Meta-CQG: A Meta-Learning Framework for Complex Question Generation over Knowledge Bases

Kun Zhang[1,2], **Yunqi Qiu**[4], **Yuanzhuo Wang**[1,2,3]*, **Long Bai**[2], **Wei Li**[4], **Xuhui Jiang**[1,2], **Huawei Shen**[1,2], **Xueqi Cheng**[2]

[1]Data Intelligence System Research Center, Institute of Computing Technology, Chinese Academy of Sciences;
[2]School of Computer Science and Technology, University of Chinese Academy of Sciences;
[3]Big Data Academy, Zhongke; [4]Baidu Inc.

{zhangkun18z, wangyuanzhuo, bailong18b, jiangxuhui19g}@ict.ac.cn
{shenhuawei, cxq}@ict.ac.cn {qiuyunqi, liwei85}@baidu.com

Abstract

Complex question generation over knowledge bases (KB) aims to generate natural language questions involving multiple KB relations or functional constraints. Existing methods train one encoder-decoder-based model to fit all questions. However, such a one-size-fits-all strategy may not perform well since complex questions exhibit an uneven distribution in many dimensions, such as question types, involved KB relations, and query structures, resulting in insufficient learning for long-tailed samples under different dimensions. To address this problem, we propose a meta-learning framework for complex question generation. The meta-trained generator can acquire universal and transferable meta-knowledge and quickly adapt to long-tailed samples through a few most related training samples. To retrieve similar samples for each input query, we design a self-supervised graph retriever to learn distributed representations for samples, and contrastive learning is leveraged to improve the learned representations. We conduct experiments on both WebQuestionsSP and ComplexWebQuestion, and results on long-tailed samples of different dimensions have been significantly improved, which demonstrates the effectiveness of the proposed framework.

1 Introduction

Question generation (QG) over knowledge base (KB) aims to generate natural language questions with structured KB queries, which has been widely used to improve the performance of question answering (QA) by data augmentation for the training corpora. It can also help chatbots ask questions during human-computer interaction.

Traditional methods (Jia and Liang, 2016; Seyler et al., 2017) rely on hand-crafted rules and templates to convert KB queries into questions, leading to poor generalization. Recently, neural approaches (Kumar et al., 2019; Chen et al., 2020) leveraged one encoder-decoder-based model to fit the entire training set, then used the trained model to generate questions in the testing phase. However, such a one-size-fits-all strategy may not perform well for generating complex questions which contain multiple KB relations or functional constraints, such as comparison and sorting, and have complex semantic structures. This is due to the uneven distribution of the training set.

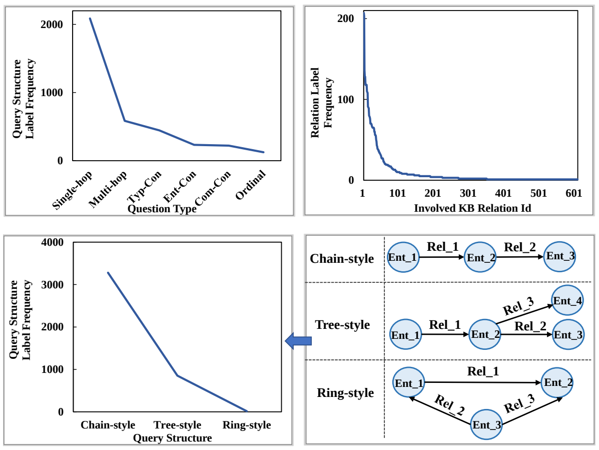

Figure 1: The distribution of question type (Typ-Con as type constraint, Ent-Con as entity constraint, Com-con as comparative constraint), relation type (the involved KB relation in the query) and query structure in training set of WebQSP. We also give the illustration of three types of query structure

Take a widely used dataset WebQuestionsSP (WebQSP) (Yih et al., 2016) as an example, questions are unevenly distributed across multiple dimensions, including question types, involved KB relations, and query structures. As illustrated in Figure 1, single-hop questions are the most common type of questions in the dataset, e.g., the question "What kind of money to take to Bahamas?" relies on a single-hop KB query "(Bahamas, currency_used, ?x)". In contrast, questions with comparative or sorting constraints are the long-tailed samples, e.g., the question " What was the first

*Corresponding author.

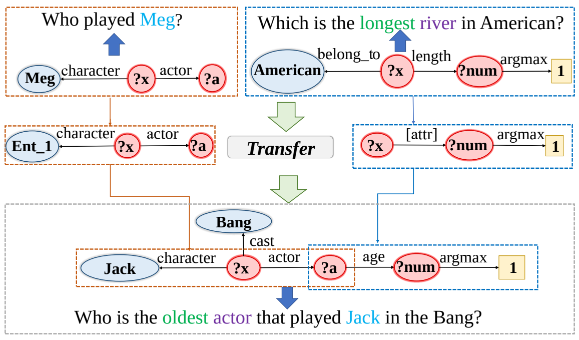

Figure 2: An example of meta-knowledge transferring.

```
SELECT ?a WHERE {
Sri Lanka location.location.adjoin_s ?x .
?x ns:location.adjoining_relationship.adjoins ?a .
?a ns:common.topic.notable_types Country .
?a ns:location.country.capital New Delhi . }
```

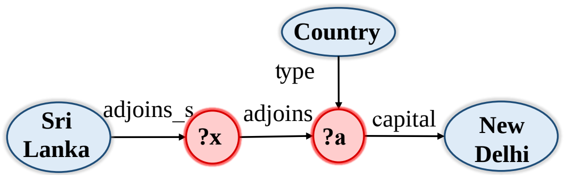

Figure 3: An example of SPARQL query and its corresponding query graph.

book Charles Dickens wrote?" requires sorting over the copyright date of Dickens' books. The same situation occurs in the dimensions of involved KB relations and query structures. Existing neural approaches can be easily biased towards dominant samples and perform poorly on long-tailed ones.

To deal with the problem of data imbalance, we resort to the process of human cognition. As shown in Figure 2, when faced with a rare query, humans can write the corresponding question according to the previously learned query patterns. In this paper, we collectively refer to these universal and transferable query patterns as *meta-knowledge*, which can help generate long-tailed questions. To learn such meta-knowledge, we propose a meta-learning framework for KBQG, namely Meta-CQG. During model training, each sample in the training set is viewed as a query set, and its similar samples are retrieved to form the support set. Our generator adapts to each query set by trials and the supervision signals on the support set. Through the model-agnostic meta-learning (MaML) (Finn et al., 2017) algorithm, the QG model can learn to generalize over varied samples to acquire the meta-knowledge, instead of fitting all samples.

To select similar samples and construct the support sets, we design a self-supervised graph retriever that takes into account the similarity between different samples in different dimensions, including question types, involved KB relations, and query structures. Specifically, the graph retriever encodes the input queries into distributed representations, and the cosine similarities between these vectors denote the similarities between different queries. Due to lack of supervision, we train the graph retriever in a self-supervised way, and contrastive learning is leveraged to improve the learned representations. To demonstrate the effectiveness of the proposed framework, we conduct extensive experiments and ablation studies on two widely-used datasets, and the results on long-tailed samples under different dimensions have been significantly improved.

In general, our main contributions are listed as follows:

- We propose a meta-learning framework for complex question generation over knowledge bases, overcoming the challenge of data imbalance.
- We design a self-supervised graph retriever to select the most similar samples to construct support set during the phase of meta-learning and help the generator better acquire meta-knowledge.
- We demonstrate the effectiveness of the proposed framework on two widely-used datasets and achieve state-of-the-art performance.

2 Preliminary

We aim to generate complex questions from queries, which can be executed on the knowledge base to get the answers to the generated questions. As the query is always displayed in the form of graph, we represent the query $\mathcal{Q}$ with a query graph $\mathcal{G}$. Then we translate the $\mathcal{G}$ to the corresponding complex question.

Knowledge Base. A knowledge base $\mathcal{K}$ is a collection of triples in the form of (s, r, o), where s, r, and o denote subject, relation, and object respectively.

Query Graph. As described in (Qiu et al., 2020), query graph $\mathcal{G}$ is a graph representation of SPARQL query. As shown in Figure 3, Our query graph consists of two types of nodes: variable nodes and non-variable nodes. Variable nodes represent ungrounded KB nodes or values. A non-variable node can be a grounded KB entity or KB type, such as Sri Lanka and Country.

Complex Question. While a simple question can be answered by a single KB triple, complex questions require more information and even functional operations, such as comparison, aggregation, and sorting.

3 Methodology

In this section, we will describe the proposed framework, Meta-CQG. Figure 4 gives an overview of our framework, which mainly consists of two parts: Query-agnostic Meta Learning (QaML) and the graph retriever. QaML trains a unique generator for each target query by learning the potential features of retrieved similar samples. The graph retriever selects a few most similar samples and construct the support set.

3.1 Query-agnostic Meta Learning

Considering the adaption cost, QaML adopts the MAML algorithm, which can be adapted to the target query via a few training samples in a few training steps. QaML contains two components, the meta learner and the adapted learner. The adapted learner is the adaptive question generator and the meta learner allocates initial parameters for the adapted learner. The meta-learning process can be divided into meta-training process and meta-testing process. In the meta-training process, the meta learner is trained on the support set data to get the adapted learner. Then, we update the meta learner through evaluating on the query set data by the adapted learner. In the meta-testing process, the meta learner is fixed and we leverage the adapted learner to encode the query set data and generate the question. We will describe the meta learner and the adapted learner in detail below.

3.1.1 Meta Learner

In this section, we will describe the meta learner, which aims to learn an initial set of parameters that can quickly adapt to a task-specific learner via similar samples.

In the meta learning setting, a task consists of a support set and a query set. The query set only contains one sample to be generated, and the support set contains the training samples which are most similar to the query set. Take a sample q as an example, the query set $\mathrm{s}_{query} = \{q\}$. We denote the top-N similar samples selected by the graph retriever to form the support set $s_{support} = \{x_{q,1}, ..., x_{q,N}\}$. We denote the query and the question in the support set as $g_{support}$ and $q_{support}$, the query set as g_{query} and q_{query}.

In the meta-training process, the meta learner allocates initial parameters for the adapted learner and is updated through evaluating on the query set data by the adapted learner. We denote the parameter of the meta learner as $\boldsymbol{\theta}$ and the adapted learner as $\boldsymbol{\theta}'$.

During the training phase, the model will be initialized with the parameters of the meta learner. After t iterations of training on $\mathrm{s}_{support}$, the model updates the parameter $\boldsymbol{\theta}$ and gets the adapted learner for the query set. In other words, the model parameterized by $\boldsymbol{\theta}$, is updated to $\boldsymbol{\theta}'$ by standard gradient descent,

$$\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} - \eta_1 \nabla_{\boldsymbol{\theta}} \mathcal{L}(g_{support}, q_{support}; \boldsymbol{\theta}), \qquad (1)$$

where $\mathcal{L}$ is the loss function.

Then, we leverage the adapted learner to obtain the loss on the query set. We apply stochastic gradient descent on the initial parameter $\boldsymbol{\theta}$, i.e. the parameter of the meta learner, by minimizing the loss from query set,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_2 \nabla_{\boldsymbol{\theta}} \mathcal{L}(g_{query}, q_{query}; \boldsymbol{\theta}') \qquad (2)$$

where η_2 is meta-learning rate. The pseudo code of meta training process is shown in Algorithm 1.

In the meta-testing process, the meta learner is fixed. We initialize the adapted learner by the parameter of the meta learner.

3.1.2 Adapted Learner

In our task, the adapted learner is the question generator, which translates the generated query graph into a natural language question. Based on the query graph constructed above, we adopt a novel graph-to-sequence model to generate sequences.

Inspired by (Guo et al., 2019), we leverage Densely Connected Graph Convolutional Network (DCGCN) as the graph encoder. It applies dense connectivity among Graph Convolutional Network (GCN) layers. Each DCGCN block consists of two sub-blocks to capture graph structure at different abstract levels. Each sub-block consists of several GCN layers, where each GCN layer is connected to all previous layers. The input of layer l for node u is defined as,

$$g_u^{(l)} = \left[\mathbf{x}_u; \mathbf{h}_u^{(1)}; \ldots; \mathbf{h}_u^{(l-1)}\right], \qquad (3)$$

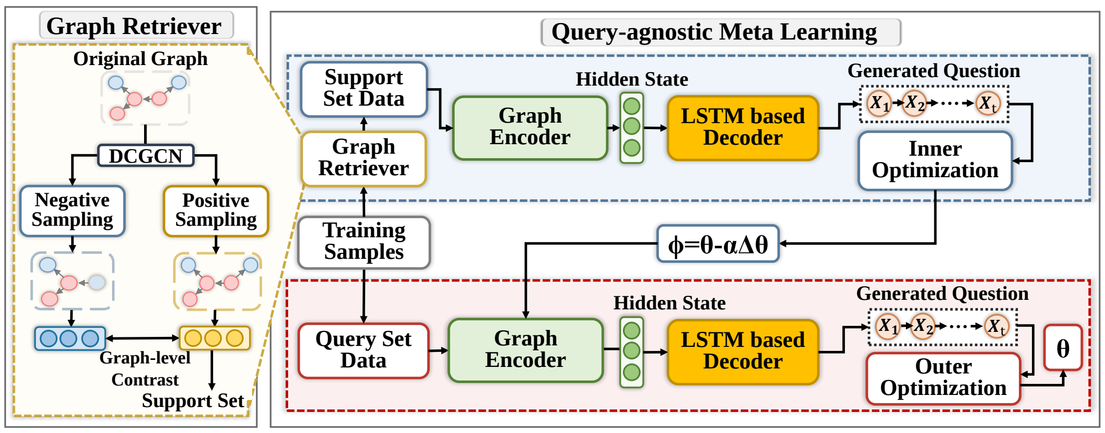

Figure 4: The overall architecture of our framework for question generation over knowledge bases. It can be divided into two parts, i.e., the graph retriever and QaML are shown on the left and right respectively. (i) Graph Retriever (left): from top to bottom, we first pre-train the graph encoder with the task of link prediction. Then we used Similarity-based Contrastive Learning (SCL) to fine-tune the pre-trained graph encoder as the graph retriever. (ii) QaML(right): Given a target query (as query set) in the training set, we retrieve similar samples to construct a support set for the target query. The blue square (above) describes the training process on the support set, while the pink square(below) describes the testing process on the query set.

Algorithm 1 Meta-training process

```
Require: Dataset: $S_{\text{train}}$ ; step hyper parameters: $\eta_1, \eta_2$;
 1: start training:
 2: Randomly initialize $\boldsymbol{\theta}$
 3: for $S_i$ in $S_{\text{train}}$ do
 4:     Expand $S_i \rightarrow D_i$
 5:     $(\mathcal{S}_i^{\text{support}}, \mathcal{S}_i^{\text{query}}) \sim Task_i$.
 6:     Evaluate $\nabla_\theta \mathcal{L}(g_{support}, q_{support}; \boldsymbol{\theta})$ using $\mathcal{S}_i^{\text{support}}$
 7:     Compute adapted parameters with gradient descent:
 8:     $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} - \eta_1 \nabla_{\boldsymbol{\theta}} \mathcal{L}(g_{support}, q_{support}; \boldsymbol{\theta})$
 9:     Evaluate $\nabla_\theta \mathcal{L}(g_{query}, q_{query}; \boldsymbol{\theta}')$ using $\mathcal{S}_i^{\text{query}}$
10:     $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_2 \nabla_{\boldsymbol{\theta}} \mathcal{L}(g_{query}, q_{query}; \boldsymbol{\theta}')$
11: end for
12: end training
```

where $[\cdot;\cdot]$ denotes the concatenation of vectors; $\mathbf{x}_u$ denotes the node embedding of u ; and $\mathbf{h}_u^{(i)}$ denotes the output of layer i for node u. We randomly initialize the embedding of nodes.

In our case, the edge information is crucial as it corresponds to important tokens in the generated question. In order to model both the node and edge information with GNNs, we utilize Levi graph transformation method to transform the input query graph into its equivalent Levi graph(Levi, 1942), which views the predicates as nodes in the graph. Following (Beck et al., 2018), we add reverse and self-loop edges to the Levi graph. To compute the graph-level embedding, we leverage the pooling-based method, which feeds the output node embedding into a fully-connected neural network and applies the element-wise max-pooling operation on all node embeddings to derive the graph embedding $\mathbf{h}_\mathcal{G} \in \mathbb{R}^d$.

We adopt an attention-based LSTM decoder (Bahdanau et al., 2014) that generates the output sequence one word at a time. The graph embedding $\mathbf{h}_\mathcal{G}$ is used as the initial input of the decoder. We carefully follow the attention mechanism used in (Tu et al., 2016).

3.2 Graph Retriever

To construct the support set, we retrieve samples that are most similar to the target KB query from training set. As queries are always be represented by graphs, this problem is defined as a graph retrieval problem. The core and most challenging part is to measure similarity between two graphs.

Considering both the structure and content information of the query graph, we adopt the neural-based method to measure graph similarity. Most neural-based models only focus on homogeneous graphs and require large-scale annotated data. However, our task lacks gold label about the similarity between graphs. Thus, we train the graph retriever in an unsupervised way. To pre-train the graph, we utilize DCGCN to encode the graph and the link predication task as downstream task.

Take (g_1, q_1) and (g_2, q_2) as an example. The query representation is calculated as:

$$\begin{aligned} \mathbf{h}_{g1} &= \mathrm{DCGCN}(g_1), \\ \mathbf{h}_{g2} &= \mathrm{DCGCN}(g_2), \end{aligned} \tag{4}$$

where $\mathbf{h}_{g1}$ and $\mathbf{h}_{g2}$ is the graph embedding of g_1 and g_2. And we adopt cosine function to calculate the query similarity:

$$\delta_g = cos(\mathbf{h}_{g1}, \mathbf{h}_{g2}) \tag{5}$$

where δ_g is the similarity score between g_1 and g_2

Only focusing on the query graph itself is hard to learn the mapping relation between query graph and question, leading to many false samples which have similar query graphs but quite different questions. Thus, we propose a Similarity-based Contrastive Learning (SCL) method to fine-tune the pre-trained graph encoder, which utilizes question semantic to emphasize some easily overlooked but crucial information on the query graph.

3.2.1 Contrastive Samples Construction

Each sample contains one query and one question, which is in the form of a graph and a sequence. We construct contrastive samples according to the similarity between queries and questions. We use fixed Bert (Devlin et al., 2018) to encode question and get its representation,

$$\begin{aligned} \mathbf{h}_{q1} &= Bert(q_1), \\ \mathbf{h}_{q2} &= Bert(q_2), \end{aligned} \tag{6}$$

where $\mathbf{h}_{q1}$ and $\mathbf{h}_{q2}$ is denoted as the representation of q_1 and q_2. We adopt cosine function to calculate question similarity δ_s as follow,

$$\delta_q = cos(\mathbf{h}_{q1}, \mathbf{h}_{q2}) \tag{7}$$

The query similarity between two sample has been presented above. Then, we define positive and negative sample selection score λ_{pos}, λ_{neg} as:

$$\begin{aligned} \lambda_{pos} &= (\delta_g - \delta_{gpos})(\delta_q - \delta_{qpos}), \\ \lambda_{neg} &= (\delta_g - \delta_{gneg})(\delta_{qneg} - \delta_q), \end{aligned} \tag{8}$$

where δ_{gpos} , δ_{qpos}, δ_{gneg}, δ_{qneg} is a series of thresholds we set according to the similarity distribution.

If both the two similarities are larger than the thresholds, i.e., $\delta_g > \delta_{gpos}$ and $\delta_q > \delta_{qpos}$, the pair of the two samples is positive.

If either $\delta_q < \delta_{qneg}$ or $\delta_g < \delta_{gneg}$, the sample pair is negative.

3.2.2 Contrastive Training

Our goal is to encourage the graph encoder $\mathcal{E}$ to learn discriminative query representations. After pre-training, We denote the node features in positive samples and negative samples as $\mathbf{X}$ and $\tilde{\mathbf{X}}$, the adjacency matrix as $\mathbf{A}$ and $\tilde{\mathbf{A}}$, the query representation in target, positive and negative sample as $\mathbf{h}_g$, $\mathbf{h}_{gp}$ and $\mathbf{h}_{gn}$. We leverage the discriminator $\mathcal{D}$ to maxmize the mutual information, such that $\mathcal{D}\left(\mathbf{h}_g, \mathbf{h}_{gn}\right)$ represents the probability scores assigned to this pair.

For the objective, we follow the intuitions from Deep Graph InfoMax (Velickovic et al., 2019) and introduce a contrastive objective $\mathcal{L}_{\mathrm{CL}}$ with a standard binary corss-entropy (BCE) loss. The $\mathcal{L}_{\mathrm{CL}}$ is defined as:

$$\begin{aligned} \mathcal{L}_{\mathrm{CL}} = \sum_{i=1}^{N} \mathbb{E}_{(\mathbf{X},\mathbf{A})}\left[\log \mathcal{D}\left(\mathbf{h}_g, \mathbf{h}_{gp}\right)\right] + \\ \sum_{j=1}^{N} \mathbb{E}_{(\tilde{\mathbf{X}},\tilde{\mathbf{A}})}\left[\log\left(1 - \mathcal{D}\left(\mathbf{h}_g, \mathbf{h}_{gn}\right)\right)\right], \end{aligned} \tag{9}$$

where N is denoted as the number of positive and negative samples sampled.

After fine-tuning with above loss, we use average pooling to compute the query representation, and retrieve the most similar samples through cosine similarity function.

4 Experiments

In this section, we evaluate Meta-CQG on two widely-used benchmark datasets and show the effectiveness of our method. We first introduce datasets and training settings. Then, we evaluate the proposed model with the state-of-the-art models on both datasets. In addition, we investigate the performance on different dimensions and the influence of different sampling strategies. Finally, we conduct human evaluation to verify the effectiveness of our method.

4.1 Datasets and Preprocessing

We conduct experiments on two widely-used datasets.

WebQSP (Yih et al., 2016) consists of 4,737 question-answer pairs. All the questions are collected through Google Suggest API, and the answers are fetched from Freebase.

Method	CWQ			WebQSP		
	BLEU-4	METEOR	Rouge-L	BLEU-4	METEOR	Rouge-L
L2A	4.01	13.78	30.59	8.01	19.45	32.58
Zero-shot	6.37	16.32	32.10	9.45	21.52	34.78
MHQG	9.35	19.42	35.78	13.34	24.88	39.14
BiGraph2seq	26.01	28.12	53.58	27.86	30.24	62.77
DCGCN	27.36	29.53	54.11	29.82	31.28	63.93
DCGCN+ROS	28.15	30.13	54.69	30.68	32.19	64.56
DCGCN+Finetune	28.43	30.51	55.07	31.37	32.44	64.92
Meta-CQG	**29.52**	**31.72**	**56.03**	**32.87**	**32.92**	**65.09**
w/o Graph Retriever	27.66	29.68	53.88	29.75	31.45	64.08
w/o SCL	28.51	30.63	55.08	31.83	31.73	64.27

Table 1: Experimental results of automatic metrics on two benchmark datasets.

CWQ (Talmor and Berant, 2018) contains 34,689 questions in total. It modified the SPARQLs in WebQSP by including more constraints, and then generated corresponding natural language questions.

Each question in both datasets has a corresponding SPARQL query. We design transformation rules to convert SPARQL query into query graph as our input. For each dataset, we randomly select 80% of the examples for training, 10% for validation, and 10% for testing.

4.2 Baseline Methods

We compare the proposed model with several baseline methods, including the current state-of-the-art model over the two benchmark datasets. **L2A** (Du et al., 2017) is an attention-based Seq-to-Seq model to generate natural language questions from context in open domain conversational systems. **Zero-shot** (Elsahar et al., 2018) is an RNN-based Seq-to-Seq model paired with an original part-of-speech copy action mechanism to generate questions. **MHQG** (Kumar et al., 2019) is a Transformer-based model for automatic generation of multi-hop questions over knowledge bases. **BiGraph2seq** (Chen et al., 2020) is a graph-to-sequence model which leverages Bidirectional Gated Graph Neural Network (Bi-GNN) as the graph encoder to encode the KB subgraphs, and enhance the RNN decoder with copying mechanism. **DCGCN** applys DCGCN as graph encoder and LSTM as decoder. **DCGCN+ROS** leverages DCGCN as the basic model and adopt the Random Over Sampling (ROS) strategy on question type to solve the data imbalance problem. **DCGCN+Finetune** leverages DCGCN to pre-train the training set. When testing, for each sample, we use the graph retriever to build an adaption and finetune the pre-trained model on it.

4.3 Implementation Details

We implement our method on PyTorch platform. The parameters with the best performance on the validation set are selected. For both datasets, the KB embeddings were randomly initialized and updated in the process of training. In meta-learning, we set N=5 when forming the support set. We set $\eta_1 = 1e-4$ (Equation 1) and $\eta_2 = 0.2$ (Equation 2). For SCL, we random select three positive and negative samples for each sample. We set $\delta_{qpos} = 0.8$, $\delta_{spos} = 0.8$, $\delta_{qneg} = 0.6$ and $\delta_{qneg} = 0.4$.

We set the number of DCGCN as 3 and 6 for the two sub-block respectively with an initial learning rate of 0.0003 is adopted as the optimizer. During decoding, beam search with beam size 10 is leveraged.

4.4 Results and Discussion

Following previous studies, we evaluate the performance by a set of N-grams-based metrics for question generation: BLEU-4(Papineni et al., 2002)(B-4.), METEOR(Banerjee and Lavie, 2005), and ROUGE-L(Lin, 2004).

Table 1 shows the results of Meta-CQG and the adopted baselines. Meta-CQG outperforms all the

Dimesion	Categories	BiGraph2Seq			DCGCN			Meta-CQG		
		B-4.	ME.	R-L.	B-4.	ME.	R-L.	B-4.	ME.	R-L.
Question Type	Single-hop(>40%)	28.56	31.38	63.89	30.56	31.79	64.30	33.21	33.15	65.83
	Multi-hop(>20%)	27.60	29.76	63.01	28.86	31.05	64.08	32.03	31.86	64.07
	Type Constraint(<10%)	26.53	28.76	60.23	28.27	30.19	62.76	31.23	31.77	63.38
	Ordinal(<5%)	23.43	27.01	53.99	24.73	27.53	57.07	29.04	30.69	63.25
Query Structure	Chain-Style(>75%)	29.38	31.23	63.65	31.45	32.96	64.45	33.18	33.42	65.79
	Tree-Style(<20%)	23.53	23.46	58.17	25.77	26.08	59.39	32.16	31.93	64.92
	Ring-Style(<5%)	6.78	17.01	48.65	9.48	20.57	50.78	23.23	30.79	56.79
Relation Type	Notable Types(>30%)	28.77	30.94	60.17	31.27	32.48	62.59	33.02	34.08	65.28
	Inventor(<10%)	17.53	22.96	47.47	20.29	24.57	51.86	27.33	31.88	62.13
	Award Honor(<1%)	4.58	14.77	33.58	7.33	15.45	37.54	20.29	24.11	52.77

Table 2: Experimental results of automatic metrics on different dimensions.

baselines on the two benchmark datasets. Specifically, Meta-CQG improves the BLEU-4 score by 3.51 on CWQ, 5.01 on WebQSP compared with BiGraph2seq. Meanwhile, Meta-CQG exceeds the baselines by a larger margin on METEOR and ROUGE-L.

Graph-to-Seq models (BiGraph2seq, DCGCN, DCGCN+ROS, DCGCN+Finetune and Meta-CQG) outperform the Seq-to-Seq models (L2A, Zero-shot and MHQG) on both datasets, which indicates the advantages of GNN-based encoders for modeling query graphs, since the RNN-based model and the transformer-based model ignores the explicit graph structure of query graphs. In addition, DCGCN-based models (DCGCN, DCGCN+ROS, DCGCN+Finetune and Meta-CQG) outperform BiGraph2seq, which indicates that DCGCN better captures the non-local interactions between the nodes compared with Bi-GNN.

Meta-CQG outperforms DCGCN+ROS by more than 1.37 on BLEU-4, which indicates that the proposed model is more effective in reducing the data imbalance problem compared with existing data re-balancing approaches. Moreover, it outperforms DCGCN+Finetune by more than 1.09 on BLEU-4, which indicates that the meta-learning framework is better at transferring meta-knowledge compared with the pretrain-finetune framwork.

4.5 Ablation study

To further analyze the effectiveness of different components, we conduct ablation studies which remove the graph retriever and SCL in Meta-CQG. The results are shown in Table 1.

Graph Retriever. To evaluate the effectiveness of graph retriever, we remove it and randomly select samples for each training sample (w/o Graph Retriever). The performance has dropped by more than 1.86 on BLEU-4. This indicates that random select samples cannot provide task-specific knowledge for the training sample, and they may introduce noise during training.

SCL. To evaluate the effectiveness of SCL, we remove the contrastive learning loss (w/o SCL). The performance has dropped by more than 1.01 on BLEU-4. This indicates that SCL utilizes question semantic to enhance query graph information.

4.6 Analyses on Different Dimensions

As mentioned above, complex questions are imbalanced in multiple dimensions. We divide the questions in the WebQSP according to each dimension and evaluate the performance of BiGraph2Seq, DCGCN, and Meta-CQG. We sample some categories from majority types and minority types respectively for each dimension. The proportion of each type in the training set is also listed after the categories.

As shown in Table 2, Meta-CQG achieves the best performance in all three dimensions.

In each dimension, Meta-CQG outperforms the other two baselines on minority types (i.e., <10% in training set), which shows the learning ability of Meta-CQG on imbalanced data. The poor performance of BiGraph2Seq and DCGCN on minority types also verifies that the data imbalance problem greatly affects the model performance.

Strategies	B-4.	ME.	R-L.
Question Type	31.07	32.87	64.97
Relation Type	31.22	32.39	64.43
Query Structure	30.09	31.53	64.27
Our Model	**32.87**	**32.92**	**65.09**

Table 3: Experimental results of different sampling strategies.

Results	CWQ			WebQSP		
	Nat.	Sem.	Cor.	Nat.	Sem.	Cor.
Win	19	37	28	35	35	29
Tie	79	59	69	59	59	64
Lose	2	4	3	6	6	7

Table 4: Wins, losses, and ties of Meta-CQG against the current SOAT (BiGraph2seq) based on the manual evaluation.

In addition, Meta-CQG outperforms the baselines on majority types (i.e., >30% in training set). This may be some samples in a majority type may be in minority type of another dimension. For example, a single-hop question may contain relation of Award Honor. Therefore, Meta-CQG is able to find similar samples across multiple dimensions.

4.7 Analyses on Different Sampling Strategies

We design different sampling strategies to verify the effectiveness of our graph retriever. First, we devise different retrievers according to the three dimensions we mentioned above. For each dimension, the retriever randomly selects a few samples that belong to the same category with the sample to be generated. The results shown in Table 3 demonstrate the effectiveness of our model and verify that our model is able to comprehensively consider the imbalance in all dimensions.

4.8 Human Evaluation

We randomly choose 100 questions from the test set of each dataset. We pair the questions generated by our model and BiGraph2seq. Two human annotators are asked to judge which is better in pairs from three aspects: naturalness, correctness, and semantic. Results are shown in Table 4. Our model outperforms BiGraph2seq as it has more winning instances than losing instances on all two datasets. These results indicate that our model improves the quality of questions from the three aspects.

5 Related Work

Question Generation over Knowledge Bases Most recent works for KBQG mainly adopt encoder-decoder models, and focus on enriching the input information. In (Serban et al., 2016) and (Indurthi et al., 2017), recurrent neural networks are introduced for generating natural language questions from KB facts. To address the challenge of unseen predicates and entity types, (Elsahar et al., 2018) leverages auxiliary contexts in the WiKidata corpus in an encoder-decoder architecture. However, the context cannot cover all predicates. Thus, (Liu et al., 2019) presents a neural encoder-decoder model that integrates diversified off-the-shelf contexts. To tackle the semantic drift problem, (Bi et al., 2020) presents a knowledge-enriched, type-constrained, and grammar-guided model. (Kumar et al., 2019) proposes a model for generating complex multi-hop and difficulty-controllable questions over knowledge bases. To model the graph-structured data, (Chen et al., 2020) applied a bidirectional Gated Graph Neural Network model to encode the KB subgraph. However, existing methods train one model to fit all questions, ignoring the data imbalance in the real world.

Meta-Learning Meta-Learning, i.e.learning-to-learn, aims to build efficient algorithms that can learn the new task quickly. In pursuing this problem, there are three categories of meta-learning methods: learning a metric space to compare low-resource testing samples and rich training samples (Snell et al., 2017; Koch et al., 2015), using an additional meta-learner to update the original learner with a few training examples (Ravi and Larochelle, 2016) and learning a good initialization parameter for fast adaptation(Finn et al., 2017). In this work, we follow the third idea and propose a meta-learning framework based on MaML to solve the data imbalance problem in complex question generation tasks.

6 Conclusion

In this paper, we focus on the task of complex question generation over knowledge bases. We propose a meta-learning framework for complex question generation, namely Meta-CQG, to deal with the data imbalance problem. To consider the imbalance of all dimensions, we adopt the MaML method to train a unique generator for each sample to be generated via a few most similar training samples. Specially, we design a self supervised graph retriever to flexibly retrieve most similar samples. We evaluate the effectiveness of Meta-CQG on two widely-used benchmark datasets, and it outperforms all the baselines.

7 Acknowledgement

Thanks to reviewers for their helpful comments on this paper. This paper is funded by the National Natural Science Foundation of China (Nos.62172393, U1836206 and U21B2046), Zhongyuanyingcai program-funded to central plains science and technology innovation leading talent program (No.204200510002) and Major Public Welfare Project of Henan Province (No.201300311200).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. *arXiv preprint arXiv:1806.09835*.

Sheng Bi, Xiya Cheng, Yuan-Fang Li, Yongzhen Wang, and Guilin Qi. 2020. Knowledge-enriched, type-constrained and grammar-guided question generation over knowledge bases. *arXiv preprint arXiv:2010.03157*.

Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2020. Toward subgraph guided knowledge graph question generation with graph neural networks. *arXiv preprint arXiv:2004.06015*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*.

Hady Elsahar, Christophe Gravier, and Frederique Laforest. 2018. Zero-shot question generation from knowledge graphs for unseen predicates and entity types. *arXiv preprint arXiv:1802.06842*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions of the Association for Computational Linguistics*, 7:297–312.

Sathish Reddy Indurthi, Dinesh Raghu, Mitesh M Khapra, and Sachindra Joshi. 2017. Generating natural language question-answer pairs from a knowledge graph using a rnn based question generation model. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 376–385.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22.

Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, page 0. Lille.

Vishwajeet Kumar, Yuncheng Hua, Ganesh Ramakrishnan, Guilin Qi, Lianli Gao, and Yuan-Fang Li. 2019. Difficulty-controllable multi-hop question generation from knowledge graphs. In *International Semantic Web Conference*, pages 382–398. Springer.

Friedrich Wilhelm Levi. 1942. *Finite geometrical systems: six public lectues delivered in February, 1940, at the University of Calcutta*. University of Calcutta.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Cao Liu, Kang Liu, Shizhu He, Zaiqing Nie, and Jun Zhao. 2019. Generating questions for knowledge bases via incorporating diversified contexts and answer-aware loss. *arXiv preprint arXiv:1910.13108*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Yunqi Qiu, Kun Zhang, Yuanzhuo Wang, Xiaolong Jin, Long Bai, Saiping Guan, and Xueqi Cheng. 2020. Hierarchical query graph generation for complex question answering over knowledge graph. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1285–1294.

Sachin Ravi and Hugo Larochelle. 2016. Optimization as a model for few-shot learning.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *arXiv preprint arXiv:1603.06807*.

Dominic Seyler, Mohamed Yahya, and Klaus Berberich. 2017. Knowledge questions from knowledge graphs. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, pages 11–18.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811*.

Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep graph infomax. *ICLR (Poster)*, 2(3):4.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.