Expanding, Retrieving and Infilling: Diversifying Cross-Domain Question Generation with Flexible Templates

Xiaojing Yu
Texas A&M University
vicky_yu@tamu.edu

Anxiao (Andrew) Jiang
Texas A&M University
ajiang@cse.tamu.edu

Abstract

Sequence-to-sequence based models have recently shown promising results in generating high-quality questions. However, these models are also known to have main drawbacks such as lack of diversity and bad sentence structures. In this paper, we focus on question generation over SQL database and propose a novel framework by *expanding*, *retrieving*, and *infilling* that first incorporates flexible templates with a neural-based model to generate diverse expressions of questions with guidance of sentence structure. Furthermore, a new activation/deactivation mechanism is proposed for template-based sequence-to-sequence generation, which learns to discriminate template patterns and content patterns, thus further improves generation quality. We conduct experiments on two large-scale cross-domain datasets. The experiments show that the superiority of our question generation method in producing more diverse questions while maintaining high quality and consistency under both automatic evaluation and human evaluation.

1 Introduction

With a growing demand for natural language interfaces for databases, automatic question generation from structured query language(SQL) query has been of special interest (Xu et al., 2018). Recently, diversity-aware question generation has shown its effectiveness in improving down-stream applications such as semantic parsing and question answering tasks (Guo et al., 2018; Sultan et al., 2020). Although neural sequence-to-sequence based generation has been a dominant approach and is able to produce a meaningful description for SQL queries, existing methods still suffer from the lack of diversity as well as bad sentence structures (Gao et al., 2019).

In the neural-based approaches, conventional ways of generating diverse sentences focus on approximate decoding techniques such as beam search (Li et al., 2016a,b; Iyyer et al., 2018) and temperature sweep (Caccia et al., 2018). Those decoding strategies generate diverse samples while sacrificing the quality of sentences. Variational auto-encoders (VAEs) have been used to generate various sentences by applying additional information as latent variables (Hu et al., 2017; Guo et al., 2018; Chen et al., 2019; Shao et al., 2019; Ye et al., 2020). However, implicit latent representation provides limited controllability over sentence structure and can be difficult to adapt to a new domain. Paraphrase (Fader et al., 2013; Berant and Liang, 2014) and syntactic-based methods (Dhole and Manning, 2020) have also been studied. However, learning a paraphrasing model relies on a large number of domain-specific paraphrase pairs, which is difficult to obtain for target databases. Besides, syntactic-based approaches apply syntactic parsers or semantic rules to the natural language utterance, thus are not applicable to SQL-to-question generation.

In the rule-based generation systems, the templates work as essential prior knowledge that contains the structural information of sentences (Wang et al., 2015; Song and Zhao, 2016; Krishna and Iyyer, 2019). This ensures the generation contains fewer grammatical errors and performs better with extractive metrics (Wiseman et al., 2017; Puzikov and Gurevych, 2018). However, their template formats are mostly strict and the valid content for each chunk should be pre-defined, which makes large set of templates difficult to obtain.

In this paper, we propose a novel method that incorporates template-based generation with a neural sequence-to-sequence model for diversity-aware question generation. Instead of applying strict templates, we use flexible templates that can be collected efficiently with less expense. These flexible templates provide high-level guidance of sentence structure while also enable sufficient flexibility for a neural-based model to fill chunks with content de-

Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, pages 3202–3212
April 19 - 23, 2021.

tails. We present our method as a three-stage framework including *expanding*, *retrieving*, and *infilling*. In the *expanding* stage, we take advantage of existing large-scale cross-domain text-to-SQL datasets to extract and collect the flexible template set automatically. In the retrieving stage, given a SQL query, the best templates are retrieved from the collected template set by measuring the semantic distance of SQL and templates in a joint template-SQL semantic space. In the filling stage, we treat each template as a masked sequence and explicitly force the generator to learn question generation with the constraint of the template. In order to help the generator to discriminate template patterns and content patterns, a unique activate/deactivation mechanism is designed for the generator to learn when to switch between template-copying state and content-filling state.

We conduct experiments on two large-scale cross-domain text-to-SQL datasets. Compared to existing approaches, our method achieves the best diversity result for both datasets with both automatic evaluation and human evaluation, while maintaining competitive quality and high consistency with SQL queries. We further demonstrate that the designed modules each contribute to a performance gain through an ablation study.

2 Related Work

2.1 Diverse Text Generation

In order to generate diverse expressions automatically, paraphrase-based methods (Qian et al., 2019; Fader et al., 2013; Berant and Liang, 2014; Dong et al., 2017; Su and Yan, 2017) have been studied. Wang et al. (2015) proposes to iteratively expand the template set and lexicons given a small number of template seeds and a large paraphrase corpus. Syntactic-based generation (Iyyer et al., 2018; Dhole and Manning, 2020) processes the given text with natural language processing techniques to produce high-quality and diverse sentences with pre-defined templates. However, the methods are not designed to deal with SQL queries and noisy table content. In recent years, neural network-based models have been widely used in text generation (Pan et al., 2019). Many studies attempt to diversify text generation by tuning latent variables of different properties, such as topic, style, and content (Fang et al., 2019; Ficler and Goldberg, 2017; Shen et al., 2019), while our method focuses on explicitly changing the sentence structure. In exemplar-based systems (Cao et al., 2018; Peng et al., 2019; Chen et al., 2019), the exemplar works as a soft constraint to guide the sequence-to-sequence generation and realize controllable diverse generation. Wiseman et al. (2018) proposes to learn a hidden semi-Markov model decoder for template-based generation for knowledge records. Most existing work requires either paraphrase pairs of the same input, reference sentences of similar content, or work effectively only in a single domain. Unlike existing works, our method only takes advantage of the large-scale cross-domain SQL-to-text datasets to collect a large number of templates. We extract templates from the datasets directly to maintain the quality of the templates. In order to find proper templates for a given SQL query, we learn a joint semantic space by instance learning and retrieve the best templates with closest semantic distance.

2.2 Question Generation

The question generation task relates to many applications such as question generation over knowledge base records (Wang et al., 2015), data-to-text generation (Wiseman et al., 2017) and question generation for question answering(QA) systems (Tang et al., 2017; Sun et al., 2019). SQL-to-question task differs from the other tasks in that SQL queries typically include new entities over different databases, which makes cross-domain generation a significant challenge. Xu et al. (2018) explores the graph-structured information in a SQL query and proposes a graph-to-sequence approach for generation. Guo et al. (2018) proposes to apply a copy mechanism and latent variables to map low-frequency entities from SQL queries to questions and generate diverse questions in an uncontrolled way. Existing SQL-to-question approaches aim at generating high-quality questions, while the diversity of the generation is less explored. In this work, we focus on generating diversified questions with the guidance of templates from cross-domain datasets.

3 Problem Formulation

Given a SQL query as the input sequence, question generation over database aims to generate a natural language question as an output sequence that accurately reflects the same meaning as the given SQL query. In this work, we generate the question by introducing an intermediate template in the generation process. Therefore, by applying

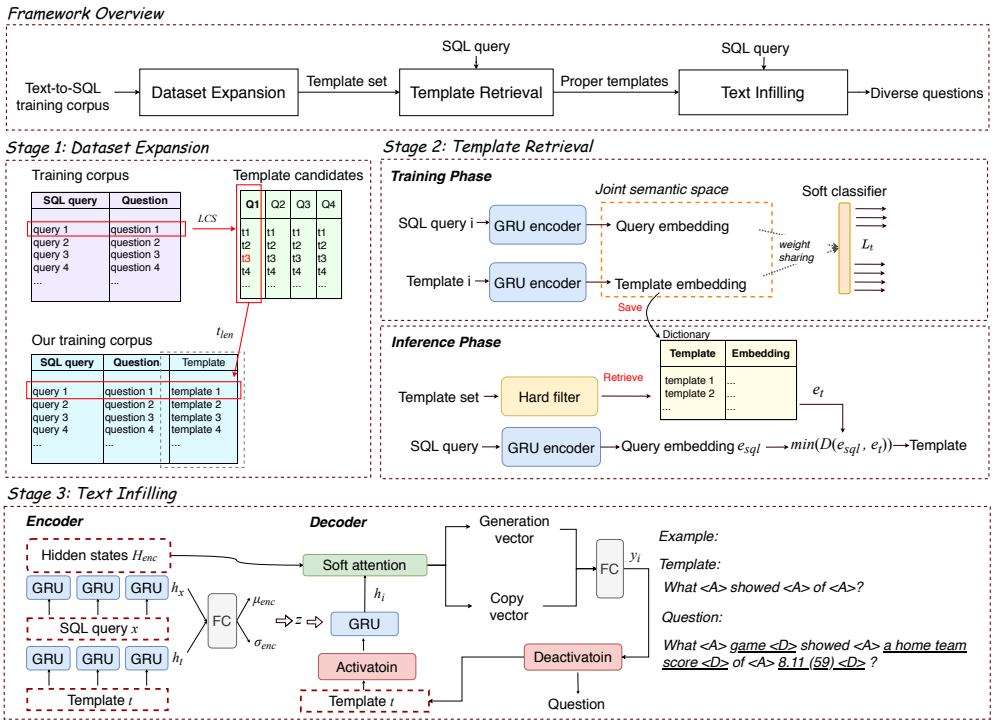

Figure 1: Architecture of our Framework.

different templates, we can generate diverse expressions of questions. Let $x = [x_1, x_2, ..., x_{|x|}]$ denote the given SQL query, $y = [y_1, y_2, ..., y_{|y|}]$ denote the gold-standard question, and $t = [t_1, t_2, ..., t_{|t|}]$ denote the corresponding template. Given a neural-based system with a set of learnable parameters θ_t^* and θ_q^*, the two-stage objective in this work can be formulated as follows:

$$\theta_t^* = \arg\max_{\theta_t} P(t|x)$$

$$\theta_q^* = \arg\max_{\theta_q} P(y|x,t)$$

4 Methodology

In this section, we introduce our framework, which learns question generation from SQL queries with the guidance of various templates, so as to increase the diversity of generation.

4.1 Framework Overview

We illustrate the framework that models the generation process in Figure 1. In brief, it includes three main stages: *expanding*, *retrieving*, and *infilling*.

Dataset Expansion The purpose of this step is to acquire a training dataset consisting of triplets `<query, question, template>`. Previous methods usually require a large corpus containing paraphrased questions to learn template structures; meanwhile, for existing text-to-SQL datasets, only `<query, question>` pairs are provided instead. To tackle those challenges, we design the longest common subsequence (**LCS**) based algorithm to automatically extract templates for each `<query, question>` pair without requiring the paraphrase pairs. Details of the algorithm are introduced in Section 4.2.

Template Retrieval After obtaining the *expanded training set*, all templates are gathered to form a large *template set* to generate diverse questions. To improve the quality and rationality of the generation, it is essential to retrieve suitable templates for it. A proper template should be consistent with the content information in a specific SQL query. For example, when the given query is `SELECT Population WHERE ( City = New York)`, the template `When is the <ph> of <ph> ?` should not be selected. For that purpose, we propose a soft classifier to learn a *joint SQL-template space*. In this way, the semantic distance can be measured between the two modalities, so that we can select proper templates by the closest semantic distance. Besides, since templates show higher inter-class similarity with the same SQL pattern, we also apply a *hard filter* to exclude the templates paired with different SQL patterns. Details of the soft classifier are introduced in Section 4.3.

Text Infilling With a encoder-decoder model

based on gated recurrent unit (GRU), we conduct question generation in the way of text infilling. The query x and template t are encoded into vectors separately by the bi-directional GRU encoder (Cho et al., 2014). Following the work of Gu et al. (2016) and Guo et al. (2018), we leverage the soft-attention and copy mechanism in the decoder construction. A Gaussian latent variable is adopted to capture the query and template variations. In the decoding process, a template t is fed into the decoder as a supervised signal to generate questions dynamically and sequentially. We propose an activation/deactivation mechanism to enforce the decoder to differentiate between the template patterns and the content patterns instead of randomly masking slots. In this way, the decoder can learn when to switch between the template reading and the content filling during generation. Details of the activation/deactivation mechanism are illustrated in Section 4.4.

4.2 Flexible Template as LCS

Consider a SQL query as a combination of a SQL pattern (i.e., the query with its content words removed) and the table information as follows:

```
SELECT COUNT( PLAYER ) WHERE
    (STATE = 'Texas')
```

where the underlined tokens are content from the table and `SELECT COUNT(<ph>) WHERE (<ph> = <ph>)` is a typical SQL pattern. Similarly, questions are composed of content words and template words. Since template words are often reused more frequently than content words, we design an effective method to extract most template words for each question in the training set. For the i-th question Q_i, we record its longest common sub-sequence (LCS) with each other question as a candidate template and construct a candidate-template dictionary d_i. The keys in d_i are the candidate sequences, and the corresponding values are the lengths of the sequences. After that, we choose the longest candidate from d_i as the template for the i-th question. The pseudo-code is described in Algorithm 1.

The candidate templates should satisfy the following rules:

- Each template should appear over 20 times.
- Each template should includes at least one of the keywords: *where, what, which, when, why, who, how, name, tell* .

When applying LCS, we mark the possible positions for content insertion between template words by placeholder `<ph>`, and format the templates like this instance: `Which <ph> has the largest <ph> ?` By ignoring the lengths of content word sequences, the templates become more flexible and can adapt to more scenarios.

Algorithm 1 LCS-based Template Extraction

Input: question set $Q = [q_1, q_2..., q_M]$; keyword set W
Output: Template set: T_{len}

for all $q_i \in Q$ **do**
  Initialize dictionary d_i
  for all $q_j \in Q$ **do**
    $c = LCS(q_i, q_j)$
    if $c \bigcap W \neq \emptyset$ **then**
      if $c \notin d_i.keys$ **then**
        $d_i[c] = 0$
      end if
      $d_i[c] += 1$
      Record position index for content
    end if
  end for
  for all $c \in d_i.keys$ **do**
    if $d_i[c] < 20$ **then**
      delete $d_i[c]$ from d_i
    end if
  end for
  $t_{len} = \arg\max_c(length(d_i.keys))$
  Update T_{len} by adding t_{len}
end for
return T_{len}

4.3 Learning Joint SQL-Template Space

As a sub-sequence of a question, a template should be close to the query in the semantic space if it is from the corresponding question, and be far away from the query if it is from an irrelevant question. Based on this intuition, we propose a soft classifier to learn a joint SQL-template space.

Soft Classifier. Classification models have been widely used in visual/textual applications. In a retrieval task, the classification model can learn feature embedding for the input, and its best matching counterpart can be found from a database by measuring the cosine distance between their embeddings. Inspired by this, we consider every `<SQL query, template>` pair in training set S as a distinct class, and learn the feature embeddings by instance-level classification. We represent each `<SQL query, template,`

class> triplet by $< x, t, n >$. Considering SQL queries and templates as objects constructed with different syntax, we encode them with two separate GRU encoders:

$$e_x = GRU_x(x)$$
$$e_t = GRU_t(t)$$

where e_x, e_t are query embedding and template embedding, respectively. In order to map SQL queries and templates to a joint feature space, we add a share-weight fully-connected layer W_s with softmax as the final classifier. The predicted probabilities over all instances are calculated as follows:

$$P(\cdot|x) = Softmax(W_s^T tanh(e_x))$$
$$P(\cdot|t) = Softmax(W_s^T tanh(e_t))$$

We jointly train the encoder and the classification layer with the following loss function:

$$\mathcal{L}_t = -\sum_{(x,t,n)\in S} (logP(n|x) + logP(n|t))$$

Inference. To enable efficient retrieval of templates, we store the template embeddings in a dictionary. In the inference phase, we can feed any SQL query into GRU_x to produce the query embedding, then remove the improper templates by a hard filter. We calculate the cosine distances between the query and the remaining templates, and sort them in descending order. The top-k nearest templates are selected for question generation.

$$\mathcal{D}(e_x, e_t) = \frac{e_x}{\|e_x\|_2} \times \frac{e_t}{\|e_t\|_2}$$

Avoid Overfitting. Since each class includes only one instance, we cannot use the classification loss of the validation set V to detect overfitting. Instead, we detect the overfitting by computing the average rank error R for the validation set:

$$R = \frac{1}{|V|^2} \sum_{i=1}^{|V|} (|r_i^a - r_i^p|)$$

where r^p and r^a are the predicted rank and actual rank, respectively. We stop training when R keeps increasing.

4.4 Decoding with A/D mechanism

The key idea of the proposed decoding method is that the generation process can be decomposed into a series of sub-generation tasks that are spaced by tokens in the template. During each sub-generation, the model can generate tokens of variable lengths. Since the decoder generates text word-by-word, it should determine where to switch between a content-filling state and a template-copying state, namely the **activation** or **deactivation** state, respectively. To achieve this, we require the decoder to activate/deactivate (A/D) generation with special switch symbols <A> and <D>. Therefore, when the decoder generates a symbol <A>, it changes the template-copying state to the content-filling state; when the decoder generates a <D>, it terminates the content-filling state and switches to the template-copying state. We also set a maximum length for each sub-generation to avoid the generation of unbounded sequences. In practice, before we train the question generator, we rewrite the question and template as follows (as an instance):

Template: <BEG> Which <A> has the largest <A> ? <END>

Question: <BEG> Which <A> one <D> has the largest <A> population among U.S. cities <D> ? <END>

We apply a pointer p to point to the current template token. With a simple GRU decoder, the state s and the generated token $\widehat{y}_i$ at the i-th step can be determined as follows:

$$s_i = \begin{cases} 1, \ \widehat{y}_{i-1} =< A >, \\ 0, \ \widehat{y}_{i-1} =< D >, \\ s_{i-1}; otherwise. \end{cases}$$

$$\widehat{y}_i = \begin{cases} Softmax(GRU(\widehat{y}_{i-1}, h_{i-1}^{dec})), \ s_i = 1, \\ t'_p, \ s_i = 0. \end{cases}$$

where h_{i-1}^{dec} is the hidden state of the $(i-1)$-th step, and t'_p is the current token of the template-copy operation, which should be updated after each copy. $\widehat{y} = [\widehat{y}_1, \widehat{y}_2, ...\widehat{y}_{|y|}]$ is the generated sequence. y' and t' are the rewritten question and template[1]. We apply teacher forcing during training, and feed the rewritten question y' as input to the decoder. During inference, we feed the rewritten template t' as input to the decoder. The training objective for generator G_q can be factorize as follows:

$$\max_{\theta_t} E_{<x,y>\sim D}[P_{\theta_q}(y'|x, t')]$$
$$= \max_{\theta_t} E_{<x,y'>\sim D}[\prod_{i\in 1:N} P_{\theta_q}(y'_i|x, t', y'_{i-1})]$$

[1]Code implementation is available at https://github.com/xiaojingyu92/ERIQG

In order to further diversify the expression of content, we introduce a latent variable z to the model. z relies on both SQL query x and template t'. We make Q a posterior distribution of z given x, t'. Then the evidence lower bound(ELBO) loss for it is:

$$\mathcal{L}_q = -E_{z\sim Q}(\sum_{i=1:N} log(P_{\theta_q}(y_i|x, t', z, y_{1:i-1}))) + D_{KL}(Q(z|x, t')||P_{\theta_q}(z))$$

where $Q_q(z|x, t') \sim N(\mu, \sigma)$. We apply a re-parameterization trick, making $z \sim N(0, I)$ and μ, σ learnable deterministic functions.

Note that the function of the template and latent variable do not overlap with each other. A large number of templates ensures the diverse sentence structure in generated questions, while the latent variable produces various expression of the content in the slots. As a part of the sentence, the <A> and <D> symbols join the back-propagation computation for optimizing the decoder's parameters. They work as additional information that guides the decoder to discriminate templates and content patterns, and learn when to terminate infilling and switch to template-copying state at each slot.

5 Experiment

Dataset To validate our framework's capability in generating diverse and controllable questions, we conduct experiments on two large-scale cross-domain text-to-SQL datasets: WikiSQL (Zhong et al., 2017) and Spider (Yu et al., 2018). WikiSQL contains 80654 query-question pairs derived from 24241 different schemas, with both validation set and test set released. Spider contains 10181 query-question pairs in 138 domains, with the validation set published. We follow the provided split settings for training and testing.

Setup We first construct the template set from the training set using our expansion method. For each SQL query in the test set, we retrieve the $k-1$ most relevant templates from the template set to generate $k-1$ different questions. We also include one question with template <BEG> <A> <END> to evaluate the model's capability in generating a complete sentence. Thus k questions for each SQL query are provided for evaluation. For our experiments, we set k = 5.

Baselines We compare our model(**ERI**)'s performance to models based on the baseline approach **QG** (Guo et al., 2018) with different diverse-sentence generation strategies. (1) **Latent Variable(QGLV):** Guo et al. (2018) applies a sequence-to-sequance network with a copy mechanism for question generation from SQL. A latent variable is introduced to generate diverse questions. (2) **Temperature Sweep(TEMPS):** We apply temperature sweep (**ts**) (Caccia et al., 2018) for decoding in **QG**. (3) **Beam Search(BEAMS):** We further combine **QG** with beam search (Li et al., 2016b) to generate diverse questions. In practice, we set the beam width to 5 and obtain sentences with top-5 highest probabilities for comparison. A Boltzmann temperature parameter α is applied to modulate the entropy of the generator. In practice, we set α = 0.7 as suggested and obtain 5 generated sentences for each query.

Evaluation Metrics We adopt the following automatic metrics to measure the quality of generated questions. (1) **maxBLEU:** The max BLEU-4 score among 5 generated questions. (2) **Coverage:** (Shao et al., 2019) This metrics measures the average proportion of input query covered by the generated questions. (3) **ParseAcc:** We use neural semantic parsers SQLova (Hwang et al., 2019) and Global-GNN (Bogin et al., 2019) to parse WikiSQL and Spider respectively. The semantic parsers translate the generated question into SQL query and calculate the exact-match accuracy with the input SQL query as the ground-truth. A higher accuracy score means the generated questions are more natural and consistent with the given SQL queries.

We adopt the following automatic metrics to measure the diversity of generated questions. (1) **self-BLEU:** (Zhu et al., 2018) The average BLEU-4 of all pairs among the k questions). (2) **self-WER:** (Goyal and Durrett, 2020) The average word error rate of all pairs among the k questions. A lower self-BLEU score and a higher self-WER score indicate more diversity in the result. (3) **Distinct-4:** (Li et al., 2016a) It measures the ratio of distinct n-grams in generated questions.

5.1 Automatic Evaluation

The automatic evaluation results are reported in Table 1 and 2. Our model ERI outperforms all baseline models in terms of the three diversity metrics for both datasets, except the self-BLEU on Spider. This shows the effectiveness of ERI in improving the diversity of its generation. Note that TEMPS performs less favorably in terms of qual-

Models	Quality			Diversity		
	Coverage ↑	ParseAcc ↑	maxBLEU ↑	Self-BLEU ↓	Self-WER ↑	Distinct-4 ↑
QGLV	70.50	**73.89**	37.75	92.86	17.39	33.46
TEMPS	11.34	3.38	5.36	84.50	36.49	59.34
BEAMS	71.49	68.09	**42.17**	79.80	37.39	54.97
ERIT(ours.)	**72.44**	72.79	28.43	**56.30**	**67.00**	**78.73**
w/o lv	70.28	69.53	24.30	57.96	64.39	75.31

Table 1: Automatic evaluation for diverse question generation on WikiSQL. **w/o lv** refers to our model without incorporating the latent variable.

Models	Quality			Diversity		
	Coverage ↑	ParseAcc ↑	maxBLEU ↑	Self-BLEU ↓	Self-WER ↑	Distinct-4 ↑
QGLV	13.76	3.97	14.45	96.90	11.73	38.16
TEMPS	6.58	3.87	4.00	**59.25**	7.59	19.73
BEAMS	13.68	4.16	**15.68**	89.39	21.12	50.17
ERI(ours.)	**15.89**	**18.09**	14.42	67.41	**53.23**	**66.96**
w/o lv	12.67	16.70	12.62	62.25	55.58	64.51

Table 2: Automatic evaluation for diverse question generation on Spider. **w/o lv** refers to our model without incorporating the latent variable.

Models	Fluency ↑	Consistency ↑	Diversity
QGLV	**4.56**	**4.64**	1.63
TEMPS	1.13	1.31	1.81
BEAMS	4.16	4.25	2.34
ERI	**4.56**	3.68	**4.31**

Table 3: Human evaluation results.

ity and diversity, as the temperature parameter α is a sensitive factor for the generation and needs tuning further. The expanded template set provides various valid sentence structures for expressing the same question which significantly contributes to the diversity expressions. But it also decreases the word-level overlapping, which leads to a relatively low maxBLEU score on WikiSQL. However, BLEU score may not be a suitable measurement for diversity-aware generation (Su et al., 2020; Shao et al., 2019). As the BLEU calculates the overlapping of n-grams, it does not necessarily reflect the quality of template-based generation. We illustrate that by an example of our model in Table 4. Although the BLEU scores are low, the generated questions are fluent and consistent with the SQL query with diversified structures.

To further measure the question quality and consistency in the semantic parsing task, we calculate the ParseAcc score. Our model performs competitively with QGLV on WikiSQL and shows a substantial improvement on Spider. The ParseAcc scores with ground-truth questions are 81.60% and 65.96% on WikiSQL and Spider, respectively. Our model also outperforms the baselines in the coverage of field names and values in the SQL query, indicating that essential terms from input are learnt and translated to questions. We also show the result of our model without the latent variable. In this setting, diversity of generated questions solely depends on selected templates. Without the latent variable, the proposed framework still outperforms the baselines in diversity metrics while maintains a good quality, which also supports that template contributes the most to the diversity of generation.

5.2 Manual Evaluation

To evaluate the quality of the generation, we run a manual evaluation to measure the quality and diversity for 800 SQL-question pairs from WikiSQL test set, produced by baseline models and our model (with $k = 5$). Each rater gives a 5-point scale for each SQL-question pair regarding the (1) **Fluency:** grammatical correctness, (2) **Consistency:** the semantic alignment with the corresponding SQL query, and (3) **Diversity:** the diverse expression of the generated questions.

We employ Fleiss' Kappa for inter-rater reliability measurement. The Kappa scores are 0.77, 0.60, 0.75 for fluency, consistency, and diversity, respectively, which indicates a good agreement on the scores. The results are presented in Table 3. Our model outperforms the baseline models in diversity and fluency. It can achieve the best trade-off for the three measurements. Although QGLV and BEAMS show the best performance in generating

SQL query	SELECT COUNT (rd #) WHERE pick # < 5
Ground-truth	how many rounds exist for picks under 5 ?
Q1 (BLEU=2.79) :	**what is the number of** rd **when the** pick number **is less than** 5 **?**
Q2 (BLEU=11.73):	**how many** rounds **have a** pick # **less than** 5 **?**
Q3 (BLEU=2.79) :	**what is the total number of** rd **where the** pick **is less than** 5 **?**
Q4 (BLEU=2.02) :	**tell me the total number of** rd **for** pick **less than** 5
Q5 (BLEU=1.51) :	**for the** pick **less than** 5 **, what was the total number of** rd # **?**

Table 4: Examples of high-quality generation with low BLEU score. Template tokens are in bold font.

Models	BLEU↑	NIST↑	ROUGH↑	METEOR↑
QGLV	32.19	4.74	64.02	64.25
*Graph2Seq	38.97	-	-	-
ERI	**48.12**	**5.24**	**76.52**	**75.76**
w/o A/D	43.30	4.85	72.41	73.76
w T	31.42	4.18	63.44	63.94
w/o T	29.76	4.05	62.37	63.17

Table 5: Performance on different sub-module combinations on WikiSQL. *: the value is cited from Xu et al. (2018)

Models	BLEU↑	NIST↑	ROUGH↑	METEOR↑
QGLV	12.60	2.39	44.37	38.74
ERI	**21.30**	**3.11**	**53.83**	**51.04**
w/o A/D	19.02	2.93	52.45	50.41
w T	12.40	2.41	45.8	39.99
w/o T	13.36	2.35	45.05	40.27

Table 6: Performance on different sub-module combinations on Spider.

high-quality sentences, they tend to create questions of fixed structures with only minor changes in expressions. With our method, the templates provide substantial changes in sentences' structures, which validates the benefit of the proposed template-extraction method. Examples from our model and the baselines are given in Table 8.

5.3 Ablation Study

For the ablation study, we present experimental results to verify performance in two aspects: (1) whether the generator in our model benefits from learning with the activation/deactivation mechanism; (2) whether our model maintains the consistency by selecting suitable templates in the joint semantic space.

Evaluation on Generator To analyze the ability of generating high-quality questions from the given templates, we extract the templates from the test set and use the corresponding template to guide the question generation. We measure the generation quality by **BLEU**, **NIST**, **ROUGE**, and **METEOR**. In order to analyze the impact of various modules in our generator, we evaluate the following versions of our framework: (1) **ERI w/o T** model that does not use templates for encoding and decoding. (2) **ERI w T** model where templates are encoded but not used as the input for decoding. (3) **ERI w/o A/D** model that applies the generator with the decoding strategy similar to a seq2seq model in Zhu et al. (2019). It treats each slot as a segment, and predicts each segment as an independent sentence. (4) **ERI** model that has all designed modules, including the A/D mechanism. The results are presented in Table 5 and 6. Our model outperforms existing methods in four metrics. Using template information improves our model on both datasets. The segmented-based infilling method gains further improvement, which shows the effectiveness of providing templates as hard constraints. By introducing the A/D mechanism to decoding, our model has seen further boosts, which demonstrates that the A/D mechanism enhances the learning of generation from the templates.

SQL-Template Consistency To validate the im-

Models	WikiSQL Dev	WikiSQL Test	Spider Dev
Random	0.1	0.07	0.80
Hard Filter	16.7	13.4	30.8
Ours.	**21.1**	**14.0**	**32.6**

Table 7: MAP results of template retrieval methods.

SQL query	SELECT attendance WHERE week < 16 and date = bye
Ground-truth	what is the attendance for a week earlier than 16 , and a date of bye ?
ERI(ours.):	
Q1:	**what is** attendance , **when** week is **less than** 16 **, and when** date is bye ?
Q2:	**which** attendance **has a** week **smaller than** 16 **and a** date **of** bye ?
Q3:	**what is the** attendance **for the** week **earlier than** 16 **and is** dated bye ?
Q4:	attendance **before** week 16 **on what** bye **was the** date ?
Q5:	how many people attended the game before week 16 on bye ?
QGLV:	
Q1:	what was the attendance for the bye game before week 16 ?
Q2 (= **Q3**,**Q4**,**Q5:**)	what is the attendance for the bye game before week 16 ?
BEAMS:	
Q1:	what was the attendance for the bye week before week 16 ?
Q2:	which attendance has a week smaller than 16 and a date of bye ?
Q3:	which attendance has a week smaller than 16 , and an date of bye ?
Q4:	what is the attendance of the game with a week less than 16 and bye ?
Q5:	what is the attendance of the game with a week less than 16 and a bye date ?
TEMPS:	
Q1:	where week date
Q2:	where week date
Q3:	where week week week week bye and week
Q4:	where week and week date bye and attendance where week
Q5:	where week bye and week attendance

Table 8: Question generation example on WikiSQL.

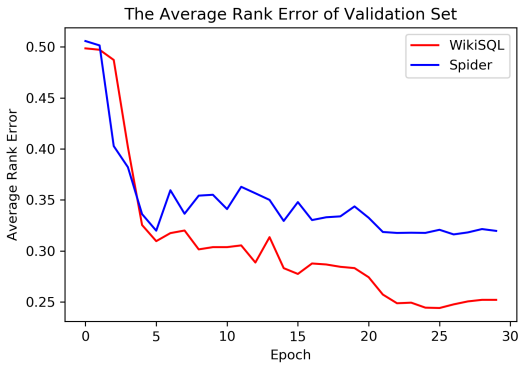

Figure 2: Average rank error in training phase.

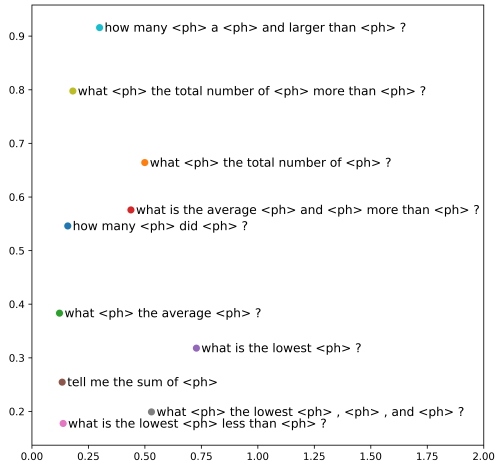

Figure 3: Visualization of the template features by tSNE.

pact of our template retrieval method, we show the average rank error of soft classifier during the training phase in Figure 2. For both datasets, the rank error decreases during training, which indicates the soft classifier can capture the semantic relation between SQL queries and templates. To observe if the soft classifier affects the performance by selecting the proper template, we also compare our 2-stage template retrieval to random strategies and hard filter in mean average precision (MAP). The result is presented in Table 7. Compared to hard filter, the soft classifier improves the MAP by 4.4%, which validates the effectiveness of the proposed template retrieval method.

Visualization of Joint Space To visualize the similarity of templates, we map feature samples to 2-dimensional space by t-Distributed Stochastic Neighbor Embedding(t-SNE) in Figure 3. The features from similar SQL-template pairs preserve closer distances, which shows the effectiveness of our instance-level classification in learning the semantic meaning in the joint feature space.

6 Conclusion

In this paper, we present a novel framework for question generation over SQL database to produce more diversified questions by manipulating the templates. We expand the template set from cross-domain SQL-to-text datasets, and retrieve proper templates from a template set by measuring the distance between the templates and the SQL query in a joint semantic space. We propose an activation/deactivation mechanism to make full use of templates to guide the question generation process. Experimental results have shown that the presented model can generate various questions while maintains their high quality. The model has also improved the matching between the templates and the content information of SQL queries.

References

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.

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019. Global reasoning over database structures for text-to-sql parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3650–3655.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2018. Language gans falling short. *arXiv preprint arXiv:1811.02549*.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. Controllable paraphrase generation with a syntactic exemplar. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5972–5984.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger

Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Kaustubh Dhole and Christopher D Manning. 2020. Syn-qg: Syntactic and shallow semantic rules for question generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 752–765.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 875–886.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618.

Le Fang, Chunyuan Li, Jianfeng Gao, Wen Dong, and Changyou Chen. 2019. Implicit deep latent variable models for text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3937–3947.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104.

Jun Gao, Wei Bi, Xiaojiang Liu, Junhui Li, Guodong Zhou, and Shuming Shi. 2019. A discrete cvae for response generation on short-text conversation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1898–1908.

Tanya Goyal and Greg Durrett. 2020. Neural syntactic preordering for controlled paraphrase generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 238–252.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640.

Daya Guo, Yibo Sun, Duyu Tang, Nan Duan, Jian Yin, Hong Chi, James Cao, Peng Chen, and Ming Zhou. 2018. Question generation from sql queries improves neural semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1597–1607.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1587–1596. JMLR. org.

Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885.

Kalpesh Krishna and Mohit Iyyer. 2019. Generating question-answer hierarchies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2321–2334.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*.

Liangming Pan, Wenqiang Lei, Tat-Seng Chua, and Min-Yen Kan. 2019. Recent advances in neural question generation. *arXiv preprint arXiv:1905.08949*.

Hao Peng, Ankur Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. 2019. Text generation with exemplar-based adaptive decoding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2555–2565.

Yevgeniy Puzikov and Iryna Gurevych. 2018. E2e nlg challenge: Neural models vs. templates. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 463–471.

Lihua Qian, Lin Qiu, Weinan Zhang, Xin Jiang, and Yong Yu. 2019. Exploring diverse expressions for paraphrase generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3164–3173.

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, et al. 2019. Long and diverse text generation with planning-based hierarchical variational model. In *Proceedings of the 2019 Conference on Empirical*

Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3248–3259.

Xiaoyu Shen, Jun Suzuki, Kentaro Inui, Hui Su, Dietrich Klakow, and Satoshi Sekine. 2019. Select and attend: Towards controllable content selection in text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 579–590.

Linfeng Song and Lin Zhao. 2016. Domain-specific question generation from a knowledge base. *arXiv preprint arXiv*, 1610.

Hui Su, Xiaoyu Shen, Sanqiang Zhao, Zhou Xiao, Pengwei Hu, Cheng Niu, Jie Zhou, et al. 2020. Diversifying dialogue generation with non-conversational text. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7087–7097.

Yu Su and Xifeng Yan. 2017. Cross-domain semantic parsing via paraphrasing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1235–1246.

Md Arafat Sultan, Shubham Chandel, Ramón Fernandez Astudillo, and Vittorio Castelli. 2020. On the importance of diversity in question generation for qa. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5651–5656.

Yibo Sun, Duyu Tang, Nan Duan, Shujie Liu, Zhao Yan, Ming Zhou, Yuanhua Lv, Wenpeng Yin, Xiaocheng Feng, Bing Qin, et al. 2019. Joint learning of question answering and question generation. *IEEE Transactions on Knowledge and Data Engineering*.

Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. Sql-to-text generation with graph-to-sequence model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 931–936.

Rong Ye, Wenxian Shi, Hao Zhou, Zhongyu Wei, and Lei Li. 2020. Variational template machine for data-to-text generation. *arXiv preprint arXiv:2002.01127*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Wanrong Zhu, Zhiting Hu, and Eric Xing. 2019. Text infilling. *arXiv preprint arXiv:1901.00158*.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100.