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.

Appendix

A Theoretical Analysis of Tokenphormer

In this section, we conduct various analyses to evaluate the effectiveness and expressiveness of Tokenphormer. Through these analyses, we aim to gain a deeper understanding of the capabilities and performance of Tokenphormer in effectively representing graph data.

A.1 Analysis of Graph Documents

In main content, we propose the following Lemma to examine the expressiveness of graph documents.

Lemma 1. *Graph documents possess the capability to distinguish non-isomorphic graphs.*

Proof. Suppose G is an undirected, weighted, and connected graph. If there exists an edge between two arbitrary nodes, denote as v_i and v_j, then its weight $w_{ij} = w_{ji} > 0$, else $w_{ij} = w_{ji} = 0$. The transition probability of random walk from v_i to v_j is:

$$P_{ij} = \frac{w_{ij}}{\sum_k w_{ik}} \tag{1}$$

Since random walk on G can be viewed as an invertible Markov chain, we have:

$$\pi(v_i)P_{ij} = \pi(v_j)P_{ji} \rightarrow \frac{\pi(v_i)}{\sum_k w_{ik}} = \frac{\pi(v_j)}{\sum_k w_{jk}} \tag{2}$$

Then, for all node v_i, we have:

$$\frac{\pi(v_i)}{\sum_k w_{ik}} = C \rightarrow \pi(v_i) = C\sum_k w_{ik} \tag{3}$$

where C is a constant. Since π is a probability distribution on all states, we have $\sum_i \pi(v_i) = 1$. Then we can get:

$$\sum_i (C\sum_k w_{ik}) = 1 \rightarrow C = \frac{1}{\sum_i \sum_k w_{ik}} \tag{4}$$

Finally, the limiting distribution of the Markov Chain is calculated:

$$\pi(v_i) = \frac{\sum_k w_{ik}}{\sum_i \sum_k w_{ik}} \tag{5}$$

Alternatively, in the scenario where G is unweighted, i.e., the weight between two connected nodes is 1, the resulting limiting distribution of the graph document can be described as follows:

$$\pi(v_i) = \frac{\sum_k w_{ik}}{\sum_i \sum_k w_{ik}} = \frac{d_i}{\sum_i d_i} = \frac{d_i}{2|E|} \tag{6}$$

Let G_a and G_b represent two arbitrary graphs. If they are isomorphic, the limiting distribution of these two graphs must equal each other. In other words, set π_{ai} ($i \in V_{G_a}$) and set π_{bj} ($j \in V_{G_b}$) are bijective to each other. In this case, the graph document ensures that it would not distinguish isomorphic graphs into different graphs.

If G_a and G_b are non-isomorphic, there are two situations. For the first situation, if G_a differs from G_b in $|V|$, it is clear that $\pi_a \neq \pi_b$ since the larger set between π_{ai} and π_{bj} is not surjective to smaller one. Thus, the graph document can easily distinguish them. For another situation, when G_a and G_b differs in $|V|$, G_a and G_b are indistinguishable to graph document if and only if G_a and G_b satisfy: $f(x) = x$ is bijective function when:

$$D = \left\{ \frac{\sum_k w_{aik}}{\sum_{ai}\sum_k w_{aik}} \middle| i \in V_{G_a} \right\}, R = \left\{ \frac{\sum_k w_{bjk}}{\sum_{bj}\sum_k w_{bjk}} \middle| j \in V_{G_b} \right\} \tag{7}$$

and

$$D = \left\{ \frac{\sum_k w_{bjk}}{\sum_{bj}\sum_k w_{bjk}} \middle| j \in V_{G_b} \right\}, R = \left\{ \frac{\sum_k w_{aik}}{\sum_{ai}\sum_k w_{aik}} \middle| i \in V_{G_a} \right\} \tag{8}$$

where D is the domain and R represents the range of rejection function $f(x) = x$. In this case, the probability for two graphs with the same $|V|$ and different structures to be indistinguishable drops rapidly with the increase of $|V|$. □

Under this analysis, the distinguishing ability of the graph document enables it to capture unique characteristics and structures inherent to the graph, thus obtaining high expressive power.

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

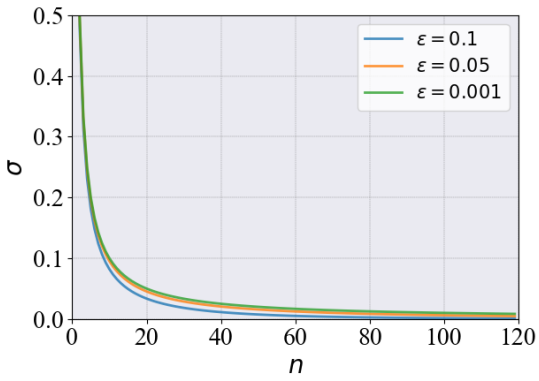

Figure 1: **Coverage Analysis.** Let $\sigma = \frac{\exp\left(-2\epsilon^2 n\right)}{n}$, the figure shows the change of σ with increase of n in case of different ϵ in equation 13.

A.2 Analysis of Walk Coverage

Walk coverage pertains to the extent of information coverage within a node's neighborhood, utilizing a limited number of walks. We classify walks based on node labels to achieve this, enabling a systematic analysis of the information encapsulated within the walks.

When two walks have the same length (k) and share identical node labels at corresponding positions, we assume the information obtained from these walks is similar. Consequently, we classify such walks into the same information type, denoted as $l_1, l_2, ..., l_k$, where l_i represents the label of node i. By doing so, we can analyze walk coverage based on the coverage of information types present in a node's neighborhood.

As proven by (Sun et al. 2022), we follow a similar approach to analyzing the walk coverage. The definition of label-limited transition matrix $P^{(l)}$ is:

$$P_{i,j}^{(l)} = \begin{cases} P_{i,j}, & Y_j = l \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where l denotes the specified label, Y_j is the label of v_j and $P_{i,j}$ is the raw transition probability of v_i to v_j. Then the probability of v_i transition to v_j through the type of $\{l_1, l_2, ..., l_k\}$ is:

$$P^{\{l_1,l_2,...,l_k\}} = \prod_{i=1}^{k} P^{(l_i)} \tag{10}$$

Then, the probability of an information type $\{l_1, l_2, \ldots, l_k\}$ starting from v_S being sampled is:

$$P_S^{\{l_1,l_2,...,l_k\}} = \sum_{i=1}^{|V|} P_{S,i}^{\{l_1,l_2,...,l_k\}} \tag{11}$$

Let $N_S^{\{l_1,l_2,...,l_k\}}$ denotes number of sampled information types in n times sampling. It is clear that $N_S^{\{l_1,l_2,...,l_k\}}$ follows a binominal distribution:

$$N_S^{\{l_1,l_2,...,l_k\}} \sim B(n, P_S^{\{l_1,l_2,...,l_k\}}) \tag{12}$$

At last, according to Hoeffding's inequality (Hoeffding 1994), we have:

$$\begin{aligned} D_S^{\{l_1,l_2,...,l_k\}} &= |\frac{N_S^{\{l_1,l_2,...,l_k\}}}{n} - P_S^{\{l_1,l_2,...,l_k\}}| \\ P(D_S^{\{l_1,l_2,...,l_k\}} > \epsilon) &\leq \frac{\exp\left(-2\epsilon^2 n\right)}{n} \end{aligned} \tag{13}$$

where $D_S^{\{l_1,l_2,...,l_k\}}$ stands for sampling deviation. Thus, for any $\epsilon > 0$, the probability of one information type being not sampled being larger than ϵ decreases rapidly (since $\exp\left(-2\epsilon^2 n\right)$ already decreases exponentially, and $\frac{\exp\left(-2\epsilon^2 n\right)}{n} \leq \exp\left(-2\epsilon^2 n\right)$ when $n \geq 1$) as the number of sampled walks n increases. (As shown in Fig. 1)

The above analysis demonstrates that, although covering all information from the node's neighborhood is challenging, obtaining most information is still achievable through a limited number of samplings. Our experiments also verified this point: when the number of walks increased to a certain degree, the experimental results tended to be stable, and the model reached the fitting.

A.3 Analysis of tokens

The proposed Tokenphormer generates multiple tokens with different granularities and learns each other's weights in a fully connected scheme through the Transformer, thus alleviating the problems of over-smoothing and over-squashing. At the same time, by generating plentiful walk-based tokens, Tokenphormer is more effective in retaining structure information than other graph Transformers.

SGPM-token. Appendix A.1 has demonstrated that the graph document can distinguish between different graphs, showcasing its proficiency in capturing the global information of the graph. The the SGPM-token, derived through SGPM, is critical in acquiring contextual information for each node within the graph document. With the generation of numerous walks for each node, the graph document ensures comprehensive coverage for every node. Additionally, the length of the graph sentence follows a normal distribution, with a mean equal to the graph's radius. This intentional design choice aligns the receptive field of the graph sentence, starting from each node, with the global characteristics of the graph. Furthermore, the graph sentences enable the exploration of far nodes by utilizing the non-backtracking random walk. This approach effectively reduces redundancy, allowing nodes to acquire diverse contexts and enriching the semantic information they gather (Chen et al. 2022).

Hop-token. The hop-token comes from the decoupled message passing layer, and its calculation is:

$$H^k = A^k X \tag{14}$$

where H^k represents the feature matrix of k-hop. H_i^k is node i's aggregated information from k-th layer of MPNN. *hop-tokens* in Tokenphormer can be viewed as MPNN with layer-wise attention added. Since MPNN already proves its expressive power in node prediction, the hop-token has the

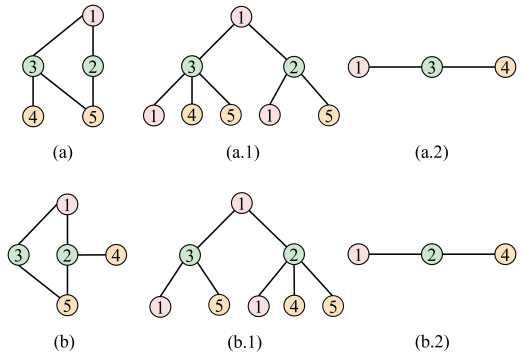

Figure 2: hop-token vs walk-token.

potential to achieve equal or better performance (Chen et al. 2023).

Walk-token. Instances arise where the hop-token falls short in distinguishing between certain situations. This is exemplified in Fig. 2, where (a) and (b) represent graphs with distinct structures, and (a.1) and (b.2) illustrate the message passing trees for node 1 in their respective graphs. In the depicted figure, the horizontal layers can be interpreted as hop-tokens, while the vertical paths represent walk-tokens. It is evident that when considering a 2-hop message passing scenario, hop-tokens cannot differentiate between graphs (a) and (b). However, walk-tokens, exemplified by (a.2) and (b.2), exhibit the potential to distinguish between the two graphs successfully.

A.4 Complexity Analysis

The generation of all tokens in Tokenphormer can be performed beforehand. SGPM is pre-trained, so the SGPM-token can be obtained by Tokenphormer easily in $O(1)$ complexity. Moreover, walks are generated before training, and Hop-tokens can also be computed in advance. So, we only consider the complexity of Tokenphormer without the token generation phase.

Time complexity. The time complexity of Tokenphormer primarily relies on the self-attention module of the Transformer. For each node in the graph, we perform computations proportional to the square of the number of tokens N_t and the dimension of the feature d_F, thus resulting in a time complexity of $O(|V| \cdot N_t^2 \cdot d_F)$, where $|V|$ denotes the number of nodes.

Space complexity. The space complexity is determined by the number of model parameters and the outputs of each layer. It can be broken down into two main parts: 1) The Transformer layer contributes $O(d_F^2 \cdot N_L)$, where N_L is the number of Transformer layers. 2) The attention matrix and hidden node representations add $O(S_b \cdot N_t^2 + S_b \cdot N_t \cdot d_F)$, where S_b denotes the batch size. Thus, the total space complexity is $O(d_F^2 \cdot N_L + S_b \cdot N_t^2 + S_b \cdot N_t \cdot d_F)$.

B Experiment

B.1 Experiment Setup

Datasets. To comprehensively evaluate the effectiveness of Tokenphormer, we conduct experiments on six homogeneous graph datasets with varying node numbers, ranging from 2,708 to 19,717. The datasets include small-sized datasets like Cora and Citeseer (around 3,000 nodes), medium-sized datasets like Flickr and Amazon Photo (around 7,500 nodes), and large-sized datasets like DBLP and Pubmed (around 18,000 nodes). We apply a 60%/20%/20% train/val/test split (consistent with Gophormer (Zhao et al. 2021) and NAGphormer (Chen et al. 2023)) for these benchmark datasets. The detailed dataset information is presented in Table 1.

Dataset	Nodes	Edges	Classes	Features	Diameter
Cora	2,708	5,278	7	1,433	19
Citeseer	3,327	4,522	6	3,703	28
Flickr	7,575	239,738	9	12,047	4
Photo	7,650	238,163	8	745	11
DBLP	17,716	52,864	4	1,639	34
Pubmed	19,717	44,324	3	500	18

Table 1: Statistics on datasets. Graph diameter is calculated using the method described in (Akiba, Iwata, and Kawata 2015).

Baselines. To comprehensively assess the effectiveness of Tokenphormer in the context of node classification tasks, a comparative analysis is conducted against a diverse set of 16 advanced baseline models. The evaluated models encompass eight prominent MPNN methods, including GCN (Kipf and Welling 2017), GAT (Veličković et al. 2018), GraphSage (Hamilton, Ying, and Leskovec 2017), APPNP (Klicpera, Bojchevski, and Günnemann 2018), JKNet (Xu et al. 2018), GPR-GNN (Chien et al. 2021), GRAND+ (Feng et al. 2022), GATv2 (Brody, Alon, and Yahav 2021) and eight innovative graph Transformer models, including GT (Dwivedi and Bresson 2021), Graphormer (Ying et al. 2021), SAN (Kreuzer et al. 2021), Gophormer (Zhao et al. 2021), ANS-GT (Zhang et al. 2022), GraphGPS (Rampášek et al. 2022), Exphormer (Shirzad et al. 2023), Gapformer (Liu et al. 2023), and NAGphormer (Chen et al. 2023).

B.2 Experiment Details

For the model configuration of Tokenphormer, in SGPM, we generate 100 non-backtracking random walks for each node on the graph as the training dataset, where the walk lengths follow a normal distribution $\mathcal{N}(\mu = \text{graph radius or } 10, \sigma^2 = 1)$, and 20 walks are generated as the validate dataset. In the *walk-token* part, we generate 100 walks with lengths starting from 4 and a ratio of mixed walk types starting from $25\% : 25\% : 25\% : 25\%$ and tune these hyper-parameters to the best. For *hop-token*, we fixed the hop number to 3 for all datasets. In model training, we adopt the AdamW optimizer (Loshchilov and Hutter 2017) and set the learning rate to $1e^{-2}$ for Flickr and Photo and $5e^{-3}$

Method	Cora	Citeseer	Flickr	Photo	DBLP	Pubmed
Tokenphormer	**91.20±0.47**	**81.04±0.24**	**92.44±0.35**	**96.14±0.14**	**85.13±0.10**	**89.94±0.20**
w/o SGPM-token	90.33±0.30	79.85±1.00	91.75±0.50	95.61±0.47	84.42±0.41	89.38±0.46
w/o walk-token	88.37±1.23	79.70±0.71	92.24±0.35	94.81±0.44	83.93±0.16	89.60±0.35
w/o hop-token	90.73±0.34	80.19±2.32	91.97±0.71	95.80±0.30	85.01±0.16	89.35±0.36
only URW	90.41±0.56	80.69±0.49	92.07±0.77	95.83±0.32	84.90±0.39	89.33±0.39
only NBRW	89.72±0.36	80.03±0.90	92.27±0.53	95.74±0.41	84.86±0.25	89.26±0.45
only NJW	90.80±0.66	80.92±0.50	92.38±0.70	95.61±0.38	84.70±0.44	89.50±0.36
only NBNJW	89.63±0.64	80.56±0.74	92.09±0.39	96.00±0.31	84.71±0.35	89.36±0.35

Table 2: Ablation Experiment. Best results are in **bold**. URW, NBRW, NJW, and NBNJW represent uniform random walk, non-backtracking random walk, neighborhood jump walk, and non-backtracking neighborhood jump walk.

for other datasets and weight decay to $1e^{-5}$. The dropout rate is set to 0.1, and the Transformer head is set to 1. The batch size is set to 2000. For SGPM, as a pre-train phase, we conducted it on a Linux server with DGX-H800 (4 $\times$ NVIDIA H800 80 GB) GPU. Tokenphormer was conducted on a Linux server with 1 R9-5950X CPU and an NVIDIA RTX 3090TI (24GB) GPU.

B.3 Ablation Study

To analyze the effectiveness of *SGPM-token*, *walk-token*, and *hop-token*, ablation experiments are carried out on all datasets, as shown in Table 2. It can be seen that with any token dropped, the accuracy decreases to some extent.

SGPM-token. *SGPM-token* captures contextual and global information of the selected node. Without *SGPM-token*, the accuracy of Tokenphormer drops in all six datasets.

Walk-token. Tokenphormer's accuracy decreases the most when *walk-token* are omitted. This is because *walk-tokens* are able to capture fine-grained information from a large neighborhood.

Hop-token. *Hop-token* ensures coverage in k-hop neighborhoods. However, for most datasets, large number of *walk-tokens* already cover all nodes in k-hop neighborhoods, resulting in a smaller accuracy decrease when *hop-token* is withdrawn.

B.4 Walk Comparison

We compare mixed walks (uniform random walks, non-backtracking random walks, neighborhood jump walks, and non-backtracking neighborhood jump walks) with four single walks. Table 2 shows that different walks vary in effectiveness across datasets, but mixed walks consistently achieve higher accuracy. For example, for citation networks like Cora and Citeseer, where information from near neighbors is crucial, walks without non-backtracking property can perform better than non-backtracking ones. For datasets like Photo, the situation changes. The non-backtracking neighborhood jump walk performs the best among all types of walks. These results indicate that mixed walks leverage the strengths of all types to enhance overall accuracy.

References

Akiba, T.; Iwata, Y.; and Kawata, Y. 2015. An exact algorithm for diameters of large real directed graphs. In *Experimental Algorithms: 14th International Symposium, SEA 2015, Paris, France, June 29–July 1, 2015, Proceedings 14*, 56–67. Springer.

Brody, S.; Alon, U.; and Yahav, E. 2021. How Attentive are Graph Attention Networks. *arXiv: Learning,arXiv: Learning*.

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.

Chien, E.; Peng, J.; Li, P.; and Milenkovic, O. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *International Conference on Learning Representations*.

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

Feng, W.; Dong, Y.; Huang, T.; Yin, Z.; Cheng, X.; Kharlamov, E.; and Tang, J. 2022. Grand+: Scalable graph random neural networks. In *Proceedings of the ACM Web Conference 2022*, 3248–3258.

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

Hoeffding, W. 1994. *Probability Inequalities for sums of Bounded Random Variables*, 409–426.

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

Klicpera, J.; Bojchevski, A.; and Günnemann, S. 2018. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. *International Conference on Learning Representations,International Conference on Learning Representations*.

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.

Loshchilov, I.; and Hutter, F. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Rampášek, L.; Galkin, M.; Dwivedi, V. P.; Luu, A. T.; Wolf, G.; and Beaini, D. 2022. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35: 14501–14515.

Shirzad, H.; Velingker, A.; Venkatachalam, B.; Sutherland, D. J.; and Sinop, A. K. 2023. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, 31613–31632. PMLR.

Sun, Y.; Deng, H.; Yang, Y.; Wang, C.; Xu, J.; Huang, R.; Cao, L.; Wang, Y.; and Chen, L. 2022. Beyond homophily: structure-aware path aggregation graph neural network. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI*, 2233–2240.

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

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.

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.