**Abstract**

Despite becoming a prevailing paradigm for organizing knowledge, most knowledge graphs (KGs) suffer from the low-resource issue due to the deficiency of data sources. The enrichment of KGs by automatic knowledge graph completion is impeded by the intrinsic long-tail property of KGs. In spite of their prosperity, existing few-shot learning-based models have difficulty alleviating the impact of the long-tail issue on low-resource KGs because of the lack of training tasks. To tackle the challenging long-tail issue on low-resource KG completion, in this paper, we propose a novel few-shot low-resource knowledge graph completion framework, which is composed of three components, i.e., few-shot learner, task generator, and task selector. The key idea is to generate and then select the beneficial few-shot tasks that complement the current tasks and enable the optimization of the few-shot learner using the selected few-shot tasks. Extensive experiments conducted on several real-world knowledge graphs validate the effectiveness of our proposed method.

1 Introduction

Recent years have witnessed a rapidly increasing amount of research attention and industry demand on knowledge graphs (KGs), which organize knowledge in the form of triples and have been playing a crucial role in many knowledge-intensive applications, such as question answering (Zhang et al., 2020c; Kazemi and Poole, 2018; Bordes et al., 2013; Tang et al., 2022) has empowered the discovery of more triples to enrich KGs, it is far from perfect to enrich low-resource KGs due to the difficulty in the discovery that stems from the relations associated with missing triples and varies with the change of relation frequency. Since most KGs generally follow long-tail distribution (Xiong et al., 2018; Zhang et al., 2020a; Nguyen et al., 2018), where a large fraction of relations have only a few triples, the existence of rare relations on low-resource KGs leads to dramatic performance degradation on the discovery of missing triples and impede the development of an effective completion model. A line of efforts (Xiong et al., 2018; Zhang et al., 2020a; Chen et al., 2019; Lv et al., 2019) attempt to improve the capability of inference on rare relations by formulating the KG completion problem into a few-shot learning framework and exploiting inference models for frequent relations to facilitate inference on rare relations. Such kind of methods require abundant training tasks to ease the effect of memorization and improve the generalization (Rajendran et al., 2020). However, low-resource KGs do not always have adequate training tasks for mimicking the few-shot learning scenarios of rare relations. For example, the Greek KG (Chen et al., 2020) used in experiments only contains 21 training tasks, as many relations are unknown or cannot offer triples for

---

1Equal contribution.

---

training task construction. Therefore, the few-shot KG completion models tend to overfit on the scarce tasks and engender unsatisfactory results.

Motivated by the above-discussed limitation on low-resource KGs, we propose to generate new few-shot tasks to augment current tasks for easing the impact of memorization and improving the generalization. Although straightforward, the materialization of this idea is non-trivial due to two crucial challenges. The first challenge (C1) is to design an effective strategy to generate new few-shot tasks. A few-shot task is formed as a set of pairs corresponding to the same relation, each of which includes two entities, i.e., head entity and tail entity. On the one hand, the discrete nature of tasks to generate hinders the harness of prevailing generative models. On the other hand, the generation of few-shot tasks with novel relations is arduous due to the lack of related data samples. If possible to generate, the second challenge (C2) is to ensure the authenticity of generated few-shot tasks. The generated tasks ineluctably consist of noisy tasks which cannot represent the designated relation and would be detrimental to few-shot learning. Thus, how to select beneficial tasks is an essential problem.

In this paper, we propose a novel few-shot low-resource KG completion framework with reinforced task generation, named FLow-KGC, to promote the KG completion on rare relations in low-resource KGs. Specifically, we formulate KG completion into a few-shot learning framework and train a simple yet effective few-shot learner to learn representations for rare relations with a small support set. The learned representation can be further used for inference. To address challenge C1, we represent few-shot tasks in a latent continuous space and then build a task generator in this latent space, rather than generating original discrete structure samples. Then the generated tasks can be utilized to update the few-shot learner. To select beneficial synthetic few-shot tasks and tackle challenge (C2), we design an adaptive task selector, which makes decisions on keeping or discarding generated tasks and receives feedback from the few-shot learner and conducts optimization using reinforcement learning. The task generator and task selector collaboratively complete the process of task generation. These three components, i.e., few-shot learner, task generator, and task selector, constitute our proposed method FLow-KGC and can be updated in an alternative optimization way.

Overall, our contributions in this work include: (1) We study the crucial few-shot KG completion on low-resource KGs and propose a novel model called FLow-KGC to mitigate the impact of the long-tail problem on completion tasks. (2) We design a task generator to create synthetic few-shot tasks and a task selector to achieve adaptive beneficial task selection, in order to improve the generalization of the few-shot learner. (3) We perform extensive experiments on several real-world low-resource KGs. The experimental results show the superior performance of FLow-KGC over the state-of-the-art with significant improvement in few-shot low-resource KG completion.

2 Related Work

2.1 Knowledge Graph Completion

Early efforts address KGC by designing algorithms based on rule learning (Galárraga et al., 2015) to discover logical rules from KGs for facilitating inductive link prediction. Nowadays, the prevailing learning paradigm for KG completion tasks (Zhang et al., 2020c) is to learn the distributed representation of entities and relations in KGs. These methods roughly fall into three categories (Bonner et al., 2021): 1) Tensor decomposition methods, such as SimplE (Kazemi and Poole, 2018), RESCAL (Nickel et al., 2012), and ComplEx (Trouillon et al., 2017). 2) Geometric methods, such as TransE (Bordes et al., 2013), RotatE (Sun et al., 2018b), and CrossE (Zhang et al., 2019). 3) Deep learning methods (Nguyen et al., 2018; Dettmers et al., 2018; Vashisht et al., 2020). However, these methods fail to handle the long-tail problem of KG completion and suffer from performance degradation of prediction on rare relations. A recent approach (Zhang et al., 2020b) attempts to alleviate this issue, but it requires a relatively large set of triples for rare relations and is incapable of handling the few-shot scenario. Recently, a few works (Chen et al., 2020; Zhou et al., 2021) design strategies to assist the KG completion by leveraging complementary knowledge from other related KGs in different languages (Chen et al., 2020; Conneau et al., 2020; Zhou et al., 2021). Although effective, they only rely on cross-lingual links (Chen et al., 2017; Sun et al., 2018a; Pei et al., 2019a; Cao et al., 2019; Pei et al., 2019b, 2020) and cannot handle new relations or rare relations with few triples.
2.2 Few-shot Knowledge Graph Completion

With the goal to overcome the shortcoming of canonical KG completion methods (Kazemi and Poole, 2018; Bordes et al., 2013; Nguyen et al., 2018) on inferring the missing triples associated with rare relations, several few-shot KG completion (FSKG) algorithms have been developed for improving completion performance on rare relations. Recent attempts (Xiong et al., 2018; Zhang et al., 2020a; Chen et al., 2019; Lv et al., 2019) on few-shot KG completion formulate the problem into a few-shot learning framework and propose metric-based approaches (Xiong et al., 2018; Zhang et al., 2020a; Sheng et al., 2020; Niu et al., 2021) and meta-learning-based approaches (Chen et al., 2019; Lv et al., 2019). Yet these few-shot KG completion models require plenty of training tasks to train a few-shot learner and cannot generalize well to rare relations in low-resource KGs due to the deficiency of relations and corresponding tasks.

3 Problem Formulation

A knowledge graph $G$ can be denoted as $G = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ refers to the set of entities and $\mathcal{R}$ denotes the set of relations, and $\mathcal{T}$ is represented as a set of triples $\{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, each of which includes a head entity $h$, a relation $r$, and a tail entity $t$. The KG completion problem is to infer the most plausible missing triples from the candidate set $\{(h, r, t)|h \in \mathcal{E} \land (h, r, t) \notin \mathcal{T}\}$ for each incomplete triple $(h, r, ?)$ (or inferring from $\{(h, r, t)|h \in \mathcal{E} \land (h, r, t) \notin \mathcal{T}\}$ for $(?, r, t)$).

In our few-shot low-resource KGC problem, the majority of relations have a few training triples. Formally, for a rare relation $r_i$, there are a support set $S_i = \{(h, t)|(h, r_i, t) \in \mathcal{T}\}$ including only $|S|$ training triples. The task is to predict the tail entity for a query triple $(h, r_i, ?)$, i.e., ranking all tail entity candidates such that the true tail entity $t$ is ranked higher than other candidates in $C_{h, r_i}$, which is defined as $C_{h, r_i} = \{t|t \in \mathcal{E} \land (h, r_i, t) \notin \mathcal{T}\}$. Following the notations in meta-learning, a few-shot task with $K = |S|$ triples is called K-shot KG completion. Since this $K$ for a rare relation $r_i$ is often small (e.g., less than 10), the learning problem is organized by working on meta-tasks $T_i = \{S_i, Q_i\}$, where a query set $Q_i = \{(h, t, C_{h, r_i})|(h, r_i, t) \in \mathcal{T}\}$.

In the meta-training stage, there are a set of few-shot tasks $T_{train} = \{T_i\}_{j=1}^{M}$, where each task $T_i$ represents a relation $r_i \in \mathcal{R}_{train}$. A few-shot learning model can be optimized using meta-training tasks following the meta-learning principle (Finn et al., 2017). In the meta-testing stage, with a set of testing tasks $T_{test} = \{T_j\}_{j=1}^{N}$, the model should make inferences on new relation $r_j \in \mathcal{R}_{test}$ corresponding to task $T_j$. Note that all relations that appear in the meta-testing tasks are unseen during meta-training, i.e., $\mathcal{R}_{train} \cap \mathcal{R}_{test} = \emptyset$.

Low-resource KGs suffer from severer incompleteness issue and consist of fewer relations. Due to the long-tail distribution of these relations, there are only a small number of relations with a high frequency qualified to offer triples for task construction, leading to insufficient $T_{train}$. Our proposed solution enriches $T_{train}$ by generating and selecting beneficial new tasks for virtual relations which are undiscovered and not present in KGs.

4 The Proposed Method

In this section, we introduce the proposed method FLow-KGC, describe each component in detail, and elaborate on the optimization and inference, following the overview shown in Figure 1.

4.1 Few-shot Learner

A few-shot learner is a basic yet essential component in few-shot learning for generalizing an inference model over rare relations with only a few triples. In particular, the few-shot learner should be able to learn accurate representation for rare relations using a small support set, then the learned representation will be applied for subsequent inference. We adopt a simple few-shot learner based on meta-learning (Chen et al., 2019) to learn representation for relations. Specifically, given a meta-training task $T_i$ with support set $S_i$ and its corresponding relation $r_i \in \mathcal{R}_{train}$, a few-shot learner $\text{FS}(_\cdot_)$ aims to generate a representation for $r_i$ by taking entities in associated triples as input. The representation of $r_i$ can be obtained by the following:

$$r_{T_i} = \frac{\sum_{|S_i|} \text{FS}(e_{h_k} \oplus e_{t_k})}{|S_i|},$$ (1)

where $(h_k, t_k)$ denotes the $k$-th pair in $S_i$, $e_{h_k}$ and $e_{t_k}$ are embeddings for entity $h_k$ and $t_k$, and $r_{T_i}$ refers to the learned representation of relation $r_i$. $\oplus$ denotes the concatenation of the embedding $x$ and $y$. The few-shot learner $\text{FS}(_\cdot_)$ is implemented by an $L$-layers fully connected neural network.

After obtaining the representation $r_{T_i}$ of relation $r_i$, we measure if $r_{T_i}$ can well represent relation
$r_i$ by a score function $f^s(\cdot)$ and define the meta-training loss on support set $S_i$ as follow:

$$L_{S_i} = \sum_{k=1}^{n} \left[ \gamma + f^s(h_k, r_i, t_k) - f^s(h_k, r_i, t_n) \right]_{+},$$

(2)

where $f^s(\cdot)$ is based on TransE (Bordes et al., 2013) algorithm and measures the plausibility of triple $(h_k, r_i, t_k)$ by $f^s(h_k, r_i, t_k) = ||e_{h_k} + r_{T_i} - e_{t_k}||$. $f^s(h_k, r_i, t_n)$ is the score for negative sample $(h_k, t_n)$ sampled from $\{(h_k, t') | t' \in E \cup \{h_k, r_i, t' \notin T_P \} \}$. And $\gamma$ is a predefined margin parameter. Then a fast update on $r_{T_i}$ can be conducted to obtain a more accurate representation of relations and works as follows:

$$\hat{r}_{T_i} = r_{T_i} - \beta \nabla_{r_{T_i}} L_{S_i},$$

(3)

where $\beta$ is the step size of the gradient updates.

Next, the updated relation representation $\hat{r}_{T_i}$ is exploited to measure the plausibility of triples in the query set $Q_i$ with a score function $f^q(\cdot)$ defined by $f^q(h_z, r_i, t_z) = ||e_{h_z} + \hat{r}_{T_i} - e_{t_z}||$, where $(h_z, t_z)$ denotes the z-th pair in $Q_i$. Then the loss function for updating few-shot learner $L_{Q_i}$ on the query set $Q_i$ is defined as follows:

$$L_{Q_i} = \sum_{z=1}^{n} \left[ \gamma + f^q(h_z, r_i, t_z) - f^q(h_z, r_i, t_c) \right]_{+},$$

(4)

where $t_c \in C_{h_z, r_i}$ is a tail entity candidate.

Lastly, the few-shot learner is optimized on all meta-training tasks as follows:

$$L_{FS} = \sum_{i=1}^{n} L_{Q_i}.$$  

(5)

### 4.2 Task Generator

The few-shot learner on low-resource KGs suffering from the deficiency of training tasks tends to overfit and memorize the given tasks and lacks generalization ability. We thus target to generate few-shot tasks to complement the meta-training set. In particular, We design a task generator in a task representation space, rather than generating new tasks in the original discrete format.

#### 4.2.1 Task Representation

We build the representation of a task based on the entities involved in the task. Specifically, for a task $T_i$ with a support set $S_i$ and a query set $Q_i$, we denote the representations of $S_i$ and $Q_i$ as $S_i \in \mathbb{R}^{K \times d}$ and $Q_i \in \mathbb{R}^{Z \times d}$, which are obtained by:

$$S_i = SH_i \oplus ST_i,$$

$$Q_i = QH_i \oplus QT_i,$$

(6)

where $SH_i \in \mathbb{R}^{K \times d}$ denotes the representation matrix of head entities in support set $S_i$ and is obtained by $SH_i = e_{h_1} \oplus \ldots \oplus e_{h_K}$. $d$ is the dimension of embeddings. Similarly, $ST_i \in \mathbb{R}^{d \times d}$ refers to the representation matrix of tail entities in support set $S_i$ and is obtained by $ST_i = e_{t_1} \oplus \ldots \oplus e_{t_K}$. $K$ denotes the size of the support set. The representation of $Q_i$ can be acquired in the same way. $QH_i$
... and QT$^i$ represent the head entities and tail entities in Q$^i$, respectively. Z denotes the size of the query set. With obtained S$^i$ and Q$^i$, a task $T^i$ can be represented by $T^i \in \mathbb{R}^{(K+Z) \times 2d}$ as follow:

$$T^i = S^i \oplus Q^i.$$  \hfill (7)

### 4.2.2 Conditional Variational Autoencoder

With learned task representations and the aim to generate meta-tasks, we expect to estimate the underlying posterior distribution $p(\cdot | \mathbf{T}, r)$ of meta-tasks with given relations for task generation by sampling. Due to the intractability of this posterior distribution, we employ a conditional variational autoencoder (CVAE) (Sohn \textit{et al.}, 2015) to circumvent the direct posterior estimation and generate meta-tasks with relations as a conditional variable.

**Encoder.** The encoder takes the representation $\mathbf{T}$ of a task $\mathcal{T}$ with its corresponding relation $r$ as input data and aims to construct a latent distribution $q_\phi(z | \mathbf{T}, r)$ to describe the distribution of meta tasks related to relation $r$, which is represented by the mean $\mu$ and standard deviation $\sigma$, and $z$ is a sample drawn from the distribution. $\mu$ and $\sigma$ are learned by two separate linear functions $f_\mu$ and $f_\sigma$ as follows:

$$\mu = f_\mu(\text{MLP}_{\text{enc}}(\text{Concat}(\mathbf{T}, r)))$$  \hfill (8)

$$\sigma = f_\sigma(\text{MLP}_{\text{enc}}(\text{Concat}(\mathbf{T}, r))),$$  \hfill (9)

where $f_\mu(x) = W_\mu x + b_\mu$ and $f_\sigma(x) = W_\sigma x + b_\sigma$, MLP$_{\text{enc}}(\cdot)$ is an L-layer fully connected neural network, Concat(,) is a concatenation operator, which first squeezes $\mathbf{T}$ to a vector $\mathbf{T}' \in \mathbb{R}^{(K+Z) d}$ then concatenates $\mathbf{T}'$ and a one-hot vector $\mathbf{o}_r$ (representing the ID of relation $r$). Here we assume there are $V$ virtual relations representing undiscovered but existent relations in the real world. These virtual relations will have generated meta-tasks. A relation $r$ then can be represented as a one-hot vector $\mathbf{o}_r \in \mathbb{R}^{(|R_{\text{train}}|+V)}$ to denote its ID.

**Decoder.** The decoder is to reconstruct the input representation $\mathbf{T}$ with the learned $\mu$ and $\sigma$, which defines the latent distribution of the given relation $r$. Given $\mu$ and $\sigma$, we can sample a latent variable $z$ from the constructed space $\mathcal{N}(\mu, \sigma)$. Yet the sampling operator makes the model non-differentiable and unable to calculate the gradient. Therefore, we adopt the reparameterization trick (Kingma and Welling, 2014) to solve this issue, which works by $z = \mu + \sigma \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ and $\odot$ denotes the element-wise multiplication. The reconstruction should involve the relation information because the reconstructed task representation is a relation-specific representation. With the sampled $z$, the decoder can be denoted as $p_\theta(\mathbf{T} | z, r)$ and the reconstruction process is defined as follow:

$$\mathbf{T}' = \text{MLP}_{\text{dec}}(z \oplus \mathbf{o}_r),$$  \hfill (10)

where MLP$_{\text{dec}}(\cdot)$ is an L-layer fully connected neural network and $\mathbf{T}' \in \mathbb{R}^{(K+Z) d}$ denotes the reconstructed representation of task $\mathcal{T}$.

With the encoder and decoder, we can define the reconstruction loss to minimize the variational lower bound by:

$$\mathcal{L}_{\text{CVAE}} = -\text{KL}(q_\phi(z | \mathbf{T}, r) || p_\theta(z | r)) + \mathbb{E}[\log p_\theta(\mathbf{T} | z, r)],$$

where the first term is a KL-divergence loss which can be rewritten as $-\text{KL}(q_\phi(z | \mathbf{T}, r) || p_\theta(z | r)) = -\frac{1}{2} \sum_{i=1}^M (-\sigma_i + \exp(\sigma_i) + \mu_i^2 - 1)$ by letting the prior distribution $p_\theta(z | r)$ be $\mathcal{N}(0, \mathbf{I})$, and the second term is the reconstruction loss defined as $\mathbb{E}[\log p_\theta(\mathbf{T} | z, r)] = \sum_{i=1}^M ||\mathbf{T}' - \mathbf{T}||^2_2$.

### 4.2.3 Task Generation

With the designed CVAE, we can generate meta-tasks that are related to a relation. We generate tasks for virtual relations that refer to undiscovered relations, which would improve the diversity of tasks and enhance the generalization of the few-shot learner. Note that we do not generate tasks for known relations, because the key issue in low-resource KGC is the absence of representative tasks for a large portion of unknown relations, and those known relations without sufficient triples for meta-task construction. Specifically, given a virtual relation $r_v$, its one-hot vector representation can be obtained, which is denoted as $\mathbf{o}_{r_v}$. Then we can sample a latent variable $z_v$ from the prior distribution $p_\theta(z | r) \sim \mathcal{N}(0, \mathbf{I})$. With the sampled $z_v$ and one-hot vector $\mathbf{o}_{r_v}$, according to Eq.(10), the well-trained decoder can be employed to generate the representation $\mathbf{T}_v'$ for the task corresponding to relation $r_v$. Then the vector $\mathbf{T}_v'$ can be unsqueezed as $\mathbf{T}_v' \in \mathbb{R}^{(K+Z) \times 2d}$, which is further decomposed as four matrices $\mathbf{S}H_v \in \mathbb{R}^{K \times d}$, $\mathbf{S}T_v \in \mathbb{R}^{K \times d}$, $\mathbf{Q}H_v \in \mathbb{R}^{Z \times d}$, and $\mathbf{Q}T_v \in \mathbb{R}^{Z \times d}$ standing for the generated representations of head and tail entities in a support set and a query set. And the four matrices form a task $\mathbf{T}_v = \{\mathbf{S}H_v, \mathbf{S}T_v, \mathbf{Q}H_v, \mathbf{Q}T_v\}$. The generation process can be repeated multiple times to obtain a set of tasks for a relation, which will be further exploited in the meta-training.
4.3 Task Selector

The generated meta-tasks are leveraged in the meta-training stage with the desire to ease overfitting and improve generalization. However, these tasks inevitably comprise noisy meta-tasks, which cannot represent the corresponding relation well, and would mislead learning of the few-shot learner. To alleviate the adverse impact of noisy tasks, we design a task selector to select beneficial meta-tasks to promote the training of the few-shot learner. Specifically, given a set of generated meta-tasks \( T_{GEN} = \{ \tilde{T}_1, ..., \tilde{T}_N \} \), where \( N \) denotes the number of tasks, the task selector learns to generate a score to measure the authenticity of a task and decide the beneficial tasks used for meta-training. For a task \( \tilde{T}_i \), the score is calculated as follows:

\[
s(\tilde{T}_i) = \frac{1}{K} \sum_{j=1}^{K} \text{MLP}_r(s(e_{h_j} + r_{\tilde{T}_i} - e_{t_j})) \quad (12)
\]

where \( \text{MLP}_r(s(\cdot)) \) is an \( L \)-layer fully connected neural network with an output dimension as 1 and parameterized by \( \psi \). Here \( r_{\tilde{T}_i} \) is obtained using \( S\tilde{H}_i \) and \( \tilde{S}T_i \) following Eq.(1). \( e_{h_j} \) denotes the embedding of \( j \)-th head entity in \( S\tilde{H}_i \) and \( e_{t_j} \) is the embedding of \( j \)-th tail entity in \( \tilde{S}T_i \). For better exploration, the task selector adopts a stochastic policy \( \pi(t) \) to make the choice on meta-tasks under the categorical distribution \( p = \text{Cat}(\cdot | T_{GEN}) \). The probability \( p(\tilde{T}_i) \) of selecting task \( \tilde{T}_i \) is calculated by \( p(\tilde{T}_i) = s(\tilde{T}_i) / \sum_{i=1}^{N} s(\tilde{T}_i) \).

With the obtained sampling probabilities, we can select \( B \) meta-tasks from \( T_{GEN} \). Then meta-training of few-shot learner \( \text{FS}(\cdot) \) can be conducted following Eq.(1) to Eq.(4) using the selected \( B \) meta-tasks. Hence the few-shot learner is optimized on \( B \) generated meta-tasks as follows:

\[
\mathcal{L}_{\text{FS}} = \sum_{i=1}^{B} \mathcal{L}_{Q_i} \quad (13)
\]

where \( \mathcal{L}_{Q_i} \) denotes the loss on generated query set.

To motivate the task selector towards the selection of precise meta-tasks, we evaluate the task selector with feedback signals from the few-shot learner, which reflect the effectiveness of meta-tasks for training the few-shot learner. Specifically, we denote \( \Phi_{OLD} \) as the parameters of the current few-shot learner and \( \Phi_{NEW} \) as the parameters after the few-shot learner undertakes a temporary update with Eq.(13). Leveraging \( \text{FS}(\cdot | \Phi_{OLD}) \) and \( \text{FS}(\cdot | \Phi_{NEW}) \) and following Eq.(1) to Eq.(4), the reward \( R \) is defined as follows:

\[
R = \tanh\left( \frac{1}{|T_{val}|} \sum_{i=1}^{T_{val}} \mathcal{L}_{Q_i}^{OLD} - \mathcal{L}_{Q_i}^{NEW} \right), \quad (14)
\]

where \( \mathcal{L}_{Q_i}^{OLD} \) and \( \mathcal{L}_{Q_i}^{NEW} \) are loss functions, defined in Eq.(5) with meta-validation tasks and obtained with \( \text{FS}(\cdot | \Phi_{OLD}) \) and \( \text{FS}(\cdot | \Phi_{NEW}) \) respectively.

Thereby performance improvement with \( \Phi_{NEW} \) against \( \Phi_{OLD} \) will reward the task selector and reinforce it to choose corresponding meta-tasks. To optimize the selector, we adopt the policy gradient algorithm REINFORCE (Williams, 1992) to overcome the issue of non-differentiability of the sampling process. The optimization works by:

\[
\psi \leftarrow \psi - \alpha \nabla_{\psi} \log \pi_\psi(R - b), \quad (15)
\]

where we use \( \pi \) with a slight abuse of notation to denote the task selector parameterized by \( \psi \) and \( \alpha \) is the learning rate. Besides, \( b \) denotes a baseline function, e.g., the moving average of the reward, for reducing computational variance.

4.4 Optimization and Inference

We employ the iterative optimization strategy to optimize three components in the proposed method, i.e., few-shot learner, task generator, and task selector. With the pre-trained representation for entities, the task generator is first optimized to generate meta-tasks \( T_{GEN} \), then the task selector is applied to conduct selection on \( T_{GEN} \), next the few-shot learner can be optimized using selected meta-tasks and given meta-tasks \( T_{train} \). With the obtained reward \( R \), we can update the task selector according to Eq.(15). The whole process is repeated with enough iterations until all components converge.

In the inference stage, we use the optimized few-shot learner to make inferences on new relations in meta-testing set \( T_{test} \). Similar to the meta-training stage, \( \text{FS}(\cdot) \) can learn relation representation using the support set in \( T_{test} \) for a new relation. Then the generated representation is leveraged to evaluate

<table>
<thead>
<tr>
<th>KGs</th>
<th>FR</th>
<th>ES</th>
<th>JA</th>
<th>EL</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Entities</td>
<td>13,176</td>
<td>12,382</td>
<td>11,805</td>
<td>5,231</td>
</tr>
<tr>
<td>#Relations</td>
<td>178</td>
<td>144</td>
<td>128</td>
<td>111</td>
</tr>
<tr>
<td>#Triples</td>
<td>49,015</td>
<td>54,066</td>
<td>28,774</td>
<td>13,839</td>
</tr>
<tr>
<td>#Meta-training Tasks</td>
<td>25</td>
<td>30</td>
<td>24</td>
<td>21</td>
</tr>
<tr>
<td>#Meta-validation Tasks</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>#Meta-testing Tasks</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>
the test triples in the query set. Note that FS(·) is not updated by using the query set anymore in the meta-testing stage.

5 Experiments

5.1 Experimental Settings

Dataset. We adopt a multilingual KG dataset (Chen et al., 2020) for evaluation. Specifically, we select four different language-specific KGs extracted from French (FR), Spanish (ES), Japanese (JA) and Greek (EL) DBpedia (Lehmann et al., 2015) as low-resource KGs, as they only have a small number of frequent relations that can offer triples for training task construction. Table 1 shows the statistics of used datasets.

Baselines. To validate the effectiveness of our method, we compare our method with two groups of baseline methods. The first group consists of canonical KG completion models including TransE (Bordes et al., 2013), DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2017), SimplE (Kazemi and Poole, 2018), and WRAN (Zhang et al., 2020b). The second group is the few-shot KG completion models, including GMatching (Xiong et al., 2018), FSRL (Zhang et al., 2020a), MetaR (Chen et al., 2019), and GANA (Niu et al., 2021).

Implementation. We implement the proposed method using the Python library Pytorch and conduct all the experiments on an NVIDIA GeForce RTX 3090Ti. Following the popular setting (Zhang et al., 2020a; Xiong et al., 2018), we remove all the relations in \( T_{\text{train}} \) and \( T_{\text{test}} \) from KG \( \mathcal{G} \) and obtain a background knowledge graph for pre-training the KG embedding leveraging DistMult (Yang et al., 2015) as the KG encoder. After that, the pre-trained embeddings of entities and relations can be used in the proposed method. Following the popular procedure and the setting in the related works (Zhang et al., 2020a; Xiong et al., 2018), we extract the few-shot learning tasks and divide them into the meta-training, meta-validation, and meta-testing tasks. Specifically, we select the relations with less than 500 but more than 50 triples for preparing the few-shot learning tasks following the popular setting (Zhang et al., 2020a; Xiong et al., 2018). Note that even though the meta-testing tasks include relations with more than 50 triples, we only sample \( K = 1 \) or \( K = 3 \) triples as the support set during the meta-testing phase and adopt the rest for evaluation to imitate the real few-shot scenario.

For the few-shot learner, we use three-layer MLP to implement FS(·) with LeakyReLU as the activation function. The hidden units of each layer in FS(·) are set as 500, 200, and 100, respectively. And we set the margin parameter \( \gamma \) as 1.0 and set the step size \( \beta \) as 5.0. For the task generator, we use two-layer MLP to implement MLP\(_{\text{enc}}(\cdot)\) and MLP\(_{\text{dec}}(\cdot)\) with ReLU as the activation function. The hidden units of each layer in MLP\(_{\text{enc}}(\cdot)\) are set as 512, and 256, respectively. And the hidden units of each layer in MLP\(_{\text{dec}}(\cdot)\) are set as 256, and 512, respectively. Besides, we use a linear layer to implement \( f_{\text{rs}}(\cdot) \) and \( f_{\sigma}(\cdot) \) with the layer size as 256. For the task selector, we use two-layer MLP to implement MLP\(_{\text{ts}}(\cdot)\) with LeakyReLU as the activation function. The hidden units of each layer in MLP\(_{\text{ts}}(\cdot)\) are set as 100, and 50, respectively. The dimension of entity embedding is set as 100 for all methods. Besides, we also find the optimal parameters or follow the original paper to achieve the best performance for baseline methods. Note that all triples in the background KGs and the meta-training tasks, as well as all triples corresponding to entity pairs in the support set of meta-validation and meta-testing tasks, should be used to train the canonical KG completion models. For optimization, we employ Adam optimizer to optimize all loss functions with a learning rate of 0.001. The model trained on the meta-training tasks can be used for the meta-validation tasks every 1000 epochs, and the model parameters and corresponding performance will be recorded. Then the model with the best performance on MRR can be used as the final model for meta-testing. Besides, we use early stopping with 30 patient epochs during meta-training. Following the previous work, we report the Hits@1, Hits@5, Hits@10, and MRR (mean reciprocal rank) results to evaluate the performance of few-shot KG completion. Each evaluation is repeated 3 times and averaged results are reported.

5.2 Experimental Results

Performance comparison. The results of all evaluated KG completion models on four different low-resource KGs with \( K = 1 \) and \( K = 3 \) are shown in Table 2. We can observe that (1) the proposed method FLow-KGC has superior performance over canonical KG completion models and few-shot KG completion models by Hits@1, Hits@5, Hits@10, and MRR; (2) The long-tail problem of low-resource KGs has a significant im-
Table 2: Few-shot KG completion comparison on low-resource KGs. The few-shot size \( K = 1 \) and \( K = 3 \). The best results are in bold, and the strongest baseline is indicated with *.

(a) The few-shot size \( K = 1 \).

<table>
<thead>
<tr>
<th>KGs</th>
<th>ES (Spanish)</th>
<th>EL (Greek)</th>
<th>FR (French)</th>
<th>JA (Japanese)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>MRR Hits@1 Hits@5 Hits@10</td>
<td>MRR Hits@1 Hits@5 Hits@10</td>
<td>MRR Hits@1 Hits@5 Hits@10</td>
<td>MRR Hits@1 Hits@5 Hits@10</td>
</tr>
<tr>
<td>TransE</td>
<td>0.068 0.003 0.127 0.191</td>
<td>0.043 0.001 0.078 0.124</td>
<td>0.078 0.001 0.159 0.213</td>
<td>0.103 0.008 0.211 0.271</td>
</tr>
<tr>
<td>DistMult</td>
<td>0.051 0.019 0.055 0.108</td>
<td>0.059 0.029 0.077 0.116</td>
<td>0.035 0.008 0.045 0.081</td>
<td>0.065 0.008 0.103 0.161</td>
</tr>
<tr>
<td>ComplEx</td>
<td>0.089 0.040 0.124 0.188</td>
<td>0.162 0.015 0.217 0.263</td>
<td>0.075 0.036 0.106 0.165</td>
<td>0.091 0.037 0.135 0.188</td>
</tr>
<tr>
<td>SimpleE</td>
<td>0.048 0.017 0.054 0.111</td>
<td>0.059 0.02 0.093 0.132</td>
<td>0.029 0.006 0.066 0.096</td>
<td>0.076 0.018 0.111 0.21</td>
</tr>
<tr>
<td>WRAN</td>
<td>0.105 0.035 0.165 0.224</td>
<td>0.084 0.053 0.117 0.242</td>
<td>0.097 0.018 0.162 0.223</td>
<td>0.096 0.042 0.135 0.206</td>
</tr>
</tbody>
</table>

(b) The few-shot size \( K = 3 \).

<table>
<thead>
<tr>
<th>KGs</th>
<th>ES (Spanish)</th>
<th>EL (Greek)</th>
<th>FR (French)</th>
<th>JA (Japanese)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>MRR Hits@1 Hits@5 Hits@10</td>
<td>MRR Hits@1 Hits@5 Hits@10</td>
<td>MRR Hits@1 Hits@5 Hits@10</td>
<td>MRR Hits@1 Hits@5 Hits@10</td>
</tr>
<tr>
<td>TransE</td>
<td>0.186 0.153 0.248 0.289</td>
<td>0.212 0.146 0.297 0.329</td>
<td>0.168 0.125 0.203 0.274</td>
<td>0.162 0.143 0.258 0.278</td>
</tr>
<tr>
<td>DistMult</td>
<td>0.195 0.167 0.254 0.265</td>
<td>0.202 0.162 0.294 0.324</td>
<td>0.172 0.128 0.216 0.273</td>
<td>0.167 0.081 0.268* 0.288</td>
</tr>
<tr>
<td>ComplEx</td>
<td>0.223 0.176 0.265 0.298</td>
<td>0.245 0.176 0.305* 0.347*</td>
<td>0.183 0.145 0.239 0.304</td>
<td>0.194 0.165 0.263 0.297*</td>
</tr>
<tr>
<td>SimpleE</td>
<td>0.242* 0.194* 0.285* 0.317*</td>
<td>0.247* 0.199* 0.284 0.338</td>
<td>0.210* 0.158* 0.256* 0.315*</td>
<td>0.227* 0.186* 0.264 0.285</td>
</tr>
</tbody>
</table>

Ablation study. To gain deeper insight into the effectiveness of each component in the proposed model, we conduct ablation studies by comparing the following variants with FLow-KGC: (1) FLow-KGC-FSL that only keeps a few-shot learner and easy to overfit the small set of training tasks. FLow-KGC-FSL outperforms FLow-KGC-FSL, because the task generator and task selector achieve the best performance among these variants because the task selector distinguish beneficial tasks from noisy tasks and utilizes the

Table 3: Results of ablation study.

<table>
<thead>
<tr>
<th>Methods</th>
<th>MRR Hits@1 Hits@5 Hits@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLow-KGC-FSL</td>
<td>0.208 0.160 0.256 0.312</td>
</tr>
<tr>
<td>FLow-KGC-w/o-S</td>
<td>0.231 0.179 0.276 0.337</td>
</tr>
<tr>
<td>FLow-KGC</td>
<td>0.240 0.185 0.282 0.348</td>
</tr>
</tbody>
</table>

EL |

<table>
<thead>
<tr>
<th>MRR Hits@1 Hits@5 Hits@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLow-KGC-FSL</td>
</tr>
<tr>
<td>FLow-KGC-w/o-S</td>
</tr>
<tr>
<td>FLow-KGC</td>
</tr>
</tbody>
</table>
selected tasks to learn a better few-shot learner.

Impact of few-shot size $K$. To investigate the impact of few-shot size $K$ on the performance of few-shot KG completion, we test MetaR, GANA, and FLow-KGC with size $K = 1, 3, 5, 7, 9$ on Spanish (ES) and Greek (EL) KGs. Figure 2 shows Hits@1 results of three methods with different $K$. We see that FLow-KGC consistently outperforms the other two baseline methods with different sizes $K$, demonstrating the effectiveness of the proposed method. And the performance of all methods gains improvement with $K$ increasing because the larger support set provides more entity pairs to optimize the few-shot learner and learn more accurate representations for rare relations.

Impact of the number of synthetic tasks. The number of virtual relations $V$ and the number of generated tasks $N_v$ for each virtual relation are hyper-parameters to decide the number of synthetic tasks. Figure 3 shows MRR results of FLow-KGC with different $V$ and $N_v$ on French (FR) and Spanish (ES) KGs. First, with a fixed $N_v = 20$ for French KG and a fixed $N_v = 25$ for Spanish KG, we find that FLow-KGC has superior performance when $V = 10$. Second, with a fixed $V = 10$, FLow-KGC with $N_v = 20$ and $N_v = 25$ performs best for French KG and Spanish KG, respectively. We think the reasons behind the observations are similar. A small $V$ or $N_v$ cannot effectively complement the current tasks, and a larger $V$ or $N_v$, might introduce more noisy tasks and damage the performance of the few-shot learner.

Impact of the proportion of task selection. FLow-KGC selects $B$ beneficial tasks from the generated tasks to achieve the adaptive selection. Here we denote the selection proportion as $\frac{B}{V \times N_v}$. To evaluate the impact of the proportion of task selection, we evaluate FLow-KGC with selection proportion 10%, 30%, 50%, 70%, 90%. Figure 4 shows the results on two KGs (ES and JA). We find that FLow-KGC with a selection proportion of 50% achieves the best performance. We think that a larger proportion unavoidably introduces more noisy tasks into the meta-training phase and a smaller proportion discards extra beneficial tasks, which hurts the effectiveness of FLow-KGC.

6 Conclusion

In this paper, we proposed a novel few-shot KG completion model to ease the adverse impact of the long-tail issue on low-resource KG completion. Specifically, we designed a task generator based on a conditional variational autoencoder to generate synthetic meta-tasks and proposed a task selector to adaptively select beneficial meta-tasks for optimizing a few-shot learner, which further provides the feedback to update the task selector following the principle of reinforcement learning. These three components constitute our method FLow-KGC. Extensive experimental results demonstrate the rationality and effectiveness of our proposed method.

Limitations

Despite achieving superior performance, our proposed method requires manual selection for hyper-parameters to decide the number of tasks, i.e., the number of virtual relations $V$ and the number of synthetic tasks $N_v$ for each virtual relation. In future work, we target developing the method with automatic adjustment to add/remove virtual rela-
tions and the corresponding tasks according to the status of the few-shot learner with the training going on by curriculum learning. Besides, although we adopt a task selector to adaptively select beneficial tasks, it is still inevitable to bring noisy tasks in the meta-training stage. We will explore the strategy to achieve better denoising.

References


Qingyu Guo, Fuizhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. IEEE Transactions on Knowledge and Data Engineering.


Guanglin Niu, Yang Li, Chengguang Tang, Ruiying Geng, Jian Dai, Qiao Liu, Hao Wang, Jian Sun, Fei Huang, and Luo Si. 2021. Relational learning with gated and attentive neighbor aggregator for few-shot knowledge graph completion. SIGIR.


ACL 2023 Responsible NLP Checklist

A  For every submission:

✓  A1. Did you describe the limitations of your work?
   Section of limitations

☒  A2. Did you discuss any potential risks of your work?
   No potential risks.

✓  A3. Do the abstract and introduction summarize the paper’s main claims?
   Left blank.

☒  A4. Have you used AI writing assistants when working on this paper?
   Left blank.

B  ☒ Did you use or create scientific artifacts?
   Left blank.

☐  B1. Did you cite the creators of artifacts you used?
   Not applicable. Left blank.

☐  B2. Did you discuss the license or terms for use and/or distribution of any artifacts?
   Not applicable. Left blank.

☐  B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
   Not applicable. Left blank.

☐  B4. Did you discuss the steps taken to check whether the data that was collected/used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect/anonymize it?
   Not applicable. Left blank.

☐  B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
   Not applicable. Left blank.

✓  B6. Did you report relevant statistics like the number of examples, details of train/test/dev splits, etc. for the data that you used/created? Even for commonly-used benchmark datasets, include the number of examples in train/validation/test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
   Left blank.

C  ✓ Did you run computational experiments?
   Section 5

✓  C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
   Section 5

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.
C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?

Section 5

C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?

Section 5

C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?

Section 5

D  X Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?

No response.

D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants’ demographic (e.g., country of residence)?

No response.

D3. Did you discuss whether and how consent was obtained from people whose data you’re using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?

No response.

D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?

No response.

D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?

No response.