DET: A Dual-Encoding Transformer for Relational Graph Embedding

Lingbing Guo^{1,2,3,*} **, Zhuo Chen**^{1,2,3}**, Jiaoyan Chen**^{4}**,**
Qiang Zhang^{1,5}**, Huajun Chen**^{1,2,3}
^{1}College of Computer Science and Technology, Zhejiang University
^{2}Zhejiang University - Ant Group Joint Laboratory of Knowledge Graph
^{3}Donghai Laboratory
^{4}Department of Computer Science, The University of Manchester
^{5}ZJU-Hangzhou Global Scientific and Technological Innovation Center
{lbguo, chenzhuo98, qiang.zhang.cs, huanjunsir}@zju.edu.cn, jiaoyan.chen@manchester.ac.uk

Abstract

Despite recent successes in natural language processing and computer vision, Transformer faces scalability issues when processing graphs, e.g., computing the full node-to-node attention on knowledge graphs (KGs) with million of entities is still infeasible. The existing methods mitigate this problem by considering only the local neighbors, sacrificing the Transformer's ability to attend to elements at any distance. This paper proposes a new Transformer architecture called Dual-Encoding Transformer (DET). DET comprises a structural encoder to aggregate information from nearby neighbors, and a semantic encoder to seek for semantically relevant nodes. We adopt a semantic neighbor search approach inspired by multiple sequence alignment (MSA) algorithms used in biological sciences. By stacking the two encoders alternately, similar to the MSA Transformer for protein representation, our method achieves superior performance compared to state-of-the-art attention-based methods on complex relational graphs like KGs and citation networks. Additionally, DET remains competitive for smaller graphs such as molecules.

Keywords: knowledge graph completion, node classification, science for AI, Transformer

1. Introduction

Transformer has become one of the most prevalent neural models for natural language processing (NLP) (Vaswani et al., 2017; Devlin et al., 2019). The self-attention mechanism leveraged by Transformer has also been extended to graph neural networks (GNNs), e.g., GAT (Velickovic et al., 2018) and its variants (Wu et al., 2019; Sun et al., 2020; Guo et al., 2020; Chen et al., 2021b; Kim and Oh, 2021; Chen et al., 2022; Bi et al., 2022). Nevertheless, these models only consider the near (usually one-hop) neighbors, which may violate the original intention of Transformer that attends to the elements at distant positions.

Recently, Graphormer (Ying et al., 2021) starts to leverage the standard Transformer for graph representation learning and has achieved superior performance on many benchmarks. However, in its scenarios of graph property prediction, the datasets are small graphs (e.g., small molecules). The full node-to-node attention is inapplicable to large graphs with millions of connected nodes, such as knowledge graphs (KGs) or citation networks (Bordes et al., 2013; Chen et al., 2017; Sun et al., 2018; Guo et al., 2019; Vashishth et al., 2020).

In addition to many self-attention-based methods considering only local neighbors (Schlichtkrull et al., 2018; Wu et al., 2019; Ye et al., 2019; Chen et al., 2021b; Kim and Oh, 2021), some existing works introduce multi-hop (usually 2- or 3-hop) neighbors (Sun et al., 2020; Guo et al., 2020; Zhao et al., 2021; Chen et al., 2023). They concentrate on the local information and ignore the useful nodes far from the node of interest. However, capturing the remote correlations is one of the most important characteristics for Transformer.

In this paper, we propose a Dual-Encoding Transformer (DET). We consider two types of neighbors, i.e., structural neighbors and semantic neighbors. Structural neighbors are the near neighbors leveraged by most existing GNNs (Velickovic et al., 2018; Wang et al., 2018; Sun et al., 2020; Kim and Oh, 2021; Bi et al., 2022), while semantic neighbors are the non-local nodes that share similarities with the node of interest in embedding space.

Figure 1 shows the workflow of DET. For structural encoding, we use the standard Transformer layer to encode the structural neighbors. For semantic encoding, we use the semantic operator $\ominus$ to find and encode the semantic neighbors. The dual encoding ensures both local aggregation and global connection, and also enables them to benefit from each other through back propagation.

Our idea of reaching remote neighbors is inspired by multiple sequence alignment (MSA) Transformer (Rao et al., 2021). As illustrated in Figure 2, MSA Transformer queries the genetic database to fetch similar proteins as "family members" to the protein of interest. The difference is that the family members in DET (i.e., semantic neighbors) are

*Corresponding author

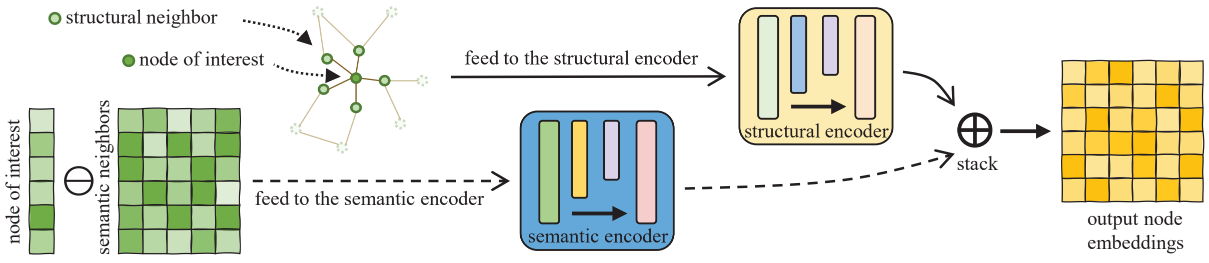

Figure 1: Overview of DET. Structural neighbors are local neighbors connected with the node of interest on the graph, while semantic neighbors are remote nodes with similar embeddings to the node of interest. The two encoders focus on encoding different aspects of neighboring information, and thus are capable of complementing each other.

obtained via self-supervised learning rather than resorting external tools.

Take Figure 2 as an example, AlphaFold (Jumper et al., 2021) uses the proteins with high MSA scores as augmented data to predict the 3D structures. Its input is not in the form of protein sequence but the alignment results produced by the MSA algorithms (Smith et al., 1981; Altschul et al., 1990). Specifically, even closely related proteins may have different lengths, encoding and non-encoding regions. Different amino acids can be also replaced with each other safely in certain circumstances. The alignment algorithms (e.g. Smith-Waterman (Smith et al., 1981)) aim to find an alignment path with maximal score to support the comparison between proteins. In the left subfigure, the result alignments have identical lengths to the query protein, but some elements in the result sequences are different from those in the query sequence. They can be insert/delete operations or other amino acids, and the scores depend on the substitution matrices like BLOSUM and PAM (Dayhoff and Eck, 1972; Henikoff and Henikoff, 1992). In our scenarios, the embeddings are the sequences to be aligned, and the similar semantics may also reside at different dimensions. To obtain such alignment paths and scores, we choose to estimate the mutual information rather than dimension-level similarity between embeddings.

The proposed DET is capable of achieving promising results in many GNN tasks: (1) DET obtains the state-of-the-art performance on entity prediction (a.k.a., KG completion) with complex knowledge graphs as input; (2) It also obtains competitive or better performance than the best-performing Transformer-based methods in node classification; (3) For conventional graph property prediction with small molecules as input, DET outperforms Graphormer (Ying et al., 2021) on PCQM4M-LSCv1 (Nakata and Shimazaki, 2017) and ZINC (Dwivedi et al., 2020). The source code and datasets are available at github.com/zjukg/DET.

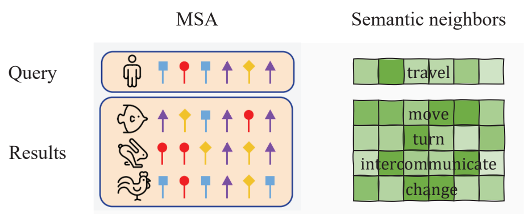

Figure 2: A comparison between MSA and semantic neighbors. The left figure is sliced from AlphaFold (Jumper et al., 2021). The right figure is an example from WordNet (Miller, 1995).

2. Related Works

Transformer Self-attention-based neural models, such as Transformer, have recently become the *de facto* choice in NLP, ranging from language modeling and machine translation (Vaswani et al., 2017; Devlin et al., 2019) to question answering (Yang et al., 2019; Yavuz et al., 2022) and sentiment analysis (Xu et al., 2019a; Cheng et al., 2021). Without loss of generality, Transformer has significant advantages over conventional sequential models like recurrent neural networks (RNNs) (Williams and Zipser, 1989; Hochreiter and Schmidhuber, 1997; Guo et al., 2019) in both scalability and efficiency.

Position embedding is one of the most important modules to Transformer. Transformer variants in different fields customize different designs in this module. For example, ViT and its followers (Dosovitskiy et al., 2021; Fan et al., 2021; Han et al., 2021) sequentially index the patches and encode the indices as 1D position embeddings. In addition to the position information, other prior knowledge can also be injected as attention biases or position embeddings into Transformer, which becomes the key to applying Transformer on graphs (Ahn et al., 2021; Chen et al., 2021a,b; Dwivedi and Bresson, 2021; Kreuzer et al., 2021; Ying et al., 2021; Bi et al., 2022; Chen et al., 2022). For example, GT (Dwivedi and Bresson, 2021) replaces the sinusoidal positional embeddings by the Laplacian eigenvectors. Graphformer (Ying et al., 2021) and its followers (Zhao et al., 2021; Chen et al., 2023) encodes centrality and shortest path distance into position embeddings. Relphormer (Bi et al., 2022) add the edge type (i.e., relation) information of KGs when encoding entity embeddings.

Non-local GNNs Some existing works also study how to capture the relationships of node of the interest to disconnected nodes (Pei et al., 2020; Yao et al., 2020; Liu et al., 2021; Min et al., 2022). Specifically, Geom-GCN (Pei et al., 2020) learns the aggregation purely based on embedding distance. Non-local-GNNs (Liu et al., 2021) computes the distance from a virtual node to all other nodes as a sorting metric to find non-local neighbors. (Yao et al., 2020; Min et al., 2022) leverage hand-crafted features to find the useful remote nodes as a complement to the local neighbors. However, most of them focus only on node classification and perform worse than the best-performing methods. Also, they usually do not distinguish between remote nodes and direct neighbors.

3. Methodology

In this section, we present the details of DET. We start from the preliminaries and then introduce the two encoding processes. Finally, we illustrate how to implement a DET.

3.1. Preliminaries

We first introduce the terminologies and notations that will be used in the following sections.

Graph We define a graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is the node set, and $\mathcal{E} = \{e_1, e_2, ..., e_m\}$ is the edge set. n and m denote the number of nodes and edges, respectively. In practice, different tasks often have more complicated graph structures. For example, molecular graphs and KGs have edge types (i.e., chemical bonds and relations). We do not discuss the details and follow the general setting to process these specific features (Chen et al., 2021b; Ying et al., 2021; Bi et al., 2022).

We consider three types of graphs in this paper. Knowledge graphs (KGs) are characterized as complex relational graphs, typically consisting of millions of nodes interconnected by thousands of relationships. The task of KG completion involves predicting missing entities from a vast set of candidate entities, making the modeling of KGs with Transformers a challenging endeavor. The graphs employed in node classification are also complex graphs, albeit with fewer labels and relationship types compared to KGs. Node classification are generally simpler than KG completion since the label space is not large. Molecules are regarded as small and simple graphs. A single molecule contains significantly fewer nodes than networks or KGs, and the types of chemical bonds are also limited. It is trivial to compute the full attention matrix on molecular graphs.

GNN and Self-attention Without loss of generality, we define a GNN as a neural network that learns a group of weights to aggregate the embeddings of the one-hop or multi-hop neighbors for the node of interest. In this sense, self-attention can be naturally treated as a GNN model. Let $\boldsymbol{Q} \in \mathbb{R}^{n\times h}$, $\boldsymbol{K} \in \mathbb{R}^{n\times h}$, $\boldsymbol{V} \in \mathbb{R}^{n\times h}$ denote the query, key, and value matrices, respectively. In this paper, they are the same node embedding matrix. h denotes the hidden layer size. Self-attention calculates the attention scores as follows:

$$\boldsymbol{A} = \textit{Softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{h}}), \quad (1)$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ records the node-to-node attention scores. We then aggregate the node embeddings with the following equation:

$$\boldsymbol{H} = \boldsymbol{A}\boldsymbol{V}, \quad (2)$$

where $\boldsymbol{H} \in \mathbb{R}^{n\times h}$ is the output embedding matrix, with each row denoting the embedding of a node.

3.2. Structural Encoding

An easy way to extend Transformer on small graphs is adding a virtual node v_c (Devlin et al., 2019) as the context node connected with all nodes in $\mathcal{G}$. Then, the output embedding for v_c can be regarded as the embedding of $\mathcal{G}$. For the large graphs like KGs or networks, we should perform self-attention only on the local (e.g., one-hop) subgraph $\mathcal{G}_i$ given the node of interest v_i. Thus, the output embedding for node v_i is

$$\mathbf{h}_i^{\text{st}} = \sum_{v_j \in \{v_i\} \cup \mathcal{N}(v_i)} \boldsymbol{A}_{cj} \mathbf{v}_j, \quad (3)$$

where $\mathbf{h}_i^{\text{st}}$ denotes the output of the structural encoder for v_i. $\boldsymbol{A}_{cj}$ denotes the attention score from the context node c to the neighbor v_j. We set $c = i$ in node classification (Zhao et al., 2021; Chen et al., 2023). $\mathcal{N}(v_i)$ is the set of local neighbors for v_i. We also accordingly add the centrality, relation type, or shortest distance path information as special position embeddings to the encoder (Chen et al., 2021b; Ying et al., 2021; Zhao et al., 2021).

3.3. Semantic Encoding

The local structural features may be insufficient for identifying a node. However, if we directly aggregate more-hop nodes, the sheer quantity of available information will overwhelm the neural network.

Table 1: The occurrence frequency of entities in FB15K-237 and WN18RR, in term of hops.

Dataset	1-hop	2-hop	3-hop	5-hop
WN18RR	2.7	8.9	30.5	483.8
FB15K-237	20.3	1781.4	64,774.9	-

Table 1 summarizes the average frequency of entities appearing as others' neighbors in different hops. We can find that the three- or more-hop neighbors of a node are shared by many others, which is why current GNNs rarely consider multi-hop (≥ 3) neighbors (Sun et al., 2020; Chen et al., 2023). To make the node of interest more distinguishable to the classifier, weighting its local neighbors is reasonable due to their less redundancy. This observation inspires us to make the first hypothesis:

Hypothesis 1. *The local neighbors are the most informative features to identify and represent the node of interest.*

Recent successes (Jumper et al., 2021; Rao et al., 2021; Baid et al., 2023) in biological science demonstrate that using the information provided by the family members greatly helps the structure prediction of proteins. Specifically, they leverage external MSA tools to collect the biological sequence alignments to protein of the interest, which enables them to capture the protein information within an evolutionary family.

If we regard the embedding as sequences with fixed lengths, then the desired remote node should have similar embeddings to the node of interest. We have illustrated this idea with Figure 2, where the semantic encoding finds the family members by estimating their mutual information rather than resorting external tools. The similar semantics may be encoded differently and reside at different dimensions. Thus, estimating the mutual information is more reliable than the dimension-level measurements, such as L1/L2 similarity.

Hypothesis 2. *The distant nodes with high mutual information scores are important features to identify and represent the node of interest.*

Our mutual information density function (van den Oord et al., 2018; Belghazi et al., 2018) $f_s : \mathbb{R}^h \times \mathbb{R}^h \to \mathbb{R}$ can be written as follows:

$$\begin{aligned} f_s(\mathbf{v}_i, \mathbf{v}_j) &= \mathbf{v}_i \ominus \mathbf{v}_j \\ &= \alpha(\hat{\mathbf{v}}_i - \hat{\mathbf{v}}_j)\boldsymbol{W}_l \\ &\quad + (1-\alpha)\mathbf{v}_i^T \mathbf{W}_m \mathbf{v}_j, \end{aligned} \tag{4}$$

where $\ominus$ is the mutual information estimator. We implement it as a combination of two terms with coefficient $\alpha = 0.1$ to control the ratio. The first term $(\mathbf{v}_i - \mathbf{v}_j)\boldsymbol{W}_s$ is L1 distance with learnable weight $\boldsymbol{W}_l \in \mathbb{R}^{h\times 1}$, while the second term $\mathbf{v}_i^{\mathsf{T}} \boldsymbol{W}_m \mathbf{v}_j$ is a standard mutual information density function with $\mathbf{W}_m \in \mathbb{R}^{h\times h}$ the product matrix. $\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_j$ denote the copy values of $\hat{\mathbf{v}}_i$ and $\hat{\mathbf{v}}_j$, respectively. We also apply vector and layer normalization in implementation for stability. Intuitively, the first term weights the precise match at every dimension, and the second term estimates the cross-dimension mutual information density.

Semantic Neighbor Fetching We seek for the semantic neighbors by empolying a self-supervised loss:

$$\begin{aligned} \mathcal{L}_{\mathsf{sn}}(v_i) = &- \mathop{\mathbb{E}}_{v_j \in \mathcal{N}(v_i)} \log\big(f_s(\mathbf{v}_i, \mathbf{v}_j)\big) \\ &+ \mathop{\mathbb{E}}_{v_k \in \mathcal{N}^-(v_i)} \log\big(f_s(\mathbf{v}_i, \mathbf{v}_k)\big). \end{aligned} \tag{5}$$

where we set $\mathcal{N}(v_i)$ as the positive example set that includes the local neighbors of v_i plus v_i itself, and $\mathcal{N}^-(v_i)$ is the negative example set that comprises of randomly-sampled distant nodes. The above loss is different from a typical contrastive loss or a mutual information maximization loss. We additionally use the local neighbors as positive examples. As stated in Hypothesis 1, local neighbors provide most informative and discriminative features. They should have high mutual information scores to the node of interest. Therefore, desired semantic neighbors are as important as the local neighbors (Hypothesis 1) and with high mutual information scores (Hypothesis 2):

Definition 1 (semantic neighbor)**.** *Semantic neighbors are those with high mutual information scores and not in the local neighbors:*

$$\mathcal{N}^{se}(v_i) = \{v_j | v_j \in \mathcal{V}\backslash\mathcal{N}(v_i), \widehat{I}(\mathbf{v}_i, \mathbf{v}_j) \geq \delta_i\}, \tag{6}$$

where $\widehat{I}(\mathbf{v}_i, \mathbf{v}_j)$ denotes our parameterized mutual information score, a normalized version of f_s. δ_i controls the threshold and can be set according to different strategies. We choose top-T candidates for each node as the semantic neighbors.

Semantic Encoding Now, we introduce how to encode the semantic neighbors with Transformer. Although we can leverage the standard dot-product attention (i.e., Equation (1)) to encode the semantic neighbor embeddings, our experiment finds that using the density function f_s (Equation (4)) to estimate the attention scores achieves better performance. It can be viewed as a mix of weighted dot-product and weighted L1 distance. We denote the corresponding attention score matrix by $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ and formulate semantic encoding as follows:

$$\begin{aligned} \mathbf{h}_i^{\mathsf{se}} &= \sum_{v_j \in \mathcal{N}^{\mathsf{se}}(v_i)} \boldsymbol{B}_{ij}\mathbf{v}_j \\ &= \sum_{v_j \in \mathcal{N}^{\mathsf{se}}(v_i)} f_s(\mathbf{v}_i, \mathbf{v}_j)\mathbf{v}_j. \end{aligned} \tag{7}$$

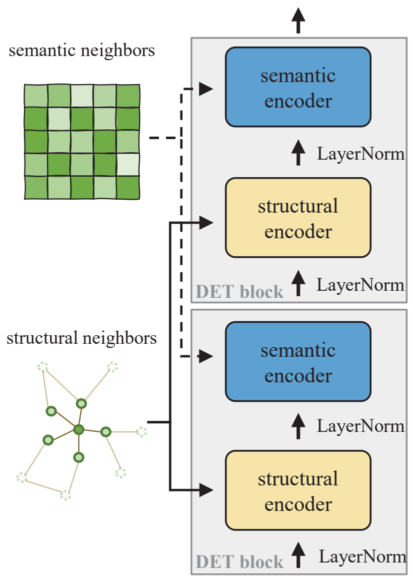

Figure 3: Example of a two-layer DET. The structural encoder and semantic encoder are stacked alternatively and feed with their neighborhood information respectively.

where $\mathcal{N}^{\text{se}}(v_i)$ denotes the set of semantic neighbors for v_i. For efficiency, we only calculate the attention scores between the semantic neighbors and node of the interest.

3.4. Dual-encoding Transformer

DET Block We first introduce how to combine the structural encoding and semantic encoding. Unlike the existing GNN method (Pei et al., 2020; Yao et al., 2020; Liu et al., 2021) that concatenate the output embeddings of different encoding layers, we build our DET block by stacking the structural encoding layer and the semantic encoding layer in an alternative fashion. As illustrated in Figure 3, DET block ensures local aggregation and global connection. It starts from a structural encoding layer whose output embeddings will contain the local neighborhood information, functioning like encoding the amino acid sequences of proteins. Then, the following semantic encoding layer will estimate the importance of the “ family members” by their local context information. By alternative stacking these two layers, these two types of encoding layers can support and enrich each other.

Implementation We illustrate the implementation of DET by Algorithm 1 and summarize the overall training process as follows: We first initialize all parameters of DET. For every few epochs, we draw semantic neighbors from the top candidates with high scores. This process can be run in parallel with the main training procedure to save time. For each DET block in each mini-batch, we stack a structural encoder and a semantic encoder and feed them different types of neighbors. Finally we jointly minimize the main task loss and semantic neighbor fetching loss, and update the parameters of DET via back-propagation.

Algorithm 1 Dual-encoding Transformer

```
1:  Input: the input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the main
    prediction loss $\mathcal{L}_{\text{main}}$, the DET model $\mathcal{M}$;
2:  Initialize all parameters;
3:  repeat
4:    Update the semantic neighbors if necessary;

5:    for each batch data $(\boldsymbol{X}, \boldsymbol{Y})$ do
6:      $\boldsymbol{H} \leftarrow \boldsymbol{X}$;
7:      for each DET block $(\mathcal{M}^{st}, \mathcal{M}^{se})$ do
8:        $\boldsymbol{H} \leftarrow \mathcal{M}^{st}(\boldsymbol{H})$ (Equation (3));
9:        $\boldsymbol{H} \leftarrow \mathcal{M}^{se}(\boldsymbol{H})$ (Equation (7));
10:     end for
11:     $\mathcal{L} \leftarrow \mathcal{L}_{\text{main}}(\boldsymbol{H}, \boldsymbol{Y}) + \mathcal{L}_{\text{sn}}(\boldsymbol{H})$;
12:     Update the parameters of $\mathcal{M}$;
13:   end for
14: until the loss $\mathcal{L}$ converges;
```

Computational Cost The design of f_s in the semantic encoder is concise and increases only a small number of parameters. Although fetching the semantic neighbors needs to iterate all nodes, we update semantic neighbors every few epochs which can be in parallel with the main procedure. Hence, the overall training time remains at the same level (please see Figure 5 in Section 5.3).

4. Experiment

We conducted extensive experiments to verify the effectiveness of DET. We are committed to releasing the source code if the paper is accepted.

4.1. KG Completion

Settings We conducted experiments on the KG completion task. The main target of KG completion is to predict the subject entity (or object entity) given an incomplete triple. We evaluated DET on two benchmark datasets FB15K-237 (Toutanova and Chen, 2015) and WN18RR (Dettmers et al., 2018), which are sampled from the real-world KGs Freebase (Bollacker et al., 2008) and WordNet (Miller, 1995), respectively. We chose the best-performing methods for comparison: the TransE-family methods TransE (Bordes et al., 2013), RotatE (Sun et al., 2019), and TuckER (Balazevic et al., 2019); and the GNN-based methods RGCN (Schlichtkrull et al., 2018), CoKE (Wang et al., 2019), CompGCN (Vashishth et al., 2020), and Relphormer (Bi et al., 2022). Specifically,

Table 2: The entity prediction results on FB15K-237 and WN18RR. The results of the baselines are extracted from (Bi et al., 2022). The best and second-best results are **boldfaced** and underlined, respectively. ↑: higher is better; ↓: lower is better. -: unavailable entry.

Model	FB15K-237				WN18RR			
	MRR↑	MR↓	Hits@1↑	Hits@10↑	MRR↑	MR↓	Hits@1↑	Hits@10↑
TransE (Bordes et al., 2013)	.310	199	.218	.495	.232	5,249	.061	.522
RotatE (Sun et al., 2019)	.338	177	.241	.533	.476	3,340	.428	.571
TuckER (Balazevic et al., 2019)	.358	-	.266	.544	.470	-	.443	.526
RGCN (Schlichtkrull et al., 2018)	.273	221	.182	.456	.402	2,719	.345	.494
CoKE (Wang et al., 2019)	.364	-	.272	.549	.484	-	.450	.553
CompGCN (Vashishth et al., 2020)	.355	197	.264	.535	.479	3,533	.443	.546
Relphormer (Bi et al., 2022)	.371	-	**.314**	.481	.495	-	.448	**.591**
DET	**.376**	**150**	.281	**.560**	**.507**	**2,255**	**.465**	.585

Table 3: The accuracy results of node classification on five benchmarks.

Model	Cora↑	Citeseer↑	Pumbed↑	Computer↑	Photo↑
GCN (Kipf and Welling, 2017)	87.33±0.38	79.43±0.26	84.86±0.19	89.65±0.52	92.70±0.20
GraphSage (Hamilton et al., 2017)	86.90±0.94	79.23±0.53	86.19±0.18	90.22±0.15	91.72±0.13
GAT (Velickovic et al., 2018)	86.29±0.53	80.13±0.62	84.40±0.05	90.78±0.13	93.87±0.11
GT (Dwivedi and Bresson, 2021)	71.84±0.62	67.38±0.76	82.11±0.39	91.18±0.17	94.74±0.13
SuperGAT (Kim and Oh, 2021)	82.70±0.60	72.50±0.80	81.30±0.50	77.44±0.26	84.53±0.32
SAN (Kreuzer et al., 2021)	74.02±1.01	70.64±0.97	86.22±0.43	89.83±0.16	94.86±0.10
Graphormer (Ying et al., 2021)	72.85±0.76	66.21±0.83	82.76±0.24	OOM	92.74±0.14
Gophormer (Zhao et al., 2021)	87.85±0.10	**80.23**±0.09	89.40±0.14	90.72±0.24	95.39±0.18
NAGphormer (Chen et al., 2023)	88.15±0.22	80.12±0.23	89.70±0.19	91.22±0.14	95.49±0.11
DET	**90.64**±0.27	80.14±0.35	**89.96**±0.20	**92.15**±0.11	**95.81**±0.13

CoKE and Relphormer also leverage Transformer to encode the structural information.

Results The main results are presented in Table 2. It is clear that DET surpassed all the baseline methods for almost all metrics. Specifically, the improvement on MR (mean rank) and MRR (mean reciprocal ranking) was most significant, which implies that DET learned better embeddings for all entities, not only for the top ones favored by Hits@1. Compared with the second-best method Relphormer, our DET has more significant advantages over the conventional triple-based methods, as it completely outperformed these methods across all metrics and datasets. on almost all metrics. We also find that the performance superiority was more significant on FB15K-237, which is a more complicated KG (has 237 different relations and $310,116$ triples) than WN18RR (has only 11 different relations and $93,003$ triples).

4.2. Node Classification

Settings Node classification aims to predict the labels of nodes in a single graph based on the node features and their relationships. We evaluated DET on five well-used benchmarks, i.e., Cora, CiteSeer, PubMed, Amazon Computer, and Amazon Photo (McAuley et al., 2015; Yang et al., 2016). We selected the following methods for comparison: the attention-based GAT (Velickovic et al., 2018), GT (Dwivedi and Bresson, 2021), SuperGAT (Kim and Oh, 2021), SAN (Kreuzer et al., 2021), Graphormer (Ying et al., 2021), Gophormer (Zhao et al., 2021) and NAGphormer (Chen et al., 2023); and the GNN-based GCN (Kipf and Welling, 2017) and GraphSage (Hamilton et al., 2017).

Results The results are presented in Table 3. On four of five datasets, DET significantly outperformed all baseline methods, including the Transformer-based ones and those considering multi-hop or non-local nodes. The unanimously promising results on all datasets empirically verified the effectiveness of the proposed semantic encoding. The performance of DET on Citeseer was slightly below that of Gophormer (Zhao et al., 2021), a multi-hop method that adapts Graphormer (Ying et al., 2021) to node classification. However, we believe that there is no contradiction to incorporate Gophormer as the structural encoder into DET to obtain a more powerful model.

4.3. Graph Property Prediction

Settings Graph property prediction aims to predict the properties of a set of small graphs. We evaluated DET on PCQM4M-LSCv1 (Hu et al., 2021) and ZINC (Dwivedi et al., 2020). The former is used in the recent Open Graph Benchmark Large-Scale Challenge, while the latter is a popular dataset in molecular graph representation learn-

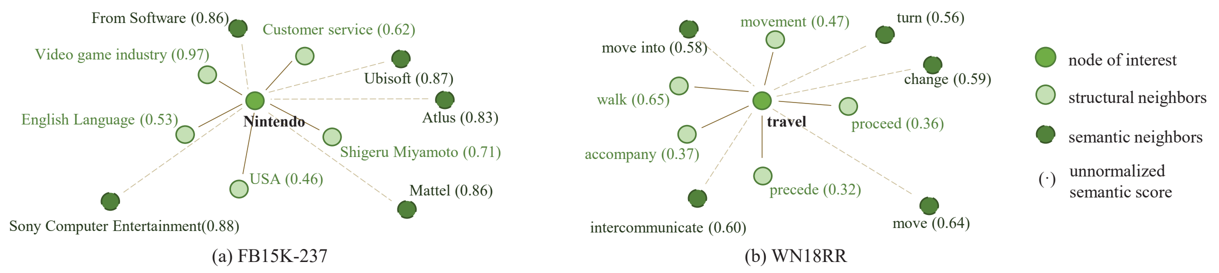

Figure 4: Examples of the semantic attention scores to different types of neighbors.

Table 4: Graph property prediction results on PCQM4M-LSCv1.

Model	#param.	train MAE↓	validate MAE↓
GCN (Kipf and Welling, 2017)	2.0M	0.1318	0.1691
GraphSage (Hamilton et al., 2017)	-	-	-
GIN (Xu et al., 2019b)	3.8M	0.1203	0.1537
GT (Dwivedi and Bresson, 2021)	83.2M	0.0955	0.1408
Graphormer (Ying et al., 2021)	47.1M	<u>0.0582</u>	<u>0.1234</u>
DET	47.1M	**0.0546**	**0.1212**

Table 5: Graph property prediction results on ZINC.

Model	#param.	test MAE↓
GCN (Kipf and Welling, 2017)	505,079	0.367
GraphSage (Hamilton et al., 2017)	505,341	0.398
GIN (Xu et al., 2019b)	509,549	0.526
GAT (Velickovic et al., 2018)	531,345	0.384
SAN (Kreuzer et al., 2021)	508,577	0.139
GT (Dwivedi and Bresson, 2021)	588,929	0.226
Graphormer (Ying et al., 2021)	489,321	<u>0.122</u>
DET	489,562	**0.113**

ing. Due to the number of nodes in each graph (molecule) is very small (usually less than 50), we directly performed attention operations on each graph. Therefore, semantic neighbor fetching module was removed. The main target of this experiment is to investigate whether semantic encoding is still helpful when all nodes can easily reach each other. We set Graphormer (Ying et al., 2021) as our structural encoder, and also selected the following methods for comparison: the attention-based methods GAT (Velickovic et al., 2018), GT (Dwivedi and Bresson, 2021), and SAN (Kreuzer et al., 2021); the GNN-based methods GCN (Kipf and Welling, 2017), GraphSage (Hamilton et al., 2017), and GIN (Xu et al., 2019b).

Results Table 4 and Table 5 summarize the experimental results measured by mean average error (MAE) on the two datasets. Due to the inaccessibility of the testing data on PCQM4M-LSCv1, we reported the MAE results on training and validation sets. Overall, DET outperformed all the baseline methods on both two datasets. Compared with Graphormer that encoded only structural information, DET had 6.2% and 7.4% MAE decline on PCQM4M-LSCv1 and ZINC, respectively. Meanwhile, the number of model parameters maintained the same level to that of baselines. Therefore, leveraging semantic neighbors was also helpful for encoding small graphs.

Overall, DET achieved competitive performance on all three types of tasks, empirically verifying its effectiveness and generality in modeling graphs.

5. Further Analysis

In this section, we delve deeper into DET and gain a more comprehensive understanding of its capabilities and performance.

5.1. Ablation Study

We conducted ablation studies to verify the effectiveness of each module in DET. We used six datasets in different tasks and present the results in Table 6. We removed the modules from DET step-by-step while keeping identical hyper-parameter settings throughout the experiments.

Semantic Neighbor Fetching The semantic neighbor fetching loss was undoubtedly important to DET. No matter if combining two encoders or not, integrating with the semantic fetching module always had better performance. The improvement was most notable on PubMed, where it yielded 3.7% and 3.8% of accuracy increases, respectively. The mean rank results on WN18RR also got worse without the fetching loss.

Semantic Encoder If we do not consider the semantic neighbor fetching loss (i.e., regarding the semantic encoder as a special attention layer), is the semantic encoder itself still useful to DET? It depends. For Cora, PubMed, and WN18RR, when we did not employ the fetching loss, DET with the semantic encoder performed worse than DET without the semantic encoder. But we observe that the situation was reversed on CiteSeer and FB15K-237. We believe that the semantic encoder may have its pros and cons compared with the standard dot-product attention layer on different datasets. The

Table 6: Ablation studies on different graphs. St. and Se. are the abbreviations of Structural and Semantic.

St. encoder	Se. encoder	Fetching loss	ZINC MAE↓	Cora Accuracy↑	CiteSeer Accuracy↑	PubMed Accuracy↑	FB15K-237 Mean Rank↓	WN18RR Mean Rank↓
√	√	√	-	**88.15**	**80.12**	**89.70**	**150**	**2,255**
√	√		**0.113**	86.21	80.07	85.63	151	2,305
√			0.122	87.79	77.95	87.97	158	2,268
	√	√	-	87.94	79.94	89.04	-	-
	√		0.515	86.64	77.60	84.20	-	-

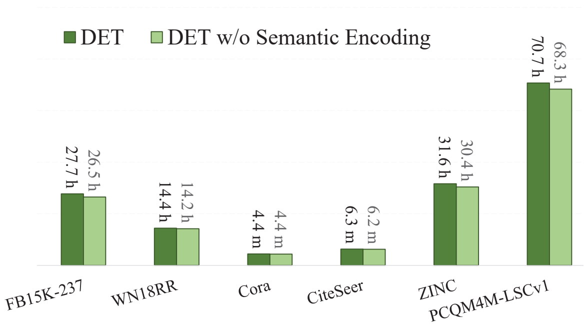

Figure 5: The training time (hours/minutes) of DET and DET w/o semantic encoding on six datasets.

semantic fetching loss is who endows the semantic encoder with the characteristic.

On the ZINC dataset, where the model performed attention operations on the whole graph, the semantic encoder was capable of estimating the semantic scores of remote nodes without the help of the fetching loss. Therefore, we can see that the dual-encoding version of DET significantly outperformed the structural encoder only version. Overall, the effectiveness of the semantic encoder is conditioned: it must get in touch with the remote nodes.

Structural Encoder The structural encoder also has merits. From the results of the 3-rd and 5-th rows in Table 6, we find that it had better performance than the semantic encoder on all datasets except CiteSeer. We also noticed that only using the semantic encoder had the worst MAE on ZINC, due to the absence of all structural information.

5.2. How does the Semantic Encoder Help the Structural Encoder?

It is worth exploring how the semantic encoder affects the structural encoder. In Figure 4, we illustrate two examples sampled from FB15K-237 and WN18RR, respectively.

The semantic scores for the structural neighbors were in line with human intuition. In the left figure, the entity *USA* has a low score although it is directly connected to *Nintendo* by relation *service_location*. Also, the verb *precede* and *accompany* obtain relatively low scores in the right figure. These neighbors are not very related to the entities of interest from the human perspective. By contrast, some one-hop neighbors get high semantic scores, e.g., the well-known director *Shigeru Miyamo* of *Nintendo* in FB15K-237 and the verb *walk* in WN18. They are the more informative entities.

For the semantic neighbors, we can see that the exploited remote neighbors are closely related to the entity of interest. For example, *Atlus* is an important game developer to *Nintendo*. Aggregating such information may be helpful when the model is asked to predict the games related to *Nintendo*. For the verb *travel* in WN18RR, *move* also shares many key features with it. We also analysed the effects of the semantic encoder to the structural encoder during the training phase.

5.3. Computational Cost

We conducted experiments to investigate the actual computational cost of DET. We employed a 32GB V100 GPU to train DET in comparison with DET w/o semantic encoding. We used same parameter settings for these two methods. The average training time are presented in Figure 5. It is clear that incorporating the semantic encoder only had a small increase in the training time on all six datasets. Particularly, the training time on Cora and CiteSeer was almost identical for two methods.

5.4. Improvement on Different Relations

We conducted an analysis of DET's performance based on relation types on the WN18RR dataset. WN18RR comprises 3,034 validation examples across 11 different relations, The mean reciprocal rank (MRR) results for DET compared with DET without the semantic encoder are listed in Table 7.

Overall, the proposed DET outperformed DET without the semantic encoder for most relations. However, we observed minor improvements and even negative gains in *verb group* and *derivationally related form*, respectively. We attribute this to these examples already possessing sufficient context information from local neighbors.

5.5. Different Semantic Operators

We conducted experiments to investigate the performance of using other semantic operators. Specifically, we evaluated the following variants:

Table 7: The MRR results of DET w/o semantic encoder and DET, in terms of relation types on WN18RR.

Relation	Count	DET w/o Se. encoder	DET	Gain
hypernym	1,174	.144	.201	39.6%
derivationally related form	1,078	.947	.945	-0.2%
member meronym	273	.237	.338	42.6%
has part	154	.200	.247	23.5%
instance hypernym	107	.302	.340	12.6%
synset domain topic of	105	.350	.415	18.6%
verb group	43	.930	.931	0.1%
also see	41	.585	.602	2.9%
member of domain region	34	.201	.336	57.2%
member of domain usage	22	.373	.451	18.2%
similar to	3	1.000	1.000	0.0%

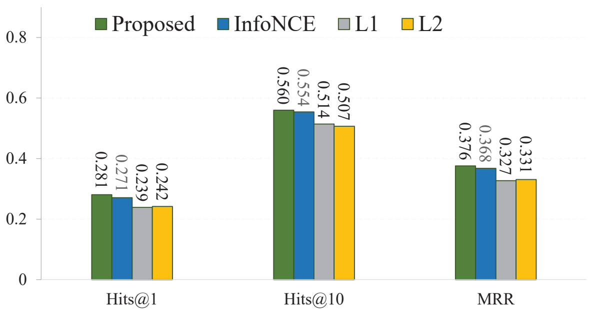

Figure 6: The performance of DET with different semantic operators on FB15K-237.

(1) InfoNCE: we replaced the proposed mutual-information-based operator (Equation (4)) with an identical estimator to that in InfoNCE (van den Oord et al., 2018); (2) L1: we replaced the proposed operator with L1 measurement; (3) L2: we replaced the proposed operator with L2 measurement. The results on FB15K-237 are presented in Figure 6. DET with L1/L2 estimators had the worst performance. Our DET significantly outperformed all the variants, including InfoNCE, demonstrating the advantages of the proposed operator.

6. Conclusion and Limitations

In this paper, we propose DET which achieves state-of-the-art performance across 9 different datasets. In DET, the structural encoder aggregates local nodes while the semantic encoder seeks for the remote nodes. Inspired by recent advances in biological sciences, DET finds the semantic neighbors with a mutual-information-based operator and stacks the two encoders alternatively. We hope DET can bring more insights and inspirations in developing new Transformer architectures. Currently, the main limitation of DET is the additional computation in ranking semantic neighbors. We plan to investigate more flexible solutions in future.

Acknowledgment

We would like to thank all anonymous reviewers for their insightful and invaluable comments. This work is funded by New Generation AI Development Plan for 2030 of China (2023ZD0120802), National Natural Science Foundation of China (NSFC62302433, USFCU23A20496), and Zhejiang Provincial Natural Science Foundation of China (No.LQ24F020007).

Sungsoo Ahn, Binghong Chen, Tianzhe Wang, and Le Song. 2021. Spanning tree-based graph generation for molecules. In *ICLR*.

Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. 1990. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410.

Gunjan Baid, Daniel E Cook, Kishwar Shafin, Taedong Yun, Felipe Llinares-López, Quentin Berthet, Anastasiya Belyaeva, Armin Töpfer, Aaron M Wenger, William J Rowell, et al. 2023. Deepconsensus improves the accuracy of sequences with a gap-aware sequence transformer. *Nature Biotechnology*, 41(2):232–238.

Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. In *EMNLP-IJCNLP*, pages 5184–5193.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, R. Devon Hjelm, and Aaron C. Courville. 2018. Mutual information neural estimation. In *ICML*, pages 530–539.

Zhen Bi, Siyuan Cheng, Ningyu Zhang, Xiaozhuan Liang, Feiyu Xiong, and Huajun Chen. 2022. Relphormer: Relational graph transformer for knowledge graph representation. *arXiv preprint arXiv:2205.10852*.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795.

Jianwen Chen, Shuangjia Zheng, Ying Song, Jiahua Rao, and Yuedong Yang. 2021a. Learning attributed graph representations with communicative message passing transformer. In *IJCAI*.

Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. 2023. Nagphormer: A tokenized graph transformer for node classification in large graphs. In *ICLR*. OpenReview.net.

Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2017. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *IJCAI*, pages 1511–1517.

Sanxing Chen, Xiaodong Liu, Jianfeng Gao, Jian Jiao, Ruofei Zhang, and Yangfeng Ji. 2021b. Hitter: Hierarchical transformers for knowledge graph embeddings. In *EMNLP*, pages 10395–10407.

Zhuo Chen, Jiaoyan Chen, Wen Zhang, Lingbing Guo, Yin Fang, Yufeng Huang, Yuxia Geng, Jeff Z Pan, Wenting Song, and Huajun Chen. 2022. Meaformer: Multi-modal entity alignment transformer for meta modality hybrid. *arXiv preprint arXiv:2212.14454*.

Junyan Cheng, Iordanis Fostiropoulos, Barry Boehm, and Mohammad Soleymani. 2021. Multimodal phased transformer for sentiment analysis. In *EMNLP*, pages 2447–2458.

Margaret O Dayhoff and Richard V Eck. 1972. *Atlas of protein sequence and structure*. National Biomedical Research Foundation.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D knowledge graph embeddings. In *AAAI*, pages 1811–1818.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*.

Vijay Prakash Dwivedi and Xavier Bresson. 2021. A generalization of transformer networks to graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2020. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*.

Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. 2021. Multiscale vision transformers. In *ICCV*, pages 6824–6835.

Lingbing Guo, Zequn Sun, and Wei Hu. 2019. Learning to exploit long-term relational dependencies in knowledge graphs. In *ICML*, pages 2505–2514.

Lingbing Guo, Weiqing Wang, Zequn Sun, Chenghao Liu, and Wei Hu. 2020. Decentralized knowledge graph representation learning. *CoRR*, abs/2010.08114.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*, pages 1024–1034.

Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. 2021. Transformer in transformer. *NeurIPS*, 34.

Steven Henikoff and Jorja G Henikoff. 1992. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919.

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

Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. 2021. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andrew J Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589.

Dongkwan Kim and Alice H. Oh. 2021. How to find your friendly neighborhood: Graph attention design with self-supervision. In *ICLR*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Devin Kreuzer, Dominique Beaini, William Hamilton, Vincent Létourneau, and Prudencio Tossou.

2021. Rethinking graph transformers with spectral attention. *arXiv preprint arXiv:2106.03893*.

Meng Liu, Zhengyang Wang, and Shuiwang Ji. 2021. Non-local graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52. ACM.

George A. Miller. 1995. WordNet: An electronic lexical database. *Communications of the ACM*, 38.

Erxue Min, Yu Rong, Tingyang Xu, Yatao Bian, Peilin Zhao, Junzhou Huang, Da Luo, Kangyi Lin, and Sophia Ananiadou. 2022. Masked transformer for neighhourhood-aware click-through rate prediction. *arXiv preprint arXiv:2201.13311*.

Maho Nakata and Tomomi Shimazaki. 2017. Pubchemqc project: a large-scale first-principles electronic structure database for data-driven chemistry. *Journal of chemical information and modeling*, 57(6):1300–1308.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. In *ICLR*. OpenReview.net.

Roshan Rao, Jason Liu, Robert Verkuil, Joshua Meier, John F. Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. 2021. Msa transformer. *bioRxiv*.

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*, pages 593–607.

Temple F Smith, Michael S Waterman, et al. 1981. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197.

Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. 2018. Bootstrapping entity alignment with knowledge graph embedding. In *IJCAI*, pages 4396–4402.

Zequn Sun, Chengming Wang, Wei Hu, Muhao Chen, Jian Dai, Wei Zhang, and Yuzhong Qu. 2020. Knowledge graph alignment network with gated multi-hop neighborhood aggregation. In *AAAI*, pages 222–229.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *CVSC*, pages 57–66.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. 2020. Composition-based multi-relational graph convolutional networks. In *ICLR*.

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 *NIPS*, pages 5998–6008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

Quan Wang, Pingping Huang, Haifeng Wang, Songtai Dai, Wenbin Jiang, Jing Liu, Yajuan Lyu, Yong Zhu, and Hua Wu. 2019. Coke: Contextualized knowledge graph embedding. *arXiv preprint arXiv:1911.02168*.

Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *EMNLP*, pages 349–357.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1:243–248.

Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, Rui Yan, and Dongyan Zhao. 2019. Relation-aware entity alignment for heterogeneous knowledge graphs. In *IJCAI*, pages 5278–5284.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2019a. BERT post-training for review reading comprehension and aspect-based sentiment analysis. In *NAACL-HLT*, pages 2324–2335.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019b. How powerful are graph neural networks? In *ICLR*.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. In *NAACL-HLT*, pages 72–77.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*, volume 48, pages 40–48.

Shaowei Yao, Tianming Wang, and Xiaojun Wan. 2020. Heterogeneous graph transformer for graph-to-sequence learning. In *ACL*, pages 7145–7154.

Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, Nitish Shirish Keskar, and Caiming Xiong. 2022. Modeling multi-hop question answering as single sequence prediction. In *ACL*, pages 974–990.

Rui Ye, Xin Li, Yujie Fang, Hongyu Zang, and Mingzhong Wang. 2019. A vectorized relational graph convolutional network for multi-relational network alignment. In *IJCAI*, pages 4135–4141.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? In *Thirty-Fifth Conference on Neural Information Processing Systems*.

Jianan Zhao, Chaozhuo Li, Qianlong Wen, Yiqi Wang, Yuming Liu, Hao Sun, Xing Xie, and Yanfang Ye. 2021. Gophormer: Ego-graph transformer for node classification. *CoRR*, abs/2110.13094.