Structure Pretraining and Prompt Tuning for Knowledge Graph Transfer

Wen Zhang*
zhang.wen@zju.edu.cn
Zhejiang University
Hangzhou, China

Yushan Zhu*
yushanzhu@zju.edu.cn
Zhejiang University
Hangzhou, China

Mingyang Chen
mingyangchen@zju.edu.cn
Zhejiang University
Hangzhou, China

Yuxia Geng
gengyx@zju.edu.cn
Zhejiang University
Hangzhou, China

Yufeng Huang
huangyufeng@zju.edu.cn
Zhejiang University
Hangzhou, China

Yajing Xu
22151361@zju.edu.cn
Zhejiang University
Hangzhou, China

Wenting Song
songwenting@huawei.com
Huawei Technologies Co., Ltd
Xi'an, China

Huajun Chen†
huajunsir@zju.edu.cn
Zhejiang University
Donghai Laboratory
Alibaba-Zhejiang University Joint
Institute of Frontier Technology

ABSTRACT

Knowledge graphs (KG) are essential background knowledge providers in many tasks. When designing models for KG-related tasks, one of the key tasks is to devise the Knowledge Representation and Fusion (KRF) module that learns the representation of elements from KGs and fuses them with task representations. While due to the difference of KGs and perspectives to be considered during fusion across tasks, duplicate and ad hoc KRF modules design are conducted among tasks. In this paper, we propose a novel knowledge graph pretraining model KGTransformer that could serve as a uniform KRF module in diverse KG-related tasks. We pretrain KGTransformer with three self-supervised tasks with sampled sub-graphs as input. For utilization, we propose a general prompt-tuning mechanism regarding task data as a triple prompt to allow flexible interactions between task KGs and task data. We evaluate pretrained KGTransformer on three tasks, triple classification, zero-shot image classification, and question answering. KGTransformer consistently achieves better results than specifically designed task models. Through experiments, we justify that the pretrained KGTransformer could be used off the shelf as a general and effective KRF module across KG-related tasks. The code and datasets are available at https://github.com/zjukg/KGTransformer.

*Both authors contributed equally to this research.
†Corresponding author.

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583301

CCS CONCEPTS

• **Computing methodologies** → **Knowledge representation and reasoning**.

KEYWORDS

knowledge graph, pretrain and fine-tune, knowledge transfer

ACM Reference Format:
Wen Zhang, Yushan Zhu, Mingyang Chen, Yuxia Geng, Yufeng Huang, Yajing Xu, Wenting Song, and Huajun Chen. 2023. Structure Pretraining and Prompt Tuning for Knowledge Graph Transfer. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3543507.3583301

1 INTRODUCTION

Knowledge Graphs (KG) representing facts as triples in the form of *(head entity, relation, tail entity)*, abbreviated as *(h,r,t)*, is a common way of storing knowledge in the world, such as *(Earth, location, inner Solar System)*[1]. In recent years, many large-scale KGs including Wididata [54], YAGO [39] and NELL [4] have been constructed and applied as background knowledge providers in machine learning tasks, such as question answering [67], image classification [60], visual reasoning [13], etc.

When designing models for KG-related tasks, one of the key tasks is to devise the Knowledge Representation and Fusion (KRF) module that learns representation of elements from KGs and fuses them with task representations. As shown in Figure 1, the knowledge graph completion model RotatE [46] represents knowledge in KGs by learning embedding for entities and relations in the complex value space and calculates the truth value of triples through a score function. The zero-shot image classification model GCNZ [60] uses a graph convolutional network and ResNet [17] to learn representations of KGs and images, respectively, and fuses them through

[1]Example from Wididata.

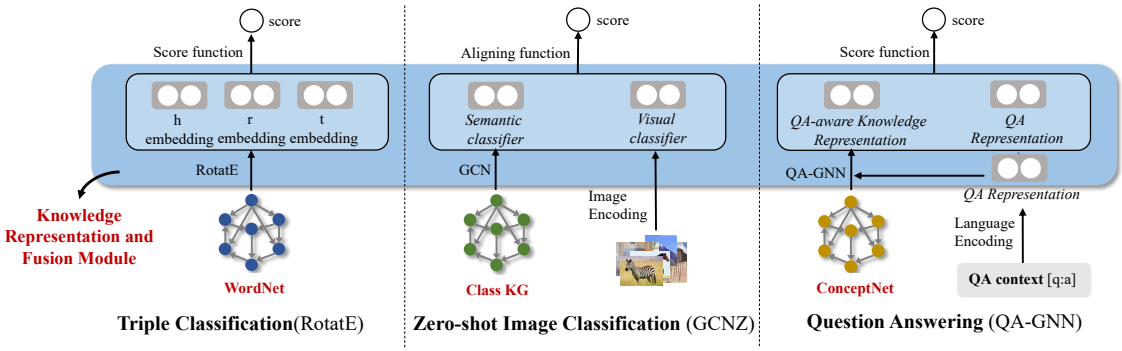

Figure 1: Example of models for KG-related tasks supported by different KGs.

aligning results from classifiers on KG representation and image representation. The knowledge-based question answering model QA-GNN [67] first encodes the query-KG graph and fuses the representation with query representation for prediction. **Due to the difference of KGs and perspectives to be considered, the KRF modules are different in KG-related task models, leaving duplicate works for ad hoc KRF module design.**

To solve this problem, the pre-trained KG model is proposed to learn universal embeddings of entities and relations that could be applied in many tasks [43, 70]. These embeddings are supposed to contain entity similarities [3], hierarchies [71] and relationships [50] that could be used for recommender system [58], entity alignment [47], question answering [20], etc., helping implicitly access knowledge in KGs. However, directly applying embeddings from a pre-trained model is insufficient and meets two challenges: (1) The first challenge is if task KGs contain different entities and relations to KGs used for pre-training, the embedding-based method could not transfer valuable information to downstream task model since the embeddings are missing. (2) The second challenge is essential interaction and fusion between KGs and task data is missing, leaving designing a fusion module as part of the work for downstream task model devising. **Thus embedding-based KG models are not ideal solutions for KRF module across KG-related tasks**.

In this work, we solve the first challenge by pre-training KG structures and transforming parameters unrelated to specific entities and relations into tasks. We solve the second challenge by a prompt tuning mechanism to enable uniform and flexible fusion between KGs and task data. As a result, **we propose a novel model KGTransformer, constructed by multiple *KGTransformer layers* with a sequence of triples as input**. It allows diverse but constrained interactions between elements in the sequence according to a neighborship matrix between elements. We propose a sub-graph pre-training method with three self-supervised tasks, i.e. Masked Entity Modeling, Masked Relation Modeling, and Entity Pair Modeling. This enables KGTransformer to capture graph structures and semantics universally existing in KGs. The set of parameters $\theta_{\mathcal{M}}$ in KGTransformer layers helps transfer graph structural knowledge that is unrelated to specific entities and relations from pre-training KGs to tasks KGs. For applying KGTransformer in KG-related tasks, we propose a general prompt tuning mechanism forming each task sample as a prompt concatenated at the end of the task KG sequence to manipulate the performance of KGTransformer.

During experiments, we pre-train KGTransformer on a hybrid dataset consisting of three benchmark datasets with diverse structures. Then we apply it to three KG-related tasks of different modalities, including one in-KG task triple classification, and two out-of-KG tasks, zero-shot image classification and question answering. We compare KGTransformer to recently proposed and specifically designed methods of these three tasks. Results show that KGTransformer performs better, proving the effectiveness of the pre-trained KGTransformer as a uniform KRF module for KG-related task models. More importantly, we prove that simply using pre-trained KGTransformer layers off the shelf with $\theta_{\mathcal{M}}$ frozen in tasks is enough to get promising results. In summary, our contributions are

- We propose the novel KGTransformer, which could capture graph structural knowledge that is transferable across KGs by being pre-trained with self-supervised tasks.
- We propose a simple yet effective prompt tuning mechanism to apply KGTransformer off the shelf to enable flexibly fusing knowledge in KGs to task data.
- We show that the pre-trained KGTransformer has the capability of transferring KG structure knowledge across KGs and is general enough to be applied in various tasks, supported by experiments on three KG-related tasks.

2 RELATED WORKS

2.1 Knowledge Graph Representation Methods

KG representation methods encode information in KGs through parameters and functions in models. They could recover the graph structures and capture semantics between entities and relations.

Embedding-based methods [3, 30, 46, 50] learn embeddings of relations and entities and model the truth value of triples through a score function with embeddings as inputs. After training, these embeddings could implicitly capture the similarities [3], hierarchies [71], relationships [46], and axioms [69] between elements in KGs, thus could be applied as general representations of elements in many tasks to transfer semantics learned from KGs to tasks.

Structure-based methods [29, 36, 49, 64] learn an encoder with sub-graph with node features as input, and a decoder for specific tasks. Typical graph-based methods are independent of embeddings of entities and thus are entity agnostic that could be applied to inductive tasks. For example, GraIL [49] generates two-dimensional node features for each entity in a sub-graph, encodes the sub-graph

through a graph-neural-network-liked module to get the graph-specific representation of entities, and uses the entity and relation representations from encoder for link prediction.

Hybrid-based methods [27, 42, 51, 62] learn encoder-decoder together with embeddings of entities and relations. Specifically, embeddings of entities rather than pre-defined node features are used for graph neural network (GNN) encoder. Our KGTransformer is also a hybrid-based method regarding KG representation. While different from existing hybrid-based methods, apart from one-hop neighbors of entities, KGTransformer also aggregates information from connected relations and two-hop neighbors.

Transformer-based methods [5, 6, 19, 23, 26, 33, 45] encode structural or semantic information in KGs through attention mechanisms [52]. GraphWriter [23] proposes a graph Transformer to solve the problems of non-hierarchical nature, long-distance dependencies, and structural variation for generating text from KGs. HittER [5] proposes a hierarchical Transformer to jointly learn entity-relation composition and relation contexts in KGs. kgTransformer [33] is pretrained on KGs by formulating logical queries as masked prediction and applied to complex query tasks. GHT [45] is proposed for temporal KGs reasoning and it utilizes Transformers to capture the instantaneous structure and evolution information. MKGformer [6] designs a hybrid Transformer that integrates visual and text representation for multimodal KGs completion tasks. TET [19] performs the entity typing task by encoding KGs' graph structure through a local Transformer, a global Transformer, and a context Transformer.

2.2 Knowledge Graph Fusion Methods

Many knowledge graphs are proposed in various domains and are used to support different downstream tasks. As a kind of important side information, researchers design sophisticated ways to incorporate knowledge graphs into their task-specific methods, and we term them knowledge graph fusion methods.

Some works [14, 16, 31, 41, 55, 65] use knowledge in KGs in an *out-of-the-box* manner. Specifically, these methods usually conduct representation learning on KGs in advance and use trained representations of entities and relations as the input of downstream models or for ensembling; such pretrained embedding can be frozen or fine-tuned during training downstream models. For example, OntoZSL [14] uses trained TransE [3] embeddings for an ontological schema to model the prior knowledge for zero-shot learning. EmbedKGQA[41] uses trained ComplEx [50] embeddings to support the answer selection in question-answering systems.

Furthermore, other works [40, 56–58, 67] fuse knowledge in an *end-to-end* manner. More precisely, these methods design learnable KG encoder modules and train them with downstream models. For instance, RippleNet [56] encodes the sub-graph preference propagation in a KG and predicts the user engaging in recommender systems. KGAT [58] employs an attention mechanism to propagate representations from neighbors for nodes in a KG containing users, items, and attributes for the recommendation. QA-GNN [67] uses a GNN to encode QA-pair-related sub-graphs for joint representation of QA text information and KG information. Our KGTransformer is also an end-to-end KG fusion method.

2.3 Pretraining Methods

The great success of pretrained language models (PLMs) [7, 25, 34, 37] has shifted the paradigm of natural language processing from *fully supervised learning* to *pretraining and fine-tuning* [32]. PLMs with fixed architectures are pretrained on large-scale corpora to learn robust general-purpose features of a language. By adding additional parameters, PLMs could be quickly adapted to downstream tasks optimized according to the task-specific objective function. The paradigm of pretraining and fine-tuning is widely adopted in other areas, such as image processing [2], multimodal processing [11, 44], and table processing [18, 61]. Since general knowledge such as axioms also exists in knowledge graphs, this inspires us to explore pretrained knowledge graph models so as to be quickly adapted to KG-related tasks.

Among pretraining methods, the auto-regressive *Transformer* [52] with multi-head self-attention module is the key part of pretrained language models [7, 34]. It has been shown to be very powerful in processing sequential data such as text. Even for non-sequential data such as image [35], video [1] and graph [68], Transformer with specific designs, such as task data serialization [66], position encoding [8], structure encoding [68], etc., could also perform reasonably well. To adapt Transformer to graph data, Ying et al. [68] propose Graphomer built on the standard Transformer with several structural encoding methods, including spatial encoding, edge encoding, and centrality encoding. In the works adapting Transformer to encode KGs, concatenating surface form or description of the head entity, relation, and tail entity to serialize triple are commonly applied [21, 66], while they pay more attention to the text of elements than graph structures. In contrast to encoding graph structure as it is or allowing connections between all elements, the KGTransformer layer we propose in this paper allows constrained connections between parts of elements that are not directly connected in KG.

During finetuning, the *prompt tuning* methods on PLM introduce a flexible and effective way to manipulate the behavior of PLMs [12, 24, 38] by adding an appropriate prompt related to the tasks. They inspire us to adapt pretrained knowledge graph models by adding the task sample as a prompt.

3 METHODOLOGY

In this section, we introduce the model structure of KGTransformer (3.1), how to pretrain KGTransformer with graph sampling strategies and pretraining tasks (3.2), and the prompt-tuning mechanism for KG-related tasks (3.3).

3.1 KGTransformer

KGTransformer is constructed by multiple KGTransformer layer built on traditional Transformer layer [52]. We adapt Transformer layer to knowledge graph with a set of triples $\mathcal{T}_{in} = \{(h_i, r_i, t_i)|i \in [1, k]\}$ as input and output the representation of each element that could be used for prediction, where k is the number of triples.

Given a $\mathcal{T}_{in}$, we first make $\mathcal{T}_{in}$ as a sequence of tokens

$$s_{in} = [[B], h_1, r_1, t_1, [S], h_2, r_2, t_2, [S], ..., h_k, r_k, t_k, [S]] \quad (1)$$

where $[B]$ and $[S]$ are special tokens indicating the beginning of sequence and separation between triples respectively.

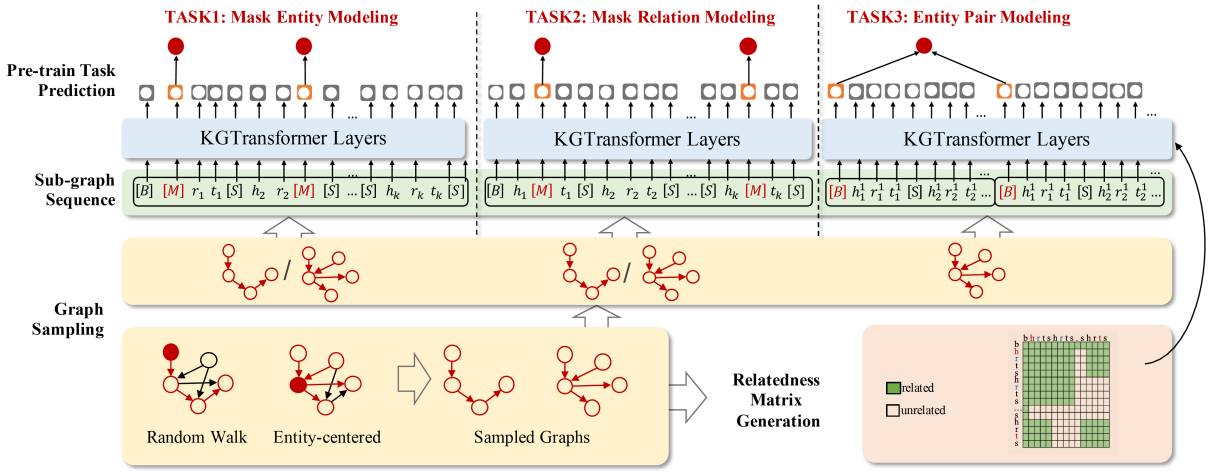

Figure 2: Overview of sub-graph pretraining of KGTransformer.

Then we generate a matrix $M \in \mathbb{R}^{|s_{in}| \times |s_{in}|}$ to indicate the neighborships between triples that are not explicitly modeled in the sequence s_{in}, where $|s_{in}|$ is the length of s_{in}.

$$M_{ij} = \begin{cases} 1 \text{ if } i = 1 \text{ or } j = 1 \\ 1 \text{ if } trp(i) \cap trp(j) \neq \emptyset \\ 0 \text{ otherwise} \end{cases}, \tag{2}$$

$$trp(n) = \{h_p, r_p, t_p\} \text{ where } p = \lfloor (n-2)/4 \rfloor + 1 \tag{3}$$

With s_{in} and M, we input s_{in} to the KGTransformer and use M to constrain the interactions between elements in s_{in}. Specifically, similar to traditional transformer, KGTransformer layer includes a self-attention module and a position-wise feed-forward network. Suppose the input of self-attention module is $H = [s_1^\top, ..., s_n^\top]^\top \in \mathbb{R}^{n \times d}$ with the ith row as the d dimensional hidden state for the ith element in the sequence. The self-attention operation $Attn()$ is

$$Q = HW_Q,\ K = HW_K,\ V = HW_V, \tag{4}$$

$$A = \frac{QK^\top \odot M}{\sqrt{d_K}} + (1 - M) * \delta, \tag{5}$$

$$Attn(H) = softmax(A)V, \tag{6}$$

where $W_Q \in \mathbb{R}^{d \times d_Q}$,$W_K \in \mathbb{R}^{d \times d_K}$,$W_V \in \mathbb{R}^{d \times d_V}$ is the projection matrix to generate the query, key, and value representation of H; $\odot$ represents element-wise multiplication; A is the attention matrix with A_{ij} capturing the similarity between the query representation of s_i and key representation of s_j; δ is a large negative number to make the A_{ij} with $M_{ij} = 0$ near to 0 after softmax function. Following traditional transformer, multi-head self-attention is applied in each KGTransformer layer. And the input of the first KGTransformer layer is sequential embedding of elements in s_{in}.

Here we discuss benefits of the KGTransformer layer with s_{in} as input sequence. (1) Compared to conventional Transformer which allows each element to attend to all elements in the sequence, KGTransformer avoids attention between elements in unrelated triples such as *(William Shakespeare, field of work, Fiction)* and *(France, capital, Paris)* since they do not share any element. (2) Compared to conventional graph neural networks [42, 53] that explicitly encode the graph structures via aggregate one-hop neighbors to entities, KGTransformer allows aggregation from one-hop and two-hop neighbors to each entity per update. For example, considering triple *(William Shakespeare, field of work, Fiction)* and *(William Shakespeare, notable book, Hamlet)*, *Hamlet* attends to one-hop neighbor *William Shakespeare* and two-hop neighbor *Fiction* in KGTransformer layer while only attends to *William Shakespeare* in traditional graph neural network. (3) KGTransformer layer enables entities to attend to elements in triples that are not directly connected but share the same relation, for example, *(China, capital, Beijing)* and *(France, capital, Paris)*. In summary, KGTransformer layer allows diverse but constrained element interactions in KGs.

Finally, KGTransformer is constructed by stacking m KGTransformer layers.

3.2 Sub-graph Pretraining

Given a knowledge graph $\mathcal{G}$, we propose to pretrain the KGTransformer by self-supervised tasks with sampled sub-graphs from $\mathcal{G}$. The overall procedure is shown in Figure 2.

3.2.1 Sub-graph Sampling. We propose two strategies to sample a sub-graph $\mathcal{G}_e$ starting from a randomly selected e as target entity.

- *Random Walk Sampling.* We randomly choose a triple $(e, r, e') \in \mathcal{G}$ or $(e', r, e) \in \mathcal{G}$, and add it into $\mathcal{G}_e$. Then regarding e' as target entity and repeat this step for k times. Such sampling captures global and long sequential relatedness between triples in KGs.
- *Entity-centered Sampling.* We randomly choose $max(k, n)$ triples from one-hop triple set $\{(e, r, e') \in \mathcal{G}\} \cup \{(e', r, e) \in \mathcal{G}\}$, where n is the number of one-hop triples of e. If one-hop triples are not sufficient that $n < k$, we sample $k - n$ two-hop triples from $\{(e', p, e'') \in \mathcal{G}\} \cup \{(e'', p, e') \in \mathcal{G}\}$. Entity-centered sampling could capture local relatedness between triples.

After sampling, we serialize the sub-graph $\mathcal{G}_e$ into a sequence s_{in} following Equation (1) as the input of KGTransformer.

3.2.2 Pretraining Tasks. In order to make KGTransformer capture the graph structures and semantics of elements in KGs, we propose

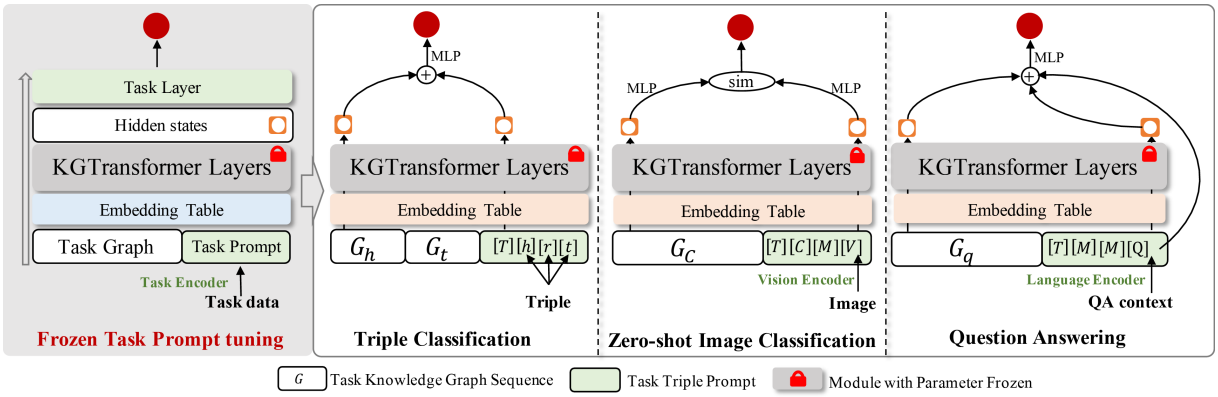

Figure 3: Overview of task prompt tuning (left) and examples of three specific tasks (right).

three self-supervised tasks. In the rest of the paper, we use s_e^m to represent the hidden state of element e from the last layer of KGTransformer.

- *Masked Entity Modeling (MEM).* We randomly replace some entities $\mathcal{M}_e$ with mask token $[M]$. For each masked entity e, we make projected s_e^m via $W_{EME} \in \mathbb{R}^{d\times d}$ to be closed to $\mathbf{E}(e)$, the embedding of entity e before being input to the first KGTransformer layer, and far away from others. The loss of MEM is

$$L_{MEM}(\mathcal{G}_e) = \sum_{e\in\mathcal{M}_e} CE(s_e^m W_{MEM}\mathbf{E}(e)^\top, 1) \quad (7)$$

$$+ \sum_{e'\in\triangle} CE(s_e^m W_{MEM}\mathbf{E}(e')^\top, 0), \quad (8)$$

  where CE() is the cross entropy loss function. $\triangle = \{e'|e' \neq e\}$ is the set of negative entities that are not equal to masked entity e.
- *Masked Relation Modeling (MRM).* We randomly replace some relations $\mathcal{M}_r$ with $[M]$. For each masked relation r, we conduct multi-classes classification and the loss is

$$L_{MRM}(\mathcal{G}_e) = \sum_{r\in\mathcal{M}_r} CE(MLP(s_r^m W_{MRM}), l_r),\ W_{MRM} \in \mathbb{R}^{d\times d} \quad (9)$$

  where $W_{MRM} \in \mathbb{R}^{d\times n_r}$ is a transformation matrix; l_r is a one-hot vector label for masked relation r.
- *Entity Pair Modeling (EPM).* Given two sub-graphs $\mathcal{G}_{e_i}$ and $\mathcal{G}_{e_j}$ of e_i and e_j, we serialize and concatenate them as input sequence and the loss function is

$$L_{EPM}(\mathcal{G}_{e_i}, \mathcal{G}_{e_j}) = CE(MLP([s^m_{[B]_{e_i}} || s^m_{[B]_{e_j}}]), l_{(e_i,e_j)}) \quad (10)$$

  where $s^m_{[B]_{e_i}}$ is the hidden state from the last KGTransformer layer corresponding to $[B]$ in the sequence created from $\mathcal{G}_{e_i}$. $[x||y]$ represents the concatenation of vector x and vector y. $l(e_i, e_j)$ is the label for similarity of e_i and e_j. We regard entity e_i and entity e_j to be similar and labeled them $l(e_i, e_j) = 1$ if they used to be the head(tail) entity of the same relation in $\mathcal{G}$, otherwise $l(e_i, e_j) = 0$.

With these three self-supervised tasks, the overall pretraining loss function is

$$L(\mathcal{G}) = \sum_{e\in\mathcal{E}} (L_{MEM}(\mathcal{G}_e^1) + L_{MRM}(\mathcal{G}_e^2) + L_{EPM}(\mathcal{G}_e^3, \mathcal{G}_{e'})) \quad (11)$$

where $\mathcal{G}_e^1$ and $\mathcal{G}_e^2$ are subgraphs of e from random walk sampling or entity-centered sampling; $\mathcal{G}_e^3$ and $\mathcal{G}_{e'}$ is subgraphs of e and e' from entity-centered sampling, where $e' \neq e$.

3.3 Task Prompt Tuning

With pretrained KGTransformer, we propose a simple and uniform prompt tuning mechanism to apply it in KG-related tasks.

Before prompt tuning, we continually train KGTransformer with KGs in tasks following the pretraining method introduced in Section 3.2, with $\theta_{\mathcal{M}}$ frozen, since the entities and relations are different in tasks. After such training, we get the element embeddings of task KGs adapted to the $\theta_{\mathcal{M}}$.

During prompt tuning, in a task, one sample includes task data $\mathcal{D}_{task}$ and a supporting graph $\mathcal{G}_{task}$. For example, in question answering, $\mathcal{D}_{task}$ is a question-choice pair and $\mathcal{G}_{task}$ is a knowledge graph extracted based on keywords in the pair. Firstly, with $\mathcal{D}_{task}$, we construct a prompt in the form of a triple sequence that $\mathcal{P}_{task} = [[T][H][R][D]]$ where $[T]$ is a special token indicating the beginning of task which is randomly initialized during tuning. $[H]$ and $[R]$ indicate the head entity and relation in the task triple. $[D]$ is the tail entity in task triple and its representation is from a task encoder with $\mathcal{D}_{task}$ as input. Secondly, for $\mathcal{G}_{task}$, we serialize it into a sequence in the form as Equation (1) and concatenate it with task prompt as the input of KGTransformer. Finally, we take out some hidden state s^m from the last KGTransformer layer and input them into a task layer for prediction. We make parameters in KGTransformer layers $\theta_{\mathcal{M}}$ frozen to keep the knowledge of graph structure unrelated to specific entities and relations learned during pretraining as it is. The overall process of task prompt tuning is shown in left part in Figure 3. The right part of Figure 3 shows examples of three specific tasks. Since the fine-tuning processes in different tasks are slightly different, we'll introduce the details of specific tasks in Section 4.

4 EXPERIMENT

In this section, we introduce the experiments of KGTransformer on KG-related tasks. We choose the tasks based on the principle to cover both in-KG and out-of-KG tasks, data of different modalities, and different overlaps with pretraining dataset. Finally, three tasks, triple classification, zero-shot image classification, and question answering, are selected.

Table 1: Statistics of WFC dataset. Std_{rel}(Std_{ent}) is the standard deviation of the number of triples of relations(entities).

	#R	#E	# T	Std_{rel}	Std_{ent}	Density($\times 10^{-6}$)
WFC	317	133435	1015556	13230	134	0.18
wn18rr	11	40943	93003	12540	9	5.04
fb15k − 237	237	14541	310115	2467	128	6.19
codex	69	77951	612437	24664	159	1.46

After pretraining, in each task, we show the results of specifically designed tasks models and KGTransformer with three settings during task tuning: re-using $\theta_{\mathcal{M}}$ and keeping them frozen (*KGTransformer*); tuning $\theta_{\mathcal{M}}$ together with task parameters (*KGT-finetune*); training KGTransformer from scratch (*KGT-scratch*).

4.1 pretraining

4.1.1 Datasets. For pretraining of KGTransformer, a KG covers diverse graph structures is preferred since KGTransformer is designed for graph structure pretraining. Thus we created a new dataset named WFC by combining three common KG datasets with different graph properties, including WN18RR, FB15k-237, and Codex. WN18RR is a dense and unbalanced KG. FB15k-237 is a dense and balanced KG. Codex is a sparse and unbalanced KG. Containing these three datasets, WFC covers diverse graph structures. The statistics of WFC and three datasets are shown in Table 1.

4.1.2 Pretraining Details. The KGTransformer is constructed with 4 KGTransformer layers. The token embeddings and $\theta_{\mathcal{M}}$ are randomly initialized. In each layer, there are 768 hidden units and 12 attention heads. The number of triples k in sampled subgraph is set to 126. Due to the disorder of the triples in the sub-graphs, the position embedding is not applied in KGTransformer.

During pretraining, given an entity e, we generate multiple samples for each task. For MEM and MRM tasks, we get a sub-graph by random-walk sampling or entity-centered sampling and randomly select 15% of triples to mask either the head or the tail entity in the MEM task and to mask relation in the MRM task, when serializing the graph into an input sequence. We randomly sampled 2 entities in current training batch as e' in Equation (8) for MEM task. For the EPM task, we first sample a positive(negative) entity pair (e, e_j) with a probability of 0.5, where e_j and e used(not used) to be the head or tail entity of the same relation. We sample the sub-graph of e and e_j by entity-centered sampling. The MLPs in Equation (9) and Equation (10) are both 1-layer. The model is implemented with Pytorch and trained on 1 Tsela-A100 GPU with batch size as 4 for 10 epochs (97 hours). The optimizer is Adam whose learning rate is 0.0001, together with a linear decay learning rate schedule with warm-up. The model size is 508M containing about 133 million parameters, among which there are 28 million parameters in $\theta_{\mathcal{M}}$, that will transfer to KG-related tasks.

4.2 Task1: Triple Classification

4.2.1 KGTransformer Implementation. In this task, $\mathcal{D}_{task} = (h, r, t)$ and $\mathcal{G}_{task}$ is the concatenation of h and t centered sub-graph, denoted as $\mathcal{G}_h$ and $\mathcal{G}_t$ respectively. $\mathcal{P}_{task} = [[T]\ [h]\ [r]\ [t]]$. In the last layer of KGTransformer, we take out the hidden states corresponding to the first token $[B]$ and task token $[T]$, denoted as $s^m_{[B]}$ and $s^m_{[T]}$ which are inputted into a feed-forward network to output the score of (h, r, t). The score of (h, r, t) is expected to be 1 for the positive one and 0 for the negative one. We use a cross-entropy loss during training. The detailed process is shown in Figure 3.

4.2.2 Training Details. We test and tune KGTransformer on WN18RR dataset. During tuning, we construct 10 negative triples by randomly replacing h or t in (h, r, t). Training batch size is set to 16 and optimizer is Adam [22] with learning rate 0.0001.

4.2.3 Results Analysis. We compare results of KGTransformer to three commonly used KG embedding methods, TransE [3], ComplEx [50], and RotatE [46], which has proved to be powerful at KG predictions. We evaluate models on 4 classification metrics, including Accuracy(Acc.), Precision, Recall, and F1 score, among which accuracy and F1 are more important metrics. Table 3 shows results.

Compared to powerful KGEs, pretrained *KGTransformer* outperforms them and overall achieves the best triple classification results, especially on metrics of Accuracy and F1 score. KGTransformer is better at recalling positive triples than KGEs and has a reasonable precision on predicted positive triples, thus resulting in the best F1 score. These results prove that the pretrained KGTransformer is applicable and effective for in-KG task triple classification.

Compared to *KGTransformer* with $\theta_{\mathcal{M}}$ frozen, overall results of *KGT-finetune* is worse, whose averaged accuracy and F1 is 88.27 while *KGTransformer* gives 89.73, showing the frozen pretrained KGTransformer layer introduce better local minima during tuning which is beyond what could be reached based on task data only. This is further supported by results of *KGT-scratch*, which is significantly worse than *KGTransformer*.

4.3 Task2: Zero-shot Image Classification

Zero-shot(ZS) image classification is a challenging task to train models on samples with *seen classes* while test models on samples with *unseen classes* that have no training samples. KGs could be used as auxiliary information for augmenting ZSL tasks[14, 60].

4.3.1 KGTransformer Implementation. Applying pretrained KGTransformer to this task, we frame the task as outputting the matching score between the input image I and target class C. $\mathcal{G}_{task}$ is the class C centered sub-graph. $\mathcal{P}_{task} = [[T]\ [C]\ [M]\ [V]]$ where $[C]$ is the class token sharing the embedding with entity $C \in \mathcal{G}_{task}$; $[V]$ is the token for image I whose representation comes from a vision encoder, e.g. ResNet[17]. We take out $s^m_{[B]}$ and $s^m_{[V]}$ and project them through an MLP layer, and then calculate the cosine similarity between them as the matching score of the input image I and class C. Considering that KG is used to augment unseen classes in this task while not inverse, after generating neighborship matrix M following Equation (3), we specifically set values in the columns corresponding to the prompt token $[T]$, $[C]$, $[M]$ and $[V]$ in M as 0. That is, $\mathcal{G}_{task}$ can directly affect the learning of the prompt representation, but not vice versa. We applied Binary Cross Entropy loss to encourage the cosine similarity to be large for positive pairs and small for negative pairs. The overall process is shown in Figure 3.

4.3.2 Training Details. Experiments are conducted on AwA-KG [15], a benchmark includes samples from AwA dataset [63] and a basic KG $\mathcal{G}_{task}$ containing hierarchies and attribute annotations of

Table 2: Statistics of dataset in three tasks. #E, #R, and #T is the number of entities, relations, and triples.

	Task KG				Task Data				Properties		
	KG	#E	#R	#T	Task Sample	# Train	#Valid	#Test	Task Type	Modality	Overlap to WFC
T1: Triple Classification	WN18RR	40943	11	-	Triple	86835	6068	6268	in-KG	KG	Yes
T2: ZSL Image Classification	AwA-KG	146	16	1595	Image	23527	-	13795	out-of-KG	Image+KG	No
T3: Question Answering	CommonsenQA	64388	16	309444	QA pair	8500	1221	1241	out-of-KG	Language+KG	No

Table 3: Results of triple classification on WN18RR[46]. Columns with gray background are more important.

	Acc.	Precision	Recall	F1
TransE [3]	88.35	93.45	82.48	87.62
RotatE [46]	88.26	93.03	82.71	87.57
ComplEx [50]	85.07	**96.73**	72.59	82.94
KGTransformer	**89.21**	85.56	**94.32**	**89.73**
KGT-finetune	87.48	83.02	94.22	88.27
KGT-scratch	67.02	67.91	64.55	66.19

Table 4: Results of ZS image classification on AwA-KG[15]

	T1	S	U	H
DeViSE[10]	43.24	**86.44**	6.40	11.91
GCNZ[60]	62.98	75.59	20.28	31.98
OntoZSL[14]	62.65	59.59	50.58	54.71
KGTransformer	**66.26**	61.13	**55.14**	**57.98**
KGT-finetune	63.59	63.61	50.08	56.04
KGT-scratch	58.96	56.48	47.93	51.85

40 seen classes and 10 unseen classes. During tuning, we first train the new embedding table on $\mathcal{G}_{task}$ with three pretraining tasks and then tune the model on the image classification task. We use pretrained ResNet [17] as vision encoder encoding image I into a 2048 dimensional vector and transform the image vector into 768 dimension through a trainable transformation matrix as the representation for token $[V]$. We tune the model with batch size set to 24 and Adam [22], whose initial rate is set at 0.0001. During tuning, we froze the parameters in ResNet for simplicity.

4.3.3 Results Analysis. In Table 4, we compare the KGTransformer with 3 zero-shot learning (ZSL) methods supporting auxiliary KGs. We evaluate our methods and baselines on both conventional ZSL and general ZSL tasks. In conventional ZSL tasks, only images with unseen classes are tested, and only unseen classes are targeted to be classified into, which means we know the test image belongs to unseen classes. In general ZSL tasks, images with seen and unseen classes are tested. Since we do not know the true class of the image is seen or unseen, the candidate classes to be classified into include all seen and unseen classes. Thus general ZSL is a more challenging task. For conventional ZSL, the class-balanced accuracy is reported (T1). For general ZSL, the class-balanced accuracy on seen classes (S), unseen classes (U), and hybrid metrics (H) are reported, where $H = \frac{2(S*U)}{S+U}$. Overall, the H value is a more important metric for general ZSL. The results are shown in Table 4.

Compared to baselines, the pretrained *KGTransformer* achieves the best results on metrics T1, U, and H and overall performs the best. On S metrics, DeViSE presents the best results showing that it is significantly biased to seen classes with limited predictive power on unseen classes. Comparing different settings of pretrained *KGTransformer* with $\theta_{\mathcal{M}}$ frozen, *KGT-finetune* achieves slightly worse results than *KGTransformer* and *KGT-scratch* present significantly worse results, which are consistent to the results on triple classification (Table 3). In summary, Table 4 shows the pretrained *KGTransformer* is applicable and effective in zero-shot image classification. It successfully builds semantic connections between seen and unseen classes, generalizes to unseen classes during test, and has a better capability of generalizing predictive power on seen classes to unseen classes than baselines.

4.4 Task 3: Question Answering (QA)

QA task is to select the correct answer given a natural language question. It is a challenging task requiring complex reasoning over constraints stated in the question consistent with the relevant knowledge in the world. KGs are used to provide commonsense background knowledge for this challenging task[67].

4.4.1 KGTransformer Implementation. Applying KGTransformer to QA, we make KGTransformer to predict the likelihood of the input question-choice pair being correct. Given a question-choice pair qc, we extract $\mathcal{G}_{task}$ from ConceptNet[43] according to a set of keywords $\mathcal{W}$ in qc following [67]. $\mathcal{P}_{task}$ = [[T] [M] [M] [Q]] where [Q] is the token for qc whose representation R_{qc} come from a language encoder, such as BERT[7], RoBERTa[34]. Following [67], in the task layer, we first concatenate $s^m_{[B]}$, $s^m_{[Q]}$ and R_{qc} and then input them into an MLP layer to predict the likelihood of the qc. The overall process is shown in Figure 3.

4.4.2 Training Details. We experiment on CommonsenQA benchmark [48], a 5-way multiple-choice QA task. We report the main results on the in-house (IH) data splits and official test set as did in [67]. We apply RoBERTa-large [34] as language encoder with the first 1024 dimensional hidden state in the sequence as R_{qc}. We transform R_{qc} into a 768-dimensional one through a transformation matrix. We regard $\mathcal{P}_{task}$ and a triple in $\mathcal{G}_{task}$ containing words in $\mathcal{W}$ as related during the construction of M. We tune the model with batch size set to 128 and Adam[22] whose initial rate set as 0.00001. In the first 4 tuning epochs, the RoBERTa-large is frozen.

4.4.3 Results Analysis. In Table 6, we compare KGTransformer to recently proposed QA methods and report their accuracy on in-house valid and test sets. All baselines are using RoBERTa-Large[34] as language encoder.

Compared to baselines, *KGTransformer* achieves the best results. Compared to the RoBERTa-large(ours) that was produced in the same experiment environment, and code framework, our model encoding KG related to question-choice pair by the pretrained KG-Transformer successfully improves the QA accuracy on both valid and test datasets. The pretrained KGTransformer with three settings all achieves better results than Roberta-Large, among which

Table 5: Ablation Study.

	Triple Classification				Zero-shot Image Classification				Question Answering	
	Acc.	Precision	Recall	F1	T1	S	U	H	IHdev	IHtest
KGTransformer	89.20	85.56	94.32	89.73	66.26	61.13	55.14	57.98	77.64	74.13
- MEM	86.85	82.15	94.16	87.74	63.44	61.49	52.77	56.80	75.84	72.60
- MRM	84.91	79.42	94.22	86.19	62.36	66.41	47.97	55.70	74.45	71.47
- EPM	87.78	83.54	94.10	88.51	65.73	64.13	52.50	57.74	76.33	72.84
-*M*	74.15	67.43	93.42	78.33	61.47	59.98	36.33	45.25	74.12	69.70

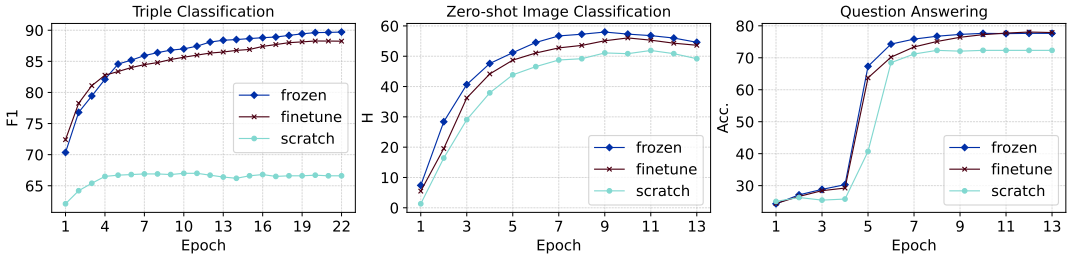

Figure 4: Prediction results of three settings of KGTransformer on three tasks during training.

Table 6: Accuracy of QA on CommonsenseQA.

	IHdev	IHtest
RoBERTa-Large [34]	73.07	68.69
RoBERTa-Large [34](ours)	72.24	68.49
+ GconAttn [59]	72.61	68.59
+ KagNet [28]	73.47	69.01
+ MHGRN [9]	74.45	71.11
+ QA-GNN [67]	76.54	73.41
+ ***KGTransformer***	77.64	74.13
+ *KGT-finetune*	**78.05**	**74.21**
+ *KGT-scratch*	72.32	69.22

results of fine-tuning are the best. Results of freezing are comparable to fine-tuning and are better than training from scratch. The reason for *KGT-finetune* performs better than *KGTransformer* might be that the QA task mainly relies on the quality of the language encoder(RoBERTa-Large), for example, RoBERTa-Large achieves good results on this task, thus fine-tuning $\theta_{\mathcal{M}}$ will make model adapt better to RoBERTa-Large and achieves better results.

4.5 Model Analysis

4.5.1 Ablation Study. In Table 5, we show the results of KGTransformer trained without mask entity modeling(-MEM), without mask relation modeling (-MRM), without entity pair modeling (-EPM), and without neighborship matrix (-M), to illustrate how effective they are for pretraining. We adopt frozen tuning in this ablation study on three tasks. The results show that removing any one of them will make the task results worse, showing they are beneficial to the pretraining of KGTransformer. Among three tasks, MRM contributes the most, and MEM contributes more than EPM. This is reasonable because MRM is the hardest task. Some MEM tasks could be trickily solved by copying entities from the sequence if the masked one is also contained in other triples. Similarly, some EPM tasks could be trickily solved by comparing whether the two entities are the head(tail) entities of the same relation in the subgraph. Without neighborship matrix, the results of three tasks are significantly worse than KGTransformer and KGT-finetune, and slightly better or worse than KGT-scratch. And we find that without the matrix, the pretraining loss did not significantly change and the pretraining model didn't converge. This proves that explicitly telling model the graph structure of KGs is effective for pretraining.

4.5.2 Training Details. In Figure 4, we show the prediction results of the pretrained KGTransformer with three settings on three tasks. We observe the same phenomena across three tasks that (1) tuning KGTransformer with $\theta_{\mathcal{M}}$ frozen leads to good results, which is usually better than or comparable to fine-tuning $\theta_{\mathcal{M}}$, and it is significantly better than training from scratch; (2) tuning KGTransformer with $\theta_{\mathcal{M}}$ frozen achieves reasonable results faster. For example, in triple classification, results of 5 epochs from *KGTransformer* are comparable to 10 epochs from *KGT-finetune*.

Thus we conclude that (1) pretraining KGTransformer on KGs with diverse structures enables it to learn global graph structure knowledge in KG that could not be sufficiently learned based on only task KG; (2) we recommend tuning the pretrained KGTransformer with $\theta_{\mathcal{M}}$ frozen to keep and transfer graph structure knowledge learned from $\mathcal{G}_{pre}$ to downstream tasks better and faster.

5 CONCLUSION AND DISCUSSION

In this paper, we propose a novel KG pretraining model KGTransformer and prove it is possible to pretrain a model with a general knowledge representation and fusion module in multiple tasks supported by different KGs. We pretrain KGTransformer on a hybrid KG with diverse graph structures and prompt tuning it uniformly on three typical KG-related tasks. KGTransformer performs better than specifically designed models across different tasks. More importantly, simply applying pretrained KGTransformer off the shelf gives promising results, showing the capability of deep graph structure transfer that the pretrained KGTransformer has.

Though effective, KGTransformer is a heavier KG model than conventional KGEs. It requires more memory and computation resources. In the future, we would like to explore how to apply the pretrained KGTransformer to applications with limited resources, such as mobile applications and edge computing.

ACKNOWLEDGMENTS

This work is funded by NSFC91846204/U19B2027. Wen Zhang is supported by Zhejiang Provincial Natural Science Foundation of China (No. LQ23F020017) and Yongjiang Talent Introduction Programme.

REFERENCES

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lucic, and Cordelia Schmid. 2021. ViViT: A Video Vision Transformer. In *ICCV*. IEEE, 6816–6826.
[2] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. 2022. BEiT: BERT Pre-Training of Image Transformers. In *ICLR*. OpenReview.net.
[3] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*. 2787–2795.
[4] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an Architecture for Never-Ending Language Learning. In *AAAI*. AAAI Press.
[5] Sanxing Chen, Xiaodong Liu, Jianfeng Gao, Jian Jiao, Ruofei Zhang, and Yangfeng Ji. 2021. HittER: Hierarchical Transformers for Knowledge Graph Embeddings. In *EMNLP (1)*. Association for Computational Linguistics, 10395–10407.
[6] Xiang Chen, Ningyu Zhang, Lei Li, Shumin Deng, Chuanqi Tan, Changliang Xu, Fei Huang, Luo Si, and Huajun Chen. 2022. Hybrid Transformer with Multi-level Fusion for Multimodal Knowledge Graph Completion. In *SIGIR*. ACM, 904–915.
[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*. Association for Computational Linguistics, 4171–4186.
[8] 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*. OpenReview.net.
[9] Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. 2020. Scalable Multi-Hop Relational Reasoning for Knowledge-Aware Question Answering. In *EMNLP (1)*. Association for Computational Linguistics, 1295–1309.
[10] Andrea Frome, Gregory S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomás Mikolov. 2013. DeViSE: A Deep Visual-Semantic Embedding Model. In *NIPS*. 2121–2129.
[11] Dehong Gao, Linbo Jin, Ben Chen, Minghui Qiu, Peng Li, Yi Wei, Yi Hu, and Hao Wang. 2020. FashionBERT: Text and Image Matching with Adaptive Loss for Cross-modal Retrieval. In *SIGIR*. ACM, 2251–2260.
[12] Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making Pre-trained Language Models Better Few-shot Learners. In *ACL/IJCNLP (1)*. Association for Computational Linguistics, 3816–3830.
[13] François Gardères, Maryam Ziaeefard, Baptiste Abeloos, and Freddy Lécué. 2020. ConceptBert: Concept-Aware Representation for Visual Question Answering. In *EMNLP (Findings) (Findings of ACL, Vol. EMNLP 2020)*. Association for Computational Linguistics, 489–498.
[14] Yuxia Geng, Jiaoyan Chen, Zhuo Chen, Jeff Z. Pan, Zhiquan Ye, Zonggang Yuan, Yantao Jia, and Huajun Chen. 2021. OntoZSL: Ontology-enhanced Zero-shot Learning. In *WWW*. ACM / IW3C2, 3325–3336.
[15] Yuxia Geng, Jiaoyan Chen, Zhuo Chen, Jeff Z Pan, Zonggang Yuan, and Huajun Chen. 2021. K-ZSL: resources for knowledge-driven zero-shot learning. *arXiv preprint arXiv:2106.15047* (2021).
[16] Yuxia Geng, Jiaoyan Chen, Wen Zhang, Yajing Xu, Zhuo Chen, Jeff Z. Pan, Yufeng Huang, Feiyu Xiong, and Huajun Chen. 2022. Disentangled Ontology Embedding for Zero-shot Learning. In *KDD*. ACM, 443–453.
[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. IEEE Computer Society, 770–778.
[18] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. TaPas: Weakly Supervised Table Parsing via Pre-training. In *ACL*. Association for Computational Linguistics, 4320–4333.
[19] Zhiwei Hu, Víctor Gutiérrez-Basulto, Zhiliang Xiang, Ru Li, and Jeff Z. Pan. 2022. Transformer-based Entity Typing in Knowledge Graphs. In *EMNLP*. Association for Computational Linguistics, 5988–6001.
[20] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge Graph Embedding Based Question Answering. In *WSDM*. ACM, 105–113.
[21] Bosung Kim, Taesuk Hong, Youngjoong Ko, and Jungyun Seo. 2020. Multi-Task Learning for Knowledge Graph Completion with Pre-trained Language Models. In *COLING*. International Committee on Computational Linguistics, 1737–1743.
[22] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[23] Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *NAACL-HLT (1)*. Association for Computational Linguistics, 2284–2293.
[24] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *EMNLP (1)*. Association for Computational Linguistics, 3045–3059.
[25] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL*. Association for Computational Linguistics, 7871–7880.
[26] Han Li, Dan Zhao, and Jianyang Zeng. 2022. KPGT: Knowledge-Guided Pre-training of Graph Transformer for Molecular Property Prediction. In *KDD*. ACM, 857–867.
[27] Zhifei Li, Hai Liu, Zhaoli Zhang, Tingting Liu, and Neal N. Xiong. 2022. Learning Knowledge Graph Embedding With Heterogeneous Relation Attention Networks. *IEEE Trans. Neural Networks Learn. Syst.* 33, 8 (2022), 3961–3973.
[28] Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning. In *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2829–2839.
[29] Qika Lin, Jun Liu, Fangzhi Xu, Yudai Pan, Yifan Zhu, Lingling Zhang, and Tianzhe Zhao. 2022. Incorporating Context Graph with Logical Reasoning for Inductive Relation Prediction. In *SIGIR*. ACM, 893–903.
[30] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI*. AAAI Press, 2181–2187.
[31] Lihui Liu, Boxin Du, Jiejun Xu, Yinglong Xia, and Hanghang Tong. 2022. Joint Knowledge Graph Completion and Question Answering. In *KDD*. ACM, 1098–1108.
[32] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586* (2021).
[33] Xiao Liu, Shiyu Zhao, Kai Su, Yukuo Cen, Jiezhong Qiu, Mengdi Zhang, Wei Wu, Yuxiao Dong, and Jie Tang. 2022. Mask and Reason: Pre-Training Knowledge Graph Transformers for Complex Logical Queries. In *KDD*. ACM, 1120–1130.
[34] 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. *CoRR* abs/1907.11692 (2019).
[35] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. In *NeurIPS*. 13–23.
[36] Sijie Mai, Shuangjia Zheng, Yuedong Yang, and Haifeng Hu. 2021. Communicative Message Passing for Inductive Relation Reasoning. In *AAAI*. AAAI Press, 4294–4302.
[37] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. (2018).
[38] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67.
[39] Thomas Rebele, Fabian M. Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. 2016. YAGO: A Multilingual Knowledge Base from Wikipedia, Wordnet, and Geonames. In *ISWC (2) (Lecture Notes in Computer Science, Vol. 9982)*. 177–185.
[40] Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-Sequence Knowledge Graph Completion and Question Answering. In *ACL (1)*. Association for Computational Linguistics, 2814–2828.
[41] Apoorv Saxena, Aditay Tripathi, and Partha P. Talukdar. 2020. Improving Multi-hop Question Answering over Knowledge Graphs using Knowledge Base Embeddings. In *ACL*. Association for Computational Linguistics, 4498–4507.
[42] 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 (Lecture Notes in Computer Science, Vol. 10843)*. Springer, 593–607.
[43] Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In *AAAI*. AAAI Press, 4444–4451.
[44] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2020. VL-BERT: Pre-training of Generic Visual-Linguistic Representations. In *ICLR*. OpenReview.net.
[45] Haohai Sun, Shangyi Geng, Jialun Zhong, Han Hu, and Kun He. 2022. Graph Hawkes Transformer for Extrapolated Reasoning on Temporal Knowledge Graphs. In *EMNLP*. Association for Computational Linguistics, 7481–7493.
[46] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR (Poster)*. OpenReview.net.
[47] Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. 2018. Bootstrapping Entity Alignment with Knowledge Graph Embedding. In *IJCAI*. ijcai.org, 4396–4402.

[48] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. In *NAACL-HLT (1)*. Association for Computational Linguistics, 4149–4158.

[49] Komal K. Teru, Etienne G. Denis, and William L. Hamilton. 2020. Inductive Relation Prediction by Subgraph Reasoning. In *ICML (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 9448–9457.

[50] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *ICML (JMLR Workshop and Conference Proceedings, Vol. 48)*. JMLR.org, 2071–2080.

[51] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. 2020. Composition-based Multi-Relational Graph Convolutional Networks. In *ICLR*. OpenReview.net.

[52] 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*. 5998–6008.

[53] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. *ICLR* (2018).

[54] Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[55] Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. 2021. Structure-Augmented Text Representation Learning for Efficient Knowledge Graph Completion. In *WWW*. ACM / IW3C2, 1737–1748.

[56] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. In *CIKM*. ACM, 417–426.

[57] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation. *Trans. Assoc. Comput. Linguistics* 9 (2021), 176–194.

[58] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *KDD*. ACM, 950–958.

[59] Xiaoyan Wang, Pavan Kapanipathi, Ryan Musa, Mo Yu, Kartik Talamadupula, Ibrahim Abdelaziz, Maria Chang, Achille Fokoue, Bassem Makni, Nicholas Mattei, and Michael Witbrock. 2019. Improving Natural Language Inference Using External Knowledge in the Science Questions Domain. In *AAAI*. AAAI Press, 7208–7215.

[60] Xiaolong Wang, Yufei Ye, and Abhinav Gupta. 2018. Zero-Shot Recognition via Semantic Embeddings and Knowledge Graphs. In *CVPR*. Computer Vision Foundation / IEEE Computer Society, 6857–6866.

[61] Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. TUTA: Tree-based Transformers for Generally Structured Table Pre-training. In *KDD*. ACM, 1780–1790.

[62] Zihan Wang, Zhaochun Ren, Chunyu He, Peng Zhang, and Yue Hu. 2019. Robust Embedding with Multi-Level Structures for Link Prediction. In *IJCAI*. ijcai.org, 5240–5246.

[63] Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. 2019. Zero-Shot Learning - A Comprehensive Evaluation of the Good, the Bad and the Ugly. *IEEE Trans. Pattern Anal. Mach. Intell.* 41, 9 (2019), 2251–2265.

[64] Xiaohan Xu, Peng Zhang, Yongquan He, Chengpeng Chao, and Chaoyang Yan. 2022. Subgraph Neighboring Relations Infomax for Inductive Link Prediction on Knowledge Graphs. In *IJCAI*. ijcai.org, 2341–2347.

[65] Zezhong Xu, Wen Zhang, Peng Ye, Hui Chen, and Huajun Chen. 2022. Neural-Symbolic Entangled Framework for Complex Query Answering. *CoRR* abs/2209.08779 (2022).

[66] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for Knowledge Graph Completion. *CoRR* abs/1909.03193 (2019).

[67] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering. In *NAACL-HLT*. Association for Computational Linguistics, 535–546.

[68] 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 *NeurIPS*. 28877–28888.

[69] Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019. Iteratively Learning Embeddings and Rules for Knowledge Graph Reasoning. In *WWW*. ACM, 2366–2377.

[70] Wen Zhang, Chi Man Wong, Ganqiang Ye, Bo Wen, Wei Zhang, and Huajun Chen. 2021. Billion-scale Pre-trained E-commerce Product Knowledge Graph Model. In *ICDE*. IEEE, 2476–2487.

[71] Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. 2020. Learning Hierarchy-Aware Knowledge Graph Embeddings for Link Prediction. In *AAAI*. AAAI Press, 3065–3072.