A Pure Transformer Pretraining Framework on Text-attributed Graphs

Yu Song
Michigan State University
songyu5@msu.edu

Haitao Mao
Michigan State University
haitaoma@msu.edu

Jiachen Xiao
Michigan State University
xiaojiac@msu.edu

Jingzhe Liu
Michigan State University
liujin33@msu.edu

Zhikai Chen
Michigan State University
chenzh85@msu.edu

Wei Jin
Emory University
wei.jin@emory.edu

Carl Yang
Emory University
j.carlyang@emory.edu

Jiliang Tang
Michigan State University
tangjili@msu.edu

Hui Liu
Michigan State University
liuhui7@msu.edu

Abstract

Pretraining plays a pivotal role in acquiring generalized knowledge from large-scale data, achieving remarkable successes as evidenced by large models in CV and NLP. However, progress in the graph domain remains limited due to fundamental challenges represented by feature heterogeneity and structural heterogeneity. Recent efforts have been made to address feature heterogeneity via Large Language Models (LLMs) on text-attributed graphs (TAGs) by generating fixed-length text representations as node features. These high-quality features reduce the previously critical role of graph structure, resulting in a modest performance gap between Graph Neural Networks (GNNs) and structure-agnostic Multi-Layer Perceptrons (MLPs). Motivated by this, we introduce a feature-centric pretraining perspective by treating graph structure as a prior and leveraging the rich, unified feature space to learn refined interaction patterns that generalizes across graphs. Our framework, Graph Sequence Pretraining with Transformer (GSPT), samples node contexts through random walk and employs masked feature reconstruction to capture pairwise proximity in the LLM-unified feature space using a standard Transformer. By utilizing unified text representations rather than varying structures, GSPT alleviates structural heterogeneity and achieves significantly better transferability among graphs within the same domain. Our approach can be easily adapted to both node classification and link prediction, demonstrating promising empirical success on various datasets. The source code is publicly available at https://github.com/SongYYYY/GSPT.

1 Introduction

Transfer learning has witnessed remarkable success in recent years, particularly exemplified by the advancements in foundation models for Natural Language Processing (NLP) [1, 2] and Computer Vision (CV) [3, 4]. These methods typically leverage self-supervised pretraining on large-scale datasets to acquire broad, generalized knowledge, which is subsequently adapted to specific tasks and datasets through fine-tuning or in-context learning. However, the graph domain predominantly adheres to a 'one-model, one-dataset' approach, where models are tailored specifically to individual datasets and tasks, deviating from the prevailing trend of 'one model serves all'.

The pursuit of cross-dataset transfer learning on graphs faces unique challenges. The immediate one is *feature heterogeneity*, i.e., the inherent mismatch in feature spaces among different datasets, as graphs denoting different data types often have features with varying dimensions and semantics [5, 6]. Previous methods circumvent this by ignoring the features and only transferring knowledge from the structural side [7, 8], or constraining the applications in vertical domains where node/edge features are naturally aligned [9, 10]. Such approaches, while somewhat effective, suffer from performance loss or restricted applicability [11]. Another challenge is *structural heterogeneity*, which arises from the vastly different structural patterns across various graphs, leading to out-of-distribution scenarios and potential negative transfer [12]. A typical example is the varying degrees of homophily across graphs [13]. Together, these challenges pose substantial obstacles to the development of a flexible, generalizable model capable of effective pretraining and knowledge transfer across diverse downstream datasets.

Recent efforts [5, 14–17] have been made to tackle feature heterogeneity by leveraging large language models (LLMs) to unify the feature spaces of text-attributed graphs (TAGs). They replace traditional shallow features like word2vec and tf-idf with language model-enhanced features and have demonstrated impressive empirical success, represented by the improved performance on graph-related tasks and the reduced gap between purely feature-based approaches (e.g., an MLP) and graph-tailored models (e.g., Graph Neural Networks) [14]. This trend motivates us to consider a conceptual shift from a structure-centric approach to a feature-centric view, suggesting the potential to enhance knowledge transfer by effectively leveraging the LLM-unified feature space.

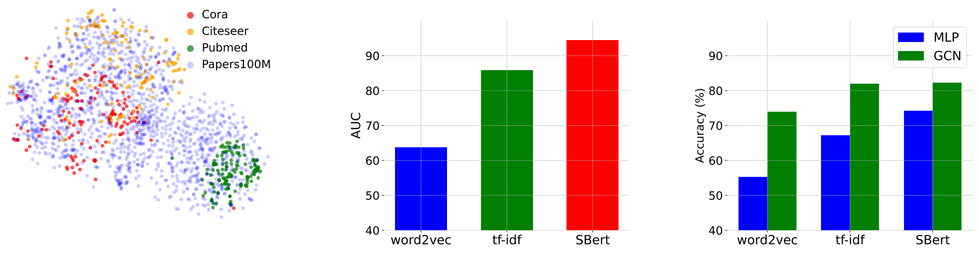
Building on this perspective, the core principle is to identify a fundamental unit that can effectively encode graph information and generalize across the LLM-unified feature space. In this study, we propose to learn a pairwise function that models the proximity of node pairs, thereby capturing the interactions embedded in the graph structure. Drawing inspiration from existing works [18–20], we adopt a feature reconstruction-based objective as the pretext task, under the assumption that the relational patterns learned through reconstruction overlap with those necessary for downstream tasks [21]. Specifically, given a (masked) center node and its context, we aim to reconstruct the feature of the center node using its context. This approach presents two key challenges. The first challenge is *how to construct an appropriate context for graphs*? Unlike sequential data such as language or grid-based data like images, graphs represent non-Euclidean structures, making it non-trivial to define the context for reconstruction. The second challenge is *how to reconstruct the center node from its context*? This involves identifying an appropriate architecture capable of accurately interpreting the context nodes and a loss function that effectively guides the model's learning process.

To address the first challenge, we propose using node sequences generated by random walks as contexts, where the entire sequence forms the receptive field, and the order of nodes preserves proximity information [22, 23]. By doing so, the original graph topology serves as a prior to retrieve relevant context nodes, facilitating effective proximity modeling for accurate reconstruction of masked features. For the second challenge, we employ a standard Transformer architecture due to its flexibility in modeling sequences with self-attention and its proven transferability across domains like CV and NLP [3, 19]. To handle the multi-dimensional and continuous nature of LM-produced features, we adopt a cosine similarity-based objective to measure reconstruction error instead of cross-entropy [24].

Putting it all together, we propose **G**raph **S**equence **P**retraining with **T**ransformer (GSPT), where a standard Transformer is used alongside a feature reconstruction objective to learn a unified model for node representations. Our framework highlights the use of text-based representations rather than the diverse structures of different graphs, leading to enhanced transferability and reduced risk of negative transfer caused by structural shifts. To evaluate the effectiveness of our method, we perform self-supervised training on the largest graph available ogbn-papers100M [25], and apply the pretrained model on various downstream datasets. Experimental results reveal that our GSPT excels in in-context node classification and link prediction, showcasing effective knowledge transfer from pretraining to downstream tasks. Furthermore, we observe a notable trend of improvement with increased pretraining data, highlighting the potential of the proposed framework. Our findings deepen the understanding of the LLM-unified feature space in graph data and provide valuable insights into the development of a versatile and generalizable graph foundation model.

2 Preliminary Study

In this section, we aim to answer the question: *What can be better transferred across graphs?* In general, a graph $G = (A, X)$ is comprised of two "modalities", i.e., the structure space represented



(a) t-SNE visualization of feature space **(b)** Link prediction **(c)** Node classification

Figure 1: (a) SentenceBert provides a unified feature space for different datasets under the same domain. The node features of three small citation graphs, i.e., Cora, Citeseer, and Pubmed, can be well covered by ogbn-papers100M, a large-scale citation network containing papers from a vast variety of research topics. (b) Advanced text embeddings are better at predicting the missing edges compared with shallow features. (c) Advanced embeddings reduce the performance gap on node classification between GCN and MLP. Experiments are conducted on Cora.

by the adjacency matrix A, and the feature space of X. Previous studies [13, 26] show that the structure space can hardly be unified since the connections may be formed due to vastly different principles even for graphs in similar domains. For instance, in a friendship network, nodes tend to form edges with others of the same gender, whereas the pattern is reversed in a dating network. Such inherent shift poses significant challenges to the transfer learning on the structure space, even leading to negative transfer [12, 27]. On the other hand, the feature space X can be effectively unified with a powerful LM, as demonstrated in Figure 1 (a), where a pretrained SentenceBERT [28] is used to generate node embeddings for different datasets within the citation domain.

Therefore, we ask: can we focus on transferring knowledge from the feature space, while reconstructing the structure information based on the unified features? To answer this question, we conduct two sets of experiments to determine the extent to which structure can be reconstructed from the feature information. Based on Figure 1, the SentenceBert embeddings significantly facilitate the structure reconstruction, demonstrated by the improved link prediction performance (b) and the reduced gap between GCN and MLP in node classification (c).

The aforementioned observations motivate us to design our pretraining framework from a feature-centric perspective: we consider graph structure as a prior and utilize features X to capture the fine-grained topology that generalizes across graphs. In particular, we propose to model the graph structure via pairwise relationships based on the unified feature space derived from LLM embeddings and transfer this function to datasets within similar domains. The pairwise relationship is useful because it is closely related to various graph-based tasks, e.g., for node classification, we can frame the problem as "whether two nodes belong to the same class"; for link prediction, the question can be framed as "whether there exists an edge between a pair of nodes". Compared with strictly adhering to the fixed inductive bias of graph topology, this approach inherently avoids the potential negative transfer due to structure mismatch and has the potential to improve with advances in LLMs to unify the node attributes.

3 Method

3.1 An Overview

In this section, we introduce our feature-centric pretraining framework Graph Sequence Pretraining with Transformer (GSPT), a direct implication of the preliminary study. It consists of two major components. First, we generate *node contexts* from the graph through random walks. This context encapsulates the structural information of the graph and is tailored to the specific task. Next, we feed the context into a standard Transformer and perform masked feature reconstruction on the large-scale pretraining dataset. This approach enables the Transformer to effectively model pairwise relationships within a unified feature space, facilitating seamless transfer to downstream datasets. Next we will illustrate our framework using the node classification task and then extend it to link prediction in Appendix A.1. The overall framework of GSPT is illustrated in Figure 2.

3.2 Context Construction

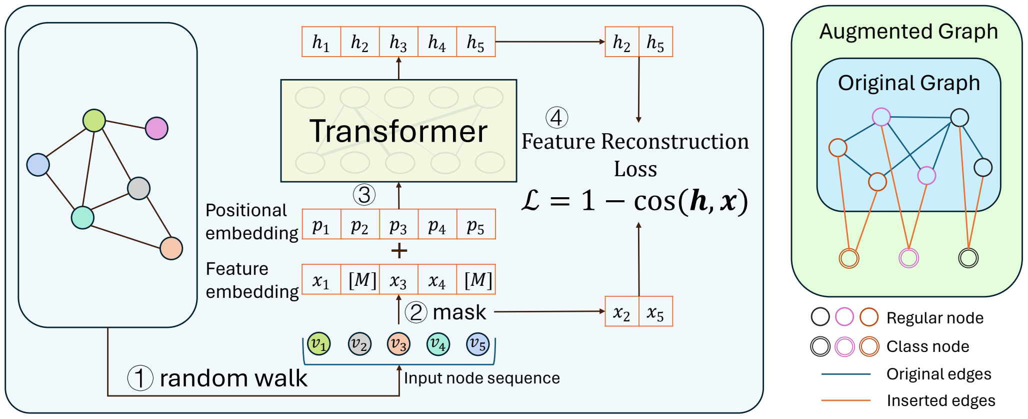

h_1 h_2 h_3 h_4 h_5
h_2 h_5
Transformer
④ Feature Reconstruction Loss
$\mathcal{L} = 1 - \cos(\boldsymbol{h}, \boldsymbol{x})$
③
Positional embedding
p_1 p_2 p_3 p_4 p_5
+
Feature embedding
x_1 $[M]$ x_3 x_4 $[M]$
② mask
x_2 x_5
v_1 v_2 v_3 v_4 v_5
① random walk
Input node sequence
Augmented Graph
Original Graph
Regular node
Class node
Original edges
Inserted edges

Figure 2: The overall framework of Graph Sequence Pretraining with Transformer (GSPT). Left: the pretraining consists of four steps: (1) generate node sequences from the graph using random walk; (2) randomly replace a portion of node features with [MASK]; (3) feed the input sequence into the Transformer and (4) compute the feature reconstruction loss with cosine similarity. Right: We construct the augmented graph by adding class nodes to the original graph and connecting correponding node pairs. GSPT performs in-context node classification by comparing the cosine similarity between the representations of regular nodes and class nodes.

Graph context aims to provide structural information that graph algorithms can leverage. Traditional methods like Graph Neural Networks (GNNs) rely on message-passing to generate contextualized node representations, which have been effective in many graph-related tasks [29–31]. However, these approaches often encode fixed inductive biases via K-hop neighborhoods, which can lead to negative transfer when applied to structurally diverse graphs [12, 13]. Moreover, extracting the ego networks poses a significant computational burden, particularly for large-scale graphs with high average degrees. In contrast, random walks (RWs) offer a more flexible mechanism for exploring graph structures, supported by both theoretical rigor and empirical success [22, 32]. Next, we demonstrate that random walks provide several key advantages for context construction in our framework. The proof is provided in Appendix A.5.

[Neighbor Coverage] Let $G = (V, E)$ be a graph, $v \in V$ a node, and $N_K(v)$ its K-hop neighborhood. Given m random walks of length l starting from v, the probability $p(l, m)$ that all nodes in $N_K(v)$ are visited by at least one random walk is bounded by:

$$p(l, m) \geq 1 - |N_K(v)| \cdot \left(\frac{H_K}{l}\right)^m,$$

where $|N_K(v)|$ is the size of the K-hop neighborhood, and H_K is an upper bound on the expected hitting time between nodes in a K-hop neighborhood, which is typically small for a low value of K.

[Proximity Preservation] Let $H(v, u)$ be the expected hitting time from v to u in a graph G. For any two nodes $u_1, u_2 \in V$ such that $H(v, u_1) < H(v, u_2)$, there exists an m large enough such that:

$$P(f(v, u_1) < f(v, u_2)) > 1 - \delta,$$

for any $\delta > 0$, where $f(v, u)$ is the average position of u in m random walks starting from v.

Remark. Proposition 1 ensures that with multiple random walks of sufficient length, the K-hop neighborhood can be covered with high probability. In practice, initiating a small number of random walks per node provides enough context for generating high-quality node representations. Proposition

2 demonstrates that the order of nodes in random walks effectively reflects node proximity via the expected hitting time, which captures both local and global path information, serving as a robust indicator of node connectivity [33]. By incorporating learnable positional embeddings into the input node features, the QKV-attention is capable of capturing this fine-grained information.

In addition to their ability to cover neighborhoods and preserve node proximity, random walks offer several other benefits. For instance, the randomness of RWs acts as a form of data augmentation, potentially improving robustness and generalization. Unlike message-passing confined to fixed neighborhoods, random walks dynamically explore nodes up to K hops away, expanding the receptive field and benefiting the modeling of long-range dependencies. From the efficiency perspective, random walks can be easily parallelized, making them scalable for large graphs.

In our implementation, we adopt the random walk approach from Node2vec [22], which uses parameters p and q to balance between local and global graph exploration. For pretraining, we generate one random walk sequence starting from each node. At inference time, we start 3 random walks per node to enhance performance.

3.3 Transformer as Backbone

With random walks as the contexts, our next step is to identify a suitable backbone to effectively learn the interaction patterns within each context. As discussed earlier, GNNs may not be the best fit for this purpose due to their fixed graph inductive biases. While Graph Transformers aim to address these issues by integrating self-attention with fixed graph structures, they often become complex and are typically tailored for specific tasks such as graph classification [34].

In contrast, the Vanilla Transformer [35] stands out as an ideal architecture for our task due to several key advantages. Firstly, Transformers enable flexible relational modeling through self-attention, allowing for dynamic and adaptable feature propagation. Secondly, their straightforward design simplifies implementation and scaling compared to more intricate variants. Lastly, they integrate several successful components in deep learning such as layer normalization and residual connections, greatly reducing the design space of the framework.

The combined use of Transformers and random walks is mutually beneficial. Random walks provide contexts in the form of sequences to be processed by Transformers, while Transformers capture fine-grained interaction patterns given the sampled contexts. With this approach, the graph topology acts as a prior to provide structural knowledge for the model, effectively addressing the potential structural shifts caused by fixed inductive bias.

3.4 Masked Feature Reconstruction

The pretext task plays a crucial role in determining the type of knowledge the model will acquire and how well it can be transferred to downstream tasks. In our approach, we emphasize pairwise relationships between nodes, as many graph-related tasks can be framed in terms of such relations.

To this end, we aim to learn meaningful interaction patterns between nodes through self-supervised learning and transfer this knowledge to unseen datasets. Inspired by the success of masked autoencoders across various domains [18, 19], we adopt masked feature reconstruction for our pretraining task. The key assumption is that the relational patterns learned for feature reconstruction overlap with those required for tasks like node classification, thus offering a robust way of knowledge transfer between pretraining and downstream applications [21]. To handle the sequential nature of RWs, we adopt the BERT [19] approach, which enables feature reconstruction in a decoder-free manner. We retain the bidirectional attention in BERT [19] as it is favored in the random walk setting where context from both preceding and succeeding nodes enhances the prediction of missing features.

Specifically, given a random walk of length l: $rw = [v_0, v_1, ..., v_{l-1}]$, the input sequence is constructed by extracting the corresponding node features $x_i \in \mathbb{R}^d$, added by the positional embeddings $p_i \in \mathbb{R}^d$, i.e., $h^0 = [x_0 + p_0, x_1 + p_1, ..., x_{l-1} + p_{l-1}]$. Before feeding to the Transformer, we randomly replace a ratio of nodes in the sequence with a [MASK] token. The goal is to reconstruct the raw node feature of the masked nodes based on the output of Transformer. Unlike the original masked language modeling (MLM) task in BERT, where a cross-entropy loss is used to predict the ID of the masked tokens, we use cosine similarity to measure how well the feature is reconstructed from the context, similar to [24]. Formally, the node representations for a given input batch $H^0 \in \mathbb{R}^{b\times l\times d}$

of b nodes are obtained through the following steps:

$$H^L = \text{TRM}(H^0), \tag{1}$$

$$H_{node} = \text{Pooling}(H^L), \tag{2}$$

where TRM is a Transformer with L layers, and H^L is the output of the final layer. Pooling denotes a pooling function that reduces the output $H^L \in \mathbb{R}^{b\times l\times d}$ to node representations $H_{node} \in \mathbb{R}^{b\times d}$. In our implementation, the pooling function is realized by taking the average of all embeddings of v_i in H^L. For simplicity, we use h_i to denote the i-th row of pooled representations H_{node}.

Finally, given a masked sequence of nodes, the loss for the feature reconstruction task is computed as follows:

$$\mathcal{L}_{\text{node}} = \frac{1}{m}\sum_{i=1}^{m}\left(1 - \cos(h_i, x_i)\right), \tag{3}$$

where $\cos(h_i, x_i)$ denotes the cosine similarity between the reconstructed representation h_i and the original feature x_i, and m is the number of masked nodes in the sequence.

3.5 Negative sampling

A random walk sequence typically form a locally connected component where nodes are highly correlated, especially in graphs exhibiting strong homophily. Such correlation implies that nodes in proximal positions within RWs are likely to share similar features. In the context of masked feature reconstruction, this can lead to a shortcut where basic aggregation functions, like mean pooling, suffice to reconstruct node features with high accuracy [24].

To enhance the model's ability to discriminate between relevant and irrelevant contextual information, we introduce noisy nodes into the input sequence as "distractor nodes." The purpose of such nodes is to ensure that the model cannot obtain a good reconstruction error without learning to attend to the truly relevant nodes in the sequence. In practice, we randomly select a node from the graph, and repeat it K times at the end of the sequence, where K is a random number from $[0, l]$. Repeating the same distractor node multiple times strategically increases the task's complexity while maintaining a manageable noise level, preventing excessive interference with the learning objective. Formally, the random walk sequence with distractor nodes is given by:

$$rw = [v_0, v_1, ..., v_{l-K-1}, v_d, v_d, ..., v_d], \tag{4}$$

where v_d is the distractor node. Note that v_d and K are sampled independently for each sequence. By complicating the reconstruction task, the model is compelled to engage more effectively with its attention mechanisms, improving its generalization capabilities by encouraging a more discerning use of contextual information. Through ablation study, we demonstrate that the inclusion of distractor nodes significantly improves the quality of pretraining.

3.6 Enabling In-context Learning

After pretraining, our framework is able to produce high-quality node representations for input graphs from similar domains. To fully utilize the knowledge of the pretrained model, we design an in-context learning framework to address few-shot learning tasks using support examples, inspired by the graph prompts in [5, 17].

This in-context learning process is illustrated in the right part of Figure 2. For an N-way K-shot task, we first introduce N class nodes to the original graph G_{ori}. We then connect the K-shot examples to their corresponding class nodes, creating an augmented graph G_{aug}. Similar to the pretraining process, we transform the augmented graph into sequences using random walks. Our pretrained backbone then performs feature propagation on these node sequences, merging the features of regular nodes and class nodes through the self-attention mechanism. The output node embeddings, both for regular and class nodes, are used for prediction by comparing the cosine similarity between the test nodes and all class nodes. Given the output embeddings $\{s_1, s_2, \ldots, s_N\}$ for the N class nodes and the embedding t for a test node, the predicted class is:

$$\hat{y} = \text{argmax}_{i\in\{1,\ldots,N\}} \cos(t, s_i). \tag{5}$$

Through this approach, our framework is able to make predictions on any unseen test graph with novel labels, without modifying the parameters of the pretrained model.

4 Experiment

In this section, we conduct experiments to validate the effectiveness of our proposed GSPT framework. Through the experiments, we aim to answer the following research questions: **RQ1:** Can our proposed pretraining framework achieve cross-graph knowledge transfer? **RQ2:** What is the underlying mechanism that promotes the positive transfer of GSPT? **RQ3:** How do different negative sampling strategies affect performance? **RQ4:** Can our framework benefit from increasing the data scale used for pretraining?

4.1 Datasets

We evaluate our framework using a variety of citation datasets. For pretraining, we utilize the ognb-papers100M [36] dataset, which comprises over 100 million nodes and 1.6 billion edges, making it the largest publicly available graph dataset. To address efficiency issues, we use the METIS algorithm to partition the entire graph into approximately 10,000 smaller graphs during pre-processing, each containing around 10,000 nodes. During pretraining, we randomly select one partition to construct a batch. For downstream tasks, we assess the performance on both node classification and link prediction using four well-known citation graphs: Cora, Citeseer, Pubmed, and Arxiv23. We neglect ogbn-arxiv in our evaluation as it is covered by ogbn-papers100M. To ensure a consistent and unified feature space among all datasets, we generate text embeddings for the raw text associated with each node using SentenceBERT [28]. Due to space limit, the results for link prediction are presented in Appendix A.2.

4.2 Few-shot Node Classification

4.2.1 Experimental Setup

Setup. We begin by evaluating our pretraining framework on node classification. In line with [5], we adopt a few-shot in-context learning setting to create N-way K-shot tasks. Specifically, we randomly select N classes from the dataset's class set and generate a K-shot prompt, where K labeled nodes are randomly chosen for each of the N classes. All K-shot examples are drawn from the original training split of the graphs. The validation and test sets are constructed by filtering the nodes of the selected classes from the original validation and test sets, respectively. In all experiments, we generate 100 templates for each of the N-way K-shot tasks and report the averaged performance.

Baselines. We compare our method against five groups of baselines. The first group includes feature-based approaches, which perform node classification by comparing the cosine similarity between the test nodes and prototype vectors denoting the centroid of the K-shot prompt nodes. To make the features contextualized w.r.t. the graph structure, we adopt the degree-normalized Laplacian to perform feature propagation following [37]. The second group represents traditional supervised learning methods, including GCN [29] and GAT [38], which are directly trained on the downstream datasets. The third group consists of self-supervised learning baselines, covering three representative methods: DGI [39], GraphMAE [24] and GRACE [40]. To assess the transferability of these SSL methods, we pretrain a GNN model using each method on ogbn-papers100M [36] and then evaluate the pretrained models on downstream datasets. Lastly, we compare our GSPT against in-context learning methods, including Prodigy [17] and OneForAll [5]. Prodigy utilizes neighbor matching and supervised training on the MAG240M dataset, while OneForAll is trained on ogbn-arxiv with few-shot learning templates directly mimicking the downstream task.

Our method. Depending on how to construct the features for the class nodes, we propose two variants of GSPT, namely, GSPT-void and GSPT-desc. GSPT-void means initializing the features for all class nodes with zero vectors. In contrast, in GSPT-desc, we use GPT4 [1] to generate a text description for each class of the downstream dataset, and use the SentenceBert [28] embeddings of the descriptions to initialize the class nodes, following [5].

4.2.2 Result Comparison

Table 1 shows the performance comparison between the five groups of methods on N-way-3-shot tasks with varying Ns. We make the following observations:

Feature-based approaches provide a solid baseline. Feature-based methods offer a straightforward implementation of model-free classification and can be practical in low-resource scenarios. Among these methods, SentenceBERT outperforms other shallow embeddings, significantly reducing the gap to supervised learning models which are trained end-to-end on downstream datasets. This indicates that using advanced language model embeddings as node features, combined with simple message-passing, allows a model-free method to generate highly contextualized and discriminative node representations.

Table 1: Performance comparison of few-shot node classification on citation datasets. We report the Accuracy (%) on 100 sampled tasks with 3-shot prompts.

Methods	Cora			Citeseer		Pubmed	Arxiv23		
	2-way	5-way	7-way	3-way	6-way	3-way	3-way	5-way	10-way
Supervised Learning									
GCN	91.90	77.22	71.60	80.38	67.46	66.56	**88.34**	82.14	67.98
GAT	**93.00**	78.19	72.28	80.96	68.02	66.82	87.75	**83.80**	69.21
Feature-based Approach									
Word2vec	78.81	56.73	48.82	69.49	53.74	56.09	60.57	59.01	44.05
Tf-idf	87.96	70.98	64.27	76.40	62.78	63.76	79.74	75.77	60.52
SentenceBert	89.39	74.56	67.91	79.10	66.70	64.55	83.66	77.01	62.94
Self-supervised Learning									
DGI	91.00	75.35	69.08	79.60	66.64	60.93	83.61	72.49	60.73
GraphMAE	90.85	76.36	70.14	79.53	67.57	64.14	OOM	OOM	OOM
GRACE	91.98	77.09	70.99	80.50	67.54	63.40	85.56	73.94	63.25
Self-supervised Transfer									
DGI	62.47	33.51	26.47	40.18	23.20	41.27	38.73	27.08	14.46
GraphMAE	88.82	73.52	66.77	75.80	62.54	62.39	27.21	24.77	11.07
GRACE	79.23	54.71	46.08	57.52	41.50	53.99	56.08	40.61	26.03
In-context Learning									
Prodigy	73.57	62.54	58.31	69.81	61.63	60.24	68.35	58.43	42.14
OFA	72.35	60.57	56.47	67.99	59.53	59.88	69.23	58.23	42.68
GSPT-void	91.92	77.00	71.40	80.02	68.37	65.51	86.06	78.26	63.12
GSPT-desc	92.89	**80.55**	**75.61**	**82.23**	**70.88**	**70.85**	87.67	81.46	**69.40**

Existing self-supervised learning methods exhibit negative transfer. We now assess the transferability of existing self-supervised learning (SSL) methods. In Table 1, 'Self-supervised Learning' refers to performing SSL on the individual downstream datasets, while 'Self-supervised Transfer' means pretraining on ogbn-papers100M and evaluating with the pretrained checkpoint. Comparing the results from the two groups, we observe a drastic performance drop in the transfer learning setting. This suggests that models pretrained with existing SSL methods cannot directly generalize to different datasets, even when they belong to similar domains. We attribute this to the structural shift from the pretraining graph to the downstream datasets, particularly highlighted by the poor performance on Arxiv23, which exhibits distinct properties compared to the pretraining dataset (See Appendix A.6 for details).

GSPT exhibits strong in-context learning ability. Both variants of GSPT achieve competitive performance across all baselines. For the two variants, GSPT-desc consistently outperforms GSPT-void, suggesting that our pretrained model can further leverage the additional information provided in the class descriptions. Among training-free methods[1] including feature-based and in-context learning approaches, GSPT surpass others by a significant margin, demonstrating its capability to promote positive transfer by effectively leveraging the pretrained knowledge. Notably, GSPT-desc achieves the highest accuracy in most scenarios, even outperforming expert GNN models trained directly on the downstream dataset. This indicates that when label information is sparse, in-context learning can be a better solution than training task-specific models.

Overall, GSPT demonstrates strong cross-graph transferability, achieving better or comparable accuracy compared to end-to-end methods on few-shot node classification tasks across different datasets, without modifying any pretrained parameters (Answer to **RQ1**).

[1]Here training-free means the methods are not trained on the downstream dataset.

4.2.3 Analysis of GSPT's in-context capability

We aim to understand the strong in-context learning ability of GSPT from the perspective of attention mechanisms, using GSPT-desc as an example. Specifically, using each class prototype as the query, we visualize the attention distribution computed by the Transformer on its context nodes of different classes. As illustrated in Figure 3, the pretrained Transformer learns to adaptively propagate messages within the augmented graph, i.e., allowing class nodes to selectively attend to regular nodes that share the same ground-truth labels, and vice versa (Answer to **RQ2**). Consequently, the similarity between intra-class nodes increases, while it decreases for inter-class nodes. In contrast, the Transformer without pretraining relies solely on the graph structure for message passing, and thus the performance is bounded by the homophily of the original graph.

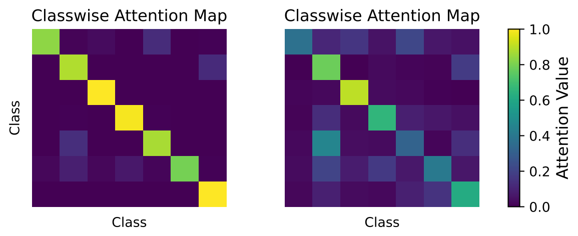


Figure 3: Attention map on classes of Cora. Left: attention weights obtained by the pretrained Transformer. Right: attention weights w/o pretraining.

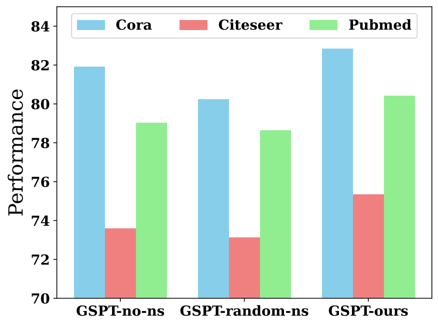


Figure 4: Ablation studies of different negative sampling strategies.

4.3 Ablation Study

In this subsection, we compare three variants of negative sampling approaches used in GSPT's pretraining stage. Specifically, GSPT-no-ns refers to pretraining without negative sampling, i.e., using the entire random walk sequences as input. GSPT-random-ns means randomly sampling K independent nodes at the end of each sequence. In GSPT-ours, we randomly sample one negative node for each sequence, and repeat the same node K times at the end of the sequence. The results are presented in Figure 4. We observe that GSPT-no-ns performs worse than GSPT-ours, demonstrating that the feature reconstruction task alone cannot compel the model to effectively learn interaction patterns without the inclusion of negative samples. On the other hand, GSPT-random-ns performs even worse, as it introduces too much noise to the input sequences, interfering with the training. Overall, our negative sampling approach, while simple and straightforward, is effective to aid the pretraining process (Answer to **RQ3**).

4.4 Scaling Effect

The NLP and CV domains have observed a scaling law in Transformer-based models, where performance on downstream tasks improves as more data is used for pretraining. To investigate whether GSPT exhibits a similar property, we conducted controlled experiments using different ratios of the pretraining dataset and evaluated the corresponding performance on downstream tasks. Specifically, we randomly selected varying proportions of METIS subgraphs for pretraining. As shown in Figure 5, GSPT's performance on both node classification and link prediction improves as more data is used for pretraining. This demonstrates the potential for further enhancement of our pretraining method if industry-scale data becomes available (Answer to **RQ4**).

5 Conclusion

In this work, we present a novel graph pretraining framework based on random walks and a standard Transformer architecture. Building upon the unified feature space provided by LLM embeddings, we leverage masked feature reconstruction to perform fully self-supervised learning to learn transferrable node representations across different graphs. By pretraining on massive-scale graphs with over 100 million nodes, our framework demonstrates impressive transferability and achieves promising performance on both node-level and link-level tasks. Our findings enhance the comprehension of the LLM-unified feature space in graph data and offer valuable insights for the creation of a versatile and generalizable graph foundation model.

References

[1] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

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

[3] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[4] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

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

[6] Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Michael Galkin, and Jiliang Tang. Graph foundation models. *arXiv preprint arXiv:2402.02216*, 2024.

[7] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1150–1160, 2020.

[8] Alex O Davies, Riku W Green, Nirav S Ajmeri, et al. Its all graph to me: Foundational topology models with contrastive learning on multiple domains. *arXiv preprint arXiv:2311.03976*, 2023.

[9] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv: Learning*, 2019.

[10] Jun Xia, Chengshuai Zhao, Bozhen Hu, Zhangyang Gao, Cheng Tan, Yue Liu, Siyuan Li, and Stan Z. Li. Mole-bert: Rethinking pre-training graph neural networks for molecules. In *International Conference on Learning Representations*, 2023.

[11] Yixin Liu, Shirui Pan, Ming Jin, Chuan Zhou, Feng Xia, and Philip S. Yu. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35:5879–5900, 2021.

[12] Zehong Wang, Zheyuan Zhang, Chuxu Zhang, and Yanfang Ye. Subgraph pooling: Tackling negative transfer on graphs. In *International Joint Conferences on Artificial Intelligence*, 2024.

[13] Haitao Mao, Zhikai Chen, Wei Jin, Haoyu Han, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. Demystifying structural disparity in graph neural networks: Can one size fit all? *Advances in Neural Information Processing Systems*, 36, 2024.

[14] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Haifang Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. Exploring the potential of large language models (llms) in learning on graphs. *ArXiv*, abs/2307.03393, 2023.

[15] Yanchao Tan, Zihao Zhou, Hang Lv, Weiming Liu, and Carl Yang. Walklm: A uniform language model fine-tuning framework for attributed graph embedding. *Advances in Neural Information Processing Systems*, 36, 2024.

[16] Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S. Dhillon. Node feature extraction by self-supervised multi-scale neighborhood prediction. *ArXiv*, abs/2111.00064, 2021.

[17] Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy Liang, and Jure Leskovec. Prodigy: Enabling in-context learning over graphs. *arXiv preprint arXiv:2305.12600*, 2023.

[18] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Doll'ar, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15979–15988, 2021.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[20] Mingyue Tang, Pan Li, and Carl Yang. Graph auto-encoder via neighborhood wasserstein reconstruction. In *International Conference on Learning Representations*, 2021.

[21] Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. *arXiv preprint arXiv:2204.04879*, 2022.

[22] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[23] Bryan Perozzi, Rami Al-Rfou, and Steven S. Skiena. Deepwalk: online learning of social representations. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014.

[24] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 594–604, 2022.

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

[26] Haitao Mao, Juanhui Li, Harry Shomer, Bingheng Li, Wenqi Fan, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. Revisiting link prediction: a data perspective. In *The Twelfth International Conference on Learning Representations*, 2024.

[27] Lu Lin, Jinghui Chen, and Hongning Wang. Spectral augmentation for self-supervised learning on graphs. *arXiv preprint arXiv:2210.00643*, 2022.

[28] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[29] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[30] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

[31] Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. Nagphormer: A tokenized graph transformer for node classification in large graphs. In *International Conference on Learning Representations*, 2022.

[32] Dexiong Chen, Till Hendrik Schulz, and Karsten Borgwardt. Learning long range dependencies on graphs via random walks. *arXiv preprint arXiv:2406.03386*, 2024.

[33] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. *Advances in neural information processing systems*, 32, 2019.

[34] Ladislav Rampavsek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and D. Beaini. Recipe for a general, powerful, scalable graph transformer. *ArXiv*, abs/2205.12454, 2022.

[35] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017.

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

[37] Felix Wu, Tianyi Zhang, Amauri H. de Souza, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, 2019.

[38] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio', and Yoshua Bengio. Graph attention networks. *ArXiv*, abs/1710.10903, 2017.

[39] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Lio', Yoshua Bengio, and R. Devon Hjelm. Deep graph infomax. *ArXiv*, abs/1809.10341, 2018.

[40] Zhu Yanqiao, Xu Yichen, Yu Feng, Liu Qiang, Wu Shu, and Wang Liang. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020.

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

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

[43] Minghao Xu, Hang Wang, Bingbing Ni, Hongyu Guo, and Jian Tang. Self-supervised graph-level representation learning with local and global structure. In *International Conference on Machine Learning*, pages 11548–11558. PMLR, 2021.

[44] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823, 2020.

[45] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[46] Jun Xia, Chengshuai Zhao, Bozhen Hu, Zhangyang Gao, Cheng Tan, Yue Liu, Siyuan Li, and Stan Z. Li. Mole-BERT: Rethinking pre-training graph neural networks for molecules. In *The Eleventh International Conference on Learning Representations*, 2023.

[47] Yu Rong, Yatao Bian, Tingyang Xu, Wei yang Xie, Ying Wei, Wen bing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. *arXiv: Biomolecules*, 2020.

[48] Jiarong Xu, Renhong Huang, Xin Jiang, Yuxuan Cao, Carl Yang, Chunping Wang, and Yang Yang. Better with less: A data-active perspective on pre-training graph neural networks. *Advances in Neural Information Processing Systems*, 36:56946–56978, 2023.

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

[50] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. Learning on large-scale text-attributed graphs via variational inference. *ArXiv*, abs/2210.14709, 2022.

A Appendix

A.1 Extending GSPT to Link Prediction

Our proposed framework can be readily adapted to the link prediction task with minor adjustments. The key consideration is to construct an appropriate context for link prediction in a sequence-based manner to accomodate the Transformer architecture. Unlike node classification, which necessitates precise feature propagation, link prediction emphasizes modeling the neighborhood information of nodes such as the number of common neighbors and their distribution [26]. To achieve this, we adopt the 'Hop2Token' [31] approach for constructing the input context, i.e., $h^0 = [x_0, x_1, \ldots, x_{l-1}]$, where x_0 represents the feature of the center node and x_k the aggregated feature of its K-hop neighborhood. This method ensures that the input sequence encapsulates the necessary neighborhood information for effective link prediction.

To better capture the global connectivity in the graph, we incorporate a structure-aware decoder on top of the Transformer. This decoder takes the output of the Transformer and integrates them based on the graph's connectivity structure, enhancing the model's ability to predict links by leveraging global graph properties. Formally, the loss function for pretraining is computed by:

$$H^L = \text{TRM}(H^0) \tag{6}$$

$$H_{node} = \text{Pooling}(H^L) \tag{7}$$

$$H^D = \text{Dec}(H_{node}, A) \tag{8}$$

$$\mathcal{L}_{\text{link}} = \frac{1}{m} \sum_{j=1}^{m} \left(1 - \cos(h_i^D, x_i)\right) \tag{9}$$

Pooling obtains the node representations by concatenating the output of the 0-hop (center) node and the averaged outputs of all other hops. $\text{Dec}(H^L, A)$ indicates a structure-aware decoder that takes node representations and the adjacency matrix A as input, and outputs the decoded embeddings H^D. For simplicity, we use h_i^D to denote the i-th row of H^D. The difference from node-level task is that (1) H^0 is constructed with multi-hop aggregated features rather than independent nodes and (2) a decoder is used before computing the reconstruction loss. In practice, we adopt a one-layer GCN as the decoder.

A.2 Link Prediction Results

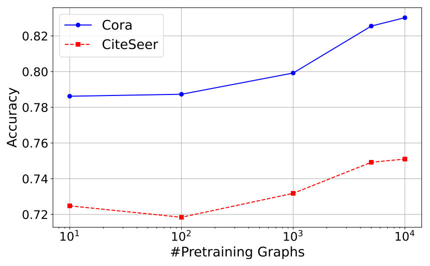
Setup. For the link prediction task, we construct the dataset by randomly sampling 80%, 10%, and 10% of the edges in the graph for the training, validation, and test sets, respectively. We use the Mean Reciprocal Rank (MRR) as our evaluation metric. Unlike the in-context node classification setting, we finetune the pretrained GSPT on the downstream dataset to better adapt it to the specific properties of the individual dataset.

Baselines. We compare our method against two categories of baselines. For feature-based approaches, we use the SentenceBert embeddings and Node2Vec embeddings which are solely based on the feature and structure, respectively. For GNNs, we include GCN and GraphSAGE for comparison. We do not compare with GNN4LP methods like [41, 42] as they explicitly incorporate pairwise information when computing the edge scores, and do not reflect the capacity of node representations. Subgraph-based methods like OneForAll [5] and Prodigy [17] present severe efficiency issues, which makes them non-applicable to large-scale pretraining for link prediction. For our method, we evaluate both the variant trained from scratch (GSPT-TFS) and the variant fine-tuned from the pretrained checkpoint (GSPT-pretrained). In all methods, we employ an MLP on top of the generated node embeddings to compute edge scores.

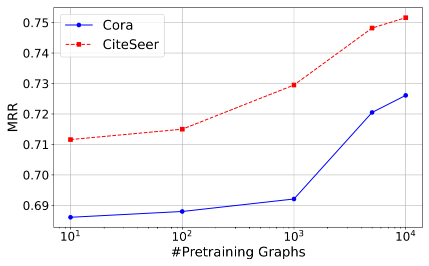
Results. As shown in Table 2, GSPT-pretrained surpasses all baseline methods. Notably, GSPT-pretrained consistently outperforms GSPT-TFS. This demonstrates that pretraining on a large-scale dataset allows the model to effectively learn the underlying principles of edge formation, leading to improved performance when transferred to downstream datasets (Answer to **RQ1**).

A.3 Scaling Effect

We show the scaling effect of GSPT by adding more data for pretraining in Figure 5.



(a) Node Classification



(b) Link Prediction

Figure 5: Scaling effect of GSPT. (a) Node classification performance on downstream datasets with linear probing. (b) Link prediction performance on downstream datasets via fine-tuning. X-axis denotes the number of METIS graphs used for pretraining. Empirically, GSPT improves as adding more data to pretraining.

Table 2: Performance comparison of link prediction on citation datasets. We adopt MRR with 100 negative samples as the evaluation metric.

Method	Cora	Citeseer	Pubmed	Arxiv23
Feature-based				
SentenceBert	55.67	64.19	63.47	70.51
Node2vec	58.06	46.26	53.53	56.50
GNN				
GCN	70.25	64.88	78.96	79.39
GraphSAGE	64.12	63.95	79.16	78.04
GSPT				
GSPT-TFS	68.61	71.16	79.48	79.67
GSPT-pretrained	**72.67**	**75.16**	**81.59**	**82.13**

A.4 Related Works

Graph self-supervised learning. Limited by feature heterogeneity, studies in graph SSL are often limited to individual graphs [24, 43, 44], or focus on specific applications in biology and chemistry [45–47], where a fixed-size ’vocabulary’ allows for consistent representation of node or edge features across different graphs. Alternatively, some approaches entirely discard the original features to facilitate knowledge transfer based solely on structural information [7, 8, 48], leading to sub-optimal performance. These constraints significantly hinder the broader application of graph SSL in real-world scenarios, particularly in developing a versatile and generalizable model.

Feature-centric graph learning. To mitigate feature heterogeneity, [5] propose converting varied node features into text and training the model within this unified text space instead of the original vector spaces. Among the text-based methodologies, [14, 49] use frozen large language models (LLMs) to create fixed text embeddings for nodes across different graphs. In contrast, [16, 50] implement a cascaded architecture that combines Language Models (LMs) and Graph Neural Networks (GNNs) to directly learn graph-aware text embeddings from original text attributes and graph topology. These approaches have demonstrated improved performance on various TAGs tasks, highlighting the critical importance of high-quality features in addition to graph structures.

Graph foundation models. Developing a graph foundation model [6] with a unified architecture that can adapt to diverse downstream tasks has been a popular research topic in the graph domain. Based on a LM-unified feature space, OneForAll [5] trains a model with a shared backbone across different data and tasks by transforming them into the same form, but its scope is limited to small-scale supervised learning. Prodigy [17] focuses on empowering graph models with the ability to "learn in context" through pretraining, yet its specialized training strategy restricts it to few-shot inference and limited data utilization due to computational inefficiency. Meanwhile, these methods still rely on message passing, which has been shown to have limitations when generalizing to graphs with different structural or attribute properties [13, 27]. In contrast to previous works, our proposed

GSPT (1) presents a novel strategy to pretrain a backbone model across graphs without requiring supervision; (2) scales to massive scale graphs with efficiency and effectiveness; (3) showcases a message passing-free pretraining strategy, opening up new avenues for graph foundation model development.

A.5 Proof of Propositions

A.5.1 Proof of Proposition 1

[Neighbor Coverage] Let $G = (V, E)$ be a graph, $v \in V$ a node, and $N_K(v)$ its K-hop neighborhood. Given m random walks of length l starting from v, the probability $p(l, m)$ that all nodes in $N_K(v)$ are visited by at least one random walk is bounded by:

$$p(l, m) \geq 1 - |N_K(v)| \cdot \left(\frac{H_K}{l}\right)^m.$$

Let H_K be an upper bound on the expected hitting time for any pair of nodes within K hops in G, i.e., for all $u \in N_K(v)$, $H(v, u) \leq H_K$.

For a single random walk of length l, the probability of missing a node $u \in N_K(v)$ can be bounded using Markov's inequality as:

$$P(\text{missing } u \text{ in one walk}) \leq \frac{H_K}{l}.$$

For m independent random walks, the probability of missing u in all walks is:

$$P(\text{missing } u \text{ in } m \text{ walks}) \leq \left(\frac{H_K}{l}\right)^m.$$

Using the union bound, the probability of visiting all nodes in $N_K(v)$ is:

$$P(\text{visiting all nodes in } N_K(v)) \geq 1 - |N_K(v)| \cdot \left(\frac{H_K}{l}\right)^m.$$

As $l \to \infty$, $\frac{H_K}{l} \to 0$, and as $m \to \infty$, $\left(\frac{H_K}{l}\right)^m \to 0$ for any $l > H_K$. Since $|N_K(v)|$ is finite, it follows that $p(l, m) \to 1$ as l and/or m increase.

A.5.2 Proof of Proposition 2

[Proximity Preservation] Let $H(v, u)$ be the expected hitting time from v to u in a graph G. For any two nodes $u_1, u_2 \in V$ such that $H(v, u_1) < H(v, u_2)$, there exists an m large enough such that:

$$P(f(v, u_1) < f(v, u_2)) > 1 - \delta,$$

for any $\delta > 0$, where $f(v, u)$ is the average position of u in m random walks starting from v.

We are interested in bounding the probability $P(f(v, u_1) \geq f(v, u_2))$, which can be written as:

$$P\left(\frac{1}{m}\sum_{i=1}^{m} T_i(v, u_1) \geq \frac{1}{m}\sum_{i=1}^{m} T_i(v, u_2)\right).$$

where $T_i(v, u)$ denotes the hitting time for the i-th random walk from v to u. This is equivalent to:

$$P\left(\frac{1}{m}\sum_{i=1}^{m}(T_i(v, u_2) - T_i(v, u_1)) \leq 0\right).$$

Define $Z_i = T_i(v, u_2) - T_i(v, u_1)$. Thus, we aim to bound the probability:

$$P\left(\frac{1}{m}\sum_{i=1}^{m} Z_i \leq 0\right).$$

The random variables Z_i are bounded, as both $T_i(v, u_1)$ and $T_i(v, u_2)$ are bounded. The expected value of Z_i is:

$$\mathbb{E}[Z_i] = H(v, u_2) - H(v, u_1) = \gamma > 0.$$

Applying Hoeffding's inequality with $\epsilon = \gamma$, we get:

$$P\left(\frac{1}{m}\sum_{i=1}^{m} Z_i \leq 0\right) = P\left(\frac{1}{m}\sum_{i=1}^{m} Z_i - \gamma \leq -\gamma\right) \leq \exp\left(-\frac{2m\gamma^2}{(b-a)^2}\right).$$

As m increases, this probability decays exponentially. Therefore, for any $\delta > 0$, there exists an m such that $P(f(v, u_1) < f(v, u_2)) > 1 - \delta$.

A.6 Datasets

Summary. The summary of datasets used in the experiments are presented in Table 3.

Table 3: Summary of datasets

Name	#Nodes	#Edges
Cora	2,708	10,858
Citeseer	3,186	8,554
Pubmed	19,717	88,670
Arxiv23	46,198	78,548
ogbn-papers100M	111,059,956	1,615,685,872

Partition. For ogbn-papers100M, we use the METIS algorithm to partition the graph into 11105 non-overlapping subgraphs. The statistics of the subgraphs are listed in Table 4. We use the implementation by dgl in our experiments.

Table 4: Summary of METIS partitions on ogbn-papers100M

#Graphs	Avg. #Nodes	Avg. #Edges	#Node Range	#Edge Range
11105	10000.90	61357.03	303 - 45748	328 - 122644

Properties. The graphs used in the experiments exhibit varying properties. To characterize these properties, we compute four commonly used graph metrics: average degree, sparsity, clustering coefficient, and homophily. The results are summarized in Table 5. For the ogbn-papers100M dataset, we compute the average statistics using 100 randomly sampled METIS subgraphs. Note that ogbn-papers100M does not contain label data, so the homophily cannot be obtained.

Table 5: Summary of graph properties

Dataset	Average Degree	Homophily	Clustering Coefficient	Graph Density
Cora	8.02	0.81	0.24	0.00144
Citeseer	5.37	0.79	0.14	0.00083
Pubmed	8.99	0.80	0.06	0.00023
Arxiv23	3.40	0.65	0.05	0.00004
Papers100M	13.77	N/A	0.16	0.00173

A.7 Accessibility

Citeseer: `https://github.com/CurryTang/Graph-LLM`

Cora, Pubmed: `https://github.com/kimiyoung/planetoid`

ogbn-papers100M: `https://ogb.stanford.edu/docs/nodeprop/`

Arxiv23: `https://github.com/XiaoxinHe/tape_arxiv_2023`

A.8 Experimental Details

GSPT. The hyperparameters used for GSPT pretraining are listed in Table 6 and Table 7. GSPT-node indicates the version for node classification, while GSPT-link denotes the one for link prediction. For the few-shot node classification tasks, we randomly initiate 3 random walks from each node, and use the validation set to select the best results. For link prediction, GSPT involves fine-tuning the pretrained model on individual datasets. We search the hyperparameters from Table 8 during fine-tuning, using an independent edge set for validation.

Table 6: Hyperparameters of GSPT-node

Hyperparameter	Value	Explanation
mask_rate	0.2	Probability of masking a node
p_random	0.2	Probability of replacing [MASK] with random nodes, see [19]
p_unchanged	0.2	Probability of keeping masked nodes unchanged, see [19]
hidden_dim	768	Dimension of hidden layers
ffn_dim	3072	Dimension of feed-forward network layers
n_layers	3	Number of Transformer layers
n_heads	12	Number of attention heads
epochs	10	Number of training epochs
weight_decay	0.01	strength of L2 regularization
peak_lr	0.0001	Peak learning rate
end_lr	0.00001	End learning rate (after decay)
warmup_updates	10000	Number of warmup updates
dropout	0.3	Dropout rate
attention_dropout	0.3	Dropout rate for attention layers
emb_dropout	0.3	Dropout rate for embeddings
p	0.25	Return parameter of random walk, see [22]
q	0.25	In-out parameter of random walk, see [22]
walk_length	20	Length of random walk

Table 7: Hyperparameters of GSPT-link

Hyperparameter	Value	Explanation
mask_rate	0.5	Probability of masking a node
p_random	0	Probability of replacing [MASK] with random nodes, see [19]
p_unchanged	0	Probability of keeping masked nodes unchanged, see [19]
hidden_dim	384	Dimension of hidden layers
ffn_dim	768	Dimension of feed-forward network layers
n_layers	2	Number of Transformer layers
n_heads	8	Number of attention heads
epochs	100	Number of training epochs
weight_decay	0	strength of L2 regularization
peak_lr	0.001	Peak learning rate
end_lr	0.0001	End learning rate (after decay)
warmup_updates	100	Number of warmup updates
dropout	0.1	Dropout rate
attention_dropout	0.1	Dropout rate for attention layers
emb_dropout	0	Dropout rate for embeddings
n_hops	3	Number of hops for feature aggregation

Baselines. For all baselines, we search the optimal hyperparameters using the validation set of individual datasets. We organize the hyperparameters based on the task type into the following.

Few-shot node classification. Table 9 contains the details for reproducing the results for end-to-end methods in Table 1. Such methods involve training specific models on individual downstream datasets.

Table 8: Hyperparameters of GSPT-link, fine-tuning.

Hyperparameter	**Value**	**Explanation**
lr	[1e-3, 1e-4]	Learning rate
projector_layers	3	Number of layers in the projector module to compute edge scores
projector_dim	256	Dimension of the projector module to compute edge scores
epochs	1000	Number of training epochs
patience	20	Patience of early stopping
batch_size	4096	Number of samples (edges) per batch

Table 9: Hyperparameters of GNNs on few-shot node classification

Hyperparameter	lr	weight_decay	hidden_dim	dropout	num_layers	num_heads
Search Range	[1e-2, 1e-3]	[0, 1e-4, 5e-4]	[64, 256, 384]	[0, 0.5]	[2]	[4, 8]

Link prediction. Table 10 presents the hyperparameters of GNN baselines for link prediction to reproduce results in Table 2. Specifically, we adopt an MLP on top of the GNNs that takes the hadamard product of node representations as input, and output the score of edge existence.

Table 10: Hyperparameters for link prediction baselines.

Model	**Hyperparameter**	**Search Range**
GNN	num_layers	[1, 2, 3]
	hidden_dim	[128, 256]
	num_heads	[1, 4]
MLP	num_layers	[1, 2, 3]
	hidden_dim	[128, 256]
General	lr	[1e-3, 1e-2]
	weight_decay	[0, 1e-5]
	dropout	[0.1, 0.5]
	batch_size	4096
	epochs	1000
	patience	20