

MultiBiSage: A Web-Scale Recommendation System Using Multiple Bipartite Graphs at Pinterest

Saket Gurukar
The Ohio State University
gurukar.1@osu.edu

Nikil Pancha
Pinterest
npancha@pinterest.com

Andrew Zhai
Pinterest
andrew@pinterest.com

Eric Kim
Pinterest
erickim@pinterest.com

Samson Hu
Pinterest
sansonhu@pinterest.com

Srinivasan Parthasarathy
The Ohio State University
srini@cse.ohio-state.edu

Charles Rosenberg
Pinterest
crosenberg@pinterest.com

Jure Leskovec
Stanford
jure@cs.stanford.edu

ABSTRACT

Graph Convolutional Networks (GCN) can efficiently integrate graph structure and node features to learn high-quality node embeddings. At Pinterest, we have developed and deployed PinSage, a data-efficient GCN that learns pin embeddings from the Pin-Board graph. PinSage relies heavily on PinSage which in turn only leverages the Pin-Board graph. However, there exist several entities at Pinterest and heterogeneous interactions among these entities. These diverse entities and interactions provide important signal for recommendations and modeling. In this work, we show that training deep learning models on graphs that captures these diverse interactions can result in learning higher-quality pin embeddings than training PinSage on only the Pin-Board graph. However, building a large-scale heterogeneous graph engine that can process the entire Pinterest size data has not yet been done. In this work, we present a clever and effective solution where we break the heterogeneous graph into multiple disjoint bipartite graphs and then develop novel data-efficient MultiBiSage model that combines the signals from them. MultiBiSage can capture the graph structure of multiple bipartite graphs to learn high-quality pin embeddings. The benefit of our approach is that individual bipartite graphs can be processed with minimal changes to Pinterest’s current infrastructure, while being able to combine information from all the graphs while achieving high performance. We train MultiBiSage on six bipartite graphs including our Pin-Board graph and show that it significantly outperforms the deployed latest version of PinSage on multiple user engagement metrics. We also perform experiments on two public datasets to show that MultiBiSage is generalizable and can be applied to datasets outside of Pinterest.

PVLDB Reference Format:

Saket Gurukar, Nikil Pancha, Andrew Zhai, Eric Kim, Samson Hu, Srinivasan Parthasarathy, Charles Rosenberg, and Jure Leskovec. MultiBiSage: A Web-Scale Recommendation System Using Multiple Bipartite Graphs at Pinterest. PVLDB, 16(4): 781-789, 2022. doi:10.14778/3574245.3574262

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit <https://creativecommons.org/licenses/by-nc-nd/4.0/> to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights

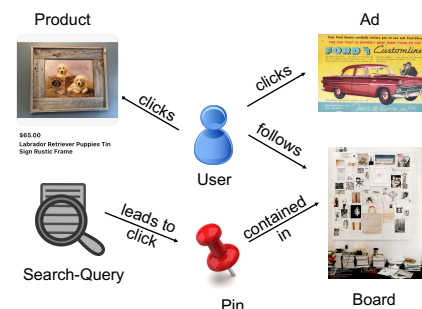


Figure 1: Pinterest’s Heterogeneous Graph

1 INTRODUCTION

Pinterest helps its users discover things they love by using 100+ billion pins present on its platform. The users discover things by exploring the personalized recommendations that are powered by Graph Convolutional Network (GCN) based recommendation system, PinSage[29]. PinSage aggregates the visual and textual features of pins along with their local graph neighborhood from the Pin-Board graph to generate the pin embeddings. The learned pin embeddings are then fed as input to several machine learning models at Pinterest for personalized recommendation, spam classification, ads recommendation, and search.

The PinSage [29] model currently relies heavily on only the Pin-Board bipartite graph to identify the local graph neighborhood of the pin. However, at Pinterest, we have multiple entities such as users, idea-pins, creators, products, ads, search-queries, boards, along with other entities. Moreover, these entities can have diverse interactions on Pinterest. Figure 1 shows an example heterogeneous graph at Pinterest. Here, a user can click on a product or an ad, a user can follow a board, a pin can belong to a board, a user can enter a search-query and then click on a pin shown in the search results.

licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 16, No. 4 ISSN 2150-8097.
doi:10.14778/3574245.3574262

We hypothesize that these interactions among entities contain rich signals that can be utilized to learn high-quality pin embeddings.

Recently, several extensions of Graph Convolutional Networks for heterogeneous graphs have been proposed [3, 15, 17, 24, 32]. These proposed models often operate on moderate-size to large-scale graphs. One such heterogeneous graph model, HGT [15], showed promising results on Open Academic Graph [33] consisting of 179 million nodes and 2 billion edges. However, the proposed version of HGT is trained on a single machine and requires all the training data (heterogeneous graph and node features) to be present on that single system. At Pinterest, in an arbitrary version of our graph, we have 2+ billion pins, 2+ billion boards, 7+ billion pin-board edges and 400+ million users [22]. The storage of graph and node features itself requires more than 3TB of space. Hence, *given the scale of the data at Pinterest, training existing heterogeneous graph models on a single machine is not feasible.*¹ There also exist a few frameworks such as Distributed Deep Graph Library (DistDGL) [34] and Aligraph [31] that support distributed training of heterogeneous GNNs models. DistDGL and Aligraph trained distributed GCNs models on Ogbn-papers100m [13] (~111 million nodes) and Taobao-large (~483 million nodes) graphs, respectively. Both DistDGL and Aligraph rely on multi-processing for "on-the-fly" sampling of node's neighbors. On a 4+ billion node graph, the "on-the-fly" sampling of neighbors in every epoch or after a few epochs, results in a high training time of the model. The running time of sampling can be decreased by increasing the number of workers but that results in additional costs. Moreover, the adoption of different external frameworks incurs significant costs in terms of infrastructure changes and data collection workflows. Hence, it is challenging to train the heterogeneous graph models on web-scale graphs.

Present work: In this work, we adopt a pragmatic approach that allows us to utilize the existing infrastructure developed at Pinterest to train heterogeneous graph models. The approach can be summarized as follows. Given a heterogeneous graph we first decompose it into multiple bipartite graphs. We then compute the local graph neighborhood of pins with the help of Pixie. To efficiently aggregate the node features and node neighbor features from multiple bipartite graphs, we propose a transformer based MultiBiSage model. Given k bipartite graphs, MultiBiSage learns k pin embeddings with the help of transformer [25] model where each pin embedding corresponds to each input bipartite graphs. These pin embeddings are then aggregated with another transformer layer that generates the final pin embedding. The benefit of our approach is that individual bipartite graphs can be processed with minimal changes to Pinterest's current infrastructure, while being able to combine information from all the graphs. This also means that **model power and expressivity does not have to sacrificed to achieve scalability.** Our proposed approach, addresses the *forementioned practical obstacles to the deployment of heterogeneous graph models at scale.*

The contributions of our work are summarized as follows

- (1) To the best of our knowledge, this is the largest-ever application of heterogeneous graph models on web-scale graphs with over 4.8+ billion nodes and 9.7+ billion edges – thereby *demonstrating the scalability of MultiBiSage.*
- (2) We show that our *evaluated solution*, MultiBiSage, significantly outperforms the currently deployed PinSage model.
- (3) MultiBiSage solves a *significant real-world problem* and we present several ablations studies and case study that helped us arrive at the design choices of MultiBiSage.

2 PRELIMINARIES

2.1 Definitions

Definition 2.1. Heterogeneous Graph: A heterogeneous graph $H = (V, E, T, R, X, \phi, \psi, \varphi)$ is a graph where V, E, T, R are the set of nodes, set of edges, set of node types, and set of relations, respectively. A node $v \in V$ has node type $\phi(v) : V \rightarrow T$ and an edge $e(u, v) \in E$ between two nodes u, v has edge type $\psi(e(u, v)) : E \rightarrow R$. A node v can have node features denoted by $\varphi(v) : V \rightarrow X$.

Heterogeneous graphs can naturally capture multi-modal interactions in the real-world. For instance, nodes can represent diverse entities such as users, pins, boards, products, and ads; while edges can represent diverse interactions types such as click, add-to-cart, and purchase.

Definition 2.2. Bipartite Graph: A bipartite graph $B = (V_1, V_2, E, X, \varphi)$ is a graph consisting of two sets of nodes V_1 and V_2 and an edge $e(u, v) \in E$ between two nodes $u \in V_1, v \in V_2$ has one edge type. A node v can have node features denoted by $\varphi(v) : V \rightarrow X$.

Definition 2.3. Heterogeneous Graph Embedding: Given a heterogeneous graph H , a heterogeneous graph embedding method learns a function $f(H) : V \rightarrow \mathbb{R}^d$ such that $\Theta = f(H)$ is a matrix consisting of d -dimensional node embeddings. The function f is learned such that similarity between nodes in the heterogeneous graph is approximated by the closeness between nodes in the embedding space.

In this work, we aim to learn an embedding function f' such that

$$f(H) \approx f'(\cup_{V_i} \cup_{V_j} \cup_{E_k} B(V_i, V_j, E_k, X, \varphi)) \quad (1)$$

as computing $f(H)$ might not always be feasible.

At Pinterest, a user interacts with a pin (denoted as *query pin*) and is shown recommendations. If the user interacts with one of the recommended pins this is denoted as an *engaged pin*. We treat the query and engaged pin as a positive pair. The goal of MultiBiSage is to learn pin embeddings such that the distance between query and engaged pin is close to each other.

3 METHODOLOGY

Given the scale of Pinterest data, we require an optimized data curation and ingestion pipeline to feed the bipartite graph features to the machine learning model. Moreover, we also require a novel model architecture to better capture the bipartite graph features in order to learn high-quality pin embeddings. We describe these details in this section.

¹If we assume that the computation of sampling node neighbors and computing node embedding from neighbors takes 1 millisecond by HGT, then it would still take 55 days to compute node embeddings of the whole 4.8+ billion node graph for ONE epoch.

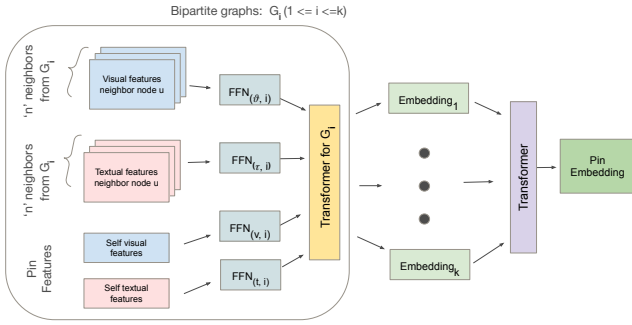


Figure 2: The model architecture of MultiBiSage.

3.1 MultiBiSage model architecture

Figure 2 shows the model architecture of MultiBiSage. Let p and x_p be the pin and embedding of pin p , respectively. Here $x_p \in \mathbb{R}^d$ is l_2 normalized d -dimensional vector. Let $v_p \in \mathbb{R}^{d_v}$ and $t_p \in \mathbb{R}^{d_t}$ be the visual and textual features of pin p , respectively. Assume there are k bipartite graphs with G_i being the i^{th} bipartite graph. Let n be the number of neighbors from each bipartite graph. Let $\vartheta_{i,p} \in \mathbb{R}^{n \times d_v}$ and $\tau_{i,p} \in \mathbb{R}^{n \times d_t}$ be the visual and textual features of the neighbors of pin p from G_i , respectively.

Mathematically, the MultiBiSage model can be described as follows. First, we pass the visual and textual features of pin and its neighbors through a feedforward neural network (FFN) to learn the intermediate representation of dimension size d_h .

$$\begin{aligned} x_{p,v_i} &= FFN(v_p) = ReLU(v_p W_{v_i} + b_{v_i}) \\ x_{p,t_i} &= FFN(t_p) = ReLU(t_p W_{t_i} + b_{t_i}) \\ x_{p,\vartheta_i} &= FFN(\vartheta_{i,p}) = ReLU(\vartheta_{i,p} W_{\vartheta_i} + b_{\vartheta_i}) \\ x_{p,\tau_i} &= FFN(\tau_{i,p}) = ReLU(\tau_{i,p} W_{\tau_i} + b_{\tau_i}) \end{aligned} \quad (2)$$

where $W_{v_i} \in \mathbb{R}^{d_v \times d_h}$, $W_{t_i} \in \mathbb{R}^{d_t \times d_h}$, $W_{\vartheta_i} \in \mathbb{R}^{d_v \times d_h}$, and $W_{\tau_i} \in \mathbb{R}^{d_t \times d_h}$ be the learnable weight matrices. Here, $b_{v_i} \in \mathbb{R}^{d_h}$, $b_{t_i} \in \mathbb{R}^{d_h}$, $b_{\vartheta_i} \in \mathbb{R}^{d_h}$ and $b_{\tau_i} \in \mathbb{R}^{d_h}$ are the biases parameters.

The intermediate representations $x_{v,i}$, $x_{t,i}$, x_{ϑ_i} , x_{τ_i} and a global token $x_{g_i} \in \mathbb{R}^{d_h}$ are then concatenated and passed to the transformer layer.

$$s_{p,i} = Concat(x_{g_i}, x_{v,i}, x_{t,i}, x_{\vartheta_i}, x_{\tau_i}) \quad (3)$$

where $s_{p,i} \in \mathbb{R}^{(1+2 \times (1+n)) \times d_h}$. Here, $s_{p,i}$ can be considered as a sequence with $1 + 2 \times (1 + n)$ number of tokens and each token is represented in d_h dimensional space.

We pass the sequence $s_{p,i}$ through the multihead attention layer [25] where each input token has three intermediate representations referred to as query (Q), key (K), and value (V). The attention (A) between all tokens is computed using the below equation.

$$A(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (4)$$

where d_k is dimension size of queries and keys. We apply attention with H number of heads as shown below:

$$\begin{aligned} MultiHead_i(Q, K, V) &= Concat(head_1, \dots, head_H)W^O \\ head_j &= A(QW_j^Q, KW_j^K, VW_j^V) \end{aligned} \quad (5)$$

where $W_j^Q \in \mathbb{R}^{d_h \times d_k}$, $W_j^K \in \mathbb{R}^{d_h \times d_k}$, $W_j^V \in \mathbb{R}^{d_h \times d_v}$ and $W^O \in \mathbb{R}^{Hd_v \times d}$ are trainable parameters and $d_k = d_v = d_h/H$.

The representation of global token x_{g_i} after passing it through multihead attention is treated as pin embedding $x_{p,i} \in \mathbb{R}^d$ of pin p from bipartite graph G_i . The pin embeddings $x_{p,i}$ are then passed to second transformer along with global token $x_g \in \mathbb{R}^d$ as shown below:

$$s = Concat(x_g, x_{p,1}, x_{p,2}, \dots, x_{p,k}) \quad (6)$$

where $s \in \mathbb{R}^{(1+k) \times d}$. The representation of global token x_g after passing it through transformer layer is L_2 normalized and considered as pin embedding $x_p \in \mathbb{R}^d$.

The time complexity of MultiBiSage is $O((1 + 2(1+n))^2 + k^2)$; since $k \ll (1+n)$, the final time complexity is $O((1+2(1+n))^2)$. The space complexity of MultiBiSage is $O(l \times d \times (1+2(1+n))^2 + l \times d \times k^2)$ where l is number of layers.²

3.1.1 Training objective. Let q_i , and e_i be the i^{th} query pin and engaged pin in a batch \mathcal{B} , respectively. Let $x_{q_i} \in \mathbb{R}^d$ and $x_{e_i} \in \mathbb{R}^d$ be the embedding of these pins computed by MultiBiSage. Also, let $C_{\mathcal{B}} = \{e_1, \dots, e_{|\mathcal{B}|}\}$ be the set of all the engaged pins in the batch. Then, we minimize the sampled softmax loss function [16] as shown below:

$$-\frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \log \frac{e^{\langle x_{q_i}, x_{e_i} \rangle} - \log Q_p(e_i | q_i)}{\sum_{e' \in C_{\mathcal{B}}} e^{\langle x_{q_i}, x_{e'} \rangle} - \log Q_p(e' | q_i)} \quad (7)$$


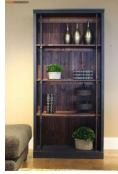



















where $\langle \cdot, \cdot \rangle$ represents the dot product between learned embeddings. Here, we are performing softmax over sampled classes $C_{\mathcal{B}}$ instead of performing softmax over all the classes. The softmax over-sampled classes introduce sampling bias in the full softmax computation. This bias is corrected through the proposal distribution $Q_p(e_i | q_i)$ which is the probability of e_i being included as a positive sample in the training batch. We utilize count-min sketch [4] to estimate $Q_p(e_i | q_i)$ in a streaming manner.

Notice that, for query-pin q_i , we are treating all the other engaged pins e_j ($e_j \neq e_i$) in the batch as negative samples. If a pin is popular, it would frequently appear as an engaged pin (say e_k) in the batch. The above loss function would then unfairly penalize engaged pin e_k , as they are more likely to be selected as negative pin during training. To address this issue, we adopt the mixed negative sampling approach [27], in which we select a set of random negatives \mathcal{M} where $|\mathcal{M}| = |\mathcal{B}|$ and compute the below loss function.

$$-\frac{1}{|\mathcal{M}|} \sum_{i=1}^{|\mathcal{M}|} \log \frac{e^{\langle x_{q_i}, x_{e_i} \rangle} - \log Q_n(e_i)}{\sum_{e' \in \mathcal{M}} e^{\langle x_{q_i}, x_{e'} \rangle} - \log Q_n(e')} \quad (8)$$

² Transformer baseline: if we simply concatenation all the neighbors of all the graphs and then pass to Transformer, the time complexity would be $O((k \times (1+2(1+n)))^2)$. The space complexity of transformer is $O(l \times d \times ((k \times (1+2(1+n))))^2)$.

Table 1: Candidate pin and its neighbors from diverse bipartite graphs. Note, the candidate pin neighbors from diverse graphs seems to follow nonsimilar data distribution. For instance, the candidate neighbors from SearchQuery-Pin include pins that follow different color and style than that of Pin-Board.

Candidate	Graph	Rank-1	Rank-2	Rank-3	Rank-4	Rank-5
	Pin-Board					
	User-Product					
	User-Ad					
	SearchQuery-Pin					

where $Q_n(e)$ represents the probability of sampling pin e . We also utilize count-min sketch [4] to estimate $Q_n(e)$ in a streaming manner. We optimize both the loss function in Equation 7 and Equation 8 by summing them with equal weightage.

3.2 Training Pipeline

Algorithm 1 shows the training pipeline of MultiBiSage. The first step is the construction of bipartite graphs from the user’s interaction logs at Pinterest. However constructing these graphs on a single machine is time-consuming and not practical. Hence, we develop a Spark-based graph generator, which given the two entity-types and the interaction type, curates the corresponding bipartite graph by parsing the logs. The Spark-based graph generator also includes a degree-based graph pruning algorithm that can reduce the number of the edges in the graph. The algorithm can be described as follows: let a and b be the specified minimum and maximum node degrees and let p be the pruning factor between 0 and 1. The pruning algorithm selects node (say u) with node degree (say d_u) greater than a and then randomly removes the edges incident on node u such that new node degree of node u is $\min(d_u * p, b)$. To construct diverse bipartite graphs, we are required to extract diverse interactions from diverse data sources, so we execute a dedicated

Algorithm 1 Training Pipeline of MultiBiSage

- 1: Construct the bipartite graphs from the logs at Pinterest.
- 2: Get the local graph structure of the nodes from the bipartite graphs.
 - a: Perform random-walks on each bipartite graph with the help of in-house random-walker Pixie system.
 - b: Select top-k highly visited neighbors of the nodes from the random-walks.
- 3: Collect training data.
 - a: Curate query-pin and engaged-pin from multiple surfaces at Pinterest.
 - b: Collect the visual and textual features of query pin, engaged pin and their neighbors.
 - c: Collect a random sample of million pins and treat these pins as negative samples.
- 4: Train MultiBiSage model with a modified Sampled Softmax [16].

Spark job for each specific bipartite graph. To ensure temporal consistency we require that the graph construction time-period is the same across all the bipartite graphs.

Once the bipartite graphs are created, our next goal is to identify the neighbors of pin nodes from the bipartite graphs. The "on-the-fly" sampling of neighbors during model training from billion-nodes and billion-edges graph is extremely inefficient and not practical.

Table 2: The statistics of Pinterest’s bipartite graphs. Graph nomenclature convention: Entity 1 - Entity 2 - EdgeType. Graphs represent a random subset of users, pins, boards, queries, ads, products, creators, idea pins, and video pins.

Graphs	Num. Entity 1	Num. Entity 2	Num. Edges
Pin-Board-ContainedIn	1,995,423,350	1,927,181,284	7,144,399,779
User-Product-LC	74,130,469	40,198,339	472,307,057
User-AD-LC	114,059,486	9,392,321	710,736,288
SearchQuery-Pin-LC	397,290,595	189,846,404	1,307,227,847
Creator-IdeaPin-Created	1,508,535	12,349,297	13,984,765
IdeaPin-User-FollowedBy	2,502,566	41,002,754	98,570,789

Hence, we identify the node neighbors before training MultiBiSage using Pixie [6] system. Pixie system loads the entire graph in 2 TB RAM machine and performs random-walks using optimized C++ threads. For each pin node, we select top-k highly visited pin neighbors. Note that an ad or product is also considered as a pin at Pinterest. Table 1 shows a sample of the top-5 highly visited pins of a candidate pin. Next, we collect the training samples and the features of the training samples with the help of Spark jobs.

4 EXPERIMENTS

4.1 Experiment Details

4.1.1 Dataset. We perform experiments on the six bipartite graphs curated from the logs present at Pinterest. The statistics of these graphs are present in Table 2. All the graphs are collected over the period of one year and represent a random subset of users, pins, boards, search-queries, ads, products, creators, idea pins, and video pins. The edge type LC refers to the long click. A long click refers to the interaction where a user clicks on a pin and does not return to the Pinterest website in 10 secs. LC often acts as a signal that the user found the item that he/she was searching for. The raw Pin-Board graph consists of 100+ billion edges. We perform the degree-based pruning to reduce the number of edges from 100+ billion to 7 billion by setting the parameters as follows: minimum node degree=10, maximum node degree=10000 and prune factor=0.86. We apply the degree-based pruning to only pin-board graph. In the case of the SearchQuery-Pin-LC graph, each search-query text is considered as a node, and no text processing is performed on the search-query text. We select 50 neighbors from every bipartite graph. Table 3 contains the training data statistics on which we train all the models. The training data is collected over a certain temporal period. Additional technical details are present in our technical report [11]. We train the models with Adam optimizer with a learning rate of 0.002 and batch size of 8032. We gradually increase learning rate in optimizer [10] using Cosine Annealing [21].

4.1.2 Technical Details. The node features details of the Pinterest dataset are summarized as follows. i) Visual: A 1024-dimensional unified visual embedding computed through a large-scale Transformer-based pretraining [1]. ii) Textual: A 64-dimensional embedding learned by Pintext system [36]. Training details: For the model parameters, we use three-layer FeedForward Networks for visual features: 1024 → 2048 → 512. The visual features FFN have dropout

Table 3: The training data volume collected over a certain temporal period.

Statistic	Count
Number of distinct Query Pins	158,489,849
Number of distinct Engaged Pins	150,706,623
Number of distinct Query-Engaged Pins	573,056,337

set to 0.25 and have ReLU activations. The textual features FFN consist of 1 linear layer: 64 → 512. We set the batch size to 8032 and the number of epochs to 100k. The output of the MultiBiSage model is L2-normalized 256- dimensional pin embeddings. The number of the attention layers is set to 2 as more layers results in slower inference time. We iterate over 100,000 steps where a batch is processed in each step. The graph construction and feature extraction jobs execute on the Spark cluster with 900 executors each with 40g executor and 40g driver memory. Model training is done in a distributed manner with two p3dn.24xlarge instances. Each p3dn.24xlarge instance has 96 custom vCPUs, 768 GB CPU Memory, and 8 NVIDIA V100 Tensor Core GPUs with 32 GB of memory each.

4.1.3 Baselines. We primarily focus on the comparison of the state-of-the-art production model PinSage currently deployed at Pinterest. We also include advanced deep learning models and a few of their variants. The baselines we consider are

Transformer [25]: We pass the visual and textual embeddings of the neighbors from the bipartite graphs to the model. The five additional bipartite graphs increase the sequence length passed to the transformer by 5x. This model can be considered as an extension of the deployed PinSage model to the six bipartite graphs.

SharedTransformer: Instead of having a transformer for every bipartite graph, we have one transformer model that is shared across multiple bipartite graph features.

NFFNTransformer: Here, the bipartite graph embeddings are passed through a feed forward neural network layer which aggregates the embeddings from all the bipartite graphs.

NSumTransformer: We perform an element-wise sum of all the bipartite graph embeddings and then pass the resultant embeddings to the transformer.

NHadamardTransformer: We perform the Hadamard product of all the bipartite graph embeddings and then pass the resultant embeddings to the transformer.

PinFeatToLastTransformer: Here, we modify the MultiBiSage model and pass the pin-features to only the last transformer layer instead of repeatedly passing them to every bipartite graph transformer.

AggregateByFFN: Here, we modify the MultiBiSage model and replace the last transformer layer with a feed forward neural network that aggregates the bipartite graph pin embeddings ($x_{p,i}$).

4.1.4 Metrics. We measure the performance of our recommendations on multiple types of engagement. These engagements happen on different surfaces at Pinterest. The organic surface refers to pins present on the home feed (the landing page with an endless set

Table 4: Baselines. The percentages refer to the percentage improvement over deployed PinSage model.

Surface/Entity Type	Engagement Type	Transformer	Shared Transformer	NSum Transformer	NHadamard Transformer	NFFN Transformer	PinFeatToLast Transformer	Aggregate ByFFN
Organic Surface	Add-to-cart Checkout	0.900 (+0.8%)	0.902 (+1.0%)	0.901 (+0.9%)	0.899 (+0.7%)	0.900 (+0.8%)	0.874 (-2.1%)	0.885 (-0.9%)
		0.898 (+0.6%)	0.900 (+0.8%)	0.900 (+0.8%)	0.901 (+0.9%)	0.902 (+1.0%)	0.875 (-2.0%)	0.885 (-0.9%)
Ads	Good-click-through	0.670 (+2.9%)	0.676 (+3.8%)	0.669 (+2.8%)	0.668 (+2.6%)	0.672 (+3.2%)	0.629 (-3.4%)	0.661 (+1.5%)
Related Products	Long-click	0.740 (+1.0%)	0.738 (+0.7%)	0.738 (+0.7%)	0.737 (+0.5%)	0.741 (+1.1%)	0.708 (-3.4%)	0.741 (+1.1%)
Idea Pin	Close-ups Repin	0.749 (+3.5%)	0.757 (+4.6%)	0.747 (+3.2%)	0.746 (+3.0%)	0.752 (+3.9%)	0.729 (+0.7%)	0.760 (+5.0%)
		0.827 (+3.8%)	0.833 (+4.5%)	0.829 (+4.0%)	0.824 (+3.4%)	0.827 (+3.8%)	0.808 (+1.4%)	0.837 (+5.0%)
Video	Close-ups Repin	0.449 (+0.7%)	0.454 (+1.8%)	0.451 (+1.1%)	0.453 (+1.6%)	0.451 (+1.1%)	0.432 (-3.1%)	0.463 (+3.8%)
		0.578 (+1.2%)	0.581 (+1.8%)	0.575 (+0.7%)	0.573 (+0.4%)	0.578 (+1.2%)	0.553 (-3.2%)	0.591 (+3.5%)

Table 5: Recommendation performance: Recall@10

Surface/Entity Type	Engagement Type	PinSage	MultiBiSage
Organic Surface	Add-to-cart Checkout	0.893	0.907 (+1.57%)
		0.893	0.907 (+1.57%)
Ads	Good-click-through	0.651	0.684 (+5.07%)
Related Products	Long-click	0.733	0.749 (+2.18%)
Idea Pin	Close-ups Repin	0.724	0.776 (+7.18%)
		0.797	0.849 (+6.52%)
Video Pin	Close-ups Repin	0.446	0.47 (+5.38%)
		0.571	0.603 (+5.60%)

of recommended pins) surface. The related products surface corresponds to the page where a user clicks on a pin and is shown pin recommendations. Entities such as Ads, Idea Pins, and Video Pins can be present on both organic and related products surface. The users can have various types of engagement over these surfaces and entities. The Add-to-Cart and Checkout engagement corresponds to the user’s actions related to purchasing. The Good-click-through engagement occurs when the user is displayed an ad and after clicking the ad, the user does not return to the website in less than 30 secs. In Long click engagement, the user clicks on a pin and does not return back to the website in less than 10 secs. Close-ups refer to the action where the user zooms in into the pin. Repin refers to the action where the user saves the pin to his/her board.

Our goal is to improve the quality of user engagement by providing better recommendations. We measure recommendations quality using the recall@10 metric. Given a pair of query pin q and engaged pin e , we first generate the query pin embedding x_q and engaged pin embedding x_e . We additionally generate the embeddings of one million randomly sampled pins. We then find the ten nearest neighbors pins of query pin q using x_q in the embedding space. The metric Recall@10 computes the fraction of times the ground-truth engaged pin e appears in the top-10 nearest neighbors of query pin.

4.2 Results

4.2.1 Comparison with Baselines: In Table 5, we compare MultiBiSage against the currently deployed PinSage version at Pinterest. The percentages in Table 5 in column MultiBiSage refers to the percentage improvement over PinSage model. We observe that

MultiBiSage outperforms PinSage on all the performance metrics. The difference in performance between MultiBiSage and PinSage is statistically significant with standard paired t-test at significance level 0.01. On Ads, Idea Pins, and Video Pins, we observe more than 5% improvement over PinSage. These results shows that training MultiBiSage on graphs that captures diverse interactions result in learning higher-quality pin embeddings than training PinSage on only Pin-Board graph. This experiment demonstrates **the efficacy and scalability** of MultiBiSage on real-world heterogeneous data.














We compare the performance of MultiBiSage with the baselines in Table 4. The percentages shown in parenthesis refer to the percentage improvement over the deployed PinSage model. The difference in performance between MultiBiSage and the baselines is statistically significant with standard paired t-test at significance level 0.01. We also observe that most of the baselines result in the improvement of recommendation performance over PinSage. This is partly due to the fact that these models can utilize the additional neighborhood context of pins from diverse bipartite graphs. Next, we observe that the distribution of the node’s neighbors from each bipartite graph is different. This can be inferred from the performance of the Shared-Transformer model where the transformer is shared across multiple bipartite graph features. The models NSum-Transformer, NHadamard-Transformer, and NFFN-Transformer recommendation is similar to that of Transformer. The difference between them is not statistically significant with standard paired t-test at significance level 0.01. The MultiBiSage model variants PinFeatToLastTransformer and AggregateByFFN recommendation performance is poorer than that of MultiBiSage.

4.2.2 Ablation Study: Next, we study the impact of incorporating each bipartite graph on the MultiBiSage performance. The result of this ablation study is in shown in Table 6. In Table 6, we train the MultiBiSage with each introduced bipartite graph along with Pin-Board graph. From Table 6, we observe MultiBiSage+UserAd+PB achieves over 4.3% improvement on Ads over deployed PinSage. At the same time, both MultiBiSage+CCI+PB and MultiBiSage+IUF+PB that are trained on graphs containing Idea pins show from 3.18% to 4.42% improvement on Idea pins over the PinSage. Since Idea pins can also contain videos, we observe that these two models show improvement over videos pin recommendation tasks. The model MultiBiSage+UserProd+PB and MultiBiSage+QueryPin+PB show improvement in the recommendation performance across multiple metrics. However, the best performance on all the metrics

Table 6: Ablation Study on Bipartite Graphs. PB, UserProd, UserAd, QueryPin, CCI, and IUF refers to Pin-Board-Contained, User-Product-LC, User-Ad-LC, SearchQuery-Pin-LC, Creator-IdeaPin-Created, IdeaPin-User-Follow bipartite graphs, respectively.

Surface/Entity Type	Engagement Type	MultiBiSage+ UserProd+PB	MultiBiSage+ UserAd+PB	MultiBiSage+ QueryPin+PB	MultiBiSage+ CCI+PB	MultiBiSage+ IUF+PB
Organic Surface	Add-to-cart Checkout	0.897 (+0.45%)	0.899 (+0.67%)	0.899 (+0.67%)	0.897 (+0.45%)	0.897 (+0.45%)
		0.898 (+0.56%)	0.896 (+0.34%)	0.897 (+0.45%)	0.897 (+0.45%)	0.897 (+0.45%)
Ads	Good-click-through	0.660 (+1.38%)	0.679 (+4.30%)	0.677 (+3.99%)	0.662 (+1.69%)	0.658 (+1.08%)
Related Products	Long-click	0.737 (+0.55%)	0.739 (+0.82%)	0.737 (+0.55%)	0.738 (+0.68%)	0.735 (+0.27%)
Idea Pin	Close-ups Repin	0.742 (+2.49%)	0.751 (+3.73%)	0.746 (+3.04%)	0.747 (+3.18%)	0.756 (+4.42%)
		0.814 (+2.13%)	0.822 (+3.14%)	0.815 (+2.26%)	0.824 (+3.39%)	0.833 (+4.52%)
Video	Close-ups Repin	0.459 (+2.91%)	0.463 (+3.81%)	0.461 (+3.36%)	0.465 (+4.26%)	0.461 (+3.36%)
		0.586 (+2.63%)	0.591 (+3.50%)	0.588 (+2.98%)	0.587 (+2.80%)	0.589 (+3.15%)

Table 7: Case study

Query-Pin	Model	Engaged-Pin	Rank-1	Rank-2	Rank-3	Rank-4	Rank-5
	PinSage	 Rank: 364					
	MultiBiSage	 Rank: 5					

is achieved by MultiBiSage trained with all six bipartite graphs. The difference in the performance of MultiBiSage with all the models is statistically significant with pair t-test at significance level 0.01.

4.2.3 Case Study: We present a case study in Table 7 where a user first interacts with the shown query pin (a foldable electric scooter) and then immediately interacts with the engaged pin. The query pin is contained in a board titled "Electric Vehicles" and that board also contains cars. As a result, the top-5 recommendations provided by the PinSage model contain vehicles including cars. However, the engaged pin's embedding learned by PinSage is not too similar to that of query pin's embedding, as the rank of engaged-pin – computed based on the similarity between the query and engaged pin's embeddings – is 364. On the other hand, MultiBiSage learns the engaged pin's embedding closer to that of query-pin. As a result, engaged pin's rank is 5 with MultiBiSage – a positive recall with ten nearest neighbors. In addition, we see that the other 3 nearest pins retrieved by MultiBiSage are also related to electric scooters.

4.2.4 Other Datasets. To illustrate the broader applicability of our work we compare MultiBiSage with state-of-the-art heterogeneous

graph models [15, 25, 26, 30]. Since, graph models are highly dependent on the particular choice of train/validation/test split [23], we compute three train/validation/test splits of the dataset using different random seeds and report average test performance of the model. We use 60%/20%/20% ratios as train/val/test splits. Dataset specific statistics are reported in Table 8. We selected OBG-MAG dataset as it is one of the relatively big publicly available dataset and IMDB dataset so that compute and memory intensive methods such as HAN can be included in the experimental comparison.

The hyper-parameters evaluated for each model are as follows: 1. HGT: heads = [2,4,6], layers=[1,2, 4]. 2. HAN: heads = [2,4,6], metapaths of IMDB= [(('movie', 'actor'), ('actor', 'movie')), (('movie', 'director'), ('director', 'movie'))], metapaths of OBG MAG = [(('paper', 'author'), ('author', 'paper')), (('paper', 'field'), ('field', 'paper')), (('paper', 'paper'), ('paper', 'paper'))] 3. Transformer: hidden channels=[128, 256, 512], heads = [2,4,6], layers=[1,2,4]. 4. NARS: number of hops = [2,4,6]. 5. MultiBiSage: heads = [2,4,6], layers=[1,2,4]. We tuned hidden channels=[128, 256, 512] for all the models. The learning rate and number of epochs for all the models is set to 0.001

We use torch-geometric [7] implementations for HGT [15] and HAN [26]. For NARS [30], we use <https://github.com/facebookresearch/NARS> implementation

Table 8: Public dataset statistics.

Dataset	Nodes	Edges
OGB_MAG	# Papers: 736,389	# Paper - Author: 7,145,660
	# Authors : 1,134,649	# Paper - Paper: 5,416,271
	# Field-of-study: 59,965	# Paper - Field: 7,505,078
	# Institutions: 8,740	# Author - Institution: 1,043,998
IMDB	# Movies: 4,278	# Movie - Director: 4,278
	# Directors: 2,081	# Movie - Actor: 12,828
	# Actors: 5,257	

Table 9: Average test micro-f1 score of the models.

Model	OGB_MAG [14]	IMDB [8]
HGT [15]	0.485 (± 0.001)	0.625 (± 0.015)
HAN [26]	OOM	0.617 (± 0.010)
NARS [30]	0.531 (± 0.001)	0.662 (± 0.012)
Transformer [25]	0.511 (± 0.004)	0.639 (± 0.014)
MultiBiSage	0.541 (± 0.003)	0.675 (± 0.013)

and 200, respectively. In case of OGB_MAG dataset, we use the metapath2vec embedding for the entities as their features. All the models are evaluated on same three train/val/test splits and are trained on same set of features for fair comparison.

We perform hyper parameter tuning for all the models and report the test micro-f1 in Table 9. The values in parenthesis in Table 9 denoted standard deviation. We observe that MultiBiSage is competitive with these baselines on these datasets. This result shows that MultiBiSage is generalizable and can be applied to datasets outside of Pinterest.

5 RELATED WORK

5.1 Related Work

5.1.1 Heterogeneous Graph Neural Networks: HAN [26] proposed a hierarchical, node-level and semantic-level attention based heterogeneous graph neural network. HAN requires performing random-walks based on meta-paths. HetGNN [31] proposed a random-walk with restart strategy that first samples heterogeneous neighbor nodes and then aggregates them based on neighbor features with Bi-LSTM. GATNE[2] utilizes heterogeneous skip-gram objective function to learn node embeddings of attributed multiplex heterogeneous graphs. HGCN [35] proposed a heterogeneous graph convolutional network that can learn fine-grained relational features of the heterogeneous graph. HGT [15] is a scalable model that can handle dynamic heterogeneous graphs. HGT introduced heterogeneous mutual attention and heterogeneous message passing to perform convolutions on heterogeneous graphs. Note that, these heterogeneous graph models are trained on a single-machine system and training them on Pinterest scale data is not feasible. Additionally training these models on a distributed setup requires solving many challenges. NARS [30], a popular technique which features heavily in the leaderboard of OGB benchmarks, performs sampling of relation subgraph based on sample relation subset and computes l^{th} power of adjacency matrix followed by a multilayer

perceptron to classify. Sampling and computing the l^{th} power of an adjacency matrix on Pinterest scale graph is simply not feasible on a single machine. To the best of our knowledge a distributed framework for running NARS does not exist.

5.1.2 Distributed Frameworks for Heterogeneous Graph Models:

The distributed training frameworks for heterogeneous graphs in general addresses three main scalability challenges related to storage, sampling and GNN operations. The AliGraph [31] system address these challenges by proposing three layers: i) Storage layer partitions the graphs using four graph partition algorithms and introduces indexing and caching operations for retrieval of node and its neighbors features, ii) Sampling layer identify the node’s neighbors which are required by GNN. Sampling is efficient as each worker sample neighbors on the assigned graph partition, and iii) Operations layers that implement aggregation and combination operation of GNN in an optimized manner. DistDGL [34] stores the graphs on multiple devices by partitioning the graph using METIS [18] algorithm. The node features are stored using a distributed key-value store. For sampling, DistDGL introduced a distributed sampler that follows client-server model to samples the node neighbors in parallel. DistDGL perform random-walks on each partition and a single random-walk does not move across different partitions. P3 [9] proposed a novel distributed training approach that eliminates partitioning overheads and high communication among workers. P3 introduced a pipelined push-pull parallelism based execution strategy for the distributed training of GNNs. We estimate that the migration to these new frameworks in the Pinterest software ecosystem would also incur significant infrastructure costs.

6 CONCLUSIONS AND FUTURE WORK

MultiBiSage offers a simple yet innovative approach for harnessing heterogeneous signals from multiple interactions in Pinterest. It leverages multiple bipartite graphs (each capturing diverse entities and different types of interactions) to learn high-quality pin embeddings on a production system at Pinterest. A key novelty is how we aggregate the pin features along with the local graph structure of nodes from multiple bipartite graphs within our proposed MultiBiSage model. The architectural design choices in MultiBiSage **facilitates high performance and scalability** to Pinterest-scale (billion-node) graphs all while achieving a significant lift (**greater than 7%**) in performance over the current production system at Pinterest - **a significant milestone**. In the future, we propose to examine if the scalability of MultiBiSage can be further enhanced by leveraging a high performance multi-level approach [5, 12, 19]. Independently, we also plan to examine if MultiBiSage can be adapted to partially labeled attributed graph models [20, 28].

ACKNOWLEDGMENTS

This work was initiated when the first author (SG) was an intern at Pinterest and subsequently enhanced after returning to OSU. SG and SP were partially supported by the NSF AI-Edge grant CNS-2112471, and would like to acknowledge the NSF MRI OAC-2018627 grant for additional research experimentation resources. Any opinions, findings, and conclusions in this material are those of the author(s) and may not reflect the views of the respective funding agency.

REFERENCES

- [1] Josh Beal, Hao-Yu Wu, Dong Huk Park, Andrew Zhai, and Dmitry Kislyuk. 2022. Billion-scale pretraining with vision transformers for multi-task visual representations. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 564–573.
- [2] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation learning for attributed multiplex heterogeneous network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1358–1368.
- [3] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. 2015. Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 119–128.
- [4] Graham Cormode and Shan Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*.
- [5] Chenhui Deng, Zhiqiang Zhao, Yongyu Wang, Zhiru Zhang, and Zhuo Feng. 2019. Graphzoom: A multi-level spectral approach for accurate and scalable graph embedding. *arXiv preprint arXiv:1910.02370* (2019).
- [6] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. 2018. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *TheWeb Conference*.
- [7] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).
- [8] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*. 2331–2341.
- [9] Swapnil Gandhi and Anand Padmanabha Iyer. 2021. P3: Distributed deep graph learning at scale. In *USENIX*. 551–568.
- [10] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyröla, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677* (2017).
- [11] Saket Gururkar, Nikil Pancha, Andrew Zhai, Eric Kim, Samson Hu, Srinivasan Parthasarathy, Charles Rosenberg, and Jure Leskovec. 2022. MultiBiSage: A Web-Scale Recommendation System Using Multiple Bipartite Graphs at Pinterest. *arXiv preprint arXiv:2205.10666* (2022).
- [12] Yuntian He, Saket Gururkar, Pouya Kousha, Hari Subramoni, Dhableswar K Panda, and Srinivasan Parthasarathy. 2021. DistMILE: A Distributed Multi-Level Framework for Scalable Graph Embedding. In *2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC)*. IEEE, 282–291.
- [13] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. *arXiv preprint arXiv:2005.00687* (2020).
- [14] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems* 33 (2020), 22118–22133.
- [15] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *TheWeb Conference*. 2704–2710.
- [16] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. *ACL* (2015).
- [17] Zhuoren Jiang et al. 2020. Task-oriented genetic activation for large-scale complex heterogeneous graph embedding. In *TheWeb Conference*. 1581–1591.
- [18] George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SDM* 20, 1 (1998), 359–392.
- [19] Jiongqian Liang, Saket Gururkar, and Srinivasan Parthasarathy. 2018. Mile: A multi-level framework for scalable graph embedding. *arXiv preprint arXiv:1802.09612* (2018).
- [20] Jiongqian Liang, Peter Jacobs, Jiankai Sun, and Srinivasan Parthasarathy. 2018. Semi-supervised embedding in attributed networks with outliers. In *SDM*.
- [21] Ilya Loshchilov and Frank Hutter. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983* (2016).
- [22] Pinterest. 2022. Your audience is here. <https://business.pinterest.com/en/audience/>.
- [23] Aleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).
- [24] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1165–1174.
- [25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.
- [26] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The world wide web conference*. 2022–2032.
- [27] Ji Yang et al. 2020. Mixed Negative Sampling for Learning Two-tower Neural Networks in Recommendations. In *TheWeb Conference*. 441–447.
- [28] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*. PMLR, 40–48.
- [29] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 974–983.
- [30] Lingfan Yu, Jiajun Shen, Jinyang Li, and Adam Lerer. 2020. Scalable graph neural networks for heterogeneous graphs. *arXiv preprint arXiv:2011.09679* (2020).
- [31] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *KDD*. 793–803.
- [32] Chuxu Zhang, Ananthram Swami, and Nitesh V Chawla. 2019. Shne: Representation learning for semantic-associated heterogeneous networks. In *WSDM*.
- [33] Fanjin Zhang, Xiao Liu, Jie Tang, Yuxiao Dong, Peiran Yao, Jie Zhang, Xiaotao Gu, Yan Wang, Bin Shao, Rui Li, et al. 2019. Oag: Toward linking large-scale heterogeneous entity graphs. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2585–2595.
- [34] Da Zheng et al. 2020. Distdgl: distributed graph neural network training for billion-scale graphs. In *2020 IEEE/ACM 10th Workshop on Irregular Applications: Architectures and Algorithms (IA3)*. IEEE, 36–44.
- [35] Zhihua Zhu, Xinxin Fan, Xiaokai Chu, and Jingping Bi. 2020. Hgcn: A heterogeneous graph convolutional network-based deep learning model toward collective classification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1161–1171.
- [36] Jinfeng Zhuang and Yu Liu. 2019. PinText: A multitask text embedding system in pinterest. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2653–2661.