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

Anonymous ACL submission

Abstract

Complex question generation (CQG) aims to generate questions involving multiple Knowledge Base (KB) relations or functional constraints. Existing methods train an encoder-decoder-based model to fit all questions. However, the questions in the real world exhibit an imbalanced distribution in many dimensions, such as question type, relation class, entity class, and query structure. This results in insufficient learning for minority class samples under different dimensions. To address this problem, we propose a meta-learning framework for complex question generation. It trains a unique generator for each sample via retrieving a few most related training samples, which can deeply and quickly dive into the content features (e.g. relation and entity) and structure features (e.g. query structure) of each sample. As retrieved samples directly determine the effectiveness of each unique generator, we design a self-supervised graph retriever to learn the potential features of samples and retrieve the most related samples according to multiple dimensions. We conduct experiments on both WebQuestionSP and ComplexWebQuestion, the results on the minority class of different dimensions have been significantly improved, which demonstrates the effectiveness of the proposed framework

1 Introduction

Complex question generation (QG) aims to generate complex questions from knowledge base (KB) queries containing multiple relations or functional constraints, such as comparison and sorting. It can be applied in the education area and improve the performance of question answering (QA) over the KB by data augmentation for training corpora.

In the real world, complex questions are unevenly distributed across multiple dimensions. As shown in Figure 1, we take the widely used dataset WebQuestionsSP (WebQSP) (Yih et al., 2016) as an example and show the uneven distribution of

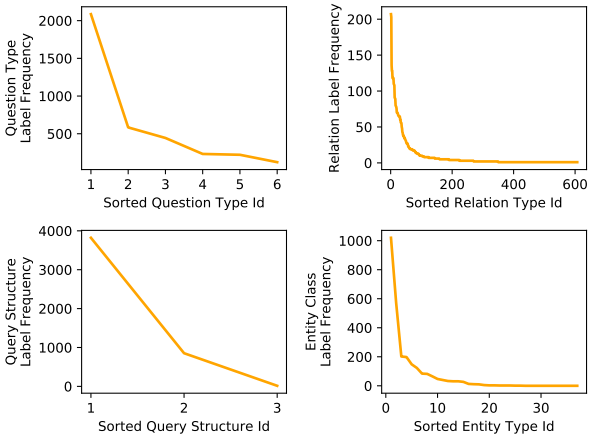


Figure 1: The distribution of question type (Single-hop, Multi-hop, type constraint, entity constraint, comparative constraint, and ordinal constraint), relation type, query structure (chain style, tree style and ring style) and entity type in training set of WebQSP.

complex questions from four dimensions: question type, relation type, entity type, and query structure, where a small portion of types have massive samples (i.e. the majority types) but the others have only a few samples (i.e. the minority types). Current methods train one model to fit all questions, and the model tends to pay more attention to the majority types while neglecting the minority ones, resulting in insufficient training on minority ones across different dimensions and making it challenging to generate various complex questions.

To deal with the data imbalance problem, we propose a meta-learning framework for complex question generation, namely Meta-CQG, where the QG model adapts to the target KB query by trials on the retrieved similar instances. Specifically, the proposed framework adopts the model-agnostic meta learning (MaML) (Finn et al., 2017) algorithm and is composed of the meta learner, the learner, and the retriever. The meta learner learns the initial parameters for the learner in the meta-training stage. In the testing stage, for each target

KB query, the retriever selects a few related samples from the training set to form the support set. Then, the learner (i.e. the question generator) are obtained by training the meta learner on the support set to adapt to the target sample. Therefore, we change the data distribution for each sample, and the learner is able to focus on the knowledge most related to the target query.

Here, one critical problem is how to select related samples and construct the support set for the framework, which comprehensively consider potential features. Since the queries are usually represented as graphs, this problem can be defined as a graph retrieval problem, which aims to retrieve most similar graphs for a given graph. The core of graph retrieval is to measure the similarity between graphs. However, in the complex question generation task there is no label data and the graph contains rich edge label information. This leads to inefficiency and inaccuracy during the retrieving process. To address the challenge, we design a self-supervised graph retriever, which adopt variational graph auto-encoder (V-GAE) (Kipf and Welling, 2016) to comprehensively model the content features and structure features.

In general, our main contributions are listed as follows:

- We propose a meta-learning framework Meta-CQG for complex question generation over knowledge bases, to overcome the challenge of data imbalance.
- We design a graph retriever to select the most related samples and construct support set for meta-learning method, which adopts V-GAE to learn the content features and structure features of queries in an unsupervised approach.
- We demonstrate the effectiveness of the proposed framework on two widely-used datasets. The results on the minority class of different dimensions have been significantly improved, which shows the advantage of our method.

2 Preliminary

We aim to generate complex questions from queries. The input of our task is a query $\mathcal{Q}$ and the output is a question $\mathcal{S}$. As the query is always displayed in the form of graph, we represent the query with a query graph $\mathcal{G}$. Then we translate the $\mathcal{G}$ to the corresponding complex question S

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), a query graph $\mathcal{G}$ is a restricted subset of λ-calculus in the graph representation. Hence, queries can be represented by query graph (Prudhommeaux, 2008). Our query graph consists of three types of nodes: constant node, variable node, and answer node. A constant node can be a grounded KB entity or KB type. Variable nodes and answer nodes represent ungrounded KB nodes or values. There are two types of edges: predicate edge and functional edge. A predicate edge represents a KB relation, such as *wife*. A functional edge indicates a functional operation, such as >, *MinatN*. We adopt query graph as our input.

Complex Question. A natural language complex question S corresponds to a query graph $\mathcal{G}$ over the $\mathcal{K}$, and involves multiple predicate edges or functional edges.

3 Methodology

In this section, we will describe the proposed framework, Meta-CQG. Figure 2 gives an overview of our framework. Our framework mainly consists of two parts: Question-agnostic Meta Learning (QaML) and the graph retreiver. QaML trains a unique generator for each sample by learning the potential features of support set data, i.e., related samples. The graph retriever selects a few most related samples and construct the support set.

3.1 Question-agnostic Meta Learning

QaML contains two parts, the learner as the question generator and the meta learner above the learner, which allocates parameters for the learner. We will describe them 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 related 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 related to it. Take a sample q as an example, the query set $\mathbf{s}_{query}$ only contains sample q , and we

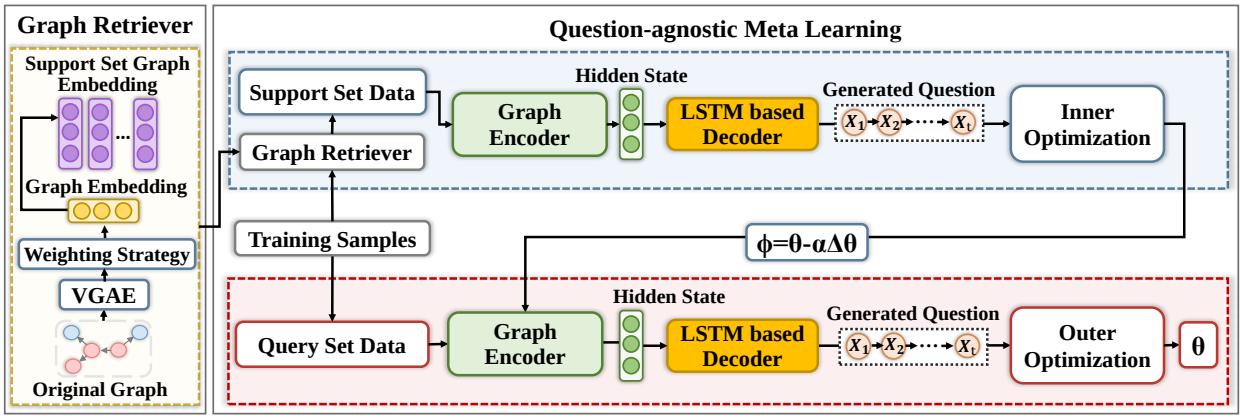


Figure 2: The overall architecture of our framework for question generation over knowledge bases.

use the graph retriever to select top-N similar samples from the training set to construct the support set $\mathbf{s}_{support}$. $\mathbf{s}_{support}$ and $\mathbf{s}_{query}$ construct one task $\mathcal{D}_{\mathrm{q}}$.

The meta-learning process can be divided into meta-training process and meta-testing process.

In the meta-training process, the meta learner trains on the support set data to get the adapted learner and test on the query set data to update the initial parameter. We denote the model parameter of learner as θ. When training on the support set, the model updates the parameter and gets the adapted learner for the query set. After t iterations of training on $\mathbf{s}_{support}$, the question generator, parameterized by θ, is updated to $\boldsymbol{\theta}'$ by standard gradient descent,

$$\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} - \eta_1 \nabla_{\boldsymbol{\theta}} \mathcal{L} \tag{1}$$

where $\mathcal{L}$ is the loss from support set.

When testing on the query set, we apply again stochastic gradient descent on the initial parameter θ by minimizing the the loss from query set $\boldsymbol{\mathcal{L}'}$,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_2 \nabla_{\boldsymbol{\theta}} \boldsymbol{\mathcal{L}'} \tag{2}$$

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

In the meta-testing process, we update the model parameters once and leverage the adapted model to encode the query set data and generate the question. We initialize the model by the parameter of the meta learner and leverage the support set data to update the parameter. Then, we obtain the adapted parameter and the adapted model is used to encode the query set data and generate the question.

Algorithm 1 Meta-training process

Require: Dataset: S_{train} ; step hyper parameters: η_1, η_2;

start training:
Randomly initialize $\boldsymbol{\theta}$
for S_i in S_{train} **do**
  Expand $S_i \rightarrow \mathrm{D}_i$
  $\left(\mathcal{S}_i^{\text{support}}, \mathcal{S}_i^{\text{query}}\right) \sim \mathcal{D}_i$.
  Evaluate $\nabla_\theta \mathcal{L}$ using $\mathcal{S}_i^{\text{support}}$
  Compute adapted parameters with gradient descent:
  $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} - \eta_1 \nabla_{\boldsymbol{\theta}} \mathcal{L}$
  Evaluate $\nabla_\theta \mathcal{L}'$ using $\mathcal{S}_i^{\text{query}}$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_2 \nabla_{\boldsymbol{\theta}} \boldsymbol{\mathcal{L}'}$
end for
end training

3.1.2 Learner

In our task, actually, the learner is the question generator. It 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. Specifically, we leverage DCGCN to encode the query graph into a low-dimensional vector. Then it adopts LSTM to decode the vector into a question.

Graph Neural Networks have achieve great success in modeling graph-like data. However, existing study which uses graph neural networks in KBQG (Chen et al., 2020) faces difficulty in capturing the non-local interactions among nodes. For example, knowledge subgraph usually contains CVT nodes, which are used to express the relationship among several other nodes. Yet this node will never appear in the question. Therefore, the generated

question should express the meaning of the multi-hop paths connected by the CVT nodes, which we call the non-local interactions.

To better capture non-local interactions among nodes, we leverage DCGCN (Guo et al., 2019) 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,

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

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.

In order to model both the node and edge information with GNNs, we utilize levi graph transformed above. 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 embeddings 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 sample to be generated, i.e., query-question pairs. 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.

Considering both the structure and content information of the query graph, we adopt the neural-based method to measure graph similarity. Most neural-based model only focus on homogeneous graphs and ignore the edge label information. To enhance the predicate information when calculating graph similarity, we utilize Levi graph transformation method to transform the input query graph into its equivalent Levi graph(Levi, 1942), which views predicate as a type of node in the graph.

In our situation, there is no gold label about the similarity between graphs. Therefore, we should train the graph retriever in an unsupervised way. With respect to the graphs, the variational graph auto-encoder (V-GAE) is a typical network to automatically learn the features of graph-structured data, which can be used as the measurement of the graph similarity. Specifically, V-GAE takes the adjacency matrix M and node features as input and tries to recover the graph adjacency matrix through the hidden layer embeddings H. We denote the node embedding as X. Specifically, V-GAE model calculates node embeddings via a GCN encoder,

$$H = GCN(X, M) \tag{4}$$

Then, the V-GAE samples the latent variables $\mathbf{h}_i \in H$ from a normal distribution,

$$q(H \mid X, M) = \prod_{i=1}^{N} q\left(\mathbf{h}_i \mid X, M\right), \tag{5}$$

The generative model is given by an inner product between latent variables,

$$p(M \mid H) = \prod_{i=1}^{N} \prod_{j=1}^{N} p\left(M_{ij} \mid \mathbf{h}_i, \mathbf{h}_j\right), \tag{6}$$

where M_{ij} are the elements of M. The training loss is given as,

$$\begin{aligned} \mathcal{L} = \mathbb{E}_{q(H|X,M)}[\log p(M \mid H)] \\ -\mathrm{KL}[q(H \mid X, M) \| p(Z)] \end{aligned} \tag{7}$$

where $\mathrm{KL}[q(\cdot)\|p(\cdot)]$ is the Kullback-Leibler divergence between $q(\cdot)$ and $p(\cdot)$. In the inference stage, the reconstructed adjacency matrix $\hat{M}$ is the inner product of the latent parameters H,

$$\hat{M} = \sigma\left(HH^T\right) \tag{8}$$

Suppose a query graph has q_n non-variable nodes, q_v variable nodes and q_p edges including predicate and functional edges. When generating questions, the variable nodes provide some semantic structure information in the questions but do not explicitly appear in the questions. However, non-variable nodes, the predicate and functional edges will transform into corresponding tokens, which will be generated in the appropriate position in the question. After Levi graph transformation, q_v is

always more than q_p and q_v, which may introduce noise during question generation. Moreover, query graph is our input, the predicate information and functional edges are critical for our study. Therefore, we design a weighting strategy that enhances the predicate information and weakens the influence of variable nodes.

After VGAE model training to convergence, we get the node representation from H, including node vector $\mathbf{h}_n$ and edge vector $\mathbf{h}_e$. We utilize average pooling to aggregate information for edge and nodes separately,

$$\begin{aligned} H_N &= \text{avgpool}\left(\mathbf{h}_n\right), \\ H_E &= \text{avgpool}\left(\mathbf{h}_e\right) \end{aligned} \tag{9}$$

We calculate their weight to get the graph embedding according to the number of non-variable nodes and edges.

$$H_G = \frac{q_n}{q_n + q_p} H_N + \frac{q_p}{q_n + q_p} H_E \tag{10}$$

Then we use graph embedding to retrieve the most similar samples through cosine similarity function.

4 Experiments

We conduct experiments on two widely-used benchmark datasets, i.e., WebQSP (Yih et al., 2016), and ComplexWebQuestions (CWQ) (Talmor and Berant, 2018). The baselines that we compare with are the state-of-the-art models over the adopted datasets.

4.1 Datasets and Preprocessing

The two adopted datasets are all from the general domain and based on Freebase (Bollacker et al., 2008). Specifically, WebQSP consists of 4,737 question-answer pairs. All the questions are collected through Google Suggest API, and the answers are fetched from Freebase with the help of Amazon Mechanical Turk. CWQ modified the SPARQLs in WebQSP by including more constraints, and then generated natural language questions with the help of templates and Amazon Mechanic Turk. It contains 34,689 questions in total. For each dataset, we randomly select 80% of the examples for training, 10% for validation, and 10% for testing.

4.2 Baseline Methods

We have several baseline methods, including the current state-of-the-art model over the two benchmark datasets. **L2A**(Du et al., 2017), 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), 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), a Transformer-based model for automatic generation of multi-hop questions over knowledge bases. **BiGraph2seq**(Chen et al., 2020), a graph-to-sequence model which leverages Bi-GNN as the graph encoder to encode the KB subgraphs, and enhance the RNN decoder with copying mechanism. **DCGCN**, our question generator which applys DCGCN as graph encoder and LSTM as decoder. We denote this model by DCGCN. **DCGCN+ROS**, We leverage DCGCN as basic model and adopt another classical strategy to solve the data imbalance in question type, i.e., Random Over Sampling (ROS).

4.3 Results and Discussion

Following previous KBQG works, we measure the method 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). Besides these automatic metrics, we conduct manual evaluations on 100 randomly chosen questions from the test set of datasets. We pair the questions generated by our model and state-of-the-art model in the test set, and scramble them. Two human annotators are asked to judge which is better in pairs from three aspects: naturalness, correctness, and semantic.

Table 1 shows the results of Meta-CQG and the adopted baselines on automatic metrics. We also conduct experiments with traditional setting, which will be discussed in Section 4.4. As is reported, Meta-CQG outperforms all the baselines on the three benchmark datasets. Specifically, Meta-CQG improves the BLEU-4 score by 4.16% on CWQ, 5.81% on WebQSP. Meanwhile, Meta-CQG exceeds the baselines by a larger margin on METEOR and ROUGE-L too.

Compared to the Seq-to-Seq model (Du et al., 2017; Elsahar et al., 2018; Kumar et al., 2019), we can see the advantages of GNN-based encoders for modeling query graphs, since the RNN-based

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
Meta-CQG	**30.17**	**32.01**	**56.58**	**33.67**	**33.08**	**65.99**
w/o Graph Retriever	27.66	29.68	53.88	29.87	31.45	64.08
w/o Weighting Strategy	29.13	31.26	55.68	32.53	32.10	64.88

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

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	33.88	34.03	62.41
	Multi-hop(>20%)	27.60	29.76	63.01	28.86	31.05	64.08	32.97	32.77	64.95
	Type Constraint(<10%)	26.53	28.76	60.23	28.27	30.19	62.76	32.60	32.86	64.59
	Ordinal(<5%)	23.43	27.01	53.99	24.73	27.53	57.07	29.91	31.50	64.09
Query Structure	Chain-Style(>75%)	29.38	31.23	63.65	31.45	32.96	64.45	34.23	34.22	66.58
	Tree-Style(<20%)	23.53	23.46	58.17	25.77	26.08	59.39	33.21	32.75	65.22
	Ring-Style(<5%)	6.78	17.01	48.65	9.48	20.57	50.78	23.23	30.79	56.79
Query Relation	Notable Types(>30%)	28.77	30.94	60.17	31.27	32.48	62.59	34.14	35.40	65.11
	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
Entity Class	Country(>30%)	28.66	31.27	63.48	30.23	32.83	64.09	34.22	33.79	67.06
	Book(<10%)	13.27	18.86	42.76	15.33	21.32	45.36	25.02	29.36	60.43
	Island Group(<5%)	5.77	15.85	32.87	9.89	17.84	38.66	22.18	27.66	54.79

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

model and the transformer-based model ignores the explicit graph structure of query graphs. Thus, Meta-CQG and BiGraph2seq outperform a large margin on both benchmarks. Instead of Bi-GNN leveraged in BiGraph2seq, we employ DC-GCN to capture the non-local interactions between the nodes in the graphs. Moreover, Meta-CQG overcomes the imbalanced problem. Hence our model outperforms than BiGraph2Seq significantly.

We conduct experiment to verify the effectiveness of our strategy to overcome date imbalance. As shown in Table 1, We achieve the balance between majority classes and minority classes through randomly copying some samples of minority classes, i.e., **DCGCN+ROS**. The results indicate the ability of our meta-learning approach to deal with the data imbalance.

As mentioned above, complex questions are imbalanced in four dimensions. We divide the questions in the WebQSP according to these dimensions and obtain the experimental results of BiGraph2Seq, DCGCN, and Meta-CQG. As there are too many categories in some dimensions, we sample one category from the majority class and two categories from the minority class for each dimension, as shown in Table 2.

As can be seen, Meta-CQG also achieves the best performance in all four dimensions. This demonstrates the effectiveness of our method for task-specific knowledge. In Table 2, We can see that

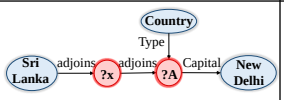
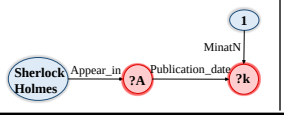
Query Graph	Question Type	Gold	BiGraph2Seq	DCGCN	Meta-CQG
	Type Constraint	What country bordering Sri Lanka has the capital new Delhi?	What country has the capital new Delhi is in Sri Lanka ?	Which country in Sri Lanka has the capital new Delhi?	Which country bordering Sri Lanka has the capital new Delhi?
	Ordinal Constraint	What is the first book Sherlock Holmes appeared in?	What is the book Sherlock Holmes written?	What is the first book Sherlock Holmes written?	What is the first book Sherlock Holmes appeared in?

Figure 3: Case study on different models for generation.

Meta-CQG delivers the best performance in the Type Constraint and Ordinal Constraint category, which account for less than 10% of the training set. It validates the outstanding learning ability of Meta-CQG for the minority class.

As shown in Table 2, DCGCN , the model that trained to fit all samples, performs worse in all categories than Meta-CQG. One possible reason is that it is difficult for the one-size-fits-all model to find the corresponding weight as the examples vary greatly. Besides, the imbalanced distribution also results in degradation of the model performance. Meta-CQG is designed to train a unique model for each sample, which can select appropriate weight to generate various questions. This demonstrates the adaptability of our model. which can quickly adapt to new samples by leveraging the similar-sample knowledge.

In all four dimensions, compared with other categories, our model has more significant improvement in the last two categories. As they account for less in the training data, the baseline model and DCGCN pay more attention to the majority class and neglect the minority class. When the proportion of category in the data decreases, the improvement effect of Meta-CQG is gradually significant. In Table 2, from the question type dimension, we also observe that the least effect of improvement is the largest proportion type, i.e., Single-hop questions . For each sample in Single-hop questions, there may be much more related samples than we set. This results in insufficient knowledge of similar samples for model learning.

In addition to automatic metrics, we also conduct a manual evaluation between Meta-CQG and the current state-of-the-art BiGraph2seq. Results are shown in Table 3. Our approach is preferred as it has more winning instances than losing instances on all two datasets. The results indicate that our model improves the quality of questions from three dimensions, i.e., naturalness, correctness, and semantic.

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

4.4 Ablation study

To have a deep insight into the design of Meta-CQG, we perform ablation studies where we remove the query graph retriever and weighting strategy, as shown in Table 1.

Graph Retriever. To evaluate the effectiveness of graph retriever, we remove it and randomly select samples for each training sample. As presented in Table 1, the results show that the performance of the model has deteriorated. This indicates that random select samples cannot provide task-specific knowledge for the training sample, and they may introduce noise during training.

Weighting Strategy. We evaluate Meta-CQG without weighting strategy on the two datasets. The results shown in Table 1 demonstrate that the strategy can improve the overall performance, as it is designed to enhance the information of predicates in the query graph .

4.5 Comparison with different sampling strategies

In this section, we design different sampling strategies to verify the effectiveness of our graph retriever. First, we devise different retrievers according to the four dimensions we mentioned above. For each dimension, the retriever selects a few samples that belong to the same class of the sample to

be generated. The results shown in Table 4 demonstrate the effectiveness of our model and verify that our model is able to comprehensively consider the imbalance in all dimensions.

Strategies	B-4.	ME.	R-L.
Question Type	31.07	32.87	64.97
Query Relation	31.22	32.39	64.43
Query Structure	30.09	31.53	64.27
Entity Class	30.48	31.98	64.42
Our Model	**33.67**	**33.08**	**65.99**

Table 4: Experimental results of different sampling strategies.

4.6 Case Study

As presented in Figure 3, we conduct a case study on generated questions. We select questions from different types to verify the effectiveness of our overall framework. For example, when generating the Ordinal Constraint question in Table, the question generated by BiGraph2seq was not smooth due to the lack of modeling non-local interactions. And DCGCN was puzzled about what token should be used to correspond to the predicate in the query graph. We can see that DCGCN failed to distinguish the two predicates for two tokens and thus generate inconsistent questions. As illustrated in Figure 3, our model can generate complex questions of high quality.

5 Related Work

Question generation over knowledge bases has been developed for a long time. Some methods (Berant and Liang, 2014) reconstructs the question text from a candidate structured query, and compare it with the original question to score the candidate query. The reconstruction is based on predefined templates. Some methods (Jia and Liang, 2016; Kočiskỳ et al., 2016; Hu et al., 2019; Cao et al., 2019) leverage the generated questions to train question answering models in a dual learning or semi-supervised learning framework. Recently, many works focus on question generation instead of augmentation for question answering. These works mainly adopt encoder-decoder models, and focus on enriching the input information. In (Serban et al., 2016), recurrent neural networks are first introduced for generating natural language questions from KB facts. In (Indurthi et al., 2017), questions are generated from an RNN based model with corresponding triples and entity types. 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, paired with a part-of-speech copy action mechanism to generate questions. 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 KBQG model. However, these methods only focus on generating one-hop or multi-hop questions from chain-like KB subgraph. The employed RNN-based models cannot handle graph-structured data. Recently (Kumar et al., 2019) proposes a model for generating complex multi-hop and difficulty-controllable questions over knowledge bases, and (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. To the best of our knowledge, we are the first to deal with the data imbalance in the complex question generation.

6 Conclusion

In this paper, we focus on the task of complex question generation over knowledge base. We propose a simple yet effective 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 related training samples. Specially, we design a self supervised graph retriever to flexibly retrieve most related samples. Besides, we propose a question generator, which leverages DCGCN to encode the queries and LSTM to decode the question . We evaluate the effectiveness of the proposed framework Meta-CQG on two widely-used benchmark datasets, and it outperforms all the baselines. In future work, we plan to explore the way of controlling the question complexity during generation.

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*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425.

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*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Ruisheng Cao, Su Zhu, Chen Liu, Jieyu Li, and Kai Yu. 2019. Semantic parsing with dual learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 51–64.

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*.

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.

Sen Hu, Lei Zou, and Zhanxing Zhu. 2019. How question generation can help question answering over knowledge base. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 80–92. Springer.

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.

Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Tomáš Kočiskỳ, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1078–1087.

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.

Eric Prudhommeaux. 2008. Sparql query language for rdf. *http://www. w3. org/TR/rdf-sparql-query/*.

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.

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*.

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*.

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.