Locality-Aware Tail Node Embeddings on Homogeneous and Heterogeneous Networks

Zemin Liu, Yuan Fang, Wentao Zhang, Xinming Zhang, Steven C.H. Hoi, *Fellow, IEEE*

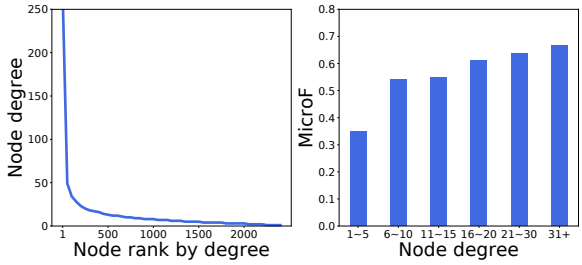
***Abstract*—While the state-of-the-art network embedding approaches often learn high-quality embeddings for high-degree nodes with abundant structural connectivity, the quality of the embeddings for low-degree or *tail* nodes is often suboptimal due to their limited structural connectivity. While many real-world networks are long-tailed, to date little effort has been devoted to tail node embeddings. In this paper, we formulate the goal of learning tail node embeddings as a *few-shot regression* problem, given the few links on each tail node. In particular, since each node resides in its own local context, we personalize the regression model for each tail node. To reduce overfitting in the personalization, we propose a *locality-aware* meta-learning framework, called *meta-tail2vec*, which learns to learn the regression model for the tail nodes at different localities. Moreover, to address the heterogeneity in nodes and edges on heterogeneous information networks (HINs), we further extend the proposed model and formulate *meta-tail2vec+*, which is based on a dual-adaptation mechanism to facilitate the locality-aware tail node embeddings on HINs. Finally, we conduct extensive experiments and demonstrate the promising results of both meta-tail2vec and its extension meta-tail2vec+.**

***Index Terms*—Meta-learning, locality-aware, tail node embeddings, homogeneous and heterogeneous networks.**

I. Introduction

NETWORK structures are prevalent in various real-world scenarios, such as social networks, citation networks and biological networks. On these networks, many problems can be formulated as node classification and link prediction, which rely heavily on effective network representations. While traditional approaches mainly focus on manual feature engineering, recent network embedding [1] and graph neural networks [2] have become the *de facto* state-of-the-art approaches for learning representations of network data. Specifically, these methods aim to encode network structures by mapping nodes into a low-dimensional vector space, in which the structural information is preserved.

Despite their success, a critical question remains open: the performance of most existing embedding methods depends on the availability of abundant structural connectivity. Although this is not a concern for medium-to-high degree nodes with many links to other nodes, low-degree nodes with very few links often suffer from the problem of limited structural information. As a result, the embedding vector for a low-degree node cannot be accurately learned from its structural information. Generally, the node degrees vary considerably across the network and are not uniformly distributed. In many networks, the degrees approximately follow the power-law distribution [1]. For instance, Fig. 1(a) illustrates the degree distribution of the Wiki network [3], a network of interlinked Wikipedia pages belonging to 19 categories. The node degrees are characterized by a long-tailed distribution, where a significant fraction of the nodes belong to the tail with very low degrees. The embeddings of these *tail* nodes are unsatisfactory, as demonstrated by the performance of node classification in Fig. 1(b). Particularly, when the node degree decreases, the performance also decreases due to less structural information.



(a) Degree distribution (b) Classification performance

Fig. 1: Distribution of node degree and its relationship to the quality of embedding vector on the Wiki network.

Unfortunately, most current network embedding or graph neural network approaches overlook the tail nodes, by regarding all nodes uniformly and adopting the same learning approach despite their diverse degrees. For example, DeepWalk [1] samples node sequences from the network and feeds them to the same skip-gram model without paying special attention to low-degree nodes, while graph neural networks [2], [4] derive the embedding of a target node by aggregating its neighboring nodes in the same manner without adapting to the degree of the target node. Not surprisingly, the tail nodes with very few links are usually under-modeled than those with many links. This inspires us to investigate the following research problem: *How do we learn effective embedding vectors for tail nodes from limited structural information?*

The problem is challenging for three reasons. (1) The tail

Z. Liu is with National University of Singapore. Work was done as a research scientist at Singapore Management University. E-mail: zeminliu@nus.edu.sg.

Y. Fang is with Singapore Management University, Singapore. E-mail: yfang@smu.edu.sg.

W. Zhang is with WeBank, China. Part of the work was done as a visiting research student at Singapore Management University. E-mail: wyattzhang@webank.com.

X. Zhang is with University of Science and Technology of China, China. E-mail: xinming@ustc.edu.cn.

S. Hoi is with Singapore Management University, Singapore. E-mail: chhoi@smu.edu.sg.

(Y. Fang and X. Zhang are the co-corresponding authors.)

Manuscript received 16 March 2022; revised 14 July 2023; accepted 26 August 2023.

nodes have very few links, which provide *scarce structural information*. The issue is especially severe when only network structures are available, without access to additional side information such as item contents in recommendation [5] and morphological information in word embedding [6]. (2) Each node is associated with a unique *locality*, and thus the embedding model should ideally become specialized from node to node. However, in practice this presents a dilemma: adapting to each node often causes overfitting, which is particularly serious for tail nodes with very few observed links. (3) When it comes to heterogeneous information networks (HINs) [7], the node heterogeneity further complicates the capture of limited structural information on tail nodes. That is, we need to simultaneously account for both the locality and heterogeneity of tail nodes given scarce structural information.

To address the first challenge of scarce structural information, we exploit the embeddings of nodes with a sufficient number of links (*i.e.*, non-tail nodes and we call them *head* nodes hereafter). Considering any network embedding model, the learned embedding vectors of head nodes tend to be more accurate than those of tail nodes, due to more abundant structural information associated with head nodes. Thus, we treat the embeddings of head nodes as our *oracle embeddings*, which can be leveraged to train a regression model capable of *reconstructing* the oracle embeddings in a self-supervised manner. Using the regression model, our goal is to predict new embeddings for tail nodes such that their quality can approach that of the oracle embeddings. To further ensure that the regression model fits both head and tail nodes, we propose to perform *link dropouts* on head nodes, to simulate the limited structural information of tail nodes.

To address the second challenge of adapting to the locality of each node, we resort to the meta-learning paradigm [8], [9], for its ability of adapting to new learning tasks (also called episodes) by learning a transferable prior common to different tasks. Known as *meta-learning*, it learns how to learn a model in the form of a prior, contrary to learning one model for all tasks (which cannot differentiate tasks) or individual models for each task (which may overfit to each task). In particular, we adopt the meta-learning framework MAML [9], which has the advantage of quick adaptation in a model-agnostic manner and has achieved considerable success in various problem domains [10], [11].

To realize the above insights in our scenario, we construct *locality-aware* tasks: one task for each node to represent its locality, and learn a regression prior that can be easily adapted to each task or locality. In each task, given a *query* node, we aim to predict its embedding vector after training on a set of *support* nodes. While the support set is often randomly sampled, for locality-aware learning, we propose to use the neighbors of the query node as the support nodes, assuming that the localities of neighboring nodes are similar. More specifically, during meta-training, each task involves a head node as the query, with the objective of learning a regression prior such that the prior can be adapted by the support nodes to accurately reconstruct the oracle embeddings of the query. During meta-testing, each task involves a tail node as the query, and the learned prior will be adapted by the support nodes before predicting a better embedding vector for the query. To make the meta-training and meta-testing tasks more similar, we again adopt link dropouts to sample only a few neighbors as the support set during meta-training, to simulate meta-testing tasks on tail nodes. Essentially, each task is a *few-shot* regression problem.

To address the third challenge of node heterogeneity on HINs, building upon the preceding two mechanisms of oracle embedding reconstruction and locality adaptation, we further account for and differentiate the node types. First, for oracle embedding reconstruction, we extend the regression model by further distinguishing neighboring nodes in terms of their types. Thus, the node heterogeneity can be integrated into the reconstruction process. Second, for preserving the local contexts of each node, nodes of the same type tend to share a common "schema" of locality adaptation. Thus, to absorb the heterogeneity into localization, we propose a dual-adaptation mechanism for nodes on a HIN w.r.t. both their node type and locality, respectively performing the *type*- and *node*-level adaptations. For type-level adaptation, we modulate the prior w.r.t. the node type of a target node by *scaling* and *shifting* operations [12], [13]. For node-level adaptation, we still resort to the same meta-learning (*i.e.*, MAML) paradigm to capture the individual locality of each node.

In summary, we propose a novel approach *meta-tail2vec* and its extension on HINs *meta-tail2vec+*, which learn to learn tail node embeddings in a few-shot regression setting. The models operate in an embedding-agnostic manner, which is flexible to work with any network embedding model. Specifically, our contributions can be summarized as follows. (1) We formulate the novel problem of learning tail node embeddings on networks, and cast it as an instance of regression via oracle reconstruction. (2) We propose a base regression model hinged on the concept of link dropouts, and formulate the locality-aware tasks on networks in a meta-learning framework, which allows for easy local adaption of the base model. (3) We further extend the proposed model to HINs and formulate meta-tail2vec+, hinged on a dual-adaptation mechanism for both type- and node-level adaptations, to capture both the heterogeneity and locality of a target node. (4) We conduct extensive experiments on five public datasets, including three homogeneous graphs and two HINs, in which both meta-tail2vec and meta-tail2vec+ achieve significant performance gains.

A preliminary version of this paper has been published as a conference paper in CIKM'2020 [14]. We summarize the main changes as follows. (1) *Introduction*: We reorganized Section I to highlight the motivation, challenges, and insights to cope with HINs for tail node embeddings. (2) *Related Work*: We reviewed more related work especially on HINs in Section II, to present a detailed background on HINs. (3) *Model*: We extended meta-tail2vec to meta-tail2vec+ in Section IV-D, based on heterogeneity-aware neighborhood aggregation and a novel mechanism of dual-adaptation. (4) *Experiments*: We conducted extensive experiments to evaluate meta-tail2vec+ in Section VI, and analyzed the results in detail.

II. Related Work

Network embedding [15] has been popular for graph representation learning, and various approaches have been proposed to generate structure-preserving node embeddings, such as random walks [1], [16], matrix factorization [17], [18], node proximity [19], and motifs or subgraphs [20], [21]. Graph neural networks (GNNs) [2], [4], [13] emerge as another powerful tool for representation learning, which are designed to integrate both node features and network structures. To capture the heterogeneity on HINs, an increasing number of recent studies investigate the heterogeneous network embeddings [7], [22], which usually embody the model by capitalizing on type-based meta-patterns, such as metapath [23], [24], metagraph [21], [25], *etc.* The widespread application of GNNs opens great opportunities for representation learning on HINs, thus several contributions of HIN-based GNNs [26]–[28] have also been devoted.

However, little attention has been paid to the learning of tail node embeddings. While several recent studies explore various techniques to deal with sparse networks, such as using dual dropouts to avoid overfitting [29], adopting the adversarial principle to learn the underlying distribution [30], [31] and leveraging the global network structure [32], they aim to increase the overall robustness and do not specifically target the most vulnerable tail nodes. Meanwhile, some studies deal with sparse observations in other domains, such as cold-start recommendation [5], [33], [34] and rare word embedding [6], [10], [35]. However, these methods rely on additional side information, such as item contents or word morphology. Though Tail-GNN [36] also addresses the same problem, it cannot cope with the tail node embeddings on heterogeneous graphs.

Several recent studies [37], [38] investigate the degree-related discrimination of nodes, and process nodes differently w.r.t. their degrees. However, they still focus on the overall performance, and do not particularly aim to improve the performance of tail nodes, which belong to a more challenging group. There have also been explorations of degree-related fairness learning on graphs [39]–[41], a concept distinct from our tail node embedding task. They primarily aim to achieve fair predictions across different groups by treating node degrees as a sensitive attribute. These studies typically employ pre-defined fairness metrics to balance loss discrepancy [39], accuracy discrepancy [40], or prediction discrepancy [41] across groups. In contrast, our work on tail node embedding specifically focuses on improving the performance of nodes with low degrees, rather than reducing discrepancies across groups. Besides, imbalanced classification has also attracted much interest in several research areas such as vision [42]–[44] and graph [45]–[47]. However, imbalanced classification in graphs aims to improve the performance of minority classes, which are characterized by having fewer labeled data compared to the majority classes. In contrast, tail node embedding aims to enhance the performance of tail nodes, which are characterized by their low degree rather than class membership.

To address the general problem of learning from less data, particularly in few-shot scenarios, meta-learning [9], [48], [49] has demonstrated considerable success in several domains including vision [48], language [10], [50] and data mining [11], [51]. These methods typically learn some prior knowledge from an abundant number of related tasks, and adapt the prior to new tasks with limited data. Few-shot learning on graphs has also been explored, including node classification on a single large graph [52], [53], and node or graph-level classification on multiple graphs [54], [55]. These methods primarily aim to achieve favorable performance on novel classes with limited labeled data, rather than to address the improvement of low-degree nodes. To the best of our knowledge, our approach is the first use of meta-learning for tail node embedding.

III. Problem Casting as Regression

We begin with the problem statement, and cast it as an instance of regression based on the idea of oracle reconstruction.

A. Problem Statement

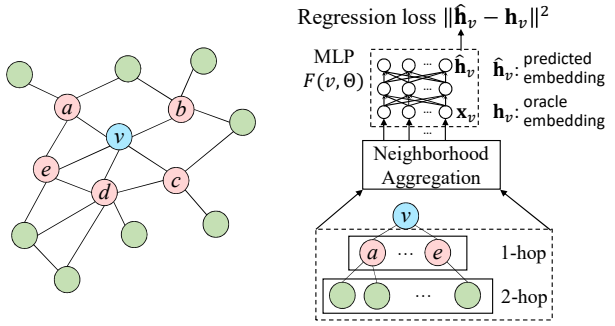
In this paper, we focus on tail node embeddings for both homogeneous and heterogeneous networks. Without loss of generality, we first give a unified definition for both of them. Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R})$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges, and $\mathcal{T}$ and $\mathcal{R}$ are the node and edge types, respectively. In particular, there exist a function $\varphi_{\mathcal{V}} : \mathcal{V} \to \mathcal{T}$ to map a node $v \in \mathcal{V}$ to its node type $\varphi_{\mathcal{V}}(v) \in \mathcal{T}$, and a function $\varphi_{\mathcal{E}} : \mathcal{E} \to \mathcal{R}$ to map an edge $e \in \mathcal{E}$ to its edge type $\varphi_{\mathcal{E}}(e) \in \mathcal{R}$. For a homogeneous graph, we have $|\mathcal{T}| = 1$ and $|\mathcal{R}| = 1$; while for a heterogeneous graph, we have $|\mathcal{T}| + |\mathcal{R}| > 2$.

Let $\mathcal{N}_v$ denote the set of neighbors of node $v \in \mathcal{V}$, and the size of the neighbor set $|\mathcal{N}_v|$ is known as the degree of v. In particular, v is a low-degree or *tail* node if its degree is no larger than some constant k. That is, the set of tail nodes is $\mathcal{V}_{\text{tail}} = \{v \in \mathcal{V} : |\mathcal{N}_v| \leq k\}$. We call the remaining nodes *head* nodes: $\mathcal{V}_{\text{head}} = \{v \in \mathcal{V} : |\mathcal{N}_v| > k\}$.

Given any base network embedding model ϕ, we denote $\mathbf{h}_v = \phi(\mathcal{G}, v) \in \mathbb{R}^d$ as the d-dimensional embedding vector of node v. Since the tail nodes have scarce structural features due to their small degrees, presumably their embeddings $\{\mathbf{h}_v : v \in \mathcal{V}_{\text{tail}}\}$ are inferior in quality to the embeddings of the head nodes $\mathcal{O} = \{\mathbf{h}_v : v \in \mathcal{V}_{\text{head}}\}$. Our goal is to learn new embedding vectors for the tail nodes $\{\hat{\mathbf{h}}_v : v \in \mathcal{V}_{\text{tail}}\}$, such that their quality is improved to eventually approach the quality of the head nodes' embeddings $\mathcal{O}$. Note that our setup is embedding-agnostic, *i.e.*, any embedding model ϕ can be used as the base embedding model.

B. Regression via Oracle Reconstruction

A major challenge of learning tail node embeddings lies in the limited structural information. That is, each tail node only has a few observed neighbors. Without assuming additional side information (*e.g.*, item contents in recommendation systems), we take advantage of the high-quality embeddings of head nodes, and investigate how high-quality embeddings can be similarly constructed for the tail nodes.



(a) 2-hop neighbors of node v (b) Embedding regression model

Fig. 2: Training a regression model to predict node embeddings reconstructed from neighboring nodes.

Specifically, we propose to cast the problem as an instance of regression. We train a regression model F on the head nodes, treating their embeddings $\mathcal{O}$, which are learned by the base embedding model ϕ, as the *oracle embeddings* w.r.t. ϕ. Given any head node v, $F(v;\Theta)$ outputs a new predicted embedding $\hat{\mathbf{h}}_v$ to approximate its oracle embedding $\mathbf{h}_v \in \mathcal{O}$. In other words, F is expected to *reconstruct* the oracle embeddings $\mathcal{O}$ of the head nodes, regardless of how the base embedding model works. Formally, we optimize the parameters Θ of the regression model F by

$$\arg\min_\Theta \sum_{v\in\mathcal{V}_{\text{head}}} \|F(v;\Theta) - \mathbf{h}_v\|^2. \tag{1}$$

Note that here we use the Euclidean norm $\|\cdot\|$, although other distance functions can also be adopted.

After training, F can be applied to the tail nodes as test instances, so as to predict their unknown oracle embeddings, supposedly attaining similar quality as $\mathcal{O}$. The improved embedding vectors of tail nodes can be subsequently used as representations for downstream tasks such as node classification and link prediction for better performance.

IV. Meta-Learned Few-Shot Regression

In this section, we first introduce a regression model to predict new embedding vectors for tail nodes, then personalize the regression for each node by formulating node-wise locality-aware tasks, and present the overall meta-learning approach *meta-tail2vec*. Furthermore, we extend *meta-tail2vec* to *meta-tail2vec+* to handle the heterogeneity on HINs.

A. Embedding Regression Model

As shown in Fig. 2, we materialize the regression model $F(v;\Theta)$ to reconstruct the oracle embedding $\mathbf{h}_v$ of a node v, with the widely used Multi-Layer Perceptron (MLP):

$$\hat{\mathbf{h}}_v = F(v;\Theta) = \mathbf{W}_2 \cdot \sigma(\mathbf{W}_1\mathbf{x}_v + \mathbf{b}_1) + \mathbf{b}_2, \tag{2}$$

where $\mathbf{x}_v \in \mathbb{R}^{d_1}$ is the input feature vector of node v. The parameters of the MLP include $\mathbf{W}_1 \in \mathbb{R}^{d_2\times d_1}$, $\mathbf{W}_2 \in \mathbb{R}^{d\times d_2}$, $\mathbf{b}_1 \in \mathbb{R}^{d_2}$ and $\mathbf{b}_2 \in \mathbb{R}^d$, *i.e.*, $\Theta = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2\}$. $\sigma(\cdot)$ is an activation function, and we adopt ReLU in this paper. Note that the size of the output layer is d, the same as the embedding dimension of the nodes. For a node v, the input to the MLP is a feature vector $\mathbf{x}_v$. In the following, we discuss the formulation of the input feature vector.

1) Neighborhood Aggregation: On a network, a node v is characterized by its context or its neighbors set, $\mathcal{N}_v$. Naturally, we can aggregate the embeddings of the neighbors to construct its input feature $\mathbf{x}_v$, an idea similar and central to many GNNs [2], [4], as

$$\mathbf{x}_v = \textsc{Aggr}(\{\mathbf{h}_i : i \in \mathcal{N}_v\}), \tag{3}$$

where $\textsc{Aggr}(\cdot)$ is an aggregator, *e.g.*, mean pooling. Generally, we can also aggregate nodes within m hops of v, as shown in Fig. 2. Defining the m-hop neighbor set of v as

$$\mathcal{N}_v^{(m)} = \bigcup\nolimits_{i\in\mathcal{N}_v^{(m-1)}} \mathcal{N}_i, \tag{4}$$

and $\mathcal{N}_v^{(1)} \equiv \mathcal{N}_v$, the input feature $\mathbf{x}_v$ can be constructed as

$$\mathbf{x}_v = \textsc{Aggr}(\{\mathbf{h}_i : i \in \mathcal{N}_v^{(1)} \cup \mathcal{N}_v^{(2)} \cup \ldots \cup \mathcal{N}_v^{(m)}\}). \tag{5}$$

Note that, although advanced strategies such as tree-LSTMs [56] and graph convolutions [2] may be employed to aggregate multiple hops, we adopt the simple mean pooling to demonstrate the effectiveness of our approach. In this specific strategy, the dimension of the input feature vector $\mathbf{x}_v$ is the same as the node embedding dimension, *i.e.*, $d_1 = d$.

2) Link Dropouts: One major flaw of such an input vector $\mathbf{x}_v$ is that the head nodes in training and the tail nodes in testing possess very different neighbor sets in terms of their abundance, *i.e.*, $|N_v| \gg |N_u|$ for some $v \in \mathcal{V}_{\text{head}}$ and $u \in \mathcal{V}_{\text{tail}}$. To make the training and testing nodes more similar, we perform *link dropouts* on the head nodes. Specifically, we randomly sample only k neighbors of each head node for aggregation, to simulate the tail nodes. That is, given the sampled neighbors $\widetilde{\mathcal{N}}_v$ of a head node v, for one-hop aggregation we have

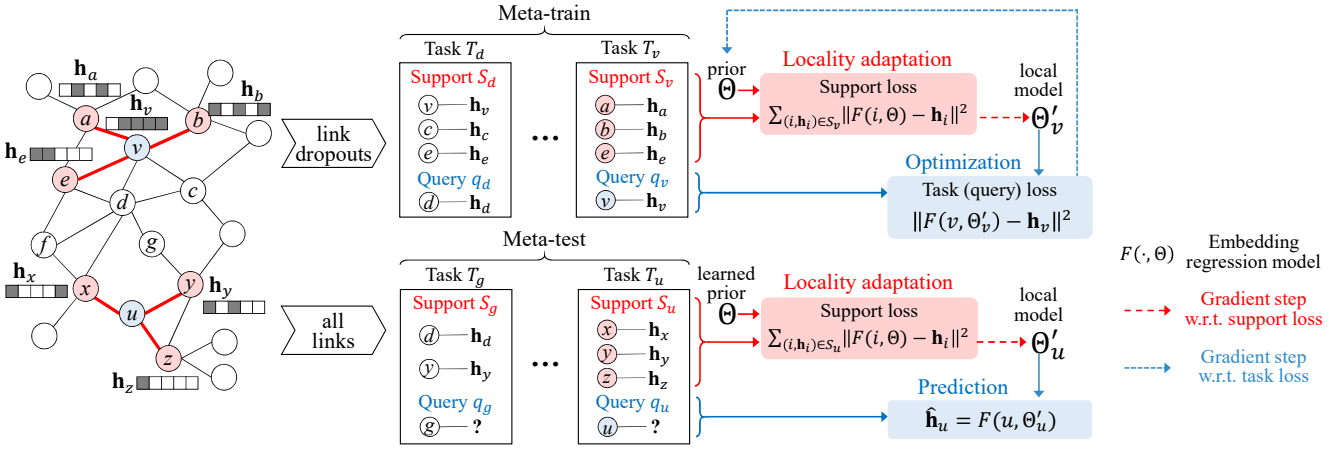
$$\mathbf{x}_v = \textsc{Aggr}(\{\mathbf{h}_i : i \in \widetilde{\mathcal{N}}_v\}), \quad \forall v \in \mathcal{V}_{\text{head}}. \tag{6}$$

The idea of link dropouts draws an interesting parallel to DropoutNet [33]. However, we do not assume any side information employed in DropoutNet like user contents, which means our problem is more challenging and we can only drop the structural information partially.

B. Locality-Aware Few-Shot Regression Tasks

When applying the regression model on all nodes, it ignores the unique locality of each node. As nodes reside across different localities on the network, assuming one global model to fit all nodes is unrealistic. At the other extreme, learning an individual model for each node, including the popular pre-training and fine-tuning strategy [57], is likely to cause severe overfitting due to the limited structural information at the locality of each tail node.

1) Locality-Aware Support Sets: To address the challenge of adapting to the locality of each node, we resort to the episodic meta-learning paradigm [8], [9]. The framework consists of many similar-natured learning tasks, divided into *meta-training* and *meta-testing* tasks. While each task is an instance in the meta-learning process, each task itself is a learning problem consisting of *support* and *query* sets (representing the usual sense of training and testing data, respectively). The



(a) Toy network (b) Locality-aware tasks (c) Locality-aware meta-learning

Fig. 3: Overall framework of our locality-aware tail node embedding model meta-tail2vec. (Best viewed in color.)

goal of meta-learning is to extract prior knowledge common to all tasks in meta-training, such that the knowledge can be quickly adapted to new tasks in meta-testing. In other words, it learns how to learn a task in the form of a common prior, instead of directly learning each task.

In our context, each task represents the unique locality of a node. At the task level, given a *query* node, we aim to predict its embedding vector after training on a set of *support* nodes. At the meta-learning level, we learn a prior regression model F parameterized by Θ from the meta-training tasks where the query of each task is a head node, and further adapt the prior to the learning of new tasks in meta-testing where the query of each task is a tail node. However, in traditional meta-learning tasks [9], [52], the support set for a query is randomly sampled. In such random sampling, the support set is not related to the query, and thus cannot reflect the unique locality of each query node. To generate locality-aware tasks, we propose to use the neighbors of the query node as the support nodes. The assumption is that the localities of neighboring nodes are similar, and thus training on these support nodes would be also applicable to the query node which lies in the vicinity of the support nodes. In other words, in each task, the support and query nodes are coupled based on their locality.

2) Formulation of Few-Shot Tasks: As the neighbor sets of head and tail nodes differ vastly in size, a meta-training task with a head node as the query has many support nodes, while a meta-testing task with a tail node as the query has few support nodes. To make the tasks more similar, we again adopt link dropouts by only sampling k neighbors as the support set for meta-training tasks to simulate meta-testing tasks. Thus, each task becomes a *few-shot* (up to k shots) regression problem, to predict the embedding vector of a query node from a few (up to k) support nodes.

We illustrate the task formulation with an example. As shown in Fig. 3(a), assuming $k = 3$, v and u is a head and tail node, respectively. On the one hand, the head node v will be used to formulate a meta-training task: v itself will be the query node, whereas we sample k nodes from v's neighbors $\mathcal{N}_v = \{a, b, c, d, e\}$, to form the support set $\widetilde{\mathcal{N}}_v$, say, $\widetilde{\mathcal{N}}_v = \{a, b, e\}$. On the other hand, the tail node u will be used to formulate a meta-testing task: u itself will be the query node, whereas we simply take all of u's neighbors $\mathcal{N}_u = \{x, y, z\}$ as the support set. More example tasks are illustrated in Fig. 3(b).

Formally, for each head node v, we define a meta-training task $T_v = (S_v, q_v)$ where $S_v = \{(i, \mathbf{h}_i) : i \in \widetilde{\mathcal{N}}_v\}$ is the support set, and $q_v = (v, \mathbf{h}_v)$ is the query. For each tail node u, we define a meta-testing task $T_u = (S_u, q_u)$ where $S_u = \{(i, \mathbf{h}_i) : i \in \mathcal{N}_u\}$ is the support set, and $q_u = (u, ?)$ is the query, where the embedding vector of u is unknown and to be predicted. Thus, the set of all meta-training tasks is $T_{\text{train}} = \{(S_v, q_v) : v \in \mathcal{V}_{\text{head}}\}$, and the set of all meta-testing tasks is $T_{\text{test}} = \{(S_u, q_u) : u \in \mathcal{V}_{\text{tail}}\}$.

In practice, we further ensure that all nodes in the support sets are head nodes, so that they all associate with an oracle embedding for adaptation to the regression model. Specifically, in meta-training, we only sample head nodes from the neighbors as the support; in meta-testing, we filter tail nodes from the support. In the rare case that all of a tail node's neighbors are also tail nodes, we will not attempt to improve its original embedding.

C. Meta-Learning of Tail Node Embeddings

We employ MAML [9] for the meta-learning of tail node embeddings, which is capable of learning a prior Θ for any model using gradient-based optimization. In our case, the prior is the embedding regression model F, parameterized by Θ. Different from the simple pre-training of a model, the prior Θ is learned in such a way that Θ can be quickly adapted to a new task by performing just one or a few gradient updates on the support set of the new task. The model Θ' adapted from the prior Θ, is a local model for the query node in the same task.

More specifically, in our meta-training, consider a task $T_v = (S_v, q_v)$. As shown in Figs. 3(b) and (c), the prior Θ itself is not directly updated or optimized by the support set S_v. Instead, it will be adapted by S_v to produce a local model Θ'_v for the query v, through one or a few gradient updates

w.r.t. S_v's loss. The adapted local model Θ'_v will be applied to the query v to predict an embedding vector $\hat{\mathbf{h}}_v$, so that the prior Θ can be updated by minimizing the task loss, *i.e.*, the distance between the predicted embedding $\hat{\mathbf{h}}_v$ and the oracle embedding $\mathbf{h}_v$ of query node v.

Formally, let the loss of the prior on the support set S_v be

$$L_{S_v}(\Theta) = \sum_{(i,\mathbf{h}_i)\in S_v} \|F(i;\Theta) - \mathbf{h}_i)\|^2. \tag{7}$$

The prior Θ will be adapted by S_v by one (or a few) gradient updates to generate a local model Θ'_v for the task T_v, as,

$$\Theta'_v = \Theta - \alpha \frac{\partial L_{S_v}(\Theta)}{\partial \Theta}, \tag{8}$$

where α is the learning rate for the adaptation. Afterwards, the local model Θ'_v will be applied on the query v, to calculate the task loss using the query q_v:

$$L_{q_v}(\Theta'_v) = \|F(v;\Theta'_v) - \mathbf{h}_v\|^2. \tag{9}$$

Subsequently, the prior Θ can be updated during meta-training by minimizing the query-based total loss of all meta-training tasks. Given the set of meta-training tasks T_{train}, we optimize the following to obtain the optimal prior:

$$\arg\min_\Theta \sum_{T_v=(S_v,q_v)\in T_{\text{train}}} L_{q_v}\left(\Theta - \alpha \frac{\partial L_{S_v}(\Theta)}{\partial \Theta}\right). \tag{10}$$

Furthermore, in meta-testing, consider a task $T_u = (S_u, q_u)$. The prior Θ, learned from meta-training, will be adapted on the support set S_u to produce a local model Θ'_u in the same way as Eq. (8). And Θ'_u will be simply applied on the query u, which is a tail node, to predict a new embedding $\hat{\mathbf{h}}_u = F(u;\Theta'_u)$ as the output of meta-tail2vec.

Complexity of data access. As training is performed over mini-batches of tasks, the time cost of our meta-training process depends on the total number of tasks N passed through, where each task contains up to k support nodes and the embedding of each node is aggregated from m-hop neighbors. Thus, the overall complexity is $O(Nkn^m)$, where n is the average degree of nodes. Typically m is a small constant such as 1 or 2, and k is also small by the definition of tail nodes. Furthermore, it is also common to perform neighborhood sampling [4] during the m-hop aggregation, and thus the average degree n is also effectively restricted to a constant. Note that the complexity analysis here is conducted from the perspective of data access. To provide a more comprehensive understanding, we will extend this analysis to include the complexity associated with representation computation in Sect. IV-E.

D. Extension to HINs: meta-tail2vec+

For tail node embeddings on heterogeneous graphs, we further extend meta-tail2vec onto HINs to cope with the heterogeneity, and propose *meta-tail2vec+*. We follow the same paradigm of meta-tail2vec for tail node embedding on HINs, but introduce improvements to the two main modules, *i.e.*, embedding regression based on neighborhood aggregation, and locality-aware meta-learning, to handle the heterogeneous types on HINs. First, for the embedding regression of a target node, the types of its m-hop neighbors are taken into consideration to discriminate their different roles. Second, for locality-aware meta-learning, we propose a dual-adaptation mechanism that extends the node-level adaption with an additional type-level adaption.

1) Heterogeneity-Aware Neighborhood Aggregation: On a HIN, each node is structurally characterized by its context (*e.g.*, neighbors within m hops). Particularly, neighboring nodes of different types usually contribute differently, thus requiring non-uniform treatment w.r.t. their types during neighborhood aggregation. As a consequence, different from Eq. (5) for homogeneous graphs, given a target node v on a HIN, we first perform a type-wise aggregation on its neighboring nodes of the same type τ to obtain $\mathbf{x}_v^\tau$. Then, we aggregate all the types to generate a heterogeneity-aware feature vector $\mathbf{x}_v^+ \in \mathbb{R}^{d_1}$ for v, which can serve as the input for the embedding regression model in Eq. (2). Formally, the input feature vector $\mathbf{x}_v^+$ of node v can be calculated as

$$\mathbf{x}_v^+ = \textsc{Aggr}(\{\mathbf{W}^\tau \mathbf{x}_v^\tau : \forall \tau \in \mathcal{T}\}), \tag{11}$$

$$\text{where} \quad \mathbf{x}_v^\tau = \textsc{Aggr}(\{\mathbf{h}_i : i \in \mathcal{N}_v^m \wedge \varphi_\mathcal{V}(i) = \tau\}). \tag{12}$$

Here $\tau \in \mathcal{T}$ is a node type, $\mathcal{N}_v^m = \mathcal{N}_v^{(1)} \cup \mathcal{N}_v^{(2)} \cup \ldots \cup \mathcal{N}_v^{(m)}$ is the set of v's neighboring nodes within m hops, and $\Theta^\mathcal{T} = \{\mathbf{W}^\tau \in \mathbb{R}^{d_1 \times d_1} : \forall \tau \in \mathcal{T}\}$ is the set of learnable weights across all the types $\mathcal{T}$. Under this formulation[1], the heterogeneity of the neighboring nodes is captured, thereby characterizing the heterogeneous semantics of the target node.

Subsequently, we also utilize the same regression model in Eq. (2) to further materialize few-shot regression, which will be illustrated in Sect. IV-D2.

2) Dual-Adaptation based Few-Shot Regression: As discussed in Sect. IV-B, we construct a group of meta-learning tasks for locality-aware adaptation. Compared to homogeneous graphs, on a HIN it is crucial to also account for the heterogeneity in the adaptation mechanism. Naturally, the node types should be taken into account when constructing a meta-learning task, to facilitate a more consistent adaptation from the support set to the query. The type-aware tasks are further equipped with a dual-adaptation mechanism, which are adapted not only at the node level to tailor to the tail nodes, but also at the type level to suit different types of tail nodes.

Type-aware task construction. First, we leverage nodes of the same type to build the support and query set in each meta-learning task, based on the intuition nodes of the same type tend to share the same role or underlying pattern. Thus, the support set that shares the same node type with the query could provide a more consistent reference for the model adaptation.

Formally, for each head node $v \in \mathcal{V}_{\text{head}}$ on a HIN, we define a meta-training task $T_v^+ = (S_v^+, q_v^+)$ such that $S_v^+ = \{(i, \mathbf{h}_i) : i \in \widetilde{\mathcal{N}}_v \wedge \varphi_\mathcal{V}(i) = \varphi_\mathcal{V}(v)\}$ is the support set, and $q_v^+ = (v, \mathbf{h}_v)$ is the query. That is, all the nodes in support set S_v^+ and query q_v^+ share the same type, *i.e.*, $\varphi_\mathcal{V}(v)$. Similarly, for a tail node $u \in \mathcal{V}_{\text{tail}}$, we define a meta-testing task $T_u^+ = (S_u^+, q_u^+)$, where

[1]Only aggregation by node types are illustrated here, while it is also possible to extend to edge types (if any) by aggregating neighboring nodes based on their edge type to the target node. For brevity, we only focus on the node types in the technical discussion.

$S_u^+ = \{(i, \mathbf{h}_i) : i \in \mathcal{N}_u \wedge \varphi_{\mathcal{V}}(i) = \varphi_{\mathcal{V}}(u)\}$ is the support set, and $q_u^+ = (u, ?)$ is the query. In both meta-training and testing tasks, as the support set is sampled from the one-hop neighbors for locality awareness (see Sect. IV-B1), it could lead to the scenario where all one-hop neighbors belong to different types from the target node. In such a case, we can extend the definition of support set to include two-hop neighbors. Finally, we can construct a set of meta-training tasks $T^+_{\text{train}} = \{(S_v^+, q_v^+) : v \in \mathcal{V}_{\text{head}}\}$, and a set of meta-testing tasks $T^+_{\text{test}} = \{(S_u^+, q_u^+) : u \in \mathcal{V}_{\text{tail}}\}$.

Type-level adaptation. On a homogeneous graph, recall that we only adapt the prior Θ into a local model Θ'_v for each task T_v, which is a node-level adaptation as we attempt to adapt the task to the locality of its query node v. However, given type-aware meta-learning tasks on a HIN, it is insufficient to perform only node-level adaptation, which overlooks the type heterogeneity across meta-learning tasks. To address this, we propose a dual-adaptation mechanism for type-aware meta-learning tasks at both the type and node levels.

We start with the type-level adaptation. MAML could potentially be utilized at the type level too, although it might prove challenging to apply in this context. Fundamentally, MAML aims to transfer a prior learned from a "base set" to a "novel set". In the vanilla meta-tail2vec, the base set involves the head nodes in meta-training, and the novel set involves the tail nodes in meta-testing. However, when dealing with the heterogeneous types, it is unclear what could be the base set and novel set, since the number of types is often small and they are known in advance on a typical heterogeneous graph. Furthermore, to construct the support sets, some labeled data are still needed. In the vanilla meta-tail2vec, the high-quality embeddings of the head nodes are treated as the oracle embeddings, providing ground-truth labels. In contrast, given heterogeneous types, there are no clear oracle embeddings associated with a type. Hence, MAML is not an ideal choice here. Instead, we resort to Feature-wise Linear Modulation (FiLM) [12], [13] for the type-level adaptation, which does not require a base set and a novel set, or the construction of support sets.

Specifically, given a type-aware task $T_v^+ = (S_v^+, q_v^+)$, we first apply a FiLM layer to customize it to local types associated with the task. More concretely, the FiLM layer modulates the parameters of the shared regression model in Eq. (2), conditioned on the type signature of the task. Modulating a shared model is a better choice than fitting individual models to distinct type signatures. On one hand, sharing a common regression model across tasks can capture the generality, while the modulation conditioned on type signatures empowers the type-level adaptation to capture heterogeneous semantics on a HIN. On the other hand, fitting individual models not only ignores the commonality among tasks, but also tends to overfit to individual type signature and incurs a significant overhead.

Common modulation operators include scaling and shifting, which are applied to all the learnable parameters of the regression model, namely, $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1$ and $\mathbf{b}_2$. In the following, we take $\mathbf{W}_1$ as an example to explain the modulation mechanism, while the other parameters ($\mathbf{W}_2, \mathbf{b}_1$ and $\mathbf{b}_2$) follow a similar process.

For the weight matrix $\mathbf{W}_1 \in \mathbb{R}^{d_2 \times d_1}$ in Eq. (2), we transform it into $\mathbf{W}_{1,v} \in \mathbb{R}^{d_2 \times d_1}$ for the task T_v^+, as follows.

$$\mathbf{W}_{1,v} = \mathbf{W}_1 \odot [(\boldsymbol{\gamma}_v^{W_1} + \mathbf{1})_{\times d_2}]^\top + [(\boldsymbol{\beta}_v^{W_1})_{\times d_2}]^\top, \quad (13)$$

where $\boldsymbol{\gamma}_v^{W_1}$ and $\boldsymbol{\beta}_v^{W_1} \in \mathbb{R}^{d_1}$ are scaling and shifting operators to modulate $\mathbf{W}_1$ into $\mathbf{W}_1^v$ for adaptation, the notation $[(\mathbf{x})_{\times n}] \in \mathbb{R}^{|\mathbf{x}| \times n}$ represents a matrix of n columns all of which are identical to the vector $\mathbf{x}$, and $\odot$ denotes element-wise multiplication. Here the scaling and shifting operators $\boldsymbol{\gamma}_v^{W_1}$ and $\boldsymbol{\beta}_v^{W_1}$ are not directly learnable, but are calculated by a secondary neural network [58] conditioned on the *type signature* of the task T_v^+ to enable type-level adaptation, as follows.

$$\boldsymbol{\gamma}_v^{W_1} = \textsc{LeakyReLU}(\mathbf{W}^{\gamma}_{W_1} \mathbf{o}_v + \mathbf{U}^{\gamma}_{W_1} \mathbf{p}_v), \quad (14)$$

$$\boldsymbol{\beta}_v^{W_1} = \textsc{LeakyReLU}(\mathbf{W}^{\beta}_{W_1} \mathbf{o}_v + \mathbf{U}^{\beta}_{W_1} \mathbf{p}_v), \quad (15)$$

where $\mathbf{W}^{\gamma}_{W_1}$, $\mathbf{W}^{\beta}_{W_1}$, $\mathbf{U}^{\gamma}_{W_1}$ and $\mathbf{U}^{\beta}_{W_1} \in \mathbb{R}^{d_1 \times |\mathcal{T}|}$ are learnable weights. The input $\mathbf{o}_v$ and $\mathbf{p}_v$ represent the type signature of T_v^+. Specifically, $\mathbf{o}_v \in \mathbb{R}^{|\mathcal{T}|}$ is a vector with each dimension corresponding to the number of neighboring nodes of each type (within m hops); $\mathbf{p}_v \in \mathbb{R}^{|\mathcal{T}|}$ is an indicator vector in which only the dimension corresponding to $\varphi_{\mathcal{V}}(v)$ is one while the others are zeros. Together, $\mathbf{o}_v$ and $\mathbf{p}_v$ characterize the distribution of node types associated with the task T_v^+, on which the modulation is conditioned to enable type-level adaptation. Lastly, $\mathbf{1} \in \mathbb{R}^{d_1}$ is a vector filled with ones to ensure the scaling factors centered around one. Likewise, $\mathbf{W}_2, \mathbf{b}_1$ and $\mathbf{b}_2$ can be modulated into $\mathbf{W}_{2,v}, \mathbf{b}_{1,v}$ and $\mathbf{b}_{2,v}$ for the task $T_v^+ = (S_v^+, q_v^+)$, respectively.

Finally, in a type-aware task T_v^+ on a HIN, we can predict the embedding of the query v, denoted by $\hat{\mathbf{h}}_v^+ \in \mathbb{R}^d$, by rewriting Eq. (2) with modulated regression parameters:

$$\hat{\mathbf{h}}_v^+ = F(v; \Theta_v) = \mathbf{W}_{2,v} \cdot \sigma(\mathbf{W}_{1,v} \mathbf{x}_v^+ + \mathbf{b}_{1,v}) + \mathbf{b}_{2,v}, \quad (16)$$

where $\Theta_v = \{\mathbf{W}_{1,v}, \mathbf{W}_{2,v}, \mathbf{b}_{1,v}, \mathbf{b}_{2,v}\}$ represents the regression parameters tailored to the task T_v^+. Note that Θ_v are not directly learnable, as they are generated by the FiLM layers with learnable parameters $\Theta^{\text{FiLM}} = \{\mathbf{W}_*^{\gamma}, \mathbf{W}_*^{\beta}, \mathbf{U}_*^{\gamma}, \mathbf{U}_*^{\beta}\}$.

Node-level adaptation. We employ the same MAML-based meta-learning paradigm to localize the prior w.r.t. the query v in a task T_v^+, in the same spirit of Eq. (10). The main difference lies in what constitutes our prior on a HIN. In Eq. (10), the prior Θ boils down to the learnable parameters of the embedding regression model. In the extended meta-tail2vec+, the parameters are expanded in order to handle the heterogeneity on a HIN. The additional parameters come from two sources: (1) Our heterogeneity-aware neighborhood aggregation in Eq. (11) requires a type-specific weight matrix for each type, *i.e.*, $\Theta^{\mathcal{T}} = \{\mathbf{W}^{\tau} : \forall \tau \in \mathcal{T}\}$; (2) Our FiLM layer for type-level adaption involves several secondary networks to generate the modulations for the shared regression model, *i.e.*, $\Theta^{\text{FiLM}} = \{\mathbf{W}_*^{\gamma}, \mathbf{W}_*^{\beta}, \mathbf{U}_*^{\gamma}, \mathbf{U}_*^{\beta}\}$. Hence, in meta-tail2vec+, the prior would be $\Theta^+ = (\Theta, \Theta^{\mathcal{T}}, \Theta^{\text{FiLM}})$, which can be optimized on the set of meta-training tasks T^+_{train} following the same loss in Eq. (10).

E. Complexity Analysis

The proposed two models, meta-tail2vec and meta-tail2vec+, predict refined node embeddings $\hat{\mathbf{h}}_v, \hat{\mathbf{h}}_v^+ \in \mathbb{R}^d$ as shown in Eqs. (2) and (16), respectively. Herein, we provide a comparison of the complexities of these two models, focusing on the computation of the refined node embeddings, since the MAML-based framework is common to both models and would not impact the relative comparison between them. We highlight that in Sect. IV-C, we have already provided a complexity analysis for meta-tail2vec from a data access perspective. Subsequently, we will offer a more in-depth complexity analysis of these models, from a representation computation perspective.

meta-tail2vec. Assuming an average degree of n, where each node's feature vector $\mathbf{x}_v \in \mathbb{R}^{d_1}$ is aggregated from its m-hop neighbors as described in Eq. (3), which entails a complexity of $O(n^m d_1)$. To further compute $\hat{\mathbf{h}}_v$, the two layers in Eq. (2) contributes a complexity of $O(d_2 \times d_1 + d \times d_2)$. Hence, the overall complexity of calculating $\hat{\mathbf{h}}_v$ is $O(n^m d_1 + d_2 \times d_1 + d \times d_2)$. As $\mathbf{x}_v$ is aggregated via a simple mean pooling, we have $d = d_1$ and the overall complexity simplifies to $O(n^m d_1 + d_2 \times d_1)$. Typically, m is a small constant (*e.g.*, 1 or 2), and n is also small per the definition of tail nodes.

meta-tail2vec+. In addition to the calculations in meta-tail2vec, the extended model on HINs, meta-tail2vec+, requires two further operations: heterogeneous aggregation and type-level adaptation. On one hand, for meta-tail2vec+ builds upon the homogeneous aggregation with a complexity of $O(n^m d_1)$, while introducing an additional heterogeneity-aware neighborhood aggregation in Eq. (11). This increases the complexity to $O(n^m d_1 + n^m d_1^2) = O(n^m d_1^2)$ because each neighboring node requires a multiplication with the matrix $\mathbf{W}^\tau \in \mathbb{R}^{d_1 \times d_1}$ based on the type τ. On the other hand, for type-level adaptation, meta-tail2vec+ needs to modulate the regression parameters $\Theta = \{\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2\}$ into their localized versions. More precisely, to compute $\mathbf{W}_1^v$ in Eq. (13), we should first compute $\gamma_v^{W_1}$ and $\beta_v^{W_1}$, which involves complexity of $O(d_1|\mathcal{T}|)$. Thus, the calculation of $\mathbf{W}_1^v$ has the complexity of $O(d_1|\mathcal{T}| + d_2 \times d_1)$ after performing the scaling and shifting. Similarly, the calculation of $\mathbf{W}_2^v$, $\mathbf{b}_1^v$ and $\mathbf{b}_2^v$ involves complexity of $O(d_2|\mathcal{T}| + d \times d_2)$, $O(d_2|\mathcal{T}| + d_2)$ and $O(d|\mathcal{T}| + d)$, respectively. Thus, the total complexity of type-level adaptation is $O((d+d_1+d_2)|\mathcal{T}| + d_2(d_1+d))$. As $d = d_1$, it can be further simplified to $O((d_1 + d_2)|\mathcal{T}| + d_2 \times d_1)$. Overall, the calculation of $\hat{\mathbf{h}}_v^+$ amounts to $O(n^m d_1^2 + (d_1 + d_2)|\mathcal{T}| + d_2 \times d_1)$.

To summarize, compared to meta-tail2vec, to predict the node embedding, meta-tail2vec+ increases the complexity by $O(n^m d_1^2 + (d_1 + d_2)|\mathcal{T}|)$. Considering that n and m are small constants, the incremental complexity is primarily related to the embedding/hidden dimensions d_1 and d_2, and is linear in the number of types $|\mathcal{T}|$ on the HIN.

V. Experiments on Homogeneous Graphs

In this section, we conduct node classification and link prediction on three public benchmark homogeneous graphs, to evaluate the performance of meta-tail2vec.

TABLE I: Summary of homogeneous datasets.

	nodes	edges	classes	multi-label	tail nodes
Wiki	2,405	17,981	19	No	1,069
Flickr	80,513	5,899,882	195	Yes	9,367
Email	1,005	25,571	42	No	235

A. Experimental Settings

1) Datasets: We conducted experiments on three public datasets. (1) *Wiki* [3] is a Wikipedia network, where each node is a page, and each edge represents the hyperlink between pages. Each page belongs to one of the 19 categories. (2) *Flickr* [1] is a network of users of the photo-sharing service, where each node is a user, and each edge represents the friendship between users. Every user belongs to one or more interest groups, such as "scenic photos". (3) *Email* [59] is an e-mail network between members of a European research institution, where each node is a member, and each edge represents the communication between members. Every member belongs to one of the 42 departments. Their statistics are summarized in Table I. Note that we regarded nodes with 5 or fewer links as the tail nodes, *i.e.*, $\{v \in \mathcal{V} : |\mathcal{N}_v| \leq 5\}$.

2) Base Embedding Models: Our approach meta-tail2vec is flexible to work with any embedding model. We experimented with two broad categories of base embedding models. First, we employ classic network embedding approaches and graph neural networks.

- *DeepWalk* [1]: a pioneering, widely adopted network embedding model, which samples an equal number of paths from each node, and feeds these paths into a skip-gram model to learn node embeddings.
- *GraphSAGE* [4]: a GNN that performs graph convolutions to aggregate features from neighbors recursively. For node features, we use node embedding vectors from DeepWalk and node degrees. We adopt its self-supervised version to learn the initial base embeddings.

Second, we employ embedding models for sparse networks.

- *SDNE* [32]: a deep network embedding model that is robust for sparse networks, by incorporating global network structures in addition to local structures.
- *ARGA* [31]: an adversarially regularized graph autoencoder, which achieves robust embedding by learning the data distribution of latent codes on the graph.
- *DDGCN* [29]: a graph convolutional network with a form of dual dropouts at both the node and edge levels, to more effectively reduce overfitting on sparse networks.

Setup and parameters. To ensure the base models achieve their respective optimal performance, we performed a grid search to tune their parameters (optimal values in italics). For DeepWalk, we searched the number of walks $\gamma \in \{5, \textit{10}, 20\}$, walk length $t \in \{\textit{40}, 100, 150\}$ and window size $w \in \{\textit{3}, 5\}$. For GraphSAGE, we adopted a two-layer architecture, chose the aggregator from {*mean*, meanpool, maxpool} and tuned the dimension of the hidden layer over $\{32, \textit{64}, 128\}$. For SDNE, we searched the weight of local structures (as opposed to global structures) $\alpha \in \{50, \textit{100}, 150\}$ and the weight of reconstruction $\beta \in \{10, \textit{30}, 50\}$. For ARGA, we tuned the

dimension of the hidden layer over $\{16, 32, \mathit{64}\}$. For DDGCN, we tuned the dropout probability $p \in \{0.1, 0.3, \mathit{0.5}\}$, and dual-dropout coefficient $\alpha \in \{0.1, \mathit{1}, 5\}$. The optimal parameters found are generally consistent with the recommended values in the literature. For all models, the dimension of embedding vectors is set to 128.

3) Baselines for Tail Node Refinement: We compared with a series of baselines that are also designed to improve tail node embeddings.

- *Biased walk*: As tail nodes are under-represented, we over-sampled random walks starting from the tail nodes. This method is only applicable to DeepWalk and GraphSAGE, since the other base models do not utilize random walks.
- *Additive* [60] aggregates the embeddings of the neighboring nodes as the output embedding for a tail node. We also compared to *Additive-2* which aggregates the embeddings of neighboring nodes within 2-hops.
- *a la carte* [61] is a further extension of the additive model, with a transformation through an auxiliary regression task. Similarly, we also compared to *a la carte-2*, which aggregates the embedding of 2-hop neighbors.
- *Nonce2vec* [62] constructs a better initialization with the additive vectors, and performs another round of training using the corresponding base embedding model.
- *Dropout*: Inspired by DropoutNet [33], for each head node we sampled only k neighbors, which were further fed into an auxiliary regression task.

Setup and parameters. The goal is to predict new embedding vectors for the tail nodes by meta-tail2vec and each baseline refinement method, w.r.t. the initial embedding vectors from each of the base embedding models.

For biased walk, we doubled the paths starting from the tail nodes compared to the head nodes. For Additive, a la carte and Nonce2vec, we used mean pooling as the aggregation function, which yields better empirical performance than min or max pooling. For a la carte and Dropout, the auxiliary regression tasks used the same regression model in our approach. For Nonce2vec, with DeepWalk, SDNE, and ARGA as the base embedding models, the improved initialization was directly used as a pre-training; with GraphSAGE and DDGCN, the improved initialization was used as nodes' initial feature vectors.

For our method meta-tail2vec, we set α, the local learning rate of adapting to the support set to 0.01, and set the global meta-learning rate to 0.001. Note that it is typical to employ a larger local learning rate in order to achieve stable training [9], [52]. The number of gradient updates during adaptation was set to 5, noting that few updates such as 1 or 3 give very close results. In the regression model F, we set the size of the hidden layer of the MLP to 1024, and aggregated nodes within 2 hops. We will also analyze the impact of the number of hops in Sect. V-C.

4) Downstream Applications: We experimented with two downstream applications on the three homogeneous datasets.

Node classification. We carried out multi-class classification on Wiki and Email where each node belongs to exactly one class, and multi-label node classification on Flickr where each node can belong to one or more classes. Specifically, we evaluated the classification performance on the tail nodes, after training a logistic regression classifier on the head. The predicted embeddings of the tail nodes and the oracle embeddings of the head nodes were used as their input features for the classifier, respectively. We adopted microF and accuracy as the evaluation metrics.

Link prediction. For nodes with 2–6 links, we adopted the common leave-one-out strategy by first removing a random link from each of them (which becomes a tail node with 5 or fewer links). Our goal is to predict the removed link. On the partial network, we constructed initial embedding vectors using each base embedding model, and predicted new embedding vectors for the tail nodes with our method and each baseline refinement method. For each tail node, we treated the removed link as a positive candidate, and randomly sampled five other non-linked nodes as negative candidates. Link prediction was then formulated as a ranking problem: given a tail node, we rank its candidates using a learning-to-rank model [63] trained on the head nodes. Likewise, the predicted embeddings of the tail nodes and the oracle embeddings of the head nodes were used as their features, respectively. We adopted the evaluation metrics of mean reciprocal rank (MRR) and hit ratio at top 1 (Hit@1).

B. Performance Comparison

We present the performance of meta-tail2vec and the baselines w.r.t. each base embedding model. As our goal is to improve tail node embeddings, we mainly focus on comparing the performance on the tail nodes. Nevertheless, to further demonstrate that head node embeddings are not adversely impacted, we also investigate the performance on the head nodes. Note that all results are averaged over 10 runs and reported with their standard deviations; the best method appears in bold, and the runner-up is underlined.

1) Comparison of Tail Node Embeddings: We first study classic base embedding models that are not specifically designed for robustness on sparse networks, followed by robust base models designed for sparse networks.

Classic base models. We report the performance of node classification in Table II w.r.t. classic base embedding models DeepWalk and GraphSAGE, respectively. On the one hand, our meta-tail2vec achieves significant improvements over the base embedding methods consistently, by 7.7%–10.9% w.r.t. DeepWalk and 9.3%–13.3% w.r.t. GraphSAGE in terms of MicroF. The results demonstrate that tail node embedding is a critical problem to address, and our proposed approach is indeed useful in refining the tail node embeddings. On the other hand, meta-tail2vec also outperforms other refinement baselines, gaining performance lifts in the range of 5.3%–11.3% over the best baseline in terms of MicroF. These baselines are sub-optimal as they only assume one model for all tail nodes, whereas meta-tail2vec can attribute its strong performance to the locality adaptation to each node under a meta-learning framework. In particular, the 2-hop variants of Additive and a la carte are often worse than their 1-hop models, which may be

TABLE II: Performance of node classification w.r.t. classic base embedding models.

		Base	Biased walk	Additive	Additive-2	a la carte	a la carte-2	Nonce2vec	Dropout	meta-tail2vec	Improv. over Base	Improv. over 2nd best
DeepWalk as the base embedding model												
Wiki	MicroF	44.27 ± 0.25	44.69 ± 0.31	45.32 ± 0.52	42.11 ± 0.76	23.65 ± 0.44	23.34 ± 0.47	44.97 ± 0.29	36.88 ± 0.65	**49.10** ± 0.23	+10.9%	+8.3%
	Accuracy	46.68 ± 0.31	47.05 ± 0.17	47.18 ± 0.29	44.73 ± 0.53	24.17 ± 0.49	24.48 ± 0.42	47.11 ± 0.22	38.13 ± 0.57	**50.70** ± 0.45	+8.6%	+7.5%
Flickr	MicroF	33.48 ± 0.26	33.61 ± 0.39	34.43 ± 0.41	32.59 ± 0.17	31.89 ± 0.47	32.25 ± 0.35	33.83 ± 0.28	33.91 ± 0.22	**36.31** ± 0.19	+8.5%	+5.5%
	Accuracy	32.44 ± 0.13	32.57 ± 0.19	33.29 ± 0.17	31.31 ± 0.24	32.13 ± 0.26	32.62 ± 0.31	33.01 ± 0.15	32.86 ± 0.09	**35.28** ± 0.25	+8.8%	+6.0%
Email	MicroF	51.32 ± 0.29	50.95 ± 0.24	52.50 ± 0.17	51.17 ± 0.23	17.88 ± 0.48	18.21 ± 0.52	51.84 ± 0.33	32.72 ± 0.45	**55.26** ± 0.18	+7.7%	+5.3%
	Accuracy	54.41 ± 0.34	54.13 ± 0.22	55.38 ± 0.43	53.82 ± 0.36	21.06 ± 0.45	21.13 ± 0.37	54.79 ± 0.19	33.85 ± 0.51	**57.78** ± 0.29	+6.2%	+4.3%
GraphSAGE as the base embedding model												
Wiki	MicroF	39.68 ± 0.24	40.07 ± 0.15	37.84 ± 0.31	35.96 ± 0.43	23.88 ± 0.47	22.52 ± 0.39	40.75 ± 0.33	19.78 ± 0.59	**44.29** ± 0.31	+11.6%	+8.7%
	Accuracy	41.22 ± 0.19	41.39 ± 0.06	39.31 ± 0.26	36.59 ± 0.25	25.71 ± 0.36	24.94 ± 0.62	41.65 ± 0.28	24.73 ± 0.42	**44.90** ± 0.12	+8.9%	+7.8%
Flickr	MicroF	29.38 ± 0.32	28.75 ± 0.31	27.86 ± 0.14	23.69 ± 0.44	30.02 ± 0.17	29.67 ± 0.20	29.85 ± 0.12	28.75 ± 0.11	**32.11** ± 0.41	+9.3%	+7.0%
	Accuracy	28.46 ± 0.08	27.52 ± 0.19	27.69 ± 0.31	22.82 ± 0.45	29.83 ± 0.22	28.18 ± 0.46	29.26 ± 0.31	28.78 ± 0.14	**31.96** ± 0.35	+12.3%	+7.1%
Email	MicroF	41.25 ± 0.17	41.07 ± 0.33	35.83 ± 0.31	34.19 ± 0.13	27.81 ± 0.44	26.97 ± 0.39	41.97 ± 0.24	23.47 ± 0.25	**46.73** ± 0.37	+13.3%	+11.3%
	Accuracy	42.61 ± 0.31	42.20 ± 0.31	37.25 ± 0.16	35.13 ± 0.35	29.41 ± 0.46	27.16 ± 0.34	43.23 ± 0.30	25.84 ± 0.18	**47.70** ± 0.46	+11.9%	+10.3%

TABLE III: Performance of link prediction w.r.t. classic base embedding models.

		Base	Biased walk	Additive	Additive-2	a la carte	a la carte-2	Nonce2vec	Dropout	meta-tail2vec	Improv. over Base	Improv. over 2nd best
DeepWalk as the base embedding model												
Wiki	MRR	75.28 ± 0.37	75.13 ± 0.41	75.81 ± 0.62	74.89 ± 0.78	76.31 ± 0.25	76.14 ± 0.33	67.42 ± 0.87	77.06 ± 0.71	**79.18** ± 0.52	+5.2%	+2.8%
	Hit@1	51.83 ± 0.42	52.04 ± 0.57	52.51 ± 0.67	51.48 ± 0.39	53.70 ± 0.61	53.59 ± 0.32	53.34 ± 0.49	54.19 ± 0.30	**57.22** ± 0.46	+10.4%	+5.6%
Flickr	MRR	50.05 ± 0.30	49.57 ± 0.19	49.80 ± 0.45	49.72 ± 0.41	50.36 ± 0.55	50.71 ± 0.65	50.83 ± 0.48	50.25 ± 0.59	**52.18** ± 0.61	+4.3%	+2.7%
	Hit@1	25.32 ± 0.24	25.63 ± 0.55	26.10 ± 0.41	26.55 ± 0.62	26.07 ± 0.30	26.39 ± 0.58	26.67 ± 0.33	26.19 ± 0.44	**28.11** ± 0.40	+11.0%	+5.4%
Email	MRR	44.17 ± 0.35	44.58 ± 0.26	44.52 ± 0.68	44.96 ± 0.28	44.49 ± 0.50	45.11 ± 0.34	44.80 ± 0.15	45.33 ± 0.08	**48.42** ± 0.55	+9.6%	+6.8%
	Hit@1	19.47 ± 0.38	19.96 ± 0.27	21.38 ± 0.15	21.66 ± 0.40	22.45 ± 0.58	22.63 ± 0.31	20.90 ± 0.44	23.02 ± 0.33	**24.31** ± 0.46	+24.9%	+5.6%
GraphSAGE as the base embedding model												
Wiki	MRR	81.36 ± 0.14	82.01 ± 0.10	80.56 ± 0.45	80.39 ± 0.21	81.82 ± 0.53	80.94 ± 0.62	82.18 ± 0.64	82.52 ± 0.40	**84.38** ± 0.61	+3.7%	+2.3%
	Hit@1	58.87 ± 0.52	58.39 ± 0.15	58.43 ± 0.61	58.92 ± 0.30	59.56 ± 0.29	59.34 ± 0.44	59.70 ± 0.37	59.93 ± 0.56	**62.04** ± 0.68	+5.4%	+3.5%
Flickr	MRR	55.83 ± 0.29	56.17 ± 0.36	55.04 ± 0.25	55.40 ± 0.58	56.28 ± 0.49	56.76 ± 0.40	56.31 ± 0.32	56.85 ± 0.71	**58.15** ± 0.43	+4.2%	+2.3%
	Hit@1	34.59 ± 0.52	35.15 ± 0.47	33.79 ± 0.38	33.36 ± 0.40	35.22 ± 0.68	35.29 ± 0.64	34.97 ± 0.50	35.74 ± 0.31	**36.92** ± 0.39	+6.7%	+3.3%
Email	MRR	46.71 ± 0.45	46.24 ± 0.29	46.05 ± 0.25	46.68 ± 0.44	47.03 ± 0.53	46.92 ± 0.30	47.18 ± 0.19	46.37 ± 0.60	**48.15** ± 0.44	+3.1%	+2.1%
	Hit@1	23.02 ± 0.23	22.73 ± 0.41	22.91 ± 0.44	22.65 ± 0.52	23.19 ± 0.39	23.14 ± 0.61	23.28 ± 0.43	23.07 ± 0.56	**24.55** ± 0.70	+6.6%	+5.4%

caused by noises from 2-hop nodes. However, our model also aggregates 2-hop nodes and attains better performance than its 1-hop version (as we will see in Sect. V-C), potentially due to the local adaptation which can effectively filter noises at each locality.

We further report the performance of link prediction in Table III w.r.t. classic base embedding models. Similar conclusions can be drawn, where meta-tail2vec outperforms the base models by 3.1%–9.6% and the best baseline by 2.1%–6.8% in terms of MRR.

Robust base models for sparse networks. We also investigate whether meta-tail2vec can also improve base models designed for robustness on sparse networks.

We report the performance of node classification in Table IV, w.r.t. each of the base models SDNE, ARGA, and DDGCN. While these base models are intended to handle sparse networks, they aim to increase the overall robustness of the learning process, and do not explicitly improve the embedding of the most vulnerable tail nodes. Thus, their performances on the tail nodes are not necessarily better than classic base models. Note that our approach meta-tail2vec is embedding-agnostic, meaning that even for base models already designed for sparse networks, we can still refine their tail node embeddings, as demonstrated by the results that meta-tail2vec outperforms the base embeddings by an average of 14.9% in terms of MicroF on node classification. On the other hand, we also compare meta-tail2vec to other baseline refinement methods. (Due to space constraints, we only present the results of Addictive, Nonce2vec, and Dropout, which are generally the best baselines among all.) Again, due to our locality-aware task formulation, the meta-learning strategy is able to adapt to the locality of each tail node well, resulting in an average performance lift of 8.5% in terms of MicroF compared to the best baseline.

Furthermore, we report the performance of link prediction in Table V. We have similar observations, where on average meta-tail2vec outperforms the robust base models by 5.2% and the best baseline by 2.0% in terms of MRR.

2) Comparison of Head Node Embeddings: While our main goal is to improve tail node embeddings, we further evaluate the performance on the head nodes to validate that their embeddings still remain competitive. In theory, head node embeddings are not changed in any way, since we only predict new embedding vectors for tail nodes. However, the performance of head nodes on a downstream application can be potentially improved, as training the downstream model can still benefit from the improved quality of tail nodes.

Thus, we further conducted an experiment on head node embeddings. We sampled and evaluated a test set comprising 10% of the head nodes, and trained a model for each down-

TABLE IV: Performance of node classification w.r.t. robust base embedding models (MiF for MicroF; Acc for accuracy).

		Base	Additive	Nonce2vec	Dropout	meta-tail2vec
SDNE as the base embedding model						
Wiki	MiF	31.38 ± 0.34	34.46 ± 0.63	32.14 ± 0.75	34.69 ± 0.48	**37.99** ± 0.83
	Acc	34.10 ± 0.71	35.62 ± 0.28	34.59 ± 0.12	36.46 ± 0.43	**38.80** ± 0.64
Flickr	MiF	34.74 ± 0.86	35.49 ± 0.47	34.73 ± 0.37	35.38 ± 0.35	**38.50** ± 0.78
	Acc	32.67 ± 0.32	34.72 ± 0.28	33.59 ± 0.73	34.64 ± 0.49	**38.03** ± 0.66
Email	MiF	29.85 ± 0.48	31.07 ± 0.21	31.83 ± 0.46	34.50 ± 0.25	**47.21** ± 0.72
	Acc	32.90 ± 0.62	34.37 ± 0.27	33.79 ± 0.60	37.85 ± 0.47	**51.70** ± 0.33
ARGA as the base embedding model						
Wiki	MiF	32.22 ± 0.46	31.19 ± 0.23	32.47 ± 0.19	32.85 ± 0.23	**33.51** ± 0.31
	Acc	34.47 ± 0.52	33.84 ± 0.10	34.79 ± 0.21	35.22 ± 0.49	**35.80** ± 0.23
Flickr	MiF	24.60 ± 0.15	23.69 ± 0.18	25.16 ± 0.20	24.71 ± 0.15	**25.93** ± 0.25
	Acc	22.81 ± 0.17	21.59 ± 0.11	24.26 ± 0.46	23.65 ± 0.39	**25.37** ± 0.16
Email	MiF	24.38 ± 0.31	23.94 ± 0.47	24.97 ± 0.35	25.48 ± 0.53	**26.11** ± 0.25
	Acc	26.57 ± 0.43	25.69 ± 0.30	27.02 ± 0.37	27.54 ± 0.21	**27.95** ± 0.16
DDGCN as the base embedding model						
Wiki	MiF	29.49 ± 0.18	27.68 ± 0.58	31.20 ± 0.44	30.37 ± 0.35	**33.02** ± 0.43
	Acc	31.39 ± 0.25	30.82 ± 0.21	33.87 ± 0.75	32.69 ± 0.40	**36.27** ± 0.41
Flickr	MiF	28.57 ± 0.47	26.92 ± 0.08	30.09 ± 0.35	29.17 ± 0.26	**31.03** ± 0.52
	Acc	25.90 ± 0.71	24.44 ± 0.12	26.76 ± 0.49	26.37 ± 0.25	**28.38** ± 0.54
Email	MiF	38.95 ± 0.67	38.73 ± 0.55	39.62 ± 0.43	39.15 ± 0.40	**41.83** ± 0.34
	Acc	39.81 ± 0.56	38.20 ± 0.35	42.32 ± 0.63	41.69 ± 0.41	**44.13** ± 0.73

TABLE V: Performance of link prediction w.r.t. robust base embedding models (H@1 for hit@1).

		Base	Additive	Nonce2vec	Dropout	meta-tail2vec
		SDNE as the base embedding model				
Wiki	MRR	72.25 ± 0.48	72.53 ± 0.30	74.44 ± 0.39	75.08 ± 0.65	**76.97** ± 0.61
	H@1	52.19 ± 0.27	51.94 ± 0.39	54.50 ± 0.61	55.21 ± 0.35	**57.58** ± 0.74
Flickr	MRR	46.82 ± 0.20	47.09 ± 0.44	48.35 ± 0.51	48.17 ± 0.29	**49.31** ± 0.46
	H@1	26.23 ± 0.16	27.00 ± 0.33	28.82 ± 0.61	28.39 ± 0.10	**29.26** ± 0.20
Email	MRR	34.02 ± 0.76	34.29 ± 0.51	36.87 ± 0.49	36.42 ± 0.55	**39.55** ± 0.50
	H@1	17.51 ± 0.24	18.65 ± 0.51	21.19 ± 0.40	20.88 ± 0.13	**22.86** ± 0.63
		ARGA as the base embedding model				
Wiki	MRR	48.57 ± 0.40	46.49 ± 0.38	49.16 ± 0.45	50.27 ± 0.14	**51.08** ± 0.20
	H@1	41.40 ± 0.52	40.49 ± 0.07	41.67 ± 0.35	42.22 ± 0.10	**43.73** ± 0.65
Flickr	MRR	35.52 ± 0.32	35.41 ± 0.72	35.69 ± 0.63	36.31 ± 0.28	**36.87** ± 0.45
	H@1	29.73 ± 0.34	28.86 ± 0.40	29.89 ± 0.62	30.51 ± 0.77	**31.37** ± 0.29
Email	MRR	26.83 ± 0.29	25.89 ± 0.47	26.91 ± 0.18	26.22 ± 0.40	**27.26** ± 0.55
	H@1	16.51 ± 0.29	16.30 ± 0.42	17.22 ± 0.40	16.89 ± 0.31	**17.87** ± 0.35
		DDGCN as the base embedding model				
Wiki	MRR	73.25 ± 0.49	74.10 ± 0.34	74.28 ± 0.15	74.92 ± 0.53	**75.31** ± 0.67
	H@1	51.28 ± 0.39	50.77 ± 0.21	51.86 ± 0.45	52.56 ± 0.32	**53.30** ± 0.61
Flickr	MRR	52.17 ± 0.40	50.74 ± 0.51	52.23 ± 0.42	51.79 ± 0.60	**52.49** ± 0.34
	H@1	37.15 ± 0.38	35.82 ± 0.85	37.53 ± 0.42	37.16 ± 0.60	**38.68** ± 0.63
Email	MRR	41.58 ± 0.45	40.83 ± 0.37	42.96 ± 0.39	42.81 ± 0.12	**43.47** ± 0.18
	H@1	27.35 ± 0.39	28.31 ± 0.63	28.58 ± 0.25	28.87 ± 0.30	**29.22** ± 0.36

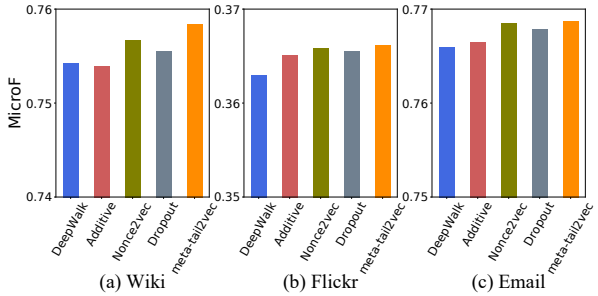


Fig. 4: Performance of node classification on head nodes w.r.t. DeepWalk as the base embedding model.

stream task on the other nodes, inclusive of the tail nodes and the remaining 90% head nodes. We tabulate the results

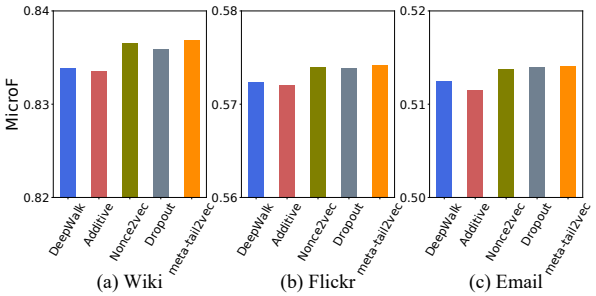


Fig. 5: Performance of link prediction on head nodes w.r.t. DeepWalk as the base embedding model.

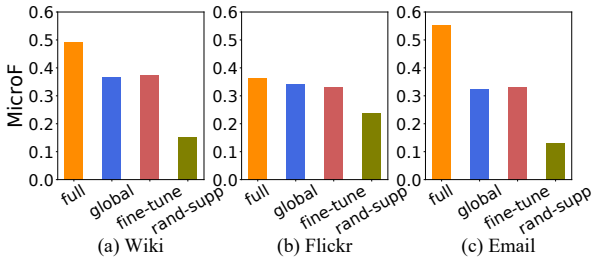


Fig. 6: Ablation study of the meta-learning strategy on node classification w.r.t. DeepWalk as the base model.

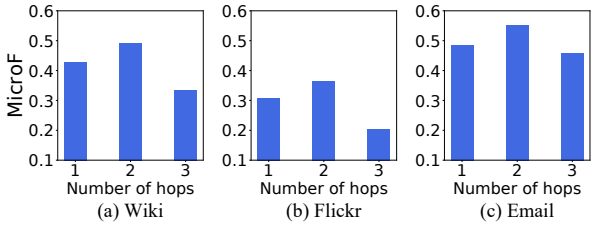


Fig. 7: Impact of number of hops on node classification.

in Figs. 4 and 5 for node classification and link prediction, respectively. As hypothesized, both meta-tail2vec and other baseline refinement approaches can slightly outperform the base embedding model, due to the improved quality of tail node embeddings in the downstream training data. However, it is not surprising that the improvements are modest compared to tail nodes, in the range of 0.3%–0.9% only. The reason is that the head node embeddings remain unchanged and their performance is only indirectly influenced by refined tail node embeddings. Nevertheless, we validated the goal of significantly improving tail node embeddings whilst head node embeddings remain robust.

C. Model Analysis and Discussion

1) Ablation Study: We analyze the contribution of our approach by an ablation study. Using DeepWalk as the base embedding model, we compare the following four variants of meta-tail2vec: (1) *full*, the full meta-tail2vec model; (2) *global*, only train one global embedding regression model on the head nodes, and predict the embedding vectors of all tail nodes with the same global model (equivalent to the Dropout baseline); (3) *fine-tune*, fine-tune the pre-trained global model on the support set of a tail node before predicting its embedding

vector; (4) *rand-supp*, the same approach as the full model except the locality awareness, which samples random nodes from the graph as the support sets.

Their performances are reported in Fig. 6. Among the four variants, we observe that the fine-tune model is only able to achieve marginally better performance than the global model, as fine-tuning on the small support set of a tail node can easily cause overfitting. In particular, on the Flickr dataset, the fine-tune model is in fact slightly worse than the global model due to overfitting. Next, the rand-supp model performs the worst, implying that the locality-aware task generation is critical for graph data. Finally, the full model performs the best, demonstrating the effectiveness of our locality-aware meta-learning approach.

2) Impact of Neighborhood Hops: We also analyze the impact of neighborhood hops on computing the input feature in Eq. (5), and present the results in Fig. 7. Generally, aggregating the 2-hop neighborhood achieves optimal performance on all datasets. Besides, aggregating the 3-hop neighborhood results in decreased performance due to more noise from the less relevant nodes.

VI. Experiments on Heterogeneous Graphs

In this section, we conduct node classification and link prediction on two benchmark heterogeneous graphs to evaluate the performance of our proposed meta-tail2vec+.

A. Experimental Settings

1) Datasets: We employ two heterogeneous graphs, *i.e.*, *DBLP* [64] and *MovieLens*[2]. *DBLP* is a bibliography network collected from DBLP[3], in which there are four node types, *i.e.*, Paper (P), Conference (C), Author (A) and Term (T), thus forming three types of links, P-C, P-A and P-T. Based on the original 4,057 labeled author nodes in four classes, we further label the paper nodes into four classes for node classification. In particular, we only label the paper nodes with the neighboring author label which covers no less than half of all labeled neighboring authors. *MovieLens* is a benchmark dataset collected by GroupLens. It contains four node types, *i.e.*, User (U), Movie (M), Actor (A), and Director (D), forming three types of links, U-M, M-A, and M-D. Movies released from 1999 to 2000 are labeled into 18 classes based on their genres. The statistics of the two datasets are summarized in Table VI.

2) Base Embedding Models: Our meta-tail2vec+ is agnostic of the heterogeneous embedding model, thus we choose one typical model from two categories of heterogeneous embedding models: heterogeneous network embedding approach *metapath2vec* [23] and heterogeneous graph neural network *HAN* [26].

- *metapath2vec* [23]: a typical heterogeneous network embedding model, which utilizes predefined meta-paths [65] to guide path sampling, and further learns node representations by feeding the paths into a skip-gram model.

[2]https://grouplens.org/datasets/movielens/
[3]https://dblp.org/

TABLE VI: Summary of heterogeneous datasets. Underlined node type indicates the target type for the classification task.

	node type	nodes	edges	classes	multi-label	tail nodes
DBLP	Paper (P)	14,376	170,794	4	No	385
	Conference (C)	20				
	Author (A)	14,475				
	Term (T)	8,920				
MovieLens	User (U)	6,040	1,019,817	18	Yes	506
	Movie (M)	3,881				
	Actor (A)	8,030				
	Director (D)	2,186				

- *HAN* [26]: a heterogeneous GNN which conducts neighborhood aggregation with both node- and semantic-level attention to aggregate neighbors w.r.t. predefined meta-paths. We utilize the DeepWalk embeddings [1] as the initial node features.

Setup and parameters. We refer to the settings and parameters in their original work and further tune them to achieve optimal performance. In particular, for metapath2vec, we employ meta-paths APA, APCPA, and APTPA for *DBLP*, while UMA and UMD for *MovieLens*. For the hyper-parameters, we set the output embedding dimension as 128, window size as 5, walk length as 100, number of walks per node as 40, and number of negative samples as 5. For HAN, we modify it in a self-supervised manner to learn the initial base embeddings, by maximizing the similarity between the 2-hop neighbors of the same type, since nodes of the same type are not directly connected in these two datasets. For the hyper-parameters, we set the dimension of the semantic-level attention vectors as 128, the dropout rate as 0.6, the dimension of the output embeddings as 128, and employ 8 heads for self-attention.

3) Baselines and Model Settings: In addition to the base embedding models as well as meta-tail2vec, we further employ a series of baselines, including *Additive*, *Additive-2*, *a la carte*, *a la carte-2* and *Dropout*, with the same settings as illustrated in Sect. V-A3. For meta-tail2vec+, we also employ the same model and parameter settings with meta-tail2vec as depicted in Sect. V-A3. However, the sampling of the locality-aware support set is extended to 2-hop neighbors since nodes of the same type are not directly connected in these two datasets.

4) Downstream Applications: We experiment with two downstream applications, node classification and link prediction. In particular, as indicated in Table VI, we target the Paper nodes on the DBLP dataset and Movie nodes on the MovieLens dataset for classification and link prediction. Unlike the homogeneous setting, we use different k values to distinguish between head and tail nodes due to the difference in data sparsity across these datasets. More specifically, for Paper nodes on DBLP, nodes with five or fewer links are designated as tail nodes, that is, $\{v \in \mathcal{V} : |\mathcal{N}_v| \leq 5\}$ (*i.e.*, $k = 5$). However, for Movie nodes on MovieLens, nodes with ten or fewer links are regarded as tail nodes, that is, $\{v \in \mathcal{V} : |\mathcal{N}_v| \leq 10\}$ (*i.e.*, $k = 10$), since MovieLens has a much higher average node degree than DBLP. Note that, for both datasets, we ensure that each head node has at least five neighbors of the same type within two hops. This

TABLE VII: Performance of node classification w.r.t. classic base heterogeneous embedding models.

		Base	Additive	Additive-2	a la carte	a la carte-2	Dropout	meta-tail2vec	meta-tail2vec+	Improv. over Base	Improv. over 2nd best
metapath2vec as the base heterogeneous embedding model											
DBLP	MicroF	75.09 ± 0.25	32.01 ± 0.43	74.30 ± 0.49	75.64 ± 0.29	73.50 ± 0.30	72.89 ± 0.21	75.14 ± 0.27	**78.02** ± 0.21	+3.9%	+3.1%
	Accuracy	80.39 ± 0.15	29.07 ± 0.38	79.35 ± 0.33	80.95 ± 0.27	78.80 ± 0.27	78.54 ± 0.34	80.85 ± 0.33	**82.63** ± 0.23	+2.8%	+2.1%
MovieLens	MicroF	43.35 ± 0.35	11.44 ± 0.57	48.40 ± 0.63	21.36 ± 0.31	42.83 ± 0.29	41.72 ± 0.39	44.77 ± 0.42	**50.05** ± 0.40	+15.5%	+3.4%
	Accuracy	35.63 ± 0.31	8.06 ± 0.62	39.43 ± 0.48	15.59 ± 0.41	34.09 ± 0.25	32.96 ± 0.26	37.49 ± 0.45	**42.61** ± 0.37	+19.7%	+8.1%
HAN as the base heterogeneous embedding model											
DBLP	MicroF	63.70 ± 0.28	64.07 ± 0.31	64.25 ± 0.38	64.18 ± 0.25	62.53 ± 0.43	63.97 ± 0.37	63.81 ± 0.23	**65.09** ± 0.19	+2.2%	+1.3%
	Accuracy	78.95 ± 0.15	79.12 ± 0.27	78.83 ± 0.30	79.02 ± 0.24	78.44 ± 0.17	78.83 ± 0.31	78.70 ± 0.29	**80.19** ± 0.22	+1.6%	+1.4%
MovieLens	MicroF	51.96 ± 0.23	52.52 ± 0.36	52.77 ± 0.31	50.78 ± 0.29	52.23 ± 0.42	51.35 ± 0.39	52.15 ± 0.30	**56.38** ± 0.27	+8.5%	+6.8%
	Accuracy	44.80 ± 0.35	45.10 ± 0.58	44.92 ± 0.47	44.40 ± 0.44	44.75 ± 0.47	44.99 ± 0.41	44.89 ± 0.40	**49.72** ± 0.31	+11.0%	+10.2%

TABLE VIII: Performance of link prediction w.r.t. classic base heterogeneous embedding models.

		Base	Additive	Additive-2	a la carte	a la carte-2	Dropout	meta-tail2vec	meta-tail2vec+	Improv. over Base	Improv. over 2nd best
metapath2vec as the base heterogeneous embedding model											
DBLP	MRR	72.40 ± 0.31	72.12 ± 0.38	73.55 ± 0.40	73.17 ± 0.18	73.67 ± 0.23	73.93 ± 0.37	72.48 ± 0.34	**75.84** ± 0.24	+4.8%	+2.6%
	Hit@1	57.54 ± 0.25	56.31 ± 0.29	58.02 ± 0.41	57.93 ± 0.22	58.11 ± 0.31	58.94 ± 0.28	57.93 ± 0.36	**61.40** ± 0.33	+6.7%	+4.2%
MovieLens	MRR	89.36 ± 0.43	88.79 ± 0.41	89.58 ± 0.46	89.88 ± 0.38	90.27 ± 0.32	90.32 ± 0.27	89.83 ± 0.25	**91.11** ± 0.29	+2.0%	+0.9%
	Hit@1	81.31 ± 0.33	82.76 ± 0.29	83.00 ± 0.37	82.93 ± 0.28	83.28 ± 0.21	83.60 ± 0.30	82.05 ± 0.28	**85.15** ± 0.36	+4.7%	+1.9%
HAN as the base heterogeneous embedding model											
DBLP	MRR	71.22 ± 0.42	70.39 ± 0.46	71.12 ± 0.29	71.36 ± 0.31	70.20 ± 0.41	71.82 ± 0.32	71.09 ± 0.32	**73.90** ± 0.33	+3.8%	+2.9%
	Hit@1	54.48 ± 0.35	53.83 ± 0.20	54.28 ± 0.24	54.85 ± 0.41	53.47 ± 0.36	55.14 ± 0.28	54.29 ± 0.27	**57.63** ± 0.39	+5.8%	+4.5%
MovieLens	MRR	76.78 ± 0.26	76.00 ± 0.28	76.52 ± 0.35	76.34 ± 0.27	76.66 ± 0.26	77.46 ± 0.31	76.56 ± 0.34	**78.25** ± 0.22	+1.9%	+1.0%
	Hit@1	58.88 ± 0.35	57.66 ± 0.21	58.49 ± 0.17	58.53 ± 0.36	58.84 ± 0.24	59.75 ± 0.29	58.42 ± 0.31	**61.68** ± 0.38	+4.8%	+3.2%

stipulation ensures that each head node has a sufficient number of candidate nodes for sampling their support nodes during the meta-learning process.

Node classification. We carry out multi-class and multi-label node classification on *DBLP* and *MovieLens*, respectively. Following the same settings in Sect. V-A4, we evaluate the performance on the tail nodes after training a logistic regression classifier on the head nodes. We also employ MicroF and Accuracy to evaluate classification performance.

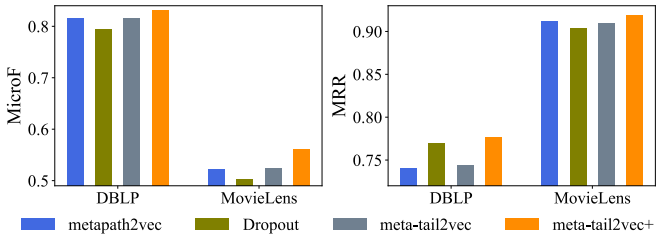
Link prediction. We adopt the same setup for link prediction on homogeneous graphs as illustrated in Sect. V-A4. In particular, for Paper nodes on *DBLP* and Movie nodes on *MovieLens* which have between 2 and $k+1$ links, we adopt the leave-one-out strategy by removing a random link from each of them and our goal is to rank the removed link higher than the other five randomly sampled negative links. Similarly, we also formulate link prediction into a ranking problem, and employ the same evaluation metrics, *i.e.*, MRR and Hit@1. In the training set, for *DBLP* dataset, we employ the head Paper nodes which have more than five 1-hop neighbors and five 2-hop neighbors as training nodes; for *MovieLens* dataset, we employ the head Movie nodes which have more than ten 1-hop neighbors and five 2-hop neighbors as training nodes. In the testing set, for *DBLP* dataset, we employ the tail Paper nodes which have no more than five 1-hop neighbors as testing nodes; for *MovieLens* dataset, we employ the tail Movie nodes which have no more than ten 1-hop neighbors as testing nodes.

B. Performance Comparison

We focus on the performance comparison on the tail nodes. We further investigate the performance on the head nodes to show that their embeddings are not adversely impacted.

1) Comparison of Tail Node Embeddings: **Node classification.** We report the performance of node classification on HINs in Table VII w.r.t. base embedding models metapath2vec and HAN, respectively. We have the following observations. First, the proposed meta-tail2vec+ outperforms its base embedding models including both metapath2vec and HAN. Though typical and powerful for representation learning on heterogeneous graphs, they focus on the performance of all nodes thus usually marginalizing the learning of challenging tail nodes. By virtue of the heterogeneity-aware neighborhood aggregation and the dual-adaptation, meta-tail2vec+ is capable of providing more semantically consistent refinement of the tail node embeddings on HINs. Second, meta-tail2vec+ can achieve better performance than other refinement baselines. As analyzed in Sect. V-B, these baselines only assume one model for all tail nodes, while meta-tail2vec+ tailors to each node for finer node representations. Besides, these baselines cannot effectively cope with the heterogeneity due to their neglect on the node types, while meta-tail2vec+ can address this issue by means of the heterogeneity-aware neighborhood aggregation and dual-adaptation.

Besides, meta-tail2vec, which does not consider the graph heterogeneity either, can be treated as an ablated variant of meta-tail2vec+. meta-tail2vec+ is able to achieve superior performance, demonstrating the effectiveness of the proposed heterogeneity-aware neighborhood aggregation and dual-adaptation-based few-shot regression in dealing with the representation learning for tail nodes on HINs.



(a) Node classification (b) Link prediction

Fig. 8: Performance on head nodes w.r.t. metapath2vec as the base embedding model.

Link prediction. We further report the performance comparison for link prediction w.r.t. the two base embedding models in Table VIII. Similar observations can also be made that our proposed meta-tail2vec+ can significantly outperform the corresponding base embedding models and the refinement baselines, and this further demonstrates the effectiveness of meta-tail2vec+ on HINs.

2) Comparison of Head Node Embeddings: Similar to the comparison of head node embeddings illustrated in Sect. V-B2, here we follow the same setting to evaluate their performance, and show the results for node classification and link prediction in Figs. 8(a) and (b), respectively. Note that, here head nodes only contain nodes from one type on each dataset, *i.e.*, Paper nodes on *DBLP* or Movie nodes on *MovieLens*. As illustrated in Sect. V-B2, in both downstream applications, meta-tail2vec is able to achieve a slight performance lift on head nodes compared to the base embedding model, as the improved tail nodes might further boost the performance of head nodes via neighborhood aggregation. Similar observations can also be made in Fig. 8. In particular, meta-tail2vec+ can outperform the base embedding model and other baselines in terms of both node classification and link prediction on head nodes. This further demonstrates the intuition that the performance of head nodes is indirectly influenced by the refined tail node embeddings. Again, our goal is to improve tail node embeddings while head node embeddings remain robust.

VII. Conclusions

In this paper, we investigated the problem of tail node embedding on both homogeneous and heterogeneous graphs. We first formulated the problem as an instance of few-shot regression, and proposed a novel approach *meta-tail2vec* for refining tail node embeddings. In particular, to personalize each tail node given its local contexts, we designed a locality-aware task generation strategy to capture the prior knowledge across nodes at different localities. We further extend *meta-tail2vec* into *meta-tail2vec+* to facilitate the locality-aware tail node embeddings on HINs. Finally, extensive experiments demonstrated their promising performance on both node classification and link prediction.

Acknowledgements

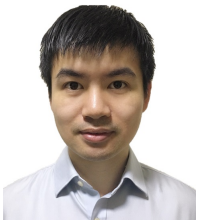
This research / project is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (Proposal ID: T2EP20122-0041), and the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-RP-2018-001). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore or National Research Foundation, Singapore.

References

[1] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*, 2014, pp. 701–710.
[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[3] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang, "Network representation learning with rich text information," in *IJCAI*, 2015, p. 2111–2117.
[4] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017, pp. 1024–1034.
[5] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, "Collaborative filtering and deep learning based recommendation system for cold start items," *Expert Systems with Applications*, vol. 69, pp. 29–39, 2017.
[6] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *TACL*, vol. 5, pp. 135–146, 2017.
[7] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han, "Heterogeneous network representation learning: A unified framework with survey and benchmark," *TKDE*, 2020.
[8] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *ICML*, 2016, pp. 1842–1850.
[9] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017, pp. 1126–1135.
[10] Z. Hu, T. Chen, K.-W. Chang, and Y. Sun, "Few-shot representation learning for out-of-vocabulary words," in *ACL*, 2019, pp. 4102–4112.
[11] H. Lee, J. Im, S. Jang, H. Cho, and S. Chung, "Melu: Meta-learned user preference estimator for cold-start recommendation," in *SIGKDD*, 2019, pp. 1073–1082.
[12] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," in *AAAI*, 2018.
[13] Z. Liu, Y. Fang, C. Liu, and S. C. Hoi, "Node-wise localization of graph neural networks," in *IJCAI*, 2021.
[14] Z. Liu, W. Zhang, Y. Fang, X. Zhang, and S. C. Hoi, "Towards locality-aware meta-learning of tail node embeddings on networks," in *CIKM*, 2020, pp. 975–984.
[15] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *TKDE*, vol. 30, no. 9, pp. 1616–1637, 2018.
[16] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016, pp. 855–864.
[17] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *CIKM*, 2015, pp. 891–900.
[18] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *SIGKDD*, 2016, pp. 1105–1114.
[19] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*, 2015, pp. 1067–1077.
[20] C. Yang, M. Liu, V. W. Zheng, and J. Han, "Node, motif and subgraph: Leveraging network functional blocks through structural convolution," in *ASONAM*, 2018, pp. 47–52.
[21] Y. Fang, W. Lin, V. W. Zheng, M. Wu, K. C.-C. Chang, and X.-L. Li, "Semantic proximity search on graphs with metagraph-based learning," in *ICDE*, 2016, pp. 277–288.
[22] Q. Mao, Z. Liu, C. Liu, and J. Sun, "Hinormer: Representation learning on heterogeneous information networks with graph transformer," in *WWW*, 2023, pp. 599–610.
[23] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *SIGKDD*, 2017, pp. 135–144.
[24] T.-y. Fu, W.-C. Lee, and Z. Lei, "Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning," in *CIKM*, 2017, pp. 1797–1806.
[25] W. Zhang, Y. Fang, Z. Liu, M. Wu, and X. Zhang, "mg2vec: Learning relationship-preserving heterogeneous graph representations via meta-graph embedding," *TKDE*, 2020.
[26] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *WWW*, 2019, pp. 2022–2032.

[27] X. Fu, J. Zhang, Z. Meng, and I. King, "Magnn: metapath aggregated graph neural network for heterogeneous graph embedding," in *WWW*, 2020, pp. 2331–2341.
[28] Q. Lv, M. Ding, Q. Liu, Y. Chen, W. Feng, S. He, C. Zhou, J. Jiang, Y. Dong, and J. Tang, "Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks," in *SIGKDD*, 2021, pp. 1150–1160.
[29] R. Cai, X. Chen, Y. Fang, M. Wu, and Y. Hao, "Dual-dropout graph convolutional network for predicting synthetic lethality in human cancers," *Bioinformatics*, 2020.
[30] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "GraphGAN: Graph representation learning with generative adversarial nets," in *AAAI*, 2018, pp. 2508–2515.
[31] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *IJCAI*, 2018, pp. 2609–2615.
[32] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *SIGKDD*, 2016, pp. 1225–1234.
[33] M. Volkovs, G. Yu, and T. Poutanen, "DropoutNet: Addressing cold start in recommender systems," in *NeurIPS*, 2017, pp. 4957–4966.
[34] J. Li, M. Jing, K. Lu, L. Zhu, Y. Yang, and Z. Huang, "From zero-shot learning to cold-start recommendation," in *AAAI*, 2019, pp. 4189–4196.
[35] Y. Pinter, R. Guthrie, and J. Eisenstein, "Mimicking word embeddings using subword RNNs," in *EMNLP*, 2017, pp. 102–112.
[36] Z. Liu, T.-K. Nguyen, and Y. Fang, "Tail-gnn: Tail-node graph neural networks," in *SIGKDD*, 2021, pp. 1109–1119.
[37] J. Wu, J. He, and J. Xu, "Demo-net: Degree-specific graph neural networks for node and graph classification," in *SIGKDD*, 2019.
[38] X. Tang, H. Yao, Y. Sun, Y. Wang, J. Tang, C. Aggarwal, P. Mitra, and S. Wang, "Investigating and mitigating degree-related biases in graph convolutional networks," in *CIKM*, 2020.
[39] J. Kang, Y. Zhu, Y. Xia, J. Luo, and H. Tong, "Rawlsgcn: Towards rawlsian difference principle on graph convolutional network," in *WWW*, 2022, pp. 1214–1225.
[40] R. Wang, X. Wang, C. Shi, and L. Song, "Uncovering the structural fairness in graph contrastive learning," *NeurIPS*, vol. 35, pp. 32 465–32 473, 2022.
[41] Z. Liu, T.-K. Nguyen, and Y. Fang, "On generalized degree fairness in graph neural networks," *AAAI*, no. 4, pp. 4525–4533, 2023.
[42] B. Kang, S. Xie, M. Rohrbach, Z. Yan, A. Gordo, J. Feng, and Y. Kalantidis, "Decoupling representation and classifier for long-tailed recognition," in *ICLR*, 2020.
[43] J. Liu, Y. Sun, C. Han, Z. Dou, and W. Li, "Deep representation learning on long-tailed data: A learnable embedding augmentation perspective," in *CVPR*, 2020, pp. 2970–2979.
[44] K. Tang, J. Huang, and H. Zhang, "Long-tailed classification by keeping the good and removing the bad momentum causal effect," *NeurIPS*, 2020.
[45] M. Shi, Y. Tang, X. Zhu, D. Wilson, and J. Liu, "Multi-class imbalanced graph convolutional network learning," in *IJCAI*, 2020, pp. 2879–2885.
[46] T. Zhao, X. Zhang, and S. Wang, "Graphsmote: Imbalanced node classification on graphs with graph neural networks," in *WSDM*, 2021, pp. 833–841.
[47] L. Qu, H. Zhu, R. Zheng, Y. Shi, and H. Yin, "Imgagn: Imbalanced network embedding via generative adversarial graph networks," *SIGKDD*, 2021.
[48] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," in *NeurIPS*, 2016, pp. 3630–3638.
[49] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *NeurIPS*, 2017, pp. 4077–4087.
[50] X. Han, H. Zhu, P. Yu, Z. Wang, Y. Yao, Z. Liu, and M. Sun, "FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation," in *EMNLP*, 2018, pp. 4803–4809.
[51] F. Pan, S. Li, X. Ao, P. Tang, and Q. He, "Warm up cold-start advertisements: Improving ctr predictions via learning to learn id embeddings," in *SIGIR*, 2019, pp. 695–704.
[52] F. Zhou, C. Cao, K. Zhang, G. Trajcevski, T. Zhong, and J. Geng, "Meta-gnn: On few-shot node classification in graph meta-learning," in *CIKM*, 2019, pp. 2357–2360.
[53] Z. Liu, Y. Fang, C. Liu, and S. C. Hoi, "Relative and absolute location embedding for few-shot node classification on graph," in *AAAI*, 2021, pp. 4267–4275.
[54] H. Yao, C. Zhang, Y. Wei, M. Jiang, S. Wang, J. Huang, N. V. Chawla, and Z. Li, "Graph few-shot learning via knowledge transfer," in *AAAI*, 2020, pp. 6656–6663.
[55] J. Chauhan, D. Nathani, and M. Kaul, "Few-shot learning on graphs via super-classes based on graph spectral measures," in *ICLR*, 2020.
[56] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *ACL*, 2015, pp. 1556–1566.
[57] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019, pp. 4171–4186.
[58] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," in *ICLR*, 2017.
[59] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, "Local higher-order graph clustering," in *SIGKDD*, 2017, p. 555–564.
[60] A. Lazaridou, M. Marelli, and M. Baroni, "Multimodal word meaning induction from minimal exposure to natural text," *Cognitive science*, vol. 41, pp. 677–705, 2017.
[61] M. Khodak, N. Saunshi, Y. Liang, T. Ma, B. Stewart, and S. Arora, "A la carte embedding: Cheap but effective induction of semantic feature vectors," in *ACL*, 2018, pp. 12–22.
[62] A. Herbelot and M. Baroni, "High-risk learning: acquiring new word vectors from tiny data," in *EMNLP*, 2017, pp. 304–309.
[63] Y. Fang, W. Lin, V. W. Zheng, M. Wu, J. Shi, K. Chang, and X. Li, "Metagraph-based learning on heterogeneous graphs," *TKDE*, 2019.
[64] M. Ji, Y. Sun, M. Danilevsky, J. Han, and J. Gao, "Graph regularized transductive classification on heterogeneous information networks," in *ECML-PKDD*. Springer, 2010, pp. 570–586.
[65] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," *VLDB*, vol. 4, no. 11, pp. 992–1003, 2011.

Zemin Liu is currently a senior research fellow with the School of Computing, National University of Singapore. He received his Ph.D. degree in Computer Science from Zhejiang University, Hangzhou, China in 2018, and B.S. Degree in Software Engineering from Shandong University, Jinan, China in 2012. His research interests lie in graph embedding, graph neural networks, and learning on heterogeneous information networks.



Yuan Fang received his Ph.D. degree in Computer Science from the University of Illinois at Urbana-Champaign, United States in 2014, and Bachelor's degree in Computer Science from National University of Singapore, Singapore in 2009. He is currently an Assistant Professor at the School of Computing and Information Systems, Singapore Management University, Singapore. Previously, he was a scientist at the Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore and a data scientist at DBS Bank, Singapore. His current research focuses on graph-based machine learning, Web and social media mining, recommendation systems and bioinformatics.

Wentao Zhang received his Master and B.S. degrees in Computer Science from the University of Science and Technology of China (USTC) in 2021 and 2018, respectively, advised by Prof. Xinming Zhang. He was a Visiting Research Student at Singapore Management University between August 2019 and April 2020. He is currently an R&D staff member at Webank. His research interests include graph representation learning.

Xinming Zhang received the B.E. and M.E. degrees in Electrical Engineering from China University of Mining and Technology, Xuzhou, China in 1985 and 1988, respectively, and the Ph.D. degree in Computer Science and Technology from the University of Science and Technology of China, Hefei, China in 2001. Since 2002, he has been with the faculty of the University of Science and Technology of China, where he is currently a Professor with the School of Computer Science and Technology. From September 2005 to August 2006, he was a Visiting Professor with the Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, Korea. His research interest includes wireless networks, big data and deep learning. He has published more than 100 papers. He won the second prize of Science and Technology Award of Anhui Province of China in Natural Sciences in 2017. He won the awards of Top Reviewers (1%) in Computer Science & Cross Field by Publons in 2019. He is a senior member of IEEE.

Steven C.H. Hoi is currently the Managing Director of Salesforce Research Asia at Salesforce and oversees Salesforce's AI research and development activities in APAC. He is also the Professor of Computer Science (on leave) at the School of Computing and Information Systems, Singapore Management University. Formerly, he was a tenured Associate Professor of School of Computer Engineering at Nanyang Technological University, Singapore. He received his Bachelor degree in Computer Science from Tsinghua University, in 2002, and both his Master and PhD degrees in Computer Science and Engineering from the Chinese University of Hong Kong, in 2004 and 2006, respectively.