Explainable Link Prediction for Emerging Entities in Knowledge Graphs

Rajarshi Bhowmik[1] and Gerard de Melo[2]

[1] Rutgers University–New Brunswick
Piscataway, New Jersey, USA
[2] Hasso Plattner Institute, University of Potsdam
Potsdam, Germany
rajarshi.bhowmik@rutgers.edu; gdm@demelo.org

Abstract. Despite their large-scale coverage, cross-domain knowledge graphs invariably suffer from inherent incompleteness and sparsity. Link prediction can alleviate this by inferring a target entity, given a source entity and a query relation. Recent embedding-based approaches operate in an uninterpretable latent semantic vector space of entities and relations, while path-based approaches operate in the symbolic space, making the inference process explainable. However, these approaches typically consider static snapshots of the knowledge graphs, severely restricting their applicability for evolving knowledge graphs with newly emerging entities. To overcome this issue, we propose an inductive representation learning framework that is able to learn representations of previously unseen entities. Our method finds reasoning paths between source and target entities, thereby making the link prediction for unseen entities interpretable and providing support evidence for the inferred link.

Keywords: Explainable link prediction · Emerging entities · Inductive representation learning

1 Introduction

Recent years have seen a surge in the usage of large-scale cross-domain knowledge graphs [17] for various tasks, including factoid question answering, fact-based dialogue engines, and information retrieval [2]. Knowledge graphs serve as a source of background factual knowledge for a wide range of applications [6]. For example, Google's knowledge graph is tightly integrated into its search engine, while Apple adopted Wikidata as a source of background knowledge for its virtual assistant Siri. Many such applications deal with queries that can be transformed to a structured relational query of the form $(e_\mathrm{s}, r_\mathrm{q}, ?)$, where e_s is the source entity and r_q is the query relation. For example, the query "*Who is the director of World Health Organization?*" can be mapped to the structured query *(World Health Organization, director, ?)* while executing it on a knowledge graph. Unfortunately, due to the inherent sparsity and incompleteness of knowledge graphs, answers to many such queries cannot be fetched directly from the existing data, but instead need to be inferred indirectly.

Furthermore, with the ever-increasing volume of the knowledge graphs, the number of emerging entities also increases. Many of these emerging entities have a small number of known facts at the time they are integrated into the knowledge graphs. Therefore, their connectivity to pre-existing entities in the knowledge graph is often too sparse.

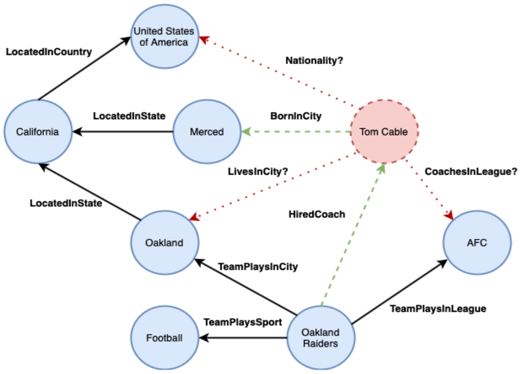

Fig. 1. A subgraph of NELL with *Tom Cable* as an emerging entity. The solid-lined circles and arrows represent the existing entities and relations. The dashed-lined circles and arrows denote an emerging entity and some of its known relationships to other existing entities. The unknown relationships that need to be inferred through inductive representation learning and explainable reasoning are shown as dotted arrows.

In recent years, embedding-based models [28] have widely been adopted to infer missing relationships in a knowledge graph. In such embedding-based models, distributed vector representations of entities and relations in the knowledge graph are used to learn a scoring function $f(e_\mathrm{s}, r_\mathrm{q}, e_\mathrm{o})$ in a latent embedding space to determine the plausibility of inferring a new fact. However, these models are lacking in terms of the interpretability and explainability of the decisions they make. One does not obtain any clear explanation of why a specific inference is warranted. For example, from the embeddings of facts *(A, born_in, California)* and *(California located_in, US)*, the fact *(A, born_in, US)* could be deduced. But logical composition steps like this one are learned implicitly by knowledge graph embeddings. This means that this approach cannot offer such logical inference paths as support evidence for an answer.

In contrast, path-based reasoning approaches operate in the symbolic space of entities and relations, leveraging the symbolic compositionality of the knowledge graph relations, thus making the inference process explainable. This means that the user can inspect the inference path, consisting of existing edges in the knowledge graph, as support evidence. To this end, purely symbolic and fast rule-mining systems, e.g., PRA [22], AMIE+ [9], and AnyBURL [25] may attain a level of performance that is comparable to embedding-based methods, but neglect many of the statistical signals exploited by the latter. To leverage the advantages of both path-based and embedding-based models, some neural-symbolic approaches [10, 11, 26, 13, 16] have as well been proposed. Some recent path-based reasoning approaches [4, 23] formulate the path-finding problem as a Partially Observable Markov Decision Process (POMDP), in which the model learns a policy to find an inference path from the source entity to the target entity using REINFORCE [41], a policy gradient based reinforcement learning algorithm.

However, most of these approaches are studied with static snapshots of the knowledge graphs, thus severely restricting their applicability for a dynamically evolving knowledge graph with many emerging entities. Except for the purely symbolic rule-mining systems mentioned above, most existing approaches that depend on learning latent representations of entities require that all entities are present during training. Therefore, these models are incapable of learning representations of arbitrary newly emerging entities not seen during training. Some recent approaches such as HyTE [5] and DyRep [34] have considered dynamically evolving temporal knowledge graphs. However, similar to embedding-based models, these models are not explainable.

To overcome this issue, we propose a joint framework for representation learning and reasoning in knowledge graphs that aims at achieving inductive node representation learning capabilities applicable to a dynamic knowledge graph with many emerging entities while preserving the unique advantage of the path-based approaches in terms of explainability. For inductive node representation learning, we propose a variant of *Graph Transformer* encoder [21, 47] that aggregates neighborhood information based on its relevance to the query relation. Furthermore, we use policy gradient-based reinforcement learning (REINFORCE) to decode a reasoning path to the answer entity. We hypothesize that the inductively learned embeddings provide prior semantic knowledge about the underlying knowledge environment to the reinforcement learning agent.

We summarize the contributions of this paper as follows: (1) We introduce a joint framework for inductive representation learning and explainable reasoning that is capable of learning representations for unseen emerging entities during inference by leveraging only a small number of known connections to the other pre-existing entities in the knowledge graph. Our approach can not only infer new connections between an emerging entity and any other pre-existing entity in the knowledge graph, but also provides an explainable reasoning path as support evidence for the inference. (2) We introduce new train/development/test set splits of existing knowledge graph completion benchmark datasets that are appropriate for inductive representation learning and reasoning.

2 Related Work

2.1 Embedding-based Methods

Knowledge graph completion can be viewed as an instance of the more general problem of link prediction in a graph [39]. Due to advances in representation learning, embedding-based methods have become the most popular approach. Such methods learn d-dimensional distributed vector representations of entities and relations in a knowledge graph. To this end, the translation embedding model TransE [3] learns the embedding of a relation as a simple translation vector from the source entity to the target entity such that $\mathbf{e}_s + \mathbf{e}_r \approx \mathbf{e}_o$. Its variants, e.g., TransH [40], TransR [24], TransD [18] consider similar objectives. Tri-linear models such as DistMult [45], along with its counterpart ComplEx [35] in the complex embedding space, use a multiplicative scoring function $f(s, r, o) = \mathbf{e}_s^{\intercal} \mathbf{W}_r \mathbf{e}_o$, where $\mathbf{W}_r$ is a diagonal matrix representing the embedding of relation r. Convolutional neural network models such as ConvE [7] and ConvKB [27] apply convolutional kernels over entity and relation embeddings to capture the interactions among them across different dimensions. These models obtain state-of-the-art results on the benchmark KBC datasets. However, none of the above-mentioned approaches deliver the full reasoning paths that license specific multi-hop inferences, and hence they either do not support multi-hop inference or do not support it in an interpretable manner. Moreover, these approaches assume a static snapshot of the knowledge graph to train the models and are not straightforwardly extensible to inductive representation learning with previously unseen entities.

2.2 Path-based Methods

An alternative stream of research has explored means of identifying specific paths of inference, which is the task we consider in this paper. To this end, the Path Ranking Algorithm (PRA) [22] uses random walks with restarts for multi-hop reasoning. Following PRA, other approaches [10, 11, 26, 13] also leverage random walk based inference. However, the reasoning paths that these methods follow are gathered by random walks independently of the query relation.

Recent approaches have instead adopted policy gradient based reinforcement learning for a more focused exploration of reasoning paths. Policy gradient based models such as DeepPath [44], MINERVA [4], MINERVA with Reward Shaping and Action Dropout [23], and M-Walk [31] formulate the KG reasoning task as a Partially Observable Markov Decision Process and learn a policy conditioned on the query relation. Such reasoning techniques have also been invoked for explainable recommendation [42, 8, 43] and explainable dialogue systems [46]. Although the inference paths are explainable in these models (if reward shaping is omitted), there may be a substantial performance gap in comparison with embedding-based models.

Another sub-category of path-based methods, e.g., AMIE+ [9], AnyBURL [25], and RuleS [16] proceed by mining Horn rules from the the existing knowledge graphs for link prediction. The body of a Horn rule provides the reasoning path. Although these approaches are capable of fast rule mining and can easily be applied to unseen emerging entities, the quality of the learned rules are affected by the sparsity of the knowledge graph.

2.3 Graph Convolution-based Methods

Graph Convolution Networks (GCNs) can be used for node classification in a homogeneous graph [20]. They are an instance of Message Passing Neural Networks (MPNN), in which the node representations are learned by aggregating information from the nodes' local neighborhood. GraphSAGE [15] attempts to reduce the memory footprint of GCN by random sampling of the neighborhood. Graph Attention Networks (GAT) [38] are a variant of GCN that learn node representations as weighted averages of the neighborhood information. However, GCN and its variants such as GAT and GraphSAGE are not directly applicable for link prediction in knowledge graphs, as they ignore the edge (relation) information for obtaining the node embeddings. To alleviate this issue, R-GCNs operate on relational multi-graphs [29], but, similar to GCNs, R-GCNs also need all nodes of the graphs to be present in memory and therefore are not scalable to large-scale knowledge graphs. Hamaguchi et al. [14] proposed a model for computing representations for out-of-KG entities using graph neural networks. The recent models such as SACN [30] and CompGCN [36] leverage the graph structure by inductively learning representations for edges (relations) and nodes (entities). However, unlike our model, these methods are not explainable.

3 Model

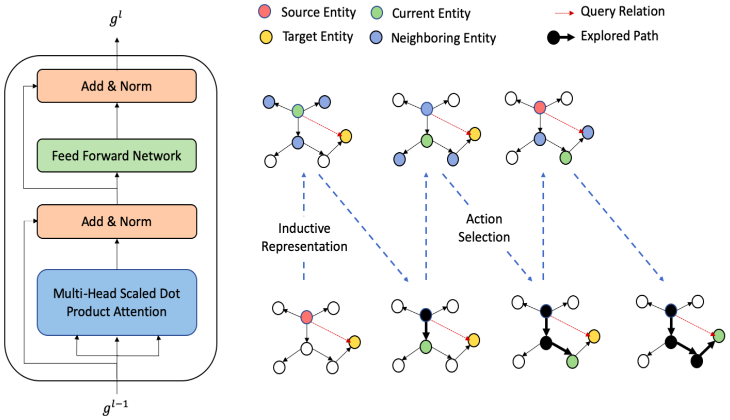

Fig. 2. A schematic diagram of a Graph Transformer block, along with an illustration of the workflow of our model, demonstrating successive applications of inductive node representation learning and action selection to find a reasoning path.

Our model consists of two modules that are subject to joint end-to-end training. The encoder module learns inductive entity embeddings while accounting for the query relation and the local neighborhood of an entity (Section 3.2). The decoder module operates on this learned embedding space of entities and relations. By leveraging the embeddings of the source entity and the query relation, the decoder module infers a reasoning path to the target entity using policy gradient-based reinforcement learning (Section 3.3). Before describing these components in more detail, Section 3.1 first provides preliminary definitions.

3.1 Problem Statement

Formally, we consider knowledge graphs $\mathcal{G}(\mathcal{E}, \mathcal{R}, \mathcal{F})$ defined as directed multi-graphs such that each node $e \in \mathcal{E}$ represents an entity, each $r \in \mathcal{R}$ represents a unique relation, and each directed edge $(e_{\mathrm{s}}, r, e_{\mathrm{o}}) \in \mathcal{F}$ represents a fact about the subject entity e_{s}.

Given a structured relational query $(e_{\mathrm{s}}, r_{\mathrm{q}}, ?)$, where e_{s} is the source entity, r_{q} is the query relation, and $(e_{\mathrm{s}}, r_{\mathrm{q}}, e_{\mathrm{o}}) \notin \mathcal{F}$, the goal is to find a set of plausible answer entities $\{e_{\mathrm{o}}\}$ by navigating paths through the existing entities and relations in $\mathcal{G}$ leading to answer entities. Note that, unlike previous methods that consider transductive settings with a static snapshot of the knowledge graph, we allow for dynamic knowledge graphs, where e_{s} may be an emerging entity, and therefore, previously unseen. Moreover, while embedding-based methods only deliver candidate answer entities, we here also seek the actual paths, i.e., sequences of nodes and edges for better interpretability.[3]

3.2 Graph Transformer for Inductive Representation Learning

The state-of-the-art embedding based models either focus on learning entity embeddings by using only the query relations, ignoring the subject entity's neighborhood, or use message passing neural networks to learn embeddings conditioned on neighboring entities and relations while being oblivious of the query relation. However, we observe that in many cases a new fact can be inferred by using another existing fact. For example, the fact *(PersonX, Place of Birth, Y)* can often help to answer to the query *(PersonX, Nationality, ?)*. Motivated by this observation, we propose a Graph Transformer architecture that learns the embedding of the source entity by iterative aggregation of neighborhood information (messages) that are weighted by their relevance to the query relation. To learn the relevance weights, our Graph Transformer model deploys *multi-head scaled dot product attention*, also known as *self-attention* [37].

Formally, we denote the local neighborhood for each entity $e_i \in \mathcal{E}$ as $\mathcal{N}_i$ such that $\mathcal{N}_i = \{e_j \mid e_j \in \mathcal{E} \wedge (e_i, r, e_j) \in \mathcal{F} \wedge r \in \mathcal{R}_{ij}\}$, where $\mathcal{R}_{ij}$ is the set of relations between entities e_i and e_j.

Each neighboring entity $e_j \in \mathcal{N}_i$ connected to e_i by a relation r sends in a message to entity e_i. The message $\mathbf{m}_{ijr}$ is a linear transformation of the fact (e_i, r, e_j) followed by the application of a non-linear function, specifically, the

[3] From here onwards, we will use the terms *node* and *entity*, as well as *edge* and *relation(ship)* interchangeably.

leaky rectified linear unit (LeakyReLU) function with a negative slope of 0.01. Formally,

$$\mathbf{m}_{ijr} = \text{LeakyReLU}(\mathbf{W}_f[\mathbf{e_i};\mathbf{r};\mathbf{e_j}]), \tag{1}$$

where $\mathbf{W}_f \in \mathbb{R}^{d\times 3d}$ is a shared parameter for the linear transformation and [;] is the concatenation operator.

To compute an attention head, our model performs linear projections of the query relation r_q, the neighborhood relations $r \in \mathcal{R}_{ij}$, and the neighborhood messages $\mathbf{m}_{ijr}$ to construct queries Q, keys K, and values V, respectively, such that $Q = \mathbf{W}_Q\mathbf{r}_\text{q}$, $K = \mathbf{W}_K\mathbf{r}$, and $V = \mathbf{W}_V\mathbf{m}_{ijr}$, where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d'\times d}$ are learnable parameters.

Next, we use the queries Q to perform a dot-product attention over the keys K. Formally,

$$\alpha_{ijr} = \frac{\exp((\mathbf{W}_Q\mathbf{r}_\text{q})^\intercal(\mathbf{W}_K\mathbf{r}))}{\sum\limits_{z\in\mathcal{N}_i}\sum\limits_{r'\in\mathcal{R}_{ij}}\exp((\mathbf{W}_Q\mathbf{r}_\text{q})^\intercal(\mathbf{W}_K\mathbf{r'}))} \tag{2}$$

We adopt the common procedure of scaling the dot products of Q and K by a factor of $\frac{1}{\sqrt{d'}}$ [37].

The attention weights are then used to aggregate the neighborhood messages. Note that *self-attention* deploys multiple attention heads, each having its own query, key, and value projectors. The aggregated messages from N attention heads are concatenated and added to the initial embedding $\mathbf{e_i}$ through a residual connection to obtain new intermediate representation

$$\hat{\mathbf{e}}_\mathbf{i} = \mathbf{e_i} + \|_{n=1}^{N}\left(\sum_{j\in\mathcal{N}_i}\sum_{r\in\mathcal{R}_{ij}}\alpha_{ijr}^{n}\mathbf{W}_V^n\mathbf{m}_{ijr}\right), \tag{3}$$

where $\|$ is the concatenation operator.

Layer normalization (LN) [1] is applied to the intermediate representation $\hat{\mathbf{e}}_i$, followed by a fully connected two-layer feed forward network (FFN) with a non-linear activation (ReLU) in between. Finally, the output of the feed forward network is added to the intermediate representation through another residual connection. The resulting embedding is again layer normalized to obtain the new representation $\mathbf{g_i}^l$ for e_i. Formally,

$$\mathbf{g_i}^l = \text{LN}(\text{FFN}(\text{LN}(\hat{\mathbf{e}}_\mathbf{i})) + \text{LN}(\hat{\mathbf{e}}_\mathbf{i})) \tag{4}$$

This pipeline is called a *Transformer block*. Figure 2 represents a schematic diagram of a Transformer block in Graph Transformers. We stack L layers of Transformer blocks to obtain the final embedding $\mathbf{g_i}^L$ for e_i.

3.3 Policy Gradient for Explainable Reasoning

To infer the answer entity, we could leverage the entity representations obtained by the Graph Transformers. However, our goal is not only to infer the answer entity, but to find a symbolic reasoning path to support the inference. Following previous work [4, 23], we formulate the reasoning task as a finite horizon, deterministic partially observable Markov Decision Process (POMDP). A knowledge graph can be seen as a partially observable environment with out-going relations at each entity node corresponding to a set of discrete actions that an agent can explore to reach the target answer from the source entity.

Knowledge Graph Environment. Formally, a Markov Decision Process is defined by a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{P}$ captures state transition probabilities, and $\mathcal{R}$ is the reward function. In a knowledge graph environment, the state space is defined as a set of tuples $s_t = (e_t, r_\mathrm{q}) \in \mathcal{S}$, where e_t is an entity node in the knowledge graph, and r_q is the query relation. The action space $A_t \in \mathcal{A}$ for a state s_t is defined as the set of outgoing edges from entity node e_t in the knowledge graph. Formally, $A_t = \{(r_{t+1}, s_{t+1}) \mid (e_t, r_{t+1}, s_{t+1}) \in \mathcal{G}\}$. Since state transitions in a KG environment are deterministic, the transition probabilities $P(s_{t+1} \mid s_t, a_t) = 1 \; \forall P \in \mathcal{P}$. The agent receives a terminal reward of 1 if it arrives at the correct answer entity at the end.

Graph Search Policy. To find a plausible path to the answer entity, the model must have a policy to choose the most promising action at each state. Note that in the KG environment, the decision of choosing the next action is not only dependent on the current state, but also on the sequence of observations and actions taken so far in the path. We use a multi-layer LSTM as a sequence encoder to encode the path history.

Formally, each state s_t is represented by a vector $\mathbf{s}_t = [\mathbf{e}_t; \mathbf{r}_q] \in \mathbb{R}^{2d}$ and each possible action $a_t \in A_t$ is represented by $\mathbf{a}_t = [\mathbf{e}_{t+1}; \mathbf{r}_{t+1}] \in \mathbb{R}^{2d}$, where $\mathbf{e}_t$, $\mathbf{e}_{t+1} \in \mathbb{R}^d$ are the embeddings of the entity nodes at timesteps t and $t+1$, respectively, that are obtained from Graph Transformer encoders. $\mathbf{r}_{t+1} \in \mathbb{R}^d$ is the embedding of an out-going relation from entity e_t, and $\mathbf{r}_q \in \mathbb{R}^d$ corresponds to the embedding of the query relation r_q. Each of these embeddings is also obtained from the Graph Transformer encoder. The path history is encoded as $\mathbf{h}_t = \mathrm{LSTM}(\mathbf{h}_{t-1}, \mathbf{a}_{t-1})$. Given the embedded action space $\mathbf{A}_t \in \mathbb{R}^{2|A_t|}$, i.e., the stacked embeddings of actions $a_t \in A_t$, and the path history $\mathbf{h}_t$, we define the parameterized policy as:

$$\pi_\theta(a_t \mid s_t) = \mathrm{Softmax}(\mathbf{A}_t(\mathbf{W_2}\mathrm{ReLU}(\mathbf{W_1}[\mathbf{h}_t; \mathbf{e}_t; \mathbf{r}_q])))$$

Policy Optimization. The policy network is trained to maximize the expected reward for all $(e_\mathrm{s}, r_\mathrm{q}, e_\mathrm{o})$ triples in the training sub-graph. The agent learns an optimal policy π_θ by exploring a state space of all possible actions. The objective of the agent is to take actions to maximize the expected end reward. Formally:

$$J(\theta) = \mathbb{E}_{(e_\mathrm{s}, r_\mathrm{q}, e_\mathrm{o})}\left[\mathbb{E}_{a_1,\ldots,a_{T-1}\sim\pi_\theta}[R(s_T | e_\mathrm{s}, r_\mathrm{q})]\right] \tag{5}$$

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{F}\rvert$			
				train	dev	test	aux
FB15k-237-Inductive	13,119	237	1,389	227,266	17,500	32,197	61,330
WN18RR-Inductive	35,928	11	4,029	67,564	3,000	11,015	19,395
NELL-995-Inductive	71,578	200	776	137,221	500	1,679	2,267

Table 1. Evaluation datasets for inductive setting

Since policy gradient uses gradient-based optimization techniques, the estimated gradient of the objective function can be derived as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{a_{1:T}\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_{1:T}|e_{\mathrm{s}}, r_{\mathrm{q}}) R(s_T|e_{\mathrm{s}}, r_{\mathrm{q}})] \tag{6}$$

$$\approx \frac{1}{N}\sum_{n=1}^{N} \nabla_\theta \log \pi_\theta(a^n_{1:T}|e_{\mathrm{s}}, r_{\mathrm{q}}) R \tag{7}$$

Here, N is the number of policy rollouts.

Each policy rollout explores a sequence of actions $a_{1:T}$. At each timestep $t \in \{1 : T\}$, the agent selects an action a_t conditioned on the current state s_t. Therefore, the gradient of the log-likelihood in Eq. 6 can be expressed as

$$\nabla_\theta \log \pi_\theta(a_{1:T}|e_{\mathrm{s}}, r_{\mathrm{q}}) = \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t, e_{\mathrm{s}}, r_{\mathrm{q}}) \tag{8}$$

Reward Shaping. Previous work [23] observed that a soft reward for the target entities is more beneficial than a binary reward. Following their work, we use pre-trained ConvE [7] embeddings for the observed entities and relations to shape the reward function. If the agent reaches the correct answer entity, it receives reward 1. Otherwise, the agent receives a reward estimated by the scoring function of the pre-trained ConvE. Note that the ConvE model is trained only on the training sub-graph of seen entities. ConvE plays no role during inference. Its only purpose is to provide *soft reward* signals during training to help the model in learning a better policy.

4 Evaluation

4.1 Datasets

We evaluate our model based on three standard benchmark knowledge graph completion datasets. (1) FB15k-237 [33], introduced as a replacement for the FB15k dataset [3]. In FB15k-237, the reverse relations are removed, rendering the dataset more challenging for inference. (2) WN18RR [7] is a subset of the WN18 benchmark dataset. Similar to FB15k-237, the reverse relations are removed for this dataset. (3) NELL-995 [44] is a subset of the 995-th iteration of NELL.

To test the effectiveness of our model for inductive representation learning and reasoning, we create new splits of training, development, and test sets for each of the three benchmark datasets mentioned above. This new split of the data is necessary, as in an inductive setting, the subject entities in the test set must not be present anywhere in the training subgraph. To satisfy this requirement, we first sample 10% of all the entities present in each of the benchmark datasets. We denote this as the set of *unseen entities* $\mathcal{U}$, while the remaining entities are denoted as *seen entities* $\mathcal{E}$. Then, we proceed to split the triples in the datasets into three disjoint sets. The first set contains the triples in which both the head and the tail entities are in $\mathcal{E}$. The second set consists of the triples with head entities belonging to $\mathcal{U}$, but tail entities in $\mathcal{E}$. In the third set, the head entities belong to $\mathcal{E}$, but the tail entities are in $\mathcal{U}$. We further split the first set into *train* and *dev* triples. The second set becomes the *test* triples, and the union of the second and the third set is denoted as *auxiliary* data. Auxiliary triples are required to obtain the local neighborhood of a source entity at inference time. Note that an emerging entity in the test set is not disconnected from the training graph. It has at least one seen entity in its neighborhood. This ensures that our model can find a path to the target entity during inference. If the emerging entity were completely disconnected from the training graph (i.e. all neighboring nodes were in $\mathcal{U}$), finding a path to the target entity would not be possible.

We append the suffix *"-Inductive"* to distinguish these newly derived datasets from their original counterparts. A summary of these datasets is presented in Table 1. To help with the reproducibility for future research on this topic, we make the datasets and our source code publicly available at:
`https://github.com/kingsaint/InductiveExplainableLinkPrediction`

4.2 Baselines

Embedding Methods. We compare our model to a set of embedding based models that perform well under the transductive setting of link prediction. Although these models are particularly unsuitable for the inductive setting, we include them to better demonstrate the challenges of applying such algorithms in an inductive setting. In particular, we compare our model to ConvE [7], TransH [40], TransR [24], and RotatE [32]. For these experiments, we adapted the PyKEEN[4] implementations of these models.

Graph Convolution Methods. We choose a state-of-the-art graph convolution-based method CompGCN [36] as a baseline. Our choice is motivated by two factors: (1) CompGCN performs strongly in the transductive setting by outperforming the other baselines for most of the datasets, and (2) since its encoder module deploys neighborhood integration through Graph Convolution Networks, it has similar characteristics to our model, and therefore, is a good candidate for inductive representation learning. We also compare our model to R-GCN [29] and SACN [30], which also leverage the graph structure to learn node representations by aggregating neighborhood information. For CompGCN and SACN, we adapted the source code made available by the authors to make them suit-

[4] https://github.com/pykeen/pykeen

able for inductive representation learning and link prediction. For R-GCN, we adapted the source code available in the DGL library[5].

Symbolic Rule Mining. We compare our model with AnyBURL [25], a purely symbolic rule mining system. AnyBURL is capable of extremely fast rule mining, has outperformed other rule mining approaches including AMIE+ [9], and produces comparable results to existing embedding-based models.

Path-based Model. Finally, we compare our model to a policy gradient-based multihop reasoning approach [23] that is similar to the decoder module of our model. We modified the source code[6] of this model to adapt it to our task.

4.3 Experimental Details

Training Protocol. Since the benchmark knowledge graph completion datasets contain only unidirectional edges $(e_\mathrm{s}, r_\mathrm{q}, e_\mathrm{o})$, for all methods, we augment the training sub-graph with the reverse edges $(e_\mathrm{o}, r_\mathrm{q}^{-1}, e_\mathrm{s})$. During the Graph Transformer based inductive representation learning, $n\%$ of local neighboring entities are randomly selected and masked. During training, we mask 50%, 50%, and 30% of neighboring nodes, respectively, for the FB15k-237, WN188RR, and NELL-995 datasets. Neighborhood masking helps in learning robust representations and reduces the memory footprint, and has been shown to be effective [15]. Following previous work [4, 23], during training of the policy network, we also retain the top-k outgoing edges for each entity that are ranked by the PageRank scores of the neighboring entities. We set the value of k for each dataset following Lin et al. [23]. Such a cut-off threshold is necessary to prevent memory overflow. Finally, we adopt the false-negative masking technique in the final timestep of the policy rollouts to guide the agent to the correct answer entities as described in previous work [4, 23], where it was found helpful when multiple answer entities are present in the training graph.

Hyperparameters. For a fair comparison to the baselines, we keep the dimensionality of the entity and relation embeddings at 200. For our model, we deploy one layer of a Transformer block ($L = 1$) and 4 attention heads ($N = 4$). We choose a minibatch size of 64 during training due to limited GPU memory. We rely on Adam [19] stochastic optimization with a fixed learning rate of 0.001 across all training epochs. Additionally, we adopt entropy regularization to improve the learning dynamics of the policy gradient method. The regularizer is weighted by a hyperparameter β set to a value within $[0, 0.1]$. We apply dropout to the entity and relation embeddings, the feedforward networks, and the residual connections. The policy rollout is done for $T = 3$ timesteps for every dataset.

[5] https://github.com/dmlc/dgl/tree/master/examples/pytorch/rgcn

[6] https://github.com/salesforce/MultiHopKG

	WN18RR-Inductive				FB15K-237-Inductive				NELL-995-Inductive			
		Hits@N				Hits@N				Hits@N		
Model	MRR	@1	@3	@10	MRR	@1	@3	@10	MRR	@1	@3	@10
TransR [24]	0.8	0.6	0.7	0.9	5.0	4.0	5.2	6.6	5.3	4.9	5.3	6.5
TransH [40]	0.0	0.0	0.0	0.0	6.2	5.4	6.3	8.0	3.6	3.4	3.6	3.6
RotatE [32]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ConvE [7]	1.9	1.1	2.1	3.5	26.3	20.0	28.7	38.8	43.4	32.5	50.3	60.9
R-GCN [29]	14.7	11.4	15.1	20.7	19.1	11.5	20.9	34.3	58.4	50.9	62.9	71.6
SACN [30]	17.5	9.7	20.3	33.5	29.9	20.5	32.8	50.0	42.4	37.0	42.9	53.2
CompGCN [36]	2.2	0.0	2.2	5.2	26.1	19.2	28.5	39.2	42.8	33.1	47.9	61.0
AnyBURL [25]	-	**48.3**	50.9	53.9	-	28.3	43.0	56.5	-	8.7	11.0	12.3
MultiHopKG [23]	45.5	39.4	49.2	56.5	38.6	29.3	43.4	56.7	74.7	69.1	78.3	84.2
Our Model w/ RS	**48.8**	42.1	**52.2**	**60.6**	**39.8**	**30.7**	**44.5**	**57.6**	**75.2**	**69.7**	**79.1**	**84.4**

Table 2. Evaluation results of our model as compared to alternative baselines on inductive variants of the WN18RR, FB15K-237, and NELL-995 datasets. The Hits@N and MRR metrics are multiplied by 100.

Evaluation Protocol. Following previous work [23], we adopt beam search decoding during inference with a beam width of 512 for NELL-995 and 256 for the other datasets. If more than one path leads to the same target entity, then the path with the maximum log-likelihood is chosen over the others. During evaluation, the auxiliary graph augments the training graph to construct the KG environment with unseen entities and their relations to the seen entities. For our model and the baselines, the embeddings of all unseen entities are initialized with Xavier normal initialization [12] at inference time.

Evaluation Metrics. We adopt the ranking based metrics *Mean Reciprocal Rank* and *Hits@k* that are also used by prior work for evaluation. We follow the *filtered setting* [3] adopted by prior approaches. In the filtered setting, the scores for the false negative answer entities are masked to facilitate correct ranking of the target entity.

4.4 Results

We present the experimental results of our method and the baselines in Table 2. The results of the embedding-based models TransH, TransR, and RotatE across all datasets demonstrates their inability to deal with entities that are unseen during training. These models are thus rendered as ineffective for inductive representation learning and reasoning. ConvE performs better than other embedding-based models we consider. Still, the much inferior performance of ConvE compared to our model shows that ConvE is not particularly suitable for inductive representation learning.

We observe that our model significantly outperforms the graph convolution network baselines CompGCN, SACN, and R-GCN across all datasets. Although these models use the neighborhood information for learning representations, unlike our method, their neighborhood integration methods do not explicitly consider the query relations.

	WN18RR-Inductive				FB15K-237-Inductive				NELL-995-Inductive			
		Hits@N				Hits@N				Hits@N		
Model	MRR	@1	@3	@10	MRR	@1	@3	@10	MRR	@1	@3	@10
Our Model w/ RS	**48.8**	42.1	**52.2**	**60.6**	**39.8**	**30.7**	**44.5**	**57.6**	**75.2**	**69.7**	**79.1**	**84.4**
Our Model w/o RS	48.2	40.1	53.0	62.2	37.8	29.4	42.6	54.0	71.1	65.3	75.0	79.9
GT + ConvTransE	1.1	0.6	1.1	1.8	22.9	17.3	24.9	33.3	47.9	40.6	51.1	61.9

Table 3. Ablation study. The Hits@N and MRR metrics are multiplied by 100.

We find AnyBURL and MultiHopKG to be the most competitive methods to ours. AnyBURL performs adequately for the WN18RR and FB15K-237 dataset while performing poorly on the NELL-995 dataset. MultiHopKG adapts surprisingly well to our dataset despite the unseen entities being initialized with Xavier normal initialization. We conjecture that the learned representations of the query and the outgoing edges (relations) have enough semantic information encoded in them to navigate to the target entity by simply exploiting the edge (relation) information. However, our proposed model holds an edge over this model with 7.2%, 3.1%, and 0.7% gains in the MRR metric for the WN18RR, FB15K-237, and NELL-995 datasets respectively.

5 Analysis

In this section, we perform further analysis of our proposed model. First, we conduct a set of ablation studies. Then, we qualitatively analyze our model's ability to provide reasoning paths as supporting evidence for inference. Finally, we analyze the effect of the cardinality of relation types on the inference process.

5.1 Ablation Study

To better understand the contribution of reward shaping in our model, we perform an ablation study, where our model is deprived of the *soft reward* signals provided by ConvE. In general, we observe that replacing reward shaping with hard binary reward deteriorates the performance of our model across all datasets. Note that our ablated version still mostly outperforms the other baseline methods.

Additionally, we experiment with a non-explainable variant of our model, in which we retain the Graph Transformer (GT) encoder, but we replace the policy gradient-based decoder with an embedding-based decoder called ConvTransE, which is also used in SACN as a decoder module. With this model, we observe a significant drop in performance. Thus, we conjecture that the policy gradient-based decoder not only provides explainability, but also is crucial for decoding.

5.2 Qualitative Analysis of Explainability

Since explainability is one of the key objectives of our model, we provide examples of explainable reasoning paths for queries that involve previously unseen source entities at inference time. Table 4 contains examples of 1-hop, 2-hop, and

Query	(William Green, worksFor, ?)
Answer	Accenture
Explanation	William Green $\xrightarrow{\text{personLeadsOrganization}}$ Accenture
Query	(Florida State, organizationHiredPerson, ?)
Answer	Bobby Bowden
Explanation	Florida State $\xleftarrow{\text{worksFor}}$ Bobby Bowden
Query	(Messi, athleteHomeStadium, ?)
Answer	Camp Nou
Explanation	Messi $\xrightarrow{\text{athletePlaysForTeam}}$ Barcelona $\xrightarrow{\text{teamHomeStadium}}$ Camp Nou
Query	(Adrian Griffin, athleteHomeStadium, ?)
Answer	United Center
Explanation	Adrian Griffin $\xrightarrow{\text{athletePlaysForTeam}}$ Knicks $\xleftarrow{\text{athletePlaysForTeam}}$ Eddy Curry $\xrightarrow{\text{athleteHomeStadium}}$ United Center
Query	(Bucks, teamPlaysInLeague, ?)
Answer	NBA
Explanation	Bucks $\xrightarrow{\text{organizationHiredPerson}}$ Scott Stiles $\xleftarrow{\text{organizationHiredPerson}}$ Chicago Bulls $\xrightarrow{\text{teamPlaysInLeague}}$ NBA

Table 4. Example queries from the NELL-995 test set with unseen source entities. The answers are supported by the explainable reasoning paths derived by our model.

3-hop reasoning paths. These examples demonstrate our model's effectiveness in learning inductive representations for the unseen entities, which helps to infer the reasoning paths.

5.3 Effect of Relation Types

Dataset	to-Many		to-1	
	%	MRR	%	MRR
FB15k-237-Inductive	77.4	31.6	22.6	75.5
WN18RR-Inductive	48.1	60.8	51.9	30.1
NELL-995-Inductive	7.6	41.4	92.4	78.5

Table 5. MRR for the test triples in inductive setting with *to-Many* and *to-1* relation types. The % columns show the percentage of test triples for each relation type.

Following Bordes et al. [3], we categorize the relations in the seen snapshot of the knowledge graph into Many-to-1 and 1-to-Many relations. The categorization is done based on the ratio of the cardinality of the target answer entities to the source entities. If the ratio is greater than 1.5, we categorize the relation as *to-Many*, otherwise as *to-1*. We analyzed the results of the test set for these two types of relations. We report the percentage of triples with these two types of relations and the corresponding MRR achieved by our model in Table 5. For FB15k-237 and NELL-995, our model performs better for *to-1* relations than for *to-many* relations. On the contrary, we observe a reverse trend for the WN18RR dataset. Note however that *to-many* relations have alternative target entities. In the current evaluation protocol, our model is punished for predicting any alternative target entity other than the ground truth target.

6 Conclusion

The ever-expanding number of entities in knowledge graphs warrants the exploration of knowledge graph completion methods that can be applied to emerging entities without retraining the model. While prior approaches assume a static snapshot of the knowledge graph, we introduce a joint framework for inductive representation learning to predict missing links in a dynamic knowledge graph with many emerging entities. Additionally, our method provides explainable reasoning paths for the inferred links as support evidence. Through experiments we demonstrate that our model significantly outperforms the baselines across the new inductive benchmark datasets introduced in this paper.

7 Acknowledgement

We thank Diffbot for their grant support to Rajarshi Bhowmik's work. We also thank Diffbot and Google for providing the computing infrastructure required for this project.

References

1. Ba, J., Kiros, J.R., Hinton, G.E.: Layer normalization. ArXiv **1607.06450** (2016)
2. Bhowmik, R., de Melo, G.: Be concise and precise: Synthesizing open-domain entity descriptions from facts. In: Proceedings of The Web Conference 2019. pp. 116–126. ACM, New York, NY, USA (2019)
3. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems 26. pp. 2787–2795 (2013)
4. Das, R., Dhuliawala, S., Zaheer, M., Vilnis, L., Durugkar, I., Krishnamurthy, A., Smola, A.J., McCallum, A.: Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. arXiv **1711.05851** (2017)
5. Dasgupta, S.S., Ray, S.N., Talukdar, P.: HyTE: Hyperplane-based temporally aware knowledge graph embedding. In: Proceedings of EMNLP 2018 (2018)
6. de Melo, G.: Knowledge graphs: Venturing out into the wild. In: ISWC 2016: Knowledge Graphs and Language Technology. Springer (2017)
7. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018). pp. 1811–1818. AAAI Press (2018)
8. Fu, Z., Xian, Y., Gao, R., Zhao, J., Huang, Q., Ge, Y., Xu, S., Geng, S., Shah, C., Zhang, Y., de Melo, G.: Fairness-aware explainable recommendation over knowledge graphs. In: Proceedings of the 43rd SIGIR 2020. ACM (2020)
9. Galárraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast rule mining in ontological knowledge bases with AMIE++. The VLDB Journal **24**(6), 707–730 (2015)

10. Gardner, M., Talukdar, P.P., Kisiel, B., Mitchell, T.M.: Improving learning and inference in a large knowledge-base using latent syntactic cues. In: Proceedings of EMNLP 2013. pp. 833–838. ACL (2013)
11. Gardner, M., Talukdar, P.P., Krishnamurthy, J., Mitchell, T.M.: Incorporating vector space similarity in random walk inference over knowledge bases. In: Proceedings of EMNLP 2014. pp. 397–406. ACL (2014)
12. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 9, pp. 249–256. PMLR (13–15 May 2010)
13. Guu, K., Miller, J., Liang, P.: Traversing knowledge graphs in vector space. In: Proceedings of EMNLP 2015. pp. 318–327. ACL (2015)
14. Hamaguchi, T., Oiwa, H., Shimbo, M., Matsumoto, Y.: Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. In: Proceedings of IJCAI. pp. 1802–1808. AAAI Press (2017)
15. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing (2017)
16. Ho, V.T., Stepanova, D., Gad-Elrab, M.H., Kharlamov, E., Weikum, G.: Rule learning from knowledge graphs guided by embedding models. In: Proceedings of ISWC 2018. pp. 72–90. Springer International Publishing, Cham (2018)
17. Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutierrez, C., Labra Gayo, J.E., Kirrane, S., Neumaier, S., Polleres, A., Navigli, R., Ngonga Ngomo, A.C., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., Zimmermann, A.: Knowledge graphs. ArXiv **2003.02320** (2020)
18. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of ACL-IJCNLP 2015. pp. 687–696. ACL (2015)
19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014), http://arxiv.org/abs/1412.6980
20. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2017)
21. Koncel-Kedziorski, R., Bekal, D., Luan, Y., Lapata, M., Hajishirzi, H.: Text Generation from Knowledge Graphs with Graph Transformers. In: Proceedings of NAACL 2019. pp. 2284–2293. ACL (Jun 2019)
22. Lao, N., Mitchell, T.M., Cohen, W.W.: Random walk inference and learning in A large scale knowledge base. In: Proceedings of EMNLP 2011. pp. 529–539. ACL (2011)
23. Lin, X.V., Socher, R., Xiong, C.: Multi-hop knowledge graph reasoning with reward shaping. arXiv **abs/1808.10568** (2018), http://arxiv.org/abs/1808.10568
24. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI Conference on Artificial Intelligence (2015)
25. Meilicke, C., Chekol, M.W., Ruffinelli, D., Stuckenschmidt, H.: An introduction to AnyBURL. In: KI 2019: Advances in Artificial Intelligence. pp. 244–248. Springer International Publishing, Cham (2019)
26. Neelakantan, A., Roth, B., McCallum, A.: Compositional vector space models for knowledge base completion. In: Proceedings of ACL 2015. ACL (2015)
27. Nguyen, D.Q., Nguyen, T.D., Nguyen, D.Q., Phung, D.: A novel embedding model for knowledge base completion based on convolutional neural network. In: Proceedings of NAACL 2018. pp. 327–333 (2018)
28. Nguyen, D.Q.: An overview of embedding models of entities and relationships for knowledge base completion. arXiv **1703.08098** (2017)

29. Schlichtkrull, M.S., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Proceedings of ESWC 2018. LNCS, vol. 10843, pp. 593–607. Springer (2018)
30. Shang, C., Tang, Y., Huang, J., Bi, J., He, X., Zhou, B.: End-to-end structure-aware convolutional networks for knowledge base completion. In: Proceedings of AAAI (2019)
31. Shen, Y., Chen, J., Huang, P.S., Guo, Y., Gao, J.: M-Walk: Learning to walk over graphs using Monte Carlo Tree Search. In: Advances in Neural Information Processing Systems 31. pp. 6786–6797. Curran Associates, Inc. (2018)
32. Sun, Z., Deng, Z.H., Nie, J.Y., Tang, J.: RotatE: Knowledge graph embedding by relational rotation in complex space. In: International Conference on Learning Representations (2019)
33. Toutanova, K., Lin, V., Yih, W., Poon, H., Quirk, C.: Compositional learning of embeddings for relation paths in knowledge base and text. In: Proceedings of ACL 2016. ACL (2016), http://aclweb.org/anthology/P/P16/P16-1136.pdf
34. Trivedi, R., Farajtabar, M., Biswal, P., Zha, H.: DyRep: Learning representations over dynamic graphs. In: ICLR (2019)
35. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: Proceedings of the 33nd International Conference on Machine Learning, ICML 2016. vol. 48, pp. 2071–2080 (2016)
36. Vashishth, S., Sanyal, S., Nitin, V., Talukdar, P.: Composition-based multi-relational graph convolutional networks. In: International Conference on Learning Representations (2020)
37. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems 30, pp. 5998–6008. Curran Associates, Inc. (2017)
38. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph Attention Networks. International Conference on Learning Representations (2018)
39. Wang, L., Wang, Y., Liu, B., He, L., Liu, S., de Melo, G., Xu, Z.: Link prediction by exploiting network formation games in exchangeable graphs. In: Proceedings of IJCNN 2017. pp. 619–626 (2017), https://ieeexplore.ieee.org/document/7965910/
40. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of AAAI 2014. pp. 1112–1119. AAAI Press (2014)
41. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. **8**(3-4), 229–256 (May 1992)
42. Xian, Y., Fu, Z., Muthukrishnan, S., de Melo, G., Zhang, Y.: Reinforcement knowledge graph reasoning for explainable recommendation. In: Proceedings of SIGIR 2019. pp. 285–294. ACM, New York, NY, USA (2019)
43. Xian, Y., Fu, Z., Zhao, H., Ge, Y., Chen, X., Huang, Q., Geng, S., Qin, Z., de Melo, G., Muthukrishnan, S., Zhang, Y.: CAFE: Coarse-to-fine knowledge graph reasoning for e-commerce recommendation. In: Proceedings of CIKM 2020. ACM (2020)
44. Xiong, W., Hoang, T., Wang, W.Y.: DeepPath: A reinforcement learning method for knowledge graph reasoning. In: Proceedings of EMNLP 2017. ACL (2017)
45. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. CoRR **abs/1412.6575** (2014)
46. Yang, K., Xinyu, K., Wang, Y., Zhang, J., de Melo, G.: Reinforcement learning over knowledge graphs for explainable dialogue intent mining. IEEE Access **8**, 85348–85358 (2020), https://ieeexplore.ieee.org/document/9083954
47. Yun, S., Jeong, M., Kim, R., Kang, J., Kim, H.J.: Graph Transformer networks. In: Advances in Neural Information Processing Systems 32, pp. 11983–11993 (2019)