). We report the FLOPs of the retriever and different LM backbones in Table 8. The results show that (1) the retriever only adds a tiny amount of FLOPs to the backbone LMs and (2) our proposed AuGLM is

[2] https://paperswithcode.com/sota/node-classification-on-cora-60-20-20-random

[3] https://paperswithcode.com/sota/node-classification-on-pubmed-60-20-20-random

[4] https://ogb.stanford.edu/docs/leader_nodeprop/

[5] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[6] https://huggingface.co/docs/transformers/en/model_doc/flan-t5

Table 5: Dataset statistics.

Name	# Nodes	# Edges	# Classes	Split Strategy	Evaluation Metric
Cora	2 708	10 556	7	Random 60/20/20%	Accuracy
Pubmed	19 717	88 648	3	Random 60/20/20%	Accuracy
ogbn-arxiv	169 343	1 166 243	40	Given split	Accuracy
ogbn-products	54 025	198 663	47	Given split	Accuracy

Table 6: Selected hyperparameters for AUGLM across different datasets.

Hyperparameter	Cora	Pubmed	ogbn-arxiv	ogbn-products
# PPR neighbors	5	2	5	5
# Semantic neighbors	5	2	5	5
Prompt template	Citation	Citation	Citation, Title Last	Amazon
# Candidate labels	3	2	3	3
LM learning rate	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
Retriever learning rate	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}
Weight decay	0	0	0	0

Table 7: GPU Memory usage (MB) with different LMs.

Model	Memory
AUGLM (T5-small)	3 098
AUGLM (T5-base)	6 572
AUGLM (T5-large)	20 308

Table 8: FLOPs comparison between different modules.

Module	FLOPs (10^9)
Retriever	2.3
T5-small	71.7
T5-base	257.2
T5-large	845.4

Table 9: Running time (ms) of different modules.

Module	Forward	Backprop
Retriever	14.7	6.1
T5-small	90.0	32.0
T5-base	104.4	66.6
T5-large	277.2	197.0

efficient.

Running time. The running time (both forward and backpropagation) of the semantic retriever and the backbone LMs on the Cora dataset is recorded. The batch size is 1. This experiment is tested on an NVIDIA A100-SXM4-80GB. Table 9 shows that the semantic retriever only adds very limited on-the-fly computation overhead compared to the LM, showing the efficiency of AUGLM.

G.2 Additional hyperparameter study

In this section, we study the model's performance with various hyperparameters.

Selection of the backbone GNN. Specifically, we study the performance of AUGLM equipped with different GNNs. we compared the performance of AUGLM equipped with GraphSAGE (used in the reported results) with the counterpart equipped with GCN (Kipf and Welling, 2017). The comparison is in Table 10.

We observed that the performance is nearly identical between GCN and GraphSAGE. This can be attributed to two factors: (1) the classification performances of GCN and GraphSAGE are similar, and (2) the GNN is used to generate prototypes and prune candidate labels, which **does not require a highly powerful GNN** for accurate classification.

Number of PPR retrieved nodes. Next, we examined the relationship between the model performance and the number of nodes retrieved. In this auxiliary experiment, we fixed the number of nodes retrieved by semantic retrieval at 5 and varied the number of nodes retrieved by PPR retrieval. The results are reported in Table 11

Interestingly, we found that the model's performance remains relatively stable when the number of PPR nodes is less than 15. However, the per-

Table 10: Performance (accuracy %) comparison of AUGLM equipped with different GNNs.

Model	Cora	Pubmed	ogbn-arxiv	ogbn-products
GraphSAGE	91.14	94.80	75.39	81.73
GCN	90.98	94.85	75.21	81.82

Algorithm 1 Training procedure for AUGLM

Input:
(1) A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$ and training labels $\mathcal{Y}_{\text{train}}$;
(2) initialized backbone LM θ;
(3) initialized semantic encoder ϕ;
(4) initialized GNN ψ.
Preprocessing:
(1) Pretrain GNN ψ on $(\mathcal{G}, \mathcal{Y}_{\text{train}})$.
(2) Generate prototypes and their text via Eqs. (6) and (7).
(3) Generate pruned label candidates for each node via Eq. (8).
while θ and ϕ not converged **do**
  Sample node $v_i \sim \mathcal{V}$ with text t_i.
  Retrieve relevant nodes' text $t_{\text{retri},i}$ via: topological retrieval (Eq. (2)) and/or semantic retrieval (Eq. (4)).
  Construct prompt with t_i, $t_{\text{retri},i}$, and $t_{\text{candidates},i}$ (from Preprocessing step (3)), based on the template (e.g., Figure 2b).
  Update θ based on Eq. (9).
  Compute $p_\phi(\cdot|t_i)$ via Eq. (10).
  Perform LM inference $|\mathcal{D}|$ times for: $\{p_{\text{LM}}(y_i|t_i, t)\}_{t \in \mathcal{D}}$ and $\tilde{p}_{\text{LM}}(\cdot|t_i, y_i)$.
  Update ϕ based on Eq. (12).
end while

Table 11: Accuracy (%) of AUGLM on ogbn-arxiv with different numbers of PPR-retrieved neighbors. The best result is **bolded**.

# Neighbors	Accuracy (%)
1	75.18
3	75.76
5	75.39
7	75.19
9	76.05
10	**76.45**
15	75.99
20	74.81
25	74.48

formance degrades when too many nodes are retrieved (more than 15). A possible explanation is that when the number of PPR nodes becomes too large, every target node's **retrieved nodes become similar** (e.g., some hub nodes are retrieved by most nodes), reducing the discriminativeness of each target node. This phenomenon is reminiscent of the "oversmoothing" problem (Li et al., 2018) in GNNs, where a GNN with too many layers and a large receptive field produces indistinguishable latent representations for all the nodes.

Other topological retrieval options. In this auxiliary experiment, we use the link predictor to retrieve relevant neighbors. Specifically, we trained a graph autoencoder (GAE) (Kipf and Welling, 2016), a basic graph neural network-based link predictor, on the given graph. Then, we retrieved the **top-**5 **most confident neighbors from the reconstructed graph** to replace those obtained through PPR retrieval. The results are presented in Table 12, where Flan-T5-small is used as the backbone LM. For better reference, we also provide a version where PPR retrieval is replaced with retrieving from 1-hop neighbors.

We observe that both 1-hop neighbor retrieval and GAE perform worse than their PPR counterparts. A possible reason is that both 1-hop neighbor retrieval and GAE are **local** retrieval methods, whereas PPR can effectively capture the **global** structure. Additionally, we note that GAE is trained using a reconstruction loss, which means it tends to assign high confidence to **existing edges**. In other words, the neighbors retrieved by GAE would be similar to those obtained through 1-hop neighbor retrieval, except for some low-degree nodes.

Other semantic retrieval options. This additional experiment uses different semantic retrievers to replace the prototype-based semantic retriever used in the proposed AUGLM. In detail, the prototype-based semantic retrieval module is replaced with a simple semantic retriever that **selects the most textually similar nodes** via inner product. Concretely, we use two pretrained mod-

Table 12: Accuracy (%) of AUGLM with different *topological* retrieval techniques across datasets. The best result for each dataset is **bolded**.

Retrieval Technique	Cora	Pubmed	ogbn-arxiv	ogbn-products
1-hop neighbors	90.59	94.33	73.97	79.53
GAE	90.83	94.42	74.01	79.85
PPR neighbors	**91.14**	**94.80**	**75.39**	**81.73**

Table 13: Accuracy (%) of AUGLM with different *semantic* retrieval techniques across datasets. The best result for each dataset is **bolded**.

Retrieval Technique	Cora	Pubmed	ogbn-arxiv	ogbn-products
Simple semantic retriever	90.68	94.37	74.46	81.21
SimTeG-tuned simple retriever	—	—	74.70	—
Prototype-based retriever (ours)	**91.14**	**94.80**	**75.39**	**81.73**

els, (1) the original all-MiniLM-L6-v2[7] and (2) a fine-tuned all-MiniLM-L6-v2 by SimTeG (Duan et al., 2023)[8]. The remaining modules, including topological retrieval and classifier guidance, were left intact, and FLAN-T5-small is used as the LM backbone. The results are reported in Table 13.

We observe that the proposed prototype-based retriever is better than both the original all-MiniLM-L6-v2-based retriever and the SimTeg-tuned simple retriever. This is because:

1. The training objective of the SimTeG-tuned retriever is to align the classification loss with a GNN model (Duan et al., 2023), similar to knowledge distillation (Hinton et al., 2015). In other words, **the SimTeG-tuned retriever is a mixture of topological and semantic retrieval**, as the GNN incorporates both topology and node features. This means that its role partially overlaps with that of the topological PPR retriever.

2. Our prototype-based retriever can retrieve textual features from **multiple nodes**, but the other two cannot achieve this.

G.3 Additional link prediction experiments

The main task of this paper is on the node classification task, but we conducted a *preliminary* experiment to adapt our proposed AUGLM to the link prediction task, further showcasing the generality of the proposed AUGLM. A systematic study to adapt AUGLM to link prediction tasks is interesting, and we leave it as future work.

Link prediction can be viewed as **a classification task for a pair of nodes**. For all modules, we made the following adaptations:

1. We retained the topological PPR retrieval for the input node pair.

2. We concatenated the text of the node pair as input for the semantic retriever. The prototypes used as the corpus of the semantic retriever were still generated by a pre-trained GNN, which is consistent with our approach for the node classification task.

3. For classifier guidance, we utilized a pre-trained graph autoencoder (GAE), whose output is the connection probability for every node pair. We transformed the connection probability into plain language based on the following rules: (1) less than 0.2: "improbable", (2) 0.2 to 0.4: "unlikely", (3) 0.4 to 0.6: "maybe", (4) 0.6 to 0.8: "likely", and (5) more than 0.8: "highly likely". The GAE's prediction (in plain language) was then incorporated into the following template.

4. The template we used is in the following format:

Prompt Template for Link Prediction

```
Please determine if the following two papers are related or not.

Paper 1's title: {Paper 1's title}
```

[7] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[8] https://huggingface.co/datasets/vermouthdky/SimTeG/tree/main/ogbn-arxiv/all-MiniLM-L6-v2/main/cached_embs

Table 14: Accuracy (%) on the preliminary link prediction task for the Cora dataset.

Method	Accuracy
GAE	89.29
AuGLM (T5-small)	93.59
AuGLM (T5-base)	**94.25**

```
Paper 1's abstract: {Paper 1's abstract}
Paper 1's related works: {Paper 1's PPR
neighbors' titles}

Paper 2's title: {Paper 2's title}
Paper 2's abstract: {Paper 2's abstract}
Paper 2's related works: {Paper 2's PPR
neighbors' titles}

Other related works: {Semantic retrieved
nodes' titles}

An expert link prediction model predicted
that the possibility of these two papers
being related is: {GAE's prediction}

Do you think these two papers are related
or not?
Please answer Yes or No.
```

We conducted preliminary experiments on the Cora dataset, following the settings from the benchmark[9]. In this setup, 5% and 10% of edges were removed for validation and testing, respectively. Also, an equal number of non-connected node pairs were used as negative samples. The accuracy results are reported in the following table.

Our key findings are as follows:

1. Our proposed AuGLM can indeed be effectively adapted to link prediction tasks.

2. By leveraging a classic link predictor (GAE), our AuGLM achieves a significant performance boost over the backbone predictor GAE, which aligns with our observations in node classification tasks.

H Extended Related Work

LMs for graphs. Recent studies have explored the ability of LMs to understand graph topology by investigating problems such as graph substructure recall (Wang et al., 2024), connectivity (Wang et al., 2023a; Perozzi et al., 2024), node/edge counting (Perozzi et al., 2024), and spatial-temporal graphs (Zhang et al., 2024b). These efforts demonstrate that, although LMs are not trained explicitly for graph tasks, they can exhibit non-trivial graph reasoning abilities under suitable prompts or encodings.

Building on these findings, more work has explored the use of LMs for tasks on text-attributed graphs (TAGs). For node classification, several methods directly prompt LMs with node and neighborhood textual information to perform inference (Ye et al., 2023; Zhao et al., 2023b; Li et al., 2024b; Qin et al., 2023; Zhou et al., 2025; Sun et al., 2023). These works vary in how they construct the input prompts, such as using structured templates (Sun et al., 2023) or similarity-based neighbor selection (Li et al., 2024b), and have shown promising performance.

Beyond node classification, LMs have also been adapted for link prediction on TAGs (Brannon et al., 2023; Tan et al., 2024; Liu et al., 2024a,b), where the textual features of node pairs and their neighborhoods are used to determine potential links. These approaches often incorporate task-specific prompt engineering or graph-aware context construction to maximize LM performance.

A notable direction is transfer and zero-shot learning, where foundation models are trained to generalize across multiple graphs or tasks. Several methods aim to build unified models with in-context learning or instruction tuning capabilities (Tang et al., 2024; Sun et al., 2025b; Pan et al., 2024; He et al., 2025; Xia and Huang, 2024; Li et al., 2024e). These include approaches that pretrain on diverse graph datasets using graph-text alignment objectives (Li et al., 2024e; Zhu et al., 2025a), instruction tuning on graph-related tasks (Sun et al., 2025b), or modular architectures for generalization (He et al., 2025). Notably, GraphCLIP (Zhu et al., 2025b) AnyGraph (Xia and Huang, 2024), and RiemannGFM (Sun et al., 2025a) illustrate how cross-domain graph pretraining with LMs can lead to strong zero-shot or few-shot transfer.

Several studies also investigate graph reasoning with LMs. Recent work explores the use of chain-of-thought prompting (Jin et al., 2024b) and tool-augmented querying (Zhang, 2023), which equip LMs with step-by-step planning capabilities to solve complex graph tasks.

Finally, hybrid models that combine LMs and GNNs have also been proposed. GIANT (Chien et al., 2022) and GLEM (Zhao et al., 2023a) en-

[9] https://paperswithcode.com/paper/variational-graph-auto-encoders

code textual features in a graph-aware manner to jointly leverage language and structure. TAPE (He et al., 2024) augments the node textual features using LMs before applying a GNN, demonstrating that LMs can enrich node representations for downstream prediction. Other hybrid methods explore LLM-to-GNN distillation (Pan et al., 2024) or GNN adapters (Huang et al., 2024), showing that language models can either enhance or transfer knowledge to traditional graph models.

In addition to the above, as this is a rapidly evolving topic, several comprehensive reviews have been proposed. Interested readers are referred to (Liu et al., 2025; Wang et al., 2025; Ren et al., 2024; Li et al., 2024d; Jin et al., 2024a)

Retrieval-augmented generation (RAG). Retrieval-augmented generation (RAG) enhances language models by granting access to external knowledge sources, typically by retrieving relevant documents from a large corpus and conditioning generation on them (Lewis et al., 2020; Karpukhin et al., 2020). This approach addresses limitations of parametric models in retaining up-to-date or factual knowledge (Hashimoto et al., 2018). Advances such as REALM (Guu et al., 2020) introduced end-to-end retriever-generator training. In contrast, RETRO (Borgeaud et al., 2022) scaled RAG to large corpora using a frozen retriever and cross-attention over retrieved chunks. Later, methods like REPLUG (Shi et al., 2023) allowed retriever optimization even with black-box LMs. Other extensions include Fusion-in-Decoder (Izacard and Grave, 2021), Atlas (Izacard et al., 2023), and multimodal RAG systems (Yasunaga et al., 2023; Zhao et al., 2023c), which retrieve both text and images. This topic has attracted much attention; interested readers are referred to (Gao et al., 2023; Zhao et al., 2024) for comprehensive surveys.

Recent studies have extended retrieval-augmented generation (RAG) to graph data. Several works focus on improving graph-based retrieval: GNN-RAG (Mavromatis and Karypis, 2024) leverages a GNN to retrieve reasoning paths from knowledge graphs, while GRAG (Hu et al., 2025) encodes k-hop ego graphs as dense embeddings to support structure-aware retrieval. HippoRAG (Jimenez Gutierrez et al., 2024) constructs a long-term memory as a graph of fact triples and applies PPR to retrieve a contextual subgraph for generation. Other approaches, such as ATLANTIC (Munikoti et al., 2023) and KnowledGPT (Wang et al., 2023b), augment retrieval with heterogeneous graph structures or programmatic access to knowledge bases, enabling more informed and faithful generation.

In addition to improving retrieval, some works apply GraphRAG to domain-specific tasks. For example, GrapeQA (Taunk et al., 2023) and HamQA (Dong et al., 2023) focus on enhancing commonsense and multi-hop QA via graph pruning, augmentation, or hyperbolic representation. In biomedical and scientific domains, (Delile et al., 2024) and DALK (Li et al., 2024a) retrieve relational subgraphs from tailored knowledge graphs to improve factual QA. GraphRAG has also been applied to generation tasks: (Edge et al., 2024) perform query-focused summarization by segmenting documents into graph-based communities. For a more comprehensive introduction, readers are referred to (Peng et al., 2024; Zhu et al., 2025c; Zhang et al., 2025).