From Anchors to Answers: A Novel Node Tokenizer for Integrating Graph Structure into Large Language Models

Yanbiao Ji*
Shanghai Jiao Tong University
Shanghai, China
jiyanbiao@sjtu.edu.cn

Chang Liu*†
Shanghai Jiao Tong University
Shanghai, China
isonomialiu@sjtu.edu.cn

Xin Chen
The Chinese University of Hong Kong
Hong Kong, China
xchen@se.cuhk.edu.hk

Dan Luo
Lehigh University
Bethlehem, PA, USA
danluo.ir@gmail.com

Mei Li
Shanghai Jiao Tong University
Shanghai, China
mei-li@sjtu.edu.cn

Yue Ding‡
Shanghai Jiao Tong University
Shanghai, China
dingyue@sjtu.edu.cn

Wenqing Lin‡
Tencent
Shenzhen, China
edwlin@tencent.com

Hongtao Lu
Shanghai Jiao Tong University
Shanghai, China
htlu@sjtu.edu.cn

Abstract

Enabling large language models (LLMs) to effectively process and reason with graph-structured data remains a significant challenge despite their remarkable success in natural language tasks. Current approaches either convert graph structures into verbose textual descriptions, consuming substantial computational resources, or employ complex graph neural networks as tokenizers, which introduce significant training overhead. To bridge this gap, we present NT-LLM, a novel framework with an anchor-based positional encoding scheme for graph representation. Our approach strategically selects reference nodes as anchors and encodes each node's position relative to these anchors, capturing essential topological information without the computational burden of existing methods. Notably, we identify and address a fundamental issue: the inherent misalignment between discrete hop-based distances in graphs and continuous distances in embedding spaces. By implementing a rank-preserving objective for positional encoding pretraining, NT-LLM achieves superior performance across diverse graph tasks ranging from basic structural analysis to complex reasoning scenarios. Our comprehensive evaluation demonstrates that this lightweight yet powerful approach effectively enhances LLMs' ability to understand and reason with graph-structured information, offering an efficient solution for graph-based applications of language models.

CCS Concepts

• **Information systems** → **Data mining**.

*Both authors contributed equally to this research.
†This work was done while Chang Liu was an intern at Tencent.
‡Corresponding authors: Yue Ding and Wenqing Lin.

CIKM '25, November 10–14, 2025, Seoul, Republic of Korea

ACM ISBN 979-8-4007-2040-6/2025/11
https://doi.org/10.1145/3746252.3761167

Keywords

Large Language Models, Positional Encoding, Knowledge Graphs

ACM Reference Format:
Yanbiao Ji, Chang Liu, Xin Chen, Dan Luo, Mei Li, Yue Ding, Wenqing Lin, and Hongtao Lu. 2025. From Anchors to Answers: A Novel Node Tokenizer for Integrating Graph Structure into Large Language Models. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3746252.3761167

1 Introduction

In recent years, Large Language Models (LLMs), such as LLaMA [59] and GPT [45], have revolutionized artificial intelligence. They have demonstrated powerful capabilities in solving various natural language processing (NLP) tasks, including question answering [36, 53], text generation [2, 50], and document understanding [27, 68]. While LLMs have primarily been applied to text data, an increasing number of applications now involve text data intertwined with structured information represented as graphs. For instance, in social networks, nodes represent entities, while edges capture the relationships between them. Both nodes and edges can also be associated with textual descriptions that detail their attributes. Since LLMs are primarily designed to model text in a sequential format, applying them to graph-related tasks presents new challenges, particularly in encoding the structural information of graphs [15, 35].

While many studies [28, 40, 65] have attempted to combine language modeling and graph representation learning with medium-sized transformer models such as BERT [12] and RoBERTa [41], efficient graph reasoning with LLMs of billions of parameters remains challenging. To leverage the strength of LLMs for graph structure understanding, existing efforts can be categorized into two groups [25, 52]: (1) **Graph Textual Conversion**, which translates a graph's structure into a descriptive textual representation [9, 26, 56]. These studies typically convert the local context of a target node into textual descriptions that incorporate relevant structural information, and then utilize large language models to predict properties such as node labels and the presence of links. The underlying assumption is that the powerful capabilities of LLMs can generalize to interpret graph-structured knowledge through textual input.

However, such descriptions typically require a large number of tokens to describe the graph structure, greatly increasing the cost of LLM inference. (2) **Graph Node Tokenizer**, which generates node embeddings for each node and then projects these embeddings into LLM token space [38, 57, 58]. With the utilization of powerful Graph Neural Networks (GNNs) as graph node tokenizers, these methods effectively reduce the inference cost by representing the graph structure with compact node tokens. However, the graph representation learning process often brings heavy training overhead. Achieving scalability comparable to LLMs requires an expressive GNN (*e.g.,* with elaborate graph convolution paradigms) of similar scale, which introduces additional computational overhead.

To enable effective and efficient LLM reasoning on graphs, a graph encoding paradigm that preserves rich graph structural information without introducing heavy training or inference overhead is needed. This naturally aligns with the motivation of graph positional encoding, which introduces extra embeddings containing structural information to disambiguate nodes and enhance graph representation learning during the training of GNNs and graph transformers [7, 14, 49]. In this paper, we introduce an anchor-based graph positional encoding scheme for graph node tokenization, and investigate its integration with LLMs across various graph-related tasks. The core of our method is the strategic selection of key nodes, referred to as anchors, which serve as reference points for encoding the graph topology. Each node is then represented based on its relative distance to these anchors, effectively capturing the structural information of the graph. Furthermore, we identify the issue of misalignment between the non-Euclidean graph space (hop-based discrete distance) and the Euclidean embedding space (continuous Euclidean distance). A rank-preserving pretraining objective is proposed to project the positional embedding into Euclidean space. We then apply task-specific tuning procedures using prompt tuning and LoRA techniques to facilitate better structural understanding of LLMs for downstream tasks. Extensive empirical studies demonstrate that NT-LLM substantially improves LLM performance across a diverse range of graph-related tasks, from basic graph analysis to complex reasoning. Our main contributions are as follows:

- We introduce a position-anchored graph encoding approach for LLMs that efficiently preserves crucial structural information while reducing the computational complexity associated with commonly used graph encoding methods.
- We identify and address the issue of misalignment between the non-Euclidean graph space and the Euclidean embedding space, which hinders the effectiveness of graph positional embedding in graph reasoning with LLMs.
- We conduct an extensive empirical evaluation on multiple graph benchmarks, covering a wide range of task complexities and graph types. Our results provide insights into the performance and generalizability of NT-LLM, highlighting its potential for adoption in various graph learning scenarios.

2 Related Work

2.1 Graph Positional Encoding

Graph Neural Networks (GNNs) have significantly advanced graph representation learning by enabling the extraction of meaningful embeddings from graph-structured data through message-passing mechanisms [4, 5, 30, 44, 61]. However, standard GNN architectures often struggle to differentiate among nodes with similar local structures but different positions within the global graph topology. Graph positional encoding addresses this limitation by enhancing node representations with positional information, allowing the capture of important structural features.

Several approaches have been developed to encode positional information in graphs. Laplacian eigenmaps [6, 7] utilize the eigenvectors of the graph Laplacian matrix for this purpose. In contrast, random walk encodings [8, 48, 69] capture structural information by simulating random walks on the graph. This method encodes the co-occurrence probabilities of nodes during these walks, thereby embedding nodes with similar neighborhoods closer in the embedding space. Rx'ecently, researchers have introduced Distance Encoding [13, 33, 49], which incorporates structural information by encoding the shortest path distances between nodes. Furthermore, Random Feature methods [1, 14] have been developed to approximate positional encodings using learnable or predefined random feature maps. To provide a comprehensive overview of these approaches, Table 1 presents a detailed comparison of various graph positional encoding methods.

2.2 LLMs in Graph-Related Tasks

The rapid advancement in LLMs have led to their successful application across various domains, leveraging their powerful sequence modeling capabilities [37, 39, 62]. In recent years, there has been a growing interest in applying LLMs to graph-related tasks, aiming to harness their ability to capture long-range dependencies and perform complex reasoning.

Initial efforts focused on directly feeding textual descriptions of graphs into LLMs to tackle tasks such as node classification and link prediction [18, 24]. While these methods demonstrated the potential of LLMs in understanding graph data, they faced significant scalability challenges due to the complexity of constructing comprehensive prompts and the loss of crucial structural information during the graph-to-text conversion process. To address these limitations, subsequent research has explored the integration of Graph Neural Networks (GNNs) with LLMs to better leverage the strengths of both paradigms [20, 23, 57]. One common approach involves using GNNs to generate structure-aware embeddings, which are then fed into LLMs for downstream tasks [57, 58]. More advanced techniques have delved into model fusion training [70], model alignment [34, 63], and the development of LLM agents specifically designed to handle graph data [10, 43].

3 Preliminary

Textual Graphs. A textual graph is a graph in which nodes and edges are associated with textual attributes. Formally, it is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \{\mathbf{T}_v\}_{v\in\mathcal{V}}, \{\mathbf{T}_e\}_{e\in\mathcal{E}})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the sets of nodes and edges, respectively. Here, $\mathbf{T}_v$ and $\mathbf{T}_e$ denote the textual attributes corresponding to each node and edge, which are usually represented by natural language descriptions.[1]

Text Encoding via Language Models. Language Models (LMs) have proven to be highly effective at encoding textual attributes

[1] In this work, we assume that the distance between two adjacent nodes is fixed at 1. The study of weighted graphs, where edge distances may vary, is left for future work.

Table 1: Comparative analysis of graph positional encoding techniques, including our proposed method.

	Laplacian Eigenmap [7]	DeepWalk [48]	PGNN [66]	HPLC [29]	RFP [14]	Ours
Encoding Scheme	eigenvectors	random walk	distance	distance, eigenvectors	random feature	distance
Local Structure	✓	✓	✓	✓	✓	✓
Global Position	×	×	✓	✓	×	✓
Euclidean Space	✓	✓	×	✓	✓	✓
Time Complexity	$O(\|\mathcal{V}\|^3)$	$O(\|\mathcal{E}\|)$	$O(\|\mathcal{V}\|^2 log^2(\|\mathcal{V}\|))$	$O(\|\mathcal{E}\|log(\|\mathcal{V}\|) + \|\mathcal{V}\|log^2(\|\mathcal{V}\|))$	-	$O(\|\mathcal{V}\|^2 + \|\mathcal{V}\|\|\mathcal{E}\|)$

in graphs, producing embeddings that capture rich semantic information. For a given textual attribute T_i associated with a node or edge i, an LM encodes this attribute into an embedding vector as follows:

$$\mathbf{x}_i = \mathbf{LM}(T_i) \in \mathbb{R}^k. \tag{1}$$

Prompt Tuning for LLMs. LLMs are trained on vast corpora of textual data, demonstrating emergent capabilities that facilitate advanced semantic understanding and exceptional task generalization. Formally, an LLM parameterized by θ takes as input a sequence of tokens $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ along with a task prompt $\mathbf{P}$, and generates an output sequence $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_r\}$. The probability distribution of the output sequence, conditioned on the concatenated input sequence and prompt $[\mathbf{P}; \mathbf{X}]$, is expressed as:

$$p_\theta(\mathbf{Y}|[\mathbf{P}; \mathbf{X}]) = \prod_{i=1}^{r} p_\theta(\mathbf{y}_i|\mathbf{y}_{<i}, [\mathbf{P}; \mathbf{X}]), \tag{2}$$

where $\mathbf{y}_{<i}$ represents the prefix of sequence $\mathbf{y}$ up to position $i-1$, and $p_\theta(\mathbf{y}_i|\mathbf{y}_{<i}, [\mathbf{P}; \mathbf{X}])$ denotes the probability of generating token $\mathbf{y}_i$ given the preceding tokens $\mathbf{y}_{<i}$ and the input $[\mathbf{P}; \mathbf{X}]$.

Prompt tuning [32] is an efficient technique for adapting LLMs to specific tasks without modifying the model's parameters. This technique keeps the pretrained LLM frozen, and optimizes a small set of continuous prompt embeddings $\{\mathbf{e}_i\}_{i=1}^n$, where n is the number of prompt tokens. These prompts are generally initialized either randomly or using the embeddings of specific tokens, and are subsequently optimized throughout the training process. Formally, the prompt embeddings can be represented as:

$$\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_n]^T, \tag{3}$$

where the dimension of the embedding space is d, and $\mathbf{E} \in \mathbb{R}^{n \times d}$. The prompt embeddings can be generated by a small trainable mapping network Φ:

$$\mathbf{E} = \Phi(\mathbf{X}), \tag{4}$$

where $\mathbf{X}$ represents the input embeddings to be transformed. This allows for more flexible and expressive prompt representations. The generation process with prompt tuning can be represented as follows:

$$p_{\theta,\Phi}(\mathbf{Y}|[\mathbf{P}; \mathbf{X}]) = \prod_{i=1}^{r} p_{\theta,\Phi}(\mathbf{y}_i|\mathbf{y}_{<i}, [\mathbf{P}; \mathbf{X}]), \tag{5}$$

where θ represents the frozen parameters of the pretrained LLM, Φ is the learnable prompt mapping network, $\mathbf{P}$ is the prompt, $\mathbf{X}$ is the input sequence, and $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_r\}$ is the output sequence.

4 Methodology

We propose NT-LLM, which can seamlessly integrate graph-structure knowledge with LLMs through two key components: **Graph Node Tokenizer** and **Task-Specific LLM Tuning**. The node tokenizer leverages carefully selected anchor nodes to encode the spatial position of each node, and positional embedding pretraining to preserve geometric relationships between nodes. The task-specific LLM tuning integrates our node position embedding with prompt tuning and low-rank adaptation, which allows LLMs to effectively leverage both textual and graph-based information. Figure 1 illustrates the overall framework of NT-LLM.

4.1 Graph Node Tokenizer

In large language models, it is straightforward to inject information about the relative or absolute position of tokens in a sequence via their index. However, this approach is not feasible for graphs due to two key differences. First, *graphs do not have an inherent linear ordering of nodes*, unlike sequences, where tokens follow a clear order. Nodes in a graph are interconnected in a complex, multidimensional structure, where relationships are defined by edges, and there is no natural start or end. Second, *the neighborhood of each node can vary significantly in size and shape*, which makes the concept of a relative or absolute "position" less meaningful. To address this challenge, we propose a novel graph node tokenizer, which consists of three key steps: anchor node identification, node encoding, and Euclidean projection.

4.1.1 Anchor Node Identification. Prior works [11, 66] have demonstrated that using anchor nodes can well capture the position of a given node with respect to all other nodes in a graph. In particular, the position of a node can be described in terms of its relative distance (*e.g.,* shortest path distance) to these anchor nodes. For efficient identification of anchor nodes, we implement a greedy anchor selection algorithm with a coverage ratio threshold. The details of this greedy selection procedure are shown in Algorithm 1. Given a coverage ratio CR and coverage radius c, we start with an empty set $\mathcal{A}$ of anchor nodes and an empty set N_{cover} of covered nodes (Line 1). Here, we define that a node u is covered by a node v only if u is in the c-hop subgraph of node v; otherwise, u is considered uncovered by v. Then, we iteratively select a new anchor node that covers the maximum set of uncovered nodes in its c-hop subgraph $N_c(v)$ (Line 3) and add these covered nodes to N_{cover} (Line 8) until the size of N_{cover} is no less than $CR * |\mathcal{V}|$ (Line 2).

The identified anchor nodes enable us to provide a unique node description for other nodes in terms of their relative distance, capturing both global and local structures within the graph.

4.1.2 Node Encoding. Given the identified anchor nodes $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$, we encode the position of each node v with respect to these anchors:

$$\hat{\mathbf{d}}_v = (d_1, d_2, \ldots, d_K), \tag{6}$$

$$d_i = \text{dist}(v, a_i), \quad \forall i \in \{1, \ldots, K\}, \tag{7}$$

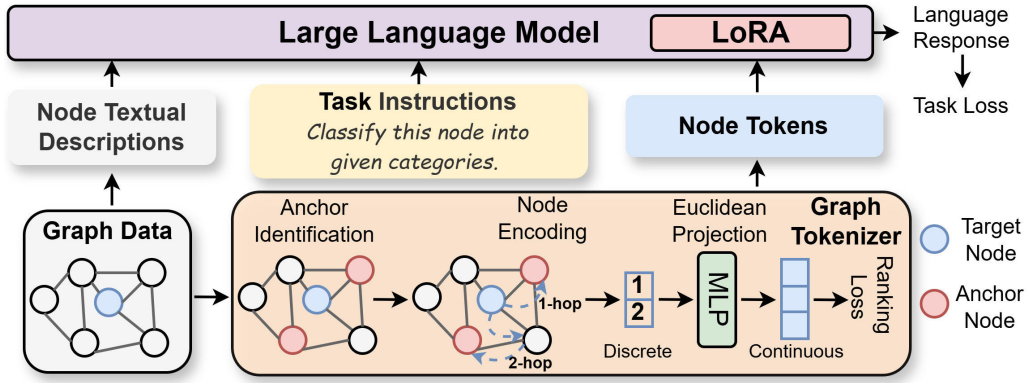

Figure 1: Overview of our proposed *NT-LLM* approach. It consists of two steps: (1) Graph Tokenizer: We select key nodes as anchors with a greedy algorithm and compute relative distances between nodes and these anchors to encode the graph structure. The relative distances are then projected into a continuous Euclidean space while preserving the partial ordering of node distances. (2) Task Tuning: We integrate the pretrained embeddings with a large language model using LoRA for task-specific fine-tuning of the LLM, enhancing the performance of downstream graph understanding tasks.

Algorithm 1 Greedy Algorithm for Anchor Node Selection

Require: Graph $G(\mathcal{V}, \mathcal{E})$, target coverage ratio CR, coverage radius c
Ensure: Set of anchor nodes $\mathcal{A}$

```
Initialize A ← ∅, N_cover ← ∅
while |N_cover| < CR * |V| do
    anchor ← arg max_{v∈V\A} |N_c(v) \ N_cover|
    if |N_c(anchor) \ N_cover| = 0 then
        break
    end if
    A ← A ∪ {anchor}
    N_cover ← N_cover ∪ N_c(anchor)
end while
return A
```

where $\text{dist}(v, a_i)$ denotes the number of hops in the shortest path between node v and anchor node a_i.

Utilizing relative distance, we can approximate the shortest distance between any two nodes u and v in the graph defined as:

$$\hat{d}(u,v) := \min_{k\in\{1,\ldots,K\}} \left(\hat{\mathbf{d}}_u[k] + \hat{\mathbf{d}}_v[k]\right), \tag{8}$$

where $\hat{\mathbf{d}}_u[k]$ means the k-th element of $\hat{\mathbf{d}}_u$. This approximation estimates the distance by identifying the anchor node that provides the minimal combined distance between u and v.

Note that our approximated shortest path distance may not be the actual shortest path distance. However, $\hat{d}(u,v)$ actually serves as an upper bound for the true shortest path distance between u and v. More formally, the error between the estimated distance and real distance is bounded by the parameters c and CR:

Lemma 4.1. *Given any two nodes u, v from a graph, the error of the estimated shortest path distance can be bounded by $2c$ with a probability no smaller than $1-(1-CR)^2$, where c is the coverage radius and CR is the coverage ratio.*

PROOF. Given node pair u, v from graph and a set of anchor nodes $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$, assume u is covered by an anchor node, denoted as a^*, then the shortest path distance between them $d(u,a^*) \le c$. Without loss of generality, we assume $d(u,a^*) < d(a^*,v)$. Note that the following error bound still holds if $d(u,a^*) > d(a^*,v)$. The error of the estimated shortest path distance between u, v is bounded by

$$\begin{aligned} err(u,v) &= \hat{d}(u,v) - d(u,v) \\ &= min_{a\in\mathcal{A}}\big(d(u,a) + d(a,v)\big) - d(u,v) \\ &\le d(u,a^*) + d(a^*,v) - d(u,v) \\ &\le d(u,a^*) + d(a^*,v) - |d(u,a^*) - d(a^*,v)| \\ &= 2d(u,a^*) \le 2c \end{aligned}$$

The error bound holds when either u or v are covered by anchor nodes. When neither u nor v is covered, this error is unbounded. The probability for this case is $(1-CR)^2$. Therefore, the probability that the error of our estimated distance is bounded is $1-(1-CR)^2$. □

4.1.3 Euclidean Projection. While anchor-based encoding enables the representation of spatial positions for nodes in a graph, it is not directly applicable for positional embeddings in LLMs. This is because shortest path distances in graph space do not correspond to distances in Euclidean space, potentially distorting actual spatial relationships. Next, we first elaborate on this argument and then present our solution.

Mismatch between Shortest Path Distance and Euclidean Distance. In LLMs, positional embeddings reflect the linear order of tokens, where proximity in the sequence corresponds to closeness in the embedding space, adhering to Euclidean-like assumptions. This enables the model to capture local relationships: tokens near each other in the input sequence are also close in the learned embedding space, preserving context and meaning. However, as demonstrated in Figure 2, when nodes 2 and 4 are set as anchor nodes, the shortest path distances between nodes 2 and 3, as well as between nodes 1

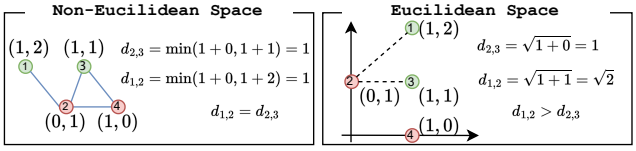

Figure 2: A toy example illustrating the discrepancy between relative distance encoding in non-Euclidean graph space and the required Euclidean space for LLM positional embeddings.

and 2, are both 1 in the graph's non-Euclidean space. In contrast, the corresponding Euclidean distances would be 1 and $\sqrt{2}$, respectively. This discrepancy in relative distances between node pairs leads to a mismatch between shortest path and Euclidean distances.

To address this issue, we propose a pretraining approach that maps the distance encoding from non-Euclidean to Euclidean space, aiming to preserve geometric relationships between nodes. The necessity of this mapping is further justified through ablation studies in Section 5.6. The pretraining process involves a learnable function $\phi : \mathbb{R}^K \to \mathbb{R}^N$ that projects the anchor-based encoding into Euclidean space:

$$\mathbf{e}_v = \phi(\hat{\mathbf{d}}_v) \in \mathbb{R}^N \tag{9}$$

where $\mathbf{e}_v$ represents the transformed node embedding for node v.

To preserve geometric relationships among nodes in the embedding space, we propose a rank-preserving training objective based on maximum likelihood estimation. The objective is to maximize the posterior probability $p(\Phi| >)$, where Φ denotes the parameters of the mapping function ϕ, and $>$ represents the desired order of distances. Assuming independence for the ordering of each pair of distances, we formulate the likelihood function as:

$$p(> |\Phi) = \prod_{(u,v),(i,j)\in\mathcal{E}} p\left(\hat{d}_\phi(u,v) > \hat{d}_\phi(i,j)|\Phi\right)^{\mathbb{I}(\hat{d}(u,v)>\hat{d}(i,j))} \cdot \left(1 - p\left(\hat{d}_\phi(u,v) > \hat{d}_\phi(i,j)|\Phi\right)\right)^{\mathbb{I}(\hat{d}(u,v)\leq\hat{d}(i,j))} \tag{10}$$

where $\hat{d}(u,v)$ denotes the estimated distance between nodes u and v, and $\hat{d}_\phi(u,v)$ represents the Euclidean distance between their corresponding mapped embeddings $\mathbf{e}_u$ and $\mathbf{e}_v$. We can model the probability of one distance being greater than another using the logistic function σ:

$$p\left(\hat{d}_\phi(u,v) > \hat{d}_\phi(i,j)|\Phi\right) := \sigma(\hat{x}_{u,v,i,j}(\Phi)), \tag{11}$$

where $\hat{x}_{u,v,i,j}(\Phi)$ denotes the difference between the Euclidean distances of the two pairs of mapped embeddings.

By maximizing the log-posterior, which is equivalent to minimizing the negative log-likelihood function, we derive the rank-preserving training objective:

$$\min_{\Phi} \mathcal{L} = -\sum_{(u,v),(i,j)\in\mathcal{E}} \mathbb{I}(\hat{d}(u,v) > \hat{d}(i,j)) \ln\sigma(\hat{x}_{u,v,i,j}(\Phi)) + \mathbb{I}(\hat{d}(u,v) \leq \hat{d}(i,j)) \ln(1-\sigma(\hat{x}_{u,v,i,j}(\Phi))) \tag{12}$$

This objective function encourages the ranking of distances between nodes in the embedding space to align with the ranking of their corresponding shortest path distances in the graph.

To facilitate practical implementation, we reformulate the objective as a binary cross-entropy (BCE) loss:

$$\min_{\Phi} \mathcal{L} = \sum_{(u,v),(i,j)\in\mathcal{E}} \text{BCE}\left(\sigma\left(\|\mathbf{e}_u - \mathbf{e}_v\|_2 - \|\mathbf{e}_i - \mathbf{e}_j\|_2\right), y\right), \tag{13}$$

where y captures the relative ordering of distances:

$$y = \mathbb{I}(\hat{d}(u,v) > \hat{d}(i,j)) = \begin{cases} 1, & \text{if } \hat{d}(u,v) > \hat{d}(i,j), \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

This pretraining approach ensures that the positional embeddings derived from graph structures are compatible with the Euclidean assumptions of LLM architectures while preserving the essential spatial relationships between nodes.

4.1.4 Time Complexity Analysis. The time complexity of the greedy algorithm for anchor node selection can be analyzed in two parts:

Initialization. Each node performs a BFS to construct its c-hop neighborhood, requiring $O(|\mathcal{V}|\cdot|\mathcal{E}|)$ time, where $|\mathcal{V}|$ is the number of nodes and $|\mathcal{E}|$ is the number of edges in the graph. The c-hop neighborhoods are stored for each node.

Anchor Selection. In each iteration, the algorithm selects an anchor and updates the coverage for remaining nodes. The worst-case time complexity for this part is $O(|\mathcal{V}|^2)$. This is because:

1. Selecting an anchor requires examining all uncovered nodes in each candidate's c-hop neighborhood ($O(|\mathcal{V}|)$ in the worst case).

2. After selecting an anchor, the algorithm must update the uncovered node counts for all other nodes' c-hop neighborhoods that overlap with the newly covered area ($O(|\mathcal{V}|)$ nodes to update, each potentially affecting $O(|\mathcal{V}|)$ other neighborhoods).

The total time complexity is thus $O(|\mathcal{V}|\cdot|\mathcal{E}| + |\mathcal{V}|^2$.

4.2 Task-Specific LLM Tuning

We now focus on adapting LLMs to leverage graph-based knowledge for specific downstream tasks. Our approach integrates prompt tuning with Low-Rank Adaptation (LoRA) for efficient and effective task-specific fine-tuning.

4.2.1 Prompt Tuning. We employ prompt tuning to incorporate pretrained graph-based knowledge into the LLM. This technique introduces a small, trainable adapter layer that transforms our pretrained anchor-based node embeddings to soft prompts. These soft prompts serve as a learned prefix to the input, guiding the model's attention and output generation.

The generation process, including our prompt tuning adapter, can be formally expressed as:

$$p_{\theta,\Phi}(Y|G,q) = \prod_{i=1}^{r} p(y_i|y_{<i}, [\mathbf{e}_G; \mathbf{e}_T; \mathbf{e}_q]), \tag{15}$$

where θ denotes the frozen LLM parameters, Φ represents the trainable parameters of the prompt tuning adapters, $\mathbf{e}_G$ is the pretrained positional encoding derived from the graph structure, $\mathbf{e}_T$ is the textual embeddings, and $\mathbf{e}_q$ represents the question designed for corresponding graph tasks. The prompt tuning adapter is a shallow neural network that maps the input embeddings to a sequence of

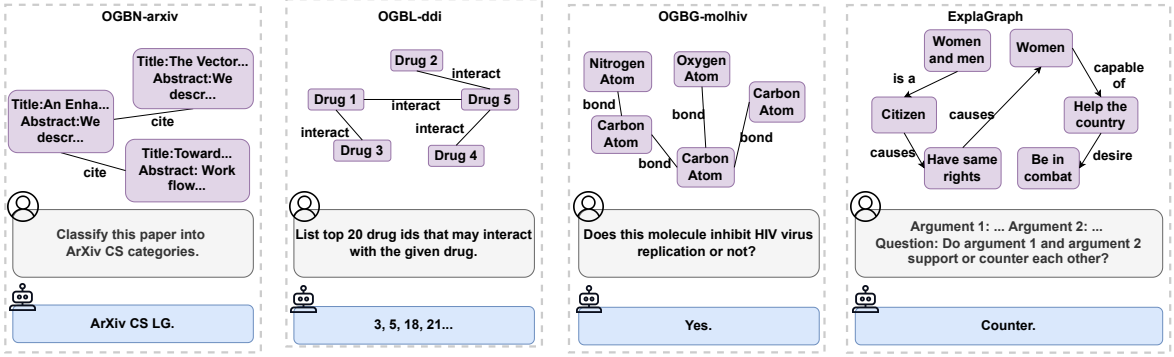

Figure 3: Illustration of dataset characteristics and LLM-based processing workflow for diverse graph-related tasks employed in our experimental setup.

continuous prompt tokens. These tokens are prepended to the input sequence before being processed by the LLM.

4.2.2 Low-Rank Adaptation (LoRA). To further enhance the LLMs' adaptability to graph-structure data, we implement Low-Rank Adaptation (LoRA) [21] in conjunction with prompt tuning. LoRA modifies the weight update mechanism of the LLM by introducing low-rank decomposition, allowing for efficient fine-tuning of the model. For each weight matrix $W \in \mathbb{R}^{dim_1 \times dim_2}$ in the LLM, we introduce a low-rank update:

$$\mathbf{W}' = \mathbf{W} + \mathbf{BA}, \tag{16}$$

where $\mathbf{B} \in \mathbb{R}^{dim_1 \times r}$ and $\mathbf{A} \in \mathbb{R}^{r \times dim_2}$ are low-rank matrices with rank $r \ll \min(dim_1, dim_2)$. This decomposition significantly reduces the number of trainable parameters, as r is typically much smaller than dim_1 and dim_2.

During the training process, only $\mathbf{A}$ and $\mathbf{B}$ are updated while the original weights $\mathbf{W}$ remain frozen. The update rule for the LoRA parameters can be expressed as:

$$\mathbf{A}_{t+1} = \mathbf{A}_t - \eta \nabla_{\mathbf{A}} \mathcal{L}(\theta, \mathbf{A}_t, \mathbf{B}_t), \tag{17}$$

$$\mathbf{B}_{t+1} = \mathbf{B}_t - \eta \nabla_{\mathbf{B}} \mathcal{L}(\theta, \mathbf{A}_t, \mathbf{B}_t), \tag{18}$$

where η is the learning rate, $\mathcal{L}$ is the task-specific loss function, and t denotes the training iteration.

The combination of prompt tuning and LoRA in our approach enables the model to effectively incorporate graph-structural knowledge while adapting to various downstream tasks.

5 Experiments

We conduct extensive experiments to demonstrate the effectiveness of our NT-LLM by investigating the following research questions:

- **RQ1**: Can NT-LLM outperform state-of-the-art methods in various graph-related tasks?
- **RQ2**: What does node position encoding learn? Does it capture the spatial information as intended?
- **RQ3**: How do different anchor selection strategies influence the performance of NT-LLM?
- **RQ4**: What influence do different design choices have on NT-LLM?
- **RQ5**: How does our tokenizer compare in efficiency to conventional message-passing GNNs and graph transformers?

5.1 Experimental Settings

5.1.1 Datasets. We evaluate our approach on diverse graph-based tasks using benchmark datasets from Cora [55], the Open Graph Benchmark (OGB) [22], and ExplaGraphs [54]. Our experiments cover node classification with Cora and OGBN-arxiv, edge prediction using OGBL-ddi, and graph property prediction employing OGBG-molhiv[2]. Additionally, we assess knowledge graph question answering tasks using the ExplaGraphs dataset. These datasets encompass a wide range of graph structures and task complexities, allowing for a comprehensive evaluation of our method. Table 2 presents key statistics for each dataset, while Figure 3 illustrates their characteristics in detail.[3]

Table 2: Dataset statistics and evaluation metrics. For OGBG-molhiv and ExplaGraphs, #Nodes and #Edges counts represent averages across all graphs in the dataset.

Dataset	#Nodes	#Edges	#Graphs	Metric
Cora	2,708	10,556	1	Accuracy
OGBN-arxiv	169,343	1,166,243	1	Accuracy
OGBL-ddi	4,267	1,334,889	1	Hits@20
OGBG-molhiv	25.5	27.5	41,127	ROC-AUC
ExplaGraphs	5.17	4.25	2,766	Accuracy

5.1.2 Baselines. We evaluate our proposed method against various baselines, including both traditional graph learning approaches and LLM-based methods:

- **GNN-based methods**: We incorporate widely-adopted GNN architectures, including Graph Convolutional Networks (GCN) [31], Graph Attention Networks (GAT) [60], and GraphSAGE [19]. Besides, we also evaluate two graph transformer models: GraphFormers [65] and Heterformer [28].
- **LLM-only methods**: We consider approaches that process graph information directly as textual sequences using LLMs. This category includes implementations utilizing zero-shot inference, prompt tuning [32], and Low-Rank Adaptation (LoRA) [21].

[2]For OGBG-molhiv, we use the SMILES strings representing molecules as textual attributes, which are not directly provided by OGB.
[3]Cora, being a similar citation network to OGBN-arxiv, was omitted from Figure 3 to avoid redundancy.

Table 3: Main results on benchmark datasets. The best performance is highlighted in bold and the second best is underlined. Δ_{prompt} and Δ_{LoRA} represent the improvements over LLM prompt tuning and LoRA baselines, respectively. * indicates the statistically significant improvements (*i.e.,* two-sided t-test with p<0.05) over the compared baseline.

Method	Cora (Accuracy↑)	OGBN-arxiv (Accuracy↑)	OGBL-ddi (Hits@20↑)	OGBG-molhiv (ROC-AUC↑)	ExplaGraphs (Accuracy↑)
GCN [31]	0.8147	0.7360	0.3707	0.7606	-
GAT [60]	0.8352	0.7366	0.4133	0.7520	-
GraphSAGE [19]	0.8265	0.7295	0.5390	0.7558	-
GraphFormers [65]	0.8910	0.7431	0.5538	0.7414	-
Heterformer [28]	0.8761	0.7390	0.5482	0.7505	-
zero-shot	0.6490	0.5406	0.3384	0.6321	0.6679
prompt tuning [32]	0.7903	0.6971	0.3592	0.6554	0.8224
LoRA [21]	0.8194	0.7323	0.3918	0.7529	0.9296
GraphGPT [57]	0.9085	0.7637	0.5011	<u>0.7851</u>	0.9052
GraphTranslator [67]	0.9351	<u>0.7748</u>	0.5425	0.7764	0.9273
G-Retriever [20]	0.9148	0.7521	0.4573	0.6920	0.9231
G-Retriever LoRA	0.9350	0.7580	0.5296	0.7635	0.9240
GRAG [23]	0.9296	0.7492	0.4617	0.6698	0.9242
GRAG LoRA	0.9473	0.7554	0.5386	0.7309	<u>0.9422</u>
NT-LLM	<u>0.9478</u>	0.7525	<u>0.5904</u>	0.7531	0.9332
Δ_{prompt}	↑ 19.93%*	↑ 7.95%*	↑ 74.47%*	↑ 14.91%*	↑ 13.47%*
NT-LLM LoRA	**0.9531**	**0.7752**	**0.6375**	**0.8045**	**0.9603**
Δ_{LoRA}	↑ 16.32%*	↑ 3.02%*	↑ 62.71%*	↑ 6.85%*	↑ 3.30%*

- **GNN-LLM hybrid methods**: We compare our approach with state-of-the-art methods that integrate GNNs and LLMs. Specifically, we include GraphGPT [57] and GraphTranslator [67], which focus on text-attributed graph representation learning with language models. Additionally, we compare our method with G-Retriever [20] and GRAG [23], which are Graph Retrieval-Augmented Generation (RAG) methods that combine GNNs and LLMs for graph-based text generation tasks.

5.2 Implementation Details

We implement all models and experiments using PyTorch [47], PyTorch Geometric [16], and the HuggingFace Transformers [64] libraries. All experiments are conducted on two NVIDIA RTX 6000 Ada GPUs, each with 48GB memory.

5.2.1 Text and LLM Components. For encoding textual attributes, we employ SentenceBERT [51]. The LLM component of all experiments is based on the pretrained LLaMA3-8B [59]. We use LLaMA3-8B in zero-shot (no fine-tuning), as well as in prompt-tuning and LoRA-based fine-tuning settings. During LLM fine-tuning with LoRA, we set the low-rank dimension to 8 and the scaling factor to 16. Optimization uses AdamW [42] with a learning rate of 1e-4 and weight decay of 0.05. Fine-tuning runs for a maximum of 10 epochs with an early stopping patience of 3. The batch size is set to 32 for OGBN-arxiv and OGBL-ddi, and to 2 for OGBG-molhiv and ExplaGraphs, according to dataset size.

5.2.2 GNN-based Methods. Our baseline and hybrid GNN models use a 4-layer architecture with hidden dimensions of 256, ReLU activation, and a dropout rate of 0.5. Graph transformer baselines utilize nested GAT architecture combined with transformer layers, where each node uses 5 uniformly sampled neighbors as context. Training runs using the AdamW optimizer for 500 epochs with an early stopping patience of 10, learning rate of 1e-3 and weight decay of 5e-4.

5.2.3 NT-LLM Implementation. In the node tokenizing stage, we set the anchor identification parameters as $c = 1$ and $CR = 0.7$, and map node encodings via a 3-layer MLP. In the LLM fine-tuning stage, we following the settings in 5.2.1.

5.2.4 GNN-LLM Hybrid Baselines. For GNN-LLM hybrid methods, we combine a 4-layer GAT with LLaMA3-8B, following the architecture and hyperparameter settings as described in their papers.

5.3 Main Results (RQ1)

Table 3 compares the performance of our proposed NT-LLM method against baselines on five benchmark datasets on the corresponding task, respectively.[4] We have the following key findings:

- **NT-LLM consistently outperforms all baseline methods across various tasks and datasets.** This observation justifies the superiority of NT-LLM and demonstrates its effectiveness and broad applicability in graph learning.
- **NT-LLM effectively addresses the challenge of enabling LLMs to understand graph structures**. In other words, NT-LLM leverages the strengths of LLMs in understanding textual attributes while benefiting from our proposed node position encoding to capture the graph topology. First, NT-LLM outperforms pure LLM and GNN baselines on all datasets. This observation demonstrates that understanding textual attributes and topology are equally important for graph learning tasks. Second, when fine-tuning NT-LLM with LoRA (fine-tuned NT-LLM), its performance surpasses LLM-GNN hybrid approaches. This suggests that NT-LLM is more effective at enabling LLMs to understand

[4]GNNs are unable to perform complex graph reasoning tasks in the ExplaGraphs dataset, thus the corresponding cells are marked with -.

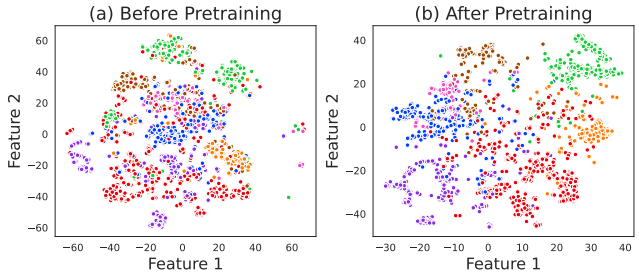

Figure 4: Embeddings visualized before and after the transformation in pretraining on Cora dataset. The colors represent the ground truth labels of nodes.

graph structures compared to intermediate solutions, *i.e.,* LLM-GNN hybrid approaches.

- **The superiority of NT-LLM in graph understanding comes from our proposed node position encoding**. In particular, the OGBL-ddi dataset lacks textual attributes. As we can see, LLM methods perform worse than GNN baseline methods, which highlights their limitations in capturing topological information from graph data. Unlike LLM methods, our proposed NT-LLM, despite not using GNNs, outperforms all baselines with over 60% improvement compared to LLM methods, demonstrating its ability to effectively encode graph structure.

In conclusion, NT-LLM shows superior performance and adaptability across various graph-related tasks and datasets. The improvements over state-of-the-art baselines, even in the absence of textual attributes, highlight the effectiveness of our proposed method in capturing both textual and structural information.

5.4 Understanding Node Position Encoding (RQ2)

To understand what node position encoding learns, in this section, we provide visualization for the learned node position embedding on the Cora dataset to gain further insights. We select this dataset because, in Cora, nodes from the same class tend to be naturally closer in the graph structure. This property allows us to directly evaluate the quality of the node position embeddings by observing how well they align with the class labels.

Figure 4 illustrates the embeddings before and after the transformation in positional embedding pretraining, shown against class labels. Prior to the transformation, nodes belonging to the same class can be separated distantly in the embedding space. However, after applying the transformation, these nodes are effectively projected into the same region, highlighting the efficacy of our pretraining approach in capturing the underlying semantic relationships among nodes. For instance, the green dots, which are dispersed before the transformation, become densely clustered afterward.

5.5 Anchor Selection Strategies Impact (RQ3)

Since anchor nodes offer a comprehensive view of the graph structure, different strategies for identifying anchor nodes may impact NT-LLM's ability to comprehend the graph. In this section, we conduct an extensive evaluation of various anchor selection strategies, on three datasets, *i.e.,* Cora, OGBN-arxiv and OGBL-ddi, using a fixed seed and the NT-LLM architecture. Subsequently, the positional embeddings are pretrained following the same procedure outlined in Section 4.1.3.

Table 4: Comparison of anchor selection strategies across three datasets. The highest performance for each dataset is shown in bold.

Strategy	Cora	OGBN-arxiv	OGBL-ddi
Degree	0.9172	0.7312	0.5731
Random	0.8891	0.6783	0.5019
Closeness [3]	0.8931	0.6392	0.4852
Eigenvector [6]	0.8424	0.6105	0.4736
PageRank [46]	0.8703	0.6641	0.5127
Betweenness [17]	0.8539	0.6428	0.4967
HPLC [29]	0.9174	0.7411	0.5613
Ours	**0.9478**	**0.7525**	**0.5904**

Table 4 presents the experimental results. Our method achieves the best performance among all evaluated strategies, surpassing traditional centrality-based approaches (such as Degree and PageRank [46]), random selection, and the landmark-based HPLC [29].

To provide a clearer insight into the advantage of our anchor selection strategy, we compare the anchor nodes selected by different strategies on Cora dataset in Figure 5. The anchor nodes selected by our method are more evenly distributed across the graph structure. In contrast, methods such as Degree, HPLC, Closeness, PageRank, and Eigenvector focus on selecting "important" nodes but fail to provide broad coverage, particularly of nodes located at considerable distances from the graph's central area.

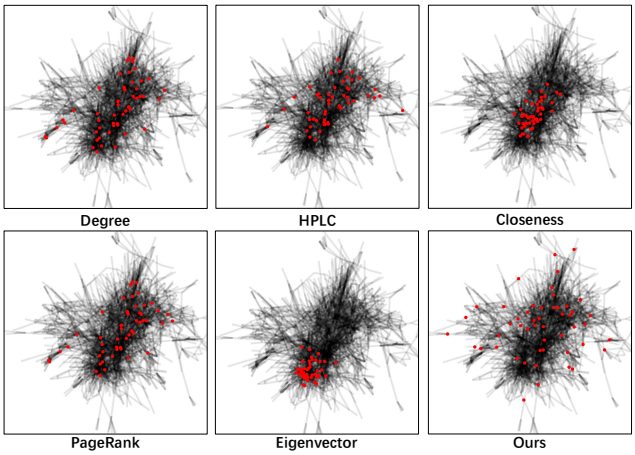

Figure 5: Distribution of anchor nodes (marked in red) selected by different strategies on the Cora dataset. Our method achieves a more even distribution, effectively covering the peripheral regions of the graph.

5.6 Ablation Studies (RQ4)

In this section, we conduct extensive ablation studies to investigate the effectiveness of each component in NT-LLM, and justify our model design choices.

Table 5: Performance comparison of NT-LLM variants across four datasets. Best results for each dataset are in bold.

Variant	Cora	arxiv	ddi	molhiv	ExplaGraphs
NT-LLM	**0.9478**	**0.7525**	**0.5904**	**0.7531**	**0.9332**
w/o PE	0.8070	0.6971	0.3592	0.6554	0.8224
w/o Pre	0.8195	0.6538	0.3791	0.6419	0.7671
w/o PT	0.7864	0.5904	0.3460	0.5834	0.7024

5.6.1 Impact of Model Components. NT-LLM has specific design features, including the node position encoding, its corresponding pretraining task, and two different strategies for LLMs to leverage node position encoding, *i.e.*, prompt tuning and low-rank adaptation. We evaluate the performance of each variant of our model on five datasets as follows:

- **w/o PE**: The NT-LLM without positional encoding, using raw node features as input to the LLM.
- **w/o Pre**: The NT-LLM without the distance transformation pretraining module, using concrete anchor-based distances as node position embeddings.
- **w/o PT**: The NT-LLM without the prompt tuning module, directly inputting all embeddings into the LLM.

Table 5 presents the results of the ablation study, which evaluates the impact of removing individual components from the proposed method. The observed performance drop across all datasets confirms the importance and complementary nature of each component within the method. In particular, we observe that node position encoding pretraining is critical for NT-LLM. The variant without pretraining (w/o Pre) experiences a significant performance drop when the pretraining module is removed, supporting our argument in Section 4.1.3. This is due to the mismatch between shortest path and Euclidean distances, which distorts actual spatial relationships. Therefore, positional embedding pretraining is an indispensable component of NT-LLM.

5.6.2 Impact of Hyperparameters. We investigate the impact of two key hyperparameters in NT-LLM: the coverage radius c and the coverage ratio CR. Figure 6 presents the relationships between these hyperparameters, model accuracy and the number of anchor nodes. The results demonstrate that smaller values of c and larger values of CR generally lead to a better performance. This trend aligns with the error bound established in Lemma 4.1. Notably, we observed that the number of anchor nodes increases exponentially as c decreases and CR increases. This relationship underscores the importance of carefully selecting these hyperparameters to balance computational complexity and model performance.

5.7 Tokenizer Efficiency (RQ5)

The only trainable component in our proposed graph tokenizer is a simple MLP, making it intuitively much more efficient than conventional message-passing GNNs or graph transformers. To validate this, we compare the efficiency of various graph tokenizers, including our own, across multiple datasets. The results are summarized in Table 6. The consistently lower number of trainable parameters and training time demonstrate the efficiency of our tokenizer.

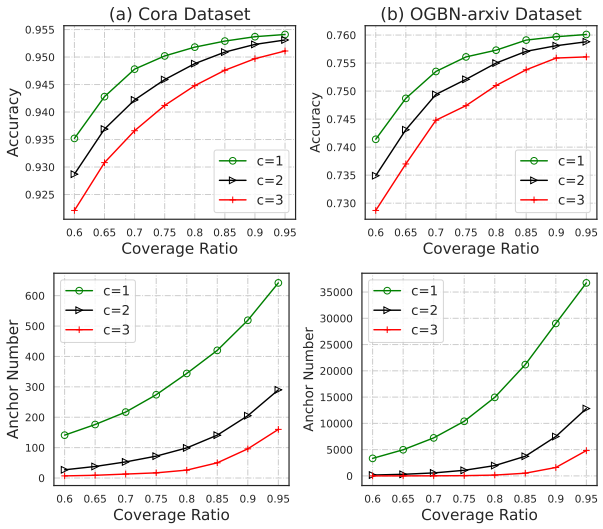

Figure 6: Effects of coverage radius (c) and coverage ratio (CR) on model accuracy and the number of anchor nodes for the Cora and OGBN-arxiv datasets. The top row shows the impact on model accuracy, while the bottom row illustrates the changes in the number of anchor nodes as c and CR vary.

Table 6: Comparison of different tokenization methods based on the number of trainable parameters and training time.

Dataset	Tokenizer	Trainable Parameters	Training Time
Cora	GAT	1.4M	2min
	GraphFormer	3.6M	10min
	NT-LLM	0.3M	<1min
OGBN-arxiv	GAT	21.7M	4h
	GraphFormers	49.2M	6h
	NT-LLM	0.7M	10min
OGBL-ddi	GAT	2.6M	3min
	GraphFormers	6.1M	13min
	NT-LLM	0.5M	1min

6 Conlusion

In the paper, we propose NT-LLM, an anchor-based graph positional encoding approach that enables efficient graph tokenization for LLMs. Our method preserves crucial structural information through anchor nodes selection without requiring extensive textual descriptions or complex GNNs. Evaluations across diverse benchmarks demonstrate significant improvements across diverse tasks from node classification to complex reasoning, confirming the effectiveness and efficiency of our proposed NT-LLM method.

Acknowledgments

This paper is supported by NSFC (No. 62176155), Shanghai Municipal Science and Technology Major Project, China, under grant No. 2021SHZDZX0102.

GenAI Usage Disclosure

This work utilizes Copilot to accelerate coding processes through intelligent code suggestions, auto-completion, and boilerplate generation, while Large Language Models (LLMs) are employed exclusively for spelling and grammar checks in paper writing. All AI-generated content has undergone thorough human review.

References

[1] Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. 2021. The Surprising Power of Graph Neural Networks with Random Node Initialization. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. International Joint Conferences on Artificial Intelligence Organization.

[2] Arkadeep Acharya, Brijraj Singh, and Naoyuki Onoe. 2023. LLM Based Generation of Item-Description for Recommendation System. In *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023, Singapore, Singapore, September 18-22, 2023*. ACM.

[3] Alex Bavelas. 1948. A mathematical model for group structures. *Human organization* 7, 3 (1948), 16–30.

[4] Dominique Beaini, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Lió. 2021. Directional Graph Networks. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR.

[5] Dominique Beaini, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Lió. 2021. Directional Graph Networks. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 748–758.

[6] Mikhail Belkin and Partha Niyogi. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering *(NIPS'01)*. MIT Press.

[7] Mikhail Belkin and Partha Niyogi. 2003. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation* (2003).

[8] Sudhanshu Chanpuriya and Cameron Musco. 2020. InfiniteWalk: Deep Network Embeddings as Laplacian Embeddings with a Nonlinearity. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020).

[9] Runjin Chen, Tong Zhao, Ajay Kumar Jaiswal, Neil Shah, and Zhangyang Wang. 2024. LLaGA: Large Language and Graph Assistant. In *Proceedings of the 41st International Conference on Machine Learning*. PMLR.

[10] Sitao Cheng, Ziyuan Zhuang, and et al. 2024. Call Me When Necessary: LLMs can Efficiently and Faithfully Reason over Structured Environments. In *Findings of the Association for Computational Linguistics ACL 2024*. Association for Computational Linguistics.

[11] Jeroen den Boef, Joran Cornelisse, and Paul Groth. 2021. GraphPOPE: Retaining Structural Graph Information Using Position-aware Node Embeddings. In *DL4KG@ISWC*.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota.

[13] Yuhui Ding, Antonio Orvieto, Bobby He, and Thomas Hofmann. 2024. Recurrent Distance Filtering for Graph Representation Learning. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research)*. PMLR, 11002–11015.

[14] Moshe Eliasof, Fabrizio Frasca, Beatrice Bevilacqua, Eran Treister, Ga Chechik, and Haggai Maron. 2023. Graph positional encoding via random feature propagation. In *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*.

[15] 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*.

[16] Matthias Fey and Jan Eric Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. *ArXiv* abs/1903.02428 (2019).

[17] Linton C. Freeman. 1978. Centrality in social networks conceptual clarification. *Social Networks* 1 (1978), 215–239.

[18] Jiayan Guo, Lun Du, and Hengyu Liu. 2023. GPT4Graph: Can Large Language Models Understand Graph Structured Data ? An Empirical Evaluation and Benchmarking. *CoRR* abs/2305.15066 (2023).

[19] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*.

[20] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering. arXiv:2402.07630 [cs.LG] https://arxiv.org/abs/2402.07630

[21] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

[22] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2021. Open Graph Benchmark: Datasets for Machine Learning on Graphs. arXiv:2005.00687 [cs.LG] https://arxiv.org/abs/2005.00687

[23] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2024. GRAG: Graph Retrieval-Augmented Generation. *arXiv* abs/2405.16506 (2024).

[24] Yuntong Hu, Zheng Zhang, and Liang Zhao. 2023. Beyond Text: A Deep Dive into Large Language Models' Ability on Understanding Graph Data. In *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*.

[25] Chao Huang, Xubin Ren, Jiabin Tang, Dawei Yin, and Nitesh Chawla. 2024. Large Language Models for Graphs: Progresses and Directions. In *Companion Proceedings of the ACM on Web Conference 2024*. 1284–1287.

[26] Xuanwen Huang, Kaiqiao Han, Yang Yang, Dezheng Bao, Quanjin Tao, Ziwei Chai, and Qi Zhu. 2024. Can GNN be Good Adapter for LLMs?. In *Proceedings of the ACM Web Conference 2024 (WWW '24)*. Association for Computing Machinery.

[27] Tomoki Ito and Shun Nakagawa. 2024. Tender Document Analyzer with the Combination of Supervised Learning and LLM-based Improver. In *Companion Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, Singapore, May 13-17, 2024*. ACM.

[28] Bowen Jin, Yu Zhang, Qi Zhu, and Jiawei Han. 2023. Heterformer: Transformer-based Deep Node Representation Learning on Heterogeneous Text-Rich Networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Long Beach, CA, USA) *(KDD '23)*. Association for Computing Machinery, New York, NY, USA, 1020–1031. https://doi.org/10.1145/3580305.3599376

[29] Minsang Kim and Seung Baek. 2024. Hierarchical Position Embedding of Graphs with Landmarks and Clustering for Link Prediction. In *Proceedings of the ACM Web Conference 2024 (WWW '24)*.

[30] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

[31] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

[32] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics.

[33] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance encoding: design provably more powerful neural networks for graph representation learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20)*.

[34] Yichuan Li, Kaize Ding, and Kyumin Lee. 2023. GRENADE: Graph-Centric Language Model for Self-Supervised Representation Learning on Text-Attributed Graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, Singapore.

[35] Yuhan Li, Peisong Wang, Xiao Zhu, Aochuan Chen, Haiyun Jiang, Deng Cai, Wai Kin (Victor) Chan, and Jia Li. 2024. GLBench: A Comprehensive Benchmark for Graph with Large Language Models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (Eds.).

[36] Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. 2024. FlexKBQA: A Flexible LLM-Powered Framework for Few-Shot Knowledge Base Question Answering. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*. AAAI Press.

[37] Xinyu Lin, Wenjie Wang, Yongqi Li, Shuo Yang, Fuli Feng, Yinwei Wei, and Tat-Seng Chua. 2024. Data-efficient Fine-tuning for LLM-based Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(SIGIR '24)*.

[38] Guangyi Liu, Yongqi Zhang, Yong Li, and Quanming Yao. 2025. Dual Reasoning: A GNN-LLM Collaborative Framework for Knowledge Graph Question Answering. arXiv:2406.01145 [cs.CL] https://arxiv.org/abs/2406.01145

[39] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024. Improved Baselines with Visual Instruction Tuning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 26286–26296. https://doi.org/10.1109/CVPR52733.

2024.02484
[40] Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S. Yu, and Chuan Shi. 2025. Graph Foundation Models: Concepts, Opportunities and Challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 47, 6 (2025), 5023–5044. https://doi.org/10.1109/TPAMI.2025.3548729
[41] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692 [cs.CL]
[42] Ilya Loshchilov and Frank Hutter. 2017. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.
[43] Linhao Luo, Yuan-Fang Li, Reza Haf, and Shirui Pan. 2024. Reasoning on Graphs: Faithful and Interpretable Large Language Model Reasoning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.
[44] Zheng Ma, Junyu Xuan, Yu Guang Wang, Ming Li, and Pietro Liò. 2020. Path integral based convolution and pooling for graph neural networks *(NIPS '20)*.
[45] OpenAI. 2024. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL] https://arxiv.org/abs/2303.08774
[46] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank Citation Ranking : Bringing Order to the Web. In *The Web Conference*.
[47] Adam Paszke and et al Gross. 2019. *PyTorch: an imperative style, high-performance deep learning library*.
[48] Bryan Perozzi, Rami Al-Rfou, and Steven S. Skiena. 2014. DeepWalk: online learning of social representations. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014).
[49] Miao Qiao, Hong Cheng, and Jeffrey Xu Yu. 2011. Querying shortest path distance with bounded errors in large graphs. In *Proceedings of the 23rd International Conference on Scientific and Statistical Database Management (SSDBM'11)*.
[50] Hua Xuan Qin, Shan Jin, Ze Gao, Mingming Fan, and Pan Hui. 2024. CharacterMeet: Supporting Creative Writers' Entire Story Character Construction Processes Through Conversation with LLM-Powered Chatbot Avatars. In *Proceedings of the CHI Conference on Human Factors in Computing Systems, CHI 2024, Honolulu, HI, USA, May 11-16, 2024*. ACM.
[51] 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 and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.
[52] Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh Chawla, and Chao Huang. 2024. A Survey of Large Language Models for Graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Barcelona, Spain) *(KDD '24)*. Association for Computing Machinery, New York, NY, USA, 6616–6626. https://doi.org/10.1145/3637528.3671460
[53] Joshua Robinson and David Wingate. 2023. Leveraging Large Language Models for Multiple Choice Question Answering. In *The Eleventh International Conference on Learning Representations*.
[54] Swarnadeep Saha, Prateek Yadav, Lisa Bauer, and Mohit Bansal. 2021. ExplaGraphs: An Explanation Graph Generation Task for Structured Commonsense Reasoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.
[55] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Magazine* 29, 3 (2008), 93–106.
[56] Yanchao Tan, Zihao Zhou, Hang Lv, Weiming Liu, and Carl Yang. 2024. WalkLM: a uniform language model fine-tuning framework for attributed graph embedding. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS '23)*.
[57] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 491–500.
[58] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Long Xia, Dawei Yin, and Chao Huang. 2024. Higpt: Heterogeneous graph language model. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2842–2853.
[59] Hugo Touvron, Louis Martin, and et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783
[60] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.
[61] Tao Wang, Di Jin, Rui Wang, Dongxiao He, and Yuxiao Huang. 2022. Powerful Graph Convolutional Networks with Adaptive Propagation Mechanism for Homophily and Heterophily. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press.
[62] Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. How Far Can Camels Go? Exploring the State of Instruction Tuning on Open Resources. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 74764–74786.
[63] Zhihao Wen and Yuan Fang. 2023. Augmenting Low-Resource Text Classification with Graph-Grounded Pre-training and Prompting. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23)*. Association for Computing Machinery.
[64] Thomas Wolf, Lysandre Debut, and et al. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *CoRR* abs/1910.03771 (2019). http://arxiv.org/abs/1910.03771
[65] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. 2021. GraphFormers: GNN-nested transformers for representation learning on textual graph. In *Proceedings of the 35th International Conference on Neural Information Processing Systems (NIPS '21)*. Curran Associates Inc., Red Hook, NY, USA, Article 2206, 13 pages.
[66] Jiaxuan You, Rex Ying, and Jure Leskovec. 2019. Position-aware Graph Neural Networks. In *International Conference on Machine Learning*.
[67] Mengmei Zhang, Mingwei Sun, Peng Wang, Shen Fan, Yanhu Mo, Xiaoxiao Xu, Hong Liu, Cheng Yang, and Chuan Shi. 2024. GraphTranslator: Aligning Graph Model to Large Language Model for Open-ended Tasks. In *The Web Conference 2024*.
[68] Yilun Zhao, Yitao Long, Hongjun Liu, Ryo Kamoi, Linyong Nan, Lyuhao Chen, Yixin Liu, Xiangru Tang, Rui Zhang, and Arman Cohan. 2024. DocMath-Eval: Evaluating Math Reasoning Capabilities of LLMs in Understanding Financial Documents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*. Association for Computational Linguistics.
[69] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. 2017. Scalable graph embedding for asymmetric proximity. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI'17)*.
[70] Yun Zhu, Yaoke Wang, Haizhou Shi, and Siliang Tang. 2024. Efficient Tuning and Inference for Large Language Models on Textual Graphs. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*.