Efficient Text-Attributed Graph Learning through Selective Annotation and Graph Alignment

Anonymous authors
Paper under double-blind review

Abstract

In the realm of Text-attributed Graphs (TAGs), traditional graph neural networks (GNNs) often fall short due to the complex textual information associated with each node. Recent methods have improved node representations by leveraging large language models (LLMs) to enhance node text features, but these approaches typically require extensive annotations or fine-tuning across all nodes, which is both time-consuming and costly. To overcome these challenges, we introduce GAGA, an efficient framework for TAG representation learning. GAGA reduces annotation time and cost by focusing on annotating only representative nodes and edges. It constructs an annotation graph that captures the topological relationships among these annotations. Furthermore, GAGA employs a two-level alignment module to effectively integrate the annotation graph with the TAG, aligning their underlying structures. Experiments show that GAGA achieves classification accuracy on par with or surpassing state-of-the-art methods while requiring only 1% of the data to be annotated, demonstrating its high efficiency.

1 Introduction

In real-world scenarios, many graphs can be effectively represented as text-attributed graphs (TAGs) (Yang et al., 2021), where nodes correspond to textual data and edges signify relationships such as citations or purchases. TAGs are widely used in domains like text classification, recommendation systems, social networks, and fake news detection (Yang et al., 2015; Juan et al., 2023). For instance, in the Ogbn-arxiv dataset, nodes represent academic papers, while edges denote citation links (Hu et al., 2020a).

Traditional graph neural networks (GNNs), such as Graph Convolutional Network (GCN) (Kipf & Welling, 2016a) and GraphSAGE (Hamilton et al., 2017), are commonly employed for tasks like node classification and link prediction on TAGs. However, these models often underutilize the rich textual information associated with nodes, which is crucial for context understanding, leading to suboptimal performance on large datasets like Ogbn-arxiv (Yan et al., 2023).

To address this, recent works leverage large language models (LLMs) to enhance textual features or refine graph structures (He et al., 2023; Sun et al., 2023). For example, TAPE (He et al., 2023) generates node annotations via LLM-predicted summaries and explanations, while methods like SIMTEG (Duan et al., 2023) fine-tune LLMs for node representations before training GNNs. Although effective, these approaches are often computationally expensive, requiring significant time and costs for annotation and fine-tuning. For instance, annotating the Ogbn-arxiv dataset with TAPE takes 9 hours and costs $128.

To overcome these challenges, we propose GAGA: Graph Alignment Guided Text-attributed Graph Learning. Instead of annotating all nodes, GAGA identifies and annotates only representative nodes (or edges), significantly reducing time and costs. These annotations are used to construct an annotation graph, which is aligned with the TAG through a novel two-level contrastive learning process. This framework efficiently integrates semantic information from LLMs and topological information from GNNs.

GAGA comprises three stages: (1) selecting and annotating representative nodes or edges using LLMs to construct an annotation graph, (2) performing two-level contrastive learning between the annotation graph

and the TAG, and (3) applying the learned representations to downstream tasks, where only the GNN is fine-tuned while the LLM remains frozen. Experimental results on six datasets demonstrate that GAGA achieves comparable or superior performance to state-of-the-art methods while reducing annotation requirements to just 1% of the data, improving efficiency by $3 \sim 100$ times. Our contributions can be summarized as follows:

- We propose GAGA, a lightweight framework for TAG representation learning. Compared to previous methods, GAGA considers an efficient strategy that only annotates representative nodes (edges), thereby reducing both the time and cost of annotation via LLMs. It recognizes and incorporates the topological relationships between these annotations by constructing an annotation graph.
- Unlike previous methods, which are based on fine-tuning the whole dataset, GAGA leverages the alignment instead. To achieve this, we align sub-annotation graphs with the subgraphs in TAG. Specifically, we developed a two-level contrastive learning (sub-graph level and prototype level), which provides significant improvement in performance. These two components could also be used in other problems.
- We provide comprehensive experiments to demonstrate the performance of GAGA. Specifically, via experiments over six datasets, GAGA achieves a classification accuracy that is comparable to or even exceeds SOTA methods for node classification and link prediction tasks. Moreover, compared to previous LLM-augmented SOTA methods, GAGA only requires 1% of the data to be annotated, making it surprisingly efficient.

2 Related Work

Language Models for Text-Attributed Graph. Traditional GNN approaches typically handle TAGs by converting textual attributes into features via shallow neural networks, such as bag-of-words, which limits the comprehensive understanding of textual semantics. Recent research has focused on embeddings based on language models (LMs) like BERT (Devlin, 2018) to address this issue. These fine-tuning pre-trained models can effectively generate deep embeddings to capture the rich semantic information within TAGs. There are two main architectures for empowering TAG tasks using LMs. The first is the cascaded architecture, where the textual information of nodes is independently encoded by LMs, and GNN models then aggregate the outputs to obtain the final embeddings, including TextGNN (Zhu et al., 2021a), GEAR (Zhou et al., 2019), GIANT (Chien et al., 2021), GPT-GNN (Hu et al., 2020b), and SimTeG (Duan et al., 2023). However, this approach separates text encoding from graph aggregation, which prevents a unified integration of the two processes. To address this limitation, the nested architecture has also been widely studied. This approach integrates text encoding and graph aggregation, performing these tasks iteratively to better unify both processes. For example, Graphormer (Yang et al., 2021), GLEM (Zhao et al., 2022) and DRAGON (Yasunaga et al., 2022) follow this nested architecture.

LLMs such as ChatGPT (Brown et al., 2020) have shown tremendous potential in various NLP tasks. However, how to apply LLMs to graph-structured data, such as TAGs, remains challenging (Chai et al., 2023; Qin et al., 2023). Chen et al. (2024) investigated the potential of LLMs in node classification tasks. Some works have already attempted to use LLMs for TAGs. He et al. (2023) proposed TAPE, which leverages LLMs to use the explanations generated during zero-shot classification as informative features for the graph. Another category of methods utilizes LLMs' strengths in text understanding to improve the graph's topology by enhancing the semantic understanding of node information. A representative example of this approach is the work by Sun et al. (2023), where they utilize LLMs to generate pseudo-labels and compute the semantic similarity between node attributes to refine the graph's topology through edge addition/removal and weight adjustment. Yu et al. (2023) propose ENG that enhances TAGs by using LLMs to generate additional nodes and the corresponding additional information. Additionally, Pan et al. (2024) employs a knowledge distillation approach to distill LLMs into graph models specifically for TAG learning. More recently, Fang et al. (2024) propose UniGLM, which trains a unified graph language model across multiple TAGs from diverse domains, fine-tuning PLMs with a domain-aware contrastive objective to improve cross-domain generalization. Xu et al. (2026) adapt off-the-shelf language models for node classification without architectural modification,

using topological and semantic retrieval augmentation together with a lightweight GNN classifier for candidate pruning. These methods further demonstrate the effectiveness of integrating LMs with graph structure, but they still rely on encoding the full set of nodes with relatively large language models, whereas GAGA only requires annotating a small fraction of nodes and aligns them with the graph through a lightweight encoder. However, focusing solely on either nodes or edges has its limitations. Concentrating only on nodes (textual information) can result in the loss of the network's topological structure, while focusing exclusively on edges makes it difficult to fully capture the semantic information of the nodes themselves. Therefore, we need a structure that combines both nodes and edges to enhance the TAG representation, which is a strength of GAGA. Moreover, as mentioned above, all previous SOTA methods are time-consuming and costly, while our GAGA is highly lightweight and achieves almost the same performance.

Explanation-Guided Learning. With the advancement of AI, the importance of research in Explainable Artificial Intelligence (XAI) has become increasingly prominent. Although researchers have made significant progress in XAI techniques, leading to more attempts to generate explanations for Deep Neural Networks (DNNs), more profound issues, such as how to apply XAI techniques to improve the performance of DNN models, warrant further attention as research progresses (Ross et al., 2017; Gao et al., 2024). In the field of computer vision, there has been extensive research on using explanation supervision to guide model training (Das et al., 2017; Linsley et al., 2018; Qiao et al., 2018; Mitsuhara et al., 2019; Zhang et al., 2019; Gao et al., 2022; Zhang et al., 2023; Saha & Roy, 2023). In NLP tasks, explanations are primarily generated through attention mechanisms or by utilizing auxiliary generative models (Bao et al., 2018; Strout et al., 2019; Zhong et al., 2019; Choi et al., 2020; Ghaeini et al., 2019; Jain et al., 2020; Stacey et al., 2022). Recently, several explanation supervision frameworks for GNNs have also emerged. For instance, GNES, proposed by Gao et al. (2021), optimizes model explanations and predictions through weak supervision and regularization of the model's explanations. Compared to these papers, in GAGA, we consider the annotation given by LLMs as an explanation, which can further guide us for contrastive learning.

3 Preliminaries

Text-Attributed Graph. A Text-attributed Graph (TAG) can be formally represented as $G = (V, A, \{s_n\}_{n \in V})$, where V is a set of N nodes, $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix, $s_n \in D^{L_n}$ is a sequential text associated with node $n \in V$, with D as the dictionary of words or tokens, and L_n as the sequence length.

Large Language Models and Prompting. Prompts can take various forms, such as a single sentence or longer paragraphs, and may include additional information or constraints to guide the model's behavior. Let $\mathcal{M}$ be an LLM that takes an input sequence $x = (x_1, x_2, \ldots, x_q)$ and produces an output sequence $y = (y_1, y_2, \ldots, y_m)$. The model is typically trained to optimize the conditional probability distribution $p(y|x)$, which assigns a probability to each possible output sequence y given x. To incorporate a prompt w with the input sequence x, we can concatenate them into a new sequence $\hat{x} = (w, x_1, x_2, \ldots, x_q)$. The conditional probability distribution $p(\hat{y}|\hat{x})$ is then computed using $\hat{x}$. Formally, the probability of the output sequence y given $\hat{x}$ is:

$$p(\hat{y}|\hat{x}) = \prod_{i=1}^{m} p(y_i|y_{<i}, \hat{x}),$$

where $y_{<i}$ represents the prefix of the sequence y up to position $i-1$, and $p(y_i|y_{<i}, \hat{x})$ denotes the probability of generating y_i given $y_{<i}$ and $\hat{x}$.

GNNs for Node Classification and Link Prediction. GNNs are utilized for both node classification and link prediction via leveraging the structural and feature information of graphs. In node classification, the objective is to assign labels to nodes based on their attributes and connections. This is achieved by updating each node's representation through a message-passing process defined by:

$$h_i^k = f^k\left(h_i^{k-1}, \mathrm{AGG}\left(\{h_j^{k-1} : j \in \mathcal{N}_i\}\right)\right) \in \mathbb{R}^d,$$

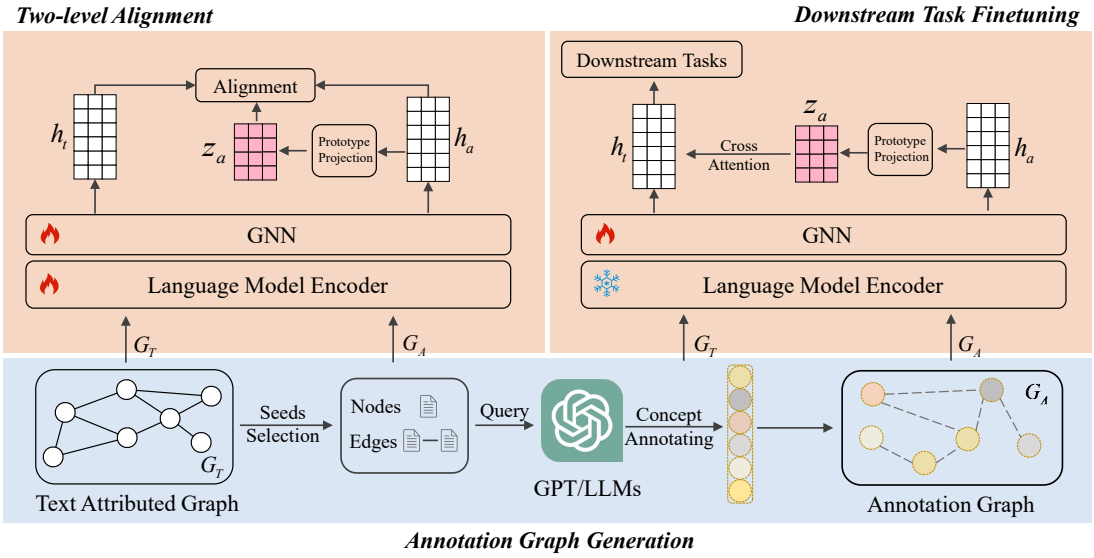


Figure 1: An overview of our GAGA framework.

where h_i^k is the representation of node i at layer k, $\mathcal{N}_i$ represents the set of neighbors, and f^k is a neural network layer that integrates the previous layer's representation with aggregated information from neighbors via a function like sum.

In link prediction, the task is to predict the probability of an edge existing between two nodes, i and j, based on their learned representations and the overall graph structure. This probability is modeled by $p(i, j) = f(h_i^k, h_j^k)$, where the function f uses the representations derived from the node classification process to assess the likelihood between node i and j.

4 GAGA: Graph Alignment Guided Text-attributed Graph Learning

In this section, we introduce our novel framework, GAGA. As illustrated in Figure 1, it primarily comprises three stages: annotation graph generation, two-level (subgraph) alignment, and downstream task fine-tuning. In the first stage, we first obtain a small but important set of nodes or edges through representative-based selection. Then, we generate annotations for these nodes or edges by prompting LLMs. These annotations are constructed into an annotation graph. In the second stage, we employ a two-level (subgraph level and prototype level) contrastive learning between the sub-annotation graphs and their corresponding sub-textual graphs, enabling lightweight fine-tuning of both the language model (LM) and GNNs using information from the selected nodes. In the third stage, which involves downstream tasks, we only fine-tune the GNN while keeping the LM frozen.

4.1 Annotation Graph Generation

The core idea of this stage is to use the information density metric to identify representative nodes and edges and then employ carefully designed prompts to utilize LLMs for annotation, thereby reducing costs.

Representative Node (Edge) Selection. As we mentioned earlier, using LLMs to annotate data can enhance model performance on TAGs. However, given that a typical graph contains tens of thousands of nodes, annotating each one can be both time-consuming and costly (Chen et al., 2023). Thus, to maintain high performance while annotating only a small number of nodes, it is crucial to select nodes (edges) with high information density (Cai et al., 2017). Information density, denoted as ϕ_{density}, serves as a representative metric to identify nodes (edges) that best "represent" the underlying data distribution in the embedding space. To locate nodes situated in dense regions of the embedding space, we first perform k-means clustering on the embeddings of all unlabeled nodes. We then compute each node's Euclidean distance to its nearest cluster center. The density score for a node v_i is determined by converting this distance into a similarity

score:

$$\phi_{\text{density}}(v_i) = \frac{1}{1 + \|\text{Emb}(v_i) - \text{Ce}(v_i)\|},$$

where $\text{Emb}(v_i)$ represents the embedding of node v_i, and $\text{Ce}(v_i)$ denotes the center of the cluster to which node v_i belongs. Intuitively, a higher value of $\phi_{\text{density}}(v_i)$ indicates that node v_i is closer to its center, making it more representative in the embedding space. We denote the selected subset of nodes as $V_a^* \subset V$, which consists of the nodes with top-s highest information density, where s is a hyperparameter.

For link prediction tasks, we need to select a set of edges. For edge $e_{i,j}$, we define its information density as the sum of the information densities of the nodes at both endpoints: $\phi_{\text{density}}(e_{i,j}) = \phi_{\text{density}}(v_i) + \phi_{\text{density}}(v_j)$. Similar to nodes, the selection of the subset of edges, V_a^*, is based on the top-$\hat{k}$ highest information density.

Annotation Generation. Next, we will annotate these selected nodes (edges). In GAGA, we prompt LLMs not only to produce predictions and explanations for the selected nodes but also to infer and understand the broader concepts and knowledge needed for these predictions and explanations. For edges, we prompt LLMs to explain the rationale behind their formation and to grasp the concepts and knowledge necessary to understand these connections. Our prompt templates can be found in Appendix D.

Annotation Graph Construction. To structurally represent the most informative knowledge contained in each node's text and leverage potential relationships between annotations, we construct an annotation graph $G_{\mathcal{A}}^* = (V_a^*, E_a^*)$. In detail, we first define each annotation for nodes or edges as a node $v_a \in V_a^*$, and then we build the annotation graph based on the similarity between each pair of nodes. For annotation node $v_a^i, v_a^j \in V_a^*$, the similarity $s(\cdot)$ is computed as follows:

$$s(v_a^i, v_a^j) = \frac{\text{Emb}(v_a^i) \cdot \text{Emb}(v_a^j)}{\|\text{Emb}(v_a^i)\| \cdot \|\text{Emb}(v_a^j)\|}.$$

We then apply the k-nearest neighbors (KNN) algorithm to capture the relationships between annotations, connecting each node to its k' nearest neighbors with edges to form the annotation graph. Formally, the edges of the annotation graph $G_{\mathcal{A}}^*$ is defined as:

$$E_a^* = \bigcup_i \{(v_a^i, v_a^j) \mid v_a^j \in N_{k'}(v_a^i)\},$$

where $N_{k'}(v_a^i)$ is set of k'-nearest neighbors of annotation node v_a^i.

4.2 Two-level Alignment

As we have only annotated a subset of nodes (edges), how to generalize these annotations becomes the key question, which is the motivation of our two-level alignment. Our alignment consists of two levels: one is for aligning sub-textual graphs and sub-annotation graphs, while the other one is for aligning sub-textual graphs and the prototypes induced by sub-annotation graphs.

Subgraph Alignment. To adapt the Language Model encoder for graph tasks, we perform graph alignment. Given that only a subset of nodes (edges) are annotated, we have to sample and align subgraphs based on this subset during this alignment process for better generalization. To achieve this, we use a straightforward approach: for each selected node v^*, we sample its k-hop neighbors in both the TAG and annotation graph $G_{\mathcal{A}}^*$ to form a sub-text graph G_T and sub-annotation graph G_A, respectively. In link prediction, for selected edges e^*, the difference lies in including the t-hop neighbors of both nodes connected by the selected edges to form the sub-annotation graph.

Through subsampling, we can obtain pairs of G_T and G_A corresponding to each selected node (or edge). Our alignment goal is to maximize the similarity between each pair of G_T and G_A, while simultaneously minimizing the similarity between non-corresponding pairs. Specifically, for each graph, nodes obtain their representations through a Language Model Encoder (LMEncoder), which are then integrated with graph structural information via a GNN. Mathematically, we have the following loss:

$$\mathcal{L}_1 = \sum_i (\|h_t^i - h_a^i\|^2 - \frac{1}{n-1} \sum_{j \neq i} \|h_t^i - h_a^j\|^2), \tag{1}$$

$$h_T = \text{GNN}(X_T), h_A = \text{GNN}(X_A),$$

$$X_T = \text{LMEncoder}(G_T), X_A = \text{LMEncoder}(G_A),$$

where $h_T = \{h_t^1, h_t^2, \ldots, h_t^n\}$ and $h_A = \{h_a^1, h_a^2, \ldots, h_a^n\}$ is the embedding matrix for G_T and G_A with number of nodes n in these subgraphs.

Prototype Alignment. As the number of selected nodes increases, the embedding matrix h_A of the annotation graph becomes increasingly large, resulting in higher computational complexity for downstream tasks. Additionally, the high-dimensional embeddings often contain significant redundancy, which can hinder the model's generalization ability. To tackle these issues, we introduce prototypes via vector quantization (VQ) (Van Den Oord et al., 2017) to map annotation graph embeddings into a more compact prototype space. This not only reduces computational overhead but also enhances semantic clustering.

Specifically, a prototype matrix Z_a is represented as a $k_p \times d$ matrix, where k_p is significantly smaller than the original number of nodes n. By fixing the number of prototypes, we effectively reduce the computational load of alignment. Specifically, each embedding vector h_a is mapped to its nearest prototype z_a in Z_a:

$$z_a = Z_a^m, \quad \text{where} \quad m = \arg\min_j \left(\|h_a - Z_a^j\|_2 \right), \tag{2}$$

where Z_a^m is the m-th row of Z_a. This process enforces semantic aggregation, reducing redundancy while encouraging the model to learn more abstract and generalizable representations of the annotation graph. To better illustrate the necessity of prototypes, see Appendix C for some visualization results.

Building on these prototype projections, we construct a contrastive loss to align textual embeddings h_t with prototype embeddings z_a. This loss is combined with the original contrastive loss into the final objective:

$$\begin{aligned} \mathcal{L} =& \alpha \sum_i \left[\|h_t^i - z_a^i\|^2 - \frac{1}{n-1} \sum_{j \neq i} \|h_t^i - z_a^j\|^2 \right] \\ &+ (1-\alpha)\mathcal{L}_1. \end{aligned} \tag{3}$$

Here the hyperparameter α governs the trade-off between textual and prototype alignment. A small α may cause the model to overfit to redundant information in the original embeddings, while a large α might lead to excessive reliance on prototypes, resulting in the loss of fine-grained semantic details. A balanced choice of α allows the model to effectively leverage the compactness and generalization benefits of prototypes while retaining critical semantic information from the original embeddings. By aligning representations in both the specific concept space and the aggregated prototype space, our method achieves an optimal balance between computational efficiency and representational granularity.

4.3 Downstream Task Finetuning

For a given downstream task, we first use the frozen Language Model encoder to obtain the representations of nodes $X_T^{'}$, which are then processed through GNN to acquire the node embeddings, i.e., $h = \text{GNN}(X_T^{'})$. Finally, by computing cross attention with the prototype embeddings of the annotations, we integrate the information of the annotations and obtain the final node representation.

$$h' = \text{softmax}(\frac{hW^Q \cdot (Z_a W^K)^T}{\sqrt{d_k}}) \cdot Z_a W^V,$$

where W^Q, W^K, W^V are query, key, Z_a is the matrix of prototypes in (2), and value weight matrices and d_k is the normalization factor.

For the node classification task, we add a fully connected layer and then pass the output through a softmax function to obtain the probability distribution for each category.

$$\hat{y} = \text{softmax}(f(h')),$$

where f is a linear function. For the link prediction task, we use the element-wise multiplication of the representations of nodes a and b as the corresponding edge representation. This feature is then passed through a fully connected layer followed by a sigmoid function to obtain the probability of the edge's existence:

$$\hat{y} = \text{sigmoid}((h'_a \odot h'_b) \cdot W).$$

5 Experiments

5.1 Experimental Setting

Datasets. We choose six datasets for evaluation: Cora (McCallum et al., 2000), PubMed (Sen et al., 2008), ogbn-arxiv (Hu et al., 2020a), ogbn-products (Hu et al., 2020a; He et al., 2023), and tape-arxiv23 (He et al., 2023) for the node classification tasks and use Cora, Citeseer (Giles et al., 1998b), and PubMed for the link prediction tasks. Details about the datasets can be found in Appendix A.1. The detailed prompts used for generating concepts for each dataset are shown in Appendix D. We split the datasets for training, validation, and testing following (He et al., 2023) for node classification tasks and HeaRT (Li et al., 2023; Mao et al., 2023) for link prediction tasks.

Baselines. In this study, we employed a variety of models for the node classification task to comprehensively evaluate the performance of different approaches on TAGs. Specifically, we selected the following 11 models: (i) MLP; (ii) GCN (Kipf & Welling, 2016a); (iii) GraphSAGE (Sun et al., 2021); (iv) RevGAT (Li et al., 2021); (v) InstructGLM (Ye et al., 2023); (vi) Graphormers (Ying et al., 2021); and (vii) TAPE (He et al., 2023) (viii) GLEM (Zhao et al., 2022) (ix) SimTEG (Duan et al., 2023) (x) ENGINE (Zhu et al., 2024) (xi) GIANT (Chien et al., 2021). Further details about the baselines are given in Appendix A.2.

Table 1: Comparison of Time (Minute) and Money Usage ($) Highlighted by Efficiency and Accuracy. OOM refers to instances where memory usage exceeded 32G of GPU memory or 64G of system memory. Timeout refers to tasks that remained incomplete after 72 hours. We highlight the best results in green.

	ogbn-arxiv			PubMed		
	Time Cost ↓	**Money Cost ↓**	**Accuracy ↑**	**Time Cost ↓**	**Money Cost ↓**	**Accuracy ↑**
TAPE	694	113.39	75.20	71	17.63	94.31
GLEM	446	7.81	76.04	50	0.87	92.57
SimTEG	1439	25.18	75.13	283	4.95	81.23
OneForAll*	185	3.24	0.6983	85	1.48	73.01
ENGINE	OOM	OOM	OOM	203	3.55	74.74
InstructGLM	OOM	OOM	OOM	OOM	OOM	OOM
Graphormer	224	3.92	66.67	45	0.783	83.65
GIANT	Timeout	Timeout	Timeout	431	7.54	76.89
GAGA(Ours)	80	2.87	76.23	17	0.49	94.61

For link prediction tasks, we selected 4 types of models, comprising a total of 19 models, as baselines: heuristic models, embedding models, GNN models, and GNN+Pairwise Info models. Heuristic models use graph structure-based scores to predict links. Embedding models learn low-dimensional representations of nodes to estimate link likelihoods, GNN models leverage multi-hop graph structures through message passing, and GNN+Pairwise Info models enhance GNNs with additional node-pair-specific information to improve link prediction accuracy. More details are given in Appendix A.2.

Evaluation Metrics. For node classification tasks, we use classification accuracy, time cost (min), and money cost of the whole framework as evaluation metrics. In the experiment of time and money comparison, time includes new textual feature annotation (like TAPE and GAGA), training and inference time. Money usage only includes the money spent on annotating new features with LLMs. For link prediction tasks, we choose MRR@10 and AUC as metrics. See details in Appendix A.4.

Experimental Setup. In our experiments, we prompt GPT-3.5-turbo-1106 (Brown et al., 2020) to generate annotations. The Language Model Encoder utilized is all-MiniLM-L6-v2 (Wang et al., 2020), and the GNN employed is a 4-layer GCN. During the annotation step, we consider 40 clusters. For the node classification task, we use 1% nodes for annotation and $\sqrt{n_{\text{edges}}}$ for link prediction annotation with edge number n_{edges}.

Table 2: Node classification performance comparison on different datasets (all values in %). The best results are highlighted with **bold**.

Model	ogbn-arxiv	PubMed	Cora	ogbn-products	tape-arxiv23
MLP	53.36 ± 0.15	86.35 ± 0.20	63.88 ± 0.12	53.85 ± 0.18	62.02 ± 0.25
GCN	71.82 ± 0.20	80.31 ± 0.15	88.24 ± 0.10	70.52 ± 0.12	63.41 ± 0.20
SAGE	71.71 ± 0.18	88.81 ± 0.15	89.11 ± 0.12	69.13 ± 0.20	64.30 ± 0.18
RevGAT	70.83 ± 0.15	88.50 ± 0.20	89.11 ± 0.10	69.64 ± 0.15	65.63 ± 0.20
Graphormer	72.81 ± 0.18	88.24 ± 0.15	80.41 ± 0.20	68.15 ± 0.12	62.87 ± 0.25
InstructGLM	75.70 ± 0.20	93.84 ± 0.15	87.08 ± 0.12	65.32 ± 0.02	70.32 ± 0.12
GLEM	75.60 ± 0.15	92.57 ± 0.20	74.69 ± 0.18	73.77 ± 0.15	78.58 ± 0.20
SimTEG	75.29 ± 0.18	81.20 ± 0.01	88.04 ± 0.12	74.51 ± 0.20	79.51 ± 0.15
ENGINE	76.02 ± 0.20	74.72 ± 0.15	**91.48 ± 0.10**	**80.05 ± 0.15**	79.76 ± 0.20
GIANT	74.26 ± 0.15	76.90 ± 0.20	85.52 ± 0.18	74.06 ± 0.12	72.18 ± 0.25
TAPE	75.20 ± 0.18	94.31 ± 0.15	91.19 ± 0.12	79.96 ± 0.20	80.80 ± 0.18
GCN*	62.17±1.03	79.83±0.29	61.19±4.11	37.90±0.59	51.52±0.18
TAPE*	69.02±0.16	91.78±0.12	83.76±2.27	35.53±0.21	51.99±0.62
GAGA(Ours)	**76.21 ± 0.15**	**94.62 ± 0.20**	89.67 ± 0.12	78.87 ± 0.18	**81.03 ± 0.25**

During the alignment process, we set α to 0.6, and prototype dimension k_p to 40. We also use 2-hop subgraphs for the alignment. We use the Adam optimizer with the learning rate $5e^{-5}$ for alignment and $1e^{-3}$ for downstream task fine-tuning. All experiments were conducted on a 32G V100 card with 10 CPU cores and a maximum memory of 64G (In Table 2, we use 4xA100 GPUs to reproduce the results for ENGINE and InstructGLM, as they cannot be executed on a V100.). We run each experiment for 5 times.

5.2 Utility Analysis

Time and Cost Comparison. Table 1 compares the time and cost (see Appendix A.3.2) efficiency of GAGA with previous LLM-based methods on the ogbn-arxiv and PubMed datasets. GAGA proves highly efficient, requiring annotations for only 1% of nodes, while other methods often annotate or fine-tune all nodes. TAPE incurs high costs due to API usage and extensive annotations. ENGINE, despite combining LLM and GNN, struggles with memory issues, as the Llama-3-7B model consumes substantial GPU resources, making it unsuitable for large datasets on a single GPU. GIANT uses self-supervised learning for node representations, but its pre-training is extremely time-intensive as node counts grow. GLEM and GraphFormers are relatively efficient among prior methods but are more costly than GAGA and fall short in accuracy compared to SOTA results and GAGA.

Node Classification. In the node classification task, as shown in Table 2, GAGA significantly outperforms previous classical GNN models, such as GCN, and transformer-based methods, such as Graphormer, across all datasets. Moreover, compared to LLM-based methods, our method still performs well; it achieves the best results on ogbn-arxiv, PubMed, and tape-arxiv23. While its performance on the Cora and ogbn-products datasets was second only to ENGINE and TAPE, the difference to the best, ENGINE, is less than 1.8%. These results demonstrate that GAGA achieves a classification accuracy that is comparable to or even exceeds SOTA methods due to the high generalization ability of our alignment method. Additionally, when comparing the performance of TAPE and GCN using only 1% of labeled data (GCN*, TAPE*), our two-level alignment method significantly outperforms them across all datasets.

Impact of the Prototype Size k_p. We used prototype embeddings to reduce the computational complexity during the downstream finetuning tasks. Here we will consider the effect of the prototype dimension. In Table 5, we can see while increasing k_p generally leads to higher computational costs, test accuracy remained relatively stable across different values. For instance, at $k_p = 10$, the model achieved a test accuracy of 76.42% with the lowest time (201.01 seconds) and memory usage (5.07 GB). As k_p increased to 1280, the accuracy slightly improved to 76.71%, but this came at the expense of significantly higher time (641.31 seconds) and memory (17.14 GB) requirements. Thus, the prototype projection is highly efficient since $k_p = 10$ already brings a good performance.

Table 3: Accuracy (%) with different backbones on ogbn-arxiv.

GNN	Test	Valid
MLP	72.83 ± 0.32	73.15 ± 0.12
GCN	76.21 ± 0.13	77.25 ± 0.09
SAGE	75.93± 0.02	76.11 ± 0.03
GAT	76.46 ± 0.22	77.01 ± 0.34
RevGAT	75.66 ± 0.17	75.87 ± 0.01

Table 4: Accuracy (%) with different ratio (%) of seed nodes on ogbn-arxiv.

Ratio	Test	Valid
0.1%	75.71 ± 0.02	77.01 ± 0.01
0.2%	75.72 ± 0.01	77.00 ± 0.03
0.4%	75.71 ± 0.01	76.99 ± 0.00
0.6%	75.71 ± 0.01	76.99 ± 0.01
0.8%	75.72 ± 0.02	77.00 ± 0.01
1.0%	76.21 ± 0.03	77.25 ± 0.02

Table 5: Experimental results for different prototype dimension k_p on ogbn-arxiv. Time and Memory are for the fine-tuning stage.

k_p	Memory (GB)	Time (s)	Test Accuracy(%)
10	5.07	201.01 ± 8.61	76.42 ± 0.07
40	5.24	212.43 ± 0.21	76.65 ± 0.01
80	5.47	223.03 ± 2.73	76.56 ± 0.09
160	5.92	242.37 ± 5.41	76.45 ± 0.05
320	7.44	291.87 ± 8.80	76.40 ± 0.03
640	10.67	413.12 ± 2.21	76.57 ± 0.10
1280	17.14	641.31 ± 6.17	76.71 ± 0.02
1693	21.31	782.28 ± 2.38	76.66 ± 0.06

5.3 Ablation Study

In this section, we conducted ablation experiments on the validation and test sets of the ogbn-arxiv dataset for node classification tasks to investigate the impact of different configurations on our model. Additional studies on the impact of text encoder, LLM, and hyperparameter α in (3) can be found in Appendix B.

Impact of the GNN Backbone on Performance. Since our framework is plug-and-play and does not alter the structure of the GNN model, it can be used with any GNN backbone. We explored the impact of different GNN backbones on classification accuracy. We tested the effects on several different backbones, including MLP, GCN, SAGE, GAT, and RevGAT in Table 3.

While using various GNNs as backbones did affect the final classification accuracy of the model, the impact was not significant. The only exception is MLP; its accuracy is lower, maybe because it is too simple to capture the relationship between the nodes. This demonstrates the robustness of our model.

Impact of Number of Selected Nodes. We study the effect of different proportions of selected nodes on performance. As shown in Table 4, while increasing the number of selected nodes can slightly improve classification accuracy, the model can already achieve strong performance (0.7571) even with just 0.1% annotated data. This efficiency is due to redundancy in the nodes' textual information. Specifically, in ogbn-arxiv, abstracts and titles of papers in the same category are highly similar, enabling effective learning from limited data. The use of high-information-density data allows the model to learn quality representations with fewer examples.

6 Limitations and Future Work

While GAGA achieves strong performance on standard text-attributed graph benchmarks, we acknowledge several limitations that also point to promising directions for future research.

Scope of evaluation. Our experiments follow the standard transductive setting on homophilic TAG benchmarks, which is consistent with prior work such as TAPE, GLEM, and SimTeG. Extending GAGA to high-heterophily graphs is a natural next step, although it would require rethinking the neighborhood-consistency assumption underlying both the graph encoder and the subgraph alignment objective. Similarly, although ogbn-arxiv already exhibits an extreme class imbalance ratio of over 942× and GAGA performs well on it, a dedicated study on long-tail benchmarks would further clarify the behavior of the alignment module under severe label skew.

Dependence on LLM annotation quality. GAGA's performance is tied to the quality of LLM-generated annotations. Our noise robustness analysis (Section B) shows that the model degrades gracefully under up to 20% annotation corruption, but domains with highly specialized or noisy text may still pose challenges. Developing annotation verification or self-correction mechanisms is an interesting direction. In privacy-constrained domains where proprietary LLM APIs are infeasible, our experiments with locally deployable open-source models (Llama 3.1 8B, Qwen 2.5 7B; see Appendix 9) indicate that GAGA remains effective, but a more systematic study across privacy-sensitive domains (e.g., medical or financial graphs) is left for future work.

Beyond text-attributed graphs. GAGA currently targets text-attributed graphs, where node attributes are natural language. Extending the framework to *heterogeneous* and *dynamic* TAGs, as well as to non-text-attributed graphs via multimodal LLMs (e.g., for image- or table-attributed nodes), would broaden the applicability of the approach and is a promising avenue for future research.

7 Conclusions

We proposed GAGA, a lightweight framework for node classification and link prediction in Text-attributed Graphs that minimizes annotation effort by focusing on 1% of representative nodes and edges. Through structural alignment and efficient annotation, GAGA achieves state-of-the-art accuracy with significantly reduced annotation cost.

References

Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

Yujia Bao, Shiyu Chang, Mo Yu, and Regina Barzilay. Deriving machine attention from human rationales. *arXiv preprint arXiv:1808.09367*, 2018.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. Active learning for graph embedding. *arXiv preprint arXiv:1705.05085*, 2017.

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

Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Hammerla, Michael M Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. *arXiv preprint arXiv:2209.15486*, 2022.

Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. Label-free node classification on graphs with large language models (llms). *arXiv preprint arXiv:2310.04668*, 2023.

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

Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S Dhillon. Node feature extraction by self-supervised multi-scale neighborhood prediction. *arXiv preprint arXiv:2111.00064*, 2021.

Seungtaek Choi, Haeju Park, Jinyoung Yeo, and Seung-won Hwang. Less is more: Attention supervision with counterfactuals for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6695–6704, 2020.

Abhishek Das, Harsh Agrawal, Larry Zitnick, Devi Parikh, and Dhruv Batra. Human attention in visual question answering: Do humans and deep networks look at the same regions? *Computer Vision and Image Understanding*, 163:90–100, 2017.

Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

Yi Fang, Dongzhe Fan, Sirui Ding, Ninghao Liu, and Qiaoyu Tan. Uniglm: Training one unified language model for text-attributed graph embedding, 2024. URL `https://arxiv.org/abs/2406.12052`.

Yuyang Gao, Tong Sun, Rishab Bhatt, Dazhou Yu, Sungsoo Hong, and Liang Zhao. Gnes: Learning to explain graph neural networks. In *2021 IEEE International Conference on Data Mining (ICDM)*, pp. 131–140. IEEE, 2021.

Yuyang Gao, Tong Steven Sun, Guangji Bai, Siyi Gu, Sungsoo Ray Hong, and Zhao Liang. Res: A robust framework for guiding visual explanation. In *proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 432–442, 2022.

Yuyang Gao, Siyi Gu, Junji Jiang, Sungsoo Ray Hong, Dazhou Yu, and Liang Zhao. Going beyond xai: A systematic survey for explanation-guided learning. *ACM Comput. Surv.*, 56(7), apr 2024. ISSN 0360-0300. doi: 10.1145/3644073. URL `https://doi.org/10.1145/3644073`.

Reza Ghaeini, Xiaoli Z Fern, Hamed Shahbazi, and Prasad Tadepalli. Saliency learning: Teaching the model where to pay attention. *arXiv preprint arXiv:1902.08649*, 2019.

C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. Citeseer: an automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries*, DL '98, pp. 89–98, New York, NY, USA, 1998a. Association for Computing Machinery. ISBN 0897919653. doi: 10.1145/276675.276685. URL `https://doi.org/10.1145/276675.276685`.

C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pp. 89–98, 1998b.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

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

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

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

Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1857–1867, 2020b.

Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C Wallace. Learning to faithfully rationalize by construction. *arXiv preprint arXiv:2005.00115*, 2020.

Ming-Hao Juan, Pu-Jen Cheng, Hui-Neng Hsu, and Pin-Hsin Hsiao. Attentive graph-based text-aware preference modeling for top-n recommendation. *arXiv preprint arXiv:2305.12976*, 2023.

Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

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

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.

Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. Training graph neural networks with 1000 layers. In *International conference on machine learning*, pp. 6437–6449. PMLR, 2021.

Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Evaluating graph neural networks for link prediction: Current pitfalls and new benchmarking. In *Neural Information Processing Systems NeurIPS, Datasets and Benchmarks Track*, 2023.

David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pp. 556–559, 2003.

Drew Linsley, Dan Shiebler, Sven Eberhardt, and Thomas Serre. Learning what and where to attend. *arXiv preprint arXiv:1805.08819*, 2018.

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

Haitao Mao, Juanhui Li, Harry Shomer, Bingheng Li, Wenqi Fan, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. Revisiting link prediction: A data perspective. *arXiv preprint arXiv:2310.00793*, 2023.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part II 22*, pp. 437–452. Springer, 2011.

Masahiro Mitsuhara, Hiroshi Fukui, Yusuke Sakashita, Takanori Ogata, Tsubasa Hirakawa, Takayoshi Yamashita, and Hironobu Fujiyoshi. Embedding human knowledge into deep neural network via attention map. *arXiv preprint arXiv:1905.03540*, 2019.

Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical review E*, 64(2): 025102, 2001.

Bo Pan, Zheng Zhang, Yifei Zhang, Yuntong Hu, and Liang Zhao. Distilling large language models for text-attributed graph learning. *arXiv preprint arXiv:2402.12022*, 2024.

Tingting Qiao, Jianfeng Dong, and Duanqing Xu. Exploring human-like attention supervision in visual question answering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32(1), 2018.

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

Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. *arXiv preprint arXiv:1703.03717*, 2017.

Gobinda Saha and Kaushik Roy. Saliency guided experience packing for replay in continual learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 5273–5283, 2023.

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

Joe Stacey, Yonatan Belinkov, and Marek Rei. Supervising model attention with human explanations for robust natural language inference. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36(10), pp. 11349–11357, 2022.

Julia Strout, Ye Zhang, and Raymond J Mooney. Do human rationales improve machine explanations? *arXiv preprint arXiv:1905.13714*, 2019.

Chuxiong Sun, Hongming Gu, and Jie Hu. Scalable and adaptive graph neural networks with self-label-enhanced training. *arXiv preprint arXiv:2104.09376*, 2021.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. *arXiv preprint arXiv:2203.00199*, 2022.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. *arXiv preprint arXiv:2302.00890*, 2023.

Zhe Xu, Kaveh Hassani, Si Zhang, Hanqing Zeng, Michihiro Yasunaga, Limei Wang, Dongqi Fu, Ning Yao, Bo Long, and Hanghang Tong. How to make lms strong node classifiers?, 2026. URL `https://arxiv.org/abs/2410.02296`.

Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, et al. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 36:17238–17264, 2023.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. Network representation learning with rich text information. In *IJCAI*, volume 2015, pp. 2111–2117, 2015.

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

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

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134*, 2023.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

Jianxiang Yu, Yuxiang Ren, Chenghua Gong, Jiaqi Tan, Xiang Li, and Xuecang Zhang. Empower text-attributed graphs learning with large language models (llms). *arXiv preprint arXiv:2310.09872*, 2023.

Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J Kim. Neo-gnns: Neighborhood overlap-aware graph neural networks for link prediction. *Advances in Neural Information Processing Systems*, 34:13683–13694, 2021.

Jiong Zhang, Wei-Cheng Chang, Hsiang-Fu Yu, and Inderjit Dhillon. Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. *Advances in Neural Information Processing Systems*, 34: 7267–7280, 2021a.

Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. *Advances in Neural Information Processing Systems*, 34: 9061–9073, 2021b.

Yifei Zhang, Siyi Gu, Yuyang Gao, Bo Pan, Xiaofeng Yang, and Liang Zhao. Magi: Multi-annotated explanation-guided learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1977–1987, 2023.

Yundong Zhang, Juan Carlos Niebles, and Alvaro Soto. Interpretable visual question answering by visual grounding from attention supervision mining. In *2019 ieee winter conference on applications of computer vision (wacv)*, pp. 349–357. IEEE, 2019.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. Learning on large-scale text-attributed graphs via variational inference. *arXiv preprint arXiv:2210.14709*, 2022.

Ruiqi Zhong, Steven Shao, and Kathleen McKeown. Fine-grained sentiment analysis with faithful attention. *arXiv preprint arXiv:1908.06870*, 2019.

Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Gear: Graph-based evidence aggregating and reasoning for fact verification. *arXiv preprint arXiv:1908.01843*, 2019.

Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71:623–630, 2009.

Jason Zhu, Yanling Cui, Yuming Liu, Hao Sun, Xue Li, Markus Pelger, Tianqi Yang, Liangjie Zhang, Ruofei Zhang, and Huasha Zhao. Textgnn: Improving text encoder via graph neural network in sponsored search. In *Proceedings of the Web Conference 2021*, pp. 2848–2857, 2021a.

Yun Zhu, Yaoke Wang, Haizhou Shi, and Siliang Tang. Efficient tuning and inference for large language models on textual graphs. *arXiv preprint arXiv:2401.15569*, 2024.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 34:29476–29490, 2021b.

A Additional Experimental Details

A.1 Datasets

Table 6: Statistics of TAGs used in this work.

Dataset	Nodes	Edges	Classes
Cora	2,708	5,429	7
PubMed	19,717	44,338	3
ogbn-arxiv	169,343	1,166,243	40
CiteSeer	3,312	4,732	6
ogbn-products	54,025	74,420	47
tape-arxiv23	46,198	78,543	40

The details of the datasets are as follows:

Cora. The Cora dataset (McCallum et al., 2000) consists of 2,708 scientific publications categorized into seven distinct classes: case-based, genetic algorithms, neural networks, probabilistic methods, reinforcement learning, rule learning, and theory. This dataset includes a citation network comprising 5,429 links. The selection process ensured that each paper in the final corpus either cites or is cited by at least one other paper.

PubMed. The PubMed dataset (Sen et al., 2008) comprises 19,717 scientific publications from the PubMed database, all related to diabetes. These publications are categorized into three distinct classes: Experimental Induced Diabetes, Type 1 Diabetes, and Type 2 Diabetes. The dataset also includes a citation network with 44,338 links.

ogbn-arxiv. The ogbn-arxiv dataset (Hu et al., 2020a) is a directed graph representing the citation network among all computer science papers on arXiv, as indexed by MAG (Wang et al., 2020). In this dataset, each node corresponds to an arXiv paper, and each directed edge signifies a citation from one paper to another. The primary task is to predict the 40 subject areas of the arXiv computer science papers, such as cs.AI, cs.LG, and cs.OS, which are manually assigned by the authors and arXiv moderators.

Citeseer. The CiteSeer dataset (Giles et al., 1998a) consists of 3312 scientific publications classified into one of six classes. The citation network consists of 4732 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 3703 unique words.

ogbn-products The ogbn-products Hu et al. (2020a) dataset is an Amazon product co-purchasing network with products as nodes and edges indicating co-purchases. The task is to predict the product category in a multi-class setup using 47 top-level categories as labels.

tape-arxiv23 The tape-arxiv23 He et al. (2023) dataset is a directed graph of citation networks among computer science arXiv papers from 2023 onwards. Each node represents a paper, and the edges show citation links. The task is to predict the 40 subject areas of these papers, such as cs.AI, cs.LG, and cs.OS, based on author and moderator labels.

A.2 Baselines

A.2.1 Node Classification

Details about the baseline models for node classification are as follows:

MLP. As a baseline model, MLP does not consider graph structure information and relies solely on node features for classification. Its performance provides a reference point for assessing the improvements offered by other models.

GCN. GCN (Kipf & Welling, 2016a) is a classical graph neural network model that effectively captures local neighborhood information through graph convolution operations. We chose GCN due to its strong performance and widespread application across many graph datasets.

GraphSAGE. GraphSAGE (Sun et al., 2021) generates node embeddings by sampling and aggregating information from neighboring nodes, making it suitable for handling large-scale graph data. We used GraphSAGE to evaluate its generalization capability across different datasets.

RevGAT. RevGAT (Li et al., 2021) combines graph attention mechanisms with reversible network structures, reducing memory consumption while maintaining performance. We selected RevGAT to explore the effectiveness of attention mechanisms on graph data.

InstructGLM. InstructGLM Ye et al. (2023) integrates the strengths of graph neural networks and language models, enhancing generalization through instruction learning. We here use InstructGLM using Llama-7b (Touvron et al., 2023) as the backbone.

Graphormer. Graphormer (Ying et al., 2021) leverages the Transformer architecture to process graph data, capturing global dependencies. We selected Graphormer to test the performance of Transformer-based approaches in graph neural networks.

TAPE. The TAPE model (He et al., 2023) leverages LLMs to capture textual information as features, which can be used to boost GNN performance on downstream tasks. A key innovation is its use of explanations as features: TAPE prompts an LLM to perform zero-shot classification, requests textual explanations for its decision-making process, and designs an LLM-to-LM interpreter to translate these explanations into informative features for downstream GNNs. For fairness, we use TAPE with the backbone of the GCN model, which is the same as our model.

GLEM. GLEM (Zhao et al., 2022) combines graph models with LLMs, like DeBERTa [12], in a variational EM framework, alternating between updating the LLM and GNN in the E-step and M-step to enhance downstream task performance.

OneForAll. OneForAll (Liu et al., 2023) represents all nodes and edges as human-readable texts and encodes them from various domains into a unified space using LLMs. The framework then adapts to different tasks by adding task-specific prompts into the input graph.

ENGINE. ENGINE (Zhu et al., 2024) introduces a lightweight, tunable G-Ladder module to each LLM layer, using a message-passing mechanism to incorporate structural information. This allows token-level outputs from each LLM layer to pass through the G-Ladder, enhancing node representations for node classification.

GIANT. GIANT (Chien et al., 2021) utilizes XR-Transformers (Zhang et al., 2021a) for neighborhood prediction, producing an LLM that generates more effective feature vectors for node classification than bag-of-words and vanilla BERT embeddings.

SimTEG. SimTeG (Duan et al., 2023) fine-tunes a pre-trained LM with parameter-efficient methods for a task like node classification, then uses the LM's last hidden states as node embeddings for GNN training, significantly boosting performance on graph benchmarks.

A.2.2 Link Prediction

For the link prediction tasks, we use the baseline models from HeaRT (Li et al., 2023; Mao et al., 2023).

Heuristic methods: Common Neighbor (CN) (Newman, 2001), Adamic Adar (AA) (Adamic & Adar, 2003), and Resource Allocation (RA) (Zhou et al., 2009) use common neighbors, while Shortest Path (Liben-Nowell & Kleinberg, 2003) and Katz (Katz, 1953) rely on path information to score link existence.

Embedding methods: Matrix Factorization (MF) (Menon & Elkan, 2011), Multilayer Perceptron (MLP), and Node2Vec (Grover & Leskovec, 2016) learn low-dimensional node embeddings to predict link likelihood.

GNN methods: Graph Convolutional Network (GCN) (Kipf & Welling, 2016a), Graph Attention Network (GAT) (Veličković et al., 2018), GraphSAGE (SAGE) (Sun et al., 2021), and Graph Autoencoder (GAE) (Kipf & Welling, 2016b) integrate multi-hop graph structures via message passing.

GNN + Pairwise Information: SEAL (Zhang et al., 2021b), BUDDY (Chamberlain et al., 2022), and Neural Bellman-Ford Network (NBFNet) (Zhu et al., 2021b) use subgraph features; Neo-GNN (Yun et al., 2021), Neighborhood Contrastive Network (NCN) (Wang et al., 2023), and NCNC (Wang et al., 2023) utilize common neighbor information, while Position Encoding Graph Neural Network (PEG) (Wang et al., 2022) employs positional encoding.

A.3 Experiment Setting Details

A.3.1 Baseline settings

For details on the settings of each baseline, we use their default hyper-parameters and settings, please refer to their respective GitHub repositories:

- **MLP, GCN, SAGE**: `https://github.com/snap-stanford/ogb/tree/master/examples/nodeproppred/arxiv`
- **TAPE, RevGAT**: `https://github.com/XiaoxinHe/TAPE/tree/main`
- **Graphormer**: `https://github.com/Microsoft/Graphormer`
- **InstructGLM**: `https://github.com/agiresearch/InstructGLM`
- **GLEM**: `https://github.com/AndyJZhao/GLEM`
- **SimTEG**: `https://github.com/vermouthdky/SimTeG`
- **ENGINE**: `https://github.com/ZhuYun97/ENGINE`
- **GIANT**: `https://github.com/amzn/pecos/tree/mainline/examples/giant-xrt`
- **HeaRT**: `https://github.com/Juanhui28/HeaRT`

A.3.2 Time and Cost Comparison Settings

In this Time and Cost Comparison experiment, `accuracy` refers to the node classification accuracy. `money` includes the computing costs, such as renting a V100 GPU instance on platforms like AWS (approximately $3.06 per hour at the time of our experiments, though this rate may have changed), as well as the costs for OpenAI API token consumption. The `time` includes the total duration of the process, encompassing the LLM annotation phase (if applicable), the subsequent alignment process (if applicable) and fine-tuning process.

A.4 Evaluation Metrics

We use MRR@10 and AUC for link prediction tasks.

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\text{rank}_i},$$

where N is the number of positive samples and rank_i is the rank of the i-th positive sample.

$$AUC = \frac{1}{|\mathcal{D}^0| \times |\mathcal{D}^1|}\sum_{i\in\mathcal{D}^0}\sum_{j\in\mathcal{D}^1}\mathbb{I}(\text{score}_i > \text{score}_j),$$

where $\mathcal{D}^0$ is the set of positive samples, $\mathcal{D}^1$ is the set of negative samples, and $\mathbb{I}$ is the indicator function.

Table 7: Link prediction results on Cora, Citeseer, and PubMed(%). The best results are highlighted with **bold**.

	Models	Cora		Citeseer		PubMed	
		MRR@10	AUC	MRR@10	AUC	MRR@10	AUC
Heuristic	CN	20.99	70.85	28.34	67.49	14.02	63.9
	AA	31.87	70.97	29.37	67.49	16.66	63.9
	RA	30.79	70.96	27.61	67.48	15.63	63.9
	Shortest Path	12.45	81.08	31.82	75.5	7.15	74.64
	Katz	27.4	81.17	38.16	75.37	21.44	74.86
Embedding	Node2Vec	37.29 ± 8.82	90.97 ± 0.64	44.33 ± 8.99	94.46 ± 0.59	34.61 ± 2.48	93.14 ± 0.18
	MF	14.29 ± 5.79	80.29 ± 2.26	24.80 ± 4.71	75.92 ± 3.25	19.29 ± 6.29	93.06 ± 0.43
	MLP	31.21 ± 7.90	95.32 ± 0.37	43.53 ± 7.26	94.45 ± 0.32	16.52 ± 4.14	98.34 ± 0.10
GNN	GCN	32.50 ± 6.87	95.01 ± 0.32	50.01 ± 6.04	95.89 ± 0.26	19.94 ± 4.24	89.69 ± 0.06
	GAT	31.86 ± 6.08	93.69 ± 0.27	48.69 ± 7.53	96.25 ± 0.20	18.63 ± 7.95	98.20 ± 0.07
	SAGE	37.83 ± 7.75	95.63 ± 0.27	47.84 ± 6.39	97.39 ± 0.15	22.74 ± 5.47	98.87 ± 0.04
	GAE	29.98 ± 3.21	95.08 ± 0.33	63.33 ± 3.14	97.06 ± 0.22	16.67 ± 0.19	97.47 ± 0.08
GNN + Pairwise Info	SEAL	26.69 ± 5.89	90.59 ± 0.75	39.36 ± 4.99	88.52 ± 1.40	38.06 ± 5.18	97.77 ± 0.40
	BUDDY	26.40 ± 4.40	95.06 ± 1.67	59.48 ± 8.96	96.72 ± 0.26	23.98 ± 5.11	98.2 ± 0.05
	Neo-GNN	22.65 ± 2.60	93.73 ± 0.36	53.97 ± 5.88	94.89 ± 0.60	31.45 ± 3.17	98.71 ± 0.05
	NCN	32.93 ± 3.83	96.76 ± 0.18	54.97 ± 6.03	97.04 ± 0.26	35.65 ± 4.60	98.98 ± 0.04
	NCNC	29.01 ± 3.83	**96.90 ± 0.28**	64.03 ± 3.67	97.65 ± 0.30	25.70 ± 4.48	99.14 ± 0.03
	NBFNet	37.69 ± 3.97	92.85 ± 0.17	38.17 ± 3.06	91.06 ± 0.15	**44.73 ± 2.12**	98.34 ± 0.02
	PEG	22.76 ± 1.84	94.46 ± 0.34	56.12 ± 6.62	96.15 ± 0.41	21.05 ± 2.85	96.97 ± 0.39
	GAGA(Ours)	**46.22 ± 2.13**	96.78 ± 0.02	**64.83 ± 3.11**	**98.13 ± 0.11**	44.31 ± 2.12	**99.24 ± 0.01**

B Additional Experiments

Results for Link Prediction. Similar to Node Classification, we conducted experimental evaluations on the link prediction task, with the results presented in Table 7 in the Appendix. Leveraging the annotations on selected edges and the two-level alignment, GAGA achieves near state-of-the-art performance across all evaluated datasets for both metrics, falling slightly short of NBFNet on the PubMed dataset for MRR@10.

Impact of Text Encoder. In this section, we aim to investigate the impact of different language models on the performance of GAGA. We replace the language model with various alternatives and compare their performance on the ogbn-arxiv dataset. In GAGA, we align annotations and graphs, both of which are represented by features extracted from a language model. Therefore, the selection of an appropriate language model significantly impacts the final performance of the model.

We also compared the effects of different language models on the overall classification accuracy. The results in Table 8 showed that using larger models may not improve classification accuracy. This may be due to the high dimensionality of embeddings in larger models, which may hinder the learning process of the GNN. Furthermore, due to the limited data utilized during the alignment process, it is possible that larger language models were not adequately aligned, contributing to the suboptimal performance observed.

Table 8: Ablation study on the effects of different LLMs for annotation.

Backbone	all-MiniLM-L6-v2	MiniLM-L12-H384-uncased	gte-Qwen2-1.5B-instruct	gte-Qwen2-7B-instruct	LLM2Vec-Meta-Llama-3-supervised
Model Size	23M	33M	1.5B	7B	8B
Test	0.7620	0.7681	0.7253	0.7515	0.7519
Valid	0.7701	0.7722	0.7404	0.7531	0.7556

Impact of α. In Figure 2, we studied the effect of the hyperparameter α in equation (3), which adjusts the model's focus between the alignment of sub annotation graph and sub textual graph, and the alignment of sub textual graph and prototype projections. As shown in the figure, when α is set to 0, the model disregards the prototypes and focuses on the original contrastive loss, making generalization ability worse as the model will learn too much redundant information. When α is set to 1, it focuses more on prototype alignment and ignores the original sub annotation graphs, making the utility worse as the project may lose too much

information. Both extreme cases result in a decline in model performance. However, when α is between 0 and 1, the model's performance remains relatively stable.

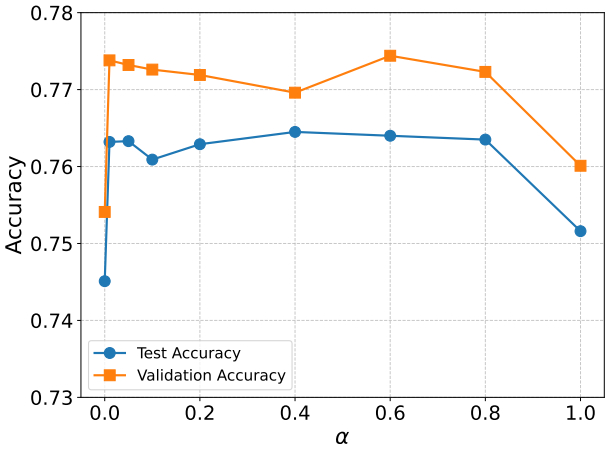


Figure 2: Impact of α on Valid and Test Accuracy on ogbn-arxiv Dataset.

Impact of LLMs. We conducted experiments with meta-llama/Meta-Llama-3.1-8B-Instruct (Llama 3.1 8B) and Qwen/Qwen2.5-7B-Instruct (Qwen 2.5 7B). The validation and test accuracies for each model on ogbn-arxiv dataset are summarized in Table **??**.

Table 9: Validation and Test Accuracies of Different LLMs (in %)

LLM	**Validation**	**Test**
Llama 3.1 8B	77.02 ± 0.09	76.06 ± 0.11
GPT3.5-turbo	77.09 ± 0.05	76.01 ± 0.04
Qwen 2.5 7B	76.98 ± 0.14	76.06 ± 0.09

The experimental results indicate that our method demonstrates strong robustness in selecting large language models (LLMs). The validation and test accuracies across different models (Llama 3.1 8B, OpenAI, and Qwen 2.5 7B) are very similar, ranging from approximately 0.7698 to 0.7709. This consistency suggests that our approach performs reliably regardless of the specific model chosen.

Moreover, the low standard deviations in the accuracy scores indicate that the models exhibit minimal variability in performance across multiple runs. This stability is a key indicator of robustness, showing that our method can maintain high performance even when faced with different input data or model selections.

Overall, the results provide confidence that our method is effective and adaptable, ensuring reliable outcomes across various LLMs.

Impact of Different Feature Combination Methods. During our downstream task fine-tuning, the rationale behind applying an expensive attention operator to combine features remained unclear. A relevant question was raised: what if we simply concatenate prototype features and annotate graph features instead?

To address this, we conducted additional experiments comparing different feature combination approaches—cross-attention, concatenation, and averaging—on the arxiv-ogbn dataset. Our results from five runs demonstrate that cross-attention consistently outperforms other methods:

These results indicate that cross-attention is the most effective feature combination approach in this setting.

Robustness to Annotation Noise To evaluate GAGA's robustness to noisy or biased LLM-generated annotations, we conduct a controlled noise-injection study on ogbn-arxiv. Specifically, we apply *shuffle noise*: we randomly swap annotations between a fraction of the selected nodes, simulating the scenario

Method	Evaluation Accuracy (%)	Test Accuracy (%)
Cross-Attention	77.82 ± 0.04	76.27 ± 0.12
Concatenation	72.91 ± 0.11	71.08 ± 0.06
Average	72.01 ± 0.11	71.98 ± 0.06
Baseline (GCN)	71.93 ± 0.05	71.82 ± 0.20

Table 10: Comparison of feature combination approaches on the arxiv-ogbn dataset.

where the LLM assigns incorrect or irrelevant explanations to a subset of nodes. We vary the shuffle rate from 0% (clean) to 20% and report both the alignment loss and the downstream test accuracy in Table 11.

Table 11: Robustness of GAGA to annotation noise on ogbn-arxiv. Shuffle noise randomly swaps annotations between selected nodes.

Shuffle Rate	**Alignment Loss**	**Test Accuracy (%)**
0% (clean)	1.272	76.21
10%	1.354	68.41
20%	1.569	61.65

The results show that GAGA degrades gracefully as the noise rate increases, and the alignment loss rises consistently (1.272 → 1.354 → 1.569), confirming that the model is genuinely learning from the annotation signal rather than overfitting to spurious patterns. We emphasize that shuffle noise is a particularly adversarial form of corruption: it does not merely remove information, but actively introduces misleading alignment targets by pairing nodes with semantically incorrect annotations. In practice, LLM-generated annotations are expected to be substantially more reliable than these synthetic noise settings, since the annotations are produced by prompting GPT-3.5 with task-specific instructions, making the effective noise rate much lower than 10%. The gradual and predictable degradation pattern further suggests that GAGA's performance can be improved simply by using a stronger LLM to produce higher-quality annotations.

B.1 Comparison with Recent LLM-Enhanced TAG Methods

To further situate GAGA among the most recent LLM-enhanced TAG methods, we compare it against UniGLM (Fang et al., 2024) and AuGLM (Xu et al., 2026) based on their originally reported numbers. UniGLM does not report results on PubMed or Cora, and its code is not publicly available, so we include only the numbers reported in the original paper. The comparison is summarized in Table 12.

Table 12: Comparison with recent LLM-enhanced TAG methods. Results for UniGLM and AuGLM are taken from their original papers. "–" denotes results not reported.

Method	**ogbn-arxiv**	**PubMed**	**ogbn-products**	**Cora**
AuGLM	76.00	95.16	82.90	91.51
UniGLM	74.34	–	77.45	–
GAGA (ours)	**76.21**	94.62	78.87	89.67

UniGLM prioritizes cross-domain generality, which comes at the cost of per-dataset performance. GAGA, by contrast, leverages target-domain graph–text alignment and achieves stronger results on individual benchmarks. More notably, AuGLM uses Flan-T5-Large (770M parameters) as its text encoder, while GAGA uses only all-MiniLM-L6-v2 (22.7M parameters) – a roughly 34× difference in model size. Despite this large gap in encoder capacity, GAGA outperforms AuGLM on ogbn-arxiv (76.21 vs. 76.00) and remains competitive on the other datasets, demonstrating that GAGA's two-stage alignment and LLM-based knowledge injection can effectively compensate for the use of a much smaller text encoder.

C Visualization of Annotation Prototypes

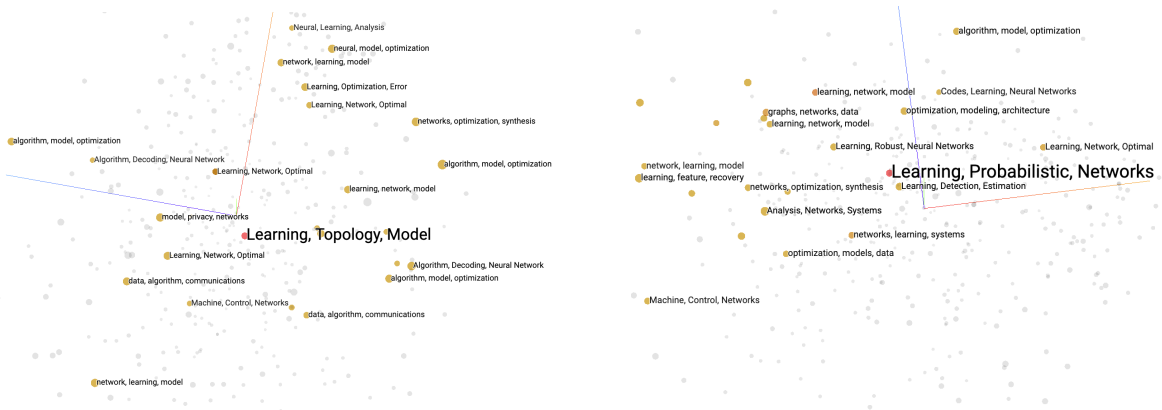
In our study, we propose the use of annotation prototypes based on the observation that there is semantic clustering among concepts. When aligning node representations, the alignment can be performed in the representation space of specific concepts and the space obtained after semantic aggregation through prototypes.

To evaluate the effectiveness of semantic aggregation, we employed GPT sampling for assessment. Specifically, on the OGBN-Arxiv dataset, we randomly selected 1,000 nodes and calculated the distance between these nodes and the prototypes. For each prototype, we identified the 10 nodes closest to it, considering these nodes as a cluster. We then used GPT-4 to label these clusters. The prompt used for this purpose was:

Prompt

node_texts
List three key words shared by the papers above.

As shown in figure 3, after obtaining the semantic labels, we applied Principal Component Analysis (PCA) for dimensionality reduction to a 3D space. The results demonstrated that nodes with similar semantic labels (prototypes) are also positioned close to each other in the semantic space, confirming the effectiveness of our semantic aggregation approach.



(a) (b)

Figure 3: Visualization of annotation prototypes on OGBN-Arxiv. Nodes with similar semantic labels are positioned close to each other in the semantic space.

We also present several case studies, as shown in Figure 4, to illustrate how the annotation prototype enhances the model's performance. When the input textual information is processed by GAGA, directly predicting the category of the node can sometimes result in inaccurate predictions. However, by incorporating prototype information and integrating it into the model's prediction through a cross-attention mechanism, the model can effectively adjust and refine its predictions. This is because the integration allows the model to leverage additional contextual knowledge from the prototypes, thereby enhancing its ability to correct initial errors and produce more accurate final predictions.

D Prompts for Generating Annotations

In this section, we present our prompts used to generate annotations.

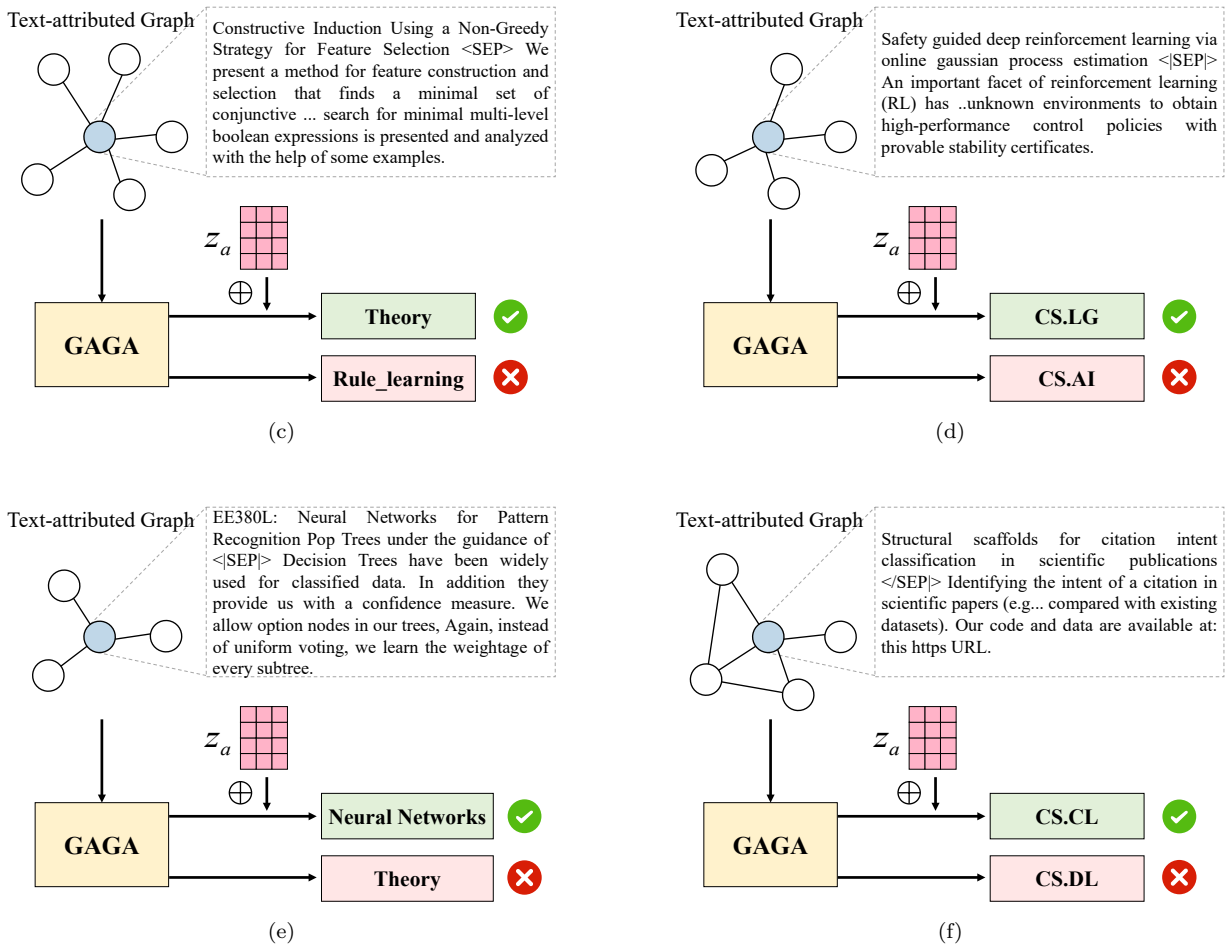


Figure 4: Visualization of the effect of annotation prototype projection.

Prompt Template

{Node's textual information} [Question]: Which of the following category does this node belong to: {possible categories}.
Give 5 likely categories as a comma-separated list ordered from most to least likely.
List the most important concepts in the paper. And you should tell me what knowledge is needed to understand the concepts. After all, you should provide your reasoning.
Your response:

Prompt for node classification on ogbn-arxiv.

Abstract: {abstract} Title: {title} Question: Which arXiv CS subcategory does this paper belong to? Give 5 likely arXiv CS sub-categories as a comma-separated list ordered from most to least likely, in the form "cs.XX", list the most important concepts in the paper. And you should tell me what knowledge is needed to understand the concepts. After all, you should provide your reasoning. Your response:

Prompt for node classification on Arxiv-2023.

Abstract: {abstract text} Title: {title text} Question: Which arXiv CS subcategory does this paper belong to? Give 5 likely arXiv CS sub-categories from most to least likely, in the form "cs.XX", List the most important concepts in the paper. And you should tell me what knowledge is needed to understand the concepts. After all, you should provide your reasoning. Your response:

Prompt for node classification on Cora.

Abstract: {abstract text} Title: {title text} Question: Which of the following sub-categories of AI does this paper belong to: Case Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning, Theory? Explain how your prediction is present in the text. List the most important concepts in the paper,and tell me what knowledge is needed to understand the concepts. After all, you should provide your reasoning.Your response:

Prompt for node classification on PubMed.

Abstract: {abstract text}
Title: {title text}
Question: Does the paper involve any cases of Type 1 diabetes, Type 2 diabetes, or Experimentally induced diabetes? Please give one or more answers of either Type 1 diabetes, Type 2 diabetes, or Experimentally induced diabetes; if multiple options apply, provide a comma-separated list ordered from most to least related, then for each choice you gave, give a detailed explanation with quotes from the text explaining why it is related to the chosen option. List the most important concepts in the paper.
And you should tell me what knowledge is needed to understand the concepts.
After all, you should provide your reasoning. Your response:

Prompt for link prediction.

Paper 1: Title: {title1} Abstract: {abstract1} Paper 2: Title: {title2} Abstract: {abstract2}
Question: Why are these two papers related? list the most important concepts in the abstract. And you should tell me what knowledge is needed to understand the concepts. Based on the concepts, explain why they are related.
Your response:

Prompt for node classification on ogbn-products.

Product description: {product description}
Question: Which of the following category does this product belong to: 1) Home & Kitchen, 2) Health & Personal Care, 3) Beauty, 4) Sports & Outdoors, 5) Books, 6) Patio, Lawn & Garden, 7) Toys & Games, 8) CDs & Vinyl, 9) Cell Phones & Accessories, 10) Grocery & Gourmet Food, 11) Arts, Crafts & Sewing, 12) Clothing, Shoes & Jewelry, 13) Electronics, 14) Movies & TV, 15) Software, 16) Video Games, 17) Automotive, 18) Pet Supplies, 19) Office Products, 20) Industrial & Scientific, 21) Musical Instruments, 22) Tools & Home Improvement, 23) Magazine Subscriptions, 24) Baby Products, 25) NAN, 26) Appliances, 27) Kitchen & Dining, 28) Collectibles & Fine Art, 29) All Beauty, 30) Luxury Beauty, 31) Amazon Fashion, 32) Computers, 33) All Electronics, 34) Purchase Circles, 35) MP3 Players & Accessories, 36) Gift Cards, 37) Office & School Supplies, 38) Home Improvement, 39) Camera & Photo, 40) GPS & Navigation, 41) Digital Music, 42) Car Electronics, 43) Baby, 44) Kindle Store, 45) Kindle Apps, 46) Furniture & Decor? Give 5 likely categories as a comma-separated list ordered from most to least likely, list the most important concepts.
And you should tell me what knowledge is needed to understand the concepts.
After all, you should provide your reasoning. Your response: