Tackling Solitary Entities for Few-Shot Knowledge Graph Completion

Yi Liang[1], Shuai Zhao[1](✉), Bo Cheng[1], Yuwei Yin[2], and Hao Yang[2]

[1] Beijing University of Posts and Telecommunications, No. 10. Xitucheng Road, Haidian, Beijing, China
{liangyi,zhaoshuaiby,chengbo}@bupt.edu.cn

[2] 2012 Labs, Huawei Technologies, CO., LTD., Shenzhen, China
{yinyuwei,yanghao30}@huawei.com

Abstract. Few-Shot Knowledge Graph Completion (FSKGC) aims to predict new facts for relations with only a few observed instances in Knowledge Graph. Existing FSKGC models mostly tackle this problem by devising an effective graph encoder to enhance entity representations with features from their directed neighbors. However, due to the sparsity and entity diversity of large-scale KG, these approaches fail to generate reliable embeddings for solitary entities, which only have an extremely limited number of neighbors in KG. In this paper, we attempt to mitigate this issue by modeling semantic correlations between entities within an FSKGC task and propose our model YANA (**Y**ou **A**re **N**ot **A**lone). Specifically, YANA introduces four novel abstract relations to represent inner- and cross- pair entity correlations and construct a Local Pattern Graph (LPG) from the entities. Based on LPG, YANA devises a Highway R-GCN to capture hidden dependencies of entities. Moreover, a query-aware gating mechanism is proposed to combine topology signals from LPG and semantic information learned from entity's directed neighbors with a heterogeneous graph attention network. Experiments show that YANA outperforms the prevailing FSKGC models on two datasets, and the ablation studies prove the effectiveness of Local Pattern Graph design.

Keywords: Knowledge graph completion · Few-shot learning · Link prediction · Graph learning · Representation learning

1 Introduction

Knowledge Graph is a vital resource for many downstream applications such as recommendation system [21], question answering [12], urban computing [10], fault detection [7] and medical data processing [5]. A knowledge graph (KG) represents facts in the form of triple (h, r, t), describing relation r between head entity h and tail entity t. Despite their large scale, KGs are usually incomplete. Thus, knowledge graph completion (KGC), which is to infer new facts from existing triples [17], has attracted widely attention in recent years.

G. Memmi et al. (Eds.): KSEM 2022, LNAI 13368, pp. 227–239, 2022.
https://doi.org/10.1007/978-3-031-10983-6_18

Current KGC approaches roughly follow encoder-decoder framework [13], while encoder focuses on learning entity and relation embeddings and the decoder aims to compute scores for new queries. These models mainly rely on sufficient triples for each relation and entity to learn a good representation. Unfortunately, due to the long-tail phenomenon in real-world KGs, a large proportion of relations have only a *few* triples [23]. Predicting new facts for these relations is called few-shot knowledge graph completion (FSKGC) [2,24]. Generally, a K-shot knowledge graph completion task aims to predict tail entities for query $(h_q, rel, ?)$ with only K associated entity pairs of relation rel as the support set. Figure 1(a) gives an example of FSKGC task.

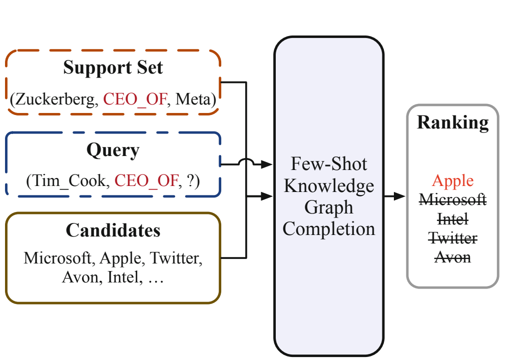
Previous researches on FSKGC can be roughly grouped into sub-graph based models and full graph models according to different network architecture. Sub-graph models [9,14,23,24] capture signals from entity's one-hop neighbors. Unlike sub-graph models, full graph models [6,11,13,22] mainly encode entities with multi-layer message passing neutral network [13] to capture features from multi-hop neighbors.

Despite their variant designs, all these methods only consider **intrinsic** topology signals from existing KG, and thus have two major limitations. **1) Graph sparsity**: A large amount of entities has a minimal number of neighbors in KG. For example, 99.2% entities of Wikidata [20] have fewer than five neighbors. **2) Neighbor diversity**: In a real-world KG, neighbors of entities are mostly irrelevant to new few-shot relations [9]. It is difficult for current models to generate reliable embeddings for these **solitary entities** with unrelated neighbors.

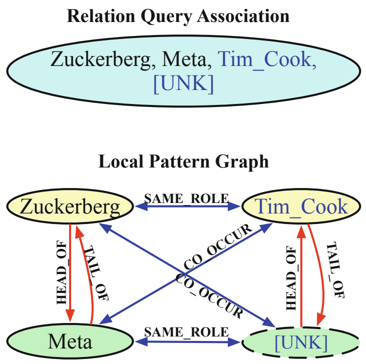
Intuitively, the correlations between entities in the same FSKGC task could provide semantic meanings towards current relation. For instance, the embedding of "Zuckerburg" may influence the representation of "Cook" in the context of relation "ceo_of" since they both act as the head entity of associated triples.

In light of this observation, in this paper, we propose a novel sub-graph based model YANA (**Y**ou **A**re **N**ot **A**lone) to tackle solitary entity issue by modeling hidden semantic correlations between entities in the same task. Specifically, as shown in Fig. 1(b), YANA collects entities involved in the FSKGC task in Fig. 1(a) to construct an entity set named *relation-query association*. Then we introduce four novel abstract relations in two categories to describe entity correlations in *relation-query association* and construct a multi-relation graph called *local pattern graph (LPG)*. With LPG, YANA devises a Highway R-GCN [13] to learn entity embeddings. A query-aware gating mechanism is proposed to integrate semantic meanings from LPG with intrinsic topology information from one-hop neighbors in KG to refine entity representations. Moreover, a transformer relation learner followed with an attentive prototypical network is applied to encode entity pairs and make predictions.

Our contributions could be summarized as follows: 1) This paper concerns the limitation of current FSKGC models on representing solitary entities and proposes a novel model YANA to address this problem by modeling correlations between entities shared the same few-shot relation task. 2) Experiments demonstrate that YANA achieves state-of-the-art performances on large-scale


Support Set
(Zuckerberg, CEO_OF, Meta)
Query
(Tim_Cook, CEO_OF, ?)
Candidates
Microsoft, Apple, Twitter, Avon, Intel, …
Few-Shot Knowledge Graph Completion
Ranking
Apple
~~Microsoft~~
~~Intel~~
~~Twitter~~
~~Avon~~

(a) Task Example



(b) Relation Query Association and Local Pattern Graph

Fig. 1. 1(a) gives an example of 1-shot knowledge graph completion task. 1(b) illustrates the corresponding *relation-query association* and the corresponding *local pattern graph* of 1(a). We devise four abstract relations to describe the correlations between entities and introduce a unified node `[UNK]` to represent the unknown tail entity of queries.

and sparse dataset Wiki and encouraging performances on the NELL. Extensive experiments also show the effectiveness of each module.

The rest of the paper is organized as follows. Section 2 introduces basic preliminaries. Section 3 gives the details of our proposed model. Section 4 presents the details of experiments. Finally, Sect. 5 concludes our work.

2 Preliminaries

Knowledge Graph. A knowledge graph $\mathcal{G}$ is formulated as $\mathcal{G} := \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, in which $\mathcal{E}$ is entity set, $\mathcal{R}$ is relation set and $\mathcal{T} := \{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$ is triple set. Background graph $\mathcal{BG}$ of $\mathcal{G}$ denotes a set of known triples available in training. Here we have $\mathcal{BG} \subset \mathcal{G}$.

***K*-Shot Knowledge Graph Completion.** Every few-shot relation $r \notin \mathcal{BG}$ is formulated as $\mathcal{D}_r = \{r, \mathcal{P}_r, \mathcal{C}_r\}$, in which $\mathcal{P}_r := \{(h, t)|(h, r, t) \in \mathcal{G}\}$ and $\mathcal{C}_r$ denote entity pairs and candidate entities of r respectively. A K-Shot Knowledge Graph Completion task refers to given a few-shot relation r with K entity pairs as support set $\mathcal{S}_r$ and a query entity pair $(h_q, ?)$, ranking golden tail entity t_q higher than other entities in $\mathcal{C}_r$.

Relation-Query Association. We formally define the *relation-query association* of a few-shot relation r with support set $\mathcal{S}_r$ and query $(h_q, ?)$ as an entity set $\mathcal{A}_r^q := \{h_1, h_2, \ldots, h_K, h_q\} \cup \{t_1, t_2, \ldots, t_K, t_{\texttt{[UNK]}}\}$. Here, we introduce a unified virtual node $t_{\texttt{[UNK]}}$ to denote the unknown tail entity in queries to ensure each query has a unique $\mathcal{A}_r^q$. For a K-shot task, we have $|\mathcal{A}_r^q| = 2K + 2$.

3 Methodology

The overall architecture of YANA is illustrated in Fig. 2. Given a query $(h_q, ?)$ along with support set $\mathcal{S}_r$ and candidates, YANA constructs a local pattern graph and applies Highway R-GCN to capture entities correlations. At the same time, a HGAT is proposed to learn from one-hop neighbors of each entity. A query-aware gating mechanism is proposed to merge information from two graph neutral networks. Latter, YANA learns relation representation and make predictions by a Transformer Relation Learner and Attentive Prototypical Network. Details of these modules will be presented in the following subsections.

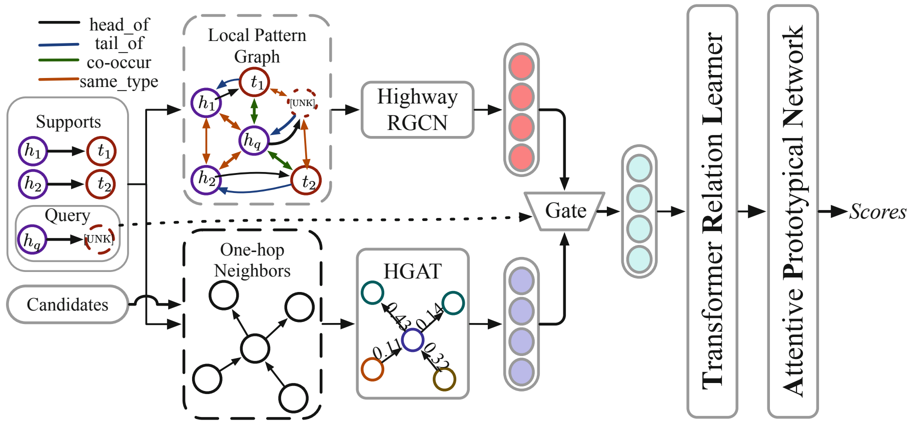


Fig. 2. Overall architecture of our model YANA.

3.1 Local Pattern Graph Construction

To model the relatedness between entities in $\mathcal{A}_r^q$, we consider two aspects of correlations and introduce four different abstract relations to convert $\mathcal{A}_r^q$ to a local pattern graph $\mathcal{G}_L$:

Inner-Pair Correlations: Two abstract relations `HEAD_OF` and `TAIL_OF` are introduced to describe the correlations of two entities within entity pair, *e.g.*, for (Zuckerburg, ceo_of, Meta), we have (Zuckerburg, `HEAD_OF`, Meta) and (Meta, `TAIL_OF`, Zuckerburg).

Across-Pair Correlations: We introduce two abstract relations `SAME_ROLE` and `CO_OCCUR` to reflect the relatedness of two entities from different triples. `SAME_ROLE` allows an entity to connect with the other K entities shared the same role (head-to-head and tail-to-tail). Meanwhile, with `CO_OCCUR`, an entity can aggregate message from the other K entities in different role (head-to-tail and tail-to-head). For entity pair (Zuckerburg, ceo_of, Meta) and (Hans, ceo_of, Version), we can have (Zuckerburg, `SAME_ROLE`, Hans) and (Hans, `CO_OCCUR`, Meta) in $\mathcal{G}_L$[1].

3.2 Message Passing over Local Pattern Graph

Relational Graph Convolution Network (R-GCN) [13] is a powerful architecture for producing latent representations of entities in multi-relational data. Thus, based on $\mathcal{G}_L$, we propose an L_R-layers Highway R-GCN network to update entity embeddings. Specifically, the $l+1$-th R-GCN layer computes the entity representation follows:

$$\mathbf{h}_i^{'l+1} = \sigma(\sum_{r=1}^{4} \sum_{j \in \mathcal{N}} (\mathbf{h}_i^l \mathbf{W}_r^l)), l = 0, 1, \ldots, L_R \tag{1}$$

$\mathbf{W}_r^l$ is relation specific matrix for the r-th relation. $\mathbf{h}_i^l$ is the l-th layer output of node i. σ indicates non-linear activation function and we use ReLU [8]. To overcome **over-smoothing problem** in multi-layer GNNs [22], we introduce highway mechanism [3,15] to enable cross layer interactions and obtain the $l+1$-layer output $\mathbf{h}_i^{l+1}$:

$$\begin{aligned} g^{l+1} &= \text{Sigmoid}(\mathbf{W}_h^{l+1} \mathbf{h}_i^{'l+1} + b_h^{l+1}) \\ \mathbf{h}_i^{l+1} &= g^{l+1} \odot \mathbf{h}_i^{'l+1} + (1 - g^{l+1}) \odot \mathbf{h}_i^l \end{aligned} \tag{2}$$

wherein $\mathbf{W}_h^{l+1} \in \mathbb{R}^{d_e}$ and $\mathbf{b}_h^{l+1} \in \mathbb{R}$ are layer-specific learnable parameters. $\odot$ is element wise dot product. We take the L-th layer output $\mathbf{h}_i^{L_R}$ as the task-specific embedding $\hat{\mathbf{e}}_i$ of the i-th entity e_i.

Note that, we assign entity i except $t_{\texttt{[UNK]}}$ in LPG with a d_e dimensional vector $\mathbf{x}_i \in \mathcal{R}^{d_e}$ obtained with embedding model ComplEx [17] and regard $\mathbf{x}_i$ as the input of Highway R-GCN, *i.e.*, $\mathbf{h}_i^0 = \mathbf{x}_i$. Besides, we represent $t_{\texttt{[UNK]}}$ with learnable vector $\mathbf{x}_{\texttt{[UNK]}} \in \mathcal{R}^{d_e}$ which initial values drawn from the normal distribution $\mathcal{N}(0, 0.5)$. To prevent noise from $\mathbf{x}_{\texttt{[UNK]}}$ to other entities, we introduce a punishment factor $\mathcal{L}_{\text{reg}} = \|\mathbf{x}_{\texttt{[UNK]}}\|_2$, in which $\|\cdot\|_2$ is the Euclidean norm.

3.3 HGAT for Encoding One-Hop Neighbors

Apart from entity correlations, direct neighbors in background graph are still a vital source to encode entities [9,14,23,24]. Hence, we devise a heterogeneous graph attention (HGAT) encoder generate the intrinsic embedding of entities in $\mathcal{A}_r^q \backslash \{t_{\texttt{[UNK]}}\}$ and $\mathcal{C}_r$ from its one-hop neighbors.

First we extract one-hop neighbors $\mathcal{N}_e = \{(r_i, t_i)\}$ starting with e from $\mathcal{BG}$. We calculate the impacts of (r_i, t_i) following:

[1] There will be eight edges for two pairs. We omit the other six samples for brevity.

$$\begin{aligned}
\mathbf{n}_i &= \tanh(\mathbf{W}_1(\mathbf{v}_{r_i} \| \mathbf{v}_{t_i})) \\
d_i &= \text{LeaklyReLU}(\mathbf{u}^T \mathbf{n}_i + b_1) \\
\alpha_i &= \frac{\exp(d_i)}{\sum_{(r_j, e_j) \in \mathcal{N}_e} \exp(d_j)}
\end{aligned} \tag{3}$$

in which $\mathbf{n}_i$ is the representation of (r_i, t_i), d_i and α_i are absolute and normalized attention value respectively. $\mathbf{v}_{r_i}$ and $\mathbf{v}_{t_i}$ are the initial embedding with dimension d_e of r_i and t_i. $\mathbf{W}_1 \in \mathbb{R}^{d_e \times 2d_e}$, $\mathbf{u} \in \mathbb{R}^{d_e}$ and $b_1 \in \mathbb{R}$ are learnable parameters. Finally, we have the intrinsic embedding $\bar{\mathbf{e}} = \sum_{i=1}^{|\mathcal{N}_e|} \alpha_i \mathbf{n}_i$.

3.4 Query-Aware Gating Mechanism

In order to automatically select relevant features and filter unrelated noises, we propose a query-aware gating mechanism to incorporate task-specific embedding $\hat{\mathbf{e}}$ with intrinsic embedding $\bar{\mathbf{e}}$ and generate reliable entity embedding:

$$\begin{aligned}
g_q &= \text{Sigmoid}(\mathbf{u}_{g_q}^T [\hat{\mathbf{h}}_q \odot \bar{\mathbf{h}}_q; \hat{\mathbf{t}}_q] + b_{g_q}) \\
\mathbf{h}_e &= \text{ReLU}(g_q \hat{\mathbf{e}} + (1 - g_q)\bar{\mathbf{e}})
\end{aligned} \tag{4}$$

wherein, $\mathbf{u}_{g_q} \in \mathbb{R}^{2d_e}$ and $b_{g_q} \in \mathbb{R}$ learnable parameters.

3.5 Transformer Relation Learner

With entity representations, we are going to encode entity pairs. A major challenge is to preserve relation patterns when extracting hidden semantic features. Inspired by the great success of transformer [14,18], we devise a transformer relation learner (TRL) to represent entity pairs. Taking an entity pair (h, r, t) of few-shot relation r as an example. We regard the pair as a sequence $X = \{h, r, t\}$. The input representation of each element follows:

$$\mathbf{z}_i^0 = \mathbf{x}_i^{\text{org}} + \mathbf{x}_i^{\text{pos}}, i = 1, 2, 3 \tag{5}$$

wherein, $\mathbf{x}_i^{\text{org}} \in \mathbb{R}^{d_e}$ is the origin embedding and $\mathbf{x}_i^{\text{pos}} \in \mathbb{R}^{d_e}$ is the learnable positional signals. With such positional signals, the multi-head self-attention module in transformer can distinguish the roles of input elements (*i.e.* head, tail entity and relation). Embeddings of h and t are from Eq. 4. As for $\mathbf{x}_r$, we use a unified random d_e dimensional vector as the initial embedding. Later, we pack the embeddings into matrix $\mathbf{Z}^0$ and feed into an L_T successive Transformer layers[2]:

$$\mathbf{Z}^l = \text{Transformer}(\mathbf{Z}^{l-1}), l = 1, 2, \ldots, L_T \tag{6}$$

[2] Due to paper length restrictions, we omit the details of transformer and refer readers to the origin paper [18].

Finally, we apply a Mean Pooling layer over the L_T-layer output $\mathbf{Z}^{L_T}$ to obtain pair representation:

$$\mathbf{z}_p = \text{MeanPooling}(\mathbf{z}_h^{L_T}, \mathbf{z}_r^{L_T}, \mathbf{z}_t^{L_T}) \tag{7}$$

3.6 Attentive Prototypical Network

Support instances may have various contributions when matching with different queries [4,14]. To allow YANA to focus more on relevant instances and filter noises during the prediction stage, we introduce an instance-level Attentive Prototypical Network (APN) to generate query-aware relation representation $\mathbf{r}_q$ and compute the matching score by automatically determine the importance of each instances towards the current query follows:

$$\begin{aligned} d(\mathbf{q}, \mathbf{s}_i) &= \|\mathbf{q} - \mathbf{s}_i\|_2 \\ \beta_i &= \frac{\exp(-d(\mathbf{q}, \mathbf{s}_i))}{\sum_{j=1}^{K} \exp(-d(\mathbf{q}, \mathbf{s}_i))} \\ \mathbf{r}_q &= \sum_{i=1}^{K} \beta_i \mathbf{s}_i \\ \text{score}(q, r) &= -d(\mathbf{q}, \mathbf{r}_q) \end{aligned} \tag{8}$$

wherein, $\|\cdot\|_2$ is the Euclidean norm. $\mathbf{q}$ and $\mathbf{s}_i$ are the representations of query and the i-th instance learned from TRL. β_i denotes the normalized semantic similarity between q and s_i.

3.7 Model Training

We follow previous FSKGC models' training regime [14,23,24] and conduct meta-training to optimize our model. In each training step, we randomly sample a relation r from $\mathcal{D}_{train}$ along with K instances from $\mathcal{P}_r$ as the support set $\mathcal{S}_r$. Positive queries $\mathcal{Q}_r^+$ are from $\mathcal{P}_r \backslash \mathcal{S}_r$. Then we obtain negative samples $\mathcal{Q}_r^-$ by polluting tail entity of each pair (h_l, t_l) in $\mathcal{Q}_r^+$ *s.t.* $t_n \in \mathcal{C}_r$ and $(h_l, t_n) \notin \mathcal{KG}$.

Finally, we apply margin-ranking loss along with $\mathcal{L}_{\text{reg}}$ to optimize our model:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\text{rank}} + \xi \mathcal{L}_{\text{reg}} \\ \mathcal{L}_{\text{rank}} &= \frac{1}{N} \sum_r \sum_{(h_l,t_l)\in\mathcal{Q}_r^+} \sum_{(h_l,t_n)\in\mathcal{Q}_r^-} \max(\gamma + \text{score}_{(h_l,t_n)} - \text{score}_{(h_l,t_l)}, 0) \end{aligned} \tag{9}$$

wherein hyper-parameter ξ is the trade-off factor between $\mathcal{L}_{\text{rank}}$ and $\mathcal{L}_{\text{reg}}$. γ is the margin distance.

Table 1. Statistics of datasets.

Dataset	Entities	Relations	Tuples	Avg-Deg	Solitary	#Train	#Valid	#Test
NELL	68545	291	181,109	5.284	85.18%	51	5	11
Wiki	4,838,244	539	5,859,240	2.422	99.20%	133	16	34

4 Experiments

4.1 Datasets and Baselines

Dataset: We conduct experiments on two FSKGC datasets, NELL and Wiki proposed by [23]. Relations with more than 50 but less than 500 instances are selected to construct train/valid/test set. Also, each relation has its own candidate set $\mathcal{C}_r$ constructed based on the entity type constraint. Table 1 lists statistics of these two datasets. Avg-Deg means the average degree of entities (including inverse relations). Solitary means the percentage of entities with less than five directed neighbors. #Train, #Valid and #Test indicate the number of tasks in train, valid and test set, respectively.

Baselines: We compare YANA with following baselines to measure the effectiveness of our model. *1) Sub-Graph Models*: **GMatching** [23]: The first model for one-shot relation learning. We extend GMatching to few-shot scenario by applying a max pooling (MaxP) or a mean pooling (MeanP) layer over K support instances to obtain 5-shot results. **FSRL** [24]: A metric-learning model with heterogeneous graph encoder and an LSTM auto-encoder for modeling instances interaction. **FAAN** [14]: A model with dynamic graph encoder discerning entity properties in different relations. **GANA** [9]: A model incorporating pre-train embeddings with sub-graph GNN to improve low-degree entity representations. *2) Full Graph models* We compare YANA with five full graph relation prediction models including **R-GCN** [13], **GNN** [11], **RA-GCN** [16], **I-GCN** [6] and **GNN-FSRP** [22]. Results of these models are taken from [22]. *3) Meta-Embedding Model*: **MetaR** [2]: A meta-learning model over TransE [1] for FSKGC. We directly report 5-shot results of MetaR from the origin paper.

4.2 Implementation Details

We initialize entities and relations in the background graph with ComplEx [17] embeddings which dimensionality is 100 for NELL and 50 for Wiki. We set the number of transformer layers to 3 and 4, and the number of heads to 4 and 8 for NELL and Wiki, respectively. The number of R-GCN layers is set to 2. Dropout with rate tuned in $\{0.1, 0.3, 0.5\}$ is applied to avoid over-fitting. Adam is used to optimize our model. We linearly increase the learning rate to $5e^{-5}$ for NELL and $6e^{-5}$ for Wiki at the very first 10k steps and decrease to 0 until the last epoch. We evaluate every 10k steps on the validation set and select models achieving the highest HITS@10 within 300k steps to make predictions on the test set. The margin γ is set to 5.0, and the trade-off ξ is 0.1. The sizes of $\mathcal{Q}_r^+$ and $\mathcal{Q}_r^-$ are both set to 128. We fix the maximum number of neighbors M to 30 for all sub-graph models to obtain results in our environment for a fair comparison.

Metrics: We use MRR and HITS@N to measure the performance of all methods. MRR is the mean reciprocal rank, and HITS@N is the proportion of correct entities ranked in the top N, with $N = 1, 5, 10$. For both HITS@N and MRR, a higher score means better performance. We perform a 5-shot KGC task for all models, *i.e.*, $K = 5$.

4.3 Main Results

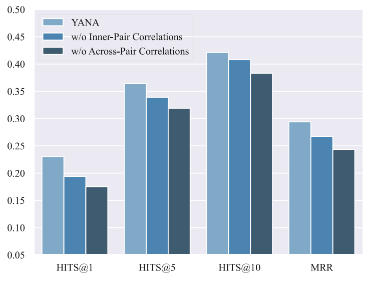
Table 2. Experiment results. Best results are in boldface. Underline indicates the second-best results.

Model	NELL				Wiki			
	HITS@1	HITS@5	HITS@10	MRR	HITS@1	HITS@5	HITS@10	MRR
GMatching (MaxP)	.113	.223	.286	.174	.197	.331	.427	.273
GMatching (MeanP)	.119	.255	.348	.188	.213	.335	.391	.284
FSRL	.169	.284	.352	.230	.167	.295	.354	.253
FAAN	.188	.361	.437	.271	.270	.391	.448	.339
GANA	.194	.395	.482	.291	.289	.392	.436	.347
MetaR (BG:In-Train)	.168	.350	.437	.261	.178	.264	.302	.221
MetaR (BG:Pre-Train)	.141	.280	.355	.209	.270	.385	.418	.323
R-GCN	.139	.346	.427	.270	.141	.310	.351	.233
GNN	.140	.351	.451	.273	.143	.316	.365	.235
RA-GCN	.144	.358	.442	.280	.146	.321	.364	.241
I-GCN	.142	.353	.436	.275	.144	.317	.358	.238
GNN-FSRP	.218	**.442**	**.518**	**.336**	.161	.338	.420	.252
YANA (Ours)	**.230**	.364	.421	.294	**.327**	**.442**	**.523**	**.380**

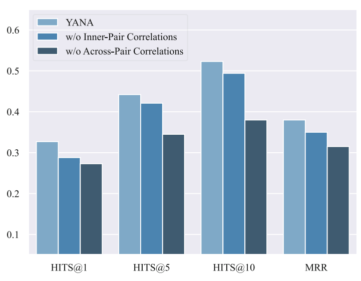
Table 2 demonstrates that YANA achieves consistent improvements compared with baseline models. To be concrete, 1) YANA achieves the state-of-the-art performance on the Wiki dataset. Note that Wiki is a large-scale and sparse dataset, where nearly 99% of entities only have less than 5 neighbors, leading to sub-optimal performance for existing models. Nevertheless, our proposed model YANA gains 3.1%, 5.0%, 7.5% and 3.3% performance improvements in HITS@1, HITS@5, HIST@10 and MRR, respectively. 2) Unlike Wiki, NELL can provide sufficient and related neighborhood information for FSKGC. The results show that full-graph models leveraging signals from multi-hop neighbors are generally more expressive than sub-graph models. Even so, YANA achieves the best HITS@1 performance, which means that our model can make more precise predictions than others.

4.4 Ablation Study

LPG Variants. The major contribution of our work is that we construct local pattern graphs by proposing two categories of abstract relations for few-shot



(a) NELL Dataset



(b) Wiki Dataset

Fig. 3. Ablation study results of abstract relations. *w/o Inner-Pair Correlations* denotes LPG without abstract relations `HEAD_OF` and `TAIL_OF`. *w/o Across-Pair Correlations* means removing `CO_OCCUR` and `SAME_ROLE` in LPG.

relation tasks. To examine the legitimacy of these abstract relations, we conduct experiments on NELL and Wiki with two different LPG constructed by removing two categories of relations, respectively. Overall results are shown in Fig. 3. From these results, we observe that on both datasets, the removal of *Inner-Pair Correlations* or *Across-Pair Correlations* leads to the performance decrement, which means that modeling the correlations between entities is beneficial to final predictions. In terms of different datasets, these entity correlations have more significant impacts on Wiki than that on NELL. It is reasonable because Wiki is far more sparse than NELL and therefore more dependent on dynamic neighbors from LPG to encode entities.

Module Variants. We further conduct ablation studies on NELL to investigate the effectiveness of each module of YANA. *w/o LPG* generates entity representation only by aggregating its one-hop neighbor embeddings with our devised HGAT module. *w/o HGAT* omits the HGAT module and allows the query-aware gating mechanism to combine the outputs of Highway R-GCN and pre-trained embeddings to represent entities. *w/o GNNs* removes the LPG and HGAT at the same time and represents entities with pre-trained embeddings learned with ComplEx [17] to examine the effectiveness of graph encoders. *w/o TRL* removes the transformer relation learner (TRL) and follows previous works [9,23,24] to represent (h, t) with the concatenation of head and tail embeddings, *i.e.*, $\mathbf{z}_p = [\mathbf{h}; \mathbf{t}]$. *w/o APN* replaces the attentive prototypical network (APN) with an LSTM matching network [19] to compute scores of queries.

Overall results are listed in Table 3. We can observe that: **1) Impacts of Entity Encoder**: The results of *w/o HGAT* prove the ability of our Highway R-GCN to capture semantic meanings from LPG for FSKGC problem. Furthermore, the results of *w/o LPG* reveal the effectiveness of our sub-graph encoding module. Besides, model performances drop sharply without graph encoders. **2) Impacts of TRL and APN**: We devise a transformer relation learner (TRL) to encode entity pairs and apply an attentive prototypical network (APN) to

Table 3. Results of model variants on NELL dataset with 5-shot. Best results are in boldface.

Models	HITS@1	HITS@5	HITS@10	MRR
YANA	**.230**	**.364**	**.421**	**.294**
w/o HGAT	.162	.315	.398	.235
w/o LPG	.181	.332	.400	.252
w/o GNNs	.097	.217	.268	.156
w/o TRL	.156	.272	.334	.221
w/o APN	.143	.309	.402	.229

compute scores for queries. Results of *w/o TRL* and *w/o APN* prove the validness of these modules.

5 Conclusion

This paper concentrated on the solitary entity issue in the few-shot knowledge graph completion problem and proposed our model YANA. Different from previous approaches relying on learning knowledge graph structure to represent entities, YANA introduces four novel abstract relations to exploit hidden correlations between entities within a few-shot relation task. With a gating mechanism, YANA can effectively combine neighbor signals from knowledge graph and task-specific features to learn more reliable embeddings for solitary entities. Experiments on two benchmark datasets NELL and Wiki demonstrated the effectiveness of YANA. Extensive ablation studies validated the effects of each module in YANA and proved the importance of the four abstract relations. Our future work may consider learning the contribution of different entities within a task and explore the interaction of task structure and the background graph.

Acknowledgements. This work is supported by Beijing Nova Program of Science and Technology (Grant No. Z191100001119031), National Natural Science Foundation of China (Grant No. U21A20468), Guangxi Key Laboratory of Cryptography and Information Security (Grant No. GCIS202111), The Open Program of Zhejiang Lab (Grant No. 2019KE0AB03), and Zhejiang Lab (Grant No. 2021PD0AB02). Yi Liang is supported by BUPT Excellent Ph.D. Students Foundation under grant CX2019136. Shuai Zhao is the corresponding author.

References

1. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS, pp. 2787–2795 (2013)
2. Chen, M., Zhang, W., Zhang, W., Chen, Q., Chen, H.: Meta relational learning for few-shot link prediction in knowledge graphs. In: EMNLP, pp. 4216–4225. Association for Computational Linguistics (2019)

3. Dai, D., Zheng, H., Sui, Z., Chang, B.: Incorporating connections beyond knowledge embeddings: a plug-and-play module to enhance commonsense reasoning in machine reading comprehension. arXiv:2103.14443 (2021)
4. Gao, T., Han, X., Liu, Z., Sun, M.: Hybrid attention-based prototypical networks for noisy few-shot relation classification. In: AAAI (2019)
5. Hu, F., Lakdawala, S., Hao, Q., Qiu, M.: Low-power, intelligent sensor hardware interface for medical data preprocessing. IEEE Trans. Inf. Technol. Biomed. **13**, 656–663 (2009)
6. Ioannidis, V.N., Zheng, D., Karypis, G.: Few-shot link prediction via graph neural networks for Covid-19 drug-repurposing. arXiv:2007.10261 (2020)
7. Li, Y., Song, Y., Jia, L., Gao, S., Li, Q., Qiu, M.: Intelligent fault diagnosis by fusing domain adversarial training and maximum mean discrepancy via ensemble learning. IEEE Trans. Industr. Inf. **17**, 2833–2841 (2021)
8. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: ICML, pp. 807–814. Omnipress (2010)
9. Niu, G., et al.: Relational learning with gated and attentive neighbor aggregator for few-shot knowledge graph completion. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (2021)
10. Qiu, H., Zheng, Q., Msahli, M., Memmi, G., Qiu, M., Lu, J.: Topological graph convolutional network-based urban traffic flow and density prediction. IEEE Trans. Intell. Transp. Syst. **22**, 4560–4569 (2021)
11. Satorras, V.G., Bruna, J.: Few-shot learning with graph neural networks. arXiv:1711.04043 (2018)
12. Saxena, A., Tripathi, A., Talukdar, P.P.: Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In: ACL (2020)
13. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 593–607. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_38
14. Sheng, J., et al.: Adaptive attentional network for few-shot knowledge graph completion. In: EMNLP, pp. 1681–1691. Association for Computational Linguistics (2020)
15. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. arXiv:1505.00387 (2015)
16. Tian, A., Zhang, C., Rang, M., Yang, X., Zhan, Z.: RA-GCN: relational aggregation graph convolutional network for knowledge graph completion. In: Proceedings of the 2020 12th International Conference on Machine Learning and Computing (2020)
17. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: ICML, vol. 48, pp. 2071–2080. JMLR.org (2016)
18. Vaswani, A., et al.: Attention is all you need. In: NIPS, pp. 5998–6008 (2017)
19. Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. In: NIPS, pp. 3630–3638 (2016)
20. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (2014)
21. Wang, X., He, X., Cao, Y., Liu, M., Chua, T.S.: KGAT: knowledge graph attention network for recommendation. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2019)
22. Wang, Y., Zhang, H.: Introducing graph neural networks for few-shot relation prediction in knowledge graph completion task. In: KSEM (2021)

23. Xiong, W., Yu, M., Chang, S., Guo, X., Wang, W.Y.: One-shot relational learning for knowledge graphs. In: EMNLP, pp. 1980–1990. Association for Computational Linguistics, Brussels, Belgium, October–November 2018
24. Zhang, C., Yao, H., Huang, C., Jiang, M., Li, Z., Chawla, N.: Few-shot knowledge graph completion. In: AAAI, vol. 34, pp. 3041–3048, April 2020