STAGE: Simplified Text-Attributed Graph Embeddings Using Pre-trained LLMs

Aaron Zolnai-Lucas[1]*, **Jack Boylan**[1]*, **Chris Hokamp**[1], **Parsa Ghaffari**[1]

[1]Quantexa,

Correspondence: {firstname}{lastname}@quantexa.com

Abstract

We present Simplified Text-Attributed Graph Embeddings (STAGE), a straightforward yet effective method for enhancing node features in Graph Neural Network (GNN) models that encode Text-Attributed Graphs (TAGs). Our approach leverages Large-Language Models (LLMs) to generate embeddings for textual attributes. STAGE achieves competitive results on various node classification benchmarks while also maintaining a simplicity in implementation relative to current state-of-the-art (SoTA) techniques. We show that utilizing pre-trained LLMs as embedding generators provides robust features for ensemble GNN training, enabling pipelines that are simpler than current SoTA approaches which require multiple expensive training and prompting stages. We also implement diffusion-pattern GNNs in an effort to make this pipeline scalable to graphs beyond academic benchmarks.

1 Introduction

A Knowledge Graph (KG) typically includes entities (represented as nodes), relationships between entities (represented as edges), and attributes of both entities and relationships (Ehrlinger and Wöß, 2016). These attributes, referred to as metadata, are often governed by a domain-specific ontology, which provides a formal framework for defining the types of entities and relationships as well as their properties. KGs can be used to represent structured information about the world in diverse settings, including medical domain models (Koné et al., 2023), words and lexical semantics (Miller, 1995), and commercial products (Chiang et al., 2019).

Text-Attributed Graphs (TAGs) can be viewed as a subset of KGs, where some node and edge metadata is represented by unstructured or semi-structured natural language text (Yang et al., 2023). Examples of unstructured data values in TAGs could include the research article text representing the nodes of a citation graph, or the content of social media posts that are the nodes of an interaction graph extracted from a social media platform. Many real-world datasets are naturally represented as TAGs, and studying how to best represent and learn using these datasets has received attention from the fields of graph learning, natural language processing (NLP), and information retrieval.

Graph Learning and LLMs With the emergence of LLMs as powerful general purpose reasoning agents, there has been increasing interest in integrating KGs with LLMs (Pan et al., 2024). Current SoTA approaches combining graph learning with (L)LMs follow either an **iterative** or a **cascading** method. *Iterative* methods involve jointly training an LM and a GNN for the given task. While this approach can produce a task-specific feature space, it may be complex and resource-intensive, particularly for large graphs. In contrast, *cascading* methods first apply an LM to extract node features which are then used by a downstream GNN model. Cascading models demonstrate excellent performance on TAG tasks (He et al., 2024; Duan et al., 2023a), although they often require multiple stages of training targeted at each pipeline component. More recent cascading techniques implement an additional step, known as text-level enhancement (Chen et al., 2024), whereby textual features are augmented using an LLM.

Simplifying Node Representation Generation To the best of our knowledge, all existing cascading approaches require multiple rounds of data generation or finetuning to achieve satisfactory results on TAG tasks (He et al., 2024; Duan et al., 2023a; Chen et al., 2024). This bottleneck increases the difficulty of applying such methods to real-world graphs. Our proposed method, STAGE, aims to simplify existing approaches by foregoing LM finetuning, and only making use of a single pre-

*Authors contributed equally.

Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024), pages 92–104

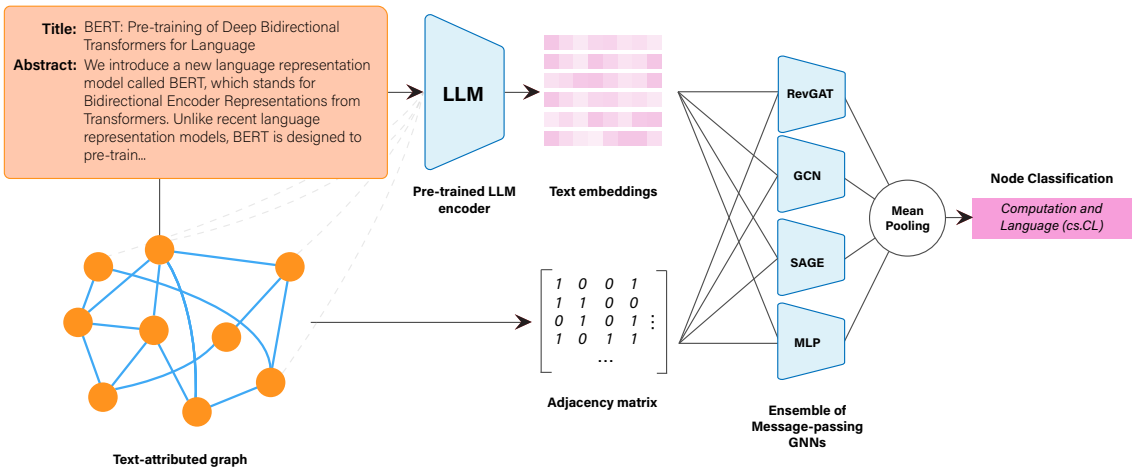

Figure 1: Our proposed approach to node classification. Firstly, the textual attributes of the input graph nodes are encoded using an off-the-shelf LLM. The text embeddings will be used alongside the graph adjacency matrix as input to train a downstream ensemble of GNNs. GNN predictions are then mean-pooled to obtain the final prediction.

trained LLM as the node embedding model, without data augmentation via prompting. We study possible configurations of this simplified pipeline and demonstrate that this method achieves competitive performance while significantly reducing the complexity of training and data preparation.

Scalable GNN Architectures The exponentially growing receptive field required during training of most message-passing GNNs is another bottleneck in both cascading and iterative approaches, becoming computationally intractable for large graphs (Duan et al., 2023b; Liu et al., 2024). Because we wish to study approaches that can be applied in real-world settings, we also explore the implementation of diffusion-pattern GNNs, such as SimpleGCN (Wu et al., 2019) and SIGN (Frasca et al., 2020), which may enable STAGE to be applied to much larger graphs beyond the relatively small academic benchmarks. Our code is available at `https://github.com/aaronzo/STAGE`.

Concretely, this work studies several ways to make learning on TAGs more efficient and scalable:

- **Single Training Stage:** We perform ensemble GNN training with a fixed LLM as the node feature generator, which significantly reduces training time by eliminating the need for multiple large model training runs.
- **No LLM Prompting:** We do not prompt an LLM for text-level augmentations such as predictions or explanations. Instead, we use only the text attributes provided in the dataset.
- **Direct Use of LLM as Text Embedding Model:** Using an off-the-shelf LLM as the embedding model makes this method adaptable to new models and datasets. We study several alternative base models for embedding generation.
- **Diffusion-pattern GNN implementation:** We contribute an investigation into diffusion-pattern GNNs which enable this method to scale to larger graphs.

The rest of the paper is organized as follows: section 2 gives an overview of related work, section 3 discusses our approach in detail, section 4 studies the performance of STAGE in various settings, and section 5 is a discussion of the experimental results.

2 Background

Text-Attributed Graphs Yan et al. (2023) suggest that integrating topological data with textual information can significantly improve the learning outcomes on various graph-related tasks. Chien et al. (2022) incorporate graph structural information into the pre-training stage of pre-trained language models (PLMs), achieving improved performance albeit with additional training overhead, while Liu et al. (2023) further adopt sentence embedding models to unify the text-attribute and

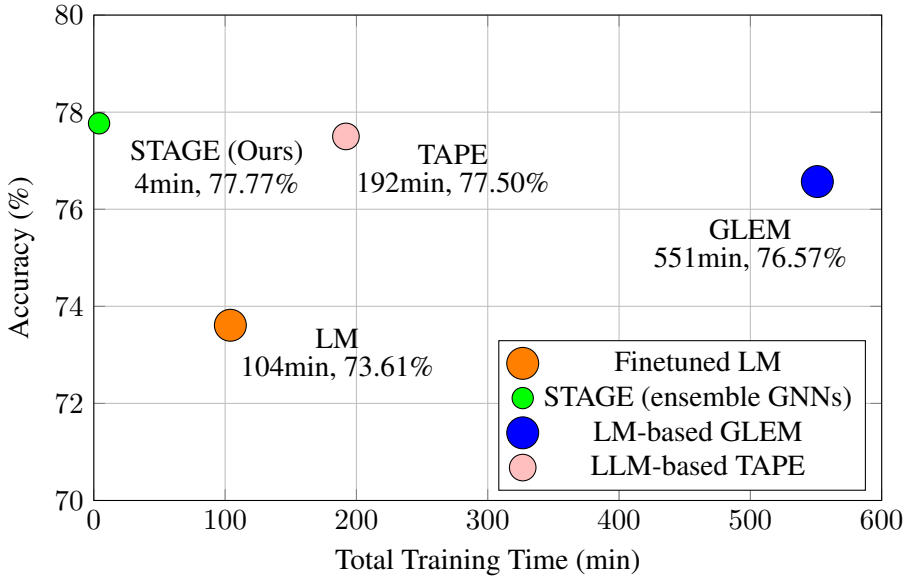

Figure 2: The performance trade-off between node classification accuracy and total training time on `ogbn-arxiv` for SoTA LM-GNN methods. The STAGE model uses text embeddings generated from Salesforce-Embedding-Mistral and an ensemble of GNNs (GCN, SAGE and RevGAT) and MLP. The size of each marker indicates the total number of trainable parameters. Figure adapted from (He et al., 2024).

graph structure feature space, proposing a unified model for diverse tasks across multiple datasets.

LLMs as Text Encoders General purpose text embedding models, used in both finetuned and zero-shot paradigms, are a standard component of modern NLP pipelines (Mikolov et al., 2013; Pennington et al., 2014; Reimers and Gurevych, 2019). As LLMs have emerged as powerful zero-shot agents, many studies have considered generating text embeddings as an auxiliary output (Muennighoff, 2022; Mialon et al., 2023). BehnamGhader et al. (2024) introduce LLM2Vec, an unsupervised method to convert LLMs into powerful text encoders by using bidirectional attention, masked next token prediction and contrastive learning, achieving state-of-the-art performance on various text embedding benchmarks.

Language Models and GNNs Graph Neural Networks have been successfully applied to node classification and link prediction tasks, demonstrating improved performance when combined with textual features from nodes (Kipf and Welling, 2017; Li et al., 2022b). Several studies show that finetuning pre-trained Language Models (PLMs), such as BERT (Devlin et al., 2019) and DeBERTa (He et al., 2021), enhances GNN performance by leveraging textual node features (Chen et al., 2024; Duan et al., 2023a; He et al., 2024).

Recent research has explored the integration of LLMs with GNNs, particularly for TAGs. LLMs contribute deep semantic understanding and commonsense knowledge, potentially boosting GNNs' effectiveness on downstream tasks. However, combining LLMs with GNNs poses computational challenges. Techniques like GLEM (Zhao et al., 2023) use the Expectation Maximization framework to alternate updates between LM and GNN modules.

Other approaches include the TAPE method, which uses GPT (OpenAI, 2023; OpenAI et al., 2024) models for data augmentation, enhancing GNN performance through enriched textual embeddings (He et al., 2024). SimTeG demonstrates that parameter-efficient finetuning (PEFT) PLMs can yield competitive results (Duan et al., 2023a). (Ye et al., 2024) suggest that finetuned LLMs can match or exceed state-of-the-art GNN performance on various benchmarks.

Building on these insights, the STAGE method focuses on efficient and scalable learning for TAGs by utilizing zero-shot capabilities of LLMs to generate representations without extensive task-specific tuning or auxiliary data generation.

3 Approach

Our cascading approach consists of two steps:

- A zero-shot LLM-based embedding generator is used to encode the title and abstract (or

equivalent textual attribute) of each node. We denote the generated node embeddings as $\mathcal{X}$.

- An ensemble of GNN architectures are trained on $\mathcal{X}$, and their predictions are mean-pooled to obtain the final node predictions.

Ensembling the predictions from multiple GNN architectures was motivated by our observation of strong performance by different models across different datasets.

3.1 Text Embedding Retrieval

For the text embedding model, we select a general-purpose embedding LLM that ranks highly on the Massive Text Embedding Benchmark (MTEB) Leaderboard[1]. Specifically, we evaluate `gte-Qwen1.5-7B-instruct`, `LLM2Vec-Meta-Llama-3-8B-Instruct`, and `SFR-Embedding-Mistral`. MTEB ranks embedding models based on their performance across a wide variety of information retrieval, classification and clustering tasks. This model is used out-of-the-box without any finetuning. An appealing aspect of LLM-based embeddings is the possibility to add instructions alongside input text to bias the embeddings for a given task. We empirically evaluate the effect of instruction biased embeddings is in Table 2 of section 4.

Node representations $\mathcal{X}$ are generated using only the title and abstract, or equivalent textual node attributes, omitting the LLM predictions and explanations provided by (He et al., 2024). $\mathcal{X}$ will then be used as enriched node feature vectors for training a downstream GNN ensemble.

3.2 GNN Training

Using the previously generated embeddings $\mathcal{X}$ as node features, we train an ensemble of GNN models on the node classification task:

$$\text{Loss}_{\text{cls}} = \mathcal{L}_\theta \left(\phi(\text{GNN}(\mathcal{X}, \mathcal{A})), \mathbf{Y} \right), \quad (1)$$

where $\phi(\cdot)$ is the classifier, $\mathcal{A}$ is the adjacency matrix of the graph and $\mathcal{Y}$ is the label. For the GNN architectures we choose GCN (Kipf and Welling, 2017), SAGE (Hamilton et al., 2018) and RevGAT (Li et al., 2022a). We also evaluate a multi-layer perceptron (MLP) (Haykin, 1994) among our GNN models. To combine the predictions from each of the K models in the ensemble, we compute the mean prediction as follows:

$$\bar{\mathbf{p}} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{p}_k, \quad (2)$$

Cross-entropy loss is used to compute the loss value.

Diffusion-based GNNs For a graph G with node features $\mathcal{X}$, a diffusion operator is a matrix A_{OP} with the same dimensions as the adjacency matrix of G. Diffused features $\mathcal{H}$ are then calculated via $\mathcal{H} = A_{\text{OP}}\mathcal{X}$.

We explored Simple-GCN (Wu et al., 2019) and SIGN (Frasca et al., 2020), both of which employ adjacency-based diffusion operators to pre-aggregate features across the graph before training. SIGN is a generalization of Simple-GCN, to extend to Personalized-PageRank (Page et al., 1998) and triangle-based operators. This allows expensive computation to be carried out by distributed computing clusters or efficient sparse graph routines such as GraphBLAS (Davis, 2019), which do not need to back-propagate through graph convolution. The prediction head can then be a shallow MLP or logistic regression. We provide implementation specifics in appendix section C to ensure repeatability.

3.3 Parameter-efficient Finetuning LLM

Motivated by the node classification performance gains seen by (Duan et al., 2023a) using PEFT, we finetune an LLM on the node classification task. Concretely, we use an LLM embedding model with a low-rank adapter (LoRA) (Hu et al., 2021a) and a densely connected classifier head. The pre-trained LLM weights remain frozen as the model trains on input text T to reduce loss according to:

$$\text{Loss}_{\text{cls}} = \mathcal{L}(\phi(\text{LLM}(T)), Y) \quad (3)$$

where $\phi(\cdot)$ is the classifier head and Y is the label. Again, we use cross-entropy loss to compute the loss value.

4 Experiments

We investigate the performance of STAGE over five TAG benchmarks: *ogbn-arxiv* (Hu et al., 2021b), a dataset of arXiv papers linked by citations; *ogbn-products* (Hu et al., 2021b), representing an Amazon product co-purchasing network; *PubMed* (Sen et al., 2008), a citation network of diabetes-related scientific publications; *Cora* (McCallum et al.,

[1] https://huggingface.co/spaces/mteb/leaderboard

Dataset	Method	h_{shallow}	h_{GIANT}	GPT3.5	LM_{finetune}	h_{TAPE}	h_{STAGE}**(OURS)**
Cora	MLP	0.6388 ± 0.0213	0.7196 ± 0.0000	0.6769	0.7606 ± 0.0378	0.8778 ± 0.0485	0.7680 ± 0.0228
	GCN	0.8911 ± 0.0015	0.8423 ± 0.0053	0.6769	0.7606 ± 0.0378	0.9119 ± 0.0158	0.8704 ± 0.0105
	SAGE	0.8824 ± 0.0009	0.8455 ± 0.0028	0.6769	0.7606 ± 0.0378	0.9290 ± 0.0307	0.8722 ± 0.0063
	RevGAT	0.8911 ± 0.0000	0.8353 ± 0.0038	0.6769	0.7606 ± 0.0378	0.9280 ± 0.0275	0.8639 ± 0.0129
	Ensemble	-	-	-	-	-	0.8824 ± 0.0155
PubMed	MLP	0.8635 ± 0.0032	0.8175 ± 0.0059	0.9342	0.9494 ± 0.0046	0.9565 ± 0.0060	0.9142 ± 0.0122
	GCN	0.8031 ± 0.0425	0.8419 ± 0.0050	0.9342	0.9494 ± 0.0046	0.9431 ± 0.0043	0.8960 ± 0.0042
	SAGE	0.8881 ± 0.0002	0.8372 ± 0.0082	0.9342	0.9494 ± 0.0046	0.9618 ± 0.0053	0.9087 ± 0.0064
	RevGAT	0.8850 ± 0.0005	0.8502 ± 0.0048	0.9342	0.9494 ± 0.0046	0.9604 ± 0.0047	0.8654 ± 0.0952
	Ensemble	-	-	-	-	-	0.9265 ± 0.0068
ogbn-arxiv	MLP	0.5336 ± 0.0038	0.7308 ± 0.0006	0.7350	0.7361 ± 0.0004	0.7587 ± 0.0015	0.7517 ± 0.0011
	GCN	0.7182 ± 0.0027	0.7329 ± 0.0010	0.7350	0.7361 ± 0.0004	0.7520 ± 0.0005	0.7377 ± 0.0010
	SAGE	0.7171 ± 0.0017	0.7435 ± 0.0014	0.7350	0.7361 ± 0.0004	0.7672 ± 0.0007	0.7596 ± 0.0040
	RevGAT	0.7083 ± 0.0017	0.7590 ± 0.0019	0.7350	0.7361 ± 0.0004	0.7750 ± 0.0012	0.7638 ± 0.0054
	Ensemble	-	-	-	-	-	0.7777 ± 0.0019
ogbn-products	MLP	0.5385 ± 0.0017	0.6125 ± 0.0078	0.7440	0.7297 ± 0.0023	0.7878 ± 0.0082	0.7277 ± 0.0054
	GCN	0.7052 ± 0.0051	0.6977 ± 0.0042	0.7440	0.7297 ± 0.0023	0.7996 ± 0.0041	0.7679 ± 0.0109
	SAGE	0.6913 ± 0.0026	0.6869 ± 0.0011	0.7440	0.7297 ± 0.0023	0.8137 ± 0.0043	0.7795 ± 0.0012
	RevGAT	0.6964 ± 0.0017	0.7189 ± 0.0030	0.7440	0.7297 ± 0.0023	0.8234 ± 0.0036	0.8083 ± 0.0051
	Ensemble	-	-	-	-	-	0.8140 ± 0.0033
tape-arxiv23	MLP	0.6202 ± 0.0064	0.5574 ± 0.0032	0.7356	0.7358 ± 0.0006	0.8385 ± 0.0246	0.7940 ± 0.0022
	GCN	0.6341 ± 0.0062	0.5672 ± 0.0061	0.7356	0.7358 ± 0.0006	0.8080 ± 0.0215	0.7678 ± 0.0024
	SAGE	0.6430 ± 0.0037	0.5665 ± 0.0032	0.7356	0.7358 ± 0.0006	0.8388 ± 0.0264	0.7894 ± 0.0024
	RevGAT	0.6563 ± 0.0062	0.5834 ± 0.0038	0.7356	0.7358 ± 0.0006	0.8423 ± 0.0256	0.7880 ± 0.0023
	Ensemble	-	-	-	-	-	0.8029 ± 0.0020

Table 1: Node classification accuracy for the Cora, PubMed, ogbn-arxiv, ogbn-products, and tape-arxiv23 datasets. The experiment is run over four seeds, with mean accuracy and standard deviation shown. The best results are coloured green (first), yellow (second), and orange (third). For h_{STAGE}, we use SFR-Embedding-Mistral as the embedding model on TA features only, and the simple task instruction to bias the embeddings. We adapt the table from (He et al., 2024) and include our results.

2000), a dataset of scientific publications categorized into one of seven classes; and *tape-arxiv23* (He et al., 2024), focusing on arXiv papers published after the 2023 knowledge cut-off for GPT3.5. We use the subset of *ogbn-products* provided by (He et al., 2024). Further details can be found in appendix Table 7.

For each experiment using Cora, PubMed or *tape-arxiv23*, 60% of the data was allocated for training, 20% for validation, and 20% for testing. For the `ogbn-arxiv` and `ogbn-products` datasets, we adopted the standard train/validation/test split provided by the Open Graph Benchmark (OGB)[2] (Hu et al., 2021b).

Our main results can be seen in Table 1. Multiple GNN models are trained using embeddings from a pre-trained LLM as node features. We ensemble the predictions across model architectures by taking the mean prediction.

Node classification accuracy is provided for various datasets, measured across multiple methods and feature types. Each column represents a specific metric or method:

- $\mathbf{h_{shallow}}$: Performance using shallow features, indicating basic attributes provided as part of each dataset
- $\mathbf{h_{GIANT}}$: Results obtained by using GIANT features as proposed by (Chien et al., 2022), designed to incorporate graph structural information into LM training
- **GPT3.5**: Accuracy when using zero-shot predictions from GPT-3.5-turbo, demonstrating the utility of state-of-the-art language models in a zero-shot setting
- $\mathbf{LM_{finetune}}$: Performance metrics reported by (He et al., 2024) after finetuning the DeBERTa (He et al., 2021) model on labeled nodes from the graph, showing the benefits of supervised finetuning
- $\mathbf{h_{TAPE}}$: Shows results for the TAPE features (He et al., 2024), which includes the original textual attributes of the node, GPT-generated predictions for each node, and GPT-generated

[2] https://ogb.stanford.edu/

Dataset	Method	$h_{\text{no instruction}}$	$h_{\text{task instruction}}$	$h_{\text{graph-aware-instruction}}$
Cora	MLP	0.7772 ± 0.0205	0.7680 ± 0.0228	0.7763 ± 0.0193
	GCN	0.8612 ± 0.0121	0.8704 ± 0.0105	0.8718 ± 0.0085
	SAGE	0.8833 ± 0.0125	0.8722 ± 0.0063	0.8704 ± 0.0109
	RevGAT	0.8630 ± 0.0119	0.8639 ± 0.0129	0.8676 ± 0.0125
	Ensemble	0.8930 ± 0.0086	0.8824 ± 0.0155	0.8875 ± 0.0118
PubMed	MLP	0.9305 ± 0.0052	0.9142 ± 0.0122	0.9185 ± 0.0145
	GCN	0.9021 ± 0.0034	0.8960 ± 0.0042	0.8978 ± 0.0046
	SAGE	0.9268 ± 0.0052	0.9087 ± 0.0064	0.9126 ± 0.0024
	RevGAT	0.8637 ± 0.0942	0.8654 ± 0.0952	0.9211 ± 0.0022
	Ensemble	0.9358 ± 0.0035	0.9265 ± 0.0068	0.9313 ± 0.0025
ogbn-arxiv	MLP	0.7417 ± 0.0015	0.7517 ± 0.0011	0.7519 ± 0.0028
	GCN	0.7336 ± 0.0029	0.7377 ± 0.0010	0.7367 ± 0.0045
	SAGE	0.7515 ± 0.0027	0.7596 ± 0.0040	0.7559 ± 0.0039
	RevGAT	0.7629 ± 0.0035	0.7638 ± 0.0054	0.7607 ± 0.0011
	Ensemble	0.7745 ± 0.0013	0.7777 ± 0.0019	0.7740 ± 0.0019
ogbn-products	MLP	0.6841 ± 0.0054	0.7277 ± 0.0054	0.7163 ± 0.0172
	GCN	0.7367 ± 0.0068	0.7679 ± 0.0109	0.7729 ± 0.0033
	SAGE	0.7543 ± 0.0065	0.7795 ± 0.0012	0.7811 ± 0.0049
	RevGAT	0.8016 ± 0.0078	0.8083 ± 0.0051	0.8000 ± 0.0078
	Ensemble	0.7991 ± 0.0034	0.8140 ± 0.0033	0.8090 ± 0.0037
tape-arxiv23	MLP	0.7803 ± 0.0014	0.7940 ± 0.0022	0.7948 ± 0.0025
	GCN	0.7518 ± 0.0044	0.7678 ± 0.0024	0.7703 ± 0.0025
	SAGE	0.7702 ± 0.0022	0.7894 ± 0.0024	0.7917 ± 0.0021
	RevGAT	0.7880 ± 0.0047	0.7880 ± 0.0023	0.7906 ± 0.0034
	Ensemble	0.8013 ± 0.0017	0.8029 ± 0.0020	0.8054 ± 0.0025

Table 2: Node classification accuracy for the Cora, PubMed, ogbn-arxiv, ogbn-products, and tape-arxiv23 datasets, demonstrating the effect of varying an instruction to bias the embeddings from the pre-trained LLM. The experiment is run over four seeds, with mean accuracy and standard deviation shown. The best results are coloured green (first), yellow (second), and orange (third). For all experiments, we use SFR-Embedding-Mistral as the embedding model on TA features only, and the simple task instruction to bias the embeddings.

explanations of ranked predictions to enrich node features.

- $\mathbf{h}_{\text{STAGE}}$: Reflects the model's performance training with node features generated by a pre-trained LLM.

Instruction-biased Embeddings Textual attributes for each node are passed to the embedding LLM together with a task description which remains constant for every text, prefixing each input with a task-specific system prompt. We evaluated 3 simple task descriptions:

1. A short prompt describing the classification task for the text, as used during the pre-training stage of the LLM.

2. A description of the types of relationships between texts to form a graph, along with the classification task description. Specific graph structure for each node is not included in the prompt, unlike the proposed method from (Fatemi et al., 2024).

3. No task description.

Our findings are summarized in Table 2. Further details of the instructions can be found in appendix Table 8.

Parameter-efficient Finetuning In Table 3 we investigate the effect of using parameter-efficient finetuning (PEFT) on the pre-trained LLM, as described in (Duan et al., 2023a). We also compare this against finetuning both the LLM (using PEFT) and the GNN in unison.

Embedding Model Type In Table 4, we compare the results when using different pre-trained LLMs as the text encoder.

Diffusion GNNs Included in Table 4, we study the performance of using SimpleGCN and SIGN models individually. Model selection and implementation details can be found in the appendix sections C and D.

Ablation Study To study the impact of each component in the GNN ensemble, we perform a detailed ablation study. The results can be found in 6.

Dataset	LLM + GNN Ensemble	$\text{LLM}_{\text{finetuned}}$	$\text{LLM}_{\text{finetuned}}$ + GNN Ensemble
Cora	0.8824 ± 0.0155	0.8063	**0.8856**
PubMed	0.9265 ± 0.0068	0.9513	**0.9559**
ogbn-arxiv	0.7777 ± 0.0019	0.7666	**0.7813**
ogbn-products	0.8140 ± 0.0033	0.8020	**0.8257**
tape-arxiv23	0.8029 ± 0.0020	0.8021	**0.8095**

Table 3: Effect of using parameter-efficient finetuning (PEFT) on the pre-trained LLM, as described in (Duan et al., 2023a). Comparison of GNN-only trained, LLM finetuned without GNNs, and LLM and GNN trained separately. The best results are highlighted in bold.

Dataset	Method	SFR-Embedding-Mistral	LLM2Vec	gte-Qwen1.5-7B-instruct
Cora	MLP	0.7680 ± 0.0228	0.8026 ± 0.0141	0.7389 ± 0.0136
	GCN	0.8704 ± 0.0105	0.8778 ± 0.0046	0.8621 ± 0.0105
	SAGE	0.8722 ± 0.0063	0.8773 ± 0.0062	0.8658 ± 0.0049
	RevGAT	0.8639 ± 0.0129	0.8810 ± 0.0033	0.8408 ± 0.0076
	Ensemble	0.8824 ± 0.0155	0.8898 ± 0.0066	0.8686 ± 0.0024
	Simple-GCN	0.7389 ± 0.0120	0.6983 ± 0.0120	0.7491± 0.0166
	SIGN	0.8819 ± 0.0074	0.8856 ± 0.0083	0.8575 ± 0.0157
PubMed	MLP	0.9142 ± 0.0122	0.9321 ± 0.0013	0.8808 ± 0.0107
	GCN	0.8960 ± 0.0042	0.8996 ± 0.0011	0.8591 ± 0.0041
	SAGE	0.9087 ± 0.0064	0.9231 ± 0.0056	0.8733 ± 0.0051
	RevGAT	0.8654 ± 0.0952	0.9312 ± 0.0026	0.8754 ± 0.0010
	Ensemble	0.9265 ± 0.0068	0.9357 ± 0.0031	0.8941 ± 0.0041
	Simple-GCN	0.7505 ± 0.0048	0.7400 ± 0.0037	0.7472 ± 0.0076
	SIGN	0.8868 ± 0.0062	0.9004 ± 0.0038	0.8611 ± 0.0084
ogbn-arxiv	MLP	0.7517 ± 0.0011	0.7331 ± 0.0033	0.7603 ± 0.0011
	GCN	0.7377 ± 0.0010	0.7324 ± 0.0014	0.7369 ± 0.0022
	SAGE	0.7596 ± 0.0040	0.7428 ± 0.0039	0.7664 ± 0.0029
	RevGAT	0.7638 ± 0.0054	0.7529 ± 0.0044	0.7738 ± 0.0009
	Ensemble	0.7777 ± 0.0019	0.7701 ± 0.0018	0.7817 ± 0.0011
	Simple-GCN	0.3337 ± 0.0107	0.3614 ± 0.0039	0.3463 ± 0.0181
	SIGN	0.6150 ± 0.0182	0.6035 ± 0.0084	0.6285 ± 0.0114
ogbn-products	MLP	0.7277 ± 0.0054	0.6913 ± 0.0052	0.7231 ± 0.0050
	GCN	0.7679 ± 0.0109	0.7479 ± 0.0128	0.7701 ± 0.0117
	SAGE	0.7795 ± 0.0012	0.7496 ± 0.0163	0.7921 ± 0.0069
	RevGAT	0.8083 ± 0.0051	0.7883 ± 0.0014	0.7955 ± 0.0096
	Ensemble	0.8140 ± 0.0033	0.7908 ± 0.0045	0.8104 ± 0.0041
	Simple-GCN	0.6216 ± 0.0052	0.6040 ± 0.0039	0.6219 ± 0.0039
	SIGN	0.6668 ± 0.0078	0.6621 ± 0.0009	0.6698 ± 0.0010
tape-arxiv23	MLP	0.7940 ± 0.0022	0.7772 ± 0.0033	0.8008 ± 0.0018
	GCN	0.7678 ± 0.0024	0.7541 ± 0.0042	0.7746 ± 0.0025
	SAGE	0.7894 ± 0.0024	0.7677 ± 0.0018	0.7975 ± 0.0016
	RevGAT	0.7880 ± 0.0023	0.7840 ± 0.0058	0.7954 ± 0.0028
	Ensemble	0.8029 ± 0.0020	0.7967 ± 0.0037	0.8065 ± 0.0022
	Simple-GCN	0.2516 ± 0.0027	0.2451 ± 0.0004	0.258 ± 0.0011
	SIGN	0.7186 ± 0.0041	0.6804 ± 0.0041	0.733 ± 0.0009

Table 4: Node classification accuracy for the Cora, PubMed, ogbn-arxiv, ogbn-products, and tape-arxiv23 datasets, demonstrating the effect of changing the pre-trained LLM text encoder. The experiment is run over four seeds, with mean accuracy and standard deviation shown. The best results are coloured green (first), yellow (second), and orange (third). For all experiments, we use TA features only, and the simple task instruction to bias the embeddings.

5 Analysis

Main Results (Table 1) We find that ensembling GNNs always leads to superior performance across datasets when taking the STAGE approach.

Despite the reduced computational resources and training data requirements, the STAGE method remains highly competitive across all benchmarks. The ensemble STAGE approach lags behind the TAPE pipeline by roughly 5% on Cora, 3.5% on Pubmed, 0.8% on ogbn-products, and 4% on tape-

arxiv23. This is a strong result when we consider that STAGE involves training only the GNN ensemble, whereas TAPE also requires two finetuned LMs to generate node features. We see marginally superior results on the ogbn-arxiv dataset using the ensemble STAGE approach.

Instruction-biased Embedding Results (Table 2) From our findings we conclude that varying the instructions to bias embeddings has little effect on downstream node classification performance for the models we evaluated. We note that while the authors of all embedding models recommend providing instructions along with input text in order to avoid degrading performance, we did not measure a performance improvement in our experiments.

This experiment further supports our claim that an ensemble approach improves robustness across datasets and methods of node feature generation.

PEFT Results (Table 3) Finetuning each LLM gave marginal performance improvements across all datasets to varying degrees; we see the largest improvement on pubmed (3%). It is of note that finetuning significantly increases the number of trainable parameters (see Table 5) and total training time. Specifically, PEFT for 7B embedding models has over 20 million trainable parameters. On a single A100 GPU, training runs lasted 6 hours on `ogbn-arxiv`.

LLM Embedding Model Comparison (Table 4) All three LLM embedding models demonstrated comparable performance on the graph tasks, with each model exhibiting marginally better results on different datasets. Notably, there was no clear winner among them. The LLM2Vec model exhibited slightly weaker performance on the larger datasets (ogbn-arxiv, ogbn-products, tape-arxiv23), while it was marginally stronger on the smaller datasets (Cora, PubMed).

Ensembling the GNN models consistently ranked among the top three models across all three LLM embedding models, delivering an average performance increase of 1%. Among the individual GNN architectures, RevGAT consistently demonstrated superior performance.

Diffusion-pattern GNN Results (Table 4) The diffusion-based GNNs yielded variable results across datasets. Specifically, SIGN emerged as the second-best performer on the Cora dataset. As expected, SIGN consistently outperformed SimpleGCN, given that it generalizes the latter. Due to its low training time, SIGN is a viable candidate for large datasets, although careful tuning of its hyper-parameters is recommended for optimal performance.

Ablation Study Results (Table 6) From our ablation study we observe that no individual GNN model outperforms any ensemble of models on any dataset. Additionally, we find that the full ensemble of MLP, GCN, SAGE and RevGAT achieve the highest and most stable accuracy scores across datasets.

Scalability An important advantage of STAGE is the lack of finetuning necessary to achieve strong results. This lies in contrast to approaches such as TAPE (He et al., 2024) and SimTeG (Duan et al., 2023a), both of which require finetuning at least one LM. Training an ensemble of GNNs and MLP head over the ogbn-arxiv dataset can be performed on a single consumer-grade GPU in less than 5 minutes. This is illustrated in Figure 2 where we compare the relationship between training time and accuracy for a number of SoTA node classification approaches. When using SIGN diffusion, training time was under 12 seconds for the ogbn-arxiv, but this came at a performance cost. Moreover, TAPE relies on text-level enhancement via LLM API calls, which adds a new dimension of cost and rate-limiting[3] to consider when adapting to other datasets.

6 Conclusions

This work introduces STAGE, a method to use pretrained LLMs as text encoders in TAG tasks without the need for finetuning, significantly reducing computational resources and training time. Additional gains can be achieved through parameter-efficient finetuning of the LLM. Data augmentation, which is orthogonal to our approach, could improve performance with general-purpose text embedding models. However, it likely remains intractable for many large-scale datasets due to the need to query a large model for each node.

We also demonstrate the effect of diffusion operators (Frasca et al., 2020) on node classification performance, decreasing TAG pipeline training time substantially. We aim to examine the scalability of diffusion-pattern GNNs on larger datasets in later work.

[3] https://platform.openai.com/docs/guides/rate-limits

Future work may aim to refine the integration of LLM encoders with GNN heads. Potential strategies include an Expectation-Maximization approach or a joint model configuration (Zhao et al., 2023). A significant challenge is the requirement for large, variable batch sizes during LLM fine-tuning due to current neighborhood sampling techniques, which necessitates increased computational power. We anticipate that overcoming these limitations will make future research more accessible and expedite iterations.

References

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *Preprint*, arXiv:2404.05961.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. 2024. Exploring the potential of large language models (llms) in learning on graphs. *Preprint*, arXiv:2307.03393.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '19. ACM.

Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S Dhillon. 2022. Node feature extraction by self-supervised multi-scale neighborhood prediction. *Preprint*, arXiv:2111.00064.

Timothy Davis. 2019. Algorithm 1000: Suitesparse:graphblas: Graph algorithms in the language of sparse linear algebra. *ACM Transactions on Mathematical Software*, 45:1–25.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

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

Keyu Duan, Zirui Liu, Peihao Wang, Wenqing Zheng, Kaixiong Zhou, Tianlong Chen, Xia Hu, and Zhangyang Wang. 2023b. A comprehensive study on large-scale graph training: Benchmarking and rethinking. *Preprint*, arXiv:2210.07494.

Lisa Ehrlinger and Wolfram Wöß. 2016. Towards a definition of knowledge graphs. In *International Conference on Semantic Systems*.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2024. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations*.

Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. 2020. Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2022. Predict then propagate: Graph neural networks meet personalized pagerank. *Preprint*, arXiv:1810.05997.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2018. Inductive representation learning on large graphs. *Preprint*, arXiv:1706.02216.

Taher H. Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, page 517–526, New York, NY, USA. Association for Computing Machinery.

Simon Haykin. 1994. *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. *Preprint*, arXiv:2006.03654.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2024. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. *Preprint*, arXiv:2305.19523.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021a. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2021b. Open graph benchmark: Datasets for machine learning on graphs. *Preprint*, arXiv:2005.00687.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *Preprint*, arXiv:1609.02907.

Constant Joseph Koné, Michel Babri, and Jean Marie Rodrigues. 2023. Snomed ct: A clinical terminology but also a formal ontology. *Journal of Biosciences and Medicines*.

Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. 2022a. Training graph neural networks with 1000 layers. *Preprint*, arXiv:2106.07476.

Rui Li, Jianan Zhao, Chaozhuo Li, Di He, Yiqi Wang, Yuming Liu, Hao Sun, Senzhang Wang, Weiwei Deng, Yanming Shen, Xing Xie, and

Qi Zhang. 2022b. House: Knowledge graph embedding with householder parameterization. *Preprint*, arXiv:2202.07919.

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

Juncheng Liu, Bryan Hooi, Kenji Kawaguchi, Yiwei Wang, Chaosheng Dong, and Xiaokui Xiao. 2024. Scalable and effective implicit graph neural networks on large graphs. In *The Twelfth International Conference on Learning Representations*.

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

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented language models: a survey. *Preprint*, arXiv:2302.07842.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Preprint*, arXiv:1310.4546.

George A. Miller. 1995. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41.

Niklas Muennighoff. 2022. Sgpt: Gpt sentence embeddings for semantic search. *Preprint*, arXiv:2202.08904.

OpenAI. 2023. Introducing chatgpt. Accessed: 2023-05-20.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, and Janko Altenschmidt. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, page 1–20.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

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

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR.

Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, et al. 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. 2023. Graphformers: Gnn-nested transformers for representation learning on textual graph. *Preprint*, arXiv:2105.02605.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2024. Language is all a graph needs. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1955–1973, St. Julian's, Malta. Association for Computational Linguistics.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2023. Learning on large-scale text-attributed graphs via variational inference. *Preprint*, arXiv:2210.14709.

A Appendix

B Negative Results

Co-training LLM and GNN: In a similar approach to iterative methods, we investigated co-training the LLM and GNN on the `ogbn-arxiv` node classification task to facilitate a shared representation space. This proved unfeasible due to the memory requirements exceeding the capacity of one A100 GPU.

C Implementation of Diffusion Operators

We implement diffusion operators from two methods, Simple-GCN (Wu et al., 2019) and SIGN (Frasca et al., 2020). In the case of SIGN, the authors omit implementation details of the operators, so we include them here.

Let A denote the adjacency matrix of a possibly directed graph G, X its node features, and D the diagonal degree matrix of G.

We denote the *random-walk normalized* adjacency $A_{\text{RW}}\& := AD^{-1}$ and the *GCN-normalized* adjacency (Kipf and Welling, 2017)

$$A_{\text{GCN}} := (D+I)^{-1/2}(A+I)(D+I)^{-1/2} \quad (4)$$

The *Personalized PageRank* matrix is then given by (Gasteiger et al., 2022):

$$A_{\text{PPR}} := \alpha\left(I_n - (1-\alpha)A_{\text{RW}}\right)^{-1} \quad (5)$$

And we denote the *triangle-based* adjacency matrix by $\mathbf{A}_\Delta$, where $(A_\Delta)_{ij}$ counts the number of directed triangles in G that contain the edge (i,j)

Diffusion is applied to node features X by matrix multiplication. Simple-GCN takes a power k of A_{GCN} as its diffusion operator, whilst SIGN diffusion generalizes this to concatenate powers of A_{GCN}, A_{PPR} and A_Δ.

Diffusion can be calculated efficiently if sparse-matrix-sparse-matrix multiplication is avoided. For both SIGN and Simple-GCN, the order of operations for applying a power of an operator A_{op} should be

$$\underbrace{A_{\text{op}}(A_{\text{op}}(...(A_{\text{op}}(X))...)}_{k \text{ times}} \quad (6)$$

as opposed to $(A_{\text{op}}^k)X$, where the operator matrix A_{op} is feasible to calculate, since the former avoids sparse matrix multiplication. In SIGN, the recursive nature of eq.6 can be exploited to reuse results for calculating successive powers.

In the case of personalized pagerank diffusion, we first use a trick from (Gasteiger et al., 2022) to approximate the diffused features of personalized pagerank matrix $A_{\text{PPR}}X$ in linear time and avoid calculative A_{PPR} directly, by viewing eq.5 as *topic-sensitive* PageRank (Haveliwala, 2002). We use the random-walk normalized adjacency matrix.

The following power iteration approximates $A_{\text{PPR}}X$ (notation from (Gasteiger et al., 2022)):

$$Z^{(0)} := X$$
$$Z^{(k+1)} := (1-\alpha)AZ^{(k)} + \alpha X$$

To compute the nth diffused power, we repeat the process n times:

$$Z_0^{(0)} = X$$
$$Z_{n+1}^{(0)} = \lim_{k\to\text{inf}} Z_n^{(k)}$$

Lastly, for triangle-based diffusion, we count triangles using linear algebra. For unweighted A we perform a single sparse matrix multiplication to obtain A^2, in which element (i,j) counts the directed paths in G for node i to node j. We then calculate

$$A_\Delta = A^T \odot A^2$$

where $\odot$ denotes the Hadamard product, which can be efficiently calculated for sparse matrices. We then normalize and diffuse features over powers of A_Δ in the same fashion as for A_{GCN}.

An implementation of these operators as GraphBLAS (Davis, 2019) code is published alongside this paper.

C.1 Parallelism of diffusion operators

All operations above can be be parallelized across columns of X, either keeping A in shared memory on one machine or keeping a copy on each executor in a distributed computing infrastructure like Apache Spark.

D Preprocessing & Model Selection for Diffusion Operators

For Simple-GCN (Wu et al., 2019), we set the degree k by selecting the highest validation accuracy from $k = 2, 3, 4$, of which $k = 2$ had the highest accuracy in each case. For SIGN (Frasca et al., 2020), we choose s, p, t from the highest validation accuracy amongst $(3,0,0)$ $(3,0,1)$ $(3,3,0)$, $(4,2,1)$ $(5,3,0)$. For **Cora** and **PubMed**, $(4,2,1)$ was chosen, and for **ogbn-arxiv**, **ogbn-products**, and **tape-arxiv23** $(3,3,0)$ was chosen. We chose the number of layers for the Inception NLP to match the number of layers in other GNNs tested, 4. We did not perform additional hyper-parameter tuning. When preprocessing the embeddings, we centered and scaled the data to unit variance for Simple-GCN and SIGN only.

E Model Trainable Parameters

Model	Trainable Parameter Count
RevGAT	3,457,678
GCN	559,111
SAGE	1,117,063
MLP	117,767
Simple-GCN	24,111
SIGN-(3,3,0)	500,271
SIGN-(4,2,1)	582,575
PEFT 7B LLM	>20M

Table 5: Trainable parameter counts for different models. 7B LLM refers to all finetuned LLM embedding models used during experiments (see Section 3.1)

F Ablation Study

To study the effect each model has on the GNN ensemble step of STAGE, we perform a detailed ablation study. The results are shown in Table 6.

G Datasets

In this section, we describe the characteristics of the node classification datasets we used during our work. The statistics are shown in Table 7.

H Instruction-biased Embeddings

In Table 8 we list the specific instructions used to 655 investigate the effect of biasing embeddings.

Method	Cora	PubMed	ogbn-arxiv	ogbn-products	tape-arxiv23
Full Ensemble	0.8824 ± 0.0155	0.9265 ± 0.0068	0.7777 ± 0.0019	0.8140 ± 0.0033	0.8029 ± 0.0020
No MLP	0.8838 ± 0.0039	0.9239 ± 0.0036	0.7748 ± 0.0012	0.8093 ± 0.0021	0.8015 ± 0.0010
No GCN	0.8685 ± 0.0209	0.9240 ± 0.0076	0.7731 ± 0.0017	0.8100 ± 0.0038	0.8028 ± 0.0023
No SAGE	0.8759 ± 0.0207	0.9258 ± 0.0110	0.7739 ± 0.0020	0.8116 ± 0.0045	0.8021 ± 0.0035
No RevGAT	0.8764 ± 0.0180	0.9272 ± 0.0052	0.7717 ± 0.0007	0.8029 ± 0.0036	0.7985 ± 0.0018
Best Individual	0.8722 ± 0.0063	0.9142 ± 0.0122	0.7638 ± 0.0054	0.8083 ± 0.0051	0.7880 ± 0.0023
Best Individual Model	**SAGE**	**MLP**	**RevGAT**	**RevGAT**	**RevGAT**

Table 6: Ablation study results for the ensemble model on various datasets. The table shows the accuracy when each component is removed from the ensemble. The experiment is run over four seeds, with mean accuracy and standard deviation shown. The best results are coloured green (first), yellow (second), and orange (third). For all experiments, we use SFR-Embedding-Mistral as the embedding model on TA features only, and the simple task instruction to bias the embeddings.

Dataset	Node Count	Edge Count	Task	Metric
Cora (McCallum et al., 2000)	2,708	5,429	7-class classif.	Accuracy
Pubmed (Sen et al., 2008)	19,717	44,338	3-class classif.	Accuracy
ogbn-arxiv (Hu et al., 2021b)	169,343	1,166,243	40-class classif.	Accuracy
ogbn-products (Hu et al., 2021b) (subset)	54,025	74,420	47-class classif.	Accuracy
tape-arxiv23 (He et al., 2024)	46,198	78,548	40-class classif.	Accuracy

Table 7: Statistics of the TAG datasets

Dataset	Prompt Type	Prompt
ogbn-arxiv, arxiv_2023, cora, pubmed	Simple Task	Identify the main and secondary category of Arxiv papers based on the titles and abstracts.
ogbn-arxiv, arxiv_2023, cora, pubmed	Graph-Aware	Identify the main and secondary category of Arxiv papers based on the titles and abstracts. Your predictions will be used in a downstream graph-based prediction that for each paper can learn from your predictions of neighboring papers in a graph as well as the predictions for the paper in question. Papers in the graph are connected if one cites the other.
ogbn-products	Simple Task	Identify the main and secondary category of this product based on the titles and description.
ogbn-products	Graph-Aware	Identify the main and secondary category of this product based on the titles and description. Your predictions will be used in a downstream graph-based prediction that for each product can learn from your predictions of neighboring products in a graph as well as the predictions for the paper in question. Products in the graph are connected if they are purchased together.

Table 8: Task descriptions for embedding bias across various datasets.