Heterogeneous Graph Transformer

Ziniu Hu*
University of California, Los Angeles
bull@cs.ucla.edu

Yuxiao Dong
Microsoft Research, Redmond
yuxdong@microsoft.com

Kuansan Wang
Microsoft Research, Redmond
kuansanw@microsoft.com

Yizhou Sun
University of California, Los Angeles
yzsun@cs.ucla.edu

ABSTRACT

Recent years have witnessed the emerging success of graph neural networks (GNNs) for modeling structured data. However, most GNNs are designed for homogeneous graphs, in which all nodes and edges belong to the same types, making it infeasible to represent heterogeneous structures. In this paper, we present the Heterogeneous Graph Transformer (HGT) architecture for modeling Web-scale heterogeneous graphs. To model heterogeneity, we design node- and edge-type dependent parameters to characterize the heterogeneous attention over each edge, empowering HGT to maintain dedicated representations for different types of nodes and edges. To handle Web-scale graph data, we design the heterogeneous mini-batch graph sampling algorithm—HGSampling—for efficient and scalable training. Extensive experiments on the Open Academic Graph of 179 million nodes and 2 billion edges show that the proposed HGT model consistently outperforms all the state-of-the-art GNN baselines by 9%–21% on various downstream tasks. The dataset and source code of HGT are publicly available at https://github.com/acbull/pyHGT.

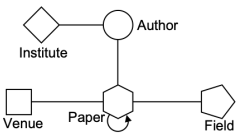
KEYWORDS

Graph Neural Networks; Heterogeneous Information Networks; Representation Learning; Graph Embedding; Graph Attention

ACM Reference Format:
Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3366423.3380027

1 INTRODUCTION

Heterogeneous graphs have been commonly used for abstracting and modeling complex systems, in which objects of different types interact with each other in various ways. Some prevalent instances of such systems include academic graphs, Facebook entity graph, LinkedIn economic graph, and broadly the Internet of Things network [13]. For example, the Open Academic Graph (OAG) [23] contains five types of nodes: papers, authors, institutions, venues (journal, conference, or preprint), and fields, as well as different types of relationships between them.



(a) The schema of heterogeneous academic networks

Author	*is_(first/last/other)_author_of*	Paper
Author	*is_affiliated_with*	Institute
Paper	*is_published_(conf/journal)_at*	Venue
Paper	*has_(L_1–L_5)_field_of*	Field
Paper	*has_citation_to*	Paper

(b) The meta relations of heterogeneous academic networks

Figure 1: The schema and meta relations of Open Academic Graph (OAG).

Over the past decade, a significant line of research has been explored for mining heterogeneous graphs. One of the classical paradigms is to define and use meta paths to model heterogeneous structures, such as PathSim [14] and metapath2vec [2]. Recently, in view of graph neural networks' (GNNs) success [4, 6, 17], there are several attempts to adopt GNNs to learn with heterogeneous networks [11, 18, 21, 22]. However, these works face several issues: First, most of them involve the design of meta paths or variants for each type of heterogeneous graphs, requiring specific domain knowledge; Second, they either simply assume that different types of nodes/edges share the same feature and representation space or keep distinct non-sharing weights for either node type or edge type alone, making them insufficient to capture heterogeneous graphs' properties; Finally, their intrinsic design and implementation make them incapable of modeling Web-scale heterogeneous graphs.

In light of these limitations and challenges, we propose to study heterogeneous neural networks with the goal of maintaining node- and edge-type dependent representations, avoiding customized meta paths, and being scalable to Web-scale heterogeneous graphs. In this work, we present the Heterogeneous Graph Transformer (HGT) architecture to deal with all these challenges.

To handle graph heterogeneity, we introduce the node- and edge-type dependent attention mechanism. Instead of parameterizing each type of edges, the heterogeneous mutual attention in HGT is defined by breaking down each edge $e = (s, t)$ based on its meta relation triplet, i.e., $\langle$ node type of s, edge type of e between s & t, node type of $t\rangle$. Figure 1 illustrates the meta relations of heterogeneous academic graphs. In specific, we use these meta relations to parameterize the weight matrices for calculating attention over each edge. As a result, nodes and edges of different types are allowed to maintain their specific representation spaces. Meanwhile,

*This work was done when Ziniu was an intern at Microsoft Research.

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380027

connected nodes in different types can still interact, pass, and aggregate messages without being restricted by their distribution gaps. Due to the nature of its architecture, HGT can incorporate information from high-order neighbors of different types through message passing across layers, which can be regarded as "soft" meta paths. That said, even if HGT take only its one-hop edges as input without manually designing meta paths, the proposed attention mechanism can automatically and implicitly learn and extract "meta paths" that are important for different downstream tasks.

To model Web-scale heterogeneous graphs, we design the first heterogeneous sub-graph sampling algorithm—HGSampling—for mini-batch GNN training. Its main idea is to sample heterogeneous sub-graphs in which different types of nodes are with similar proportion, since the direct usage of existing (homogeneous) GNN sampling methods, such as GraphSage [4], FastGCN [1] and LADIES [24], results in highly imbalanced ones regarding to both node and edge types. In addition, it is also designed to keep the sampled sub-graphs dense for minimizing the loss of information. With HGSampling, all the GNN models, including our proposed HGT, can train and infer on arbitrary-size heterogeneous graphs.

We demonstrate the effectiveness and efficiency of the proposed Heterogeneous Graph Transformer on the Web-scale Open Academic Graph comprised of 179 million nodes and 2 billion edges, making this the largest-scale representation learning yet performed on heterogeneous graphs. Experimental results suggest that HGT can significantly improve various downstream tasks over state-of-the-art GNN baselines by 9%–21%. We further conduct case studies to show the proposed method can indeed automatically capture the importance of implicit meta paths for different tasks.

2 HETEROGENEOUS GRAPH TRANSFORMER

In this section, we present the Heterogeneous Graph Transformer (HGT). Its idea is to use the **meta relations** of heterogeneous graphs to parameterize weight matrices for the heterogeneous mutual attention, message passing, and propagation steps.

2.1 Heterogeneous Graphs

Heterogeneous graphs [13] (a.k.a., heterogeneous information networks) are an important abstraction for modeling relational data and many real-world complex systems. Formally, a heterogeneous graph is defined as a directed graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$ where each node $v \in \mathcal{V}$ and each edge $e \in \mathcal{E}$ are associated with their type mapping functions $\tau(v) : V \rightarrow \mathcal{A}$ and $\phi(e) : E \rightarrow \mathcal{R}$, respectively.

Meta Relation. For an edge $e = (s, t)$ linked from source node s to target node t, its meta relation is denoted as $\langle \tau(s), \phi(e), \tau(t) \rangle$. Naturally, $\phi(e)^{-1}$ represents the inverse of $\phi(e)$. The classical meta path paradigm [13–15] is defined as a sequence of such meta relations.

Notice that, to better model real-world heterogeneous networks, we assume that there may exist multiple types of relations between two nodes. For example, in OAG there could be different types of relations between the *author* and *paper* nodes by considering the authorship order, e.g., "the first author of" and "the last author of".

2.2 Overall HGT Architecture

Figure 2 shows the overall architecture of Heterogeneous Graph Transformer. Given a sampled heterogeneous sub-graph (Cf. Section 3), HGT extracts all linked node pairs, where target node t is linked by source node s via edge e. The goal of HGT is to aggregate information from s to get a contextualized representation for target node t. Such process can be decomposed into three components: *Heterogeneous Mutual Attention*, *Heterogeneous Message Passing* and *Target-Specific Aggregation*.

We denote the output of the (l)-th HGT layer as $H^{(l)}$, which is also the input of the $(l+1)$-th layer. By stacking L layers, we can get the node representations of the whole graph $H^{(L)}$, which can be used for end-to-end training or fed into downstream tasks.

2.3 Heterogeneous Mutual Attention

The first step is to calculate the mutual attention between source node s and target node t. We first give a brief introduction to the general attention-based GNNs as follows:

$$H^l[t] \leftarrow \underset{\forall s \in N(t), \forall e \in E(s,t)}{\textbf{Aggregate}} \Big(\textbf{Attention}(s,t) \cdot \textbf{Message}(s) \Big) \quad (1)$$

where there are three basic operators: **Attention**, which estimates the importance of each source node; **Message**, which extracts the message by using only the source node s; and **Aggregate**, which aggregates the neighborhood message by the attention weight.

For example, the Graph Attention Network (GAT) [17] adopts an additive mechanism as **Attention**, uses the same weight for calculating **Message**, and leverages the simple average followed by a nonlinear activation for the **Aggregate** step. Formally, GAT has

$$\textbf{Attention}_{GAT}(s,t) = \underset{\forall s \in N(t)}{\text{Softmax}} \Big(\vec{a} \Big(WH^{l-1}[t] \parallel WH^{l-1}[s] \Big) \Big)$$
$$\textbf{Message}_{GAT}(s) = WH^{l-1}[s]$$
$$\textbf{Aggregate}_{GAT}(\cdot) = \sigma \Big(\text{Mean}(\cdot) \Big)$$

Though GAT is effective to give high attention values to important nodes, it assumes that s and t have the same feature distributions by using one weight matrix W. Such an assumption, as we've discussed in Section 1, is usually incorrect for heterogeneous graphs, where each type of nodes can have its own feature distribution.

In view of this limitation, we design the **Heterogeneous Mutual Attention** mechanism. Given a target node t, and all its neighbors $s \in N(t)$, which might belong to different distributions, we want to calculate their mutual attention grounded by their **meta relations**, i.e., the $\langle \tau(s), \phi(e), \tau(t) \rangle$ triplets.

Inspired by the architecture design of Transformer [16], we map target node t into a Query vector, and source node s into a Key vector, and calculate their dot product as attention. The key difference is that the vanilla Transformer uses a single set of projections for all words, while in our case each meta relation should have a distinct set of projection weights. To maximize parameter sharing while still maintaining the specific characteristics of different relations, we propose to parameterize the weight matrices of the interaction operators into a source node projection, an edge projection, and a target node projection. Specifically, we calculate the h-head

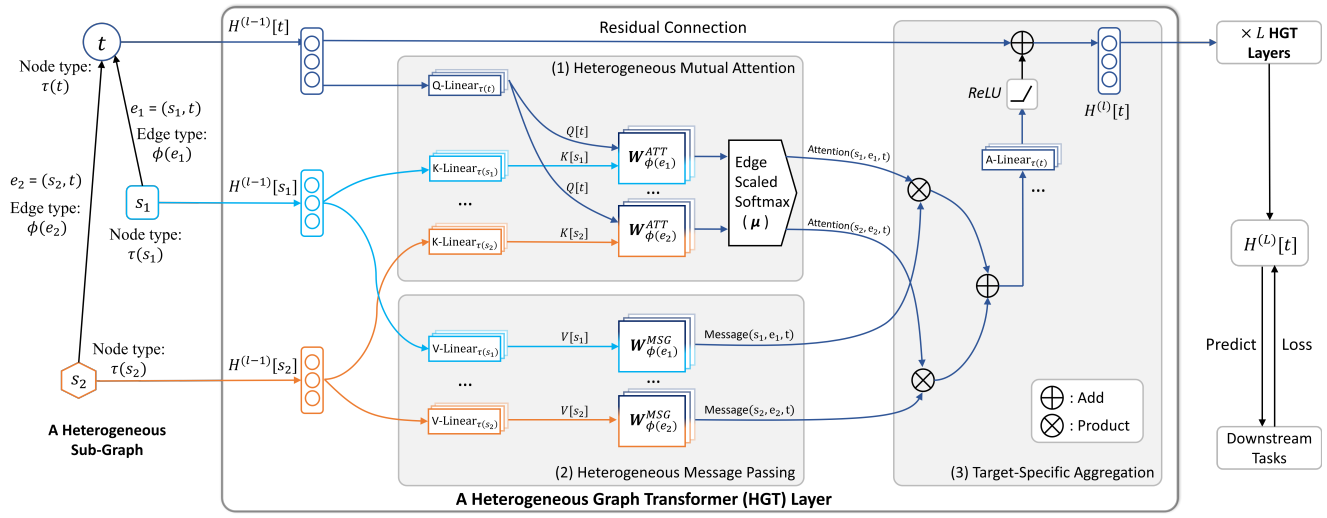


Figure 2: The Overall Architecture of Heterogeneous Graph Transformer. Given a sampled heterogeneous sub-graph with t as the target node, s_1 & s_2 as source nodes, the HGT model takes its edges $e_1 = (s_1, t)$ & $e_2 = (s_2, t)$ and their corresponding meta relations $< \tau(s_1), \phi(e_1), \tau(t) >$ & $< \tau(s_2), \phi(e_2), \tau(t) >$ as input to learn a contextualized representation $H^{(L)}$ for each node, which can be used for downstream tasks. Color decodes the node type. HGT includes three components: (1) meta relation-aware heterogeneous mutual attention, (2) heterogeneous message passing from source nodes, and (3) target-specific heterogeneous message aggregation.

attention for each edge $e = (s, t)$ (See Figure 2 (1)) by:

$$\textbf{Attention}_{HGT}(s, e, t) = \underset{\forall s \in N(t)}{\text{Softmax}}\Big(\Big\|_{i \in [1,h]} ATT\text{-}head^i(s, e, t) \Big) \quad (2)$$

$$ATT\text{-}head^i(s, e, t) = \Big(K^i(s)\, W^{ATT}_{\phi(e)}\, Q^i(t)^T \Big) \cdot \frac{\mu_{\langle \tau(s), \phi(e), \tau(t) \rangle}}{\sqrt{d}}$$

$$K^i(s) = \text{K-Linear}^i_{\tau(s)}\Big(H^{(l-1)}[s] \Big)$$

$$Q^i(t) = \text{Q-Linear}^i_{\tau(t)}\Big(H^{(l-1)}[t] \Big)$$

First, for the i-th attention head $ATT\text{-}head^i(s, e, t)$, we project the $\tau(s)$-type source node s into the i-th *Key* vector $K^i(s)$ with a linear projection $\text{K-Linear}^i_{\tau(s)} : \mathbb{R}^d \rightarrow \mathbb{R}^{\frac{d}{h}}$, where h is the number of attention heads and $\frac{d}{h}$ is the vector dimension per head. Note that $\text{K-Linear}^i_{\tau(s)}$ is indexed by the source node s's type $\tau(s)$, meaning that each type of nodes has a unique linear projection to maximally model the distribution differences. Similarly, we also project the target node t with a linear projection $\text{Q-Linear}^i_{\tau(t)}$ into the i−th Query vector.

Next, we need to calculate the similarity between the Query vector $Q^i(t)$ and Key vector $K^i(s)$. One unique characteristic of heterogeneous graphs is that there may exist different edge types (relations) between a node type pair, e.g., $\tau(s)$ and $\tau(t)$. Therefore, unlike the vanilla Transformer that directly calculates the dot product between the Query and Key vectors, we keep a distinct edge-based matrix $W^{ATT}_{\phi(e)} \in \mathbb{R}^{\frac{d}{h} \times \frac{d}{h}}$ for each edge type $\phi(e)$. In doing so, the model can capture different semantic relations even between the same node type pairs. Moreover, since not all the relationships contribute equally to the target nodes, we add a prior tensor $\mu \in \mathbb{R}^{|\mathcal{A}| \times |\mathcal{R}| \times |\mathcal{A}|}$ to denote the general significance of each meta relation triplet, serving as an adaptive scaling to the attention.

Finally, we concatenate h attention heads together to get the attention vector for each node pair. Then, for each target node t, we gather all attention vectors from its neighbors $N(t)$ and conduct softmax, making it fulfill $\sum_{\forall s \in N(t)} \textbf{Attention}_{HGT}(s, e, t) = \mathbf{1}_{h \times 1}$.

2.4 Heterogeneous Message Passing

Parallel to the calculation of mutual attention, we pass information from source nodes to target nodes (See Figure 2 (2)). Similar to the attention process, we would like to incorporate the meta relations of edges into the message passing process to alleviate the distribution differences of nodes and edges of different types. For a pair of nodes $e = (s, t)$, we calculate its multi-head **Message** by:

$$\textbf{Message}_{HGT}(s, e, t) = \Big\|_{i \in [1,h]} MSG\text{-}head^i(s, e, t) \quad (3)$$

$$MSG\text{-}head^i(s, e, t) = \text{M-Linear}^i_{\tau(s)}\Big(H^{(l-1)}[s] \Big)\, W^{MSG}_{\phi(e)}$$

To get the i-th message head $MSG\text{-}head^i(s, e, t)$, we first project the $\tau(s)$-type source node s into the i-th message vector with a linear projection $\text{M-Linear}^i_{\tau(s)} : \mathbb{R}^d \rightarrow \mathbb{R}^{\frac{d}{h}}$. It is then followed by a matrix $W^{MSG}_{\phi(e)} \in \mathbb{R}^{\frac{d}{h} \times \frac{d}{h}}$ for incorporating the edge dependency. The final step is to concat all h message heads to get the $\textbf{Message}_{HGT}(s, e, t)$ for each node pair.

2.5 Target-Specific Aggregation

With the heterogeneous multi-head attention and message calculated, we need to aggregate them from the source nodes to the target node (See Figure 2 (3)). Note that the softmax procedure in Eq. 2 has made the sum of each target node t's attention vectors

to one, we can thus simply use the attention vector as the weight to average the corresponding messages from the source nodes and get the updated vector $\widetilde{H}^{(l)}[t]$ as:

$$\widetilde{H}^{(l)}[t] = \bigoplus_{\forall s \in N(t)} \Big(\textbf{Attention}_{HGT}(s, e, t) \cdot \textbf{Message}_{HGT}(s, e, t) \Big).$$

This aggregates information to the target node t from all its neighbors (source nodes) of different feature distributions.

The final step is to map target node t's vector back to its type-specific distribution, indexed by its node type $\tau(t)$. To do so, we apply a linear projection A-Linear$_{\tau(t)}$ to the updated vector $\widetilde{H}^{(l)}[t]$, followed by a non-linear activation and residual connection [5] as:

$$H^{(l)}[t] = \sigma\Big(\text{A-Linear}_{\tau(t)}\widetilde{H}^{(l)}[t]\Big) + H^{(l-1)}[t]. \tag{4}$$

In this way, we get the l-th HGT layer's output $H^{(l)}[t]$ for the target node t. Due to the "small-world" property of real-world graphs, stacking the HGT blocks for L layers (L being a small value) can enable each node reaching a large proportion of nodes—with different types and relations—in the full graph. That is, HGT generates a highly contextualized representation $H^{(L)}$ for each node, which can be fed into any models to conduct downstream heterogeneous network tasks, such as node classification and link prediction.

Through the whole model architecture, we highly rely on using the **meta relation**—$\langle \tau(s), \phi(e), \tau(t) \rangle$—to parameterize the weight matrices separately. This can be interpreted as a trade-off between the model capacity and efficiency. Compared with the vanilla Transformer, our model distinguishes the operators for different relations and thus is more capable to handle the distribution differences in heterogeneous graphs. Compared with existing models that keep a distinct matrix for each meta relation as a whole, HGT's triplet parameterization can better leverage the heterogeneous graph schema to achieve parameter sharing. On one hand, relations with few occurrences can benefit from such parameter sharing for fast adaptation and generalization. On the other hand, different relationships' operators can still maintain their specific characteristics by using a much smaller parameter set.

3 HGSampling FOR WEB-SCALE TRAINING

In this section, we present an efficient Heterogeneous Mini-Batch Graph Sampling algorithm—HGSampling—to enable both HGT and traditional GNNs to handle Web-scale heterogeneous graphs. HGSampling is able to 1) keep a similar number of nodes and edges for each type and 2) keep the sampled sub-graph dense to minimize the information loss and reduce the sample variance.

Algorithm 1 outlines the HGSampling algorithm. Its basic idea is to keep a separate node budget $B[\tau]$ for each node type τ and to sample an equal number of nodes per type with an importance sampling strategy to reduce variance. Given node t already sampled, we add all its direct neighbors into the corresponding budget with Algorithm 2, and add t's normalized degree to these neighbors in line 5, which will then be used to calculate the sampling probability. Such normalization is equivalent to accumulate the random walk probability of each sampled node to its neighborhood, avoiding the sampling being dominated by high-degree nodes. Intuitively, the higher such value is, the more a candidate node is correlated with the currently sampled nodes, and thus should be given a higher probability to be sampled.

Algorithm 1 Heterogeneous Mini-Batch Graph Sampling

Require: Adjacency matrix A for each $\langle \tau(s), \phi(e), \tau(t) \rangle$ relation pair; Output node Set OS; Sample number n per node type; Sample depth L.
Ensure: Sampled node set NS; Sampled adjacency matrix $\hat{A}$.

1: $NS \leftarrow OS$ // Initialize sampled node set as output node set.
2: Initialize an empty Budget B storing nodes for each node type with normalized degree.
3: **for** $t \in NS$ **do**
4: Add-In-Budget(B, t, A, NS) // Add neighbors of t to B.
5: **end for**
6: **for** $l \leftarrow 1$ to L **do**
7: **for** source node type $\tau \in B$ **do**
8: **for** source node $s \in B[\tau]$ **do**
9: $prob^{(l-1)}[\tau][s] \leftarrow \frac{B[\tau][s]^2}{\|B[\tau]\|_2^2}$ // Calculate sampling probability for each source node s of node type τ.
10: **end for**
11: Sample n nodes $\{t_i\}_{i=1}^n$ from $B[\tau]$ using $prob^{(l-1)}[\tau]$.
12: **for** $t \in \{t_i\}_{i=1}^n$ **do**
13: $OS[\tau].add(t)$ // Add node t into Output node set.
14: Add-In-Budget(B, t, A, NS) // Add neighbors of t to B.
15: $B[\tau].pop(t)$ // Remove sampled node t from Budget.
16: **end for**
17: **end for**
18: **end for**
19: Reconstruct the sampled adjacency matrix $\hat{A}$ among the sampled nodes OS from A.
20: **return** OS and $\hat{A}$;

After the budget is updated, we then calculate the sampling probability in Algorithm 1 line 9, where we calculate the square of the cumulative normalized degree of each node s in each budget. As proved in [24], using such sampling probability can reduce the sampling variance. Then, we sample n nodes in type τ by using the calculated probability, add them into the output node set, update its neighborhood to the budget, and remove it out of the budget in lines 12–15. Repeating such procedure for L times, we get a sampled sub-graph with L depth from the initial nodes. Finally, we reconstruct the adjacency matrix among the sampled nodes. By using the above algorithm, the sampled sub-graph contains a similar number of nodes per type (based on the separate node budget), and is sufficiently dense to reduce the sampling variance (based on the normalized degree and importance sampling), making it suitable for training GNNs on Web-scale heterogeneous graphs.

4 EVALUATION

In this section, we evaluate the proposed Heterogeneous Graph Transformer on the Open Academic Graph (OAG) [23]—the largest publicly available heterogeneous academic dataset. We conduct the Paper-Field prediction, Paper-Venue prediction, and Author Disambiguation tasks. We also take case studies to demonstrate

Dataset	#nodes	#edges	#papers	#authors	#fields	#venues	#institutes	#P-A	#P-F	#P-V	#A-I	#P-P
OAG	178,663,927	2,236,196,802	89,606,257	88,364,081	615,228	53,073	25,288	300,853,688	657,049,405	89,606,258	167,449,933	1,021,237,518

Table 1: Open Academic Graph (OAG) Statistics.

Algorithm 2 Add-In-Budget

Require: Budget B storing nodes for each type with normalized degree; Added node t; Adjacency matrix A for each $\langle \tau(s), \phi(e), \tau(t) \rangle$ relation pair; Sampled node set NS.
Ensure: Updated Budget B.

```
for each possible source node type τ and edge type φ do
    D̂_t ← 1 / len(A_⟨τ,φ,τ(t)⟩[t]) // get normalized degree of added node t regarding to ⟨τ, φ, τ(t)⟩.
    for source node s in A_⟨τ,φ,τ(t)⟩[t] do
        if s has not been sampled (s ∉ NS) then
            B[τ][s] ← B[τ][s] + D̂_t // Add candidate node s to budget B with target node t's normalized degree.
        end if
    end for
end for
return Updated Budget B
```

how HGT can automatically learn and extract meta paths that are important for downstream tasks*.

4.1 Web-scale Datasets

To examine Heterogeneous Graph Transformer and its real-world applications, we use the Open Academic Graph (OAG) [12, 23] as our experimental basis. OAG consists of more than 178 million nodes and 2.236 billion edges, making them at least two–three magnitudes larger than the other datasets that are commonly used in existing heterogeneous GNN and heterogeneous graph mining studies. Besides, it is far more distinguishable than previously wide-adopted small citation graphs used in GNN studies, such as Cora, Citeseer and Pubmed [6, 17], which only contain thousands of nodes. The graph statistics are listed in Table 1, in which P–A, P–F, P–V, A–I, and P–P denote the edges between paper and author, paper and field, paper and venue, author and institute, and the citation links between two papers. In addition, the 'Field' nodes in OAG are categorized into six levels from L_0 to L_5, which are organized with a hierarchical tree. Therefore, we differentiate the 'Paper–Field' edges corresponding to the field level.

4.2 Experimental Setup

Tasks and Evaluation. We evaluate the HGT model on four different real-world downstream tasks: the prediction of Paper–Field (L_1), Paper–Field (L_2), and Paper–Venue, and Author Disambiguation. The goal of the first three node classification tasks is to predict the correct L_1 and L_2 fields that each paper belongs to or the venue it is published at, respectively. We use different GNNs to get the contextual node representation of the paper and use a softmax output layer to get its classification label. For author disambiguation, we select all the authors with the same name and their associated papers. The task is to conduct link prediction between these papers and candidate authors. After getting the paper and author node representations from GNNs, we use a Neural Tensor Network to get the probability of each author-paper pair to be linked.

For all tasks, we use papers published before the year 2015 as the training set, papers between 2015 and 2016 for validation, and papers between 2016 and 2019 as testing. We choose NDCG and MRR, which are two widely adopted ranking metrics [7, 8], as the evaluation metrics. All models are trained for 5 times and, the mean and standard variance of test performance are reported.

Baselines. We compare HGT with several state-of-the-art GNNs, including both homogeneous—GCN [6] and GAT [17]—and heterogeneous GNNs—RGCN [11], HetGNN [22], and HAN [18]. To examine the effectiveness of the heterogeneous components in our model, we also propose the $\text{HGT}_{\text{noHeter}}$ model, which uses the same set of weights for all meta relations, as the ablation study. All baselines as well as our own model are implemented via the PyTorch Geometric (PyG) package [3].

We use our HGSampling algorithm proposed in Section 3 for all baseline GNNs to handle the large-scale OAG graph. To avoid data leakage, we remove out the links we aim to predict (e.g. the Paper-Field link as the label) from the sub-graph.

Input Features. As we don't assume the feature of each node type belongs to the same distribution, we are free to use the most appropriate features to represent each type of nodes. For each paper, we use a pre-trained XLNet [19, 20] to get the representation of each word in its title. We then average them weighted by each word's attention to get the title representation for each paper. The initial feature of each author is then simply an average of his/her published papers' representations. For the field, venue, and institute nodes, we use the metapath2vec model [2] to train their node embeddings by reflecting the heterogeneous network structures.

The homogeneous GNN baselines assume the node features belong to the same distribution, while our feature extraction doesn't fulfill this assumption. To make a fair comparison, we add an adaptation layer between the input features and all used GNNs. This module simply conducts different linear projections for nodes of different types. Such a procedure can be regarded to map heterogeneous data into the same distribution, which is also adopted in literature [18, 22].

Implementation Details. We use 256 as the hidden dimension throughout the neural networks for all baselines. For all multi-head attention-based methods, we set the head number as 8. All GNNs keep 3 layers so that the receptive fields of each network are exactly the same. All baselines are optimized via the AdamW optimizer [10] with the Cosine Annealing Learning Rate Scheduler [9]. For each model, we train it for 200 epochs and select the one with the lowest

*The dataset and code are publicly available at https://github.com/acbull/pyHGT.

GNN Models		GCN [6]	RGCN [11]	GAT [17]	HetGNN [22]	HAN [18]	$HGT_{noHeter}$	HGT
#Parameter / Batch Time		1.69M / 0.46s	8.80M / 1.24s	1.69M / 0.97s	8.41M / 1.35s	9.45M / 2.27s	3.12M / 1.11s	7.44M / 1.48s
Paper–Field (L_1)	NDCG	.508±.141	.511±.128	.534±.103	.543±.084	.544±.096	.571±.089	**.595±.089**
	MRR	.556±.136	.565±.105	.610±.096	.616±.076	.622±.092	.649±.081	**.675±.082**
Paper–Field (L_2)	NDCG	.218±.074	.228±.046	.239±.049	.236±.062	.242±.051	.250±.045	**.258±.052**
	MRR	.222±.067	.232±.052	.248±.045	.250±.053	.258±.049	.262±.057	**.271±.064**
Paper–Venue	NDCG	.265±.066	.276±.051	.270±.057	.262±.071	.280±.062	.297±.058	**.306±.064**
	MRR	.258±.070	.236±.047	.260±.052	.246±.059	.278±.067	.293±.061	**.317±.048**
Author Disambiguation	NDCG	.612±.064	.619±.057	.645±.063	.649±.052	.660±.049	.668±.059	**.683±.066**
	MRR	.738±.042	.755±.048	.797±.044	.803±.058	.821±.056	.835±.043	**.847±.043**

Table 2: Experimental results of different methods on Open Academic Graph (OAG).

validation loss as the reported model. We use the default parameters used in GNN literature and donot tune hyper-parameters.

4.3 Results

We summarize the experimental results of the proposed model and baselines in Table 2. All experiments for the four tasks are evaluated in terms of NDCG and MRR. It shows that in terms of both metrics, the proposed HGT model significantly and consistently outperforms all baselines for all tasks. Take, for example, the Paper–Field (L_1) classification task, HGT achieves performance gains over baselines by 9–19% in terms of NDCG and 9–21% in terms of MRR (i.e., the performance difference divided by the baseline performance). When compared to HetGNN and HAN—the two dedicated heterogeneous GNN baselines, on average, the relative NDCG improvements of HGT for all four tasks are 8% and 6%, respectively. Moreover, HGT has fewer parameters and comparable batch time than all the heterogeneous graph neural network baselines, including RGCN, HetGNN, and HAN. This suggests that by modeling heterogeneous edges according to their meta relation schema, we are able to have better generalization with fewer resource consumption.

Ablation Study. The core component in HGT is the meta relation parameterization. To further analyze its effect, we conduct an ablation study. $HGT_{noHeter}$ only maintains a single set of parameters for all relations, which is equivalent to the vanilla Transformer applied on graphs. We can see that after removing this component, the NDCG performance drops 3.2%, demonstrating the importance of our meta relation parameterization.

Besides, we also try to implement a baseline that keeps a unique weight matrix for each relation. However, such a baseline contains too many parameters so that our experimental setting doesn't have enough GPU memory to optimize it. This also indicates that using the meta relation to parameterize weight matrices can achieve competitive performance with fewer resources.

4.4 Visualize Meta Relation Attention

To illustrate how the incorporated meta relation schema can benefit the heterogeneous message passing process, we pick the schema that has the largest attention value in each of the first two HGT layers and plot the meta relation attention hierarchy tree in Figure 3. For example, to calculate a paper's representation, ⟨Paper, *is_published_at*, Venue, $is\_published\_at^{-1}$, Paper⟩, ⟨Paper, $has\_L_2\_field\_of$, Field, $has\_L_5\_field\_of^{-1}$, Paper⟩, and ⟨Institute, $is\_affiliated\_with^{-1}$, Author, $is\_first\_author\_of$, Paper⟩ are the three most important meta relation sequences, which can be regarded as meta paths *PVP, PFP,* and *IAP*, respectively. Note that these meta paths and their importance are automatically learned from the data without manual design. Another example of calculating an author node's representation is shown on the right. Such visualization demonstrates that Heterogeneous Graph Transformer is capable of implicitly learning to construct important meta paths for specific downstream tasks, without manual customization.

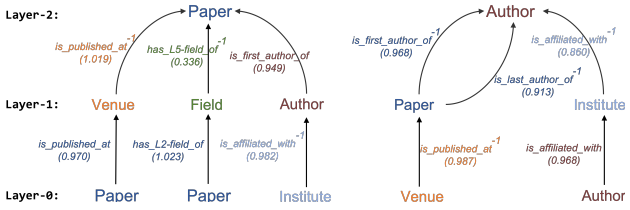


Figure 3: Hierarchy of the learned meta relation attention.

5 CONCLUSION

In this paper, we propose the Heterogeneous Graph Transformer (HGT) architecture for modeling Web-scale heterogeneous graphs. To model heterogeneity, we use the meta relation $\langle \tau(s), \phi(e), \tau(t) \rangle$ to decompose the interaction and transformation matrices, enabling HGT to have the similar modeling capacity with fewer resources. To conduct efficient and scalable training of HGT on Web-scale data, we design the heterogeneous Mini-Batch graph sampling algorithm—HGSampling. We conduct comprehensive experiments on the Open Academic Graph to show that the proposed HGT model can capture graph heterogeneity and outperform all the state-of-the-art GNN baselines on various downstream tasks.

Acknowledgements. We would like to thank Xiaodong Liu for helpful discussions. This work is partially supported by NSF III-1705169, NSF CAREER Award 1741634, NSF#1937599, Okawa Foundation Grant, and Amazon Research Award.

REFERENCES

[1] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *ICLR'18*.

[2] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *KDD '17*.

[3] Matthias Fey and Jan Eric Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. *ICLR 2019 Workshop: Representation Learning on Graphs and Manifolds* (2019).

[4] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS'17*.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR'16*.

[6] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR'17*.

[7] Hang Li. 2014. *Learning to Rank for Information Retrieval and Natural Language Processing, Second Edition*. Morgan & Claypool Publishers. https://doi.org/10.2200/S00607ED2V01Y201410HLT026

[8] Tie-Yan Liu. 2011. *Learning to Rank for Information Retrieval*. Springer. https://doi.org/10.1007/978-3-642-14267-3

[9] Ilya Loshchilov and Frank Hutter. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *ICLR'17*.

[10] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *ICLR'19*.

[11] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *ESWC'2018*.

[12] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Paul Hsu, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *WWW Companion 2015*.

[13] Yizhou Sun and Jiawei Han. 2012. *Mining Heterogeneous Information Networks: Principles and Methodologies*. Morgan & Claypool Publishers.

[14] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. In *VLDB '11*.

[15] Yizhou Sun, Brandon Norick, Jiawei Han, Xifeng Yan, Philip S. Yu, and Xiao Yu. 2012. Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. In *KDD'12*.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS'17*.

[17] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR'18*.

[18] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous Graph Attention Network. In *KDD'19*. 2022–2032.

[19] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Transformers: State-of-the-art Natural Language Processing. arXiv:cs.CL/1910.03771

[20] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NeurIPS'19*.

[21] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. 2019. Graph Transformer Networks. In *NeurIPS'19*.

[22] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous Graph Neural Network. In *WWW'19*.

[23] Fanjin Zhang, Xiao Liu, Jie Tang, Yuxiao Dong, Peiran Yao, Jie Zhang, Xiaotao Gu, Yan Wang, Bin Shao, Rui Li, and Kuansan Wang. 2019. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. In *KDD'19*.

[24] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. 2019. Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks. In *NeurIPS'19*.