Text Graph Transformer for Document Classification

Haopeng Zhang **Jiawei Zhang**
IFM Lab, Department of Computer Science,
Florida State University, FL, USA
haopeng,jiawei@ifmlab.org

Abstract

Text classification is a fundamental problem in natural language processing. Recent studies applied graph neural network (GNN) techniques to capture global word co-occurrence in a corpus. However, previous works are not scalable to large-sized corpus and ignore the heterogeneity of the text graph. To address these problems, we introduce a novel *Transformer* based heterogeneous graph neural network, namely Text Graph Transformer (TG-Transformer). Our model learns effective node representations by capturing structure and heterogeneity from the text graph. We propose a mini-batch text graph sampling method that significantly reduces computing and memory costs to handle large-sized corpus. Extensive experiments have been conducted on several benchmark datasets, and the results demonstrate that TG-Transformer outperforms state-of-the-art approaches on text classification task.

1 Introduction

Text classification is a widely studied problem in natural language processing and has been addressed in many real-world applications such as news filtering, spam detection, and health record systems (Kowsari et al., 2019; Che et al., 2015; Zhang et al., 2018). The objective is to assign corresponding labels to textual units based on text representations.

Deep learning models like Convolutional Neural Networks (CNN) (Kim, 2014) and Recurrent Neural Networks (RNN) (Hochreiter and Schmidhuber, 1997) have been applied for text representation learning instead of traditional hand-crafted features, such as n-gram and bag-of-words (BoW) (Joulin et al., 2016). Researchers have recently turned to Graph Neural Network (GNN) to exploit global features in text representation learning, which learns node embedding by aggregating information from neighbors through edges. Defferrard et al. (2016) first generalized CNN to graph for text classification task. Then Yao et al. (2019) applied Graph Convolution Network (GCN) (Kipf and Welling, 2016) on a corpus level heterogeneous text graph and achieved state-of-the-art performance. Liu et al. (2020) further improved classification accuracy by expanding the text graph with semantic and syntactic contextual information.

However, these GCN-based models on heterogeneous text graphs suffer from two practical issues. Firstly, none of these models are scalable to large-sized corpus due to high computation and memory costs. Calculation of all the nodes in the graph is required at each layer during training. Secondly, all these models ignore the heterogeneity of the text graph, which consists of both document and word nodes. Distinguishing nodes of different types will benefit node representation learning.

To address the above problems, we propose a novel *Transformer*-based heterogeneous GNN model, namely Text Graph Transformer (TG-Transformer). Instead of learning based on the full text graph, we propose a text graph sampling method that enables subgraph mini-batch training. The significantly reduced computing and memory costs make the model scalable to large-sized corpus. Moreover, we utilize *Transformer* to aggregate information in subgraph batch with two proposed graph structural encodings. We also distinguish the learning process of different type nodes to fully utilize the heterogeneity of text graph. The main contributions of this work are as follows:

1. We propose Text Graph Transformer, a heterogeneous graph neural network for text classification. It is the first scalable graph-based method for the task to the best of our knowledge.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 8322–8327,
November 16–20, 2020.

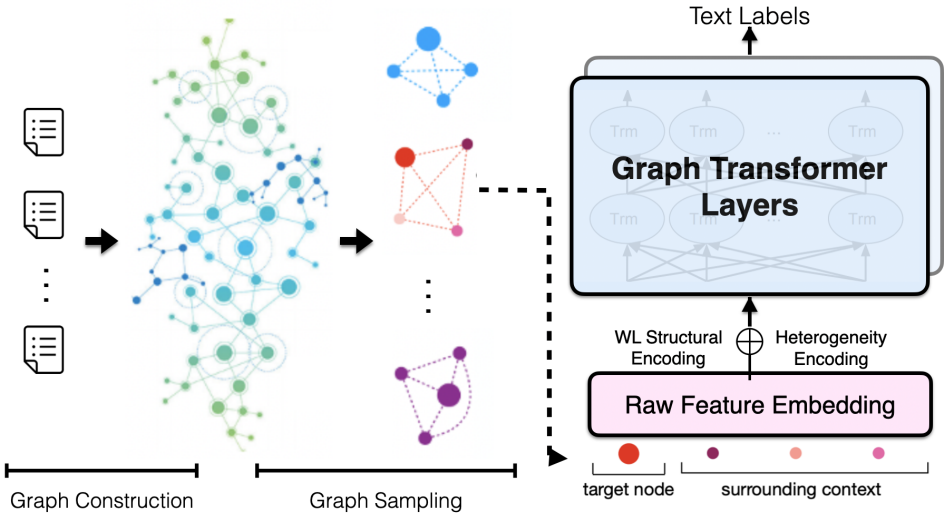

Figure 1: Overall Structure of TG-Transformer

2. We propose a novel heterogeneous text graph sampling method that significantly reduces computing and memory costs.

3. We perform experiments on several benchmark datasets, and the results demonstrate the effectiveness and efficiency of our model.

2 Methodology

In this section, we introduce TG-Transformer in great detail. First, we present how to construct a heterogeneous text graph for a given corpus. Then, we introduce our text graph sampling method, which can generate subgraph mini-batch from the text graph. These subgraph batches will be fed into TG-Transformer to learn efficient node representations for classification. The overall structure of our model is shown in Fig. 1.

2.1 Text Graph Building

To capture global word co-occurrence within corpus, we build a heterogeneous text graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \mathcal{F})$. The text graph contains two types of nodes: word nodes ($\mathcal{U}$) representing all documents in the corpus and document nodes ($\mathcal{V}$) representing all the words in the corpus vocabulary. The text graph also contains two types of edges: word-document edges ($\mathcal{E}$) and word-word edges ($\mathcal{F}$). Word-document edges are built based on word occurrence within documents with edge weights measured by the term frequency-inverse document frequency (TF-IDF) method. Word-word edges are built based on local word co-occurrence within sliding windows in the corpus, with edge weights measured by point-wise mutual information (PMI):

$$\text{PMI}(w_i, w_j) = \log \frac{p_{i,j}}{p_i p_j} = \log \frac{N_{i,j} N}{N_i N_j}, \quad (1)$$

where $N_i, N_j, N_{i,j}$ are the number of sliding windows in a corpus that contain word w_i, word w_j and both w_i, w_j. N is the total number of sliding windows in the corpus.

2.2 Text Graph Sampling

To reduce computing and memory cost, we propose a text graph sampling method. Instead of learning based on the entire text graph, TG-Transformer is trained on sampled subgraph mini-batch, making it scalable to large-sized corpus. We separate sub-graph sampling as a pre-process step in an unsupervised manner for controlling the time costs in model learning.

We first calculate the intimacy matrix $\mathbf{S}$ of the text graph based on pagerank algorithm:

$$\mathbf{S} = \alpha \cdot (\mathbf{I} - (1 - \alpha) \cdot \overline{\mathbf{A}})^{-1}, \quad (2)$$

where factor $\alpha \in [0, 1]$ is usually set as 0.15. $\overline{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix, $\mathbf{A}$ is the adjacency matrix of the text graph, and $\mathbf{D}$ is its corresponding diagonal matrix. Each entry $\mathbf{S}_{i,j}$ measures the intimacy score between node i and node j.

For any document target node $v_i \in \mathcal{V}$, we sample its context subgraph $\mathcal{C}(v_i)$ of size k by selecting its top k intimate neighbour word nodes $u_j \in \mathcal{U}$.

Meanwhile, for any word target node $u_i \in \mathcal{U}$, we first calculate the ratios of two type incident

Table 1: Statistics of the experiment datasets. V denotes the vocabulary size, C the number of classes, and W the average number of words per document.

Dataset	Train	Test	V	C	W
R8	5,485	2,189	7,688	8	65.72
R52	6,532	2,568	8,892	52	69.82
Ohsumed	3,357	4,043	14,157	23	135.82
IMDB	278,732	69,683	115,831	10	325.6
Yelp 2014	900,309	225,077	476,191	5	148.8

edge:

$$r_w(u_i) = \frac{|\mathcal{F}(u_i)|}{|\mathcal{F}(u_i)| + |\mathcal{E}(u_i)|}, \quad (3)$$

$$r_d(u_i) = \frac{|\mathcal{E}(u_i)|}{|\mathcal{F}(u_i)| + |\mathcal{E}(u_i)|}, \quad (4)$$

where $\mathcal{F}(u_i), \mathcal{E}(u_i)$ are the sets of word-word edges, word-document edges incident to u_i with intimacy score larger than threshold θ. We sample its context subgraph $\mathcal{C}(u_i)$ of size k by selecting its top $k \cdot r_w(u_i)$ intimate neighbour word nodes and its top $k \cdot r_d(u_i)$ intimate neighbour document nodes, respectively.

2.3 Text Graph Transformer

Based on the sampled subgraph mini-batch, TG-Transformer will update the text graph nodes' representations iteratively for classification. We build one model for each target node type (document/word) to model heterogeneity. The input of our model will be raw feature embeddings of nodes in subgraph batch injected by the following two extra structural encodings:

Heterogeneity Encoding The heterogeneity encoding can capture the document and word types in the text graph. Similar to the segment encoding in (Devlin et al., 2018), we use 0 and 1 to encode document nodes and word nodes, respectively.

Weisfeiler-Lehman Structural Encoding We adopt the WL Role Embedding by (Zhang et al., 2020a) to capture the structure of text graph. The Weisfeiler-Lehman (WL) algorithm (Niepert et al., 2016) can label nodes according to their structural roles in the graph. For node v_j (document or word node) in the sampled subgraph, we can denote its WL code as $WL(v_j) \in \mathbb{N}$, and the encoding is defined as:

$$\left[\sin\left(\frac{\mathrm{WL}\left(v_j\right)}{10000^{\frac{2l}{d_h}}}\right), \cos\left(\frac{\mathrm{WL}\left(v_j\right)}{10000^{\frac{2l+1}{d_h}}}\right)\right]_{l=0}^{\left\lfloor \frac{d_h}{2} \right\rfloor}. \quad (5)$$

These two encodings have the same dimension (i.e., d_h) as the original raw feature embeddings, so we add them together as the initial node representation for the input subgraph, which can be denoted as $\mathbf{H}^{(0)}$.

Graph Transformer Layer The D layer graph transformer will aggregate information from subgraph batch to learn the target node representation. Each graph transformer layer contains three trainable matrices: $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_v \in \mathbb{R}^{d_h \times d_h}$ and queries $\mathbf{Q}$, keys $\mathbf{K}$ and values $\mathbf{V}$ are generated by multiplying the input correspondingly:

$$\{\mathbf{Q}, \mathbf{K}, \mathbf{V}\} = \mathbf{H}^{(l-1)} \left\{\mathbf{W}_Q^{(l)}, \mathbf{W}_K^{(l)}, \mathbf{W}_V^{(l)}\right\}. \quad (6)$$

Then a TG-Transformer layer can be donated as:

$$\begin{aligned} \mathbf{H}^{(l)} &= \text{G-Transformer}\left(\mathbf{H}^{(l-1)}\right) \\ &= \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}}\right)\mathbf{V} + \text{G-Res}, \end{aligned} \quad (7)$$

where G-res refers to the graph residual term in (Zhang and Meng, 2019) to solve the over-smoothing issue of GNNs. The output of the last layer $\mathbf{H}^{(D)}$ will be averaged as the final representations $\mathbf{z}$ of the target node and fed into a *softmax* classifier:

$$\mathbf{z} = \text{softmax}(\text{average}(\mathbf{H}^{(D)}) \in \mathbb{R}^{d_y \times 1}. \quad (8)$$

Based on the sampled subgraphs for all the nodes in the training set, e.g., $\mathcal{T}$, we can define the cross-entropy based loss function as:

$$\ell = -\sum_{n \in \mathcal{T}} \sum_{f=1}^{d_y} \mathbf{y}_n(f) \log \mathbf{z}_n(f), \quad (9)$$

Table 2: Text classification accuracy results. Models with "*" utilize pre-trained Glove word embeddings. For the scores not reported in the existing works, we mark them with '-' in the table.

Model	R8	R52	Ohsumed	IMDB	Yelp 2014
CNN*	95.7±0.5	87.6±0.4	58.4±1.0	42.7±0.4	66.1±0.6
LSTM*	96.1±0.2	90.5±0.8	51.1±1.5	52.1±0.3	68.4 ±0.1
fastText*	96.1±0.2	92.8±0.1	57.7±0.5	45.2±0.4	66.2±0.6
Text GCN	97.0±0.1	93.7±0.1	67.7±0.3	-	-
Text GNN*	97.8±0.2	94.6±0.3	69.4±0.6	-	-
Tensor GCN*	98.0±0.1	95.0±0.1	70.1±0.2	-	-
TG-Transformer*	98.1±0.1	95.2±0.2	70.4±0.4	53.4±1.2	69.8±0.6

where $n \in \mathcal{T}$ denotes the target word/document nodes in the training set, d_y is the label vector dimension, and $\mathbf{y}_n$ represents the ground-truth label vector of node n.

3 Experiment

3.1 Experimental Setup

Datasets We evaluate the effectiveness of our model on five benchmarked datasets: R52 and R8 Reuters dataset[1] for news documents classification, Ohsumed dataset[2] for medical bibliographic classification, and two large-scale review rating datasets: IMDB and Yelp 2014. Detailed statistics of the datasets are summarized in Table 1.

Baselines We compare our method with three classical baseline models: **CNN** in (Kim, 2014), **LSTM** in (Liu et al., 2016) and **fastText** in (Joulin et al., 2016) using the average of word/n-grams embeddings. In addition, we compare with three state-of-the-art GNN-based models: **TextGCN** in (Yao et al., 2019) using GCN, **Text GNN**[3] in (Huang et al., 2019) using text level graphs and **TensorGCN** in (Liu et al., 2020) using semantic and syntactic contextual information.

Implementation We set the node representation dimension as 300 and initialize with Glove word embeddings (Pennington et al., 2014). We train a 2-layer graph transformer with a hidden size 32 and 4 attention heads. We use mini-batch SGD with Adam optimizer (Kingma and Ba, 2014), and the dropout rate is set as 0.5. The initial learning rate as 0.001, and we decay it with weight decay $5e^{-4}$. 10 percent of the training set is randomly selected as validation set, and we stop training if the validation set loss does not decrease for 10 consecutive epochs.

Table 3: Training time per epoch of GNN-based models.

Model	R52	Ohsumed
Text GCN	2.64	3.48
Tensor GCN	4.32	5.13
TG-Transformer	0.83	1.17

3.2 Experiment Results

Table 2 presents the classification accuracy of our model compared with baseline methods. GNN-based models generally perform better than sequential and bag-of-word models due to its ability to model global word co-occurrence in the corpus, and TG-Transformer outperforms other graph models with much less memory and computing cost. This is likely due to the utilization of the text graph's heterogeneity and effective representing learning by Graph Transformer Layers. Moreover, TG-Transformer performs well on large-sized corpus such as IMDB and Yelp 14. We also evaluate model efficiency with training time per epoch, as shown in Table 3. It can be observed that our text graph sampling method reduces the computing cost significantly and makes our model scalable to large corpus, where previous GNN-based models such as Text GCN are not applicable due to computing power limit.

Hyperparameter Here we analyze the effects of subgraph sizes k in sampling. We notice parameter k has a large influence on the model performance since it defines the number of neighbor nodes used to update the target node representation. During parameter tuning, we notice the learning performance improves steadily as k increases from 1 to an optimal value (i.e.,23 for R8) and starts decreasing as

[1] https://www.cs.umb.edu/ smimarog/textmining/datasets/
[2] http://disi.unitn.it/moschitti/corpora.htm
[3] We give this name for simplicity.

Table 4: Ablation study results.

Settings	R52	Ohsumed
Original	95.2	70.4
(1) Without structural encodings	94.8	69.6
(2) Without pre-trained Emb.	93.5	66.9
(3) Simultaneous updating	94.7	69.3

k further increases. The same trend is noticed for all datasets. The computing cost to train the model also increases as k goes larger, but is still less than other GNN-based models.

Ablation Study We perform ablation studies to analyze our model further, as shown in Table 4. In (1), we remove the two structural encodings and only use raw feature embeddings as input. The decreased performance demonstrates that the structural encodings capture some useful heterogeneous graph structure information. In (2), we remove pre-trained word embeddings and initialize all the nodes with random vectors. Model performance has a larger decrease, demonstrating the significance of pre-trained word embeddings and initial node representations on our model. In (3), we train one model to update and learn both subgraph batch target node types. The slightly decreasing classification accuracy reflects the importance of modeling heterogeneity information of the text graph.

4 Related Work

4.1 Text Classification

Traditional text classification studies rely on handcrafted features like BoW (Zhang et al., 2010) and n-gram (Wang and Manning, 2012). With the development of deep learning, researchers applied CNN (Kim, 2014; Zhang et al., 2015), LSTM (Tai et al., 2015; Liu et al., 2016), word embedding techniques (Joulin et al., 2016; Pennington et al., 2014), attention mechanism (Yang et al., 2016; Wang et al., 2016) in text classification models and kept improving accuracy. Recently, graph based text classification models received growing attention due to its ability to model global information in corpus (Yao et al., 2019; Peng et al., 2018; Zhang et al., 2020b; Nikolentzos et al., 2019). Our paper follows this line of works on developing novel GNN for text classification.

4.2 Graph Neural Network

Representative examples of GNN models proposed by present include GCN (Kipf and Welling, 2016), Graph Attention Network (GAT)(Veličković et al., 2017) and Graph SAGE (Hamilton et al., 2017). GCN models are based on approximated graph convolutional operator while GAT relies on self-attention mechanism. Recently, *Transformer* (Vaswani et al., 2017) models have been applied in novel GNN designs (Hu et al., 2020; Zhang et al., 2020a).

5 Conclusion

In this paper, we proposed a scalable heterogeneous graph model, TG-Transformer, for text classification. Experimental results prove its effectiveness and efficiency compared to state-of-the-art methods. It also enables parallelization and pre-training in GNN models for further research.

Acknowledgement

This work is partially supported by NSF through grant IIS-1763365 and by FSU.

References

Zhengping Che, David Kale, Wenzhe Li, Mohammad Taha Bahadori, and Yan Liu. 2015. Deep computational phenotyping. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 507–516.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, pages 2704–2710.

Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng WANG. 2019. Text level graph neural network for text classification. *arXiv preprint arXiv:1910.02356*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. 2019. Text classification algorithms: A survey. *Information*, 10(4):150.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. 2020. Tensor graph convolutional networks for text classification. *arXiv preprint arXiv:2001.05313*.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023.

Giannis Nikolentzos, Antoine J-P Tixier, and Michalis Vazirgiannis. 2019. Message passing attention networks for document understanding. *arXiv preprint arXiv:1908.06267*.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference*, pages 1063–1072.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*, pages 90–94. Association for Computational Linguistics.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377.

Jiawei Zhang and Lin Meng. 2019. Gresnet: Graph residual network for reviving deep gnns from suspended animation. *ArXiv, abs/1909.05729*.

Jiawei Zhang, Haopeng Zhang, Li Sun, and Congying Xia. 2020a. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*.

Jinghe Zhang, Kamran Kowsari, James H Harrison, Jennifer M Lobo, and Laura E Barnes. 2018. Patient2vec: A personalized interpretable deep representation of the longitudinal electronic health record. *IEEE Access*, 6:65333–65346.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Yin Zhang, Rong Jin, and Zhi-Hua Zhou. 2010. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52.

Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. 2020b. Every document owns its structure: Inductive text classification via graph neural networks. *arXiv preprint arXiv:2004.13826*.