Heterformer: Transformer-based Deep Node Representation Learning on Heterogeneous Text-Rich Networks

Bowen Jin
University of Illinois at Urbana-Champaign
bowenj4@illinois.edu

Yu Zhang
University of Illinois at Urbana-Champaign
yuz9@illinois.edu

Qi Zhu
University of Illinois at Urbana-Champaign
qiz3@illinois.edu

Jiawei Han
University of Illinois at Urbana-Champaign
hanj@illinois.edu

ABSTRACT

Representation learning on networks aims to derive a meaningful vector representation for each node, thereby facilitating downstream tasks such as link prediction, node classification, and node clustering. In heterogeneous text-rich networks, this task is more challenging due to (1) *presence or absence of text*: Some nodes are associated with rich textual information, while others are not; (2) *diversity of types*: Nodes and edges of multiple types form a heterogeneous network structure. As pretrained language models (PLMs) have demonstrated their effectiveness in obtaining widely generalizable text representations, a substantial amount of effort has been made to incorporate PLMs into representation learning on text-rich networks. However, few of them can jointly consider heterogeneous structure (network) information as well as rich textual semantic information of each node effectively. In this paper, we propose Heterformer, a ***Heter****ogeneous Network-Empowered Trans****former*** that performs contextualized text encoding and heterogeneous structure encoding in a unified model. Specifically, we inject heterogeneous structure information into each Transformer layer when encoding node texts. Meanwhile, Heterformer is capable of characterizing node/edge type heterogeneity and encoding nodes with or without texts. We conduct comprehensive experiments on three tasks (*i.e.*, link prediction, node classification, and node clustering) on three large-scale datasets from different domains, where Heterformer outperforms competitive baselines significantly and consistently. The code can be found at https://github.com/PeterGriffinJin/Heterformer.

CCS CONCEPTS

• **Computing methodologies** → **Learning latent representations**; • **Information systems** → **Data mining**.

KEYWORDS

Text-Rich Network, Pretrained Language Model, Transformer.

ACM Reference Format:
Bowen Jin, Yu Zhang, Qi Zhu, and Jiawei Han. 2023. Heterformer: Transformer-based Deep Node Representation Learning on Heterogeneous Text-Rich Networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599376

KDD '23, August 6–10, 2023, Long Beach, CA, USA.

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599376

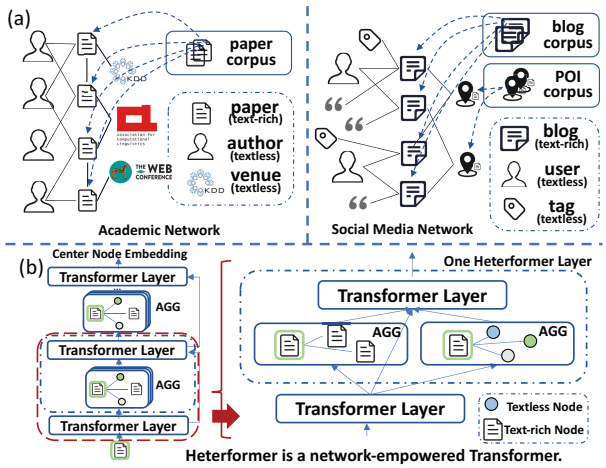


Figure 1: (a) Examples of heterogeneous text-rich networks: an academic network and a social media network. (b) An illustration of our heterogeneous network-empowered Transformer, Heterformer. One Heterformer layer is zoomed out. AGG denotes neighbor aggregation on the network.

1 INTRODUCTION

Heterogeneous text-rich networks are ubiquitously utilized to model real-world data such as academic networks [41], product networks [7], and social media [9]. Such networks often have two characteristics: (1) *Text-rich*: **some** types of nodes are associated with textual information. For instance, papers in academic networks [41] have their titles and abstracts; tweets in social media networks [9] have their tweet contents. (2) *Heterogeneous*: nodes and edges in the network are multi-typed. For example, academic networks [41] have paper, author, and venue nodes; product networks [7] have edges between users and products reflecting "purchase" and "view" relations. In such text-rich networks (Figure 1(a)), to obtain satisfying node representations which can be generalized to various tasks such as link prediction [40], node classification [46], and recommendation [53], the model needs to consider both **text semantics** and **heterogeneous** structure (network) information.

To capture the rich text semantic signals, Transformer [43] is a powerful architecture featured by its fully connected attention mechanism. Taking Transformer as the backbone, pretrained language models (PLMs) [4, 6, 25] learned from web-scale corpora can obtain contextualized semantic representations of words and documents. These representations are demonstrated to be of high quality in various text mining tasks [3, 24, 38]. To introduce the advanced capability of PLMs into node representation learning on text-rich networks, existing works mainly adopt a cascaded architecture [18, 23, 54, 59], where the textual information of each node

is first encoded via PLMs and then aggregated via graph encoders [12, 22, 44]. In such cases, the link connecting two nodes is not utilized when generating their text representations. In fact, linked nodes can benefit each other regarding text semantics understanding. For example, the term "Transformer" in a paper cited by many machine learning papers should refer to a deep learning model rather than an electrical engineering component. GraphFormers [50] further introduce a nested Transformer architecture for deep information integration between text encoding modules and network encoding modules. Patton [16] proposes two strategies to pretrain GraphFormers on text-rich networks. Yet, they adopt a strict homogeneous network assumption (*i.e.*, all the nodes are associated with semantically rich text and are of the same type), which is hard to be satisfied in practice.

As a matter of fact, real-world text-rich networks are usually heterogeneous with the heterogeneity coming from two sources: the presence or absence of text and the diversity of types.

- **Presence or Absence of Text.** Not every node in text-rich networks exhibits rich textual information. Instead, some nodes are not guaranteed to contain textual information (*e.g.*, many users are not associated with text information in social media networks). Based on the presence or absence of text, nodes can be categorized into **text-rich** nodes (associated with semantically rich text, *e.g.*, tweets and papers) and **textless** nodes (without semantically rich text, *e.g.*, users and authors). Text-rich nodes can intuitively contribute to representation learning with the rich texts, while textless nodes can also be strong semantic indicators in the network. For example, given a paper node about "Byzantine" which is linked to an author node (textless) with many paper neighbors related to "distributed system", we can infer that "Byzantine" here refers to a computer network term rather than an empire in history. Since both text-rich nodes and textless nodes should be considered, how to leverage both kinds of nodes in representation learning with PLMs is an open question that needs to be answered.
- **Diversity of Types.** As previously stated, a large number of real-world networks contain nodes and edges of different types. For example, there are at least three types of nodes ("paper", "author", and "venue" nodes) in academic networks [41]; there are at least four types of edges ("click", "view", "cart", and "purchase" edges) between users and products in e-commerce networks [7]. Different types of nodes/edges have different traits and their features may fall in different feature spaces. For instance, the feature of a user may contain gender, age, and nationality while the feature of an item may contain price and quality. How to handle such complex structural information while preserving the diverse type information simultaneously with PLMs is a crucial issue that needs to be solved.

Present Work. To this end, we propose a network-empowered Transformer (Figure 1(b)), *i.e.*, Heterformer, for node representation learning on heterogeneous text-rich networks, while capturing the two sources of heterogeneity mentioned above. Specifically: (1) In *the whole model*, we introduce virtual neighbor tokens inside each Transformer layer (initialized by the corresponding layer in a PLM, *e.g.*, BERT [6]) for text encoding, to fuse representations of each node's text-rich neighbors, textless neighbors, and its own content via the fully connected attention mechanism. The virtual neighbor token hidden states are attention-based aggregations of neighbor node embeddings. (2) To deal with the *presence or absence of text*, two virtual neighbor tokens are utilized to capture the semantic signals from text-rich neighbors and textless neighbors, respectively. Furthermore, we propose an embedding warm-up stage for textless nodes to obtain better initial embeddings before the whole model training. (3) To capture the *diversity of types*, we use type-specific transformation matrices to project different types of nodes into the same latent space. When calculating virtual neighbor token hidden states, the aggregation module collects information from its neighbors by characterizing edge types in the attention mechanism. The overall model is optimized via an unsupervised link prediction objective [12, 32].

The main contributions of our paper are summarized as follows:

- We formalize the problem of node representation learning on heterogeneous text-rich networks, which involves joint encoding of heterogeneous network structures and textual semantics.
- We point out heterogeneity from two sources and propose a heterogeneous network-empowered Transformer architecture called Heterformer, which deeply couples text encoding and heterogeneous structure (network) encoding.
- We conduct comprehensive experiments on three public text-rich networks from different domains, where Heterformer outperforms competitive baseline models (including GNN-cascaded Transformers and nested Transformers) significantly and consistently on various tasks, including link prediction, node classification, and node clustering.

2 PRELIMINARIES

2.1 Heterogeneous Text-rich Networks

Definition 2.1. Heterogeneous Networks [39]. A heterogeneous network is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$, where $\mathcal{V}$, $\mathcal{E}$, $\mathcal{A}$, $\mathcal{R}$ represent the sets of nodes, edges, node types, and edge types, respectively. $|\mathcal{A}| + |\mathcal{R}| > 2$. A heterogeneous network is also associated with a node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and an edge type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$.

Definition 2.2. Text-Rich Nodes and Textless Nodes. In a heterogeneous network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$, $v \in \mathcal{V}$ is **text-rich** if it is associated with semantically rich text information $doc \in \mathcal{D}$. $\mathcal{D}$ is the document set. Otherwise, it is **textless**. We assume that nodes of the same type are either all text-rich or all textless.

Definition 2.3. Heterogeneous Text-Rich Networks [37, 54]. A heterogeneous network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \mathcal{D})$ is a heterogeneous text-rich network if $\mathcal{D} \neq \varnothing$, $\mathcal{A} = \mathcal{A}_{\text{TR}} \cup \mathcal{A}_{\text{TL}}$, $\mathcal{A}_{\text{TR}} \cap \mathcal{A}_{\text{TL}} = \varnothing$ and $\mathcal{A}_{\text{TR}} \neq \varnothing$, where $\mathcal{A}_{\text{TR}}$ and $\mathcal{A}_{\text{TL}}$ denote the sets of text-rich node types and textless node types, respectively.

2.2 Transformer

A large number of PLMs (*e.g.*, BERT [6]) utilize the multi-layer Transformer architecture [43] to encode texts. Each Transformer layer adopts a multi-head self-attention mechanism to gain a contextualized representation of each text token. Specifically, let $\boldsymbol{H}^{(l)} = [\boldsymbol{h}_1^{(l)}, \boldsymbol{h}_2^{(l)}, ..., \boldsymbol{h}_n^{(l)}]$ denote the output hidden states of the l-th Transformer layer, where $\boldsymbol{h}_i^{(l)} \in \mathcal{R}^d$ is the representation of the text token at position i. Then, the multi-head self-attention (MHA) in the (l+1)-th Transformer layer is calculated as

$$\text{MHA}(\boldsymbol{H}^{(l)}) = \Big\Vert_{t=1}^{k} \text{head}^t(\boldsymbol{H}_t^{(l)}) \tag{1}$$

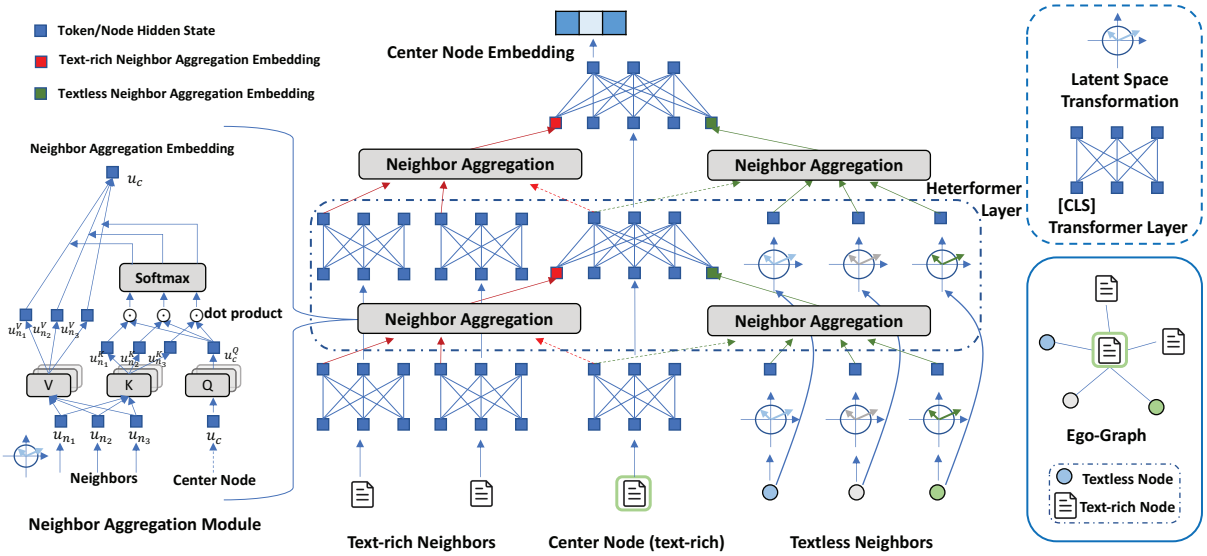


Figure 2: The overall architecture of Heterformer. There are two layers in the figure, while in experiments we have 11 layers. Different color denotes different types of nodes. The whole encoding procedure of Heterformer can be found in Appendix A.1.

$$\text{head}^t(\boldsymbol{H}_t^{(l)}) = \boldsymbol{V}_t^{(l)} \cdot \text{softmax}(\frac{\boldsymbol{K}_t^{(l)\top}\boldsymbol{Q}_t^{(l)}}{\sqrt{d/k}}) \tag{2}$$

$$\boldsymbol{Q}_t^{(l)} = \boldsymbol{W}_{Q,t}^{(l)}\boldsymbol{H}_t^{(l)}, \quad \boldsymbol{K}_t^{(l)} = \boldsymbol{W}_{K,t}^{(l)}\boldsymbol{H}_t^{(l)}, \quad \boldsymbol{V}_t^{(l)} = \boldsymbol{W}_{V,t}^{(l)}\boldsymbol{H}_t^{(l)}, \tag{3}$$

where $\boldsymbol{W}_{Q,t}, \boldsymbol{W}_{K,t}, \boldsymbol{W}_{V,t}$ are query, key, and value matrices to be learned by the model, k is the number of attention head and $\|$ is the concatenate operation.

2.3 Problem Formulation

Definition 2.4. Node Representation Learning on Heterogeneous Text-Rich Networks. Given a heterogeneous text-rich network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \mathcal{D})$, the task is to build a model $f_\Theta : \mathcal{V} \to \mathbb{R}^d$ with parameters Θ to learn meaningful node representation vectors for both text-rich and textless nodes, taking heterogeneous network structures and text semantics into consideration. The learned node embeddings should be able to generalize to various downstream tasks, such as link prediction, node classification and node clustering.

3 METHODOLOGY

In this section, we present the details of Heterformer, the architecture of which is shown in Figure 2. We first introduce how to conduct text-rich node encoding by jointly considering text information and heterogeneous structure information via a Transformer-based architecture. Then, we illustrate how to perform effective textless node learning with heterogeneous node type projection and embedding warm-up. Finally, we discuss how to conduct unsupervised model training.

3.1 Text-Rich Node Encoding

3.1.1 Network-aware Node Text Encoding with Virtual Neighbor Tokens. Encoding text doc_{v_i} of node v_i in a heterogeneous text-rich network differs from encoding plain text, mainly because node texts are associated with network structure information, which can provide auxiliary signals [12]. For example, a paper cited by many deep learning papers in an academic network can be highly related to machine learning. Given that text semantics can be well captured by a multi-layer Transformer architecture [6], we propose a simple but effective way to inject network signals into the Transformer encoding process. The key idea is to introduce *virtual neighbor tokens*. Given a node v_i and its associated texts doc_{v_i}, let $\boldsymbol{H}_{v_i}^{(l)} \in \mathcal{R}^{d\times n}$ denote the output hidden states of all text tokens in doc_{v_i} after the l-th model layer ($l \geqslant 1$). In each layer, we introduce two virtual neighbor tokens to represent v_i's text-rich neighbors $\hat{N}_{v_i}$ and textless neighbors $\check{N}_{v_i}$ in the network respectively. Their embeddings are denoted as $\hat{\boldsymbol{z}}_{v_i}^{(l)}$ and $\check{\boldsymbol{z}}_{v_i}^{(l)} \in \mathcal{R}^d$, which are concatenated to the original text token sequence hidden states as follows (We will discuss how to obtain $\hat{\boldsymbol{z}}_{v_i}^{(l)}$ and $\check{\boldsymbol{z}}_{v_i}^{(l)}$ in Section 3.1.2.):

$$\widetilde{\boldsymbol{H}}_{v_i}^{(l)} = \hat{\boldsymbol{z}}_{v_i}^{(l)} \,\|\, \boldsymbol{H}_{v_i}^{(l)} \,\|\, \check{\boldsymbol{z}}_{v_i}^{(l)}. \tag{4}$$

After the concatenation, $\widetilde{\boldsymbol{H}}_{v_i}^{(l)}$ contains information from both v_i's accompanied text doc_{v_i} and its neighbors in the network, *i.e.*, $\hat{N}_{v_i}$ and $\check{N}_{v_i}$. (It is worth noting that we insert two virtual neighbor tokens to take the *presence or absence of text heterogeneity* of nodes into consideration.) To let the text token representations carry network signals, we adopt a multi-head attention mechanism (MHA):

$$\text{MHA}(\boldsymbol{H}_{v_i}^{(l)}, \widetilde{\boldsymbol{H}}_{v_i}^{(l)}) = \mathop{\Big\|}_{t=1}^{k} \text{head}^t(\boldsymbol{H}_{v_i,t}^{(l)}, \widetilde{\boldsymbol{H}}_{v_i,t}^{(l)}),$$

$$\boldsymbol{Q}_t^{(l)} = \boldsymbol{W}_{Q,t}^{(l)}\boldsymbol{H}_{v_i,t}^{(l)}, \quad \boldsymbol{K}_t^{(l)} = \boldsymbol{W}_{K,t}^{(l)}\widetilde{\boldsymbol{H}}_{v_i,t}^{(l)}, \quad \boldsymbol{V}_t^{(l)} = \boldsymbol{W}_{V,t}^{(l)}\widetilde{\boldsymbol{H}}_{v_i,t}^{(l)}.$$

In the equation above, the multi-head attention is asymmetric (*i.e.*, the keys $\boldsymbol{K}$ and values $\boldsymbol{V}$ are augmented with virtual neighbor embeddings but queries $\boldsymbol{Q}$ are not) to avoid network information being overwritten by text signals and utilize refreshed neighbor representations from each Transformer layer. The output of MHA

includes updated network-aware representations of text tokens. Then, following the Transformer architecture [43], the updated representations will go through a feed-forward network (FFN) to finish our (l+1)-th model layer encoding. Formally,

$$\begin{aligned}\widetilde{\boldsymbol{H}}_{v_i}^{(l)'} &= \mathrm{LN}(\boldsymbol{H}_{v_i}^{(l)} + \mathrm{MHA}(\boldsymbol{H}_{v_i}^{(l)}, \widetilde{\boldsymbol{H}}_{v_i}^{(l)})),\\ \boldsymbol{H}_{v_i}^{(l+1)} &= \mathrm{LN}(\widetilde{\boldsymbol{H}}_{v_i}^{(l)'} + \mathrm{MLP}(\widetilde{\boldsymbol{H}}_{v_i}^{(l)'})).\end{aligned} \tag{5}$$

where $\mathrm{LN}(\cdot)$ denotes the layer normalization function. After L model layers, the final representation of the `[CLS]` token will be used as the node representation of v_i, *i.e.*, $h_{v_i} = \boldsymbol{H}_{v_i}^{(L+1)}$`[CLS]`.

3.1.2 MHA-based Heterogeneous Neighbor Aggregation. The virtual neighbor token embeddings $\widehat{\boldsymbol{z}}_{v_i}^{(l)}, \check{\boldsymbol{z}}_{v_i}^{(l)}$ in Eq.(4) should be the information concentration of v_i's text-rich neighbors and textless neighbors respectively. Inspired by the MHA mechanism in Transformer architectures [43], we design a multi-head attention module to aggregate the information from neighbors and capture the *type heterogeneity* associated with the *edges*. The neighbor aggregation vector $\bar{\boldsymbol{z}}_{v_i}^{(l)}$ ($\in \{\widehat{\boldsymbol{z}}_{v_i}^{(l)}, \check{\boldsymbol{z}}_{v_i}^{(l)}\}$) of the l-th layer for v_i is calculated as follows:

$$\begin{aligned}\bar{\boldsymbol{z}}_{v_i}^{(l)} &= \Big\Vert_{t=1}^{k} \mathrm{head}^t(\boldsymbol{h}_{v_i,t}^{(l)}, \{\boldsymbol{h}_{v_j\to v_i,t}^{(l)} | v_j \in \bar{N}_{v_i}\}),\\ &= \Big\Vert_{t=1}^{k} \sum_{v_j\in\bar{N}_{v_i}\cup\{v_i\}} \bar{\alpha}_{v_i v_j,t}^{(l)} \bar{\boldsymbol{W}}_{V,t}^{(l)} \boldsymbol{h}_{v_j\to v_i,t}^{(l)}.\end{aligned} \tag{6}$$

In the equation, $\bar{x} \in \{\hat{x}, \check{x}\}$ (x can be $\boldsymbol{z}_v$, N_v, α, W). $\hat{x}$ denotes text-rich instances and $\check{x}$ denotes textless instances. $\boldsymbol{h}_{v_i,t}^{(l)} \in \mathbb{R}^{\frac{d}{k}}$ represents the t-th chunk of $\boldsymbol{h}_{v_i}^{(l)}$ (For a text-rich node v_s, $\boldsymbol{h}_{v_s}^{(l)} = \boldsymbol{H}_{v_s}^{(l)}$`[CLS]`; For a textless node v_p, we will discuss how to obtain $\boldsymbol{h}_{v_p}^{(l)}$ in Section 3.2.). d is the dimension of vectors $\boldsymbol{h}_{v_i}^{(l)}$. $\bar{\boldsymbol{W}}_V$ is the value projection matrix. $\alpha_{v_i v_j}$ is the attention weight of v_j to v_i which is calculated as follows,

$$\begin{aligned}\bar{\alpha}_{v_i v_j,t}^{(l)} &= \mathrm{softmax}(\bar{e}_{v_i v_j,t}^{(l)})\\ &= \mathrm{softmax}(\mathrm{Norm}((\bar{\boldsymbol{W}}_{Q,t}^{(l)} \boldsymbol{h}_{x,t}^{(l)})^\top (\bar{\boldsymbol{W}}_{K,t}^{(l)} \boldsymbol{h}_{v_j\to v_i,t}^{(l)}))).\end{aligned} \tag{7}$$

$\bar{\boldsymbol{W}}_Q, \bar{\boldsymbol{W}}_K$ are query and key projection matrices, respectively. $\mathrm{Norm}(\cdot)$ is the scale normalization function, *i.e.*, $\mathrm{Norm}(x) = x/\sqrt{d/k}$. $\boldsymbol{h}_{v_j\to v_i}^{(l)}$ is the propagation vector from v_j to v_i, which is calculated as follows depending on the edge type,

$$\boldsymbol{h}_{v_j\to v_i}^{(l)} = \boldsymbol{W}_r \boldsymbol{h}_{v_j}^{(l)}, \text{ where } \psi(e_{v_j v_i}) = r. \tag{8}$$

In the equation, $e_{v_j v_i}$ denotes the edge between v_j and v_i; $\boldsymbol{W}_r$ is the edge type-aware projection matrix, which is designed for capturing the *edge* semantic *heterogeneity*.

3.2 Textless Node Encoding

3.2.1 Node Type Heterogeneity-based Representation. In this section, we will discuss how to obtain $\boldsymbol{h}_{v_p}$ for a textless node v_p. Although they lack semantically rich text, textless nodes can be quite important as they may contribute significant signals to their neighbors in real-world heterogeneous networks. For example, in an academic network, two papers published in the same venue (textless node) can be on similar topics. Moreover, textless nodes can also be target nodes for downstream tasks such as author classification.

To align with how we encode text-rich nodes using Transformer-based architectures [43] (which is introduced in Section 3.1), a straightforward idea of textless node encoding is to represent each textless node v_p as a high-dimensional learnable embedding vector $\boldsymbol{h}_{v_p}$ (*e.g.*, $\boldsymbol{h}_{v_p} \in \mathbb{R}^{768}$ to be compatible with BERT-base [6] used in text-rich node encoding). Nevertheless, the large population of textless nodes will then introduce a large number of parameters to our framework, which may finally lead to model underfitting. In addition, due to *node type heterogeneity*, different types of nodes can naturally belong to different latent semantic spaces. In summary, we design a simple function to calculate $\boldsymbol{h}_{v_p}^{(l)}$ as follows,

$$\boldsymbol{h}_{v_p}^{(l)} = \boldsymbol{W}_{\phi_i}^{(l)} \boldsymbol{h}_{v_p}^{(0)}, \quad \text{where } \phi(v_p) = \phi_i, \ \ \phi_i \in \mathcal{A}_{\mathrm{TL}}. \tag{9}$$

We set $\boldsymbol{h}_{v_p}^{(0)}$ up as a low-dimensional embedding vector of v_p (*e.g.*, $\boldsymbol{h}_{v_p}^{(0)} \in \mathbb{R}^{64}$) and project it into a more flexible high-dimensional space with a projection matrix $\boldsymbol{W}_{\phi_i}^{(l)}$. For different textless node type $\phi_i \in \mathcal{A}_{\mathrm{TL}}$, we will have different type-specific projection matrix $\boldsymbol{W}_{\phi_i}^{(l)}$ for the transformation. By this design, the *node type heterogeneity* is captured. It is worth noting that the column vectors of $\boldsymbol{W}_{\phi_i}^{(l)}$ can be viewed as "semantic topic embedding vectors" [5] for nodes in type ϕ_i, and each entry of $\boldsymbol{h}_{v_p}^{(0)}$ represents v_p's weight towards one particular topic. In our experiment, we adopt a shared $\boldsymbol{W}_{\phi_i}^{(l)}$ for each layer, since we find this design can contribute to the best performance.

3.2.2 Textless Node Embedding Warm Up. In real-world heterogeneous networks, textless nodes are of a large population. For example, in academic graphs (*e.g.*, DBLP [41]), there are millions of author nodes and thousands of venue nodes, which are not naturally associated with semantically rich texts; in social networks (*e.g.*, Twitter [49]), there are millions of user nodes and hashtag nodes which are textless. Learning textless node embeddings from scratch in Eq. (9) seems to be a solution to capture node semantics. However, the presence of a substantial amount of textless nodes in the framework would add a considerable amount of parameters and may result in the underfitting of the model. Therefore, the parameter initialization of $\boldsymbol{h}_{v_p}^{(0)}$ and $\boldsymbol{W}_{\phi_i}^{(l)}$ will be crucial towards model optimization [11]. To this end, we design a warm-up step to distill information from semantic-rich PLMs into textless node embeddings, which gives textless node embeddings good initializations before Heterformer finetuning. The philosophy is to align the initial textless node embeddings into the same latent space with representations generated by the PLM (which we utilize to initialize the Transformer layers). The warm-up step is shown below:

$$\min_{\boldsymbol{h}_{v_p}^{(l)}} \mathcal{L}_{\boldsymbol{w}} = \sum_{\substack{v_p\in\mathcal{V}\\ \phi(v_p)\in\mathcal{A}_{\mathrm{TL}}}} \sum_{v_u\in\widehat{N}_{v_p}} -\log \frac{\exp(\bar{\boldsymbol{h}}_{v_u}^\top \boldsymbol{h}_{v_p}^{(l)})}{\exp(\bar{\boldsymbol{h}}_{v_u}^\top \boldsymbol{h}_{v_p}^{(l)}) + \sum_{v_u'} \exp(\bar{\boldsymbol{h}}_{v_u'}^\top \boldsymbol{h}_{v_p}^{(l)})}, \tag{10}$$

where v_u' is a text-rich node as a negative sample; $\bar{\boldsymbol{h}}_{v_u}$ is the output vector (corresponding to the `[CLS]` token) from the PLM after encoding the text-rich node v_u. Note that parameters in the PLM are fixed here to make this warm-up process efficient. The PLM utilized here should be the same as that loaded for Transformer layers in Heterformer (Section 3.3.2). After the warm-up, semantic

information from the PLM will be transferred to textless node embeddings, which will benefit Heterformer training a lot. We will demonstrate the effectiveness of this process in Section 4.6.

3.3 Model Training

3.3.1 Training Objective. To train our model, we define the following likelihood objective with parameters Θ:

$$\max_{\Theta} O = \prod_{\substack{v_i \in \mathcal{V} \\ \phi(v_i) \in \mathcal{A}_{\text{TR}}}} \prod_{\substack{v_j \in N_{v_i} \\ \phi(v_j) \in \mathcal{A}_{\text{TR}}}} p(v_j | v_i; \Theta), \tag{11}$$

Here, the conditional probability $p(v_j|v_i;\Theta)$ is calculated as follows:

$$p(v_j|v_i;\Theta) = \frac{\exp(\boldsymbol{h}_{v_j}^\top \boldsymbol{h}_{v_i})}{\sum_{v_u \in \mathcal{V}, \phi(v_u) \in \mathcal{A}_{\text{TR}}} \exp(\boldsymbol{h}_{v_u}^\top \boldsymbol{h}_{v_i})}, \tag{12}$$

where $\boldsymbol{h}_{v_i} = \boldsymbol{h}_{v_i}^{(L+1)}$ is the output node embedding generated by Heterformer with parameters Θ; L is the number of layers in Heterformer. However, calculating Eq. (11) requires enumerating all (v_j, v_i) pairs, which is costly on big graphs. To make the calculation more efficient, we leverage the negative sampling technique [15, 30] to simplify the objective and obtain our loss function below.

$$\min_{\Theta} \mathcal{L} = \sum_{\substack{v_i \in \mathcal{V} \\ \phi(v_i) \in \mathcal{A}_{\text{TR}}}} \sum_{\substack{v_j \in N_{v_i} \\ \phi(v_j) \in \mathcal{A}_{\text{TR}}}} -\log \frac{\exp(\boldsymbol{h}_{v_j}^\top \boldsymbol{h}_{v_i})}{\exp(\boldsymbol{h}_{v_j}^\top \boldsymbol{h}_{v_i}) + \sum_{v_u'} \exp(\boldsymbol{h}_{v_u'}^\top \boldsymbol{h}_{v_i})}. \tag{13}$$

In the equation above, v_u' stands for a random negative sample. In our implementation, we use "in-batch negative samples" [20, 50] to reduce the encoding cost.

3.3.2 Parameter Initialization for Token Embeddings & Transformer Layers. It is shown in [10] that a good parameter initialization before downstream task fine-tuning is essential for deep learning models. Recently, significant improvements achieved by PLMs [4, 6, 25] in various NLP tasks have also demonstrated this finding. In Heterformer, a large proportion of parameters in Heterformer are token embeddings and parameters in transformer layers. Fortunately, these parameters are well pre-trained in many PLMs [1, 4, 6, 25, 34, 51]. As a result, Heterformer can directly load this part of initial parameters from a PLM.

3.4 Discussions

3.4.1 Discussion on the connection between Heterformer and GNNs. According to Figure 2, Heterformer adopts a Transformer-based architecture. Meanwhile, it can also be viewed as a graph neural network (GNN) model. In general, a GNN layer is consisted of neighbor propagation and aggregation to obtain node representations [12, 22, 44] as follows,

$$\boldsymbol{a}_{v_i v_j}^{(l-1)} = \text{PROP}^{(l)}\left(\boldsymbol{h}_{v_i}^{(l-1)}, \boldsymbol{h}_{v_j}^{(l-1)}\right), (\forall v_j \in N_{v_i}); \tag{14}$$

$$\boldsymbol{h}_{v_i}^{(l)} = \text{AGG}^{(l)}\left(\boldsymbol{h}_{v_i}^{(l-1)}, \{\boldsymbol{a}_{v_i v_j}^{(l-1)} | v_j \in N_{v_i}\}\right). \tag{15}$$

Analogously, in Heterformer, Eq. (8) can be treated as the propagation function $\text{PROP}(\cdot)$, while the aggregation step $\text{AGG}(\cdot)$ is the combination of Eqs. (6), (4) and (5). Essentially, the propagation and aggregation function in Heterformer are both heterogeneity-aware mechanisms.

3.4.2 Discussion on Complexity. **Time Complexity:** Given a center node with m text-rich neighbors (each of which has p tokens) and n textless neighbors, the time complexity of each Heterformer layer's encoding step is $O(p^2(m+1)+m+n)$, which is on par with the complexity $O(p^2(m+1))$ of per GNN-cascaded Transformers layer since $m, n \ll p^2 m$. Another straightforward idea of fusing center node text information with its neighbor representations is to directly concatenate token embeddings of the center node, its text-rich neighbors, and its textless neighbors together and feed them into a PLM. However, in this way, the time complexity of one such layer becomes $O((p(m+1)+n)^2)$, which is significantly larger than that of our method. **Memory Complexity:** Given a network with N textless nodes and T parameters in the Transformer layers, the parameter complexity of Heterformer is $O(T + Nd)$, which is the same with heterogeneous GNN-cascaded Transformers [35].

4 EXPERIMENT

4.1 Experimental Settings

4.1.1 Datasets. We conduct experiments on three datasets (*i.e.*, DBLP [41], Twitter [55], and Goodreads [45]) from three different domains (*i.e.*, academic papers, social media posts, and books). For DBLP*, we extract papers published from 1990 to 2020 with their author and venue information. For Twitter†, we merge the original LA and NY datasets to form a larger dataset. For Goodreads‡, we remove books without any similar books, and the remaining books with their meta-data fields form the dataset. The main statistics of the three datasets are summarized in Table 1.

Table 1: Dataset statistics. *: text-rich node types.

Dataset	Node	Edge
DBLP	# paper*: 3,597,191 # venue: 28,638 # author: 2,717,797	# paper-paper: 36,787,329 # venue-paper: 3,633,613 # author-paper: 10,212,497
Twitter	# tweet*: 279,694 # POI*: 36,895 # hashtag: 72,297 # user: 76,398 # mention: 24,089	# tweet-POI: 279,694 # user-tweet: 195,785 # hashtag-tweet: 194,939 # mention-tweet: 50,901
Goodreads	# book*:1,097,438 # shelves: 6,632 # author: 205,891 # format: 768 # publisher: 62,934 # language code: 139	# book-book: 11,745,415 # shelves-book: 27,599,160 # author-book: 1,089,145 # format-book: 588,677 # publisher-book: 591,456 # language code-book: 485,733

4.1.2 Baselines. We compare Heterformer with two groups of baselines: **GNN-cascaded Transformers** and **Nested Transformers**. The former group can be further classified into *homogeneous GNN-cascaded Transformers*, including BERT+MeanSAGE [12], BERT+MaxSAGE [12] and BERT+GAT [44], and *heterogeneous GNN-cascaded Transformers*, including BERT+RGCN [35], BERT+HAN [46], BERT+HGT [13] and BERT+SHGN [27]. The latter group includes the recent GraphFormers [50] model. However, GraphFormers can only deal with homogeneous textual networks. To apply it to heterogeneous text-rich networks, we add heterogeneous graph propagation

*https://originalstatic.aminer.cn/misc/dblp.v12.7z
†https://drive.google.com/file/d/0Byrzhr4bOatCRHdmRVZ1YVZqSzA/view?resourcekey=0-3_R5EWrLYjaVuysxPTqe5A
‡https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/home

Table 2: Experiment results on link prediction. *: Heterformer significantly outperforms the best baseline with p-value < 0.05.

	Method	DBLP			Twitter			Goodreads		
		PREC	MRR	NDCG	PREC	MRR	NDCG	PREC	MRR	NDCG
	MeanSAGE	0.7019	0.7964	0.8437	0.6489	0.7450	0.7991	0.6302	0.7409	0.8001
	BERT	0.7569	0.8340	0.8726	0.7179	0.7833	0.8265	0.5571	0.6668	0.7395
Homo GNN	BERT+MeanSAGE	0.8131	0.8779	0.9070	0.7201	0.7845	0.8275	0.7301	0.8167	0.8594
Homo GNN	BERT+MAXSAGE	0.8193	0.8825	0.9105	0.7198	0.7845	0.8276	0.7280	0.8164	0.8593
Homo GNN	BERT+GAT	0.8119	0.8771	0.9063	0.7231	0.7873	0.8300	0.7333	0.8170	0.8593
Homo GNN	GraphFormers	0.8324	0.8916	0.9175	0.7258	0.7891	0.8312	0.7444	0.8260	0.8665
Hetero GNN	BERT+RGCN	0.7979	0.8633	0.8945	0.7111	0.7764	0.8209	0.7488	0.8303	0.8699
Hetero GNN	BERT+HAN	0.8136	0.8782	0.9072	0.7237	0.7880	0.8306	0.7329	0.8174	0.8597
Hetero GNN	BERT+HGT	0.8170	0.8814	0.9098	0.7153	0.7800	0.8237	0.7224	0.8112	0.8552
Hetero GNN	BERT+SHGN	0.8149	0.8785	0.9074	0.7218	0.7866	0.8295	0.7362	0.8195	0.8613
Hetero GNN	GraphFormers++	0.8233	0.8856	0.9130	0.7159	0.7799	0.8236	0.7536	0.8328	0.8717
	Heterformer	**0.8474***	**0.9019***	**0.9255***	**0.7272***	**0.7908***	**0.8328***	**0.7633***	**0.8400***	**0.8773***

and aggregation in its final layer. This generalized model is named GraphFormers++. To verify the importance of both text and network information in text-rich networks, we also include vanilla GraphSAGE [12] and vanilla BERT [6] in comparison. Detailed information about the baselines can be found in Appendix A.2.

4.1.3 Reproducibility. For all compared models (baselines and Heterformer), we adopt the same training objective and the 12-layer BERT-base-uncased [6] as the backbone PLM for a fair comparison. The Adam optimizer [21] with a learning rate 1e-5 and in-batch negative samples with training batch size 30 are used to fine-tune the model. In-batch testing is used for efficiency and the test batch size is 100, 300, and 100 for DBLP, Twitter, and Goodreads, respectively. The maximum length of the PLM is set to be 32, 12, and 64 on the three datasets according to their average document length. For heterogeneous GNN approaches, the embedding size of textless nodes is 64. We run experiments on one NVIDIA RTX A6000 GPU.

Following previous studies on network representation learning, we consider three fundamental tasks for quantitative evaluation: link prediction, node classification, and node clustering.

4.2 Link Prediction

Settings. Link prediction aims to predict missing edges in a network. In order to evaluate the models' ability to encode both text semantics and network structure, we focus on link prediction between two text-rich nodes. Specifically, on DBLP, Twitter, and Goodreads, the prediction is between paper-paper, tweet-POI, and book-book, respectively. The model is trained and tested with in-batch negative sampling and we adopt a 7:1:2 train-dev-test split. Precision@1 (PREC), Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG) are used as evaluation metrics. Given a query node u, PREC measures whether the key node v linked with u is ranked the highest in the batch; MRR calculates the average of the reciprocal ranks of v; NDCG further takes the order and relative importance of v into account and here we calculate on the full candidate list, the length of which equals to test batch size.

Results. Table 2 shows the performance of all compared methods. From Table 2, we can observe that: (a) Heterformer outperforms all the baseline methods consistently; (b) Transformer+GNN models perform better than both vanilla GNN and vanilla BERT, which demonstrates the importance of encoding both text and network signals in text-rich networks; (c) Network-empowered Transformers including Heterformer, GraphFormers, and GraphFormers++ are more powerful than GNN-cascaded Transformers; (d) By considering network heterogeneity, Heterformer can have better performance than GraphFormers in heterogeneous text-rich networks. (e) Heterformer yields a larger performance improvement when the network is more dense and heterogeneous (*i.e.*, DBLP, Goodreads vs. Twitter).

4.3 Node Classification

Settings. In node classification, we train a 2-layer MLP classifier to classify nodes with the generated node embeddings from each model as input. The node embeddings are fixed in order to test their representation quality. The experiments are conducted on DBLP and Goodreads (because node labels are available in these two datasets) for both **text-rich** and **textless** nodes. For **text-rich** node classification, we focus on paper nodes and book nodes in DBLP and Goodreads, respectively. We select the most frequent 30 classes in DBLP and keep the original 10 classes in Goodreads. Also, we study both *transductive* and *inductive* node classification to understand the capability of our model comprehensively. For *transductive* node classification, the model has seen the classified nodes during representation learning (using the link prediction objective), while for *inductive* node classification, the model needs to predict the label of nodes not seen before. For **textless** node classification, we focus on author nodes in both DBLP and Goodreads. The label of each author is obtained by aggregating the labels of his/her publications. We separate the whole dataset into train set, validation set, and test set in 7:1:2 in all cases and each experiment is repeated 5 times in this section with the average performance reported. Further information can be found in Appendix A.3.3.

Results. Tables 3 and 4 demonstrate the results of different methods in *transductive* and *inductive* **text-rich** node classification. We observe that: (a) our Heterformer outperforms all the baseline methods significantly on both tasks, showing that Heterformer can learn more effective node representations for these tasks; (b) Heterogeneous network-based Transformer methods generally achieve better results than homogeneous network-based Transformer methods, which demonstrates the necessity of encoding heterogeneity in heterogeneous text-rich networks; (c) Heterformer generalizes

Table 3: Transductive text-rich node classification.

Method	DBLP		Goodreads	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1
BERT	0.6119	0.5476	0.8364	0.7713
BERT+MaxSAGE	0.6179	0.5511	0.8447	0.7866
BERT+MeanSAGE	0.6198	0.5522	0.8420	0.7826
BERT+GAT	0.5943	0.5175	0.8328	0.7713
GraphFormers	0.6256	0.5616	0.8388	0.7786
BERT+HAN	0.5965	0.5211	0.8351	0.7747
BERT+HGT	0.6575	0.5951	0.8474	0.7928
BERT+SHGN	0.5982	0.5214	0.8345	0.7737
GraphFormers++	0.6474	0.5790	0.8516	0.7993
Heterformer	**0.6695***	**0.6062***	**0.8578***	**0.8076***

Table 4: Inductive text-rich node classification.

Method	DBLP		Goodreads	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1
BERT	0.5996	0.5318	0.8122	0.7371
BERT+MaxSAGE	0.6117	0.5435	0.8368	0.7749
BERT+MeanSAGE	0.6129	0.5431	0.8350	0.7721
BERT+GAT	0.5879	0.5150	0.8249	0.7590
GraphFormers	0.6197	0.5548	0.8330	0.7683
BERT+HAN	0.5948	0.5165	0.8279	0.7626
BERT+HGT	0.6467	0.5835	0.8390	0.7798
BERT+SHGN	0.5955	0.5202	0.8280	0.7626
GraphFormers++	0.6386	0.5696	0.8427	0.7848
Heterformer	**0.6600***	**0.5976***	**0.8507***	**0.7977***

Table 5: Textless node classification.

Method	DBLP		Goodreads	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1
BERT+HAN	0.0604	0.0270	0.4726	0.2464
BERT+HGT	0.0883	0.0539	0.4758	0.1963
BERT+SHGN	0.0619	0.0286	0.4733	0.2457
BERT+RGCN	0.2201	0.1687	0.5768	0.3948
GraphFormers++	0.1072	0.0698	0.5007	0.2772
Heterformer	**0.3817***	**0.3305***	**0.6292***	**0.4835***

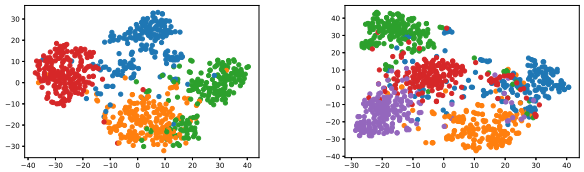
quite well on unseen nodes as its performance on inductive node classification is quite close to that on transductive node classification. Moreover, Heterformer even achieves higher performance in inductive settings than the baselines do in transductive settings. Table 5 reports the result on **textless** node classification, where we have the following findings: (a) Heterformer outperforms all heterogeneous network-based Transformer methods significantly. (b) Compared with text-rich node classification, the improvement of Heterformer on textless node classification over baselines is more significant, indicating that Heterformer better captures neighbors' text semantics in textless node representations.

4.4 Node Clustering

Settings. For node clustering, we utilize KMeans [19] to cluster the nodes based on their representations generated by the models. The data and categories used in Section 4.3 for text-rich node classification are used here again, but nodes with more than one ground-truth label are filtered. The number of clusters K is set as the number of categories. For DBLP, since the dataset is quite large, we pick the 10 most frequent categories and randomly select 20,000 nodes for efficient evaluation. NMI and ARI [14] are used as evaluation metrics. Since the performance of KMeans can be affected by the initial centroids, we run each experiment 10 times and report the average performance. In addition to quantitative evaluation, we conduct visualization to depict the distribution of Heterformer embeddings, where t-SNE [42] is utilized to project node embeddings into a 2-dimensional space and the nodes are colored based on their ground-truth label. Further information can be found in Appendix A.3.4 and A.3.5.

Table 6: Node clustering.

Method	DBLP		Goodreads	
	NMI	ARI	NMI	ARI
BERT	0.2570	0.3349	0.2325	0.4013
BERT+MaxSAGE	0.2615	0.3490	0.2205	0.4173
BERT+MeanSAGE	0.2628	0.3488	**0.2449**	**0.4329**
BERT+GAT	0.2598	0.3419	0.2408	0.4185
GraphFormers	0.2633	0.3455	0.2362	0.4139
BERT+HAN	0.2568	0.3401	0.2391	0.4266
BERT+HGT	0.2469	0.3392	0.2427	0.4296
BERT+SHGN	0.2589	0.3431	0.2373	0.4171
GraphFormers++	0.2566	0.3432	0.2372	0.4211
Heterformer	**0.2707***	**0.3639***	0.2429	0.4199



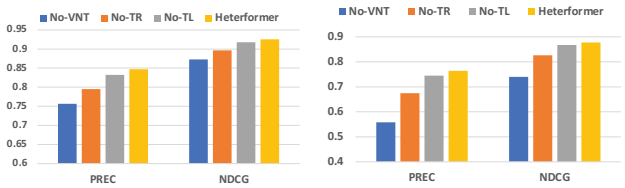
(a) DBLP (b) Goodreads

Figure 3: Embedding visualization.

Results. The quantitative result can be found in Table 6, where Heterformer is the best on DBLP and outperforms most baselines on Goodreads. The embedding visualization of Heterformer is presented in Figure 3. In both datasets, the clustering structure is quite evident, indicating that node representations learned by Heterformer are category-discriminative, even though the training process is based on link prediction only.

4.5 Ablation and Parameter Studies

4.5.1 Ablation Study on Virtual Neighbor Tokens. In Section 3.1.1, signals from both text-rich and textless neighbors are incorporated and finally contribute to center node encoding serving as two virtual neighbor tokens. To study the effectiveness of information from both text-rich and textless neighbors, we conduct a model study of several Heterformer variants: (a) **No-VNT** adds no **V**irtual **N**eighbor **T**okens and only encodes textual information for each node; (b) **No-TR** (**T**ext-**R**ich) only adds one virtual neighbor token corresponding to textless neighbors in Eq. (4); (c) **No-TL** (**T**ext**L**ess) only adds one

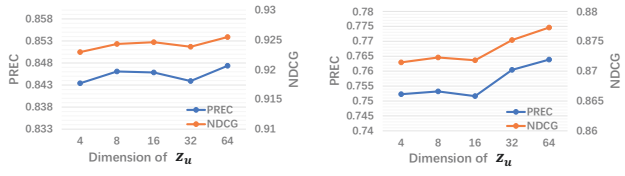


(a) DBLP (b) Goodreads

Figure 4: Ablation study on neighbor aggregation.

Table 7: Ablation study on heterogeneous projection.

Method	DBLP		Goodreads	
	PREC	MRR	PREC	MRR
Heterformer	0.8474	0.9019	0.7633	0.8400
Heterformer w/o hp	0.8415	0.8983	0.7493	0.8325



(a) DBLP (b) Goodreads

Figure 5: Effect of textless node embedding dimension.

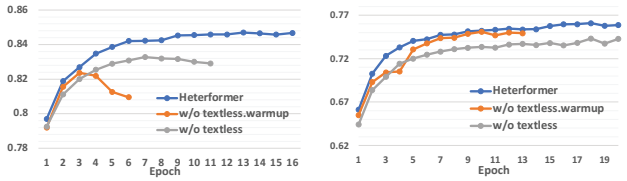
virtual neighbor token corresponding to text-rich neighbors in Eq. (4); (d) **Heterformer** is our full model. The results of link prediction for these variants are shown in Figure 4. We can find that: (a) Heterformer outperforms all model variants, which demonstrates that signals from both text-rich and textless neighbors are essential for center node encoding; (b) No-TL performs better than No-TR, implying that text-rich neighbors are more important than textless neighbors since they contain rich text semantics.

4.5.2 Ablation Study on Type Heterogeneous Projection Matrices. In Eq. (8) and Eq. (9), we propose to utilize different projection matrices (so-called heterogeneous projection, *hp* for short) for different types of nodes and edges. In this section, we conduct an ablation study to verify the effectiveness of this design. The model with the same projection matrices for different types of nodes and edges is denoted as **Heterformer w/o hp**, while our full model is denoted as **Heterformer**. The results are shown in Table 7. From the result, Heterformer consistently outperforms Heterformer w/o hp on both DBLP and Goodreads, which demonstrates the importance of this design and the necessity of modeling node/edge type heterogeneity.

4.5.3 Dimension of Textless Node Embedding. To understand the effect of textless node embedding dimension, we test the performance of Heterformer in link prediction with the embedding dimension varying in 4, 8, 16, 32, and 64. The result is shown in Figure 5. It can be seen that the performance of Heterformer generally increases as the embedding dimension becomes larger. This is intuitive since the more parameters z_u has (before overfitting), the more information it can represent.

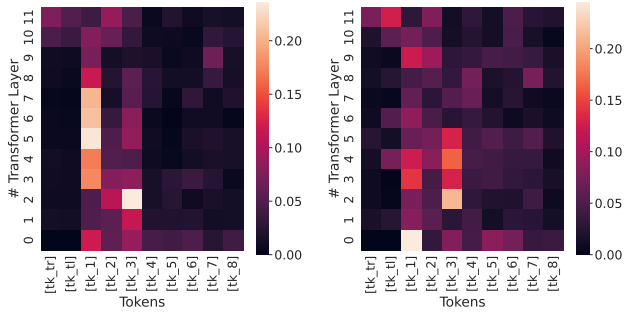
4.6 Textless Node Embedding Warm-Up

4.6.1 Training Curve Study. In Section 3.2.2, we propose a method to warm up textless node embeddings. Now, we empirically demonstrate the effectiveness of such a warm-up process. The training



(a) DBLP, Validation PREC (b) Goodreads, Validation PREC

Figure 6: Performance of Heterformer during the training process with and without textless node warm-up.



(a) w/o warm-up (b) w/ warm-up

Figure 7: Self-attention probability map study of Heterformer with and without textless node warm-up for a random sample. The x-axis corresponds to different key/value tokens and the y-axis corresponds to different Heterformer layers.

processes (Section 4.2) of Heterformer without textless node warm-up (**w/o textless.warm-up**) and the full Heterformer model are shown in Figure 6. We also show Heterformer without the utilization of textless node neighbor information (**w/o textless**) as reference. The x-axis denotes the number of training epochs, while the y-axis represents PREC on the validation set. Training is terminated if PREC on the validation set does not increase for three consecutive epochs. It is shown that: (a) On both datasets, Heterformer with textless node embedding warm-up can have better performance than that without textless node warm-up; (b) On DBLP, Heterformer without textless node embedding warm-up cannot even outperform Heterformer without the utilization of textless neighbor information. This finding implies the necessity of good initialization for textless node embeddings. Since modeling textless nodes can improve the representation capacity (see Section 4.5.1) in text-rich networks, our warm-up strategy is, therefore, verified to be effective towards model convergence.

4.6.2 Attention Map Study. In order to understand how the warm-up step proposed in Section 3.2.2 benefits Heterformer training, we further conduct a self-attention probability map study for a random sample from DBLP in Figure 7. We random pick up a token from this sample and plot the self-attention probability of how different tokens (x-axis), including virtual neighbor tokens ([tk_tr] and [tk_tl] are the text-rich and the textless neighbor virtual tokens respectively) and the first eight original text tokens ([tk_x], x∈ {1..8}), will contribute to the encoding of this random token in different layers (y-axis). From the figure, we can find that the virtual neighbor tokens (first two columns from left) are more deactivated for the model without warm-up, which means that the information

Table 8: Scalability Study for BERT+MeanSAGE, Graphformers and Heterformer on Goodreads.

Model	Time	Memory
BERT+MeanSAGE	440.18ms	19,637MB
GraphFormers	490.27ms	20,385MB
Heterformer	508.27ms	20,803MB

from neighbors is not well utilized during encoding. However, the neighbor virtual tokens (first two columns from left) become more activated after warm-up, bringing more useful information from neighbors to enhance center node text encoding.

4.7 Scalability Study

We conduct theoretical analysis on time complexity and memory complexity for Heterformer in Section 3.4.2. In this section, we perform an empirical time and memory efficiency comparison among BERT+MeanSAGE, GraphFormers, and Heterformer. The evaluation is performed on one NVIDIA RTX A6000 GPU. The result is shown in Table 8. For time complexity, we run each model for one mini-batch (each mini-batch contains 30 samples) and report the average running time. For memory complexity, we report the GPU memory needed to train the corresponding models. From the results, we can find that the time and memory cost of training Heterformer is quite close to that of BERT+MeanSAGE and GraphFormers.

5 RELATED WORK

5.1 Pretrained Language Models

Pretrained language models (PLMs) aim to learn general language representations from large-scale corpora, which can be generalized to various downstream tasks. Early studies on PLMs mainly focus on context-free text embeddings such as word2vec [30] and GloVe [31]. Recently, motivated by the fact that the same word can have different meanings conditioned on different contexts, deep language models such as ELMo [33], BERT [6], RoBERTa [25], XL-Net [51], ELECTRA [4], and GPT [1, 34] are proposed to capture the contextualized token representations. These models employ the Transformer architecture [43] to capture long-range and high-order semantic dependency and achieve significant improvement on many downstream NLP tasks [24, 29, 47]. However, these models mainly focus on text encoding. In contrast, Heterformer leverages both text and heterogeneous structure (network) information when the latter is available.

5.2 Heterogeneous Graph Neural Networks

Graph neural networks (GNNs) such as GCN [22], GraphSAGE [12], and GAT [44] have been widely adopted in representation learning on graphs. Since real-world objects and interactions are often multi-typed, recent studies have considered extending GNNs to heterogeneous graphs [39]. The basic idea of heterogeneous graph neural networks (HGNNs) [2, 13, 35, 46, 52, 53] is to leverage node types, edge types, and meta-path semantics [40] in projection and aggregation. For example, HAN [46] proposes a hierarchical attention mechanism to capture both node and meta-path importance; HGT [13] proposes an architecture similar to Transformer [43] to carry out attention on edge types. For more HGNN models, one can refer to recent surveys [8, 48]. Lv et al. [27] further perform a benchmark study of 12 HGNNs and propose a simple HGNN model based on GAT. Despite the success of these models, when some types of nodes carry text information, they lack the power of handling textual signals in a contextualized way. In contrast, Heterformer jointly models text semantics and heterogeneous structure (network) signal in each Transformer layer.

5.3 Text-Rich Networks

Most previous studies on *homogeneous* text-rich networks adopt a "cascaded architecture" [18, 23, 26, 58, 59]. One drawback of such models is that text and network signals are processed consecutively, so the network information cannot benefit text encoding. To overcome this drawback, GraphFormers [50] introduce nested Transformers so that text and node features can be encoded jointly. Edgeformers [17] introduce graph-empowered Transformers for representation learning on textual-edge networks. However, they assume that the network is homogeneous and all nodes have text information. These assumptions do not usually hold in real-world text-rich networks. Most previous studies on *heterogeneous* text-rich networks focus on specific text-related tasks. For example, HyperMine [37] and NetTaxo [36] study how network structures can benefit taxonomy construction from text corpora; LTRN [56] and MATCH [57] leverage document metadata as complementary signals for text classification. In comparison, Heterformer focuses on the generic representation learning task. As far as we know, SHNE [54] is the major previous work also studying representation learning on heterogeneous text-rich networks. However, it still adopts the "cascaded architecture" mentioned above and does not explore the power of Transformer encoders (as it was proposed before BERT [6]). In comparison, Heterformer proposes a heterogeneous network-empowered Transformer which can jointly capture textual signals and structure signals.

6 CONCLUSIONS

In this paper, we introduce the problem of node representation learning on heterogeneous text-rich networks and propose Heterformer, a heterogeneous network-empowered Transformer architecture to address the problem. Heterformer can jointly capture the heterogeneous structure (network) information and the rich contextualized textual information hidden inside the networks. Experimental results on various graph mining tasks, including link prediction, node classification, and node clustering, demonstrate the superiority of Heterformer. Moreover, the proposed framework can serve as a building block with different task-specific inductive biases. It would be interesting to see its future applications on real-world text-rich networks such as recommendation, abuse detection, tweet-based network analysis, and text-rich social network analysis.

ACKNOWLEDGMENTS

This work was supported in part by US DARPA KAIROS Program No. FA8750-19-2-1004 and INCAS Program No. HR001121C0165, National Science Foundation IIS-19-56151, IIS-17-41317, and IIS 17-04532, and the Molecule Maker Lab Institute: An AI Research Institutes program supported by NSF under Award No. 2019897, and the Institute for Geospatial Understanding through an Integrative Discovery Environment (I-GUIDE) by NSF under Award No. 2118329. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily represent the views, either expressed or implied, of DARPA or the U.S. Government.

REFERENCES

[1] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *NeurIPS*.

[2] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation learning for attributed multiplex heterogeneous network. In *KDD*. 1358–1368.

[3] Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S Dhillon. 2020. Taming pretrained transformers for extreme multi-label text classification. In *KDD*. 3163–3171.

[4] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *ICLR*.

[5] Peng Cui, Le Hu, and Yuanchao Liu. 2020. Enhancing extractive text summarization with topic-aware graph neural networks. In *COLING*.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.

[7] Xin Luna Dong, Xiang He, Andrey Kan, Xian Li, Yan Liang, Jun Ma, Yifan Ethan Xu, Chenwei Zhang, Tong Zhao, Gabriel Blanco Saldana, et al. 2020. AutoKnow: Self-driving knowledge collection for products of thousands of types. In *KDD*.

[8] Yuxiao Dong, Ziniu Hu, Kuansan Wang, Yizhou Sun, and Jie Tang. 2020. Heterogeneous Network Representation Learning. In *IJCAI*. 4861–4867.

[9] Ahmed El-Kishky, Thomas Markovich, Serim Park, Chetan Verma, Baekjin Kim, Ramy Eskander, Yury Malkov, Frank Portman, Sofía Samaniego, Ying Xiao, et al. 2022. Twhin: Embedding the twitter heterogeneous information network for personalized recommendation. In *KDD*. 2842–2850.

[10] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. 2009. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *AISTATS*. 153–160.

[11] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 249–256.

[12] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*. 1025–1035.

[13] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *WWW*. 2704–2710.

[14] Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification* 2, 1 (1985), 193–218.

[15] Bowen Jin, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. 2020. Multi-behavior recommendation with graph convolutional networks. In *SIGIR*.

[16] Bowen Jin, Wentao Zhang, Yu Zhang, Yu Meng, Xinyang Zhang, Qi Zhu, and Jiawei Han. 2023. Patton: Language Model Pretraining on Text-Rich Networks. *ACL*.

[17] Bowen Jin, Yu Zhang, Yu Meng, and Jiawei Han. 2023. Edgeformers: Graph-Empowered Transformers for Representation Learning on Textual-Edge Networks. In *ICLR*.

[18] Di Jin, Xiangchen Song, Zhizhi Yu, Ziyang Liu, Heling Zhang, Zhaomeng Cheng, and Jiawei Han. 2021. BiTe-GCN: A New GCN Architecture via Bidirectional Convolution of Topology and Features on Text-Rich Networks. In *WSDM*.

[19] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. 2002. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE TPAMI* 24, 7 (2002), 881–892.

[20] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP*.

[21] Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

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

[23] Chaozhuo Li, Bochen Pang, Yuming Liu, Hao Sun, Zheng Liu, Xing Xie, Tianqi Yang, Yanling Cui, Liangjie Zhang, and Qi Zhang. 2021. AdsGNN: Behavior-Graph Augmented Relevance Modeling in Sponsored Search. In *SIGIR*. 223–232.

[24] Yang Liu. 2019. Fine-tune BERT for extractive summarization. *arXiv preprint arXiv:1903.10318* (2019).

[25] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv:1907.11692* (2019).

[26] Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2020. Fine-grained Fact Verification with Kernel Graph Attention Network. In *ACL*.

[27] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. 2021. Are we really making much progress?: Revisiting, benchmarking and refining heterogeneous graph neural networks. In *KDD*. 1150–1160.

[28] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27, 1 (2001), 415–444.

[29] Yu Meng, Yunyi Zhang, Jiaxin Huang, Yu Zhang, and Jiawei Han. 2022. Topic discovery via latent space clustering of pretrained language model representations. In *WWW*. 3143–3152.

[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.

[31] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. 1532–1543.

[32] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. 701–710.

[33] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*. 2227–2237.

[34] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[35] Michael 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*. 593–607.

[36] Jingbo Shang, Xinyang Zhang, Liyuan Liu, Sha Li, and Jiawei Han. 2020. Nettaxo: Automated topic taxonomy construction from text-rich network. In *WWW*.

[37] Yu Shi, Jiaming Shen, Yuchen Li, Naijing Zhang, Xinwei He, Zhengzhi Lou, Qi Zhu, Matthew Walker, Myunghwan Kim, and Jiawei Han. 2019. Discovering hypernymy in text-rich heterogeneous information network by exploiting context granularity. In *CIKM*. 599–608.

[38] Suzanna Sia, Ayush Dalmia, and Sabrina J Mielke. 2020. Tired of topic models? clusters of pretrained word embeddings make for fast and good topics too! *EMNLP* (2020).

[39] Yizhou Sun and Jiawei Han. 2012. Mining heterogeneous information networks: principles and methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery* 3, 2 (2012), 1–159.

[40] 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. *PVLDB* 4, 11 (2011), 992–1003.

[41] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *KDD*. 990–998.

[42] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR* 9, 11 (2008).

[43] 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 *NIPS*. 5998–6008.

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

[45] Mengting Wan and Julian McAuley. 2018. Item recommendation on monotonic behavior chains. In *RecSys*. 86–94.

[46] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW*. 2022–2032.

[47] Guangxu Xun, Kishlay Jha, Jianhui Sun, and Aidong Zhang. 2020. Correlation networks for extreme multi-label text classification. In *KDD*. 1074–1082.

[48] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. 2020. Heterogeneous network representation learning: A unified framework with survey and benchmark. *IEEE TKDE* (2020).

[49] Jaewon Yang and Jure Leskovec. 2011. Patterns of temporal variation in online media. In *WSDM*. 177–186.

[50] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. 2021. GraphFormers: GNN-nested Transformers for Representation Learning on Textual Graph. In *NeurIPS*.

[51] 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*. 5754–5764.

[52] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. *NeurIPS* 32 (2019), 11983–11993.

[53] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *KDD*. 793–803.

[54] Chuxu Zhang, Ananthram Swami, and Nitesh V Chawla. 2019. Shne: Representation learning for semantic-associated heterogeneous networks. In *WSDM*.

[55] Chao Zhang, Guangyu Zhou, Quan Yuan, Honglei Zhuang, Yu Zheng, Lance Kaplan, Shaowen Wang, and Jiawei Han. 2016. Geoburst: Real-time local event detection in geo-tagged tweet streams. In *SIGIR*. 513–522.

[56] Xinyang Zhang, Chenwei Zhang, Xin Luna Dong, Jingbo Shang, and Jiawei Han. 2021. Minimally-Supervised Structure-Rich Text Categorization via Learning on Text-Rich Networks. In *WWW*. 3258–3268.

[57] Yu Zhang, Zhihong Shen, Yuxiao Dong, Kuansan Wang, and Jiawei Han. 2021. MATCH: Metadata-Aware Text Classification in A Large Hierarchy. In *WWW*.

[58] Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2019. GEAR: Graph-based Evidence Aggregating and Reasoning for Fact Verification. In *ACL*. 892–901.

[59] Jason Zhu, Yanling Cui, Yuming Liu, Hao Sun, Xue Li, Markus Pelger, Tianqi Yang, Liangjie Zhang, Ruofei Zhang, and Huasha Zhao. 2021. Textgnn: Improving text encoder via graph neural network in sponsored search. In *WWW*. 2848–2857.

A SUPPLEMENTARY MATERIAL

A.1 Summary of Heterformer's Encoding Procedure

Algorithm 1: Encoding Procedure of Heterformer

Input : The center node v_i, its text-rich neighbors $\widehat{N}_{v_i}$ and textless neighbors $\breve{N}_{v_i}$. Initial token sequence embedding $\boldsymbol{H}_{v_j}^{(0)}$ for $v_j \in \widehat{N}_{v_i} \cup \{v_i\}$.

Output : The embedding $\boldsymbol{h}_{v_i}$ for the center node v_i.

begin

// obtain text-rich nodes' first layer encoded token embeddings

for $v_j \in \widehat{N}_{v_i} \cup \{v_i\}$ **do**

$\boldsymbol{H}_{v_j}^{(0)'} \leftarrow \text{Normalize}(\boldsymbol{H}_{v_j}^{(0)} + \text{MHA}^{(0)}(\boldsymbol{H}_{v_j}^{(0)}))$;

$\boldsymbol{H}_{v_j}^{(1)} \leftarrow \text{Normalize}(\boldsymbol{H}_{v_j}^{(0)'} + \text{MLP}^{(0)}(\boldsymbol{H}_{v_j}^{(0)'}))$;

end

// obtain textless nodes' initial embedding after warm-up

for $v_s \in \breve{N}_{v_i}$ **do**

$\boldsymbol{h}_{v_s}^{(0)} \leftarrow \text{WarmUp}(v_s)$;

end

for $l = 1, ..., L$ **do**

// text-rich neighbor aggregation

for $v_j \in \widehat{N}_{v_i} \cup \{v_i\}$ **do**

$\boldsymbol{h}_{v_j}^{(l)} \leftarrow \boldsymbol{H}_{v_j}^{(l)}\texttt{[CLS]}$;

end

$\widehat{\boldsymbol{z}}_{v_i}^{(l)} \leftarrow \text{AGG}(\{\boldsymbol{h}_{v_j}^{(l)} | v_j \in \widehat{N}_{v_i} \cup \{v_i\}\})$;

// textless neighbor aggregation

for $v_s \in \breve{N}_{v_i}$ **do**

$\boldsymbol{h}_{v_s}^{(l)} \leftarrow \boldsymbol{W}_{\phi_i}^{(l)} \boldsymbol{h}_{v_s}^{(0)}$, where $\phi(v_s) = \phi_i$;

end

$\breve{\boldsymbol{z}}_{v_i}^{(l)} \leftarrow \text{AGG}(\{\boldsymbol{h}_{v_s}^{(l)} | v_s \in \breve{N}_{v_i} \cup \{v_i\}\})$;

// obtain the center node's token embedding for next layer

$\widetilde{\boldsymbol{H}}_{v_i}^{(l)} \leftarrow \widehat{\boldsymbol{z}}_{v_i}^{(l)} \parallel \boldsymbol{H}_{v_i}^{(l)} \parallel \breve{\boldsymbol{z}}_{v_i}^{(l)}$;

$\widetilde{\boldsymbol{H}}_{v_i}^{(l)'} \leftarrow \text{Normalize}(\boldsymbol{H}_{v_i}^{(l)} + \text{MHA}^{(l)}(\boldsymbol{H}_{v_i}^{(l)}, \widetilde{\boldsymbol{H}}_{v_i}^{(l)}))$;

$\boldsymbol{H}_{v_i}^{(l+1)} \leftarrow \text{Normalize}(\widetilde{\boldsymbol{H}}_{v_i}^{(l)'} + \text{MLP}^{(l)}(\widetilde{\boldsymbol{H}}_{v_i}^{(l)'}))$;

// update text-rich neighbors' token embeddings

for $v_j \in \widehat{N}_{v_i}$ **do**

$\boldsymbol{H}_{v_j}^{(l)'} \leftarrow \text{Normalize}(\boldsymbol{H}_{v_j}^{(l)} + \text{MHA}^{(l)}(\boldsymbol{H}_{v_j}^{(l)}))$;

$\boldsymbol{H}_{v_j}^{(l+1)} \leftarrow \text{Normalize}(\boldsymbol{H}_{v_j}^{(l)'} + \text{MLP}^{(l)}(\boldsymbol{H}_{v_j}^{(l)'}))$;

end

end

return $\boldsymbol{h}_{v_i} \leftarrow \boldsymbol{H}_{v_i}^{(L+1)}\texttt{[CLS]}$;

end

A.2 Details of Baselines

We have 11 baselines including vanilla text/graph encoding models, GNN-cascaded Transformers, and nested Transformers.

Vanilla text/graph models:

- **MeanSAGE** [12]: This is a GNN method utilizing the mean function to aggregate information from neighbors for center node representation learning. The initial node feature vector is bag-of-words weighted by TF-IDF. The number of entries in each attribute vector is the vocabulary size of the corresponding dataset, where we keep the most representative 10000, 2000, and 5000 words for DBLP, Twitter, and Goodreads, respectively, according to the corpora size.
- **BERT** [6]: This is a benchmark PLM pretrained on two tasks: next sentence prediction and mask token prediction. For each text-rich node, we use BERT to encode its text and take the output hidden state of the `[CLS]` token as the node representation.

Homogeneous GNN-cascaded Transformers:

- **BERT+MeanSAGE** [12]: We stack BERT with MeanSAGE (*i.e.*, using the output text representation of BERT as the input node attribute vector of MeanSAGE). The BERT+MeanSAGE model is trained in an end-to-end way. (Both parameters in BERT and GNN are finetuned.) Other BERT+GNN baselines below have the same cascaded architecture.
- **BERT+MaxSAGE** [12]: MaxSAGE is a GNN method utilizing the max function for neighbor aggregation to generate center node representation.
- **BERT+GAT** [44]: GAT is a GNN method with an attention-based neighbor importance calculation, and the importance scores are utilized as weights to aggregate neighbors.

Homogeneous Nested Transformers:

- **GraphFormers** [50]: This is the state-of-the-art nested Transformer model, which has graph-based propagation and aggregation in each Transformer layer.

Since homogeneous baselines assume all nodes are associated with text information, when applying them to our datasets, we remove all textless nodes. Therefore, homogeneous baselines cannot be used for textless node classification (*i.e.*, Table 5).

Heterogeneous GNN-cascaded Transformers:

- **BERT+RGCN** [35]: RGCN is a heterogeneous GNN model. It projects neighbor representations into the same latent space according to the edge types. The initial embeddings for textless nodes are learnable vectors for baselines in this section which is the same to Heterformer.
- **BERT+HAN** [46]: HAN is a heterogeneous GNN model. It proposes a heterogeneous attention-based method to aggregate neighbor information.
- **BERT+HGT** [13]: HGT is a heterogeneous GNN model. Inspired by the Transformer architecture, it utilizes multi-head attention to aggregate neighbor information obtained by heterogeneous message passing.
- **BERT+SHGN** [27]: SHGN is a heterogeneous GNN model. Motivated by the observation that GAT is more powerful than many heterogeneous GNNs [27], it adopts GAT as the backbone with enhancements from learnable edge-type embeddings, residual connections, and normalization on the output embeddings.

Heterogeneous Nested Transformers:

- **GraphFormers++** [50]: To apply GraphFormers to heterogeneous text-rich networks, we add heterogeneous graph propagation and aggregation in its final layer. The generalized model is named GraphFormers++.

A.3 Dataset Description

A.3.1 Training. We train our model in an unsupervised way via link prediction. For each paper in DBLP, we select one neighbor paper for it and construct a positive node pair. For each POI in Twitter, we select one neighbor tweet for it to make up a positive node pair. For each book in Goodreads, one neighbor book is selected to build a positive node pair. All these node pairs are used as positive training samples. The model is then trained via in-batch negative sampling.

A.3.2 Link Prediction. The training, validation, and testing sets in this section are the same as those in Section A.3.1.

A.3.3 Node classification. The 30 categories for DBLP papers are: "Artificial intelligence", "Mathematics", "Machine learning", "Computer vision", "Computer network", "Mathematical optimization", "Pattern recognition", "Distributed computing", "Data mining", "Real-time computing", "Algorithm", "Control theory", "Discrete mathematics", "Engineering", "Electronic engineering", "Theoretical computer science", "Combinatorics", "Knowledge management", "Multimedia", "Computer security", "World Wide Web", "Human-computer interaction", "Control engineering", "Parallel computing", "Information retrieval", "Software", "Artificial neural network", "Communication channel", "Simulation", and "Natural language processing".

The 10 categories for Goodreads books are: "children", "fiction", "poetry", "young-adult", "history, historical fiction, biography", "fantasy, paranormal", "non-fiction", "mystery, thriller, crime", "comics, graphic", and "romance".

A.3.4 Node Clustering. For DBLP, since the dataset is quite large, we pick the most frequent 10 categories and randomly select 20,000 nodes for efficient evaluation. The 10 selected categories are: "Artificial intelligence", "Mathematics", "Machine learning", "Computer vision", "Computer network", "Mathematical optimization", "Pattern recognition", "Distributed computing", "Data mining", and "Real-time computing". For Goodreads, we use all 10 categories in the original dataset for clustering.

A.3.5 Embedding Visualization. In this section, we use t-SNE [42] to project node embeddings into low-dimensional spaces. Nodes are colored based on their ground-truth labels. To make the visualization clearer, we select 4 naturally separated categories for DBLP and 5 for Goodreads. The 4 selected categories for DBLP are "Mathematics", "Computer networks", "Information retrieval", and "Electronic engineering". The 5 selected categories for Goodreads are "fiction", "romance", "mystery, thriller, crime", "non-fiction", and "children".

A.4 Reproducibility Settings

A.4.1 Hyper-parameters. For a fair comparison, the training objective for all compared methods including Heterformer and baselines are the same. The hyper-parameter configuration for the node representation learning process can be found in Table 9, where "neighbor sampling" means the number of each type of neighbor sampled for the center node during learning.

In Section 4.3, we adopt a multi-layer perceptron (MLP) with 3 layers and hidden dimension 200 to be our classifier. We employ Adam optimizer [21] and early stop 10 to train the classifier. For text-rich node classification, the learning rate is set as 0.001. While for textless node classification, the learning rate is 0.01.

Table 9: Hyper-parameter configuration.

Parameter	DBLP	Twitter	Goodreads
learning rate	1e-5		
weight decay	1e-3		
adam epsilon	1e-8		
early stop	3		
textless embedding	64		
chunk k	12		
train batch size	30		
test batch size	100	300	100
PLM backbone	BERT-base-uncased		
token sequence length	32	12	64
neighbor sampling	paper:5 authors:3 venue:1	tweet:6 mention:2 tag:3,user:1	book:5, shelves:5 author:2, language code:1 publisher:1, format:1

Table 10: Case study of query-based retrieval on DBLP. Top-7 retrieved papers are shown for each method.

	Query: news recommendation with personalization
	Retrieved Paper Title
BERT	(✗) News Recommenders: Real-Time, Real-Life Experiences
	(✗) News recommender systems – Survey and roads ahead
	(✗) A Survey on Challenges and Methods in News Recommendation
	(✓) Personalized news recommendation: a review and an experimental investigation
	(✓) Interweaving Trend and User Modeling for Personalized News Recommendation
	(✗) A multi-perspective transparent approach to news recommendation
	(✗) Workshop and challenge on news recommender systems
GraphFormers	(✓) Personalized news recommendation based on links of web
	(✗) Interpreting News Recommendation Models
	(✗) Do recommendations matter?: news recommendation in real life
	(✓) Personalized News Recommendation Based on Collaborative Filtering
	(✓) LOGO: a long-short user interest integration in personalized news recommendation
	(✗) The Intricacies of Time in News Recommendation
	(✗) Workshop and challenge on news recommender systems
Heterformer	(✓) User attitudes towards news content personalization
	(✓) A system for generating personalized virtual news
	(✓) Personalized News Recommendation Based on Collaborative Filtering
	(✗) Automatic news recommendations via aggregated profiling
	(✓) Design and Deployment of a Personalized News Service
	(✓) The design and implementation of personalized news recommendation system
	(✓) Personalizing news content: An experimental study

A.5 Case Study: Paper Retrieval

To further demonstrate the capability of Heterformer in encoding text semantics, we present a case study of query-based paper retrieval on DBLP.

Settings. The models are asked to retrieve relevant papers for a user-given query based on the inner product of the encoded query embedding and the paper embedding, where the query embedding is obtained by encoding query text only with each model.

Results. Table 10 lists the top-7 papers for the query "*news recommendation with personalization*" retrieved by BERT, GraphFormers, and Heterformer. It is shown that our model can have more accurate retrieved results than both baselines. In fact, according to network homophily [28], papers on the same topics (*e.g.*, *personalization/news recommendation*) are likely to have connections (*i.e.*, become text-rich neighbors) or share similar meta-data (*i.e.*, share similar textless neighbors). While BERT can consider text information only and GraphFormers enriches text information with text-rich neighbors only, our Heterformer is capable of utilizing both text-rich neighbors and textless neighbors to complement text signals via network-empowered Transformer encoding (Section 3.1), which finally contributes to higher retrieval accuracy.