Mask and Reason: Pre-Training Knowledge Graph Transformers for Complex Logical Queries

Xiao Liu*
Tsinghua University
liuxiao21@mails.tsinghua.edu.cn

Shiyu Zhao*
Tsinghua University
sy-zhao19@mails.tsinghua.edu.cn

Kai Su*
Tsinghua University
suk19@mails.tsinghua.edu.cn

Yukuo Cen
Tsinghua University
cyk20@mails.tsinghua.edu.cn

Jiezhong Qiu
Tsinghua University
jiezhongqiu@outlook.com

Mengdi Zhang
Meituan-Dianping Group
zhangmengdi02@meituan.com

Wei Wu
Meituan-Dianping Group
wuwei30@meituan.com

Yuxiao Dong†
Tsinghua University
yuxiaod@tsinghua.edu.cn

Jie Tang†
Tsinghua University
jietang@tsinghua.edu.cn

Abstract

Knowledge graph (KG) embeddings have been a mainstream approach for reasoning over incomplete KGs. However, limited by their inherently shallow and static architectures, they can hardly deal with the rising focus on complex logical queries, which comprise logical operators, imputed edges, multiple source entities, and unknown intermediate entities. In this work, we present the Knowledge Graph Transformer (kgTransformer)[1] with masked pre-training and fine-tuning strategies. We design a KG triple transformation method to enable Transformer to handle KGs, which is further strengthened by the Mixture-of-Experts (MoE) sparse activation. We then formulate the complex logical queries as masked prediction and introduce a two-stage masked pre-training strategy to improve transferability and generalizability. Extensive experiments on two benchmarks demonstrate that kgTransformer can consistently outperform both KG embedding-based baselines and advanced encoders on nine in-domain and out-of-domain reasoning tasks. Additionally, kgTransformer can reason with explainability via providing the full reasoning paths to interpret given answers.

CCS Concepts

• **Computing methodologies** → **Knowledge representation and reasoning**; **Unsupervised learning**; **Learning latent representations**; • **Information systems** → **Data mining**.

Keywords

Knowledge Graph; Pre-Training; Graph Neural Networks

*The authors contributed equally to this research.
†Jie Tang and Yuxiao Dong are the corresponding authors.
[1]The code is available at https://github.com/THUDM/kgTransformer

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539472

ACM Reference Format:
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 *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539472

1 INTRODUCTION

Knowledge graphs (KGs) store and organize human knowledge about the factual world, such as the human-curated Freebase [2] and Wikidata [40] as well as the semi-automatic constructed ones—NELL [4] and Knowledge Vault [8]. Over the course of KGs' development, representation learning for querying KGs is one of the fundamental problems. Its main challenge lies in the incomplete knowledge and inefficient queries. Web-scale KGs are known to suffer from missing links [41], and the specialized querying tools such as SPARQL cannot deal well with it.

Knowledge graph embeddings (KGEs), which aim at embedding entities and relations into low-dimensional continuous vectors, have thrived in the last decade [3, 36]. Specifically, KGEs have found wide adoptions in the simple KG completion problem $(h, r, ?)$, which features a single head entity h, a relation r and the missing tail entity. However, real-world queries can be more complicated with imputed edges, multiple source entities, Existential Positive First-Order (EPFO) logic, and unknown intermediates, namely, the *complex logical queries*.

Figure 1 illustrates a decomposed query graph for the complex logical query "What musical instrument did Minnesota-born Nobel Prize winner play?". The query composes two source entities ("Minnesota" & "Nobel Prize", denoted as "●"), First-Order logic operator conjunction (∧), unknown intermediate entities ("people that were born in Minnesota and won Nobel Prize", denoted as "○"), unknown target entities ("musical instruments they may play", denoted as "○"), and potential missing edges. A major challenge of answering such query is the exponential complexity along its growing combinations of hops and logical operators. Additionally, the rich context information surrounding multiple entities and relations in a single query should also be taken into account during reasoning.

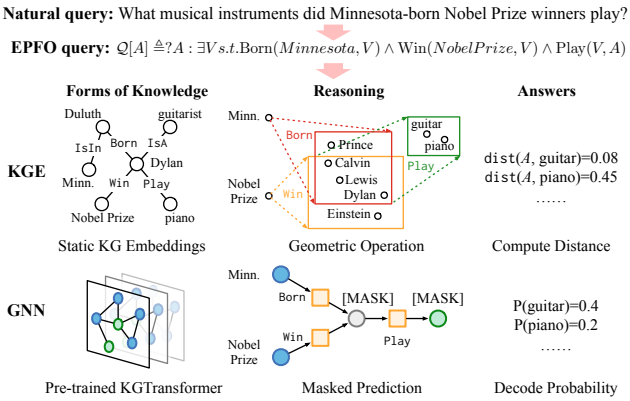


Figure 1: EPFO query reasoning: KGE-based reasoners vs. Pre-trained kgTransformer. The masked prediction training can endow GNNs with natural capability to answer EPFO queries.

Consequently, such query goes beyond the capability of existing KGE-based approaches. First, most KGEs' architectures are shallow and use static vectors, limiting their expressiveness and capacity to capture massive patterns. Second, the training objective of recovering first-order missing links does not comply with the high-order graph nature of complex logical queries, and hence KGEs cannot handle complex queries without training auxiliary logical functions on sampled supervised datasets [1, 12, 28]. In addition, existing KG benchmark datasets [28, 35, 43] usually consist of limited task-specific types of queries, practically prohibiting KGE methods from generalizing well to queries with out-of-domain types.

Contributions. In this work, we propose to learn deep knowledge representations for answering complex queries from these two perspectives: architecture and training objective. At the architecture level, the goal is to design a deep model specified for KGs such that the complex logical queries can be handled. At the training level, the model is expected to learn general instead of task-specific knowledge from KGs and is thus enabled with strong generalizability for out-of-domain queries. To this end, we present kgTransformer—a Transformer-based GNN architecture—with self-supervised pre-training strategies for handling complex logical queries.

kgTransformer. We develop kgTransformer to encode KGs. Specifically, to represent relations in KGs, we propose the *Triple Transformation* strategy that turns relations to relation-nodes and thus transforms a KG into a directed graph without edge attributes. To further enlarge the model capacity with low computation cost, we adopt the *Mixture-of-Experts* strategy to leverage the sparse activation nature of Transformer's feed-forward layers. Altogether, these strategies enable the kgTransformer architecture with high-capacity and computational efficiency, making it capable of answering the EPFO queries on KGs.

Masked Pre-Training. To further improve kgTransformer's generalizability, we introduce a masked pre-training framework to train it. We formulate complex logical query answering as a masked prediction problem. During pre-training, we randomly sample subgraphs from KGs and mask random entities for prediction. It includes two sequential stages of *dense initialization*, which targets at enriching the model by training on dense and arbitrary-shaped contexts, and *sparse refinement*, which is trained on sparse and clean meta-graphs to mitigate the gap between pre-training and downstream queries.

Extensive experiments on two widely-used KG benchmarks, i.e., FB15k-237 and NELL995, demonstrate kgTransformer's performance advantages over state-of-the-art—particularly KGE-based—approaches on nine in-domain and out-of-domain downstream reasoning challenges. Additionally, the case studies suggest that masked pre-training can endow kgTransformer's reasoning with explainability and interpretability via providing predictions over unknown intermediates.

2 The EPFO Logical Queries

We introduce the Existential Positive First-Order (EPFO) logical queries on KGs [28] and identify the unique challenges.

EPFO. Let $\mathcal{G} = (\mathcal{E}, \mathcal{R})$ denote a KG, where $e \in \mathcal{E}$ denotes an entity and $r \in \mathcal{R}$ is a binary predicate (or relation) $r : \mathcal{E} \times \mathcal{E} \to \{\text{True}, \text{False}\}$ that indicates whether a relation holds for a pair of entities. Given the First-Order logical existential ($\exists$) and conjunctive ($\wedge$) operations, the conjunctive queries are defined as:

$$\begin{aligned} \mathcal{Q}[A] \triangleq ?A : \exists E_1, \ldots, E_m . e_1 \wedge \ldots \wedge e_n \\ \text{where } e_i = r(c, E), \text{ with } E \in \{A, E_1, \ldots, E_m\}, c \in \mathcal{E}, r \in \mathcal{R} \\ \text{or } e_i = r(E, E'), \text{ with } E, E' \in \{A, E_1, \ldots, E_m\}, E \neq E', r \in \mathcal{R}. \end{aligned} \tag{1}$$

where A refers to the (unknown) target entity of the query, $E_1, \ldots, E_m$ refer to existentially quantified bound variables (i.e., unknown intermediate entity sets), and c refers to the source entity. Given the query $\mathcal{Q}$, the goal is to find its target entity set $\mathcal{A} \subseteq \mathcal{E}$ that satisfies $a \in \mathcal{A}$ iff $\mathcal{Q}[a]$ is true.

Besides the conjunctive queries, EPFO also covers the disjunctive ($\vee$) queries. A rule-of-thumb practice is to transform an EPFO query into the Disjunctive Normal Form [1, 6, 28]. In other words, a disjunctive query can be decomposed into several conjunctive queries, and a rule can be applied to synthesize conjunctive results for disjunctive predictions.

Challenges. Compared to the KG completion task in which the KGE-based methods are prevalent, the EPFO queries can be multi-hop; their numerous combinations the test reasoner's out-of-domain generalizability. All these characteristics together pose the following unique challenges to reasoners:

- **Exponential complexity:** The complexity of EPFO queries grows exponentially as the hop increases [27], requiring high-capacity and advanced models to handle them. KGE-based reasoners rely on embeddings and simple operators [19, 28] to reason in a "left-to-right" autoregressive fashion. However, there are evidences [1] showing that such models' performance gradually saturates as the embedding dimension grows to 1000. Additionally, during reasoning, the first-encoded entities are unaware of the later-encoded, ignoring the useful bidirectional interactions.
- **Transfer and generalization:** After training, an ideal reasoner is expected to transfer and generalize to out-of-domain queries. But existing EPFO reasoners [1, 20, 28] are directly trained on a limited number of samples within a few query types (i.e., 1p,

2p, 3p, 2i, and 3i in Figure 2 (b)) in a supervised manner, leaving many entities and relations in original KGs untouched. Thus the reasoners are prohibited from grasping knowledge of diverse forms and larger contexts beyond those existing types can express, consequently harming the generalizability.

In summary, these challenges make EPFO queries different from conventional KG completion, which only involves single-hop and single-type queries. In this work, we explore how to effectively handle EPFO queries with the pre-training and fine-tuning paradigm.

3 The KG Pre-Training Framework

In this section, we introduce kgTransformer—a Transformer-based graph neural network (GNN)—for handling EPFO queries on KGs. To endow kgTransformer with strong generalization, we design a masked pre-training and fine-tuning framework. The overall KG pre-training and reasoning framework is illustrated in Figure 2.

3.1 The kgTransformer Architecture

As discussed above, the relatively simple architecture of KGE-based reasoners limits their expressiveness. To overcome this issue, we propose a Transformer-based GNN architecture, kgTransformer, with the Mixture-of-Expert strategy to scale up model parameters while keeping its computational efficiency.

Transformer for KGs. Transformer [38], a neural architecture originally proposed to handle sequences, has achieved early success in the graph domain [18]. To apply transformers to KGs, there are two questions to answer: *1) How to encode the node adjacency in graph structures*, and *2) how to model both entities and relations.*

First, there have been well-trodden practices for encoding node adjacency in graph transformers. For node and graph classification, it is common to view graphs as sequences of tokens with positional encoding, ignoring the adjacency matrices [45, 47]. For link prediction, however, adjacency matrices can be crucial and should be masked to self-attention for better performance [18]. As EPFO reasoning is intrinsically a link prediction problem, we follow HGT [18] to mask adjacency matrices to self-attention.

Second, how to incorporate both entities and relations into Transformer's computation has thus far seldom studied. Here, we present the *Triple Transformation* operation, denoted as function $L(\cdot)$, to turn relations into relation-nodes and consequently transform KGs into directed graphs without edge attributes.

For each directed triple (h, r, t) in $\mathcal{G}$ with $r(h, t) = \text{True}$, we create a node r_{ht} in $L(\mathcal{G})$ (i.e., a *relation-node*) that connects to both h and r. By mapping each relation edge in $\mathcal{G}$ to a relation-node, the resultant graph becomes $L(\mathcal{G}) = (\mathcal{E}', \mathcal{R}')$ where $\mathcal{E}' = \mathcal{E} \cup \{r_{ht} | r(h, t) = \text{True}\}$ and the unattributed edge set $\mathcal{R}' = \{r : r(h, r_{ht}) = r(r_{ht}, t) = \text{True} | r(h, t) = \text{True}\}$. In practice, the computational cost of the triple transformation is low as the reasoning graphs for EPFO queries are usually very small and sparse (~10^1 entities and relations).

Given the input embeddings $\mathbf{x}_e^{(0)} \in \mathbb{R}^d$ for $e \in \mathcal{E}'$ with dimension d, we add a special node-type embedding $t_{\mathbb{I}(e \in \mathcal{E})} \in \mathbb{R}^d$ to distinguish whether it is an entity-node or a relation-node. In the k-th layer of kgTransformer, $d_h = d/H$ and H denotes the number of attention heads, and the multi-head attention is computed as

$$\text{Attn}_i^{(k)} = \text{softmax}(Q^\intercal K / \sqrt{d_h}) V, \tag{2}$$

where $Q = \mathbf{x}_e^{(k-1)} \mathbf{W}_Q$ and $\{K, V\} = \|_{n \in \mathcal{N}(e)} \mathbf{x}_n^{(k-1)} \{\mathbf{W}_K, \mathbf{W}_V\}$. Here $\mathbf{W}_{\{Q,K,V\}} \in \mathbb{R}^{d \times d_h}$, $\|$ denotes concatenation, and $\mathcal{N}(e)$ refers to node e's neighbor set.

Next, the feed-forward network $\text{FFN}(x)$ is applied to attention-weighted outputs projected by $\mathbf{W}_O \in \mathbb{R}^{d \times d}$ as

$$\mathbf{x}_e^{(k)} = \text{FFN}(\Big\|_{i=1}^{H} \text{Attn}_i^{(k)} \cdot \mathbf{W}_O), \text{FFN}(\mathbf{x}) = \sigma(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \tag{3}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d \times 4d}$, $\mathbf{W}_2 \in \mathbb{R}^{4d \times d}$, and σ is the activation function (e.g., GeLU [14]). Note that FFN is critical for Transformers to capture massive patterns and proved equivalent to the key-value networks [11].

Different from 1) existing KGE-based reasoners with the "left-to-right" reasoning order and 2) the sequence encoder-based model [20] that can only reason on acyclic query graphs, kgTransformer designs an architecture to aggregate information from all directions in each layer's computation, making it more flexible and capable of answering queries in arbitrary shapes. Figure 2 (a) illustrates the kgTransformer architecture.

Mixture-of-Experts (MoE). Though kgTransformer's architecture allows it to capture complicated reasoning patterns, the number of its parameters soars up quadratically with the embedding dimension d, which is a common challenge faced by Transformers.

As mentioned earlier, Transformer's FFN is known to be equivalent to the key-value networks [11] where a key activates a few values as responses. Given $x \in \mathbb{R}^d$, FFN's intermediate activation

$$\sigma(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1) = [\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_i, ..., \mathbf{x}_j, ..., \mathbf{x}_{4d}] = \underbrace{[0, 0, ..., \mathbf{x}_i, ..., \mathbf{x}_j, ..., 0]}_{\text{Most of elements are 0}} \tag{4}$$

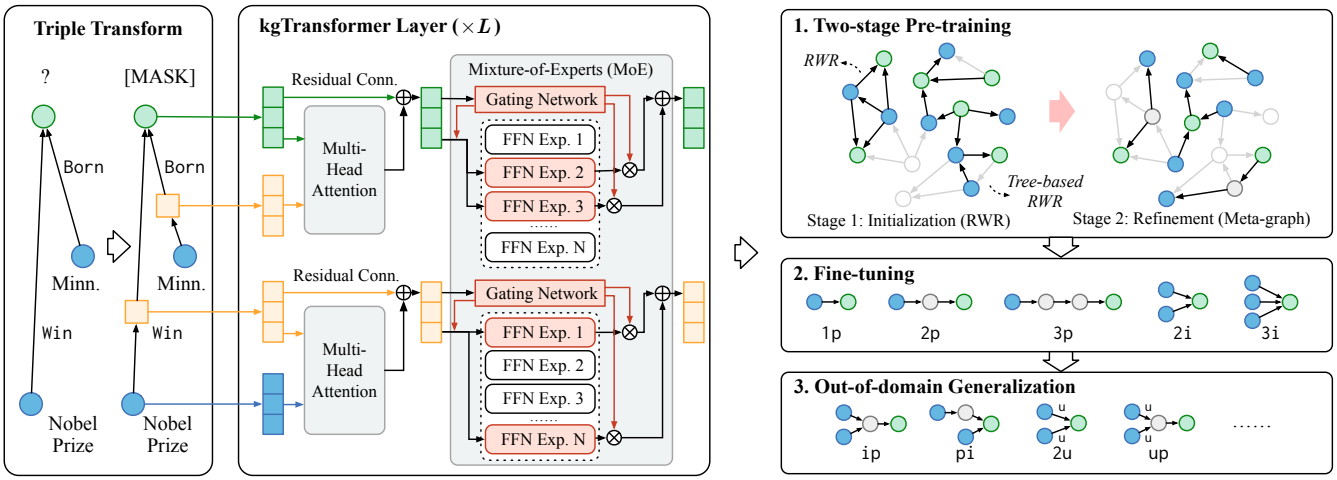
can be extremely sparse, where $\mathbf{W}_1 \in \mathbb{R}^{d \times 4d}$. The level of sparsity varies with tasks. For instance, a recent study [49] shows that usually less than 5% of neurons are activated for each input in NLP. In our preliminary experiments on EPFO queries (Cf. Figure 3), only 10%-20% neurons are activated for certain inputs (except the last decoder layer).

Thus, we propose to leverage the sparsity of kgTransformer via the Mixture-of-Experts (MoE) strategy [30, 32]. MoE first decomposes a large FFN into blockwise experts, and then utilizes a light gating network to select experts to be involved in the computation. For example, an FFN with $\mathbf{W}_1 \in \mathbb{R}^{d \times 16d}$ and $\mathbf{W}_2 \in \mathbb{R}^{16d \times d}$ (4 times larger than that in Equation 4) can be transformed into

$$\text{FFN}(\mathbf{x}) = \{\text{FFN}_{\text{Exp}}^{(i)}(\mathbf{x}) = \sigma(\mathbf{x}\mathbf{W}_1^{(i)} + \mathbf{b}_1)\mathbf{W}_2^{(i)} + \mathbf{b}_2 \mid i \in [1, 8], i \in \mathbb{N}\} \tag{5}$$

where $\mathbf{W}_1^{(i)} \in \mathbb{R}^{d \times 2d}$, $\mathbf{W}_2^{(i)} \in \mathbb{R}^{2d \times d}$. Each $\text{FFN}_{\text{Exp}}^{(i)}$ is referred to as an *expert*. Given a gating network, which is usually a trainable matrix $\mathbf{W}_{\text{gate}} \in \mathbb{R}^{d \times N}$ with N as the number of experts, we can select the top-2 experts for computation [21] as

$$\mathbf{x}_e^{(k)} = \sum_{(i,s) \in S} s \cdot \text{FFN}_{\text{Exp}}^{(i)}(\mathbf{x}_e^{(k-1)}) \tag{6}$$



(a) Architecture: kgTransformer & Mixture-of-Experts (b) Masked Pre-training & Fine-tuning

Figure 2: The KG Pre-Training and Reasoning Framework. (a) kgTransformer with Mixture-of-Experts is a high-capacity architecture that can capture EPFO queries with exponential complexity. (b) Two-stage pre-training trades off general knowledge and task-specific sparse property. Together with fine-tuning, kgTransformer can achieve better in-domain performance and out-of-domain generalization.

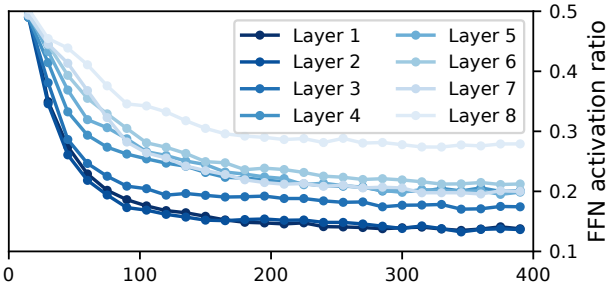


Figure 3: FFN's activation ratio of kgTransformer along pre-training steps (w/o MoE) in preliminary experiments.

where $S = \{(i, s) | s \in \text{softmax}(\text{Top-2}(\mathbf{x}_e^{(k-1)} \mathbf{W}_{\text{gate}})), i \text{ is the index of } s\}$. By using MoE, we can significantly enlarge the actual model size with approximately the same training computation as the FFN in Equation 4. During inference, all experts are included in the computation and the FFN output becomes the weighted average of every expert's output. The implementation details can be found in Appendix C.

3.2 Masked Pre-Training and Fine-Tuning

To further improve the generalizability of kgTransformer, we present a masked pre-training and fine-tuning framework for KG reasoning, which is illustrated in Figure 2 (b).

3.2.1 *Two-stage Pre-training: Initialization and Refinement*

Most existing EPFO reasoners train the embeddings over a limited number of sampled queries with few specific query types in a supervised manner [28]. The low coverage of entities and relations in original KGs in training usually leads to poor transferability to queries with unseen entities in testing. In addition, the reasoners are expected to answer out-of-domain types of queries after training. For example, the standard benchmark [28] trains the model on 5 types of queries (1p, 2p, 3p, 2i, and 3i), and asks the model to test on additional types of queries (ip, pi, 2u, and up).

To overcome this challenge, we propose to do masked pre-training for kgTransformer. The main idea is to randomly sample arbitrary-shaped masked subgraphs from the original KGs' training set to pre-train the model. Its advantages lie in the large receptive fields of sampled subgraphs, diverse shapes of queries, and the high coverage over original KGs. To fully explore general knowledge from pre-training, we should sample masked query graphs as dense and large as possible; however, the query graphs during downstream reasoning are usually small and sparse. Thus, such strategy would cause a mismatch between pre-training and downstream distributions, resulting in performance degradation.

To mitigate the issue, we introduce a *two-stage* masked pre-training strategy for kgTransformer. The first stage aims to initialize kgTransformer with KGs' general knowledge, and the second one further refines its ability for small and sparse queries during inference.

Stage 1: Dense Initialization. The goal of this stage is to enable kgTransformer with the general knowledge in KGs via masked pre-training over dense and large sampled subgraphs.

We use two random-walk-based strategies to sample on original KGs' training set. One is the random walk with restart (RWR), for which we make no assumption on the shapes of subgraphs it may sample. Another is a tree-based RWR strategy, for which we constrain the shape of sampled graphs as tree structures to cater conjunctive queries. We include a majority of induced relations between sampled entities for denser contexts.

Both methods are used to produce sampled query subgraphs. Given a kgTransformer parameterized by θ, a set of random sampled entities $\mathcal{E}_{\text{mask}}$ that are masked by a special mask embedding turn the trainable input embeddings $\mathbf{X} = [\mathbf{x}_{e_0}^{(0)}, ..., \mathbf{x}_{e_T}^{(0)}]$ into mask

corrupted $\hat{\mathbf{X}}$. The dense initialization stage seeks to optimize

$$\mathcal{L}_{\text{init}} = -\sum_{e_{\text{m}} \in \mathcal{E}_{\text{mask}}} \log p_\theta(\mathbf{x}_{e_{\text{m}}} | \hat{\mathbf{X}}, \mathbf{A}) \tag{7}$$

where $\mathbf{A}$ is the adjacency matrix. The prediction is done simultaneously for all masked nodes, rather in an autoregressive manner.

Stage 2: Sparse Refinement. The goal of this stage is to further enhance the model's capacity by using meta-graph sampling for the EPFO queries that are commonly sparse and small.

We adopt the basic EPFO patterns (1p, 2p, 3p, 2i, and 3i) as meta-graphs, which involve no more than four entities and relations. No induced relations between entities are reserved. Following real query practices, except for source entities ("●" in Figure 2(b)), other entities are all masked but only the target entity ("○") is predicted in each query type.

Given each meta-graph type, the set of masked entities $\mathcal{E}_{\text{mask}} = \mathcal{E}_{\text{inter}} \cup \mathcal{E}_{\text{target}}$ where $\mathcal{E}_{\text{inter}}$ and $\mathcal{E}_{\text{target}}$ refer to the sets of intermediate and target entities, respectively. The sparse refinement stage only optimizes the error $\mathcal{L}_{\text{refine}}$ on target entity $e_{\text{t}} \in \mathcal{E}_{\text{target}}$ as the same form of $\mathcal{L}_{\text{init}}$ as

$$\mathcal{L}_{\text{refine}} = -\sum_{e_{\text{t}} \in \mathcal{E}_{\text{target}}} \log p_\theta(\mathbf{x}_{e_{\text{t}}} | \hat{\mathbf{X}}, \mathbf{A}) \tag{8}$$

During kgTransformer pre-training, these two stages are conducted sequentially rather than in parallel.

3.2.2 *Fine-tuning*

We introduce the fine-tuning strategy of kgTransformer for downstream reasoning tasks. Though with the sparse refinement stage during pre-training, our preliminary experiments (Cf. Table 2) show that fine-tuning is still necessary for kgTransformer as the labeled supervision information is used to tune the pre-trained model for downstream tasks.

Specifically, we fine-tune the pre-trained kgTransformer using the downstream EPFO reasoning datasets in the form of masked prediction. Given a set of feasible answers $\mathcal{E}_A$ for a masked query graph, the fine-tuning loss function per query is formulated as

$$\mathcal{L}_{\text{ft}} = -\frac{1}{|\mathcal{E}_A|} \sum_{e_a \in \mathcal{E}_A} \log \frac{\exp(\mathbf{x}_{e_a}^{(L)\intercal} \mathbf{u}_{e_a})}{\exp(\mathbf{x}_{e_a}^{(L)\intercal} \mathbf{u}_{e_a}) + \sum_{e \notin \mathcal{E}_A} \exp(\mathbf{x}_e^{(L)\intercal} \mathbf{u}_e)} \tag{9}$$

where other answers' predicted logits are ignored when computing the loss for one answer, and per query loss is averaged among all answers' losses. $\mathbf{u}_e$ refers to entity e's embedding in the decoder.

In practice, an intuitive strategy is to fine-tune the pre-trained kgTransformer on the corresponding downstream training set for each reasoning task. However, the recent NLP studies [26] have demonstrated that the multi-task fine-tuning can avoid over-fitting and let tasks benefit from each other. Inspired from this observation, we first jointly fine-tune kgTransformer on all possible downstream training sets, and then fine-tune it for each single task. In addition, we observe that a combination of several downstream training sets can sometimes be much better than one single task's training set (Cf. Table 6). Finally, the best checkpoint for testing is selected on the basis of per task validation set performance.

3.2.3 *Out-of-domain Generalization*

To better test the model's generalizability to unseen queries, four specific types of out-of-domain queries, *ip, pi, 2u, up*, are provided only in the validation and test sets as illustrated in Figure 2 (b).

Among them, the conjunctive queries *ip* and *pi* can be represented as query graphs. For the disjunctive/union queries *2u* and *up*, we adopt the idea of Disjunctive Normal Form [6]. Specifically, we first predict on the decomposed conjunctive queries respectively, then normalize each probability distribution into its rank, and finally combine the ranks by taking the highest rank. Instead of taking the mean or max of probabilities, we find that re-scoring the entity prediction by its rank from the probability distribution in each decomposed conjunctive query is more effective, as the probability distributions for different decomposed queries can be of different scales.

4 Experiments

In this section, we evaluate kgTransformer's capacity to reason for EPFO queries with at least one imputed edge, i.e., for answers that cannot be obtained by direct KG traverses [29]. We employ two benchmarks FB15k-237 and NELL995 with nine different reasoning challenges including both in-domain and out-of-domain queries following prior settings in Query2Box [28].

Datasets. The statistics of FB15k-237 [35] and NELL995 [44] can be found in Table 7. We use the standard training/validation/test edge splits [28] to pre-train the kgTransformer model. $\mathcal{G}_{train}$ contains the training edges, and the subgraphs are sampled from $\mathcal{G}_{train}$ for pre-training. We search for hyper-parameters over $\mathcal{G}_{valid}$. Here we do not include FB15k as it is known to suffer from major test leakage through inverse relations as illustrated in [35], which consequently proposes the updated dataset FB15k-237. See Table 8 in Appendix for more detailed information about the datasets.

In addition, the query-answer datasets $[\![q]\!]_{\text{train}}$, $[\![q]\!]_{\text{valid}}$, $[\![q]\!]_{\text{test}}$ are constructed for logic query reasoning by Query2Box [28], including chain-shaped queries (*1p, 2p, 3p*), conjunctive queries (*2i, 3i, ip, pi*), and disjunctive queries (*2u, up*). For a query q, the three sets of answers are obtained by performing subgraph matching over three graphs: $[\![q]\!]_{train}$ from $\mathcal{G}_{train}$, $[\![q]\!]_{valid}$ from $\mathcal{G}_{valid} \backslash \mathcal{G}_{train}$ and $[\![q]\!]_{test}$ from $\mathcal{G}_{test} \backslash \mathcal{G}_{valid}$. Therefore, it does not spoil the train/dev/test split in the original dataset. We use $[\![q]\!]_{train}$ for fine-tuning and training, $[\![q]\!]_{valid}$ for validating, and $[\![q]\!]_{test}$ for testing. Such design allows us to test if the model can predict non-trivial answers that are unable to obtain by traversing the given KG.

The Evaluation Protocol. We follow the evaluation protocol in Query2Box [28] including the filtering setting [3] and metrics calculation. In filtering setting, since the answers to most queries are not unique, we rule out other correct answers when calculating the rank of one answer a. Hits at K(Hits@Km) is used as the metric, which is different from Hits@K as it requires to first average on Hits@K per query and then average over all queries. See more details in Appendix B).

4.1 Main Results

We compare kgTransformer with various methods based on both KGEs and sequence encoders, including GQE [12], Q2B [28],

Table 1: Hits@3m for complex query reasoning. (bold denotes the best results; underline denotes the second best results).

Dataset	Model	Avg	Avg w/o u	In-domain					Out-of-domain			
				1p	2p	3p	2i	3i	ip	pi	2u	up
NELL995	GQE [12]	0.248	0.270	0.417	0.231	0.203	0.318	0.454	0.081	0.188	0.200	0.139
	Q2B [28]	0.306	0.317	0.555	0.266	0.233	0.343	0.480	0.132	0.212	0.369	0.163
	EmQL [34][1]	0.277	0.294	0.456	0.231	0.172	0.331	0.483	0.143	0.244	0.226	0.207
	BiQE [20]	-	0.344	0.587	0.305	0.326	0.371	0.531	0.103	0.187	-	-
	CQD(CO) [1]	0.368	0.370	**0.667**	0.265	0.220	**0.410**	0.529	0.196	0.302	**0.531**	0.194
	CQD(Beam) [1]	0.375	0.385	**0.667**	0.350	0.288	**0.410**	0.529	0.171	0.277	**0.531**	0.156
	kgTransformer	**0.399**	**0.408**	0.625	**0.401**	**0.367**	0.405	**0.546**	**0.203**	**0.306**	0.469	**0.270**
FB15k-237	GQE [12]	0.230	0.250	0.405	0.213	0.153	0.298	0.411	0.085	0.182	0.167	0.160
	Q2B [28]	0.268	0.283	0.467	0.240	0.186	0.324	0.453	0.108	0.205	0.239	0.193
	EmQL [34][1]	0.219	0.241	0.389	0.201	0.154	0.275	0.386	0.101	0.184	0.115	0.165
	BiQE [20]	-	0.293	0.439	0.281	0.239	0.333	0.474	0.110	0.177	-	-
	CQD(CO) [1]	0.272	0.290	0.512	0.213	0.131	0.352	0.457	0.146	0.222	0.281	0.132
	CQD(Beam) [1]	0.290	0.315	**0.512**	0.288	0.221	0.352	0.457	0.129	0.249	**0.284**	0.121
	kgTransformer	**0.325**	**0.350**	0.459	**0.312**	**0.276**	**0.398**	**0.528**	**0.189**	**0.286**	0.263	**0.214**

[1] EmQL's reported results are not under the standard metric. We have verified the mismatch with its authors and re-evaluated the performance.

Table 2: Ablation on pre-training & fine-tuning (Hits@3m).

	FB15k-237	NELL995
kgTransformer (Stage 1 + Stage 2)	**0.336**	0.395
-only Stage 1 in pre-training	0.308	0.307
-only Stage 2 in pre-training	0.307	**0.399**
-w/o fine-tuning	0.301	0.368
-w/o pre-training	0.262	0.288

BiQE [20], EmQL [34], and the state-of-the-art baseline CQD [1]. Detail descriptions can be found in Appendix E.

Table 1 reports the Hits@3m results for all query types on FB15k-237 and NELL995. For FB15K-237, Note that the two-stage pre-training is sequentially adopted for FB15K-237 and only the second stage of pre-training is used for NELL995 (Cf. Section 4.2 for detailed analysis). Among most cases, kgTransformer obtains the best performance on both datasets. In contrast with BiQE, which is also based on Transformer, the kgTransformer model achieves average improvements of 6.4% (18.6% relative) and 5.7% (19.5% relative) without union operation on NELL995 and FB15k-237, respectively. It demonstrates the proposed method's architecture superiority in terms of the model capacity and generalizability. In addition, kgTransformer can perform union operations, supporting the complete set of EPFO queries.

Compared to the previous state-of-the-art CQD, kgTransformer obtains 2.4% (6.4% relative) and 3.5% (12.1% relative) improvements on average over NELL995 and FB15k-237, respectively. For out-of-domain reasoning queries on FB15k-237, kgTransformer outperforms CQD by significant margins for most types, demonstrating pre-training's capability in helping the model gain out-of-domain generalizability.

The major deficiency of kgTransformer happens for the *1p* query, which is identical to the conventional knowledge graph completion problem. First, kgTransformer's multi-layer Transformer architecture may be unfriendly to queries with limited contexts. In addition, the training objective of kgTransformer focuses on complex queries with multiple entities and relations instead of those with two entities and one relation in a triplet. We will the improvement of kgTransformer for the *1p* query for future research.

4.2 Ablation Study

Pre-Training and Fine-Tuning Strategies. We analyze the necessity of two-stage pre-training and the function of fine-tuning through ablation study on both datasets. The contributions of fine-tuning and the pre-training stages are summarized in Table 2, where the average Hits@3m results for all query types are reported.

We observe that without fine-tuning, the performance of kgTransformer drops about 3% in absolute numbers for both datasets, and further without pre-training, the performance drops are enlarged to 7% on FB15k-237 and 11% on NELL995. This demonstrates that both pre-training and fine-tuning can help improve the model capacity of kgTransformer for complex query reasoning. In addition, the pre-trained kgTransformer without fine-tuning can achieve comparable (0.368 vs. 0.375 on NELL99) or even better (0.301 vs. 0.290 on FB15k-237) results than CQD [1]—the previously state-of-the-art method.

Furthermore, two pre-training stages introduce different inductive biases towards the underlying logic. kgTransformer with only stage 1 pre-training assumes the dense correlation between pairs of entities, while its stage 2 pre-training assumes that the relations

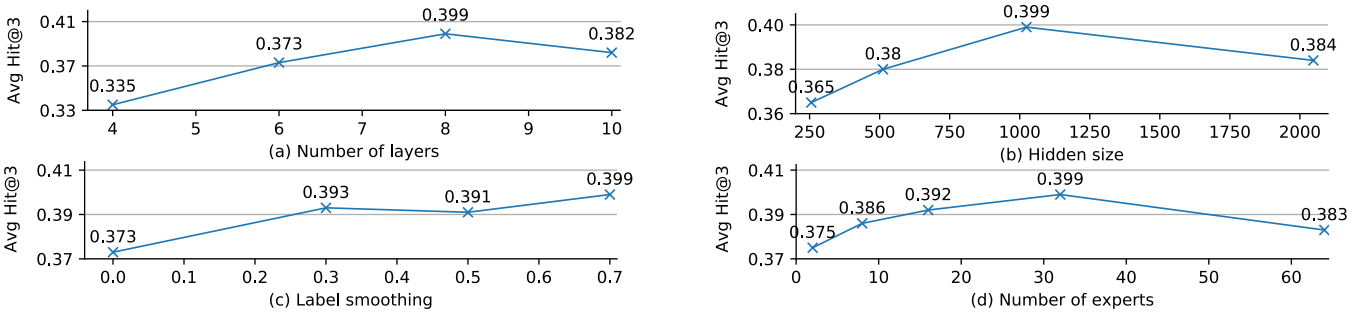


Figure 4: Hyperparameter analysis on NELL995 (Hits@3m). The default setting is: 8 layers, 1024 for hidden size, $\alpha = 0.7$ for label smoothing, and 32 experts.

Table 3: Training and inference time (ms) per step along with different numbers of experts using per step batch size 64.

	Number of experts			
	2	8	16	32
Pre-training stage 1	41.46	42.02	45.38	49.30 (+18.9%)
Pre-training stage 2	17.37	18.83	19.78	24.09 (+38.7%)
Fine-tuning	32.41	32.91	36.46	37.47 (+15.6%)
Inference	12.86	13.36	13.85	14.35 (+11.6%)

among entities follow the chain-rule. We observe that kgTransformer with both stages obtains the good results, while kgTransformer with only stage 2 pre-training generates similar performance on NELL995, due to the fact that it is sparser than FB15k-237 with weaker relations within clusters and stronger connections in chains.

MoE Efficiency. We present an efficiency analysis in terms of the numbers of experts in pre-training, fine-tuning, and inference stages with a batch size of 64 in both training and inference. Table 3 reports the running time in millisecond. We observe that MoE can significantly increase the model capacity with limited increase of computational cost. Take the case of 32 experts for example, MoE helps to enlarge kgTransformer 16× in size by only taking 11.6%-38.7% of additional computation.

Hyperparameters. We conduct the ablation experiments on hyperparameters, including the number of layers, hidden size, the value of label smoothing, and the number of experts. Figure 4 reports the results on NELL995 in terms of Hits@3m.

- **Number of layers:** In Figure 4 (a), we test the performance of different number of layers. It suggests that a sufficient and proper model depth (8 in this case) is essential for kgTransformer, which can be attributed to the relation between the depth and the size of the receptive field. Previously studies have shown that very deep GNNs may suffer from over-smoothing and training instability, thus causing performance drops [9].
- **Hidden size:** Figure 4 (b) studies the influence of hidden size. We observe that the increase of hidden size helps to capture massive EPFO reasoning patterns, but a too-large one (e.g., 2048) can harm the performance, as it may suffer from the similar training difficulty witnessed in KGE-based reasoners [1].
- **Label smoothing:** Label smoothing is crucial to kgTransformer's pre-training, as it mitigates the bias in the pre-training data due to the on-the-fly sampling. Through experiments in Figure 4 (c), the optimal value $\alpha = 0.7$ for NELL995 is exceptionally high in comparison with that in natural language processing ($\alpha = 0.1$). It indicates that by training for long epochs (around 1000 epochs) with a rather large embedding size (1024), the kgTransformer model is prone to overfitting in pre-training. We also observe that exclusion of label smoothing will yield a considerable performance decrease by 2.6%.
- **Number of experts:** We evaluate the performance of kgTransformer with different numbers of experts in Figure 4 (d). The correspondence of sparse nature in KG reasoning and mixture-of-experts is one key point in kgTransformer. Compared with the vanilla Transformer (2 experts), kgTransformer achieves performance improvements with more and more experts until 32 (16× large in model size), demonstrating the power of high-capacity models in EPFO challenges.

4.3 Case Study

Interpretability. In masked querying, the model is enforced to produce plausible predictions for every node in the graph because it does not know the queried nodes in advance. Such property implies the interpretability of the results. The embeddings of intermediate variables can be fed into the prediction layer to generate a result, providing paths to understand how the model produces the answers.

Take the query "Which school is in the same country as Elmira?"—formally, $?A, \exists E, \text{location\_in}(Elmira, E) \wedge \text{in\_country}(E, A)$—in Table 4 for example, the predicted answer is "Davidson College". To verify this answer, we check what the result of E is. Inputting the embedding of E into the decoder, it actually gives "U.S." as the top prediction, which is indeed a valid intermediate entity. More cases concerning interpretability for 3p queries are provided in Table 5 in Appendix.

All-Direction Reasoning. Unlike almost all reasoners' step-by-step searching, kgTransformer is a Transformer-based GNN that captures information from all directions simultaneously. As the second row in Table 4 shows that when the intermediate entity

Table 4: Case study on 2p examples. (a) Interpretability. Filling a certain entity as the tail prediction, we use kgTransformer to predict the unknown intermediates to test its interpretability. **(b) All-direction Reasoning.** In 2p, intermediate and tail entities are masked. kgTransformer considers both relations when predicting intermediate node (marked **orange**). If the second relation is masked (i.e., conventional autoregressive reasoning), intermediate's predictions are valid for the first relation, but no longer for the second (marked red).

Case	Head (Groundtruth)		Intermediate entity (Top 3 prediction)		Tail (Prediction)
Interpretability	Elmira(U.S. City)	*location in*	U.S., Canada, UK	*in country*	Davidson College
All-direction Reasoning	*Salt*(2010 film)	*has subject*	**CIA**, Espionage, PTSD	*employs*	George Bush
			Serial killer, Espionage, PTSD	*(masked)*	-

is required to fit the *employs* relation, "CIA" is more suitable than "Espionage" and "PTSD" as it is an agency. Without the ability to read from tail to head, the model would suggest "Serial killer" instead, which is inadequate for future deductions.

5 Related Work

Existing works on complex logical queries are based on KGEs [1, 5, 10, 12, 28, 34, 39], sequence encoders [20], or rules [25]. Most of them are based on traditional KGEs to adapt for complex logical reasoning with certain geological functions to manipulate the embedding spaces [10, 12, 28, 29, 34] or logical norms over link predictors [1]. Graph Query Embedding (GQE) [12] is proposed to use deep sets as a way for query intersection logic. Logical operators are reformulated as trainable geometric functions in the entity embedding space. Q2B [28] embeds queries as boxes (i.e., hyper-rectangles), and the points inside which are considered as a potential answer entities of the query. Box Lattice [39] is proposed to learn representations in the space of all boxes (axis-aligned hyper-rectangles). EmQL [34] proposes an embedding method with count-min sketch that memories the relations in training set well. CQD [1] uses the previous neural link predictor ComplEx [36] as the one-hop reasoner and T-norms [19] as logic operators. By assembling neural link predictors with T-norms, it translates each query into an end-to-end differentiable objectives. BetaE [29] utilizes Beta distribution for the embeddings. It takes the advantage of well-defined logic operation over distributions and turns the original logic operation over the embedding space into the one over the distribution space. These approaches can easily incorporate logic operations, but are hard to generalize well to unseen and more complicated queries, as their architectures of pure embeddings limit their expressiveness.

There have been efforts to employing advanced architectures for complex logical queries. BiQE [20] introduces the Transformer and mask training, decomposing EPFO queries into sequences to fit the vanilla Transformer's input constraint. However, its nature to process sequences only allows it to answer queries in the shape of directed acyclic graphs (DAGs), which only cover a small set of all possible EPFO queries. In addition, it still follows the supervised training fashion and does not make the full use of the KGs.

Pre-training graph neural networks for better transferability and generalizability has recently aroused wide interest in graph community, inspired by the progress in language [7] and vision. Generally, these pre-training strategies can be categorized into two different self-supervised objectives [22], that is, generative pre-training [16, 17] and contrastive pre-training [24, 46]. Nevertheless, they generally focus on pre-training graph neural networks on academic networks [48], biochemical substances [33], or e-commerce product graphs [15]. Few efforts have been paid to the challenges of pre-training graph neural networks on knowledge graphs.

6 Conclusion

We present the Knowledge Graph Transformer (kgTransformer), a Transformer-based GNN, to pre-train for complex logical reasoning by leveraging the masked pre-training and fine-tuning approach. To adapt to graph structural data, we introduce the Triple Transformation and Mixture-of-Experts strategies for a high-capacity Transformer architecture, and propose a two-stage pre-training framework to gradually endow kgTransformer with the ability to transfer and generalize. Extensive experiments and ablation study demonstrate the effectiveness of kgTransformer's architecture and pre-training methods.

Notwithstanding the promising results, there exist limitations of kgTransformer that require future studies. First, kgTransformer is not competitive on 1p and 2u reasoning queries which are equivalent to the traditional KG completion problem), as we have not injected inductive biases known to be critical for GNNs on KGs (e.g., CompGCN [37]). Second, the KG pre-training is limited to a single KG and does not benefit from cross-KG knowledge transfer. It would be an interesting direction to fuse knowledge from multiple KGs via pre-training to gain further improvements on reasoning. Third, as the recent work [31] indicates, a sequence-to-sequence Transformer that leverages text labels of entities as inputs and outputs can serve as a promising architecture for KG completion in certain scenarios. Finally, to jointly model the KG structure and text information in KG reasoning also remains an unsolved challenge.

ACKNOWLEDGEMENT

We thank the reviewers for their valuable feedback to improve this work. This work is supported by Technology and Innovation Major Project of the Ministry of Science and Technology of China under Grant 2020AAA0108400 and 2020AAA0108402, Natural Science Foundation of China (Key Program, No. 61836013), and National Science Foundation for Distinguished Young Scholars (No. 61825602).

REFERENCES

[1] Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. 2021. Complex Query Answering with Neural Link Predictors. In *ICLR*.

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

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *NIPS* 26 (2013).

[4] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*.

[5] William W. Cohen, Matthew Siegler, and Alex Hofer. 2019. Neural Query Language: A Knowledge Base Query Language for Tensorflow. arXiv:1905.06209

[6] Brian A Davey and Hilary A Priestley. 2002. *Introduction to lattices and order, Second Edition*. Cambridge university press.

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

[8] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *SIGKDD*. 601–610.

[9] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. 2020. Graph Random Neural Network for Semi-Supervised Learning on Graphs. *NeurIPS* (2020).

[10] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic neural networks. In *NeurIPS*. 5345–5355.

[11] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. In *EMNLP*. 5484–5495.

[12] William L Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding logical queries on knowledge graphs. In *NIPS*. 2030–2041.

[13] Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. 2021. Fastmoe: A fast mixture-of-expert training system. *arXiv preprint arXiv:2103.13262* (2021).

[14] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).

[15] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *NeurIPS* 33 (2020), 22118–22133.

[16] W Hu, B Liu, J Gomes, M Zitnik, P Liang, V Pande, and J Leskovec. 2020. Strategies For Pre-training Graph Neural Networks. In *ICLR*.

[17] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. Gpt-gnn: Generative pre-training of graph neural networks. In *SIGKDD*.

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

[19] Erich Peter Klement, Radko Mesiar, and Endre Pap. 2013. *Triangular norms*. Vol. 8. Springer Science & Business Media.

[20] Bhushan Kotnis, Carolin Lawrence, and Mathias Niepert. 2021. Answering Complex Queries in Knowledge Graphs with Bidirectional Sequence Encoders. In *AAAI*, Vol. 35. 4968–4977.

[21] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. In *ICLR*.

[22] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2021. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[23] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. arXiv:1711.05101 [cs.LG]

[24] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *SIGKDD*. 1150–1160.

[25] Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. 2020. RNNLogic: Learning Logic Rules for Reasoning on Knowledge Graphs. In *ICLR*.

[26] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 21, 140 (2020), 1–67.

[27] Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Denny Zhou, Jure Leskovec, and Dale Schuurmans. 2021. SMORE: Knowledge Graph Completion and Multi-hop Reasoning in Massive Knowledge Graphs. *arXiv preprint arXiv:2110.14890* (2021).

[28] Hongyu Ren, Weihua Hu, and Jure Leskovec. 2019. Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings. In *ICLR*.

[29] Hongyu Ren and Jure Leskovec. 2020. Beta Embeddings for Multi-Hop Logical Reasoning in Knowledge Graphs. In *Neural Information Processing Systems*.

[30] Lior Rokach. 2010. *Pattern classification using ensemble methods*. Vol. 75. World Scientific.

[31] Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-Sequence Knowledge Graph Completion and Question Answering. In *ACL*. 2814–2828.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* (2017).

[33] Teague Sterling and John J Irwin. 2015. ZINC 15–ligand discovery for everyone. *Journal of chemical information and modeling* 55, 11 (2015), 2324–2337.

[34] Haitian Sun, Andrew Arnold, Tania Bedrax Weiss, Fernando Pereira, and William W Cohen. 2020. Faithful Embeddings for Knowledge Base Queries. *NIPS* 33 (2020).

[35] Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. Association for Computational Linguistics, Beijing, China, 57–66. https://doi.org/10.18653/v1/W15-4007

[36] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*, Vol. 48.

[37] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based Multi-Relational Graph Convolutional Networks. In *ICLR*.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NIPS* 30 (2017).

[39] Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. 2018. Probabilistic Embedding of Knowledge Graphs with Box Lattice Measures. arXiv:1805.06627

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

[41] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *WWW*. 515–526.

[42] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On layer normalization in the transformer architecture. In *ICML*. PMLR, 10524–10533.

[43] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *EMNLP*. 564–573.

[44] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. ACL Workshop, Copenhagen, Denmark, 564–573. https://doi.org/10.18653/v1/D17-1060

[45] 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? *NeurIPS* 34 (2021).

[46] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *NeurIPS* 33 (2020), 5812–5823.

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

[48] Fanjin Zhang, Xiao Liu, Jie Tang, Yuxiao Dong, Peiran Yao, Jie Zhang, Xiaotao Gu, Yan Wang, Bin Shao, Rui Li, et al. 2019. Oag: Toward linking large-scale heterogeneous entity graphs. In *SIGKDD*. 2585–2595.

[49] Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2021. MoEfication: Conditional Computation of Transformer Models for Efficient Inference. *arXiv preprint arXiv:2110.01786* (2021).

[50] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. 2019. Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks. arXiv:1911.07323

A Sampling Method

In the stage 1 of pre-training, we used random-walk based graph sampling strategies to sample dense and large subgraphs on the training set. We adopt LADIES [50] sampling method and a variant of random walk called meta-tree sampling.

Meta-tree Sampling. As a variant of random walk with restart, meta-tree sampling strategy does not necessarily restart from the target node every time. Instead, it restarts from any sampled node with equal probability. The change is inspired by the intention to sample more neighbors for each node rather than for the target node in a subgraph. In comparison with tuning the parameters of RW, meta-tree sampling provides a direct way to spread the *density* across all the sampled nodes. Such a *density spread* is essential for preventing overfitting to highly dense area. Besides, to sample a dense graph, we specify the restart probability to be 1.0, which means it jumps to any sampled node after one step and continue. In this way, meta-tress sampling balances between the width and the depth and prevents overfitting problem in our setting. Note that we do not sample a node more than once, so the sampled subgraph contains no cycle and forms a tree.

B Evaluation Protocol

To be more precise, we denote the whole entity set as $\mathcal{V}$, and the filtered entity set w.r.t qeury q as $\mathcal{S} = \mathcal{V} \backslash [\![q]\!]_{\text{test}}$.

$$\text{rank}_{\mathcal{V}}(a) = 1 + \sum_{x \in \mathcal{S}} \mathbb{1}[\text{rank}(x) < \text{rank}(a)] \tag{10}$$

The metric Mean Reciprocal Rank (MRR) is caculated as $\frac{1}{\text{rank}}$ and Hits at K(Hits@Km) is calculated as $\mathbb{1}[rank < K]$. In this paper, we report the Hits@3m metric, the frequency of the correct answer to appear in top 3 ranking. The metric of a query is an average of the metrics over all its predicted answers, and the metric of a type of query is an average of the metrics over all queries of this type.

C Reproducibility

In this section, we describe the experiment setting in more details.
Label Smoothing. Label smoothing is a technique of regularization.

It utilizes soft one-hop labels instead of hard ones to add noise in the training, reduces the weights of true labels when calculating the training loss and thus prevents overfitting when training. Suppose y_h is the original true label, y_{ls} is the smoothed label, α is the parameter of label smoothing, K is the number of classes.

$$y_{ls} = (1 - \alpha) y_h + \frac{\alpha}{K} \tag{11}$$

Since the data used in pre-training is the full graph, much larger than the query data in fine-tuning, we add label smoothing only in pre-training. We used $\alpha = 0.1$ for FB15k-237 and $\alpha = 0.7$ for NELL. Note that because the data used in fine-tuning is limited, we do not add label smoothing in fine-tuning.

Sampling Parameters. We leverage several sampling methods in the two pre-training stages where sampling ratio makes a difference to the final performance. In pre-training stage 1, for FB15k-237, the ratio between meta-tree sampling and LADIES sampling is $1:1$; for NELL, we only use meta-tree sampling. In pre-training stage 2, the ratio between chain-like meta-graphs and branch-like meta-graphs can be various. We applied grid search within the range $[1:20, \ldots 1:2, 1:1, 2:1, \ldots 20:1]$ and set the ratio to be $4:1$ for FB15k-237 and $10:1$ for NELL. Besides, in the stage 1 of pre-training, for each sampled subgraphs, we keep 80% of the induced edges between nodes and limit the number of nodes in each subgraph within $[8, 16]$. In the stage 2 of pre-training, we do not add induced edges and limit the size of meta-graphs to be less than 4.
In stage 2 pre-training, we sample two patterns of small graphs for refinement, chain-like subgraphs and branch-like subgraphs. We first select a target node from all the nodes uniformly at random. For the chain-like subgraph sampling, we sample a chain by the Markov process. In each step, we sample a neighbor of the current node independently. It is allowed to sample a node more than once in a chain. For the branch-like subgraph sampling, we sample multiple neighbors of the target node. We drop those sampled graphs whose target node has no or only one neighbor and prevent sampling the same neighbors multiple times to avoid meaningless cases.

Mixture-of-Experts (MoE). We have already discussed the importance of MoE in the setting of multiple patterns reasoning especially when the FFN is sparsely activated. Note that we use Pre-LN (Pre-Layer Normalization, which refers to placing the layer normalization in the residual connections and appending an auxiliary layer normalization before the final linear decoder [42]) instead of Post-LN (Post-Layer Normalization, which refers to vanilla Transformer's design of placing the layer normalization between the residual blocks) in MoE, as evidences show that Pre-LN converges faster and more robustly compared to Post-LN [42]. For the number of experts, using 2 experts of MoE means the original setting of FFN. We applied grid search over the number of experts within $[2, 4, 8 \ldots 32, 64]$ and select 32 as the number of experts. We implement the MoE strategy using the FastMoE[2] [13], an open-source pytorch-based package for MoE with transformers.

Training Parameters. To enhance the generalization ability of our model, we add some noise to the *mask* in pre-training stage 1, following the strategies in BERT. For the nodes to be masked,

- 80% of the time, we replace the node with [mask] token;
- 10% of the time, we keep the node unchanged;
- 10% of the time, we replace the node with a random node.

Such masking ratio forces our model to keep a contextual representation for every node and thus learn the neighborhood information. We train our model with batch size 258 in pre-training and 12288 in fine-tuning. We use AdamW[23] with $lr = 1e-4, \beta_1 = 0.9, \beta_2 = 0.999$, exponential decay rate of 0.997. We also use a dropout probability of 0.1 every layer.

D Combinatorial Fine-Tuning

In the fine-tuning stage, we conduct multi-task fine-tuning followed by single-task fine-tuning. Multi-task fine-tuning refers to fine-tuning the pretrained model with all of the query-answer training sets, including $1p, 2p, 3p, 2i, 3i$. Single-task fine-tuning follows multi-task fine-tuning by tuning with only part of the query-answer training sets, either a single query's set or a combination of several

[2] https://github.com/laekov/fastmoe

Table 5: Case study on 3p examples (Interpretability). Filling a certain entity as the prediction for tail, we use kgTransformer to predict the unknown intermediate node embedding to test its interpretability. Compared to 2p, 3p queries have one more intermediate entity. Each line correspond to a back-query and the intermediate entities reacts to the change of tail entities.

Head (Ground truth)		**Intermediate 1** (Back-queried)		**Intermediate 2** (Back-queried)		**Tail** (Prediction)
TV5 (TV network)	*leader title*	President	*company with title*	Verizon	*region in*	New York
		President		eBay		California
		President		Ford Motor		Michigan
Priyanka Chopra	*gender*	Female	*risk factors*	Hypothyroidism	*risk factors*	Depression
		Female		Cirrhosis		Peritonitis
		Female		Cirrhosis		Liver failure
Clarinet	*role*	Paquito D'Rivera	*Profession*	Musician	*specialization of*	Artist
		Imogen Heap		Singer-songwriter		Musician
		Harry Shearer		Author		Writer

Table 7: Dataset statistics and details of splitting.

Dataset	#Ent.	#Rel.	#Edges	Train	Valid	Test
FB15k-237	14,505	237	310,079	272,115	17,526	20,438
NELL995	63,361	200	142,804	114,213	14,324	14,267

Table 8: Splitting statistics of query-answer dataset.

Dataset	Train		Valid		Test	
	1p	others	1p	others	1p	others
FB15k-237	149689	149689	20101	5000	22812	5000
NELL-995	107982	107982	16927	4000	16927	4000

Table 6: Preliminary experiments on combinatorial strategies in the single-task tuning after multi-task tuning on NELL995. Base case refers to the results after multi-task tuning. Bold results indicate the most effective single-task tuning method for each query. In this version not all the strategies reported in the final version are applied.

	In-domain					Out-of-domain			
	1p	2p	3p	2i	3i	ip	pi	2u	up
Base	0.605	0.345	0.277	0.373	0.521	0.164	0.252	0.428	0.265
1p	0.609	0.351	0.282	0.377	**0.523**	0.178	**0.267**	**0.432**	0.258
2p	0.613	0.355	0.278	0.379	0.515	0.166	0.258	0.432	0.269
3p	0.609	0.328	0.285	0.378	0.522	0.162	0.253	0.425	0.262
2i	0.594	0.339	0.273	**0.390**	0.521	0.155	0.249	0.427	0.254
3i	0.590	0.334	0.274	0.356	0.522	0.152	0.237	0.428	0.255
1p, 2p	0.612	**0.358**	0.281	0.368	0.499	**0.183**	0.263	0.431	**0.272**
1p, 3p	**0.615**	0.348	0.287	0.362	0.491	0.169	0.253	0.428	0.263
1p, 2p, 3p	0.611	0.351	**0.288**	0.363	0.494	0.183	0.257	0.431	0.266

query's sets. To compare different combinations in single-task fine-tuning, we just pick an arbitrary checkpoint in pretraining stage with preliminary result, do multi-task fine-tuning and then show different combinations of single-task fine-tune.

E Comparison Methods

In this section, we briefly go through the compared baselines for reference.

- **GQE [12]** utilizes deep set for intersection (conjunctive) logical operation as learned geometric functions in this space.
- **Q2B [28]** embeds queries as hyper-rectangles and answers as points inside the rectangles.
- **EmQL [34]** utilizes count-min sketch for query embedding in order to be more faithful to deductive reasoning. It focuses on deductive reasoning which does not require generalization.
- **BiQE [20]** utilizes bi-directional attention mechanism and positional embedding to capture interactions within a graph.
- **CQD [1]** uses ComplEx as one-hop reasoner and various T-norms as logic operators. It provides end-to-end differentiable objective by uniting one-hop reasoners and logic operators.

F Statistics of Datasets

We provide the information of the splitting of original datasets in Table 7 and the splitting of the query-answer datasets in Table 8.