Inductive Logical Query Answering in Knowledge Graphs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Formulating and answering logical queries is a standard communication interface for knowledge graphs (KGs) and their representations. Alleviating the notorious incompleteness of real-world KGs, neural methods achieved impressive results in link prediction and complex query answering tasks by learning representations of entities, relations, and queries. Still, most existing query answering methods are inherently transductive and cannot be generalized to KGs containing new entities without retraining entity embeddings. In this work, we study the inductive query answering task where inference is performed on a graph containing new entities with queries over both seen and unseen entities. To this end, we devise two mechanisms leveraging inductive *node* and *relational structure* representations powered by graph neural networks (GNNs). Experimentally, we show that inductive models are able to perform logical reasoning at inference time over unseen nodes generalizing to graphs up to 500% larger than training ones. Exploring the efficiency–effectiveness trade-off, we find the inductive *relational structure* method generally achieves higher performance, while the inductive *node representation* method is able to answer complex queries in the *inference-only* regime without any training on queries and scale to graphs of millions of nodes.

1 Introduction

Traditionally, querying knowledge graphs (KGs) is performed via databases using structured query languages like SPARQL. Databases can answer complex queries relatively fast under the assumption of *completeness*, i.e., there is no missing information in the graph. In practice, however, KGs are notoriously incomplete [29]. Embedding-based methods that learn vector representations of entities and relations are known to be effective in *simple link prediction* predicting heads or tails of query patterns *(head, relation, ?)*, e.g., *(Einstein, graduate, ?)*, forming the field of *KG completion* [1, 14].

Complex queries are graph patterns expressed in a subset of first-order logic (FOL) with operators such as intersection ($\wedge$), union ($\vee$), negation ($\neg$) and existentially quantified ($\exists$) variables[1], e.g., $?U.\exists V : \texttt{Win}(\texttt{NobelPrize}, V) \wedge \texttt{Citizen}(\texttt{USA}, V) \wedge \texttt{Graduate}(V, U)$ (Fig. 1). Complex queries define a superset of link prediction on KGs. The conventional link prediction task can be viewed as a complex query with a single triplet pattern without logic operators, e.g., $\texttt{Citizen}(\texttt{USA}, V)$, which we also denote as a *projection* query.

To tackle complex queries on incomplete knowledge graphs, *query embedding* methods are proposed to execute logic operations in the latent space, including variants that employ geometric [12, 20, 35], probabilistic [21, 7], neural-symbolic [23, 6, 4], neural [18, 3], and GNN [9, 2] approaches for learning entity, relation, and query representations.

[1] The universal quantifier ($\forall$) is often discarded as in real-world KGs there is no node connected to all others.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

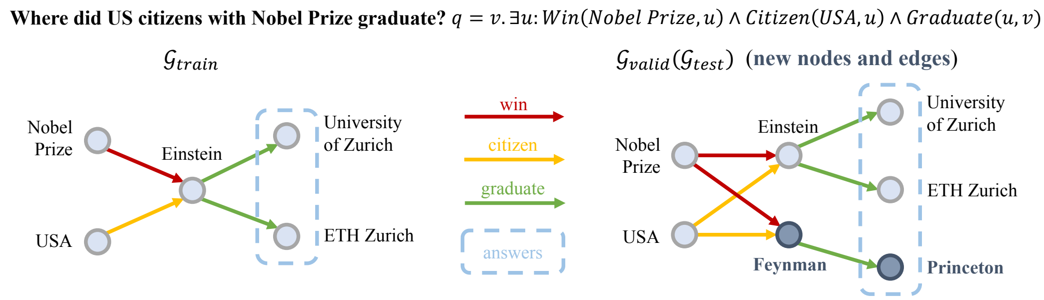

Figure 1: Inductive query answering problem: at inference time, the graph is updated with new nodes `Feynman` and `Princeton` and edges such that the same query now has more answers.

However, this very fact of learning a separate embedding for each entity makes those methods inherently *transductive* i.e., they are bound to the space of learned entities and can not be generalized to unseen entities without retraining the whole embedding matrix which can be prohibitively expensive in large graphs. The problem is illustrated in Fig. 1: given a graph about `Einstein` and a logical query *Where did US citizens with Nobel Prize graduate?*, transductive QE methods learn to execute logical operators and return the answer set {`University of Zurich, ETH Zurich`}. Then, the graph is updated with new nodes and edges about `Feynman` and `Princeton`, and the same query now has more correct answers {`University of Zurich, ETH Zurich, Princeton`} as new unseen entities satisfy the query as well.

Such *inductive inference* is not possible for transductive models as they do not have representations for new `Feynman` and `Princeton` nodes. In the extreme case, inference graphs might be disconnected from the training one and only share the set of relations. Therefore, inductive capabilities are a key factor to enable transferring trained query answering models onto updated or entirely new KGs.

In this work, we study answering complex queries in the inductive setting, where the model has to deal with unseen entities at inference time. Inspired by recent advancement in inductive link prediction on KGs [36, 10], we devise two solutions for learning inductive representations for complex query: 1) The first solution, NodePiece-QE, extends the inductive node representation model NodePiece [10] for complex query answering. NodePiece-QE learns **inductive representations of each entity** as a function of tokens from a fixed-size vocabulary, and answers complex query with a non-parametric logical query executor [4]. The advantages of NodePiece-QE are that it only needs to be trained on simple link prediction data, answers complex queries in the *inference-only* mode, and that it can scale to large KGs. 2) The second solution, NBFNet-QE, extends the inductive link prediction model NBFNet [36] for complex query answering. NBFNet-QE learns **inductive representations of the relational structure** without entity embeddings, and uses the relational structure between the query constants and the answers to make the prediction. NBFNet-QE can be trained end-to-end on complex queries, achieves much better performance than NodePiece-QE, but struggles to scale to large KGs.

To the best of our knowledge, this is the first work to study complex logical query answering in the inductive setting. Conducting experiments on a novel benchmarking suite of 10 datasets, we find that (i) both inductive solutions exhibit non-trivial performance answering logical queries over unseen entities and query patterns; (ii) inductive models demonstrate out-of-distribution generalization capabilities to graphs up to 500% larger than training ones; (iii) akin to updatable databases, inductive methods can successfully find new correct answers to known training queries after adding new nodes and edges; (iv) the inductive *node representation* method scales to answering logical queries over a graph of 2M nodes with 500k new, unseen nodes; (v) GNN-based models still exhibit difficulties [17, 32] generalizing to larger graphs than they were originally trained on.

2 Related Work

Knowledge Graph Completion. Knowledge graph completion, a.k.a. simple link prediction, has been widely studied in the *transductive* paradigm [5, 30, 24, 34], i.e., when inference is performed on the same training graph with a fixed set of entities. Generally, these methods learn a shallow embedding vector for each entity. We refer the audience to respective surveys [1, 14] covering dozens of transductive embedding methods. The emergence of message passing [11] and Graph Neural Networks (GNNs) has led to more advanced, *inductive* representation learning approaches that model entity or triplet representations as a function of the graph structure in its neighborhood. GraIL [25] learns triplet representations based on the subgraph structure surrounding the two entities. NeuralLP [31], DRUM [22] and NBFNet [36] learn the pairwise entity representations based on the set of relation paths between two entities. NodePiece [10] learns entity representations from a fixed-size vocabulary of tokens that can be anchor nodes in a graph or relation types.

Complex Query Answering. In the complex (multi-hop) query answering setup with logical operators, existing models employ different approaches, e.g., geometric [12, 20, 35], probabilistic [21, 7], neural-symbolic [23, 6, 4], neural [18, 3], and GNN [9, 2]. Still, all the approaches are created and evaluated exclusively in the transductive mode where the set of entities does not change at inference time. To the best of our knowledge, there is no related work in inductive logical query answering when an inference graph contains new entities. With our work, we aim to bridge this gap and extend inductive representation learning algorithms to logical query answering. In particular, we focus on the inductive setup where an inference graph is a superset of a training graph[2] such that (i) inference queries require reasoning over both seen and new entities; (ii) original training queries might have more correct answers at inference time with the addition of new entities.

3 Preliminaries and Problem Definition

Knowledge Graph and Inductive Setup. Given a finite set of entities $\mathcal{E}$, a finite set of relations $\mathcal{R}$, and a set of triples (edges) $\mathcal{T} = (\mathcal{E} \times \mathcal{R} \times \mathcal{E})$, a knowledge graph $\mathcal{G}$ is defined as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$. Accounting for the inductive setup, we define a *training* graph $\mathcal{G}_{train} = (\mathcal{E}_{train}, \mathcal{R}, \mathcal{T}_{train})$ and *inference* graph $\mathcal{G}_{inf} = (\mathcal{E}_{inf}, \mathcal{R}, \mathcal{T}_{inf})$ such that $\mathcal{E}_{train} \subset \mathcal{E}_{inf}$ and $\mathcal{T}_{train} \subset \mathcal{T}_{inf}$. That is, the *inference* graph extends the training graph with new entities and edges[3].The inference graph $\mathcal{G}_{inf}$ is an incomplete part of the not observable complete graph $\hat{\mathcal{G}}_{inf} = (\mathcal{E}_{inf}, \mathcal{R}, \hat{\mathcal{T}}_{inf})$ with $\hat{\mathcal{T}}_{inf} = \mathcal{T}_{inf} \cup \mathcal{T}_{pred}$ whose missing triples $\mathcal{T}_{pred}$ have to be predicted at inference time.

First-Order Logic Queries. Applied to KGs, a first-order logic (FOL) query $\mathcal{Q}$ is a formula that consists of constants $\mathcal{C}$ ($\mathcal{C} \subseteq \mathcal{E}$), variables $\mathcal{V}$ ($\mathcal{V} \subseteq \mathcal{E}$, existentially quantified), relation *projections* $R(a, b)$ denoting a binary function over constants or variables, and logic symbols ($\exists, \wedge, \vee, \neg$). The answers $A_{\mathcal{G}}(\mathcal{Q})$ to the query $\mathcal{Q}$ are assignments of variables in a formula such that the instantiated query formula is a subgraph of the complete graph $\hat{\mathcal{G}}$.

Fig. 1 illustrates the logical form of a query *Where did US citizens with Nobel Prize graduate?* as $?U.\exists V : \texttt{Win}(\texttt{NobelPrize}, V) \wedge \texttt{Citizen}(\texttt{USA}, V) \wedge \texttt{Graduate}(V, U)$ where `NobelPrize` and `USA` are *constants*; `Win`, `Citizen`, `Graduate` are *relation projections* (labeled edges); V, U - *variables* such that V is an existentially quantified free variable and U is the projected bound *target* of the query. Common for the literature, we aim at predicting assignments of the query *target* whereas assignments of intermediate variables might not always be explicitly interpreted depending on the model architecture. In the example, the answer set $A_{\mathcal{G}}(\mathcal{Q})$ is a binding of a target variable U to constants `University of Zurich` and `ETH Zurich`.

Inductive FOL Queries. In the standard transductive query answering setup, query constants and variables at both training and inference time belong to the same set of entities, i.e., $\mathcal{C}_{train} = \mathcal{C}_{inf} \subseteq \mathcal{E}, \mathcal{V}_{train} = \mathcal{V}_{inf} \subseteq \mathcal{E}$. In the inductive setup covered in this work, query constants and variables at inference time belong to a different and larger set of entities $\mathcal{E}_{inf}$ from the inference graph $\mathcal{G}_{inf}$, i.e., $\mathcal{C}_{train} \subseteq \mathcal{E}_{train}, \mathcal{V}_{train} \subseteq \mathcal{E}_{train}$ but $\mathcal{C}_{inf} \subseteq \mathcal{E}_{inf}, \mathcal{V}_{inf} \subseteq \mathcal{E}_{inf}$. This also leads to the fact that training queries executed over the inference graph might have more correct answers, i.e., $A_{\mathcal{G}_{train}}(\mathcal{Q}) \subseteq A_{\mathcal{G}_{inf}}(\mathcal{Q})$. For example (cf. Fig. 1), the inference graph is updated with new nodes `Feynman`, `Princeton` and their new respective edges. The same query now has a larger set of intermediate variables satisfying the formula (`Feynman`) and an additional correct answer `Princeton`. Therefore, inductive generalization is essential for obtaining representations of such new nodes and enabling logical reasoning over both seen and new nodes, i.e., finding more answers to known queries in larger graphs or answering new queries with new constants. In the following section, we describe two approaches for achieving inductive generalization with different parameterization strategies.

[2]The set of relation types is fixed.

[3]Note that the set of relation types $\mathcal{R}$ remains the same.

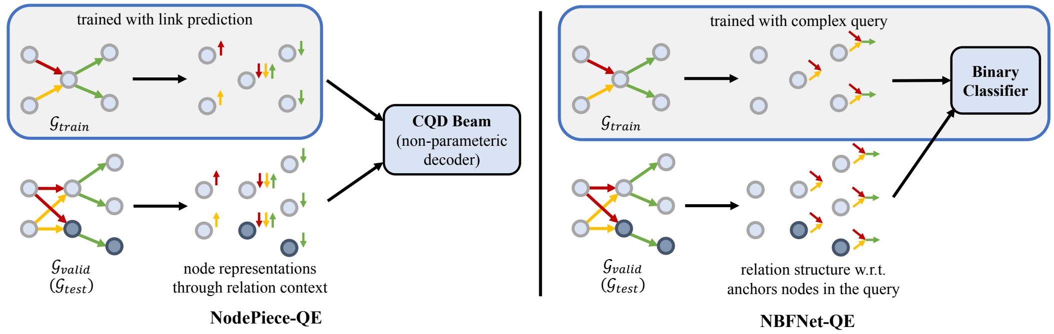

Figure 2: Inductive node representation (NodePiece-QE, left) and relational structure (NBFNet-QE, right) strategies for complex logical query answering. In NodePiece-QE, we obtain inductive node representations through the invariant set of tokens (here, through incident relation types). NodePiece-QE is the inference-only approach and is pre-trained with simple *1p* link prediction and can be directly applied to inductive complex queries with a non-parametric decoder (e.g., CQD Beam). In NBFNet-QE, we learn the the relative structure of each node w.r.t. the anchor nodes in the query. NBFNet-QE is trainable end-to-end with *complex queries*.

4 Method

Inductive Representations of Complex Queries. Given a complex query $\mathcal{Q} = (\mathcal{C}, \mathcal{R}_{\mathcal{Q}}, \mathcal{G}))$, the goal is to rank all possible entities according to the query. From a representation learning perspective, this requires us to learn a conditional representation function $f(e|\mathcal{C}, \mathcal{R}_{\mathcal{Q}}, \mathcal{G})$ for each entity $e \in \mathcal{E}$. Transductive methods learn a shallow embedding for each answer entity $e \in \mathcal{E}$, and, therefore, cannot generalize to unseen entities. For inductive methods, the function $f(e|\mathcal{C}, \mathcal{R}_{\mathcal{Q}}, \mathcal{G})$ should generalize to some unseen answer entity e' (or unseen constant entity $c' \in \mathcal{C}'$) at inference time. Here, we discuss two solutions for devising such an inductive function.

The first solution is to **parameterize the representation of each entity** e **as a function of an invariant vocabulary** of *tokens* that does not change at training and inference. Particularly, the vocabulary might consist of unique relation types $\mathcal{R}$ that are always the same for $\mathcal{G}_{train}$ and $\mathcal{G}_{inf}$, and we are able to infer the representation of an unseen answer entity (or an unseen constant entity) as a function of its incident relations (cf. Fig. 2 left). The idea has been studied in NodePiece [10] for simple link prediction. Here, we adopt a similar idea to learn inductive entity representations for complex query answering. Once we obtain the representations for unseen entities, we can use any off-the-shelf decoding method (e.g., CQD-Beam [4]) for predicting the answer to the complex query. We denote this strategy as NodePiece-QE.

The second solution is to **parameterize** $f(e|\mathcal{C}, \mathcal{R}_{\mathcal{Q}}, \mathcal{G})$ **as a function of the relational structure**. Intuitively, an answer of a complex query can be decided solely based on the relational structure between the query constants and the answer (Fig. 1). Even after anonymizing entity names (and, hence, not learning any explicit entity embedding), we are still able to infer `Princeton` as an answer since it forms a distinctive relational structure with the query constants and conforms to the query structure. Similarly, intermediate nodes will be deemed correct if they follow a relational structure . In other words, we do not need to know the answer node is `Princeton`, but only need to know the relative position of `Princeton` w.r.t. the constants like `Nobel Prize` and `USA`. Based on this idea, we design $f(e|\mathcal{C}, \mathcal{R}_{\mathcal{Q}}, \mathcal{G})$ to be a relational structure search function. Such an idea has been studied in Neural Bellman-Ford Networks (NBFNet) [36] to search for a single relation in simple link prediction. Here, we chain several NBFNet instances with differentiable logic operations to learn inductive complex query in an end-to-end fashion. We denote this strategy as NBFNet-QE.

4.1 NodePiece-QE: Inductive Node Representation

Here, we aim at reconstructing node representations for seen and new entities without learning shallow node embedding vectors. To this end, we employ NodePiece [10], a compositional tokenization approach that learns an invariant vocabulary of *tokens* shared between training and inference graphs. Formally, given a vocabulary of tokens $t_i \in T$, each entity e_i is deterministically hashed into a set of representative tokens $e_i = [t_1, \dots, t_k]$. An entity vector $\boldsymbol{e}_i$ is then obtained as a function of token embeddings $\boldsymbol{e}_i = f_\theta([\boldsymbol{t}_i, \dots, \boldsymbol{t}_k]), \boldsymbol{t}_i \in \mathbf{T}^{|T| \times d}$ where the encoder function $f_\theta : \mathbb{R}^{k \times d} \rightarrow \mathbb{R}^d$ is parameterized with a neural network θ.

Since the set of relation types $\mathcal{R}$ is invariant for training and inference graphs, we can learn relation embeddings $\mathbf{R}^{|\mathcal{R}| \times d}$ and our vocabulary of learnable tokens T is comprised of distinct relation types such that entities are hashed into a set of unique incident relation types. For example (cf. Fig. 2 left), a middle node from a training graph $\mathcal{G}_{train}$ is hashed with a set of relations $e_i = [$↓↓↑$]$ that stands for two unique incoming relations ↓↓ and one unique outgoing relation ↑. Passing the hashes through f_θ, we can reconstruct the whole entity embedding matrix $\mathbf{E}^{|\mathcal{E}_{train}| \times d}$. Additionally, it is possible to enrich entity and relation embeddings by passing them through a relational GNN encoder [28] over a target graph $\mathcal{G}$: $\mathbf{E}', \mathbf{R}' = \text{GNN}(\mathbf{E}, \mathbf{R}, \mathcal{G})$. In both ways, the entity embedding matrix $\mathbf{E}$ encodes a *joint* probability distribution $p(h, r, t)$ for all triples in a graph.

Having a uniform featurization mechanism for both seen and unseen entities, it is now possible to apply any previously-transductive complex query answering model with learnable entity embeddings and logical operators [20, 9, 21, 6]. Moreover, it was recently shown [4] that a combination of simple link prediction pre-training and a non-parametric logical executor allows to effectively answer complex FOL queries in the *inference-only* regime without training on any complex query sample. We adopt this Continuous Query Decomposition algorithm with beam search (CQD-Beam) as the main query answering decoder. CQD-Beam relies only on entity and relation embeddings $\mathbf{E}, \mathbf{R}$ pre-trained on a simple *1p* link prediction task. Then, given a complex query, CQD-Beam applies *t-norms* and *t-conorms* [16] that execute conjunctions ($\wedge$) and disjunctions ($\vee$) as non-parametric algebraic operations in the embedding space, respectively.

In our inductive setup (Fig. 2), we train a NodePiece encoder f_θ and relation embeddings $\mathbf{R}$ (and optionally a GNN) on the *1p* link prediction task over the training graph $\mathcal{G}_{train}$. We then apply the learned encoder to materialize entity representations of the inference graph $\mathbf{E}^{|\mathcal{E}_{inf}| \times d}$ and send them to CQD-Beam that performs a non-parametric decoding of complex FOL queries over new inference entities. The inference-only nature of NodePiece-QE is designed to be challenging and probing the abilities for zero-shot generalization in performing complex logical reasoning over larger graphs.

4.2 NBFNet-QE: Inductive Relational Structure Representation

The second strategy relies on learning inductive relational structure representations instead of explicit node representations. Having the same set of relation types $\mathcal{R}$ at training and inference time, we can parameterize each entity based on the relative relational structure between it and the anchor nodes in a given query. For instance (Fig. 2 right), given a query with a particular relational structure and a set of anchor nodes, the representation of each node captures its relational structure relative to the anchor nodes. Each neighborhood expansion step is equivalent to the *projection* step. In our example, immediate neighboring nodes will capture the intersection pattern , and further nodes, in turn, capture the extended *intersection-projection* structure .

Therefore, a node is likely to be an answer if its captured (or predicted) relational structure conforms with the query relational structure. As long as the set of relations is fixed, relation *projection* is performed in the same way for training or new unseen nodes. The idea of a one-hop (*1p*) projection for simple link prediction has been proposed by Neural Bellman-Ford Networks (NBFNet) [36].

In particular, given a relation *projection* query $(h, r, ?)$, NBFNet assigns unique initial states $\boldsymbol{h}^{(0)}$ to all nodes in a graph by applying an indicator function $\boldsymbol{h}_e^{(0)} = \textsc{Indicator}(h, v, r)$, i.e., a head node h is initialized with a learnable relation embedding $\boldsymbol{r}$ and all other nodes are initialized with zeros. Then, NBFNet applies L relational message passing GNN layers where each layer l has its own learnable relation embedding matrix $\mathbf{R}_l$ obtained as a projection of the initial relation $\mathbf{R}_l = \mathbf{W}_l \boldsymbol{r} + \boldsymbol{b}_l$. Final layer representations $\boldsymbol{h}^{(L)}$ are passed through an MLP and activation function σ to get a probability distribution over all nodes in a graph $p(t|h, r) = \sigma(\text{MLP}(\boldsymbol{h}^{(L)}))$. As each query spawns a uniquely initialized graph and message passing procedure, NBFNet is seen to be applying a *labeling trick* [33] to model a conditional probability distribution $p(t|h, r)$ which is provably more expressive than a joint distribution $p(h, r, t)$ produced by standard graph encoders.

Applied to complex queries, chaining k NBFNet instances allows to answer k-hop projection queries, e.g., two instances for *2p* queries. NBFNet-QE employs NBFNet as a *trainable* projection operator and endows it with differentiable, non-parametric *product* logic for modeling conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$) over the *fuzzy sets* of all entities $\boldsymbol{x} \in [0,1]^{\mathcal{E}}$, i.e., after applying a logical operator (discussed in Appendix A), each entity's degree of truth is associated with a scalar in range $[0,1]$. For the next hop projection, the indicator function initializes a node state with a relation vector $\boldsymbol{r}_i$ weighted by a scalar probability predicted in the previous hop x_e: $\boldsymbol{h}_e^{(0)} = x_e \boldsymbol{r}_i$. Differentiable logical operators allow training NBFNet-QE end-to-end on complex queries.

5 Experiments

We designed the experimental agenda to demonstrate that inductive representation strategies are able to: (1) answer complex logical queries over new, unseen entities at inference time, i.e., when query anchors are new nodes (Section 5.2); (2) predict new correct answers for known *training* queries when executed over larger inference graphs, i.e., when query anchors come from the training graph but variables and answers belong to the larger inference graph (Section 5.3); (3) generalize to inference graphs of up to 500% larger than training graphs; (4) scale to inductive query answering over graphs of millions of nodes when updated with 500k new nodes and 5M new edges (Section 5.4).

5.1 Setup & Dataset

Dataset. Due to the absence of inductive logical query benchmarks, we create a novel suite of datasets[4] based on FB15k-237 [26] (open license) and following the BetaE [21] query sampling methodology. Given a source graph with $\mathcal{E}$ entities, we sample $|\mathcal{E}_{train}| = r \cdot |\mathcal{E}|, r \in [0.1, 0.9]$ nodes to induce a training graph $\mathcal{G}_{train}$. For validation and test graphs, we split the remaining set of entities into two non-overlapping sets each with $\frac{1-r}{2}|\mathcal{E}|$ nodes. We then merge training and unseen nodes into the inference set of nodes $\mathcal{E}_{inf}$ and induce inference graphs for validation and test from those sets, respectively, i.e., $\mathcal{E}_{inf}^{val} = \mathcal{E}_{train} \cup \mathcal{E}_{val}$ and $\mathcal{E}_{inf}^{test} = \mathcal{E}_{train} \cup \mathcal{E}_{test}$. That is, validation and test inference graphs both extend the training graph but their sets of new entities are disjoint. Finally, we sample and remove 15% of edges $\mathcal{T}_{\text{pred}}$ in the inference graphs as missing edges for link prediction pre-training and query sampling. Overall, we sample 9 such datasets varying r to obtain ratios of inference graph size to the training graph $\mathcal{E}_{inf}/\mathcal{E}_{train}$ from 105% to 550%.

For each dataset, we employ the query sampler from BetaE [21] to extract 14 typical query types *1p/2p/3p/2i/3i/ip/pi/2u/up/2in/3in/inp/pin/pni*. Training queries are sampled from the training graph $\mathcal{G}_{train}$, validation and test queries are sampled from their respective inference graphs $\mathcal{G}_{inf}$ where at least one edge belongs to $\mathcal{T}_{\text{pred}}$ and has to be predicted at inference time.

As inference graphs extend training graphs, training queries are very likely to have new answers being executed over $\mathcal{G}_{inf}$ with simple graph traversal and without any link prediction. We create an additional set of true answers for all training queries executed over the test inference graph $\mathcal{G}_{inf}^{test}$ to measure the generalization capabilities of query answering models. This is designed to be an inference task and extends the *faithfullness* experiment [23]. Dataset statistics can be found in Appendix B.

Evaluation Protocol. Following the literature [21], query answers are separated into two sets: *easy answers* that only require graph traversal over existing edges, and *hard answers* that require inferring missing links to achieve the answer node. For the main experiment, evaluation involves ranking of *hard* answers against all entities having easy ones filtered out. For evaluating training queries on inference graphs, we only have *easy* answers and rank them against all entities. We report Hits@10 as the main performance metric on different query types.

Implementation Details. All NodePiece [10]-based models were pre-trained until convergence on a simple *1p* link prediction task with the relations-only vocabulary and entity tokenization, MLP encoder, and ComplEx [27] scoring function. We used a 2-layer CompGCN [28] as an optional message passing encoder on top of NodePiece features. The non-parametric CQD-Beam [4] decoder for answering complex queries is tuned for each query type based on the validation set of queries, most of the setups employ a *product t-norm*, sigmoid entity score normalization, and beam size of 32. Following the literature, the NBFNet-QE models were trained on 10 query patterns (*1p/2p/3p/2i/3i/2in/3in/inp/pin/pni*) where *ip/pi/2u/up* are only seen at inference time. Each model employs a 4-layer NBFNet [36] as a trainable projection operator with DistMult [30] composi-

[4] Available to reviewers, will be published under the CC0 license

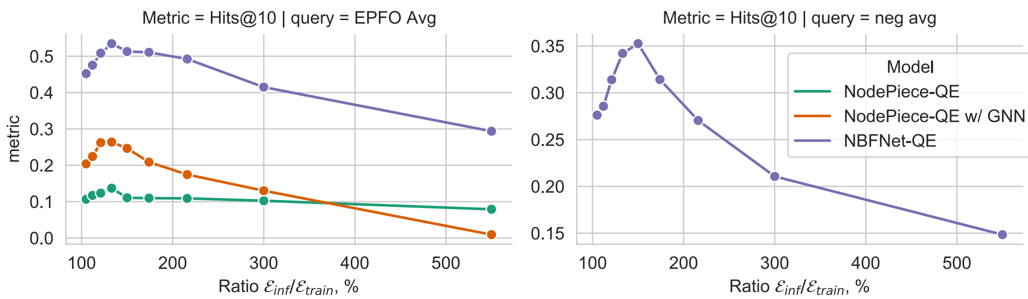

Figure 3: Aggregated Hits@10 performance of **test queries** (involving unseen entities) executed on inference graphs of different ratios compared to training graphs. NodePiece-based models are *inference-only* and support EPFO queries, NBFNet-QE is trainable and supports negation queries.

Table 1: Test Hits@10 results (%) on answering inductive FOL queries when $\mathcal{E}_{inf}/\mathcal{E}_{train} = 175\%$. avg$_p$ is the average on EPFO queries ($\wedge$, $\vee$). avg$_n$ is the average on queries with negation.

Model	avg$_p$	avg$_n$	1p	2p	3p	2i	3i	pi	ip	2u	up	2in	3in	inp	pin	pni
Inference-only																
NodePiece-QE	11.0	-	21.3	8.9	5.1	13.0	14.7	9.8	8.7	9.7	7.5	-	-	-	-	-
NodePiece-QE w/ GNN	20.9	-	34.4	15.6	10.5	28.7	33.4	19.2	16.2	17.8	12.2	-	-	-	-	-
Trainable																
NBFNet-QE	51.1	31.4	66.1	40.9	31.2	73.0	83.3	58.3	41.3	37.8	27.8	31.1	44.3	28.4	25.2	28.0

tion function and PNA [8] aggregation. Other logical operators ($\wedge, \vee, \neg$) are executed with the non-parametric *product t-norm*. Both NodePiece-QE and NBFNet-QE are implemented[5] with PyTorch [19] and trained with the Adam [15] optimizer. NodePiece-QE models were pre-trained and evaluated on a single Tesla V100 32 GB GPU whereas NBFNet-QE models were trained and evaluated on 4 Tesla V100 16GB. All hyperparameters are listed in Appendix D.

5.2 Complex Query Answering over Unseen Entities on Differently Sized Inference Graphs

First, we probe *inference-only* NodePiece-based embedding models and *trainable* NBFNet-QE in the inductive setup, i.e., query answering over unseen nodes requiring link prediction over unseen nodes. Table 1 summarizes the results on a reference dataset with ratio $\mathcal{E}_{inf}/\mathcal{E}_{train}$ of 175% while Fig. 3 illustrates a bigger picture on all datasets (we provide a detailed breakdown by query type for all splits in Appendix C). We observe that even inference-only models pre-trained solely on simple *1p* link prediction exhibit non-trivial performance in answering queries with unseen entities. Paired with an additional GNN encoder, the inference-only baseline exhibits significantly better performance over all query types and inference graphs up to 300% larger than training graphs.

The trainable NBFNet-QE models expectedly outperform non-trainable baselines and can tackle queries with negation ($\neg$). Here, we confirm that the *labeling trick* [33] and conditional $p(t|h, r)$ modeling better capture the relation projection problem than joint $p(h, r, t)$ encoding approaches.

Still, all evaluated models with message passing, both inference-only NodePiece-QE with GNN and trainable NBFNet-QE, suffer from increasing the size of the inference graph and having more unseen entities. Reaching best results on $\mathcal{E}_{inf}/\mathcal{E}_{train}$ ratios around 130%, both approaches steadily deteriorate up until final 550% by 20 absolute Hits@10 points on EPFO queries and negation queries. We attribute this deterioration to the known generalization issues [17, 32] of message passing GNNs when performing inference over much larger graph than the network has seen during training. On the other hand, a simple NodePiece-QE model without message passing retains similar performance independently of the inference graph size.

Lastly, we observe that lower performance of inference-only NodePiece models can be also attributed to underfitting (cf. train graph charts in Fig. 4). Although *1p* link predictors were trained until convergence (on the inductive validation set of missing triples), the performance of training queries

[5] Source code is available to reviewers

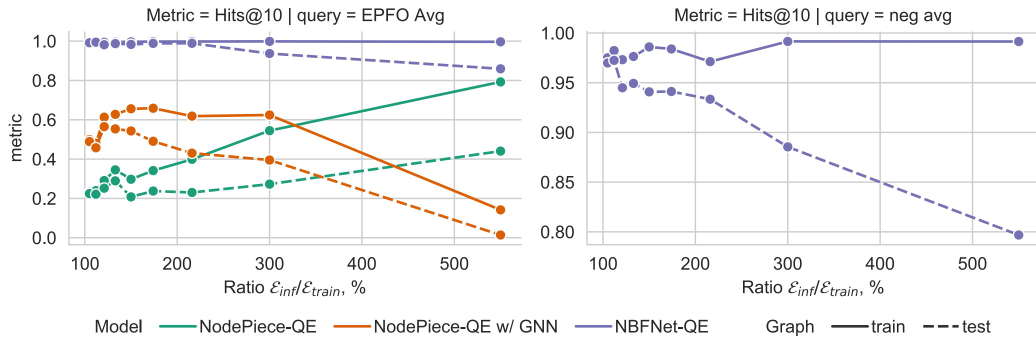

Figure 4: Aggregated Hits@10 performance of **training queries** on the original training and extended test inference graphs where queries have new correct answers. NodePiece-based models are *inference-only* and support EPFO queries, NBFNet-QE is trainable and supports negation queries.

on training graphs with *easy answers* that require only relation traversal without predicting missing edges is not yet saturated. This fact suggests that better fitting entity featurization (obtained by NodePiece or other strategies) could further improve the test performance in the inference-only regime. We leave the search of such strategies for future work.

5.3 Predicting New Answers for Training Queries on Larger Inference Graphs

Simulating the incremental addition of new edges in graph databases, we evaluate the performance of our inference-only and trainable QE models on *training* queries on the original training graph and extended inference graph (with added test edges). As databases are able to immediately retrieve new answers to known queries after updating the graph, we aim at exploring and quantifying this behaviour of neural reasoning models. In this experiment, we probe training queries and their *easy answers* that require performing only graph traversal without predicting missing links in the inference graph. While execution of training queries over the *training* graph indicates how well the model could fit training data, executing training queries over the bigger *inference* graph with new entities aims to capture basic reasoning capabilities of QE models in the inductive regime.

Particular challenges arising when executing training queries over a bigger graph are: (1) the same queries can have more correct answers as more new nodes and edges satisfying the query pattern might have been added (as in Fig. 1); (2) more new entities create a “distractor” setting with more false positives. Generally, evaluation of training queries on the inference graph can be considered as an extended version of the *faithfullness* [23] evaluation that captures how well a trained model can answer original training queries, i.e., memorization capacity. In all 9 datasets, most of training queries have at least one new correct answer in the inference graph (more details in Appendix B).

Fig. 4 illustrates the performance of the inference-only NodePiece-QE (without and with GNN) and trainable NBFNet-QE. Generally, NBFNet-QE fits the training query data almost perfectly confirming the original finding [36] that NBFNet can perform graph traversal akin to symbolic rule-based models. NBFNet-QE can also find new correct answers on graphs up to 300% larger than training ones. Then, the performance quickly deteriorates which we attribute to the *distractor* factor with more unseen entities and the already mentioned generalization issue on larger inference graphs.

The inference-only NodePiece-QE models, as expected, do not fully fit the training data as they were never trained on complex queries. Still, the inference-only models exhibit non-trivial performance in finding more answers on graphs up to 200% larger than training ones with relatively small performance margins compared to training queries. The most surprising observation is that GNN-free NodePiece-QE models improve the performance on both training and inference graphs as the graphs (and the $\mathcal{E}_{inf}/\mathcal{E}_{train}$ ratio) grow larger while GNN-enriched models steadily deteriorate. We attribute this growth to the relation-based NodePiece tokenization and its learned features that tend to be more discriminative in larger inference graphs where new nodes have smaller degree and thus can be better identified by their incident relation types. We provide more experimental results for each dataset ratio with breakdown by query type in Appendix C.

5.4 Scaling to Millions of Nodes on WikiKG-QE

Finally, we perform a scalability experiment evaluating complex query answering in the inductive mode on a new large dataset *WikiKG-QE* constructed from OGB WikiKG 2 [13] (CC0 license). While the original task is transductive link prediction, we split the graph into a training graph of 1.5M entities (5.8M edges, 512 unique relation types) and validation (test) graphs of 500k unseen nodes (5M known and 600k missing edges) each. The resulting validation (test) inference graphs are therefore of 2M entities and 11M edges with the $\mathcal{E}_{inf}/\mathcal{E}_{train}$ ratio of 133% (details are in Appendix B).

None of GNN-enabled models can scale to such sizes, so we use a basic inference-only NodePiece-QE. Due to the problem size, we only sample 10k EPFO queries of each type from the *test inference* graph to run in the inference-only regime. Each query has at least one missing edge to be predicted at inference. The answers are ranked against all 2M entities in the filtered setting (in contrast to the OGB task that ranks against 1000 pre-computed negative samples) and Hits@100 as the target metric.

We pre-train a NodePiece encoder (in addition to relation types, we tokenize nodes with a vocabulary of 20k anchor nodes, total 3M parameters in the encoder) with the ComplEx decoder on *1p* link prediction over the training graph for 1.5M steps (see Appendix D for hyperparameters). Then, the graph is extended with 500k new nodes and 5M new edges forming the inference graph. Then, using the pre-trained encoder, we materialize representations of entities (both seen and new) and relations from this inference graph. Finally, CQD-Beam executes the queries against the bigger inference graph extended with 500k new nodes and 5M new edges.

Table 2: Test Hits@100 of NodePiece-QE on WikiKG-QE (2M nodes, 11M edges including 500k new nodes and 5M new edges) in the inference-only regime. avg_p is the average on EPFO queries.

Model	**avg$_p$**	**1p**	**2p**	**3p**	**2i**	**3i**	**pi**	**ip**	**2u**	**up**
NodePiece-QE	6.6	19.0	2.1	1.8	11.0	15.8	4.4	2.8	1.3	1.5

As shown in Table 2, we find a non-trivial performance of the inference-only model on EPFO queries demonstrating that inductive *node representation* QE models are able to scale to graphs with hundreds of thousands of new nodes and millions of new edges in the zero-shot fashion. That is, answering complex queries over unseen entities is available right upon updating the graph without the need to retrain a model. This fact paves the way for the concept of *neural graph databases* capable of performing zero-shot inference over updatable graphs without expensive retraining.

6 Limitations and Future Work

Limitations. With the two proposed inductive query answering strategies, we observe a common trade-off between the performance and computational complexity. That is, inductive *node representation* models like NodePiece-QE are fast, scalable, and can be executed in the inference-only regime but underperform compared to the inductive *relational structure representation* models like NBFNet-QE. On the other hand, NBFNet-QE incurs high computational costs due to executing each query on a uniquely initialized graph instance. Alleviating this issue is a key to scalability.

Societal Impact. The inductive setup assumes running inference on (partly) unseen data, that is, the nature of this unseen data might be out-of-distrbution, unknown and potentially malicious. This fact has to be taken into account when evaluating predictions and overall system trustworthiness.

Conclusion and Future Work. In this work, we defined the problem of inductive complex logical query answering and proposed two possible parameterization strategies based on *node* and *relational structure* representations to deal with new, unseen entities at inference time. Experiments demonstrated that both strategies are able to answer complex logical queries over unseen entities as well as identify new answers on larger inference graphs. In the future work, we plan to extend the inductive setup to completely disjoint training and inference graphs, expand the set of supported logical query patterns aligned with popular queries over real-world KGs, enable reasoning over continuous features like texts and numbers, support more KG modalities like hypergraphs and hyper-relational graphs, and further explore the concept of neural graph databases.

References

[1] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann. Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[2] Dimitrios Alivanistos, Max Berrendorf, Michael Cochez, and Mikhail Galkin. Query embedding on hyper-relational knowledge graphs. In *International Conference on Learning Representations*, 2022.

[3] Alfonso Amayuelas, Shuai Zhang, Xi Susie Rao, and Ce Zhang. Neural methods for logical reasoning over knowledge graphs. In *International Conference on Learning Representations*, 2022.

[4] Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. Complex query answering with neural link predictors. In *International Conference on Learning Representations*, 2021.

[5] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795, 2013.

[6] Xuelu Chen, Ziniu Hu, and Yizhou Sun. Fuzzy logic based logical query answering on knowledge graph. In *International Conference on Machine Learning*. PMLR, 2021.

[7] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. Probabilistic entity representation model for reasoning over knowledge graphs. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[8] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271, 2020.

[9] Daniel Daza and Michael Cochez. Message passing query embedding. *arXiv preprint arXiv:2002.02406*, 2020.

[10] Mikhail Galkin, Etienne Denis, Jiapeng Wu, and William L. Hamilton. Nodepiece: Compositional and parameter-efficient representations of large knowledge graphs. In *International Conference on Learning Representations*, 2022.

[11] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017.

[12] Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. *Advances in Neural Information Processing Systems*, 31, 2018.

[13] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[14] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2022.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[16] Erich-Peter Klement, Radko Mesiar, and Endre Pap. Triangular norms. position paper I: basic analytical and algebraic properties. *Fuzzy Sets Syst.*, 143(1):5–26, 2004.

[17] Boris Knyazev, Graham W. Taylor, and Mohamed R. Amer. Understanding attention and generalization in graph neural networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4204–4214, 2019.

[18] Bhushan Kotnis, Carolin Lawrence, and Mathias Niepert. Answering complex queries in knowledge graphs with bidirectional sequence encoders. *CoRR*, abs/2004.02596, 2020.

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019.

[20] Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *International Conference on Learning Representations*, 2019.

[21] Hongyu Ren and Jure Leskovec. Beta embeddings for multi-hop logical reasoning in knowledge graphs. *Advances in Neural Information Processing Systems*, 33, 2020.

[22] Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. Drum: End-to-end differentiable rule mining on knowledge graphs. *Advances in Neural Information Processing Systems*, 32, 2019.

[23] Haitian Sun, Andrew O. Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W. Cohen. Faithful embeddings for knowledge base queries. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[24] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[25] Komal K. Teru, Etienne Denis, and Will Hamilton. Inductive relation prediction by subgraph reasoning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9448–9457. PMLR, 2020.

[26] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China, July 2015. Association for Computational Linguistics.

[27] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR, 2016.

[28] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*, 2020.

[29] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In Chin-Wan Chung, Andrei Z. Broder, Kyuseok Shim, and Torsten Suel, editors, *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, pages 515–526. ACM, 2014.

[30] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[31] Fan Yang, Zhilin Yang, and William W. Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2319–2328, 2017.

[32] Gilad Yehudai, Ethan Fetaya, Eli A. Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 11975–11986. PMLR, 2021.

[33] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 9061–9073. Curran Associates, Inc., 2021.

[34] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. Quaternion knowledge graph embeddings. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 2731–2741, 2019.

[35] Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu. Cone: Cone embeddings for multi-hop reasoning over knowledge graphs. *Advances in Neural Information Processing Systems*, 34, 2021.

[36] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 34, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code and sample data are included in the supplementary material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Dataset creation process is described in Section 5.1 with more details in Appendix B. Hyperparameters are specified in Appendix D.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We observe negligible variance w.r.t. random seeds
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Training details are specified in Section 5.1 and in Appendix D.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Due to the overall size, we include a sample of the benchmarking suite in the supplemental material and will openly publish the whole dataset.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] No personal data involved
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] The datasets are anonymized, we discuss it in Appendix B.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]