GraphTW: Graph-Guided Token Weight for Node Classification on Text-Attributed Graph

Anonymous ACL submission

Abstract

Text-Attributed Graphs (TAGs) are common in social, citation, and e-commerce networks, where nodes have rich text and complex structures. The key challenge is the lack of fine-grained interaction between graph structure and node semantics. Existing methods lack fine-grained interaction between graph structure and node semantics, often decoupling text encoding from graph modeling or simply concatenating neighboring texts, resulting in long sequences and high computational costs. In this work, we propose GraphTW, a Graph-guided Token Weight framework for TAGs. GraphTW estimates the relative importance of text at the token level by leveraging graph signals, enabling graph-guided text representations. To control sequence length while preserving key information, GraphTW retains all tokens of the target node and employs a Gumbel selection mechanism to select informative tokens from neighboring nodes, sampled via a weighted random walk that balances structural and semantic similarity. The selected tokens are then concatenated with the target node's tokens and fed into a language model to obtain enhanced node representations for classification. Experiments on standard TAG datasets show consistent improvements; for example, on ArXiv_2023, GraphTW achieves 89.12% accuracy, outperforming the strongest baseline by 4.33 percentage points.

1 Introduction

Text-Attributed Graphs (TAGs) are a class of structured data where nodes are interconnected via edges and augmented with rich textual attributes. (Yang et al., 2021; Pan et al., 2024; Zhao et al., 2024). In these graphs, the topology captures structural dependencies among entities, while the textual attributes convey rich semantic context for individual nodes. This data modality is pervasive in real-world applications; for instance, citation networks utilize edges to represent scholarly links alongside paper abstracts, and e-commerce platforms leverage co-purchase relationships complemented by product descriptions and reviews (Yang et al., 2016; Yan et al., 2023).

The core challenge in TAG learning lies in effectively modeling the interactive relationship between graph information and node textual semantics, thereby fully leveraging their complementary strengths. However, most existing TAG learning methods adopt a modeling paradigm where graph and textual attributes are modeled separately. Specifically, these methods typically rely on Graph Neural Networks (GNNs) (Kipf, 2016; Hamilton et al., 2017) to capture topological dependencies between nodes. In contrast, node textual attributes are treated as additional features, often represented separately via bag-of-words models (Zhao et al., 2022) or pre-trained language models (Duan et al., 2023), and then fused with graph embeddings at the node level. Under this paradigm, the interaction between graph and textual semantics often remains coarse-grained, struggling to capture finer-grained associations between them. This limitation constrains the ability of graph information to guide textual semantic modeling.

Based on the aforementioned challenges, a straightforward strategy for TAG representation learning involves flattening all nodes into a long sequence. The corresponding node texts are subsequently processed by a pre-trained language model for encoding, after which the resulting sequences are concatenated with the graph representation or jointly modeled (Zhao et al., 2023). Although this approach effectively leverages contextual semantic information from text, the input sequence length often expands rapidly when dealing with large graphs or lengthy node texts. This expansion often surpasses the maximum context window of language models, leading to significant computational complexity and memory overhead. Furthermore, the direct truncation required to meet model input con-

straints inevitably results in the loss of critical information, thereby diminishing the model's effectiveness and practicality in large-scale TAG scenarios.

In this work, we propose **Graph**-guided **T**oken **W**eight framework for TAGs, named **GraphTW**, which leverages node graph information to guide the modeling of token weights. It achieves deep collaborative modeling of graph and node textual semantics. Specifically, GraphTW uses node graph embeddings as queries, interacting with corresponding token representations through cross attention mechanisms to assign graph-guided weights to different tokens in the text. Furthermore, to fully exploit both local and higher-order structural contexts, this paper proposes a random walk strategy based on joint graphical and textual perception. During sampling, the strategy integrates graph-based node similarity with textual semantic similarity to generate a more discriminative set of contextual nodes. To address the resulting long-text modeling challenge, GraphTW leverages both pre-computed token weights and a Gumbel selection mechanism to pick tokens from contextual text. This effectively controls sequence length while maximizing retention of critical information, enabling efficient and scalable text-property graph representation learning.

We highlight our contributions as follows:

- We introduce GraphTW, a novel framework that explicitly captures the interplay between graph topology and textual semantics in TAGs. It employs graph-guided token scores and a straight-through Gumbel top-k selection mechanism to influence token-level semantic modeling, overcoming the limitations of existing methods that separate graph and text processing.

- GraphTW samples other nodes via a weighted random walk and selects their informative tokens using the straight-through Gumbel top-k mechanism. Concatenating these tokens to the target node's sequence yields an enriched representation that integrates local and higher-order context.

- Extensive experiments on several benchmark TAG datasets demonstrate that the proposed method consistently outperforms representative baseline models. This indicates that the method can leverage graph information to model fine-grained token-level weight for node texts, thereby achieving deep synergy between graph and text semantics and enhancing classification accuracy.

2 Preliminary

In this section, we will briefly introduce the concepts and background knowledge relevant to our proposed framework and provide formal definitions for concepts related to text-attributed graphs, large language models, and graph neural networks.

2.1 Text-Attributed Graphs

TAG is a data representation that integrates graph structural information with textual semantic information. Nodes represent textual entities such as documents, sentences, or user-generated content, while edges denote relationships or interaction structures between nodes (He et al., 2023). Formally, we define a TAG as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \{S_n\}_{n\in\mathcal{V}})$, where $\mathcal{V}$ is a set of N nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges, $\mathcal{X}$ is the initial feature of the node, and $\{S_n\}$ is a collection of text sequences associated with the nodes. In this paper, we investigate node classification on TAGs. Specifically, given some labeled nodes $\mathcal{V}_L \subset \mathcal{V}$, the goal is to predict the labels of the remaining unlabeled nodes $\mathcal{V}_U = \mathcal{V} \setminus \mathcal{V}_L$ (He et al., 2023).

2.2 Large Language Models

Large Language Models (LLMs) are neural networks pre-trained on massive text corpora using self-supervised objectives (e.g., masked language modeling or autoregressive prediction), enabling them to capture rich linguistic and contextual patterns (Hernández and Amigó, 2021). Modern LLMs are built upon the Transformer architecture, whose core is the self-attention mechanism. Self-attention allows a model to weigh different tokens in a sequence by projecting each token into three vectors: Query (Q), Key (K), and Value (V). The attention output is computed as:

$$Attention(Q, K, V) = softmax\left(\frac{QK}{\sqrt{d_k}}\right)V, \tag{1}$$

where d_k is the dimension of the key vectors, used for stable gradient scaling (Vaswani et al., 2017). In practice, multi-head attention (MHA) runs multiple such attention functions in parallel and concatenates their outputs, allowing the model to capture diverse semantic and syntactic patterns across different subspaces.

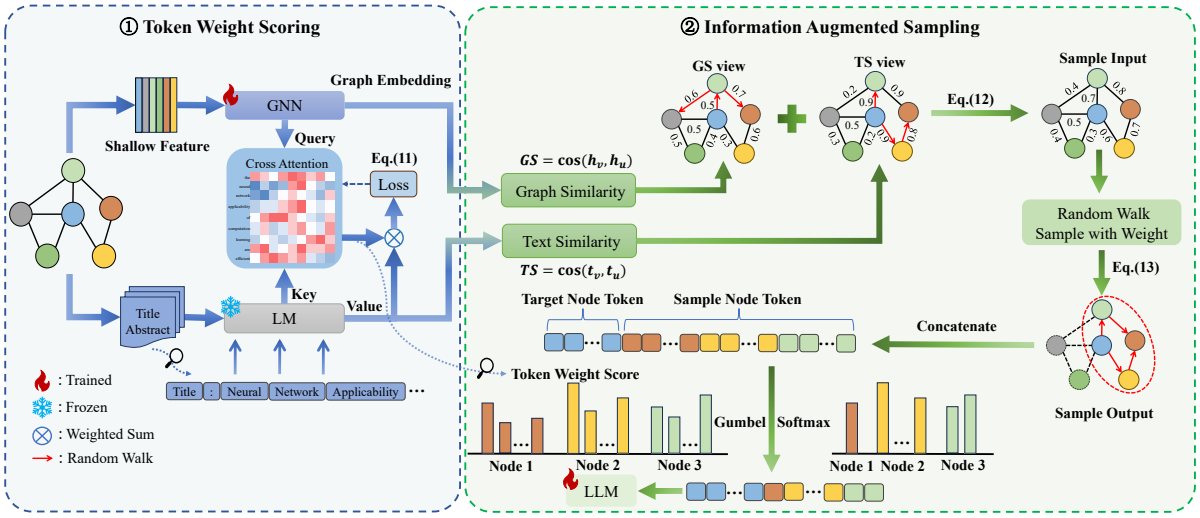

Figure 1: Overall framework of GraphTW. ① The Token Weight Scoring module uses a graph structure to guide token weight scoring, enabling the removal of redundant tokens while preserving key information. ② The Information Augmented Sampling module generates weighted node samples and trains the model using these informative node sequences.

2.3 Graph Neural Networks

GNNs process graph-structured data via iterative message passing. For a node v, the k-th layer updates its representation as:

$$AGG_v^{(k)} = f_{agg}\left(\left\{h_u^{(k-1)}, u \in \mathcal{N}_v^{(k)}\right\}\right), \quad (2)$$

$$h_v^{(k)} = f_{update}\left(h_u^{(k-1)}, AGG_v^{(k)}\right), \quad (3)$$

where $f_{agg}(\cdot)$ denotes the neighbor information aggregation function and $f_{update}(\cdot)$ denotes the node representation update function. In practice, both functions can be implemented using multilayer perceptrons, attention mechanisms, or other differentiable parameterized transformations. $h_v^{(k)}$ is the node embedding of v in the k-th layer. $\mathcal{N}_v^{(k)}$ denotes the set of neighbours of node v in the k-th layer.

3 Methodology

In this section, we present the GraphTW architecture designed for node classification on TAGs. The key idea of this method is to assign weights to node tokens guided by the graph, and then further enhance the node feature representations through graph and semantics-aware sampler strategies. The enhanced representations are ultimately fed into downstream GNNs for node classification tasks.

3.1 Token Weight Scoring

In TAGs, an intuitive and common approach to enable nodes to leverage their contextual information is to concatenate the text attributes of the target node and its neighbors, then feed this concatenated input into a Language Model (LM). However, due to the often substantial number of node neighbors and their text lengths, this strategy leads to a dramatic increase in input sequence length. This significantly elevates the computational overhead of the LM, potentially exceeding its maximum input length capacity and limiting the method's practical application on large-scale graphs.

To mitigate this, we design a token weight scoring module to assign weights to tokens. Given a TAG $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \{S_n\}_{n\in\mathcal{V}})$, we denote the typical GNN as $g_\theta(A, \mathcal{X})$, where θ is the set of learnable parameters, $A \in \{0,1\}^{N\times N}$ is the adjacency matrix of graph, and $\mathcal{X}$ is the node features processed by shallow methods such as bag-of-words (BoW) (Harris, 1954) or skip-gram (Mikolov et al., 2013). Then, we adopt GraphSAGE (Hamilton et al., 2017) as our GNN backbone. For a target node v, GraphSAGE samples a fixed-size neighbor set $\mathcal{N}_v$ and updates its representation as:

$$h_v^{(k)} = f_{\text{update}}\left(h_v^{(k-1)} \parallel f_{\text{agg}}\left(\left\{h_u^{(k-1)} \mid u \in \mathcal{N}_v\right\}\right)\right), \quad (4)$$

where $\parallel$ denotes concatenation. The model is trained for node classification by minimizing the cross-entropy loss on labeled nodes $\mathcal{V}_L$:

$$\mathcal{L} = \mathbb{E}_{v\in\mathcal{V}_L}\mathrm{CE}(\widehat{y}_v := \mathcal{C}(h_v^{(k)}),\ y_v), \quad (5)$$

where $\mathcal{C}$ is a linear classifier, $h_v^{(k)}$ the final-layer

embedding, y_v the ground-truth label, and $\widehat{y}_v$ the predicted label.

After training GraphSAGE, we obtain node graph embeddings that capture topological information. These embeddings serve as query vectors to guide text modeling. For each node, we encode its text using a pre-trained LM, such as RoBERTa (Liu et al., 2019) or BERT (Devlin et al., 2019), to obtain a text embedding:

$$T_n = \mathrm{LM}(S_n) \in \mathbb{R}^d, \tag{6}$$

where d is the output dimension.

To quantify token importance, we design a cross-attention module where the node's graph embedding h_v serves as the query and its token embeddings T_v serve as the key and value. The token weight scoring module can be defined as follows:

$$Weight_v = \sigma\left(\frac{\left(h_v^{(k)} W_q\right)\left(T_v W_k\right)^T}{\sqrt{d}}\right), \tag{7}$$

where $W_q, W_k \in \mathbb{R}^{d\times d'}$ are the parameter matrices for the query and key, and σ denotes the softmax function. The node-level textual representation is then obtained as a weighted sum of token embeddings:

$$s_v = Weight_v T_v. \tag{8}$$

Intuitively, tokens with higher attention weights contribute more to the final representation. Finally, s_v is fed into a linear classifier $\mathcal{C}$ and optimized with cross-entropy loss:

$$\mathcal{L}_{\text{weight}} = \mathbb{E}_{v\in\mathcal{V}_L}\mathrm{CE}\big(\widehat{y}_v := \mathcal{C}(s_v),\ y_v\big). \tag{9}$$

To prevent the token weight scores from becoming overly concentrated during optimization, which may limit the model's ability to capture diverse textual information, we incorporate a KL-divergence regularization term into the training objective. This regularization term constrains the divergence between the token weight distribution and a uniform distribution, thereby suppressing extreme weight assignments and encouraging the model to attend more evenly to multiple informative tokens. Specifically, the regularization term is defined as:

$$\mathcal{L}_{reg} = \mathbb{E}_{v\in\mathcal{V}_L} D_{KL}\left(U \parallel Weight_v\right), \tag{10}$$

where U is a uniform distribution.

The overall training loss of the token weight scoring model can be denoted as:

$$\mathcal{L}_{TWS} = \mathcal{L}_{weight} + \lambda\mathcal{L}_{reg}, \tag{11}$$

where λ is the regularization parameter for controlling the distribution of importance score.

3.2 Information Augmented Sampling

The contextual textual semantics between nodes can provide valuable supplementary information for target nodes. For instance, in citation networks, the textual content of neighboring nodes often contains key terms or concepts relevant to the target node, thereby contributing to a more comprehensive characterization of the node's semantic features.

The textual content of neighboring nodes often provides useful supplementary information for a target node. For example, in citation networks, neighboring papers may contain key terms that help characterize the target node. To capture such context while jointly considering graph structure and text semantics, we adopt a weighted random walk to construct a context node set for each target node. For a node v, the transition weight to a neighbor u is defined as:

$$w_{vu} = \alpha\cdot S\left(h_v, h_u\right) + (1-\alpha)\cdot S\left(t_v, t_u\right), \tag{12}$$

where $\alpha \in [0,1]$ controls the balance of graph structural similarity and textual similarity, and $S(\cdot,\cdot)$ denotes a similarity function. The transition probability is then normalized over neighbors:

$$p_{vu} = \frac{w_{vu}}{\sum_{k\in\mathcal{N}_v} w_{vk}}. \tag{13}$$

This weighting mechanism encourages the random walk to sample nodes that are relevant to the target node in both structural and semantic spaces, providing more informative contextual signals for downstream learning.

Let $\{u_1, u_2, ..., u_n\} \in \mathcal{N}_v$ denote a set of neighboring nodes sampled for node v via the proposed weight-aware random walk strategy. We first take the full token sequence of node v as $T_v = (t_v^1, ..., t_v^{L_v})$, where L_v is its number of valid tokens. For the neighbor nodes, we concatenate their token sequences into a single sequence $T_{\mathcal{N}} = (t_{u_1}^1, ..., t_{u_n}^{L_{u_n}})$. However, directly selecting a subset of tokens from $T_{\mathcal{N}}$ is a discrete operation that is non-differentiable, hindering end-to-end training. To address this, we adopt the Gumbel Softmax (Maddison et al., 2017) to make the selection process differentiable:

$$\widehat{\mathbf{m}}_i = \mathrm{softmax}\left((\log \boldsymbol{w}_i + \boldsymbol{g}_i)/T\right), \tag{14}$$

$$g_{i,j} = -\log(-\log u_{i,j}), \tag{15}$$

where $u_{i,j} \sim Uniform(0,1)$ and T is a temperature parameter. We define $\boldsymbol{w}_i$ as a set of learnable parameters, with initial values given by the token weight score $Weight_i$ from Equation 7.

To select exactly k tokens, where $k = \min(R, |T_{\mathcal{N}}|)$ and $R = K - L_v$ is the remaining capacity up to the maximum length K, we employ a Gumbel-Top-k operator. The relaxed one-hot mask for each token is given by the Gumbel-Softmax distribution, and the top-k indices are sampled in a fully differentiable manner using the straight-through estimator. In practice, we implement the selection as:

$$\mathbf{m} = \text{GumbelTopK}\big(\boldsymbol{w} + \boldsymbol{g},\ k,\ T\big), \tag{16}$$

where $\mathbf{m} \in \{0,1\}^{|T_{\mathcal{N}}|}$ is a binary mask indicating selected tokens, $\boldsymbol{w}$ is the vector of final scores $w(t_i)$, and $\boldsymbol{g}$ is the Gumbel noise vector.

We apply an annealing schedule to the temperature T, initializing it at a high value to encourage exploration and gradually decreasing it during training, thereby forcing the discrete distribution to become increasingly sharp and eventually yielding hard selections. This annealing strategy stabilizes the learning process and improves convergence. Finally, the sequence for node v is obtained as:

$$S_v = (t_v^1, ..., t_v^{L_v}, t_{u_1}^1, ..., t_{u_n}^k). \tag{17}$$

After constructing the sequence S_v for each target node v, we employ a language model $LM(\cdot)$ to encode the sequence, enabling it to model the node-level textual semantics while effectively incorporating graph-aware contextual information, thereby producing the final embedding representation:

$$\mathcal{L}_{LM} = \mathbb{E}_{v \in \mathcal{V}_L} CE\left(\widehat{y}_v := \mathcal{C}_v(S_v), y_v\right). \tag{18}$$

3.3 Graph Neural Network Training

After the above training stage, each node obtains a high-dimensional representation that encodes both semantic and structural information. Based on these enhanced node features, we further train a GNN in an end-to-end manner to capture higher-order dependencies among nodes. Specifically, let H be the node representations from the proposed model. We then apply a GNN to H and define the loss as:

$$\mathcal{L}_{GNN} = \mathbb{E}_{v \in \mathcal{V}_L} CE\left(\widehat{y}_v := \mathcal{C}(GNN(A, H)_v), y_v\right). \tag{19}$$

Moreover, the training processes of the GNN and LM are fully decoupled, allowing the proposed framework to incorporate arbitrary off-the-shelf GNN and LM models flexibly. This cascaded design enables independent optimization of each component, which improves the overall flexibility and scalability of the approach and facilitates seamless integration with diverse model architectures and downstream applications.

4 Experiments

4.1 Datasets

For the node classification task, we evaluate the performance of the proposed GraphTW method on six widely used TAG benchmark datasets: **Cora** (McCallum et al., 2000), **Citeseer** (Giles et al., 1998), **Arxiv_2023** (He et al., 2023), **WikiCS** (Mernyei and Cangea, 2020), **OGBN-Products** (Hu et al., 2020a), and **Ele-Photo** (Yan et al., 2023). Detailed descriptions of these datasets are provided in Appendix A.

4.2 Baselines

To evaluate the effectiveness of our proposed model, we compare it with several baseline models, which can be categorized into three groups: **(1) Traditional GNN-based methods.** These methods aggregate information from neighboring nodes to learn node representations, including MLP (Rumelhart et al., 1985), GCN (Kipf, 2016), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2017), and NodeFormer (Wu et al., 2022). **(2) LM-based methods.** These methods encode node-associated texts using language models to obtain semantic-rich node representations, such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019). **(3) GNN-LLM integration methods.** These methods combine structural modeling of GNNs with semantic understanding of LLMs, e.g., GLEM (Zhao et al., 2022), TAPE (He et al., 2023), SimTeG (Duan et al., 2023), ENGINE (Zhu et al., 2024), and GraphBridge (Wang et al., 2024).

4.3 Implementation Details

For traditional GNN baselines, we use the original node features (e.g., bag-of-words or one-hot). For LM baselines, we directly input raw node texts and fine-tune the language model. In our framework, the token weight scoring module uses a 3-layer GraphSAGE with a hidden dimension of 768 to produce query vectors. For text encod-

ing, we use RoBERTa-large as the language model. Additionally, we fine-tune RoBERTa-large using Low-Rank Adaptation (LoRA) (Hu et al., 2022) with a rank of 8 for the information-augmented sampling module, where the Gumbel temperature is initialized to 1 and linearly annealed over half of the total training steps. The two GraphSAGE models are independently optimized. The dataset splits are strictly aligned: for Cora, Citeseer, and Arxiv_2023, we adopt fixed random seed splits with a 6:2:2 train/validation/test ratio; for WikiCS, OGBN-Products, and Ele-Photo, we follow the original splits provided by their respective sources, consistent with prior work. We implement GraphTW in PyTorch and run experiments on an NVIDIA A100 GPU. Detailed hyperparameters and configurations are provided in Appendix B.

4.4 Main Results

The primary experimental results are summarized in Table 1, which reports the test set accuracies of all models across the datasets considered in this study.

In the category of GNN-based methods, MLP performs poorly across all datasets, primarily because MLP relies solely on node attributes for inference while completely ignoring topological information within the graph structure. This limitation severely restricts its capability when handling highly interconnected TAGs. While other GNN variants (such as GCN, GraphSAGE, and GAT) improve performance by aggregating neighborhood information, they remain suboptimal compared to our proposed method. The bottleneck lies in the fact that these traditional GNNs rely on static and shallow initial feature representations, which do not fully capture the deep semantic nuances of the textual attributes. For instance, on the semantically complex Arxiv_2023 dataset, the best-performing GNN baseline (GraphSAGE) achieves only 68.76% accuracy, lagging behind GraphTW by a significant margin of 20.36%. This performance gap demonstrates that simply aggregating neighborhood features without deeply modeling and integrating text semantics is insufficient for achieving state-of-the-art results on TAGs.

For LM-based methods, pre-trained language models such as BERT and RoBERTa can effectively capture the textual semantics of nodes; however, their performance across all test datasets remains significantly lower than that of advanced LLM-GNN integration methods. The fundamental reason for this systematic gap is the inherent lack of structural information. These models process only the textual attributes of individual nodes and cannot leverage the graph's topology or neighborhood relationships. For example, on the Citeseer citation network dataset, the best-performing LM model, RoBERTa-large, achieves an accuracy of only 72.26%, whereas a model incorporating graph structure information reaches 89.02%, a difference of more than 16 percentage points. This clearly demonstrates that the inability to model graph structure is the primary bottleneck limiting the performance of LM-based methods and highlights the importance of integrating semantic understanding with structural modeling capabilities.

For LLM-GNN integration approaches, existing methods have achieved good results but often treat textual information and graph structural information as relatively independent processing stages, failing to achieve deep semantic interaction. While the GraphBridge method enables text-graph interaction by aggregating neighboring node features, it is constrained by the semantic quality of the neighboring text and is prone to introducing noise. In contrast, our proposed GraphTW innovatively pre-trains a graph model to generate query vectors specifically for downstream tasks. This knowledge-guided fusion strategy endows the model with global semantic prior knowledge and enhanced noise filtering capabilities, enabling deep and precise collaboration between text and graph interactions at a higher semantic level. GraphTW achieves superior performance on the Arxiv_2023 dataset, which contains many isolated nodes. Its pre-trained GNN-based query mechanism reduces reliance on local neighborhood information, making the model more robust and effective in scenarios with sparse graph connectivity. Experimental results conclusively demonstrate that incorporating pre-trained knowledge to guide deep text-graph integration in TAGs is more effective than relying solely on local data aggregation for extracting node features critical to classification tasks.

4.5 Sensitive Analysis

The regularization term λ. The regularization mechanism is designed to discourage the token reduction module from assigning excessively high importance scores to individual tokens. We empirically examine the effect of the regularization coefficient λ defined in Equation 11 by evaluating a range of values $\{0.01, 0.1, 0.5, 1, 10\}$. As shown

Table 1: Experimental results of node classification. We report the mean accuracy with a standard deviation of 5 runs with different random seeds. '-' indicates that datasets do not support for this method. The best result for each dataset is **bolded**.

Category	Methods	Cora	Citeseer	Arxiv_2023	WikiCS	OGBN-Products	Ele-Photo
GNN-based	MLP	$76.12^{\pm 1.51}$	$70.28^{\pm 1.13}$	$65.41^{\pm 0.16}$	$68.11^{\pm 0.76}$	$58.11^{\pm 0.23}$	$62.21^{\pm 0.17}$
	GCN	$88.12^{\pm 1.13}$	$71.98^{\pm 1.32}$	$66.99^{\pm 0.19}$	$76.82^{\pm 0.62}$	$69.84^{\pm 0.52}$	$80.11^{\pm 0.09}$
	GraphSAGE	$87.60^{\pm 1.40}$	$72.44^{\pm 1.11}$	$68.76^{\pm 0.51}$	$76.65^{\pm 0.84}$	$70.64^{\pm 0.20}$	$79.79^{\pm 0.23}$
	GAT	$85.13^{\pm 0.95}$	$72.73^{\pm 1.18}$	$67.61^{\pm 0.24}$	$77.04^{\pm 0.55}$	$69.70^{\pm 0.25}$	$80.38^{\pm 0.37}$
	NodeFormer	$88.48^{\pm 0.33}$	$75.74^{\pm 0.54}$	$67.44^{\pm 0.42}$	$75.47^{\pm 0.46}$	$67.26^{\pm 0.71}$	$77.30^{\pm 0.06}$
LM-based	BERT	$79.70^{\pm 1.70}$	$71.92^{\pm 1.07}$	$77.15^{\pm 0.09}$	$78.13^{\pm 0.63}$	$76.23^{\pm 0.19}$	$68.79^{\pm 0.11}$
	RoBERTa-base	$78.49^{\pm 1.36}$	$71.66^{\pm 1.18}$	$77.33^{\pm 0.16}$	$76.91^{\pm 0.69}$	$76.01^{\pm 0.14}$	$69.12^{\pm 0.15}$
	RoBERTa-large	$79.79^{\pm 1.31}$	$72.26^{\pm 1.80}$	$77.70^{\pm 0.35}$	$77.79^{\pm 0.89}$	$76.29^{\pm 0.27}$	$71.22^{\pm 0.09}$
LLM-GNN Integration	GLEM	$89.11^{\pm 0.19}$	$78.24^{\pm 0.31}$	$78.91^{\pm 0.40}$	$77.99^{\pm 0.72}$	$78.59^{\pm 0.27}$	$82.11^{\pm 0.66}$
	TAPE	$88.56^{\pm 0.88}$	-	$80.21^{\pm 0.31}$	-	$79.76^{\pm 0.23}$	-
	SimTeG	$88.78^{\pm 1.05}$	$79.59^{\pm 1.56}$	$80.51^{\pm 0.33}$	$80.13^{\pm 0.76}$	$78.55^{\pm 0.66}$	$82.49^{\pm 0.17}$
	ENGINE	$88.49^{\pm 1.10}$	$78.02^{\pm 0.87}$	$77.45^{\pm 0.46}$	$80.21^{\pm 0.29}$	$78.83^{\pm 0.80}$	$82.68^{\pm 0.09}$
	GraphBridge	$92.73^{\pm 1.00}$	$86.81^{\pm 1.09}$	$84.79^{\pm 0.29}$	$80.73^{\pm 0.41}$	$80.22^{\pm 0.47}$	$84.18^{\pm 0.15}$
	GraphTW	$\mathbf{95.02^{\pm 0.85}}$	$\mathbf{89.02^{\pm 0.76}}$	$\mathbf{89.12^{\pm 0.86}}$	$\mathbf{81.98^{\pm 0.38}}$	$\mathbf{80.58^{\pm 0.23}}$	$\mathbf{84.42^{\pm 0.36}}$

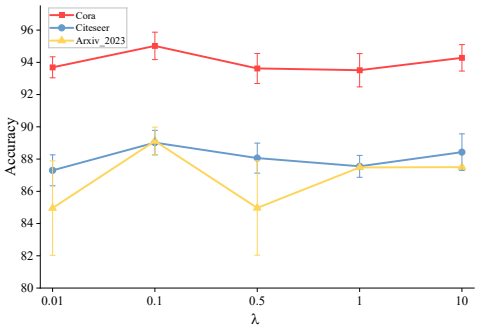

Figure 2: Sensitive analysis of regularization term λ.

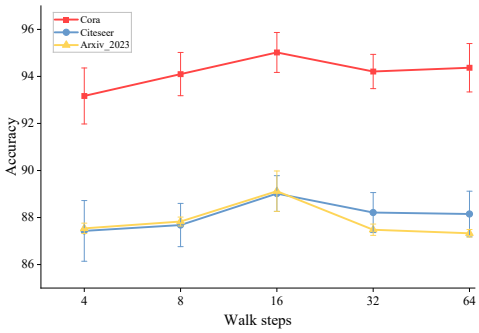

Figure 3: Sensitive analysis of the number of walk steps.

in Figure 2, the best performance is achieved when $\lambda = 0.1$. In contrast, overly small or large values of λ result in overly sharp or overly smooth importance score distributions, respectively, which adversely affect model performance.

The number of walk steps. During the construction of the sequence S defined in Equation 17, we employ a weighted random walk with a restart strategy to sample neighboring nodes, which plays a crucial role in integrating rich contextual textual information. The length of the walk, treated as a key hyper-parameter, determines the range of neighboring nodes involved in the sampling process and consequently affects the scope of contextual information that can be captured.

We empirically analyze the influence of the walk length by evaluating values from the set $\{4, 8, 16, 32, 64\}$. As shown in Figure 3, a small walk length (e.g., 4) fails to capture sufficient contextual information, leading to suboptimal performance. In contrast, excessively long walks (e.g., 64) tend to over-expand the neighborhood, which may obscure fine-grained local information and introduce noisy context. These observations suggest that a moderate number of walk steps, such as 16, provides a better balance between preserving local structural semantics and incorporating broader contextual information, resulting in improved performance.

4.6 Ablation Study

In this section, we evaluate the effectiveness of the token weight scoring mechanism and the weighted random walk sampler strategy. We consider three alternative token reduction baselines, termed Random, TF-IDF, and Truncation, which select tokens randomly, based on TF-IDF scores, and from the beginning of the texts, respectively. Specifically, 'w/o weight' indicates that no weighting information is incorporated during the random walk sam-

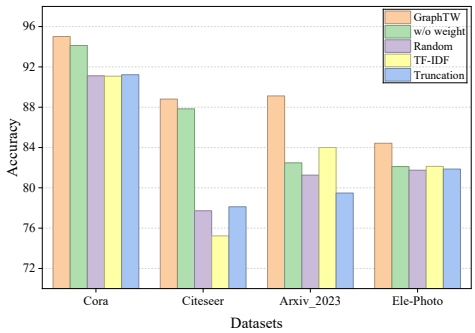

Figure 4: Experimental results of ablation study.

pler process, i.e., an unweighted random walk strategy is adopted. Figure 4 demonstrates that our token reduction module consistently outperforms existing reduction methods, highlighting its effectiveness in identifying and preserving crucial tokens. Moreover, the incorporation of a weighted random walk sampler strategy is crucial, as it facilitates the collection of structurally and semantically relevant contextual information, thereby further enhancing model performance.

5 Related Work

5.1 Representation Learning on TAGs

TAGs are widely used in practice, driving growing research interest in representation learning on TAGs (Luo et al., 2025). Given the strong performance of LLMs in language representation, prior work often uses LLMs to encode node text as initial features for GNNs, e.g., GIANT (Chien et al., 2021), GPT-GNN (Hu et al., 2020b), TextGNN (Zhu et al., 2021). Another line of work enhances node semantics with LLM-generated rich text embeddings as supplementary features (Zhang et al., 2020; Yang and Cui, 2021). For instance, TAPE (He et al., 2023) generates interpretive explanations for node semantics, while KEA (Chen et al., 2024) enriches node text with knowledge entities. Although LLMs offer deep semantic understanding and common-sense knowledge, effectively integrating them with GNNs still faces high computational overhead and optimization challenges. Techniques such as GLEM (Zhao et al., 2022) employ an Expectation-Maximization (EM) framework to alternately update the LM and GNN modules.

Overall, existing TAG learning methods mainly follow a cascaded paradigm, where LLMs and GNNs are trained separately and combined at the representation level (Duan et al., 2023). Some methods introduce interactions between LLMs and GNNs, but often rely on iterative or alternating optimization, leading to high cost and limited scalability (Pang et al., 2022; Li et al., 2021; Bi et al., 2021). Moreover, most approaches focus on node-level semantics and neglect contextual information from neighboring nodes, limiting the effective fusion of graph structure and text semantics.

5.2 Token Reduction for LMs

Existing LM-based text modeling methods show strong representational power for long sequences, but face two major limitations when processing large-scale token sequences. First, the quadratic complexity of standard self-attention leads to rapidly growing computational and memory costs as sequence length increases (Kong et al., 2025). Second, long sequences often contain many redundant tokens irrelevant to downstream tasks, which may distract the model from capturing key information (Bergner et al., 2025).

In TAGs, incorporating neighbor text to enhance node representations further lengthens the input sequence, making direct modeling inefficient and suboptimal. For instance, GraphBridge (Wang et al., 2024) aggregates neighbor representations as queries and uses cross-attention to assign token importance scores for token reduction. Similarly, PoWER-BERT (Goyal et al., 2020) leverages dynamic redundant word representations within self-attention and progressively eliminates word vectors to improve inference efficiency.

6 Conclusion

GraphTW bridges graph topology and text semantics by leveraging graph-guided token weights and a Gumbel selection mechanism to select informative tokens. A weighted random walk retrieves relevant neighbor nodes, and the selected tokens are concatenated with the target node's tokens to form a compact input sequence. Extensive experiments on multiple benchmarks demonstrate that GraphTW consistently outperforms traditional GNNs, language models, and existing graph-text baselines, confirming the effectiveness of fine-grained, graph-guided token selection for node classification on text-attributed graphs.

7 Limitations

This work presents GraphTW, a framework that first computes initial token importance scores using graph embeddings, then retains all target node tokens and employs a Gumbel selection mechanism to pick informative tokens from neighbor nodes sampled via a weighted random walk. However, the weighted random walk combines graph structural similarity and text semantic similarity using a simple weighted sum, which may not fully capture their complex nonlinear interactions. Developing more sophisticated fusion strategies to better model such interactions is a promising direction for future work.

References

Benjamin Bergner, Christoph Lippert, and Aravindh Mahendran. 2025. Token cropr: Faster vits for quite a few tasks. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 9740–9750.

Shuxian Bi, Chaozhuo Li, Xiao Han, Zheng Liu, Xing Xie, Haizhen Huang, and Zengxuan Wen. 2021. Leveraging bidding graphs for advertiser-aware relevance modeling in sponsored search. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2215–2224.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and 1 others. 2024. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

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*.

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

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *International Conference on Machine Learning*, pages 3690–3699. PMLR.

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

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Bryan Hooi, and 1 others. 2023. Explanations as features: Llm-based features for text-attributed graphs. *arXiv preprint arXiv:2305.19523*, 2(4):8.

Adrián Hernández and José M Amigó. 2021. Attention mechanisms and their applications to complex systems. *Entropy*, 23(3):283.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

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

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

TN Kipf. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Zhenglun Kong, Yize Li, Fanhu Zeng, Lei Xin, Shvat Messica, Xue Lin, Pu Zhao, Manolis Kellis, Hao Tang, and Marinka Zitnik. 2025. Token reduction should go beyond efficiency in generative models–from vision, language to multimodality. *arXiv preprint arXiv:2505.18227*.

Chaozhuo Li, Bochen Pang, Yuming Liu, Hao Sun, Zheng Liu, Xing Xie, Tianqi Yang, Yanling Cui, Liangjie Zhang, and Qi Zhang. 2021. Adsgnn: Behavior-graph augmented relevance modeling in sponsored search. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 223–232.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ruilin Luo, Tianle Gu, Lin Wang, Yunfeng Zhou, Songtao Jiang, Lei Wang, and Yujiu Yang. 2025. Fair text-attributed graph representation learning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 14330–14353.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 188–197.

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*, pages 1836–1845.

Bochen Pang, Chaozhuo Li, Yuming Liu, Jianxun Lian, Jianan Zhao, Hao Sun, Weiwei Deng, Xing Xie, and Qi Zhang. 2022. Improving relevance modeling via heterogeneous behavior graph learning in bing ads. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3713–3721.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. Learning internal representations by error propagation. Technical report.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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

Yaoke Wang, Yun Zhu, Wenqiao Zhang, Yueting Zhuang, Liyunfei Liyunfei, and Siliang Tang. 2024. Bridging local details and global context in text-attributed graphs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14830–14841.

Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. 2022. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in Neural Information Processing Systems*, 35:27387–27401.

Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, and 1 others. 2023. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 36:17238–17264.

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

Yiping Yang and Xiaohui Cui. 2021. Bert-enhanced text graph neural network for classification. *Entropy*, 23(11):1536.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR.

Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*.

Wenqiao Zhang, Zheqi Lv, Hao Zhou, Jia-Wei Liu, Juncheng Li, Mengze Li, Yunfei Li, Dongping Zhang, Yueting Zhuang, and Siliang Tang. 2024. Revisiting the domain shift and sample uncertainty in multi-source active domain transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16751–16761.

Huanjing Zhao, Beining Yang, Yukuo Cen, Junyu Ren, Chenhui Zhang, Yuxiao Dong, Evgeny Kharlamov, Shu Zhao, and Jie Tang. 2024. Pre-training and prompting for few-shot node classification on text-attributed graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4467–4478.

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

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

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

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

A Dataset Description

We evaluated our method on six widely used text-attributed graph datasets. The details of these datasets are summarized as follows:

Cora (McCallum et al., 2000) dataset comprises 2708 scientific publications, categorized into seven classes: Case-Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning, and Theory. In the citation network, each publication cites or is cited by at least one other paper, resulting in a total of 5429 edges.

Citeseer (Giles et al., 1998) contains 3186 publications grouped into six categories: Agents, Machine Learning, Information Retrieval, Databases, Human–Computer Interaction, and Artificial Intelligence. Each publication is classified according to its title and abstract, making it suitable for evaluating text-based node classification methods.

Arxiv_2023 (He et al., 2023) dataset represents a directed citation network of computer science papers from arXiv published in 2023 or later. In this graph, nodes correspond to papers and directed edges indicate citation relationships, similar to the OGBN-ArXiv dataset. Each paper is labeled with one of 40 subject areas, such as cs.AI, cs.LG, and cs.OS, with labels assigned by the authors and arXiv moderators. The dataset is commonly used for benchmarking node classification methods on large-scale citation networks.

WikiCS (Mernyei and Cangea, 2020) is derived from Wikipedia and is designed for benchmarking Graph Neural Networks. It comprises ten computer science branches as classes, exhibiting high connectivity, and node features are extracted from the corresponding article texts.

OGBN-Products (Hu et al., 2020a) dataset, originally comprising 2 million nodes and 61 million edges, is reduced using a node sampler strategy from TAPE (He et al., 2023) to form the OGBN-Products subset, containing 54,000 nodes and 74,000 edges. In this graph, nodes correspond to Amazon products, and edges represent co-purchase relationships. The task is to classify each product into one of 47 top-level categories.

ELe-Photo (Yan et al., 2023) derived from the Amazon Electronics dataset (Ni et al., 2019), is a product graph where nodes represent electronic items and edges indicate frequent co-purchases or co-views. Each node is labeled according to a three-level product taxonomy. Node text attributes are obtained from the user review with the highest number of votes, or a randomly selected review if no highly-voted reviews are available. The dataset is used for the task of classifying products into 12 categories.

B Training Details

In this section, we give the implementation details about our method GraphTW.

Token Weight Scoring Module. We first train the GNN model with the learning rate set to 0.01. Subsequently, the cross-attention module and the classifier are jointly trained while keeping the parameters of the language encoder frozen. For each dataset, the model is trained for 100 epochs, and an early stopping strategy with a patience of 10 epochs is employed. In addition, the learning rate is selected by searching over {1e-3, 5e-4, 1e-4}.

Information Augmented Sampling Module. To obtain enriched node representations, we employ a weighted random walk sampler strategy to construct node sequences, where the walk length is selected from $\{8, 16, 32\}$. During this stage, the language models are trained for a number of epochs chosen from $\{10, 20\}$, and optimization is performed using the AdamW optimizer. For BERT-base and RoBERTa-base, full-parameter fine-tuning is applied, with the learning rate selected from {1e-4, 5e-5, 1e-5}. In contrast, RoBERTa-large is fine-tuned using a LoRA strategy with rank 8, and the learning rate is searched over {1e-3, 5e-4, 1e-4}. For the LLaMA2-7B model reported in Table 4, we adopt a LoRA configuration with rank 4 and explore learning rates within {5e-4, 1e-4, 5e-5}.

Graph Neural Networks Module. After obtaining node representations from the language models, we proceed to train the GNN models

Table 2: Data statistics. Nodes, Edges, Classes, and Avg.degrees mean the number of nodes, edges, classes, and average degrees for each dataset, respectively. Avg.tokens represents the average number of tokens per node in each dataset when using the RoBERTa-base's tokenizer (Wang et al., 2024).

Datasets	**Nodes**	**Edges**	**Avg.tokens**	**Avg.degrees**	**Classes**
Cora	2708	5429	194	3.90	7
Citeseer	3186	4277	196	1.34	6
Arxiv_2023	46198	78543	253	1.70	40
WikiCS	11701	215863	545	36.70	10
OGBN-Products(subset)	54025	74420	163	2.68	47
Ele-Photo	48362	500928	185	18.07	12

(e.g., GraphSAGE). The training process is conducted for a variable number of epochs selected from $\{100, 200, 300\}$, and an early stopping strategy with a patience of 20 epochs is applied for each dataset. Optimization is performed using the Adam optimizer, with the learning rate tuned over $\{$1e-2, 5e-3, 1e-3$\}$.

C Alternative LMs for Weight Scoring

Table 3: Results using BERT-base and RoBERTa-base as alternative language model backbones for token reduction

Datasets	$\text{GraphTW}_{\text{(BERT-base)}}$	$\text{GraphTW}_{\text{(RoBERTa-base)}}$
Cora	$93.23^{\pm 0.76}$	$94.63^{\pm 0.96}$
Citeseer	$85.68^{\pm 1.06}$	$88.65^{\pm 0.87}$
Arxiv_2023	$83.32^{\pm 0.16}$	$89.06^{\pm 0.46}$
WikiCS	$78.65^{\pm 0.52}$	$79.65^{\pm 0.38}$
OGBN-Products	$76.62^{\pm 0.56}$	$78.42^{\pm 0.45}$
Ele-Photo	$79.88^{\pm 0.12}$	$80.36^{\pm 0.23}$

The proposed token weight scoring module can be seamlessly integrated with a wide range of language model backbones. In this section, we evaluate its effectiveness by adopting BERT-base (Devlin et al., 2019) and RoBERTa-base (Liu et al., 2019) as alternative backbone models. Table 3 reports the experimental results obtained when applying these backbone models to the token reduction task.

D Effectiveness with LLM Backbone

As shown in Section 4.4, the proposed approach remains effective when instantiated with relatively compact language models, such as RoBERTa-large. Beyond this setting, we further extend our framework to support causal LLMs, which have recently demonstrated strong performance across a wide range of natural language processing tasks (Zhang et al., 2024). We fine-tune the LLM using LoRA with a rank setting of 4 and report the resulting accuracy. The experimental results obtained with LLaMA2-7B (Touvron et al., 2023) as the backbone LLM are summarized in Table 4. Our method consistently outperforms both standalone LM-based approaches (e.g., LLaMA2) and existing LLM-based integration methods (e.g., SimTeG and ENGINE). These results validate the effectiveness and generality of our approach under an LLM backbone and further demonstrate the importance of effectively bridging local and global information in text-attributed graph modeling.

Table 4: Experimental results when utilizing LLaMA2-7B as the Large Language Model backbone. We use boldface to denote the best performance.

Methods	Cora	WikiCS	Ele-Photo
LLaMA2	$82.80^{\pm 1.37}$	$80.82^{\pm 0.48}$	$72.06^{\pm 0.10}$
$\text{SimTeG}_{\text{(LLaMA2)}}$	$92.84^{\pm 0.13}$	$82.55^{\pm 0.51}$	$82.05^{\pm 0.17}$
$\text{ENGINE}_{\text{(LLaMA2)}}$	$91.48^{\pm 0.32}$	$81.56^{\pm 0.97}$	$83.75^{\pm 0.08}$
$\text{GraphBridge}_{\text{(LLaMA2)}}$	$93.64^{\pm 0.44}$	$84.18^{\pm 0.68}$	$84.35^{\pm 0.12}$
$\text{GraphTW}_{\text{(LLaMA2)}}$	$\mathbf{94.52}^{\pm \mathbf{1.13}}$	$\mathbf{84.96}^{\pm \mathbf{0.62}}$	$\mathbf{84.69}^{\pm \mathbf{0.46}}$

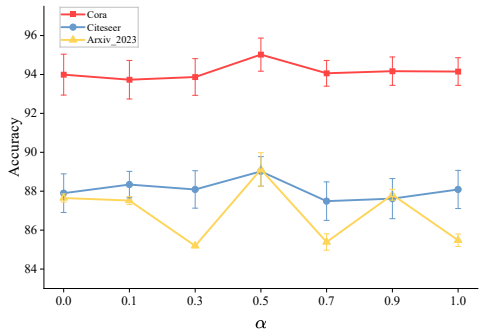

Figure 5: Sensitive analysis of the balance coefficient α.

E The balance coefficient α.

We further investigate the effect of the balance coefficient α in Equation 12, which controls the relative

contribution of graph structural similarity and textual semantic similarity in the weighted random walk sampler strategy. Specifically, we evaluate $\alpha \in \{0, 0.1, 0.3, 0.5, 0.7, 0.9, 1\}$ to analyze how varying the emphasis between structural and semantic information influences model performance.

As illustrated in Figure 5, the model achieves the best performance when $\alpha = 0.5$, indicating that assigning equal importance to graph structure and textual semantics yields the most effective contextual node sampler. When α is close to 0, the sampler strategy relies primarily on textual similarity, which leads to a slight performance degradation due to insufficient exploitation of graph structure. Conversely, when α approaches 1, the random walk is dominated by structural similarity, which may limit semantic diversity and reduce the effectiveness of contextual information aggregation. These results indicate that neither structural nor textual information alone is sufficient; instead, a balanced integration of both yields the most effective performance.