P-INT: A Path-based Interaction Model for Few-shot Knowledge Graph Completion

Jingwen Xu[1,2], **Jing Zhang**[1,2]*, **Xirui Ke**[1,2],
Yuxiao Dong[3], **Hong Chen**[1,2], **Cuiping Li**[1,2], **Yongbin Liu**[4]
[1] Key Laboratory of Data Engineering and Knowledge Engineering of Ministry of Education
[2] School of Information, Renmin University of China
[3] Microsoft Research , [4] School of Computer, University of South China
{xujingwen, zhang-jing, kexirui, chong, licuiping}@ruc.edu.cn
ericdongyx@gmail.com, yongbinliu03@gmail.com

Abstract

Few-shot knowledge graph completion is to infer the unknown facts (i.e., query head-tail entity pairs) of a given relation with only a few observed reference entity pairs. Its general process is to first encode the implicit relation of an entity pair and then match the relation of a query entity pair with the relations of the reference entity pairs. Most existing methods have thus far encoded an entity pair and matched entity pairs by using the direct neighbors of concerned entities. In this paper, we propose the P-INT model for effective few-shot knowledge graph completion. First, P-INT infers and leverages the paths that can expressively encode the relation of two entities. Second, to capture the fine grained matches, P-INT calculates the interactions of paths instead of mixing them for each entity pair. Extensive experimental results demonstrate that P-INT outperforms the state-of-the-art baselines by 11.2–14.2% in terms of Hits@1. Our codes and datasets are online now[1].

1 Introduction

Large scale knowledge graphs (KGs) can benefit various applications, such as question answering (Han et al., 2020), retrieval (Liu et al., 2018), and recommender systems (Wang et al., 2018). The completion of KGs plays a critical role in these applications. To complete KGs, most existing embedding based techniques demand sufficient triplets for each relation as the training data, such as TransE (Bordes et al., 2013), RotatE (Sun et al., 2019), ConvE (Dettmers et al., 2018), and CompGCN (Vashishth et al., 2020). However, the majority of relations have limited (few-shot) triplets in commonly used KGs (Xiong et al., 2018). For example, 96.9% of all the relations are with fewer than 5 triplets in Freebase,

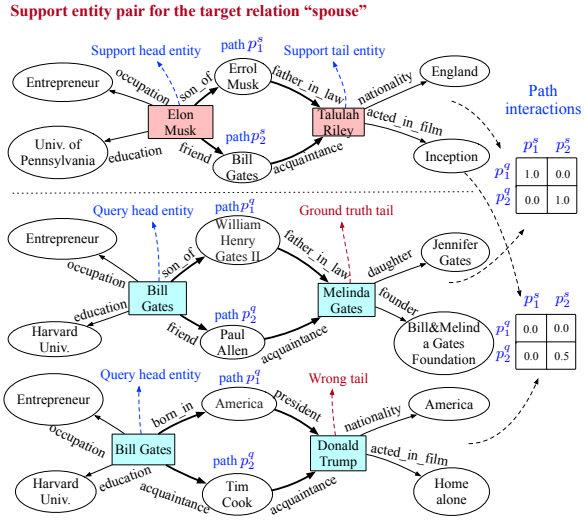


Figure 1: A motivating example.

limiting the expressive power of conventional KG embedding models.

Existing Models and Challenges. Formally, the few-shot KG completion problem (Xiong et al., 2018) is to predict the tail entity t for a query head entity h and a target relation r, where h and t compose the query entity pair and r is described by few-shot entity pairs, named as the support entity pairs. Meta-learning based models are commonly used for solving the problem and the fundamental part is to accurately represent the target relation presented by support entity pairs. Existing methods differ from each other in the way to represent the relation. For example, GMatching (Xiong et al., 2018) simply averages the embeddings of neighbors to represent an entity, and concatenates the head and tail embeddings to represent their relation. FSRL (Zhang et al., 2020a) and FAAN (Sheng et al., 2020) further leverage the attention mechanism to distinguish the different effects of neighbors when representing an entity.

Despite the significant efforts in few-shot KG completion, many issues remain largely unex-

* Corresponding author.
[1] https://github.com/RUCKBReasoning/P-INT

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 385–394
November 7–11, 2021.

plored. First, while the path-based KG reasoning models (e.g., MINERVA (Das et al., 2018), Multihop (Lin et al., 2018), RuleGuider (Lei et al., 2020) and RNNLogic(Qu et al., 2020)) have shown that the paths are expressive to represent the relation between two entities, most existing few-shot learning models ignore them and consider only the direct neighbors of the concerned entities. Take Figure 1 for example, given the support pair ("Elon Musk", "Talulah Riley") to describe a target relation "spouse", we aim to infer the tail entity for the query head entity "Bill Gates". Depending on the direct neighbors[2], the correct tail entity "Melinda Gates" and the wrong tail "Donald Trump" cannot be distinguished, as their neighbors are both similar to the neighbors of the support tail "Talulah Riley". On the contrary, if analyzing the paths between the head and tail entities, we observe that the path "son_of $\rightarrow$ farther_in_law" connecting "Bill Gates" and "Melinda Gates" is more related to the paths that connect the support entity pair than the paths connecting "Bill Gates" and "Donald Trump".

Second, to represent an entity, existing few-shot completion models aggregate all the neighbors' embeddings, which may introduce noises into it. For example, compared with the four neighbors of the support tail "Talulah Riley", "Melinda Gates" has two different neighbors and "Donald Trump" has only one different neighbor. Thus, mixing all the neighbors will likely favor "Donald Trump" as the output, negatively influencing the completion performance. Similarly, to represent an entity pair by its associated paths, mixing all the paths will also dilute the really helpful paths.

Present Work. To address the challenges above, we present P-INT—a Path-based INTeraction model. The basic idea is to leverage the paths from the head to the tail entities to represent an entity pair. Then inspired by the widely adopted interaction focused matching model in information retrieval (Guo et al., 2016) (Tang et al., 2020), P-INT computes the interactions of the path embeddings for capturing the fine grained matches. For example, in Figure 1, instead of learning a good representation for an entity pair based on the paths $\{p_1^s, p_2^s\}$ or $\{p_1^q, p_2^q\}$, we directly calculate the similarities between $\{p_1^s, p_2^s\}$ and $\{p_1^q, p_2^q\}$.

The contributions are summarized as follows:

- We incorporate the paths between entity pairs to solve the few-shot KG completion.
- We propose an interaction based model to match the paths, which can capture the fine grained matches and thus reduce the negative influence of the noisy paths.
- The experimental results on two datasets demonstrate that our model significantly outperforms the best baseline by 11.2–14.2% in terms of Hits@1.

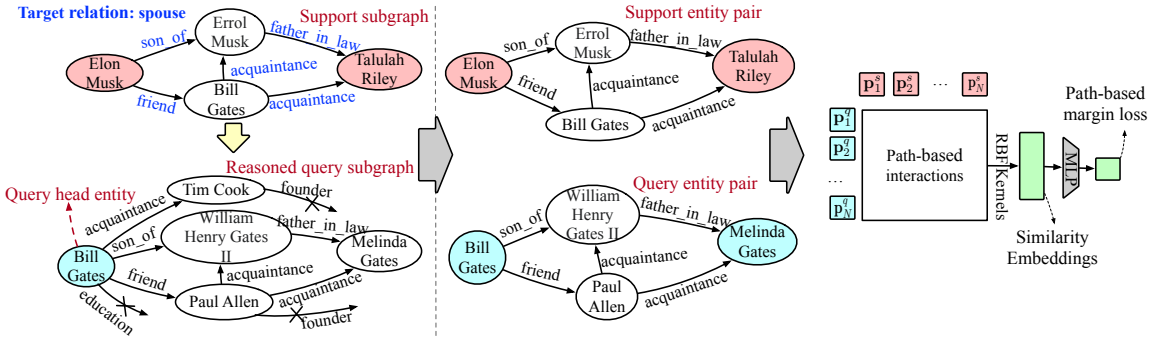
2 Related Work

To solve the few-shot KG completion problem, meta-learning models including the optimization-based and the metric-based categories have been both investigated (Zhang et al., 2021). The optimization-based model is based on the MAML algorithm. For example, MetaR (Chen et al., 2019) represents a relation by the support entity pairs and transfers the meta information of the relation from the support entity pairs to the query entity pairs. Meta-KGR (Lv et al., 2019) and FIRE (Zhang et al., 2020b) represent a relation using the paths traversed by a multi-hop agent and then transfer the meta information similarly as MetaR. However, the latter two models ignore the interactions of paths.

The metric-based model measures the similarity between two entity pairs based on the representations of the entity pairs. For example, GMatching (Xiong et al., 2018) averages the neighbors of an entity, concatenates the head and the tail for an entity pair, applies a transformer to aggregate all the support entity pairs, and finally matches the aggregated support pairs with each query entity pair by a LSTM-based meta learner. FSRL (Zhang et al., 2020a) aggregates the neighbors by a fixed attention mechanism, applies a recurrent autoencoder to aggregate all the support pairs, and matches the aggregation with the query entity pair by LSTM. FAAN (Sheng et al., 2020) aggregates the neighbors by a dynamic attention mechanism, applies a transformer to encode an entity pair, aggregates all the support pairs by an attention mechanism, and finally matches the aggregation with the query pair by dot product.

This paper studies a metric-based meta-learning model, which changes the representation-based similarity measurement into an interaction-based measurement on the paths connecting the head and the tail entities.

[2]Note that neighbors of an entity denote the neighboring **relations** such as "occupation" of "Elon Musk" but not the concrete neighboring entities such as "Entrepreneur".



(a) **Reasoning** (b) **Matching**

Figure 2: The framework of P-INT. The reasoning component is to infer the paths from a query head entity to the candidate tail entities and the matching component is to compute the interactions of paths.

3 Problem Definition

A KG is denoted by $\mathcal{G} = \{(h, r, t) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}$, where $\mathcal{E}$ and $\mathcal{R}$ denote the sets of its entities and relations, respectively.

Problem 1. ***Few-Shot KG Completion.*** *Given a relation $r \in \mathcal{R}$ and its support entity pairs $\mathcal{S}_r = \{(h_s, t_s)|(h_s, r, t_s) \in \mathcal{G}\}$ with small size $|\mathcal{S}_r|$[3], for each query head entity h_q, the goal is to complete the triplet $(h_q, r, ?)$ with $?$ as the ground truth tail t_q to be predicted from the candidate entity[4] set $\mathcal{C}_q$.*

We follow the common setting of few-shot KG completion (Xiong et al., 2018). Specifically, predicting triplets for a relation r is defined as a task. For each task of r, the related entity pairs are provided, that is, $\mathcal{D}_r = \{\mathcal{S}_r, \mathcal{Q}_r\}$, where $\mathcal{S}_r$ consists of K-shot support entity pairs and $\mathcal{Q}_r = \{(h_q, t_q)\}$ contains the query entity pairs, each of which is composed of a query head h_q and a ground truth tail t_q. The relations $\mathcal{R} = \{r\}$ are separated into $\mathcal{R}_{\text{background}}$, $\mathcal{R}_{\text{meta-train}}$, $\mathcal{R}_{\text{meta-validation}}$ and $\mathcal{R}_{\text{meta-test}}$. Correspondingly, their triplets form the background knowledge graph $\mathcal{G}_{\text{background}}$, the meta-training set $\mathbb{D}_{\text{meta-train}}$, the meta-validation set $\mathbb{D}_{\text{meta-validatiaon}}$ and the meta-test set $\mathbb{D}_{\text{meta-test}}$.

4 The Proposed Model

This section introduces the proposed P-INT, which consists of a reasoning component to infer the paths from a query head entity to the candidate tail entities according to the support entity pair, and a matching component to compute the interactions between the support entity pairs and each query entity pair. In addition, the attention mechanism that considers the relevance to the target relation is applied to each path. A kernel aggregation function is adopted to extract the similarity features from the interactions. Figure 2 presents the whole framework of P-INT.

4.1 The Reasoning Component

Given a query head entity h_q, the reasoning component aims to infer the subgraph that might contain the ground truth tail t_q. One straightforward way is to extend the multi-hop neighbors of h_q, which would make the resultant subgraph extremely large with the increase of the hop number. Therefore, we restrict the neighbor size at each hop and extend a support-relevant subgraph. Below, we introduce how to extract the support subgraph and reason the query subgraph.

Extract the Support Subgraph. For each support entity pair $(h_s, t_s) \in \mathcal{S}_r$, we leverage the two-side BFS algorithm (Xiong et al., 2017b) to extract its support subgraph. Specifically, given the maximal path length T, we perform the $\lceil T/2 \rceil$-hop BFS starting from h_s, collecting neighbors of different hops, i.e., $\mathcal{N}_1^h, \cdots, \mathcal{N}_{\lceil T/2 \rceil}^h$, and the corresponding paths of different lengths from h_s to these neighbors—left paths. Similarly, starting from t_s, we also perform the $\lfloor T/2 \rfloor$-hop BFS to obtain neighbors $\mathcal{N}_1^t, \cdots, \mathcal{N}_{\lfloor T/2 \rfloor}^t$ and the corresponding paths from t_s to them—right paths. We then calculate the intersection $\{h_s\} \cap \mathcal{N}_1^t, \mathcal{N}_1^h \cap \mathcal{N}_1^t, \mathcal{N}_2^h \cap \mathcal{N}_1^t, \mathcal{N}_2^h \cap \mathcal{N}_2^t, \cdots, \mathcal{N}_{\lceil T/2 \rceil}^h \cap \mathcal{N}_{\lfloor T/2 \rfloor}^t$. The intersected neighbors are used to connect the left and right paths to generate paths from h_s to h_t of different lengths, denoted as $\mathcal{P}(h_s, t_s)$. When we restrict the maximal path length as T, two-side $T/2$-hop BFS can reduce the path search space and improve the efficiency compared with one T-hop BFS. Take

[3] $|\mathcal{S}_r|$ is set as 1 or 5 in the experiments.

[4] Candidates are constrained by entity types (Xiong et al., 2018).

the support pair ("Elon Musk", "Talulah Riley") in Figure 2 as an example, when $T = 3$, we need to enumerate the left paths "Elon Musk $\xrightarrow{son\_of}$ Errol Musk" (length 1), "Elon Musk $\xrightarrow{friend}$ Bill Gates" (length 1), "Elon Musk $\xrightarrow{son\_of}$ Errol Musk $\xrightarrow{acquaintance^{-1}}$ Bill Gates" (length 2), and "Elon Musk $\xrightarrow{friend}$ Bill Gates $\xrightarrow{acquaintance}$ Errol Musk" (length 2). We also need to enumerate the right paths "Talulah Riley $\xrightarrow{father\_in\_law^{-1}}$ Errol Musk" and "Talulah Riley $\xrightarrow{acquaintance^{-1}}$ Bill Gates" of length 1. Thus $\mathcal{N}_1^h = \mathcal{N}_2^h = \mathcal{N}_1^t = \{\text{Errol Musk}, \text{Bill Gates}\}$. After performing $\mathcal{N}_1^h \cap \mathcal{N}_1^t$ and $\mathcal{N}_2^h \cap \mathcal{N}_1^t$ and using the intersected entities to connect the left and the right paths, we can obtain the paths "Elon Musk $\xrightarrow{son\_of}$ Errol Musk $\xrightarrow{father\_in\_law}$ Talulah Riley" (length 2), "Elon Musk $\xrightarrow{friend}$ Bill Gates $\xrightarrow{acquaintance}$ Talulah Riley" (length 2), "Elon Musk $\xrightarrow{son\_of}$ Errol Musk $\xrightarrow{acquaintance^{-1}}$ Bill Gates $\xrightarrow{acquaintance}$ Talulah Riley" (length 3), and "Elon Musk $\xrightarrow{friend}$ Bill Gates $\xrightarrow{acquaintance}$ Errol Musk $\xrightarrow{father\_in\_law}$ Talulah Riley" (length 3).

We aggregate the relations in $\mathcal{P}(h_s, t_s)$ as the set of the support relations, denoted as $\mathcal{R}^s$. Finally we merge the support relations of all the support pairs in $\mathcal{S}_r$ into a unified $\mathcal{R}^s$. In the above example, $\mathcal{R}^s$ is composed of "son_of", "friend", "father_in_law" and "acquaintance".

Reason the Query Subgraph. Before retrieving the query subgraph, we calculate the cosine similarity a_{ij} between each pair of relations in $\mathcal{G}$, i.e., $a_{ij} = \frac{\mathbf{r}_i \cdot \mathbf{r}_j}{\|\mathbf{r}_i\| \cdot \|\mathbf{r}_j\|}$, where $\mathbf{r}_i$ is the pretrained embedding by TransE on $\mathcal{G}_{\text{background}}$. To reason a query subgraph, we restrict the number of the maximal extended neighbors at each hop as L. At the τ-th hop, we extend L entities with N_L neighbors in total. For each neighbor r_i[5], we retrieve the similarities of r_i with every support relation in $\mathcal{R}^s$, i.e.,$\{a_{ij}\}_{j=1}^{|\mathcal{R}^s|}$, and get the maximal value $a_i^{\max} = \max_j\{a_{ij}\}$. Then we sample L neighbors from N_L neighbors with the probability of each neighbor as:

$$P(r_i, t_i) = \begin{cases} 1/N_L & \text{with probability } \epsilon; \\ a_i^{\max} / \sum_{i=1}^{N_L} a_i^{\max} & \text{with probability } 1-\epsilon, \end{cases} \quad (1)$$

where ϵ is a hyper parameter to determine the probability of random sampling, which is inspired by dropout (Srivastava et al., 2014). Specifically, during training, we inject random sampling by setting $\epsilon = 0.8$ to increase the variance of the negative instances. For testing, we set $\epsilon = 0$ and directly return the top-L neighbors at each hop to make the reasoned query subgraph relevant to the support subgraph as much as possible.

Note we treat the one-to-many relation with the same relation but different tail entities as different neighbors to be extended, because extending different tails will generate different paths. After T hops, we extend a query subgraph with at most $T \times L$ entities. Simultaneous to reasoning, we can trace all paths from h_q to every extended entity in the subgraph, which is denoted as $\mathcal{P}(h_q, t_q)$ for (h_q, t_q).

4.2 The Matching Component

Path-based Matching. Path-based matching component captures the fine-grained matches of paths. Specifically, for each tuple (h_s, t_s, h_q, t_q) containing a support entity pair (h_s, t_s) and a query entity pair (h_q, t_q), we (1) represent each path and (2) compute the path interactions between $\mathcal{P}(h_s, t_s)$ and $\mathcal{P}(h_q, t_q)$ with (3) path attentions.

Represent Paths. We use GRU to embed each path $p \in \mathcal{P}(h_s, t_s) \cup \mathcal{P}(h_q, t_q)$. Specifically, we first represent a path by a sequence of relations in it, i.e., $p = (r_1, r_2, \cdots, r_{|p|})$. Then we treat the sequence of the relation embeddings pretrained by TransE—$(\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_{|p|})$—as the input of a GRU model to generate an aggregated embedding as the path embedding $\mathbf{p}$. We ignore the entities in a path for the reason that different entity pairs describing the same relation share similar relation sequence patterns rather than similar entity sequence patterns. For example in Figure 2, the relation sequence "son_of $\rightarrow$ father_in_law" is shared between ("Elon Musk", "Talulah Riley") and ("Bill Gates","Melinda Gates"), but the entities "Errol Musk" and "William Henry Gates II" on the two paths are different.

Calculate Path Interactions. Given the set of the path embeddings $\{\mathbf{p}_1^s, \mathbf{p}_2^s, \cdots, \mathbf{p}_N^s\}$ of (h_s, t_s) and $\{\mathbf{p}_1^q, \mathbf{p}_2^q, \cdots, \mathbf{p}_N^q\}$ of (h_q, t_q), we build the similarity matrix $\mathbf{S}^{\mathcal{P}}$ with each element $s_{ij}^{\mathcal{P}}$ representing the similarity between $\mathbf{p}_i^s$ and $\mathbf{p}_j^q$. Then we apply an aggregation function to extract the similarity fea-

[5] As mentioned in Section 1, a neighbor represents a neighboring relation instead of a neighboring entity.

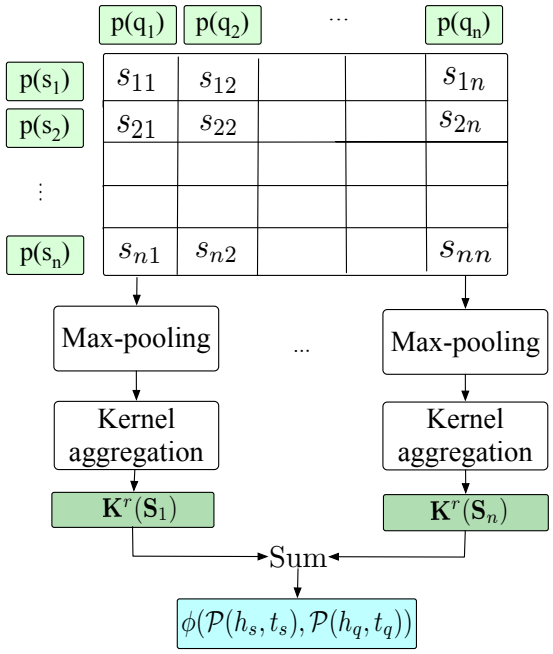


Figure 3: The interaction of paths.

tures from $\mathbf{S}^{\mathcal{P}}$. Since the paths extracted from the graphs are disordered and independent from each other, the RBF kernel aggregation function (Xiong et al., 2017a) is used to extract the accumulated similarity features.

Specifically, we first apply a max pooling operation on each row $\mathbf{S}_i^{\mathcal{P}} = \{s_{ij}^{\mathcal{P}}\}_{j=1}^{N}$ to get the maximal similarity $s_i^{\max} = \max_j\{s_{ij}^{\mathcal{P}}\}$, which represents the most probably matched counterpart in $\mathcal{P}(h_s, t_s)$ for $p_i \in \mathcal{P}(h_q, t_q)$. Next, $s_i^{\max}$ is transformed into a Γ-length feature vector $\mathbf{K}(s_i^{\max})$ by Γ RBF kernels, where each feature $K_\gamma(s_i^{\max})$ is made by the γ-th RBF kernel with mean μ_γ and variance σ_γ (Eq.(2)). All the kernels represent a distribution of the similarities and thus $K_\gamma(s_i^{\max})$ represents how likely $s_i^{\max}$ is close to μ_γ, i.e., the γ-th similarity feature. The transformation from the one-dimension similarity to the Γ-dimension similarity vector can improve the discrimination ability of the similarity features. Finally, $[K_1(s_i^{\max}), \cdots, K_\Gamma(s_i^{\max})]$ of different rows ($i = 1, \cdots, |\mathcal{P}(h_q, t_q)|$) are summed into a similarity embedding $\phi(\mathcal{P}(h_s, t_s), \mathcal{P}(h_q, t_q))$ (abbreviated as ϕ in Eq.(3)), which represents the similarity between (h_q, t_q) and (h_s, t_s).

The concrete operations are shown in Figure 3 and summarized as follows:

$$K_\gamma(s_i^{\max}) = \exp\left[-\frac{(s_i^{\max} - \mu_\gamma)^2}{2\sigma_\gamma^2}\right], \quad (2)$$

$$\phi = \sum_{i=1}^{|\mathcal{P}(h_q,t_q)|} \log\left[K_1(s_i^{\max}), \cdots, K_\Gamma(s_i^{\max})\right], \quad (3)$$

In the above statement, N denotes the maximal number of paths of all the entity pairs. $\mathbf{S}^{\mathcal{P}}$ will be padded zero if the real number $|\mathcal{P}(h, t)|$ is less than N. The kernel with $\mu = 1$ and $\sigma \to 0$ only considers the exact matches, while others can capture the semantic matches between paths.

Add Path Attentions. A path can be a more confident evidence of the final matching result if it is more relevant to the target relation. For example, in Figure 2, in the subgraph of the support entity pair, the path "son_of → father_in_law" is more relevant to the target relation "spouse" than the path "friend → acquaintance". Thus we calculate relation-aware attentions for different paths to distinguish their effects on the final matching result. Specifically, the attention score α_j for a path $p_j^s \in \mathcal{P}(h_s, t_s)$ and the attention score β_i for a path $p_j^q \in \mathcal{P}(h_q, t_q)$ are computed respectively as:

$$\begin{aligned} \alpha_j &= \text{softmax}((\mathbf{t}^s - \mathbf{h}^s)\mathbf{W}\mathbf{p}_j^s + b), \quad (4) \\ \beta_i &= \text{softmax}((\mathbf{t}^q - \mathbf{h}^q)\mathbf{W}\mathbf{p}_i^q + b), \end{aligned}$$

where $\mathbf{t}^s - \mathbf{h}^s$ represents the semantics of the target relation expressed by the support entity pair. Inspired by FAAN (Sheng et al., 2020), we use $(\mathbf{t}^s - \mathbf{h}^s)\mathbf{W}\mathbf{p}_j^s$ to represent the relevance of the path p_j^s to the target relation. Similarly, $\mathbf{t}^q - \mathbf{h}^q$ represents the semantics of the relation expressed by the query entity pair and its relevance to a path is computed in the same way as the support pair. Then the similarity $s_{ij}^{\mathcal{P}}$ is updated by the attentions:

$$s_{ij}^{\mathcal{P}} = \mathbf{p}_i^q \times \mathbf{p}_j^s \times \alpha_j \times \beta_i. \quad (5)$$

Training Process. We use a MLP layer to convert the path based similarity embedding ϕ into a score and then perform max pooling on the scores of all K-shot support pairs to output a single score $g(h_q, t_q, \mathcal{S}_r)$, i.e.,

$$g(h_q, t_q, \mathcal{S}_r) = \max_{(h_s,t_s)\in\mathcal{S}_r} \text{MLP}(\phi(\mathcal{P}(h_s, t_s), \mathcal{P}(h_q, t_q))). \quad (6)$$

The pairwise loss function with regard to r is defined as:

$$\mathcal{L}_r = \sum_{\substack{(h_q,t_q)\\ \in \mathcal{Q}_r}} \sum_{\substack{(h_q,t_q^-)\\ \in \mathcal{Q}_r^-}} \max\{0, m + g(h_q, t_q^-, \mathcal{S}_q) - g(h_q, t_q, \mathcal{S}_q))\}, \tag{7}$$

where m is a margin separating positive and negative instances. $\mathcal{L}_r$ is our training objective to be minimized. $\mathcal{Q}_r^- = \{(h_q, t_q^-)\}$ are the corrupt query entity pairs corresponding to the correct query entity pairs $\mathcal{Q}_r = \{(h_q, t_q)\}$. The parameters of the relation embeddings, the GRU, the attention layer, and the MLP layers are to be optimized. The negative tail entity t_q^- is sampled from the reasoned query subgraph explained in Section 4.1. To improve the expressive ability of GRU when training, we avoid sampling the confusing negative tail entities, i.e., those that belong to the paths containing the ground truth tail t_q. For example in Figure 2, in the reasoned query subgraph, "William Henry Gates II" and "Paul Allen" won't be sampled as negative tails as they are in the same path with the ground truth "Melinda Gates". We discard (h_q, t_q) if the ground truth tail entity t_q cannot be found in the reasoned query subgraph or the given support entity pairs are all disconnected.

Time Complexity Analysis. The support subgraphs are preprocessed before training. We need $O(T \times L)$ time complexity to reason each query subgraph, $O(N^2)$ to calculate the path interactions. The total time complexity $O(T \times L + N^2)$ is acceptable when we set L = 100, T = 3 and N = 50.

Few-Shot Prediction. Given a new relation r in $\mathbb{D}_{\text{meta-test}}$ or $\mathbb{D}_{\text{meta-validation}}$, for a query head h_q, we first reason the query subgraph and then compute a score for each candidate $t \in \mathcal{C}_q$ as the path-based similarity $g(h_q, t, \mathcal{S}_r)$. We predict the correct tail according to the scores.

5 Experiments

5.1 Experimental Settings

Dataset. We conduct our experiments on two datasets—NELL-One[6] and FB15k237-One. NELL-One is a well adopted benchmark dataset which is published by Xiong et al. (Xiong et al., 2018). Since Wiki-One used by Xiong et al. (Xiong et al., 2018) is too sparse to present the effect of our model, we build another dataset—FB15k237-One—from the dataset of FB15k-237 (Toutanova et al., 2015). Following the few-shot dataset construction process (Xiong et al., 2018), we select the relations from FB15k-237 with less than 500 but more than 50 triples to compose the few-shot tasks, and treat the rest of relations and the corresponding triples as the background knowledge graph.

NELL-one and FB15k237-one contain 67 and 45 few-shot tasks respectively composed by the selected relations. Correspondingly, the partition 51/5/11 of the 67 tasks and the partition 32/5/8 of the 45 tasks are used for training/validation/testing. The details of the data statistics are shown in Table 2.

Comparison Methods. We compare with four existing few-shot learning baselines, MetaR, GMatching, FSRL, and FAAN (cf. Section 2 for model details), which are all evaluated in their papers on the two benchmarks for few-shot KG completion. The general KG embedding models such as TransE, DisMult, ComplEx, and RotatE have been proved to be worse than the few-shot learning models by the above baselines. Thus we leave out their results due to the page limit.

Evaluation Metrics. We evaluate the ranking of the ground truth tail entity t_q for each query head entity h_q among the candidates $\mathcal{C}_q$ by MRR and Hits@k. MRR is the mean reciprocal rank and Hits@k is the proportion of the ground truth entities ranked in the top k, with $k = 1, 5, 10$.

Implementation Details. For all the models, we initialize the entity and relation embeddings by TransE. We set M, the maximal neighbor size, to 300 and the embedding dimension as 100 for NELL-One and FB15k237-One. The K-shot ($K = 1, 5$) support pairs are selected randomly and fixed for all the models. We run the released code and adopt the default hyperparameters for each baseline. For GMatching, we choose the transformer and mean pooling to aggregate all the support pairs. For MetaR, we choose the pretrained setting. We follow GMatching to constrain the candidates by entity types[7].

For our model, we set the maximal path number N as 50 and the maximal path length T as 3. We set L, the maximal neighbors to be extended at each hop, as 100. For the MLP on ϕ, we apply a

[6] https://github.com/xwhan/One-shot-Relational-Learning

[7] The reported results of FSRL on NELL-One are different from the original results as FSRL specifies 1000 candidates instead of type-restricted candidates.

NELL-One	MRR		Hits@10		Hits@5		Hits@1	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
GMatching	0.180	0.184	0.281	0.279	0.231	0.230	0.129	0.129
FSRL	0.149	0.142	0.286	0.284	0.184	0.175	0.102	0.088
FAAN	0.190	0.276	0.330	0.426	0.252	0.366	0.123	0.192
MetaR	0.227	0.227	0.344	0.340	0.297	0.282	0.163	0.164
P-INT	**0.389**	**0.405**	**0.546**	**0.506**	**0.518**	**0.503**	**0.275**	**0.317**
FB15k237-One	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
GMatching	0.166	0.201	0.340	0.363	0.253	0.277	0.076	0.114
FSRL	0.222	0.227	0.498	0.491	0.399	0.367	0.090	0.106
FAAN	0.256	0.304	**0.510**	**0.541**	0.405	0.434	0.138	0.188
MetaR	0.152	0.203	0.339	0.377	0.268	0.291	0.058	0.107
P-INT	**0.378**	**0.401**	0.501	0.483	**0.493**	**0.482**	**0.272**	**0.330**

Table 1: Overall performance of few-shot KG completion on NELL-One and FB15k237-One.

Dataset	#Ent	#Rel	#Tri	#Tasks	Con.
NELL-One	68,545	358	181,109	67	0.777
FB15k237-One	14,478	237	309,621	45	0.99

Table 2: Data statistics. #Ent, #Rel, #Tri and #Tasks are the number of the entities, relations, triplets and tasks respectively, Con. indicates the connectivity, which is the ratio of the connected entity pairs.

Variants	MRR	Hits@10	Hits@5	Hits@1
Path-based Mixture Matching				
P-MIX (aveP)	0.281	0.529	0.449	0.145
P-MIX (fixAttP)	0.344	0.520	0.460	0.225
P-MIX (dynAttP)	0.325	0.524	0.439	0.225
Path-based Interaction Matching				
P-INT(noAtt)	0.349	0.537	0.471	0.252
P-INT(sAtt)	**0.389**	**0.546**	**0.518**	**0.275**
P-INT(sqAtt)	0.376	0.527	0.493	0.274

Table 3: Results of the path-based mixture/interaction matching on NELL-One (One-shot learning).

21×200 linear layer, a 200×1 linear layer plus a sigmoid activation function. In Eq.(2), we use 20 semantic matching kernels, where μ is from 0.025 to 0.975 with interval 0.05 for calculating ϕ, σ is set as 0.1. And we use an exact matching kernel with $\mu = 1.0$ and $\sigma = 10^{-3}$ for our model. The margin m in Eq.(7) is set as 1. By default, we merge the results of the K-shot support pairs by max pooling strategy. The attention α is set by Eq.(4) and β is set to 1 for calculating ϕ.

5.2 Experimental Results

Overall Performance. Table 1 shows the overall performance. Compared with the best results of the baselines, P-INT significantly improves 11.2% Hits@1 on NELL-One and 13.4% Hits@1 on FB15k237-One respectively for 1-shot link prediction. For 5-shot link prediction, P-INT improves 12.5% on NELL-One and 14.2% on FB15k237-One in terms of Hits@1. The results present the superior advantages of P-INT.

Effect of Path Disconnection. Since a few disconnected support entities pairs in the test/valid set are discarded when evaluating P-INT, it might be a little bit unfair to compare its results with other methods as well as comparing its own 1-shot with 5-shot results. And that might also result in the poorer performance of P-INT on 5-shot than 1-shot of NELL-One. To make a completely fair comparison, we make a variant dataset NELL-One-Filter by filtering out disconnected entity pairs in the test/valid set and present the performance of P-INT on it in Figure 4. Compared with 1-shot, the 5-shot increases by 5.7%, 1.2%, 8.7%, 0.7% on MRR, Hits@10, Hits@5, Hits@1 on NELL-One-Filter. We also present the result of MetaR and FAAN, the two best baselines, on this filtered test set. It shows that our model is consistently better than the baselines.

Effect of Path-Based Interaction. We study the effect of the proposed interaction matching between paths. We create P-MIX, a variant of P-INT,

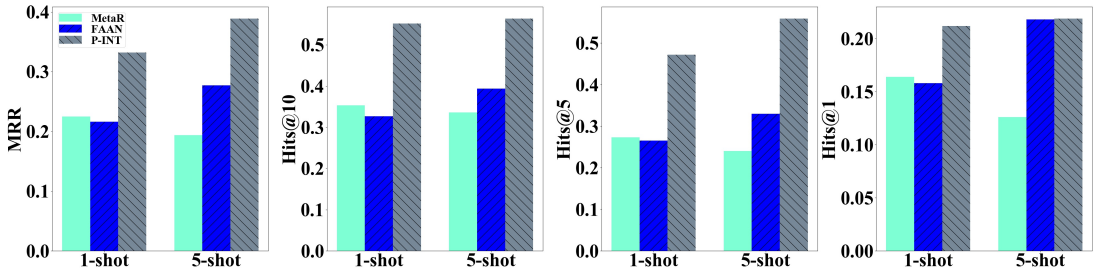


Figure 4: Performance on NELL-One-Filter.

by mixing the embeddings of all the paths for the support entity pair and also for the query entity pair, and then matching the mixed path embeddings between them, instead of computing the interaction matrix between each pair of their paths. We try different pooling strategies, including the mean pooling of GMatching (Xiong et al., 2018), the fixed attention pooling of FSRL (Zhang et al., 2020a) (i.e., the attention is only based on the path embedding itself) and the dynamic attention pooling of FAAN (Sheng et al., 2020) (Eq.(4)). We present the results of the path-based mixture matching and interaction matching methods for one-shot learning on NELL-One in Table 3. From the results, we can see that all the interaction matching methods significantly outperform the mixture matching methods by 2.7-5.0% in terms of Hits@1. This is because the interaction based matching can capture both the exact matches and the semantic matches in a fine-grained manner, which can reduce the negative effect of the noisy paths.

We also try various attention strategies on paths and present the results of P-INT in Table 3. P-INT(noAtt) pays the same attention on different paths, i.e., $\alpha_j = 1, \beta_i = 1, \forall i, j$ in Eq.(4). P-INT(sAtt) sets $\beta_i = 1$ but α_j by Eq.(4). P-INT(sqAtt) sets both α_j and β_i by Eq.(4). The results show that P-INT(sAtt) performs the best among all the attention strategies.

Effect of Pooling Strategies on Support Pairs. We study the effect of different pooling strategies on support pairs. We try max pooling, which maps the similarity embedding of each support pair into a score and then retrieve the maximal score. We also try attention pooling, which computes the attention for the similarity embedding of each support pair according to the embedding itself, aggregates all the embeddings by the attentions, and then maps the mixed embedding into a score. We present the results of the two pooling strategies for P-INT on NELL-One in Figure 5. The max pooling strategy

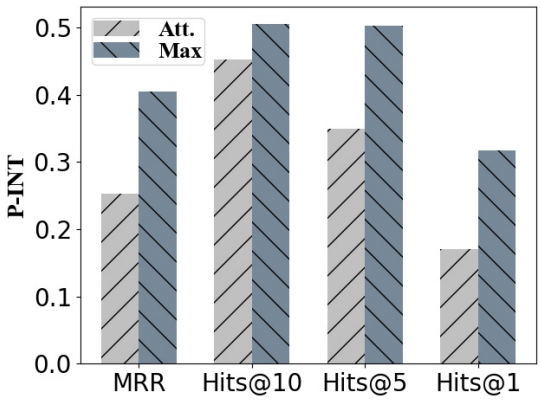


Figure 5: Results of attention (i.e. Att.) and max pooling strategies for 5-shot support pairs on NELL-One.

increases by 15.2%, 5.3%, 15.3%, 14.7% on MRR, Hits@10, Hits@5, Hits@1 on NELL-One compared with attention pooling strategy .The results show that the simplest max pooling strategy consistently performs the best for different evaluation metrics.

Discussions. There are some limitations of the proposed P-INT. First, the model heavily depending on paths is affected by the sparsity of KGs. It fails when the head and the tail entities are disconnected. Second, the reasoning result is quite sensitive to the selection of the support entity pairs. When the paths that can explain the relation between the query head and the correct answer are totally different from those of the support pairs, it fails to reason the correct answer. Third, we follow all the existing work (Chen et al., 2019; Xiong et al., 2018; Sheng et al., 2020; Zhang et al., 2020a) to put the ground truth answer at the first of the candidate entities and rank the candidates according to the predictive scores. However, it may encounter the problem that candidates with the same predictive scores cannot be distinguished when predicting. We will address these remaining problems in the future.

6 Conclusion

This paper solves the few-shot knowledge graph completion by building the interactions of paths between the support entity pair and the query entity pair. The paths are expressive to represent a relation and the interaction matching can capture fine grained matches between entity pairs to reduce the negative influence of the noisy paths. Experimental results on two benchmarks show that P-INT achieves the best performance among all the state-of-the-art few-shot learning models.

Acknowledgments

This work is supported by National Natural Science Foundation of China (62076245, 62072460, 62172424); National Key Research & Develop Plan(2018YFB1004401); Beijing Natural Science Foundation (4212022); CCF-Tencent Open Fund.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems*, pages 1–9.

Mingyang Chen, Wen Zhang, Wei Zhang, Qiang Chen, and Huajun Chen. 2019. Meta relational learning for few-shot link prediction in knowledge graphs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4216–4225.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over knowledge bases with reinforcement learning. In *Proceedings of the 6th Workshop on Automated Knowledge Base Construction*, pages 1–18.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64.

Jiale Han, Bo Cheng, and Xu Wang. 2020. Open domain question answering based on text enhanced knowledge graph with hyperedge infusion. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1475–1481.

Deren Lei, Gangrong Jiang, Xiaotao Gu, Kexuan Sun, Yuning Mao, and Xiang Ren. 2020. Learning collaborative agents with rule guidance for knowledge graph reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 8541–8547.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3243–3253.

Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Entity-duet neural ranking: Understanding the role of knowledge graph semantics in neural information retrieval. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2395–2405.

Xin Lv, Yuxian Gu, Xu Han, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2019. Adapting meta knowledge graph information for multi-hop reasoning over few-shot relations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. 2020. Rnnlogic: Learning logic rules for reasoning on knowledge graphs. In *Proceedings of the 2020 International Conference on Machine Learning*.

Jiawei Sheng, Shu Guo, Zhenyu Chen, Juwei Yue, Lihong Wang, Tingwen Liu, and Hongbo Xu. 2020. Adaptive attentional network for few-shot knowledge graph completion. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 1681–1691.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *Proceedings of the 7th International Conference on Learning Representations*.

Xiaobin Tang, Jing Zhang, Bo Chen, Yang Yang, Hong Chen, and Cuiping Li. 2020. Bert-int: A bert-based interaction model for knowledge graph alignment. In *IJCAI*, pages 3174–3180.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the*

2015 conference on empirical methods in natural language processing, pages 1499–1509.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020. Composition-based multi-relational graph convolutional networks. In *Proceedings of the 7th International Conference on Learning Representations*.

Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 417–426.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017a. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 55–64.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017b. DeepPath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-shot relational learning for knowledge graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1980–1990.

Chuxu Zhang, Huaxiu Yao, Chao Huang, Meng Jiang, Zhenhui Li, and Nitesh V Chawla. 2020a. Few-shot knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3041–3048.

Chuxu Zhang, Lu Yu, Mandana Saebi, Meng Jiang, and Nitesh Chawla. 2020b. Few-shot multi-hop relation reasoning over knowledge bases. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 580–585.

Jing Zhang, Bo Chen, Lingxi Zhang, Xirui Ke, and Haipeng Ding. 2021. Neural, symbolic and neural-symbolic reasoning on knowledge graphs. *AI Open*, 2:14–35.