The Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22)

Inductive Relation Prediction by BERT

Hanwen Zha, Zhiyu Chen, Xifeng Yan
University of California, Santa Barbara
{hwzha, zhiyuchen, xyan}@cs.ucsb.edu

Abstract

Relation prediction in knowledge graphs is dominated by embedding based methods which mainly focus on the transductive setting. Unfortunately, they are not able to handle inductive learning where unseen entities and relations are present and cannot take advantage of prior knowledge. Furthermore, their inference process is not easily explainable. In this work, we propose an all-in-one solution, called BERTRL (BERT-based Relational Learning), which leverages pre-trained language model and fine-tunes it by taking relation instances and their possible reasoning paths as training samples. BERTRL outperforms the SOTAs in 15 out of 18 cases in both inductive and transductive settings. Meanwhile, it demonstrates strong generalization capability in few-shot learning and is explainable. The data and code can be found at https://github.com/zhw12/BERTRL.

Introduction

Knowledge graphs (KGs) are essential in a wide range of tasks such as question answering and recommendation systems (Ji et al. 2020). As many knowledge graphs are substantially incomplete in practice, knowledge graph completion (KGC) becomes a must in many applications (Nickel et al. 2016).

Embedding-based methods such as TransE (Bordes et al. 2013), Complex (Trouillon et al. 2017), ConvE (Dettmers et al. 2018), RotatE (Sun et al. 2019) and TuckER (Balažević, Allen, and Hospedales 2019), achieve the state-of-the-art performance on a few KGC benchmarks. However, the drawbacks of these approaches are obvious as they are limited to the *transductive* setting where entities and relations need to be seen at training time. In reality, new entities and relations emerge over time (*inductive* setting). The cost of retraining may be too high for dynamically populated knowledge graphs. In addition to the inductive setting, explainability, few-shot learning and transfer learning cannot be easily solved by these specialized embedding methods.

Logical induction methods partially meet the aforementioned need by seeking probabilistic subgraph patterns (GraIL(Teru, Denis, and Hamilton 2020), CoMPILE(Mai et al. 2021), and TACT (Chen et al. 2021)), logical rules (AMIE (Galárraga et al. 2013), RuleN (Meilicke et al. 2018)) or their differentiable counterparts (NEURAL-LP (Yang, Yang, and Cohen 2017), DRUM (Sadeghian et al. 2019)). The following shows a logical rule which is explainable, can be generalized, and can handle unseen entities,

$$(x, \textit{president\_of}, y) \wedge (z, \textit{capital\_of}, y) \rightarrow (x, \textit{work\_at}, z). \quad (1)$$

These logical rules introduce inductive ability for predicting missing links in KG. For example, once the rule in (1) is learned, the model can generalize to other *president*, *capital* and *country*.

Despite the compelling advantage of the existing logical induction methods, their inductive learning power is limited as it only exploits the structural information while ignoring the textual information associated with entities and relations, and furthermore, prior knowledge carried in these texts. This weakens the model's usability when only small knowledge graphs are available – a typical few-shot setting. Moreover, none of them can handle unseen but relevant relations in KG completion.

In this work, we propose an all-in-one solution, called BERTRL (BERT-based Relational Learning), a model that combines rule-based reasoning with textual information and prior knowledge by leveraging pre-trained language model, BERT (Devlin et al. 2019). In BERTRL, we linearize the local subgraph around entities in a target relation (h, r, t) into paths $p : (h, r_0, e_1), (e_1, r_1, e_2), \ldots, (e_n, r_n, t)$, input $(h, r, t) : p$ to BERT, and then fine-tune. BERTRL is different from KG-BERT (Yao, Mao, and Luo 2019) where only relation instance (h, r, t) is fed to BERT. While this difference looks small, it actually lets BERTRL reason explicitly via paths connecting two entities. KG-BERT's prediction is mainly based on the representation of entities and relations: Knowledge graph is memorized inside BERT and reasoning is implicit. In BERTRL, knowledge is dynamically retrieved from the knowledge graph during inference: Reasoning is conducted explicitly, which enables BERTRL to achieve explainability and much higher accuracy. Table 1 illustrates the difference among these approaches.

Our approach naturally generalizes to unseen entities. It also has the potential to handle some unseen relations. Empirical experiments on inductive knowledge graph completion benchmarks demonstrate the superior performance of BERTRL in comparison with state-of-the-art baselines: It

Method	Transductive Setting	Inductive Setting			Prior Knowledge	Explainable
		Unseen Entities	Unseen Relations	Reasoning with context		
TuckER	✓	×	×	×	×	×
RuleN	✓	✓	×	✓	×	✓
GraIL	✓	✓	×	✓	×	×
KG-BERT	✓	✓	✓	×	✓	×
BERTRL (ours)	✓	✓	✓	✓	✓	✓

Table 1: Comparison of BERTRL with other relation prediction algorithms on their capability of handling the transductive setting, unseen entities in the inductive setting, their potential of dealing with unseen relations, usage of prior knowledge, the explainability of their inference process, and whether they can reason with the context of entities in the knowledge graph explicitly. BERTRL and KG-BERT are provided with knowledge graph, entity names and relation names. We take TuckER as a representative of embedding-based methods.

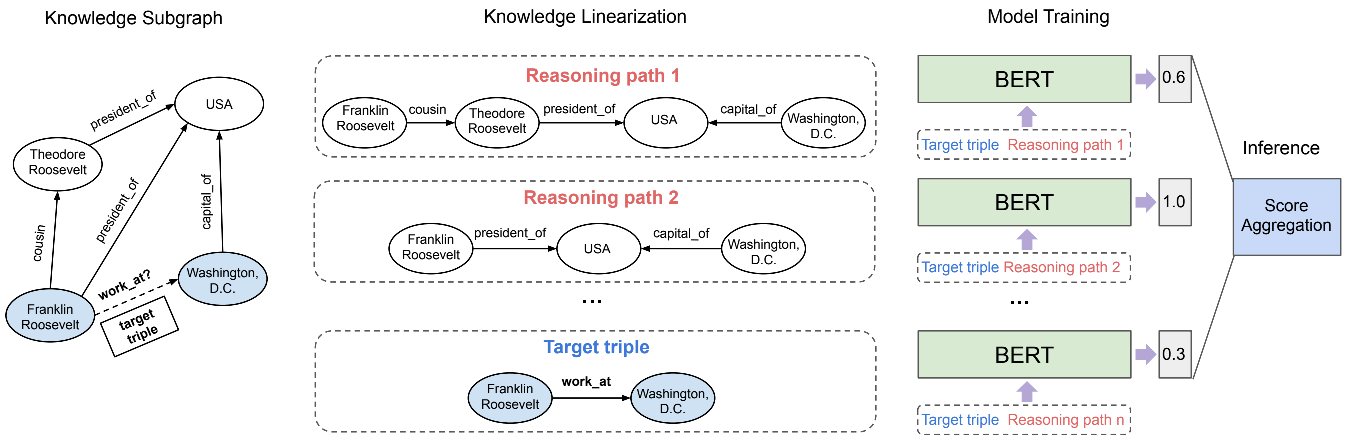

Figure 1: The BERTRL pipeline.

achieves an absolute increase of 6.3% and 5.3% in Hits@1 and MRR on average. In a few-shot learning scenario, it can even achieve a maximum of 32.7% and 27.8% absolute Hits@1 and MRR improvement.

In the transductive setting, BERTRL performs competitively with the state-of-the-art embedding methods and surpasses the inductive learning counterparts. In few-shot learning (partially transductive), BERTRL again introduces a larger margin over the baselines.

Finally, we analyze how BERTRL performs in unseen relation prediction, its explainability, its training and inference time, and conduct an ablation study on a few design choices.

Proposed Approach

Problem Formulation. Knowledge graph consists of a set of triples $\{(h_i, r_i, t_i)\}$ with head, tail entities $h_i, t_i \in \mathcal{E}$ (the set of entities) and relation $r_i \in \mathcal{R}$ (the set of relations). Given an incomplete knowledge graph G, the relation prediction task is to score the probability that an unseen relational triple (h, r, t) is true, where h and t denote head and tail entities and r refers to a relation. (h, r, t) is also called target relational triple.

Our model scores a relational triple in two steps: (Step 1) Extracting and linearizing the knowledge $G(h, t)$ surrounding entities h and t in G; (Step 2) Scoring the triple with $G(h, t)$ by fine-tuning the pre-trained language model BERT.

Model Details

Step 1: Knowledge Linearization. The knowledge $G(h, t)$ surrounding entities h and t in a knowledge graph G provides important clues for predicting missing links between h and t. $G(h, t)$ could be exploited in various ways: It could be any subgraph around h and t and even not necessarily be connected. However, the different choices of $G(h, t)$ will affect the model complexity and its explainability. RuleN (Meilicke et al. 2018) uses all the paths connecting h and t up to k length. GraIL(Teru, Denis, and Hamilton 2020) uses a subgraph that merges all of these paths, aiming to leverage structural information. In order to use pre-trained language models like BERT, we need to linearize $G(h, t)$ as $\ell(G(h, t))$ and concatenate it with (h, r, t) as valid input to BERT,

$$(h, r, t) : \ell(G(h, t)). \tag{2}$$

Our intuition is that BERT shall have the capability of learning signals in $G(h, t)$ that could be correlated with (h, r, t), and BERT shall be able to handle noisy and erroneous inputs.

Subgraph. One straightforward linearization of a subgraph would be concatenating text of its edges one by one separated by a delimiter such as a semicolon. This formalism has two major issues. First, local subgraphs could be very large: The size grows exponentially with respect to their diameters. Hence concatenated edges may not fit into the available BERT models. Second, the subgraph edges are unordered, which might incur additional cost for BERT to learn orders and produce correct scoring. We will show experiment of subgraph-based linearization design in ablation study.

Paths. Another linearization method is collecting all of the paths up to length k connecting h and t. We call them *reasoning paths*. Each reasoning path between h and t consists of a sequence of triples $h \rightarrow t$: $(h, r_0, e_1), (e_1, r_1, e_2), ..., (e_n, r_n, t)$.

There are two ways of leveraging reasoning paths: One called *combined paths*, puts all the paths together as one input to BERT, thus allowing the interaction across different path units. The other called *individual paths*, takes each path as a separate input to BERT. Each reasoning path induces the target triple individually with a certain confidence score, and the final result is an aggregation of individual scores. In practice, the first method generates one sample concatenating all paths, while the second one separates each path into individual training samples.

Intuitively, the *combined paths* representation is more expressive as it could consider all the paths together and should perform better. The *individual paths* representation might generate many false associations as most of the paths are irrelevant to the target triple. Surprisingly, we found BERT is robust to those false associations taken in the training stage and is able to pick up true ones. We suspect that the individual paths representation has simpler training samples and likely most relation predictions can be achieved by one path in the existing KGC benchmarks.

Our final design takes the individual paths representation. The performance of different designs is presented in ablation study.

In order to better leverage the knowledge learned in a pre-trained language model, we adopt a natural language template (Schick and Schütze 2020) to transform target triple and reasoning path into a "somewhat natural" sentence. Take Figure 1 as an example. It could be “[CLS] Question: Franklin Roosevelt work at what ? Is the correct answer Washington D.C. ? [SEP] Context: Franklin Roosevelt president of USA; Washington D.C. capital of USA;” Each individual path will form a training/inference instance.

Step 2: BERT Scoring. In BERTRL, since we take individual paths as a linearization approach, each pair of triple and reasoning path is scored individually. For each target triple, one or a few reasoning paths would indicate the truth of the triple. This forms a multi-instance learning problem (Carbonneau et al. 2018), where predictions need to be aggregated for a bag of instances. We take a simplified realization - training individually and applying maximum aggregation of bag scoring at inference time.

BERTRL uses a linear layer on top of [CLS] to score the triple's correctness, which can be regarded as a binary classification problem. It models the probability of label y ($y \in \{0, 1\}$) given the text of triple (h, r, t) and the text of reasoning path $h \rightarrow t$,

$$p(y|h, r, t, h \rightarrow t). \tag{3}$$

At inference time, the final score of a target triple (h, r, t) is the maximum of the positive class scores over all of its reasoning paths:

$$score(h, r, t) = \max_{\rho} p(y = 1|h, r, t, h \xrightarrow{\rho} t). \tag{4}$$

The path corresponding to the maximum score can be used to explain how the prediction is derived. We leave a more sophisticated aggregation function for future study.

Training Regime

In order to train BERTRL, both positive and negative examples are needed. We follow the standard practice to view existing triples in KG as positive. Then, for each positive triple, we do *negative sampling* to sample m triples corrupting its head or tail. Specifically, we randomly sample entities from common k-hop neighbors of head and tail entities, and make sure negative triples are not in KG. We do not include empty reasoning path examples in training, and always give a minimum confidence score for empty path in inference.

When constructing reasoning paths for a triple, we hide the triple in KG and find other paths to simulate missing link prediction. As the maximum length of the reasoning paths increases, the number of paths may grow exponentially. Many paths are spurious and not truly useful for inducing the triple. We do *path sampling* at training time to get at most n paths between target entities and take shorter paths first.

Finally we use cross entropy loss to train our model:

$$\mathcal{L} = -\sum_{\tau} (y_\tau \log p_\tau + (1 - y_\tau) \log(1 - p_\tau)), \tag{5}$$

where $y_\tau \in \{0, 1\}$ indicates negative or positive label, and $\tau \in \mathbb{D}^+ \cup \mathbb{D}^-$. The negative triple set $\mathbb{D}^-$ is generated by previously mentioned method that corrupts head entity h or tail entity t in a positive triple $(h, r, t) \in \mathbb{D}^+$ with a sampled entity h' or t', i.e.,

$$\mathbb{D}^- = \{(h', r, t) \notin \mathbb{D}^+ \cup (h, r, t') \notin \mathbb{D}^+\}. \tag{6}$$

Experiments

We evaluate our method on three benchmark datasets: WN18RR (Dettmers et al. 2018), FB15k-237 (Toutanova et al. 2015), and NELL-995 (Xiong, Hoang, and Wang 2017), using their inductive and transductive subsets introduced by GraIL(Teru, Denis, and Hamilton 2020) [1]. WN18RR is a subset of WordNet, a KG contains lexical relations between words. FB15k-237 is a subset of Freebase, a large KG of real-world facts. NELL-995 is a dataset constructed from high-confidence facts of NELL, a system constantly extracting facts from the web. The statistics of these datasets are given in Table 2; the details of the variants will be given later.

Through experiments, we would like to answer the following questions about BERTRL: (1) How does it generalize to relation prediction with unseen entities in the inductive

[1]https://github.com/kkteru/grail

	split	#relations	#nodes	#links
WN18RR	train	9	2,746	6,670
	ind-test	8	922	1,991
	train-1000	9	1,362	1,001
	train-2000	9	1,970	2,002
FB15k-237	train	180	1,594	5,223
	ind-test	142	1,093	2,404
	train-1000	180	923	1,027
	train-2000	180	1,280	2,008
	train-rel50	50	1,310	3,283
	train-rel100	100	1,499	3,895
NELL-995	train	88	2,564	10,063
	ind-test	79	2,086	5,521
	train-1000	88	893	1,020
	train-2000	88	1,346	2,011

Table 2: Statistics of the three datasets and their variants.

setting? (2) How does it perform in the traditional transductive setting? (3) Does it work well in few-shot learning? (4) Does it have the potential to generalize to unseen relations? (5) How its reasoning path explains prediction? (6) What is the training and inference time? (7) How important is the knowledge linearization design?

Baselines and Implementation Details. We compare BERTRL with the state-of-the-art inductive relation prediction methods GraIL(Teru, Denis, and Hamilton 2020), CoMPILE(Mai et al. 2021) and RuleN (Meilicke et al. 2018). GraIL and CoMPILE use graph neural network to reason over local subgraph structures, while CoMPILE emphasizes graph directionality. RuleN explicitly derives path-based rules and shows high precision. We use the public implementation provided by the authors and adopt the best hyper-parameter settings in their work. Differentiable logical rule learning methods like NeurLP (Yang, Yang, and Cohen 2017) and DRUM (Sadeghian et al. 2019) are not included, as their performance is not as good as GraIL, CoMPILE and RuleN. For the transductive setting, we pick one of the state-of-the-art embedding methods, TuckER (Balažević, Allen, and Hospedales 2019) and path-based method MINERVA (Das et al. 2018), as representatives for evaluation. For TuckER, we use implementation in LibKGE (Broscheit et al. 2020) with the provided best configuration in the library. For MINERVA, we use the official implementation and best configuration provided by authors.

We also compare against a BERT-based KGC method KG-BERT (Yao, Mao, and Luo 2019), where only relation triple (h, r, t) is fed to BERT. This is a special case of BERTRL with an empty reasoning path. In our experiments, we do not feed additional description other than entity and relation names as KG-BERT (Yao, Mao, and Luo 2019) did. The descriptions such as Wikipedia and WordNet synsets definitions often directly contain the missing entities, making knowledge graph completion leans to relation extraction rather than graph reasoning. Therefore, in this work, we keep a relatively pure knowledge graph setting as many graph embedding algorithms do. The extra requirement of names are readily available in most scenarios. In practice, both BERTRL and KG-BERT can be extended to accept additional information as this is what BERT is designed for.

Both BERTRL and KG-BERT were implemented in PyTorch using Huggingface Transformers library (Wolf et al. 2020). We fine-tune BERT-base for BERTRL and KG-BERT with Adam optimizer. The best learning rate and training epoch are selected based on validation set. Learning rate 5e-5 is set for BERTRL and 2e-5 for KG-BERT, and training epoch is 2 and 5 respectively. We sample 10 negative triples in negative sampling, and 3 reasoning paths in path sampling, and keep increasing the size does not improve performance.

Evaluation Task. Following GraIL(Teru, Denis, and Hamilton 2020), our default evaluation task is to predict $(h, r, ?)$ and $(?, r, t)$: specifically, ranking each test triple among 50 other negative candidates. The negative triples are not in KG and generated by randomly replacing head (or tail) entity of each test triple. The sampling is going to speed up the evaluation process. The performance will be lower if the ranking is done among the full entity set.

Metrics. We evaluate models on Hits@1 and Mean Reciprocal Rank (MRR). Hits@1 measures the percentage of cases in which positive triple appears as the top 1 ranked triple, while MRR takes the average of the reciprocal rank for positive triples.

Inductive Relation Prediction

We first evaluate the model's ability to generalize to unseen entities. In a fully inductive setting, the entities seen in training and testing are completely disjoint. For all the methods, we extract paths from the target head entity to the tail entity with length up to 3 or the subgraph containing these paths.

Datasets. We conduct our experiment using the inductive subsets of WN18RR, FB15k-237, and NELL-995 introduced by (Teru, Denis, and Hamilton 2020). Each subset consists of a pair of graphs *train-graph* and *ind-test-graph*. The former is used for training, and the latter provides an incomplete graph for relation prediction. *train-graph* contains all the relations present in *ind-test-graph*. However, their entity sets do not overlap. In GraIL, WN18RR, FB15k-237, and NELL-995 each induces four random inductive subsets (v1, v2, v3 and v4). We pick one subset for each (WN18RR v1, FB15k-237 v1 and NELL-995 v2). For each inductive dataset, we did stratified sampling on *train-graph* to create few-shot variants. The links are down-sampled to a number around 1,000 and 2,000, while keeping an unchanged proportion of triples for each relation. The few-shot training graph *train-1000* and *train-2000* contain all relations in its full setting, thus covering the relations in *test-graph* as well. The statistics of these variants are shown in Table 2.

Results. BERTRL significantly outperforms the baselines in most settings as shown in Tables 3 and **??**, particularly by around 10 absolute Hits@1 and MRR points in FB15k-237 and NELL-995. These two KGs have more relations and are associated with open-world knowledge (learned by BERT) compared with WN18RR. Methods like GraIL, CoMPILE and RuleN are not able to incorporate such prior knowledge.

	WN18RR			FB15k-237			NELL-995		
	1,000	2,000	6,678 (full)	1,000	2,000	5,223 (full)	1,000	2,000	10,063 (full)
RuleN	0.649	0.737	0.745	0.207	0.344	0.415	0.282	0.418	0.638
GraIL	0.516	**0.769**	**0.769**	0.273	0.351	0.390	0.295	0.298	0.554
CoMPILE	0.550	0.593	0.617	0.222	0.327	0.402	0.174	0.225	0.639
KG-BERT	0.364	0.404	0.436	0.288	0.317	0.341	0.236	0.236	0.244
BERTRL	**0.713**	0.731	0.755	**0.441**	**0.493**	**0.541**	**0.622**	**0.628**	**0.715**

Table 3: Inductive results (Hits@1)

	WN18RR			FB15k-237			NELL-995		
	1,000	2,000	6,678 (full)	1,000	2,000	5,223 (full)	1,000	2,000	10,063 (full)
RuleN	0.681	0.773	0.780	0.236	0.383	0.462	0.334	0.495	0.710
GraIL	0.652	**0.799**	**0.799**	0.380	0.432	0.469	0.458	0.462	0.675
CoMPILE	0.650	0.691	0.715	0.315	0.416	0.483	0.310	0.383	0.748
KG-BERT	0.471	0.525	0.547	0.431	0.460	0.500	0.406	0.406	0.419
BERTRL	**0.765**	0.777	0.792	**0.526**	**0.565**	**0.605**	**0.736**	**0.744**	**0.808**

Table 4: Inductive results (MRR)

In the few-shot setting, BERTRL stays robust and outperforms the baselines by an even larger margin. When more links are dropped in training graph, BERTRL achieves more performance gain over the baselines. BERTRL enjoys all sources of knowledge: structural (reasoning paths), textual (embedding), and prior knowledge (pre-trained language model). They all play an important role in knowledge graph completion.

In both settings, BERTRL performs better than KG-BERT, the version without reasoning paths inputted. It shows that incorporating paths allows pre-trained language models to gain explicit reasoning capability. On the other hand, with the triple information alone, KG-BERT is able to make a certain amount of correct inferences, suggesting that prior knowledge stored in pre-trained language models can be leveraged to do knowledge graph completion as manifested in (Yao, Mao, and Luo 2019). BERTRL combines explicit reasoning capability, prior knowledge, and language understanding all together in one model and has significant advantages.

Transductive Relation Prediction

BERTRL can also be applied in the transductive setting and be compared with the baselines.

Datasets. To evaluate the transductive performance, we train these models on *train-graph* introduced in the inductive setting and test on links with the same set of entities. We use a list of test triples with 10% size of *train-graph*. In a few-shot setting, we reuse the few-shot *train-graph* used in the inductive setting and tested on the aforementioned test links. At testing time, full *train-graph* is used to collect knowledge around target entities (otherwise, the setting will be close to the inductive one). The few-shot setting makes datasets partially transductive, as some entities become unseen when links are dropped randomly. For TuckER and MINERVA, we assign a minimum score for both positive and negative triples containing unseen entities.

Results. Tables 5 and 6 show that BERTRL outperforms the baselines in most of full and few-shot settings. It performs competitively with TuckER in the full setting and surpasses RuleN and GraIL. It implies that BERTRL's strong performance is not limited to inductive learning. In the few-shot setting, *train-graph* becomes sparse and unseen entities appear in testing. BERTRL again largely outperforms all the methods, which once more demonstrates the advantage of simultaneously exploiting all knowledge sources.

Unseen Relation Prediction

As BERTRL leverages a pre-trained language model, it has the potential to predict unseen relations in a zero-shot setting, which is not possible for traditional inductive learning methods like RuleN and GraIL. In this section, we examine how BERTRL can generalize for unseen relations.

Datasets. We create down-sampled training datasets from full FB15k-237 *train-graph*, and test on *ind-test-graph*. Specifically, we sample 50 and 100 relations weighted by their proportion in *train-graph*. Triples with sampled relations are collected to create graphs *train-rel50* and *train-rel100*. Each relation in FB15k-237 has a multi-level hierarchy, e.g., */film/language*. Words are shared in different relations, which helps the generalization to unseen relations.

Results. Table 8 shows Hits@1 results. Both KG-BERT and BERTRL make some correct predictions even without seeing the relations in training. Furthermore, Table 7 shows the best and worst performed unseen relation prediction on *train-rel50*. The relation names are simplified.

For each unseen relation, we manually identified a relevant relation in training set. The best performing relations often have close meaning counterparts in training, while the worst performing relations do not. This phenomenon indicates that in zero-shot setting, BERTRL generalizes to unseen relations through similar text. We suspect that knowledge captured by pre-trained language models also helps zero-shot learning.

	Transductive			Transductive (Few-shot)					
	WN18RR	FB15k-237	NELL-995	WN18RR		FB15k-237		NELL-995	
	6,670	5,223	10,063	1,000	2,000	1,000	2,000	1,000	2,000
RuleN	0.646	0.603	0.636	0.548	0.605	0.374	0.508	0.365	0.501
GraIL	0.644	0.494	0.615	0.489	0.633	0.267	0.352	0.198	0.342
MINERVA	0.632	0.534	0.553	0.106	0.248	0.170	0.324	0.152	0.284
TuckER	0.600	0.615	**0.729**	0.230	0.415	0.407	0.529	0.392	0.520
BERTRL	**0.655**	**0.620**	0.686	**0.621**	**0.637**	**0.517**	**0.583**	**0.526**	**0.582**

Table 5: Transductive results (Hits@1)

	Transductive			Transductive (Few-shot)					
	WN18RR	FB15k-237	NELL-995	WN18RR		FB15k-237		NELL-995	
	6,670	5,223	10,063	1,000	2,000	1,000	2,000	1,000	2,000
RuleN	0.669	0.674	0.736	0.567	0.625	0.434	0.577	0.453	0.609
GraIL	0.676	0.597	0.727	0.588	0.673	0.375	0.453	0.292	0.436
MINERVA	0.656	0.572	0.592	0.125	0.268	0.198	0.364	0.182	0.322
TuckER	0.646	0.682	**0.800**	0.258	0.448	0.457	0.601	0.436	0.577
BERTRL	**0.683**	**0.695**	0.781	**0.662**	**0.673**	**0.618**	**0.667**	**0.648**	**0.693**

Table 6: Transductive results (MRR)

Explainability

As stated in introduction, rules like (1) are explainable to humans. BERTRL achieves certain explainability by leveraging reasoning paths and implicitly memorizes these rules through training. For a prediction task $(h, r, ?)$, BERTRL is going to generate many instances for different tail entity t by concatenating triple (h, r, t) with each path $h \to t$. Those with the highest scores are chosen as the answer. We can regard the path chain as the explanation of deriving (h, r, t). We conduct manual case study using FB15k-237 dataset as an example. The texts are simplified.

The following KG completion query *(Chris, acts_in_film, ?)* is to find what film the actor Chris acts in. The instance ranked highest by BERTRL consists of target triple *(Chris, acts_in_film, Jackie Brown)*, reasoning path *(Chris, nominated_for_same_award_with, Robert); (Robert, acts_in_film, Jackie Brown);* and an assigned score 0.95. It could be naturally explained as follows: Chris likely acts in film Jackie Brown, since Robert shares the same award nomination with Chris and also acts in Jackie Brown.

We then examined the percentage of the explanations that are meaningful to humans. We randomly sampled 100 test triples from FB15k-237 and ask human annotators to check their top-1 path chains highly scored by BERTRL. Human judges found that 84% of the path chains make sense, indicating strong explainability.

Model Efficiency

Running Time. The training time of BERTRL gradually increases as the number of training triples grows. The inference time of BERTRL relates to *test-graph* rather than training data size, but it is much slower than light-weight rule-based methods like RuleN. In practice, the running time is highly implementation and device dependent. It can be further improved with more efficient transformer library and better device.

Generative Models. In classification models like BERTRL and KG-BERT, each candidate triple needs to be feed into BERT at least once. This may lead to an efficiency problem for large scale candidate sets. An additional light-weight method to filter the candidate sets may become a performance bottleneck. One potential solution is to leverage pre-trained

Unseen relation	Hits@1	Similar seen relation
/film/film_format	1.000	/film/genre, /film/language
/person/spouse_s./marriage/spouse	1.000	/person/spouse_s./marriage/type_of_union
/pro_athlete/teams./sports_team_roster/team	1.000	/football_player/current_team./sports_team_roster/team
/artist/origin	0.000	-
/record_label/artist	0.100	-
/ethnicity/languages_spoken	0.250	/person/languages

Table 7: Examples of the best and worst performing unseen relation prediction of BERTRL, trained on a 50 relations subset of FB15k-237.

	50 relations	100 relations
KG-BERT	0.266	0.450
BERTRL	0.485	0.500

Table 8: Unseen relation prediction results (Hits@1)

	1,000	2,000	full
Subgraph (edge list)	0.361	0.398	0.463
Combined paths	0.351	0.461	0.505
5 sampled individual paths	0.466	0.490	0.532
10 sampled individual paths	0.449	0.505	0.500
BERTRL (individual paths)	0.441	0.493	0.541

Table 9: Ablation study of BERTRL variants (Hits@1)

generative models such as BART (Lewis et al. 2019) and T5 (Raffel et al. 2020) instead of BERT. The generative model is fed head entity and relation (h, r) and generates the missing tail entity t. By building a prefix tree of all possible entities, and constraining beam search to decode tokens within that tree (De Cao et al. 2020), we could enforce the model to only generate KG entities.

Ablation Study

Table 9 shows the effect of different design choices in BERTRL, mainly knowledge linearization and path sampling. We use the FB15k-237 inductive dataset and its few-shot subset for evaluation.

Combined Paths. As discussed in the approach section, *combined paths* is one way linearizing structural knowledge. Although it includes more information in one input, it does not outperform *individual paths*. This indicates that BERT struggles to learn from complex input when training data is limited, which might be explained by Occam's razor.

Subgraph (Edge List). Edge list is the worst performing linearization option. Linking entities in the input and then recognizing patterns could be more challenging for BERT than reasoning along paths where edges are ordered by their connection.

Path Sampling. We evaluate the performance of *path sampling* by randomly selecting n paths between entities. Path sampling could speed up training as the training data becomes small. The performance is still good even when the number of sampled paths is very small, indicating BERTRL is robust to the size of the training set.

Related Work

Transductive Models. Most existing knowledge graph completion methods are embedding based, such as TransE (Bordes et al. 2013), Complex (Trouillon et al. 2017), ConvE (Dettmers et al. 2018), RotatE (Sun et al. 2019) and TuckER (Balažević, Allen, and Hospedales 2019). These methods learn low-dimensional embedding of entities and relations to capture relational information of the graph. They are naturally transductive and need expensive retraining for new nodes in inductive setting.

Methods like R-GCN (Schlichtkrull et al. 2018), DeepPath (Xiong, Hoang, and Wang 2017), MINERVA (Das et al. 2018) and DIVA (Chen et al. 2018), learn to aggregate information from local subgraph and paths. However, they cannot be directly applied to the inductive setting as entity/node specific embeddings are needed.

Inductive Models. In contrast to the transductive setting, probabilistic rule learning AMIE (Galárraga et al. 2013) and RuleN (Meilicke et al. 2018) could apply learned rules to unseen entities. NeuralLP (Yang, Yang, and Cohen 2017) and DRUM (Sadeghian et al. 2019) learn differentiable rules in an end-to-end manner. GraIL(Teru, Denis, and Hamilton 2020), CoMPILE(Mai et al. 2021) and TACT (Chen et al. 2021) inference through graph neural networks. These methods are in nature inductive as they learn entity irrelevant rules or models and conduct reasoning with knowledge graph information only. Besides these studies, there are methods learning to generate inductive embedding for unseen nodes. Some approaches (Hamilton, Ying, and Leskovec 2017; Bojchevski and Günnemann 2018) rely on the node features which may not be easily acquired in many KGs. Others (Wang et al. 2019; Hamaguchi et al. 2017) generate embedding for unseen nodes by learning to aggregate neighborhood embeddings using GNNs or estimate embedding under translational assumption (Dai et al. 2020). However, those paradigms require a certain number of known entities and cannot be applied to entirely new graphs.

Pre-trained Language Models. Pre-trained language models, such as BERT (Devlin et al. 2019), GPT-2 (Radford et al. 2019), BART (Lewis et al. 2019), T5 (Raffel et al. 2020) and GPT-3 (Brown et al. 2020) revolutionize recent natural language processing studies. LAMA (Petroni et al. 2019) shows that no-tuned pre-trained language models themselves already capture some factual knowledge.

KG-BERT (Yao, Mao, and Luo 2019) leverages BERT in knowledge graph completion. It feeds target triple texts to BERT and learns to predict the triple existence. The simple representation suffers in the inductive setting. it does not learn a general reasoning mechanism like GraIL and BERTRL.

Conclusion

We proposed BERTRL, a pre-trained language model based approach for knowledge graph completion. By taking reasoning path and triple as input to a pre-trained language model, BERTRL naturally handles unseen entities and gains the capability of relational reasoning. In few-shot learning, it outperforms competitive baselines by an even larger margin. It has the potential to generalize to unseen relations in a zero-shot setting. It not only achieves the state-of-the-art results in inductive learning, but also shown to be effective in transductive learning. Overall, this work opens a new direction of combining the power of pre-trained language model and logic reasoning.

Acknowledgements

The authors would like to thank the anonymous reviewers for their thoughtful comments. This research was sponsored in part by Visa Research and the National Science Foundation under Grant No. IIS 1528175. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

Balažević, I.; Allen, C.; and Hospedales, T. M. 2019. TuckER: Tensor Factorization for Knowledge Graph Completion. In *Empirical Methods in Natural Language Processing*.

Bojchevski, A.; and Günnemann, S. 2018. Deep Gaussian Embedding of Attributed Graphs: Unsupervised Inductive Learning via Ranking. In *ICLR*.

Bordes, A.; Usunier, N.; García-Durán, A.; Weston, J.; and Yakhnenko, O. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*.

Broscheit, S.; Ruffinelli, D.; Kochsiek, A.; Betz, P.; and Gemulla, R. 2020. LibKGE-A knowledge graph embedding library for reproducible research. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 165–174.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Carbonneau, M.-A.; Cheplygina, V.; Granger, E.; and Gagnon, G. 2018. Multiple instance learning: A survey of problem characteristics and applications. *Pattern Recognition*, 77: 329–353.

Chen, J.; He, H.; Wu, F.; and Wang, J. 2021. Topology-Aware Correlations Between Relations for Inductive Link Prediction in Knowledge Graphs. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, 6271–6278. AAAI Press.

Chen, W.; Xiong, W.; Yan, X.; and Wang, W. Y. 2018. Variational Knowledge Graph Reasoning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 1823–1832.

Dai, D.; Zheng, H.; Luo, F.; Yang, P.; Chang, B.; and Sui, Z. 2020. Inductively Representing Out-of-Knowledge-Graph Entities by Optimal Estimation Under Translational Assumptions. *arXiv preprint arXiv:2009.12765*.

Das, R.; Dhuliawala, S.; Zaheer, M.; Vilnis, L.; Durugkar, I.; Krishnamurthy, A.; Smola, A.; and McCallum, A. 2018. Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases using Reinforcement Learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

De Cao, N.; Izacard, G.; Riedel, S.; and Petroni, F. 2020. Autoregressive Entity Retrieval. In *International Conference on Learning Representations*.

Dettmers, T.; Pasquale, M.; Pontus, S.; and Riedel, S. 2018. Convolutional 2D Knowledge Graph Embeddings. In *AAAI*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*.

Galárraga, L. A.; Teflioudi, C.; Hose, K.; and Suchanek, F. 2013. AMIE: Association Rule Mining Under Incomplete Evidence in Ontological Knowledge Bases. In *WWW '13*.

Hamaguchi, T.; Oiwa, H.; Shimbo, M.; and Matsumoto, Y. 2017. Knowledge Transfer for Out-of-knowledge-base Entities: A Graph Neural Network Approach. In *IJCAI*.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*.

Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; and Yu, P. S. 2020. A survey on knowledge graphs: Representation, acquisition and applications. *arXiv preprint arXiv:2002.00388*.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Mai, S.; Zheng, S.; Yang, Y.; and Hu, H. 2021. Communicative Message Passing for Inductive Relation Reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4294–4302.

Meilicke, C.; Fink, M.; Wang, Y.; Ruffinelli, D.; Gemulla, R.; and Stuckenschmidt, H. 2018. Fine-Grained Evaluation of Rule- and Embedding-Based Systems for Knowledge Graph Completion. In *ISWC*.

Nickel, M.; Murphy, K.; Tresp, V.; and Gabrilovich, E. 2016. A Review of Relational Machine Learning for Knowledge Graphs. *IEEE*.

Petroni, F.; Rocktäschel, T.; Riedel, S.; Lewis, P.; Bakhtin, A.; Wu, Y.; and Miller, A. 2019. Language Models as Knowledge Bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2463–2473.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140): 1–67.

Sadeghian, A.; Armandpour, M.; Ding, P.; and Wang, D. Z. 2019. DRUM: End-To-End Differentiable Rule Mining On Knowledge Graphs. *Advances in Neural Information Processing Systems*, 32: 15347–15357.

Schick, T.; and Schütze, H. 2020. It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. *arXiv preprint arXiv:2009.07118*.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, 593–607. Springer.

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR*.

Teru, K. K.; Denis, E.; and Hamilton, W. L. 2020. Inductive Relation Prediction by Subgraph Reasoning. *arXiv: Learning*.

Toutanova, K.; Chen, D.; Pantel, P.; Poon, H.; Choudhury, P.; and Gamon, M. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*.

Trouillon, T.; Dance, C. R.; Éric Gaussier; Welbl, J.; Riedel, S.; and Bouchard, G. 2017. Knowledge Graph Completion via Complex Tensor Factorization. *JMLR*.

Wang, P.; Han, J.; Li, C.; and Pan, R. 2019. Logic Attention Based Neighborhood Aggregation for Inductive Knowledge Graph Embedding. In *AAAI*.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. Online: Association for Computational Linguistics.

Xiong, W.; Hoang, T.; and Wang, W. Y. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *EMNLP*.

Yang, F.; Yang, Z.; and Cohen, W. W. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *NIPS*.

Yao, L.; Mao, C.; and Luo, Y. 2019. KG-BERT: BERT for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.