Tokenphormer: Structure-aware Multi-token Graph Transformer for Node Classification

Zijie Zhou[1*], Zhaoqi Lu[1, 2, 3*], Xuekai Wei[1], Rongqin Chen[1], Shenghui Zhang[1], Pak Lon Ip [1], Leong Hou U[1†]

[1]University of Macau
[2]The Hong Kong Polytechnic University
[3]Guangdong Institute of Intelligent Science and Technology, China
{zhou.zijie, lu.zhaoqi}@connect.um.edu.mo; xuekaiwei2-c@my.cityu.edu.hk; {chen.rongqin, zhang.shenghui}@connect.um.edu.mo; {paklonip, ryanlhu}@um.edu.mo

Abstract

Graph Neural Networks (GNNs) are widely used in graph data mining tasks. Traditional GNNs follow a message passing scheme that can effectively utilize local and structural information. However, the phenomena of over-smoothing and over-squashing limit the receptive field in message passing processes. Graph Transformers were introduced to address these issues, achieving a global receptive field but suffering from the noise of irrelevant nodes and loss of structural information. Therefore, drawing inspiration from fine-grained token-based representation learning in Natural Language Processing (NLP), we propose the Structure-aware Multi-token Graph Transformer (Tokenphormer), which generates multiple tokens to effectively capture local and structural information and explore global information at different levels of granularity. Specifically, we first introduce the walk-token generated by mixed walks consisting of four walk types to explore the graph and capture structure and contextual information flexibly. To ensure local and global information coverage, we also introduce the SGPM-token (obtained through the Self-supervised Graph Pre-train Model, SGPM) and the hop-token, extending the length and density limit of the walk-token, respectively. Finally, these expressive tokens are fed into the Transformer model to learn node representations collaboratively. Experimental results demonstrate that the capability of the proposed Tokenphormer can achieve state-of-the-art performance on node classification tasks.

Code — https://github.com/Dodo-D-Caster/Tokenphormer
Appendix — https://doi.org/10.48550/arXiv.2412.15302

1 Introduction

Graph structures are commonly seen in both physical and virtual domains, such as anomaly detection (Deng and Hooi 2021), traffic (Kong, Guo, and Liu 2024), and recommendation (Agrawal et al. 2024). Mining tasks face a significant challenge due to the inherent complexity of various graphs and their intricate features. Graph Neural Networks (GNNs) have emerged as a successful learning framework capable of solving various graph-based tasks. Traditional GNNs follow the message passing scheme (Hamilton, Ying, and Leskovec 2017), aggregating neighboring nodes to update node representation. However, these methods are hindered by issues such as over-smoothing (Chen et al. 2020) and over-squashing (Alon and Yahav 2021).

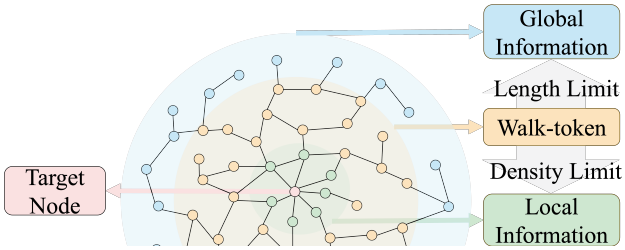

Figure 1: Idea of Tokenphormer.

Transformer (Vaswani et al. 2017) has demonstrated remarkable performance due to its powerful modeling capability, making it a potent architecture for graphs as well. The traditional graph Transformer treats the entire input graph as a fully connected entity, where individual nodes are regarded as independent tokens, and a unified sequence encompassing all nodes is constructed, thus alleviating the limitations of Message Passing Neural Networks (MPNNs). This approach allows the graph Transformer to extend the receptive field to the global, which benefits node representation learning. However, this scheme also results in high computational complexity, introduces noise from irrelevant nodes, and leads to a loss of structural information (Kreuzer et al. 2021). This limitation potentially hinders a comprehensive understanding of the entire graph.

In recent years, researchers have developed token-based methods (Kreuzer et al. 2021; Zhao et al. 2021; Zhang et al. 2022) that generate a sequence for each node , which have shown improved performance compared to the traditional graph Transformer. However, these models encounter difficulties when dealing with various graphs, as they lack the flexibility to thoroughly *explore the complete graph structure* due to the monotonous token design. For instance, NAGphormer (Chen et al. 2023), which serves as a repre-

[*]These authors contributed equally.
[†]Corresponding author.

sentative and has achieved competitive results on homogeneous graphs, utilizes hop information to form tokens. These tokens are coarse-grained since they overlook crucial information, such as relationships among nodes within the same hop and data beyond the designated k-hop neighborhood, limiting the model's performance on diverse graph types like heterogeneous graph.

To address the aforementioned limitations, token generation strategies should be enhanced in two key aspects: 1) **Fine-grained.** Given the diverse and complex nature of graph structures, tokens should be more fine-grained, effectively capturing intricate structures and node relationships. 2) **Comprehensive.** To gain a better understanding of the entire graph, tokens should provide an approximate full coverage, capturing both global and local information for improved expressiveness. A natural question arises: *Can we design token generation strategies that can create multiple effective tokens for graph nodes that substantially improve the ability to investigate varied graph structures?*

To answer this question, motivated by fine-grained token-based representation learning in Natural Language Processing (NLP), we propose *graph serialization*, which transforms the graph structure into *graph document* consisting of *graph sentences* using random walk, to effectively serialize the entire graph as well as node's neighborhood with minimal information loss. Graph serialization is the foundation for designing multiple token generators, enabling a more flexible exploration of diverse graph types.

Figure 1 shows the idea of Tokenphormer. To meet the fine-grained requirement, we utilize *walk-token* to flexibly explore the graph (yellow area in Figure 1). Specifically, the *walk-token* incorporates four distinct walk types, each representing a serialized node's neighborhood with varying receptive fields and walking strategies. These walk types are designed to capture a diverse range of information, making them suitable for various graph structures. The four types are: uniform random walk, non-backtracking random walk, our proposed neighborhood jump walk, and non-backtracking neighborhood jump walk. It should be noted that each walk type is non-irreplaceable, just as a tailored design of data augmentation to address the requirement of fine-grained.

Given the limited receptive field of *walk-token*, we address the comprehensive requirement by introducing two additional token types: *SGPM-token* (blue area in Figure 1) and *Hop-token* (green area in Figure 1), extending the length and density limit of *walk-token*. Specifically, *SGPM-tokens* are obtained from the Self-supervised Graph Pre-train Model (SGPM), which pre-trains on the graph document (serialized entire graph), ensuring the global coverage of the *walk-token*. Moreover, *Hop-tokens* are generated through each hop's message, ensuring the local coverage of the *walk-token*.

Finally, we introduce Tokenphormer (Structure-aware Multi-token Graph Transformer), a node classification approach that leverages the power of Transformer to jointly learn all these token types. Tokenphormer pioneers *the generation of diverse fine-grained and comprehensive tokens for data augmentation, adaptly extracting information across a spectrum of scales, from local to global contexts* and is adaptive to diverse graphs, outperforming other methods on both homogeneous and heterogeneous graphs. The main contributions of this work are as follows:

- We present Tokenphormer, a pioneering node classification approach that generates multiple tokens rich in structural information and varying granularities through diverse walk strategies, to address the limitations of existing graph Transformers.
- We propose an effective way to serialize the graph into a graph document and introduce SGPM to capture global and contextual information from our proposed graph documents in the pre-train period.
- Experimental results demonstrate that Tokenphormer outperforms existing state-of-the-art graph Transformers and mainstream MPNNs on most homogeneous and heterogeneous graph benchmark datasets, showcasing its applicability to diverse graphs.

2 Related Work

2.1 Existing Graph Learning Architectures

GNN methods can be generally categorized nto Message Passing Neural Networks (MPNNs) and graph Transformers. The MPNN approach involves two main steps: aggregation and update. Each node first aggregates features from its neighbors and then updates its representation by combining its features with the aggregated data (Xu et al. 2018; Gilmer et al. 2017). Notable models like GCN (Kipf and Welling 2017), GAT (Veličković et al. 2018), and GraphSAGE (Hamilton, Ying, and Leskovec 2017) have demonstrated strong performance with this method. However, MPNNs face challenges such as over-smoothing (Chen et al. 2020) and over-squashing (Alon and Yahav 2021). To address these issues, methods like DropEdge (Huang et al. 2020), which randomly removes edges during training, and techniques to reduce message passing redundancy (Chen et al. 2022), have been developed.

The Transformer architecture (Vaswani et al. 2017; Schmitt et al. 2021; Dufter, Schmitt, and Schütze 2022; Kong et al. 2023; Dwivedi et al. 2023) has gained popularity in graph representation learning as a solution to these problems. For example, GT (Dwivedi and Bresson 2021) uses Laplacian eigenvectors for positional encoding, while GraphTrans (Wu et al. 2021) and GROVER (Rong et al. 2020) incorporate GNNs as auxiliary modules to capture structural information. Graphormer (Ying et al. 2021) integrates centrality and spatial encoding to add structural inductive bias into the attention mechanism. These methods treat the entire graph as a sequence of node tokens and are of high computation complexity. Recent advancements like Gophormer (Zhao et al. 2021) sample relevant nodes to form token sequences, shifting the Transformer's training focus from the entire graph to node sequences. NAGphormer (Chen et al. 2023) introduces the Hop2Token mechanism, converting neighborhood features from each hop into different token sequences. However, though the complexity is decreased, the simplex design of tokens limits these models' performance on diverse graphs.

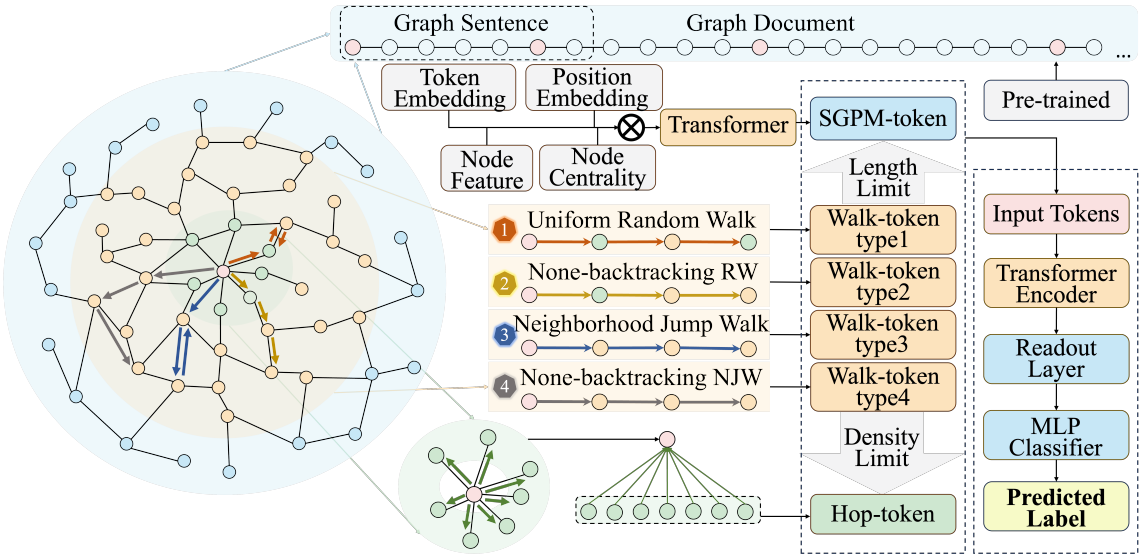

Figure 2: Framework. RW refers to random walk while NJW stands for neighborhood jump walk. Tokenphormer generates diverse tokens with different levels of granularity for the target node (red node), respectively *walk-tokens* (yellow area), *SGPM-token* (blue area) and *hop-token* (green area), comprehensively mining essential information from the whole graph. Then, all these tokens are constructed into a sequence and fed into the Transformer-based backbone to jointly learn the final node representation. Finally, an MLP-based module is employed for node classification tasks.

2.2 Token-based Representations

NLP has advanced significantly due to deep learning and large language models (LLMs) like GPT (Radford et al. 2018). These improvements hinge on sophisticated token representations that capture semantic and syntactic nuances, enabling applications like sentiment analysis and machine translation. Techniques like Word2Vec (Mikolov et al. 2013) and GloVe (Pennington, Socher, and Manning 2014) use neural networks to place words in continuous vector spaces, enhancing contextual understanding. Models like BERT (Devlin et al. 2019) and GPT (Radford et al. 2018) further improve this with contextualized embeddings that consider surrounding words, leading to nuanced semantic comprehension.

In pursuing advanced token representations, there is a shift towards enhanced tokens within graph nodes (Zhang et al. 2020). Traditional graph Transformers (Yun et al. 2019; Ying et al. 2021) use independent nodes as tokens, resulting in high computational complexity and increased noise. Node2Sequence methods like Gophormer (Zhao et al. 2021) sample ego-graphs and add global nodes to form sequences but still overlook edge information, limiting node representation. The Hop2Token method proposed by NAGphormer (Chen et al. 2023) aggregates each hop into a token but remains coarse-grained. To make the input tokens of the graph Transformers more fine-grained, inspired by subword and character-level tokenization (Mikolov et al. 2013), generating multiple tokens for nodes can better capture graph structure details, nuances, and semantic intricacies, aligning with the trend of comprehensive token representations to enhance graph models.

3 The Proposed Method

3.1 Notations and Problem Formulation

Consider $G = (V, E)$ be an unweighted graph with node set V and edge set E, where $V = \{v_1, v_2, \cdots, v_n\}$ and $n = |V|$. Each node's feature vector $x_v \in X$, where $X \in \mathbb{R}^{n \times d_F}$ is a feature matrix with d_F dimensions to describe the attribute information of nodes. The adjacency matrix of G is denoted by $A \in \mathbb{R}^{n \times n}$. A random walk in the graph G follows a transition probability matrix P, which varies depending on the type of walk. Here, P_{ij} denotes the probability of node i choosing node j as its next state. In this paper, we focus mainly on the semi-supervised node classification task. Specifically, given nodes in the training set T_V with known labels Y_v and feature vectors X_v for $v \in V$, our aim is to predict the unknown Y_u for all $u \in (V - T_V)$.

3.2 Preliminaries

Markov chain and graph random walk. A Markov chain (Chung 1967) is a stochastic process where transitions between states depend only on the current state, reflecting its memoryless property. It has the recurrence property if it can revisit certain states. A *random walk* (Lovász 1993) on a graph is a type of Markov chain, where each step depends on the current vertex.

Central limit theorem in graphs. The central limit theorem for Markov chains states that if two independent chains with arbitrary initial distributions share the same transition matrix, their limiting distributions will converge to the same value as time approaches infinity (Jones 2004).

The limiting distribution can be reached by a Markov

chain with *recurrence* and *aperiodic* property, satisfying:

$$\lim_{n\to\infty}(S_n = s_i) = \pi(s_i) \tag{1}$$

where S_n is the n-th state of the Markov chain and s_i is the i-th state in the state space. A random walk on a connected and non-bipartite graph is recurrent and aperiodic, guaranteeing convergence to the limiting distribution and satisfying the central limit theorem, as formalized in the following lemma:

Lemma 1. *If G is a connected, non-bipartite graph, then for any initial distribution π_0 on $v \in V$, we have:*

$$\lim_{n\to\infty}(\pi_0 P^n)(v) = \pi(v) \tag{2}$$

3.3 Graph Serialization

Graph serialization encompasses transforming a graph structure into a sequence structure and there exist two cases:

Case 1: Serializing a graph by a long walk. Leveraging the principles of the central limit theorem (Lemma 1) in the context of graphs offers a viable strategy for graph serialization. When the length of a random walk on a graph tends towards infinity, the initial distribution's impact diminishes, leading to a more comprehensive depiction of the underlying graph structure. We name the document generated through this method the *Graph Document*, defined as follows:

Definition 1 (Graph sentence and document)**.** *Graph sentence refers to an individual walk in the graph; the collection of a large number of graph sentences starting from every node in the graph constitutes a graph document.*

Nevertheless, it is crucial to acknowledge that employing an infinite-length walk on a graph is impractical during training. In practice, the limiting distribution can be approximated through a finite number of random walks, each with a limited length. Let $walk(v_i, v_j)$ denote a walk starting from node v_i and ending at node v_j. Then $walk(v_i, v_j)$ and $walk(v_j, v_k)$ can be viewed as partitions of $walk(v_i, v_k)$. By concatenating the graph sentences, we can create significantly longer walks that encompass all the nodes in the graph.

Case 2: Serializing nodes by diverse walks. Despite serializing the whole graph into graph document, it is also important to serialize the node's neighborhood for node classification tasks. To ensure that all sentences in the graph are strongly connected with the target node, the graph serialization approach can involve multiple diverse random walks initiated from the target node. Moreover, We also demonstrate that the coverage speed of a node's neighborhood increases exponentially through this neighborhood serialization method in Appendix A.2.

3.4 Graph Tokenization

Based on graph serialization, graph tokenization tokenizes the graph into various token types to meet the requirement of fine-grained and comprehensive node representation.

Walk-token Walk-tokens aim to achieve fine-grained representation, aligning with the second case of graph serialization. They are generated through four distinct walk types, each capturing information from nearby and distant neighbors at varying granularity levels.

(1) Uniform and non-backtracking random walk. *Uniform random walk* randomly selects the next node from the current node's neighbors. Additionally, we explore non-backtracking random walk, which exhibits faster convergence to its limiting distribution compared to uniform random walk in most cases (Alon et al. 2007).

Definition 2 (Non-backtracking random walk)**.** *A non-backtracking random walk on G is a sequence $(v_0, v_1, \ldots, v_k)$ of vertices $v_i \in V$ where v_{i+1} is randomly selected from the neighbors of v_i, and it must satisfy the condition $v_{i+1} \neq v_{i-1}$ for $i = 1, \ldots, k-1$.*

The non-backtracking random walk avoids revisiting the last encountered node. This seems to contradict the Markov property. However, by redefining the state space from nodes to edges, the non-backtracking random walk can still be seen as a Markov chain (Kempton 2016), allowing us to use the earlier mentioned theorem.

Let the edge connecting node u to node v be denoted as (u, v). The transition probability matrix $\tilde{P}((u, v), (x, y))$ for the non-backtracking random walk can be expressed as:

$$\tilde{P}((u,v),(x,y)) = \begin{cases} \dfrac{1}{d_v - 1} & if\ v = x,\ y \neq u \\ 0 & otherwise \end{cases} \tag{3}$$

where d_v denotes the degree of node v.

Furthermore, non-backtracking random walks can traverse longer distances and reduce redundancy in the generated walk sequences (Chen et al. 2022).

(2) Neighborhood jump walk and non-backtracking version. Conventional random walk approaches are limited by walk length. While the information from nearby neighbors is more significant, disregarding distant neighbors is not advisable. Inspired by dilated convolution (Yu and Koltun 2016), which has been proved to be effective (Contreras, Ceberio, and Kreinovich 2021), we define the neighborhood jump walk as a random walk with dilated transition probability. Unlike uniform random walks, the neighborhood jump walk can jump to any node within its k-hop neighborhood and continue recursively. Thus, the neighborhood jump walk's reachable area expands by the neighborhood's radius. Below is the definition of the neighborhood jump walk:

Definition 3 (Neighborhood jump walk)**.** *A neighborhood jump walk on G is a sequence $(v_0, v_1, ..., v_k)$ of vertices $v_i \in V$ where v_{i+1} is chosen among the nodes in the k-hop neighborhood of v_i. The transition probability of v_i to choose v_{i+1} in v_i's k-hop neighborhood follows k-step probability propagation.*

The transition probability matrix for the neighborhood jump walk, described in Definition 3, is computed using a k-step probability propagation process. In the given graph (Figure 3), for the 1-st step, the probability of node 1 (100%) is split averagely to its neighbors, node 2 (50%) and node 3 (50%). This process iterates to calculate the likelihood of node 1 reaching each node within its 3-hop neighborhood. By aggregating and normalizing these probabilities, the overall probability of node 1 visiting any node in its 3-hop neighborhood is established.

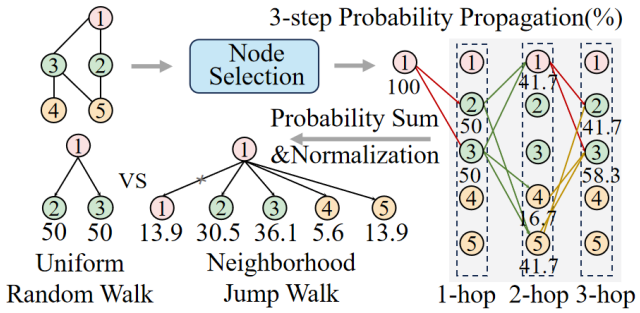

Figure 3: Transition Probability of Neighborhood Jump Walk.

These probabilities can be calculated in advance, thereby preserving the Markov property. Moreover, to prevent a node from revisiting itself, the probability of self-loops is eliminated. The likelihood of returning to the last visited node can also be negated to embed the *non-backtracking property*, which can accelerate the walk's convergence.

SGPM-token SGPM-token extends the length limit of walk-tokens, aligning with the first case of graph serialization and facilitating a comprehensive exploration of positional nuances across nodes in the graph. SGPM-tokens are created by applying weights derived from the Self-supervised Graph Pre-trained Model (SGPM).

Self-supervised Graph Pre-trained Model, SGPM. Self-supervised pre-trained models have emerged as a highly effective approach for learning representations from unlabeled NLP data. Inspired by this, we introduce SGPM to learn from graph document unsupervisedly.

(1) Document generation. As stated in Definition 1, a graph document consists of numerous graph sentences, which are walks generated by individual nodes with varying lengths. In the pre-training phase of SGPM, preserving the raw node connections is crucial for learning the graph's structure and contextual information. Thus, we use non-backtracking random walks as our final choice, as they extend the walk while maintaining original relationships. The walk length follows a normal distribution, $\mathcal{N}(\mu, \sigma^2)$, where the mean μ corresponds to the graph's radius R and the standard deviation σ is set to 1.

(2) Tokenizer and input representation. After generating the graph document, the next step involves tokenizing the sentences. Similar to tokenization in NLP, a *vocabulary* is created to encompass all tokens. In graph structures, each node maps directly to a corresponding token. Additionally, five special tokens are added: `PAD` for sequence padding, `UNK` for unknown nodes, `CLS` and `SEP` for sequence boundaries, and `MASK` for replacing hidden nodes.

Once the vocabulary is established, the graph sentences are tokenized into sequences comprised of these tokens. Before input into the SGPM, input representations for each token in the sequence are needed. Notably, let S denote the set of tokens, and for each token $j \in S$, the token embeddings T_j and position embeddings P_j are added, which are commonly used in NLP pre-train models. Furthermore, our input representation h_j is enriched by incorporating node features X_j and measures of node centrality C_j. Finally, we fuse these embeddings using sum pooling:

$$h_j = \text{SUM}(T_j, P_j, X_j, C_j) \tag{4}$$

(3) Graph pre-training process. Based on experience from NLP, we employ the Masked Language Model (MLM) (Devlin et al. 2019) as SGPM's pre-train task. Let $\mathbf{H} = [h_1, h_2, \ldots, h_N]$ represent the input sequence of tokens, and N the total number of tokens. Initially, 15% of the tokens are randomly replaced with the special token `MASK`. This masked sequence is then input into the bidirectional Transformer model, which generates individual scores $\mathbf{S} = [s_1, s_2, \ldots, s_N]$ for each token. These scores represent the likelihood of each token given its context and are then compared to the actual ground truth values $\mathbf{Y} = [y_1, y_2, \ldots, y_N]$, where y_i is the true label of the masked token h_i. The cross-entropy loss is calculated as:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(s_i) \tag{5}$$

Hop-token *Hop-token* is designed to capture hop information within a limited k-hop neighborhood, ensuring a structured and comprehensive representation of a node's local context. In Tokenphormer, the k-th *Hop-token* of the target node is computed as decoupled $(k+1)$-th layer of message passing, as depicted in Fig 2. This approach allows us to incorporate the information from nodes within the k-hop radius, providing a rich representation of the node's immediate surroundings.

3.5 Tokenphormer

Tokenphormer aims to utilize diverse tokens to jointly learn node embeddings for node classification. As illustrated in Figure 2, each node v has one SGPM-token, n hop-tokens, and m walk-tokens. These tokens are sequentially input into the Transformer encoder for processing, which includes multi-head self-attention (MSA) and position-wise feed-forward network (FFN) components, to compute the output of the l-th Transformer layer H_v^l:

$$\begin{aligned} \hat{H}_v^l &= MSA(Norm(H_v^{l-1})) + H_v^{l-1} \\ H_v^l &= FFN(Norm(\hat{H}_v^l) + \hat{H}_v^l \end{aligned} \tag{6}$$

where $l = 1, \ldots, L$ denotes the l-th Transformer layer. Within each mini-batch of inputs, we apply Layer Normalization using the $Norm(\cdot)$ function.

Since tokens are generated using different strategies, they may have varying contributions to the target node embedding. Instead of employing commonly used readout functions such as mean and summation, we employ an attention-based readout function to learn the different weights of token embeddings. Let $H \in R^{K \times d_h}$ denote the token representation of a node. Here, K represents the total number of tokens, d_h is the hidden dimension of the Transformer, and

Method	Year	Cora	Citeseer	Flickr	Photo	DBLP	Pubmed
GCN	2017	87.33±0.38	79.43±0.26	61.49±0.61	92.70±0.20	83.62±0.13	86.54±0.12
GAT	2017	86.29±0.53	80.13±0.62	54.29±2.56	93.87±0.11	84.19±0.19	86.32±0.16
GraphSAGE	2018	86.90±0.84	79.23±0.53	60.37±0.27	93.84±0.40	84.73±0.28	86.19±0.18
APPNP	2018	87.15±0.43	79.33±0.35	**93.25±0.24**	94.32±0.14	84.40±0.17	88.43±0.15
JKNet	2018	87.70±0.65	78.43±0.31	53.66±0.40	-	84.57±0.28	87.64±0.26
GPR-GNN	2021	88.27±0.40	78.46±0.88	-	94.49±0.14	-	89.34±0.25
GATv2	2022	-	-	-	-	84.96±0.47	85.75±0.55
GRAND+	2022	85.8x±0.4x	75.6x±0.4x	-	94.75±0.12	-	88.64±0.09
GT	2020	71.84±0.62	67.38±0.76	68.59±0.64	94.74±0.13	81.04±0.27	88.79±0.12
Graphormer	2021	72.85±0.76	66.21±0.83	66.16±0.24	92.74±0.14	80.93±0.39	82.76±0.24
SAN	2021	74.02±1.01	70.64±0.97	70.26±0.73	94.86±0.10	83.11±0.32	88.22±0.15
Gophormer	2021	87.85±0.10	80.23±0.09	91.51±0.28	-	85.20±0.20	89.40±0.14
ANS-GT	2022	88.60±0.45	80.25±0.39	-	94.41±0.62	-	89.56±0.55
GraphGPS	2022	-	-	-	95.06±0.13	-	88.94±0.16
Exphormer	2023	-	-	-	95.27±0.42	-	89.52±0.54
Gapformer	2023	87.37±0.76	76.21±1.47	-	94.81±0.45	**85.50±0.43**	88.98±0.46
NAGphormer	2023	90.56±0.39	80.02±0.80	89.66±0.63	95.49±0.11	84.62±0.13	89.60±0.14
Tokenphormer	ours	**91.20±0.47**	**81.04±0.24**	92.44±0.35	**96.14±0.14**	85.13±0.10	**89.94±0.20**

Table 1: Comparison of Tokenphormer with baselines on various datasets. The best results are in **bold**. 'x' and '-' mean unknown numbers. Part values of NAGphormer are run by ourselves due to missing of standard deviation in raw paper (Chen et al. 2023).

H_k is the k-th token representation of the node. We calculate the attention parameter between different tokens by:

$$\alpha_k = exp(H_k W_a^\top) / \sum_{i=1}^{K} (exp(H_i W_a^\top)) \tag{7}$$

where $W_a \in R^{1\times 2d_h}$ denotes the learnable parameter matrix, and $i = 1, \ldots, K$. Based on this, the final node representation H_{fin} is aggregated as follows:

$$H_{fin} = \sum_{k=1}^{K} \alpha_k H_k \tag{8}$$

3.6 Analysis of Components

Based on Definition 1 and the analysis presented in graph serialization part, it is evident that graph documents can converge to the limiting distribution. Notably, this distribution tends to vary across different graphs. Consequently, we propose the following Lemma to examine the expressiveness of graph documents and prove it through graph isomorphism, which can be found in Appendix A.1.

Lemma 2. *Graph documents possess the capability to distinguish non-isomorphic graphs.*

Additionally, we also analyze the walk coverage of walk tokens mathematically and draw the conclusion that it is feasible to approximate full information coverage with a limited number of walk tokens. We prove that the probability of a particular information type being uncovered decreases at a rate faster than exponential decay. The detailed proof can be found in Appendix A.2.

The expressiveness of different token types is also analyzed in Appendix A.3. Our analysis concludes that SGPM-tokens can capture global information, Hop-tokens have the potential to match or exceed the performance of MPNNs, and Walk-tokens demonstrate superiority over hop-wise tokens, such as those in NAGphormer (Chen et al. 2023). Lastly, we analyze the time and space complexity of Tokenphormer. Detailed analyses are provided in Appendix A.4.

4 Experiments

4.1 Comparison of Tokenphormer with Baselines

Table 1 compares all benchmark methods (Detailed descriptions of methods and datasets can be found in Appendix B.1) with Tokenphormer for node prediction tasks. Tokenphormer's results, presented as mean values with standard deviations from 10 runs with different random seeds, demonstrate state-of-the-art performance, outperforming most baselines on six datasets of varying sizes, proving the model's effectiveness.

Tokenphormer outperforms most MPNN baselines, showcasing strong generalization ability due to two factors: (1) Utilizing *hop-tokens* to enhance the density limit of *walk-tokens*, approximating layer-wise MPNN; (2) Capturing information beyond traditional MPNNs with diverse token types. As message passing layers increase, MPNNs suffer from over-smoothing and over-squashing, limiting their capacity to utilize information from distant neighbors.

Tokenphormer's superiority over graph Transformer baselines is due to several advantages: (1) *SGPM-tokens* extend the length limit of *walk-token*, capturing valuable contextual and global information for each node; (2) *Hop-tokens* extend the density limit of *walk-token*, retaining essential information from neighboring nodes and maintaining local relationships; (3) *Walk-tokens* generated through four types of walks add flexibility in capturing information from near and far neighbors. These mechanisms enable Tokenphormer to learn node representations through fine-grained tokens, capturing intricate details and relationships across the entire graph, leading to superior performance.

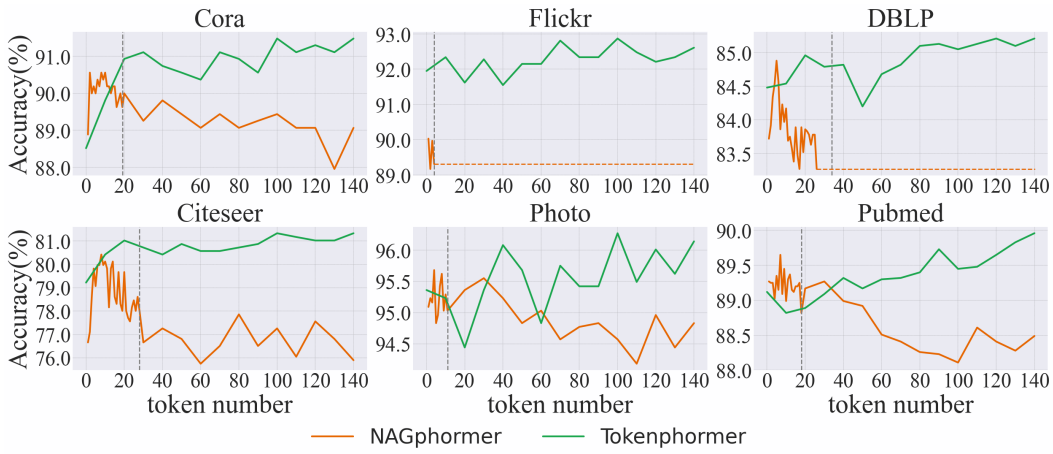

Figure 4: Expressiveness Comparison. The grey dashed line denotes graph diameter. For Flickr and DBLP, the orange dashed line means the NAN result.

4.2 Ablation Study

The effectiveness of various token types was verified through ablation experiments conducted on all datasets. The results, detailed in Appendix B.3, show a decrease in accuracy with the removal of any token type.

Furthermore, we compared mixed walks of uniform random walks, non-backtracking random walks, neighborhood jump walks, and non-backtracking neighborhood jump walks with four single walks. The findings, which can be found in Appendix B.4, also demonstrate the superiority of mixed walks.

4.3 Expressiveness Comparison

Tokenphormer enhances model performance by utilizing a more extensive set of valuable tokens. To evaluate Tokenphormer's token expressive ability with a maximum number of tokens, we conducted a thorough quantitative and qualitative analysis, comparing NAGphormer (Chen et al. 2023) and Tokenphormer under various token number settings.

For Tokenphormer, we use a step size of 10 *walk-tokens* to explore up to 140 *walk-tokens* with 1 fixed *SGPM-token* and 3 *hop-tokens*. For NAGphormer, we set step size as 1 for token numbers ranging from 1 to graph diameter and 10 for other cases. Figure 4 shows the quantitative performance across six benchmark datasets. As the token number increases, NAGphormer's performance rises rapidly, peaks before the token number equals the graph diameter, and then declines, falling below Tokenphormer. This means that NAGphormer's Hop2Token strategy still suffers from some extent of over-smoothing with the token number's increase. Conversely, Tokenphormer improves in accuracy and becomes more stable with more tokens.

4.4 Experiments on Heterogeneous Datasets

Three more heterogeneous datasets, including Cornell, Wisconsin, and Actor, were also evaluated (Table 2). Results show that even without SGPM, Tokenphormer is comparable to other methods and highly expressive on heterogeneous graphs. 1) GCN-based models perform poorly on heterogeneous graphs as they aggregate information from directly connected nodes. 2) Some Transformer-based models perform poorly, indicating susceptibility to irrelevant noise. 3) Tokenphormer performs remarkably well, especially better than NAGphormer, indicating its ability to explore graphs comprehensively and flexibly, thus obtaining richer information for target node representation learning.

Method	Year	Cornell	Wisconsin	Actor
GCN	2017	45.67±7.96	52.55±4.27	28.73±1.17
GAT	2017	47.02±7.66	57.45±3.51	28.33±1.13
APPNP	2018	41.35±7.15	55.29±3.90	29.42±0.81
GATv2	2022	50.27±8.97	52.74±3.96	28.79±1.47
SAN	2021	50.85±8.54	51.37±3.08	27.12±2.59
Gapformer	2023	**77.57±3.43**	83.53±3.42	36.90±0.82
NAGphormer	2023	56.22±8.08	63.51±6.53	34.33±0.94
Tokenphormer	ours	76.22±2.13	**86.17±2.30**	**37.01±0.83**

Table 2: Experiments on Heterogeneous Datasets. The results of Tokenphormer here are obtained without SGPM-tokens. Best results are in **bold**. The experiment results of baselines are taken from (Liu et al. 2023).

5 Conclusion

This paper proposes Tokenphormer, a novel graph Transformer that overcomes the limitations of MPNNs and graph Transformers by generating fine-grained and comprehensive node tokens. Walk-tokens, created through diverse walks with varying walk tendencies and receptive fields, ensure fine-grained exploration of the graph. SGPM-tokens capture global information, extending the length limit of walk-tokens for greater comprehensiveness, while hop-tokens, generated by decoupling message-passing layers, enhance local coverage. Tokenphormer achieves state-of-the-art performance on node classification on diverse datasets, demonstrating its effectiveness for various graph tasks.

Acknowledgments

This work was supported by the Science and Technology Development Fund Macau SAR (0003/2023/RIC, 0052/2023/RIA1, 0031/2022/A, 001/2024/SKL for SKL-IOTSC), the Research Grant of University of Macau (MYRG2022-00252-FST), Shenzhen-Hong Kong-Macau Science and Technology Program Category C (SGDX20230821095159012), and Wuyi University joint Research Fund (2021WGALH14). This work was performed in part at SICC which is supported by SKL-IOTSC, University of Macau.

References

Agrawal, N.; Sirohi, A. K.; Kumar, S.; et al. 2024. No Prejudice! Fair Federated Graph Neural Networks for Personalized Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 10775–10783.

Alon, N.; Benjamini, I.; Lubetzky, E.; and Sodin, S. 2007. Non-backtracking random walks mix faster. *Communications in Contemporary Mathematics*, 09(04): 585–603.

Alon, U.; and Yahav, E. 2021. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *International Conference on Learning Representations*.

Chen, D.; Lin, Y.; Li, W.; Li, P.; Zhou, J.; and Sun, X. 2020. Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04): 3438–3445.

Chen, J.; Gao, K.; Li, G.; and He, K. 2023. NAGphormer: A Tokenized Graph Transformer for Node Classification in Large Graphs. In *Proceedings of the International Conference on Learning Representations*.

Chen, R.; Zhang, S.; U, L. H.; and Li, Y. 2022. Redundancy-Free Message Passing for Graph Neural Networks. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 4316–4327. Curran Associates, Inc.

Chung, K. L. 1967. Markov chains. *Springer-Verlag, New York*.

Contreras, J.; Ceberio, M.; and Kreinovich, V. 2021. Why Dilated Convolutional Neural Networks: A Proof of Their Optimality. *Entropy*, 23(6).

Deng, A.; and Hooi, B. 2021. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 4027–4035.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North*.

Dufter, P.; Schmitt, M.; and Schütze, H. 2022. Position information in transformers: An overview. *Computational Linguistics*, 48(3): 733–763.

Dwivedi, V. P.; and Bresson, X. 2021. A Generalization of Transformer Networks to Graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*.

Dwivedi, V. P.; Liu, Y.; Luu, A. T.; Bresson, X.; Shah, N.; and Zhao, T. 2023. Graph transformers for large graphs. *arXiv preprint arXiv:2312.11109*.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 1263–1272. PMLR.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Huang, W.; Rong, Y.; Xu, T.; Sun, F.; and Huang, J. 2020. Tackling over-smoothing for general graph convolutional networks. *arXiv preprint arXiv:2008.09864*.

Jones, G. L. 2004. On the Markov chain central limit theorem. *Probability Surveys*, 1: 299 – 320.

Kempton, M. 2016. Non-backtracking random walks and a weighted Ihara's theorem. *arXiv preprint arXiv:1603.05553*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

Kong, K.; Chen, J.; Kirchenbauer, J.; Ni, R.; Bruss, C. B.; and Goldstein, T. 2023. GOAT: A global transformer on large-scale graphs. In *International Conference on Machine Learning*, 17375–17390. PMLR.

Kong, W.; Guo, Z.; and Liu, Y. 2024. Spatio-Temporal Pivotal Graph Neural Networks for Traffic Flow Forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 8627–8635.

Kreuzer, D.; Beaini, D.; Hamilton, W.; Létourneau, V.; and Tossou, P. 2021. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34: 21618–21629.

Liu, C.; Zhan, Y.; Ma, X.; Ding, L.; Tao, D.; Wu, J.; and Hu, W. 2023. Gapformer: Graph transformer with graph pooling for node classification. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI-23)*, 2196–2205.

Lovász, L. 1993. Random walks on graphs. *Combinatorics, Paul erdos is eighty*, 2(1-46): 4.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient Estimation of Word Representations in Vector Space. *International Conference on Learning Representations,International Conference on Learning Representations*.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I.; et al. 2018. Improving language understanding by generative pre-training.

Rong, Y.; Bian, Y.; Xu, T.; Xie, W.; Wei, Y.; Huang, W.; and Huang, J. 2020. Self-supervised graph transformer on

large-scale molecular data. *Advances in Neural Information Processing Systems*, 33: 12559–12571.

Schmitt, M.; Ribeiro, L. F. R.; Dufter, P.; Gurevych, I.; and Schütze, H. 2021. Modeling Graph Structure via Relative Position for Text Generation from Knowledge Graphs. In Panchenko, A.; Malliaros, F. D.; Logacheva, V.; Jana, A.; Ustalov, D.; and Jansen, P., eds., *Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15)*, 10–21. Mexico City, Mexico: Association for Computational Linguistics.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wu, Z.; Jain, P.; Wright, M.; Mirhoseini, A.; Gonzalez, J. E.; and Stoica, I. 2021. Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34: 13266–13279.

Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.-i.; and Jegelka, S. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 5453–5462. PMLR.

Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; and Liu, T.-Y. 2021. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34: 28877–28888.

Yu, F.; and Koltun, V. 2016. Multi-Scale Context Aggregation by Dilated Convolutions. In *ICLR*.

Yun, S.; Jeong, M.; Kim, R.; Kang, J.; and Kim, H. J. 2019. Graph transformer networks. *Advances in neural information processing systems*, 32.

Zhang, J.; Zhang, H.; Xia, C.; and Sun, L. 2020. Graph-Bert: Only Attention is Needed for Learning Graph Representations. *arXiv: Learning,arXiv: Learning*.

Zhang, Z.; Liu, Q.; Hu, Q.; and Lee, C.-K. 2022. Hierarchical graph transformer with adaptive node sampling. *Advances in Neural Information Processing Systems*, 35: 21171–21183.

Zhao, J.; Li, C.; Wen, Q.; Wang, Y.; Liu, Y.; Sun, H.; Ye, Y.; and Xie, X. 2021. Gophormer: Ego-Graph Transformer for Node Classification. *CoRR*, abs/2110.13094.