Few-shot Text-to-SQL Translation using Structure and Content Prompt Learning

ZIHUI GU, Renmin University of China, China
JU FAN*, Renmin University of China, China
NAN TANG, QCRI / HKUST (GZ), Qatar / China
LEI CAO, MIT CSAIL / University of Arizona, USA
BOWEN JIA, Renmin University of China, China
SAM MADDEN, MIT CSAIL, USA
XIAOYONG DU, Renmin University of China, China

A common problem with adopting Text-to-SQL translation in database systems is poor generalization. Specifically, when there is limited training data on new datasets, existing few-shot Text-to-SQL techniques, even with carefully designed textual prompts on pre-trained language models (PLMs), tend to be ineffective. In this paper, we present a divide-and-conquer framework to better support few-shot Text-to-SQL translation, which divides Text-to-SQL translation into two stages (or sub-tasks), such that each sub-task is simpler to be tackled. The first stage, called the *structure stage*, steers a PLM to generate an SQL structure (including SQL commands such as SELECT, FROM, WHERE and SQL operators such as "<", ">") with placeholders for missing identifiers. The second stage, called the *content stage*, guides a PLM to populate the placeholders in the generated SQL structure with concrete values (including SQL identifies such as table names, column names, and constant values). We propose a hybrid prompt strategy that combines learnable vectors and fixed vectors (*i.e.,* word embeddings of textual prompts), such that the hybrid prompt can learn contextual information to better guide PLMs for prediction in both stages. In addition, we design *keyword constrained decoding* to ensure the validity of generated SQL structures, and *structure guided decoding* to guarantee the model to fill correct content. Extensive experiments, by comparing with ten state-of-the-art Text-to-SQL solutions at the time of writing, show that SC-Prompt significantly outperforms them in the few-shot scenario. In particular, on the widely-adopted Spider dataset, given less than 500 labeled training examples (5% of the official training set), SC-Prompt outperforms the previous SOTA methods by around 5% on accuracy.

CCS Concepts: • **Theory of computation** → *Parsing*; • **Information systems** → *Relational database query languages*.

Additional Key Words and Phrases: text-to-SQL; pre-trained language model; prompt learning

ACM Reference Format:
Zihui Gu, Ju Fan, Nan Tang, Lei Cao, Bowen Jia, Sam Madden, and Xiaoyong Du. 2023. Few-shot Text-to-SQL Translation using Structure and Content Prompt Learning. *Proc. ACM Manag. Data* 1, 2, Article 147 (June 2023), 28 pages. https://doi.org/10.1145/3589292

*Ju Fan is the corresponding author.

Authors' addresses: Zihui Gu, Renmin University of China, China, guzh@ruc.edu.cn; Ju Fan, Renmin University of China, China, fanj@ruc.edu.cn; Nan Tang, QCRI / HKUST (GZ), Qatar / China, ntang@hbku.edu.qa; Lei Cao, MIT CSAIL / University of Arizona, USA, lcao@csail.mit.edu; Bowen Jia, Renmin University of China, China, bowenjia@ruc.edu.cn; Sam Madden, MIT CSAIL, USA, madden@csail.mit.edu; Xiaoyong Du, Renmin University of China, China, duyong@ruc.edu.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
2836-6573/2023/6-ART147 $15.00
https://doi.org/10.1145/3589292

1 INTRODUCTION

Text-to-SQL, which translates natural language (NL) questions to SQL queries, promises to enable sophisticated data querying to non-technical end users. Supporting NL questions over tables has many applications. For example, data visualization companies, such as Tableau and Power BI, use NL questions to assist the construction of dashboards, where the required structured queries over tables are relatively simple, *e.g.,* most queries do not need joins or nested loops. In addition, supporting Text-to-SQL translation has high demand from the database industry, *e.g.,* Oracle [15] and SalesForce [21].

The "Pre-train, Fine-tune" Paradigm. Language models, *e.g.,* GPT-3 [2] and T5 [18], are *pre-trained* on very large corpora to learn general knowledge. *Fine-tuning* adapts these PLMs to downstream tasks using task-specific objective functions and datasets, *e.g.,* question answering [32] and text summarization [41]. Recent studies, based on Text-to-SQL benchmarks such as Spider [39], show that the state-of-the-art (SOTA) performance on Text-to-SQL is achieved by fine-tuning large PLMs, such as Picard [24], which fine-tunes T5 [18] in a simple end-to-end fashion.

Few-shot Text-to-SQL. A common scenario for Text-to-SQL is that oftentimes, for a new dataset or domain, sufficient high-quality training data is not available and obtaining it requires expert knowledge, which is very expensive and labor-intensive to acquire. In this paper, we refer to the case of limited training data as *few-shot Text-to-SQL*. The "pre-train, fine-tune" paradigm tends to be ineffective in this few-shot setting for Text-to-SQL, because the PLMs have *poor generalization* given limited training data on new datasets.

One possible strategy to mitigate poor generalization is through prompting [14], where textual prompts are used to explicitly guide PLMs to better reason about the tasks, *e.g.,* "generate an SQL query" or "write a Java program". Unfortunately, for the few-shot Text-to-SQL scenario when there is not enough training data, traditional prompting techniques also cannot solve the problem, as we illustrate in the following example.

Example 1 (Prompting for Text-to-SQL). *Consider the database D in Figure 1(a) with two tables,* `highschooler` *and* `friend`. *Let N be an NL question that asks for high school students who do not have friends. As shown in the figure, the ground truth SQL query Q uses the EXCEPT operator to return the student IDs from the table* `highschooler` *that are not in the table* `friend`.

We fine-tuned the pre-trained T5-large [18] model with the textual prompt P in Figure 1(b) using 5% *of the training data of the database D from the Spider [39] Text-to-SQL benchmark, which is around* 300 *(NL, SQL) pairs. During testing, the fined-tuned model still outputs the incorrect SQL query Q', where the column name* `student_id` *should be* `id`. *After careful analysis, we find that the cause of these failures is a discrepancy between the pre-training tasks and our Text-to-SQL task, where the limited training data is not sufficient to adapt the PLM to new datasets.* □

A Divide-and-Conquer SC-Prompt Framework. To address this problem, we propose **S**tructure and **C**ontent **P**rompt learning (SC-Prompt), a novel *divide-and-conquer* framework that divides the Text-to-SQL task into two sub-tasks, such that each sub-task is simpler and more amenable to limited training data, compared with the end-to-end Text-to-SQL task. Specifically, we consider two stages (or sub-tasks) for Text-to-SQL translation:

(1) A **structure stage** that steers a PLM to generate an SQL structure with placeholders for missing contents, and
(2) A **content stage** that guides a PLM to populate the placeholders in the generated SQL structure with concrete values.

To better illustrate this idea, consider the following example.

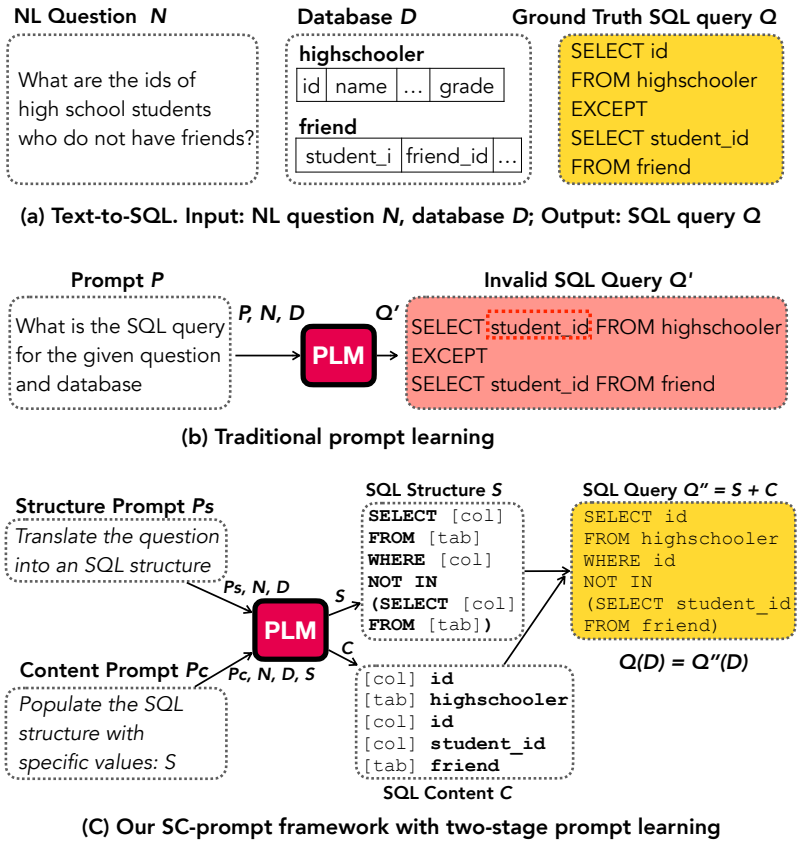


Fig. 1. Text-to-SQL with few-shot prompt learning.

Example 2 (SC-Prompt for Text-to-SQL). *Consider the example in Figure 1 (a) again. Figure 1 (c) depicts our two-stage strategy.*

(1) **Structure stage.** *In this stage, we use a structure prompt* $\mathbf{P}_S$ *to generate an SQL structure S containing only SQL commands* (e.g., *SELECT, FROM, WHERE) and operators* (e.g., *<, >*), *while leaving all contents to be placeholders* (e.g.,`[col]`, `[tab]`, *etc).*

(2) **Content stage.** *In this stage, we use a content prompt* $\mathbf{P}_C$, *the previously generated SQL structure S, together with the NL question N and the database D to guide a PLM to fill the content placeholders* (e.g., *the first* `[col]` *should be column "*`id`*", the first* `[tab]` *should be table "*`highshooler`*", and so on).*

Finally, we combine the SQL structure S and SQL content C to generate an SQL query Q′′. Compared with the ground truth query Q in Figure 1 (a), we note that SC-Prompt may predict a different but equivalent SQL structure (e.g., *EXCEPT vs. NOT IN), and correctly predicts the SQL query,* i.e., $Q(D) = Q''(D)$. □

Next, based on Example 2, we discuss why SC-Prompt is better than traditional end-to-end Text-to-SQL translation in the few-shot learning setting. The structure stage focuses only on predicting an SQL structure from an NL question, which is much easier than predicting a complete SQL query. For the content stage, the task is to fill the content placeholders, a *fill-in-the-blank* task, which is

well aligned with the self-supervised mask language modeling for pre-training many PLMs (*e.g.,* BERT [4] and T5 [18]).

Challenges and Solutions. Effectively realizing the proposed SC-Prompt framework raises two main technical challenges.

The first challenge is that manually designed *textual prompts* [2] are not flexible enough to effectively guide PLMs to solve different prediction problems in the two stages. An intuitive way to improve predictions is to provide more contextual information. To this end, we propose a *hybrid prompt* strategy that combines learnable vectors [13] and fixed vectors (*i.e.,* word embeddings of textual prompts). The learnable vectors are tuned during training and are used to provide contextual information to better guide PLMs for prediction in both stages.

The second challenge is how to improve the decoding process of the PLMs. A typical auto-regressive decoding process may not be effective, because large PLMs, such as T5 [18] used in our paper, usually have a large number of (sub-word) tokens, which would increase the possibilities of generating invalid SQL queries. To this end, we propose a *fine-grained constrained decoding* strategy to help prune the search space of both stages to make invalid SQL queries less likely. In particular, we design *keyword-constrained decoding* to ensure the validity of generated SQL structures, and *structure-guided decoding* to guarantee that the model fills the correct content.

Contributions. We summarize our contributions as follows.

(1) We study the problem of *few-shot* Text-to-SQL translation (Section 2), and introduce a *divide-and-conquer* framework SC-Prompt that divides the Text-to-SQL task into two simpler stages (sub-tasks), namely a structure stage and a content stage (Section 3).

(2) We propose two novel techniques to tackle the challenges of this problem: structure and content prompt construction (Section 4) and fine-grained constrained decoding (Section 5).

(3) We conduct extensive experiments on three Text-to-SQL benchmarks with different difficulty levels. Our experimental results show that SC-Prompt significantly outperforms the advanced Text-to-SQL solutions in the *few-shot* scenario (Section 7) [1].

2 FEW-SHOT TEXT-TO-SQL

This section first formalizes Text-to-SQL translation in Section 2.1, and then presents how to adopt PLMs to solve the problem (Sections 2.2 and 2.3). Next, we propose a decomposition strategy, which divides an SQL query into SQL structure and SQL content, to address the challenge of few-shot Text-to-SQL (Section 2.4).

2.1 Text-to-SQL

SQL Queries. We consider standard SQL queries in relational databases, which consist of the following elements[2].

- ***Commands***: These are SQL reserved keywords such as SELECT, WHERE, COUNT, etc.
- ***Operators***: These include SQL arithmetic operators (*e.g.,* "+", "−", "∗"), bit-wise operators (*e.g.,* "&", "|"), comparison operators (*e.g.,* ">", "<"), and logical operators (*e.g.,* "NOT IN").
- ***Identifiers***: These refer to database specific objects, such as tables, columns, etc (*e.g.,* column `student_id` in Figure 1).
- ***Constant Values***: These are fixed data values, which can be categorical, numerical and textual values. Note that the textual values would be enclosed in single quote marks.

[1] Code is available at https://github.com/ruc-datalab/SC-prompt

[2] https://en.wikipedia.org/wiki/SQL_reserved_words

Text-to-SQL. Let N be an NL question, D be a relational database containing n tables $\{T_1, T_2, \ldots, T_n\}$, and c_{ij} be the j-th column of the i-th table T_i. The problem of *Text-to-SQL* is to generate an SQL query Q based on the question N and the database D.

2.2 Transformer-based Encoder-Decoder Models

Transformer Models. Transformer-based pre-trained language models (PLMs) [12, 18, 28] have achieved tremendous success for textual data. Pre-trained on a massive amount of text corpora, PLMs like BERT [4], BART [12], T5 [18], GPT-3 [2], etc. can effectively learn rich knowledge and have been widely used in a variety of applications.

Encoder-Decoder Models. Transformer-based encoder-decoder models are the SOTA for many generative tasks which generate an output object O given the input object I, such as sentiment analysis, machine translation, and Text-to-SQL. In general, such a model consists of an **encoder** and a **decoder**:

The **encoder** $f_{\mathsf{encode}}(\cdot)$ converts the input I into a high-dimensional vector $\mathbf{x}$, *i.e.,* $\mathbf{x} = f_{\mathsf{encode}}(I)$.

The **decoder** $f_{\mathsf{decode}}(\cdot)$ decodes the hidden vector $\mathbf{x}$ into an output sequence O in an auto-regressive way. Auto-regressive generation says that the probability distribution of a sequence $s = \{w_1, w_2, \ldots, w_n\}$ can be decomposed into the product of conditional next word distributions:

$$P_\Theta(s|I) = \prod_{i=1}^{n} P(w_i|w_{1:i-1}; I) \tag{1}$$

where Θ denotes the parameters of the decoder, I denotes the input of the encoder, and n denotes the length of the sentence s. Therefore, the decoder aims to find the sentence with the highest probability as the output. There are many decoding methods that can be used to achieve this goal, such as greedy search, beam search, etc.

2.3 Text-to-SQL with Encoder-Decoder Models

Transformer-based encoder-decoder models have been widely used to support Text-to-SQL, and representative models include NQG-T5 [26], PICARD [24], UnifiedSKG [34], etc. For Text-to-SQL, the input I to the encoder is an NL question N and a database schema D, and the output O from the decoder is an SQL query Q.

Note that, we also abuse the term D as a database, not only a database schema, when it is evaluated by an SQL query, *e.g.,* $Q(D)$ and $Q''(D)$ in Figure 1(c).

Text-to-SQL Fine-tuning. Task-specific *fine-tuning* effectively adapts PLMs to downstream tasks using task-specific objective functions and datasets. For supporting Text-to-SQL translation, these PLMs are typically fine-tuned with Text-to-SQL benchmarks such as Spider [39] and CoSQL [38].

2.4 SQL Decomposition

As discussed earlier, when there is limited training data (*i.e.,* few-shot), it is hard to fine-tune PLMs to directly output correct SQL queries. A viable solution is to decompose SQL queries into smaller components, as these decomposed components are easier to learn. In this paper, we propose to decompose each SQL query into two parts: SQL structure and SQL content, as defined below.

SQL Structure. SQL structure includes the SQL elements with only *SQL commands* and *SQL operators*. In addition, we use placeholders within an SQL structure, such as [tab], [col], and [val], to represent to-be-filled identifiers and constant values.

For example, in Figure 1(c), if we replace the identifiers in the gold query with placeholders, it will be converted into an SQL structure representation: "`SELECT [col] FROM [tab] EXCEPT SELECT [col] FROM [tab]`".

SQL Content. SQL content includes *SQL identifiers* and *constant values*. More specifically, we concatenate placeholders and corresponding identifiers or constant values as the representation of SQL content. For example, the representation of SQL content in Figure 1(c) is: "`[col] id [tab] highschooler [col] id [col] student_id [tab] friend`".

SQL Query = SQL Structure + SQL Content. The SQL structure and the SQL content corresponding to one NL query can be trivially combined as an SQL query (see Figure 1(c)).

3 SOLUTION OVERVIEW

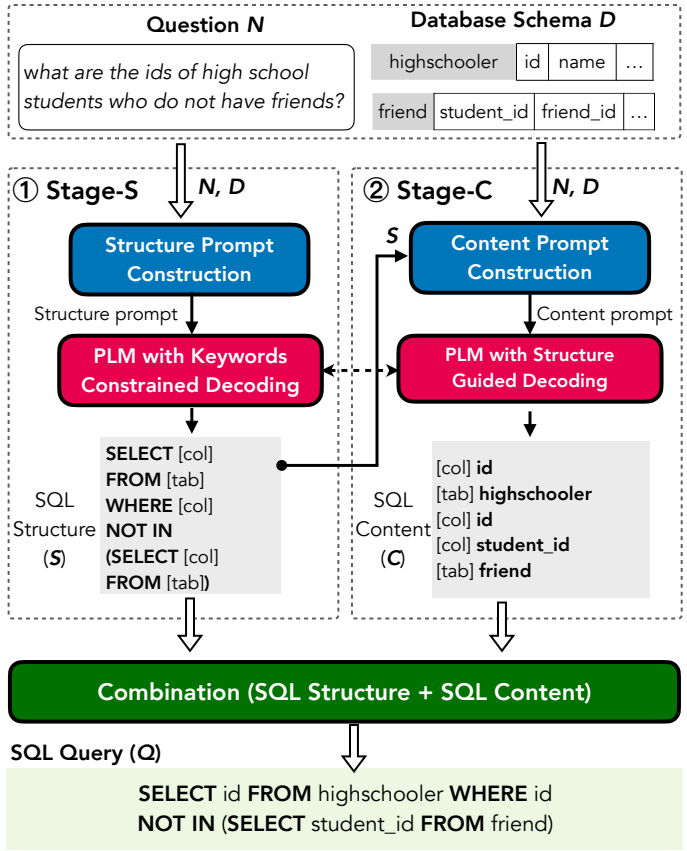


Fig. 2. An overview of SC-Prompt. First, the structure stage steers the PLM to generate an SQL structure S. Then, the content stage guides the PLM to generate SQL content C based on the structure S. Finally, the structure S and content C is combined to generate an SQL query Q.

Figure 2 shows an overview of our SC-Prompt framework for supporting few-shot Text-to-SQL in two stages.

(1) The structure stage (**Stage-S**) steers the PLM to generate an SQL structure S, based on the NL question N, the database schema D, and a structure prompt.
(2) The content stage (**Stage-C**) guides the PLM to populate the placeholders in the generated SQL structure with specific values, resulting an SQL content C.

We then combine the predicted SQL structure S and SQL content C to generate an SQL query Q as described in Section 2.4.

There are two core modules in our SC-Prompt framework, as shown in Figure 2: *Prompt Construction* (blue rectangles) and *Constrained Decoding* (red rectangles), which we describe next.

Prompt Construction. Intuitively, the aim of prompt construction is to design a function $f_{\texttt{prompt}}$ to transforms the NL question N and the database schema D into an *instruction* for a PLM, which we denote as prompt $\mathbf{P}$ where $\mathbf{P} = f_{\texttt{prompt}}(N, D)$.

A straightforward implementation for $f_{\texttt{prompt}}$ is to use a *natural language template*, with input and output placeholders, *e.g.,*

$$\texttt{Translate NL to SQL, } [N] \ [D], \ \underline{[Z]}$$

where $[N]$ and $[D]$ are input placeholders, which can be populated with each input instance (N, D), and $\underline{[Z]}$ is the output blank, which will be predicted by the PLM. Obviously, this prediction task can be formulated as the masked language modeling (MLM) problem, which is the pre-training objective of most PLMs.

However, it is non-trivial to design an effective prompting function $f_{\texttt{prompt}}$ that will result in the PLM making accurate predictions predictions. In SC-Prompt, we divide the procedure into two stages: Stage-S and Stage-C, shown in Figure 2. Our approach is to design different prompts for different stages, although, even then, it is difficult to *manually* design appropriate prompts that effectively guide the PLM in each stage.

To address these challenges, we introduce a novel *hybrid prompt* strategy that combines learnable vectors for learning more contextual information and fixed vectors (*i.e.,* word embeddings of manually designed textual prompts) for capturing domain knowledge. This hybrid prompt, in the form of a high-dimensional vector, is fed to a PLM. More details of prompt construction for both stages are given in Section 4.

Constrained Decoding. Given a constructed prompt $\mathbf{P}$ in the form of a vector, a typical (encoder-decoder) PLM first uses its encoder to convert $\mathbf{P}$ into a high-dimensional hidden vector $\mathbf{x}$, and then uses its decoder to generate an SQL query Q, *i.e.,*

$$Q = f_{\texttt{decode}}(\mathbf{x}) = f_{\texttt{decode}}(f_{\texttt{encode}}(\mathbf{P})) \tag{2}$$

A typical solution for decoder $f_{\texttt{decode}}$ is the *auto-regressive* method, as described in Section 2.2. Large PLMs, such as T5 [18] used in our paper, usually have a large number of (sub-word) tokens, which increases the possibilities of generating invalid SQLs. For this reason, *constrained decoding* is introduced to make invalid SQL queries unpresentable [24]. However, existing solutions are designed for directly generating SQL, instead of the structures and contents in our SC-Prompt framework. Therefore, we introduce a *fine-grained constrained decoding* method, which takes full advantages of our structure and content prompting mechanism. More details of constrained decoding are given in Section 5.

4 PROMPT CONSTRUCTION

In this section, we first discuss two conventional prompt construction methods and their limitations in Section 4.1. Then, we introduce our novel hybrid prompt construction method and its learning method in Section 4.2 and Section 4.3 respectively.

4.1 Conventional Prompt Construction Methods

Textual Prompts. The most common way to construct a prompt is to design a textual template, as mentioned in Section 3. Typically, a textual prompt consists of discrete tokens[17], such as a task description, to guide the PLMs to obtain the desired output. For example, we can use prompts like "`translate to french`, `[input]`, `[output]`" to exploit PLMs' excellent ability to handle generation

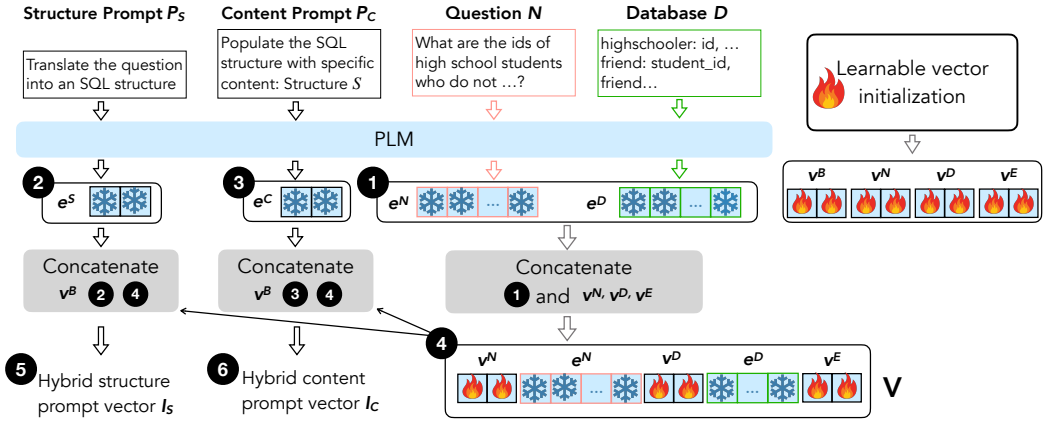


Fig. 3. Hybrid prompt construction.

tasks including text summarization and machine translation. This method constructs prompts in a human-interpretable way, which is currently the most intuitive method.

Note that, a textual prompt needs to be converted into a sequence of vectors, in order to be used by the encoder-decoder Text-to-SQL architecture. This is often performed using word embeddings of PLMs where each input token will be converted into a vector. For example, the NL question N, after being tokenized as $\texttt{tok}(N) = t_1, \ldots, t_n$, will be converted into a sequence of vectors as $e^N = v_1, \ldots, v_n$, similar for the embeddings e^D of the database schema D (see Figure 3 for $\texttt{tok}(N)$, $\texttt{tok}(D)$, e^N and e^D). These prompts are considered to be *fixed*, meaning that they will not be updated (or tuned) during training on new datasets.

Learnable Vectors. This prompt consists of learnable vectors (*a.k.a.* continuous vectors [11, 13]), which can be optimized during training. For example, prompt tuning [11] prepends a series of continuous vectors to the input as a prompt; this shows good performance on the SuperGLUE benchmark [29]. This method expands the design space of prompt beyond human-interpretable language, allowing PLMs to learn which features should be included in prompts.

Limitations. The main limitation of textual prompts is that they are designed manually, and thus highly rely on the designers' expertise. Even the most experienced designers struggle to discover optimal prompt templates [10]. The main limitation of learnable vectors, as empirically verified by PPT [7], is that prompt tuning on learnable vectors performs poorly with few-shot training and is greatly influenced by the prompt initialization methods (see Section 4.2 for different initialization methods).

4.2 Hybrid Prompt Construction

Although both methods have limitations, their advantages, as discussed above, are complementary. The main advantage of textual prompts is that it is able to capture experts' domain knowledge, and the main advantage of learnable vectors is its adaptability to new datasets during training. Based upon this observation, we propose a novel *hybrid prompt* method to combine the best of both worlds. Recall that SC-Prompt has two stages, as shown in Figure 2, the SQL structure stage and the SQL content stage. Thus, we construct a hybrid prompt I_S as the input for the structure stage, and a hybrid prompt I_C as the input for the content stage. Specifically, the construction process is composed of an *input preprocessing step* and a *hybrid prompt construction step*, as shown in Figure 3.

Input Preprocessing Step. We first tokenize the input NL question N and database schema D. The query N is tokenized into a sequence of tokens as $\texttt{tok}(N) = t_1, t_2, \ldots, t_n$, and the flattened

database schema D containing table names and column names into another sequence of tokens as $\texttt{tok}(D) = t'_1, t'_2, \ldots, t'_m$.

Afterwards, we use a PLM to convert the tokenized sequence to a vector sequence, where each token will be converted to one high-dimensional vector. That is, the tokenized question $\texttt{tok}(N)$ will be converted into e^N, and the tokenized database schema $\texttt{tok}(D)$ will be converted into e^D, as shown in Figure 3-❶. Note that, we use the snowflake symbol (❄) to denote that these vectors are fixed, which will not be tuned during training.

Hybrid Prompt Construction Step. Based on the above tokenization results, we construct the textual prompts and the learnable vectors, respectively, and finally combine them to obtain the hybrid prompt, as described as follows.

Sub-task Specific Textual Prompt. Since the two sub-tasks have different goals, we add textual prompts composed of specific task descriptions to guide the model to produce the desired result. The structure prompt $\mathbf{P}_s$ and content prompt $\mathbf{P}_c$ are designed as follows:

- $\mathbf{P}_s$ = *"translate the question into an SQL structure"*.
- $\mathbf{P}_c$ = *"populate the SQL structure with specific content: SQL structure S"*.

Each of the above prompts will first be tokenized and then converted into a vector sequence (same as the process of obtaining e^N), where e^S is the vector sequence for $\mathbf{P}_s$ (see Figure 3-❷) and e^C is the vector sequence for $\mathbf{P}_c$ (see Figure 3-❸). Note that, these vector sequences are also fixed.

Learnable Vectors. As it is hard to design appropriate fixed prompts to effectively guide PLMs, we introduce learnable vectors, which can be tuned during training, for learning more contextual information. The basic idea is to introduce four learnable vectors. Each one is designed for a different purpose, *e.g.,* question, database, and task, to help the PLM understand different aspects of the input.

Formally, we introduce four learnable vectors $\{v^B, v^N, v^D, v^E\}$, which are designed for different purposes: (1) v^N is for learning NL question N specific features, (2) v^D is for learning database D specific features, and (4) v^B and v^E are for learning task specific features at the beginning and end of the input context, respectively. The main intuition is that due to the positional encoding mechanism of the transformer-based PLMs, the learnable vectors at the specific points can implicitly learn the specific contextual information during training, so as to help the PLM understand different parts of the input (*e.g.,* the database schema) during prediction. Note that we only put the learnable vectors in the input embedding layer, which is different from prefix-tuning [13] that prepends the vectors to each layer in the encoder stack, including the input layer. Our experiments show that this simpler solution is sufficient to achieve good performance in SC-Prompt. We conduct an ablation study to demonstrate that our method works better than using one single learnable vector with respect to one purpose (*e.g.,* question, database, or task). Please refer to Section 7.2 for more details.

Next we describe three methods to initialize these four vectors.

(1) *Random Initialization* randomly initializes all the vectors based on a normal distribution.

(2) *Vocabulary Initialization* randomly samples words from the vocabulary of the PLM to initialize the vectors.

(3) *Keyword Initialization* limits the initialization space to keywords related to our Text-to-SQL task, *e.g.,* "database", "table", "question", etc.

Prompts Combination. Let the embedding of database e^D be $e^D = e^{D_{name}} \oplus e^T$, where "$\oplus$" is for vector concatenation, $e^{D_{name}}$ is e^D's sub-sequence *w.r.t.* the database name, and e^T is e^D's sub-sequence *w.r.t.* the table schema.

We first combine ❶ the embedding for the NL question and database schema with the learnable vectors $\{v^N, v^D, v^E\}$ as follows:

$$\mathbf{V} = v^N \oplus e^N \oplus v^D \oplus e^{D_{name}} \oplus e^T \oplus v^E \tag{3}$$

The combined vector sequence $\mathbf{V}$ is depicted in Figure 3-❹. As discussed above, the reason to do such combination is to learn features for each learnable vector with different semantics.

For the structure stage, we concatenate the above vector sequence $\mathbf{V}$, the task-specific vector v^B, and the vectors for the structured prompt e^S as the hybrid structure prompt (see Figure 3-❺):

$$I_S = v^B \oplus e^S \oplus \mathbf{V} \tag{4}$$

This hybrid structure prompt I_S will be used as the input to the *structure stage*.

For the content stage, we concatenate the above vector sequence $\mathbf{V}$, the task-specific vector v^B, and the vectors for the content prompt e^C as the hybrid content prompt (see Figure 3-❻):

$$I_C = v^B \oplus e^C \oplus \mathbf{V} \tag{5}$$

This hybrid content prompt I_C will be used as the input to the *content stage*.

Example 3 (Hybrid prompt construction for Text-to-SQL). *Consider the NL question N and the database D in Figure 1.*

Input preprocessing. *First, the database D is flattened to "Database: Network. Table highschooler: id, name, ..., grade; Table friend: student_id, ... ". And the flattened database is tokenized to a token list:* $\texttt{tok}(D) = [$*"database", ":", "network", ...*$]$. *Similarly, the NL question N is also tokenized to* $\texttt{tok}(N) = [$*"what", "are", ...*$]$. *Second, the base PLM is used to convert the tokens to high-dimensional vectors:* $\texttt{tok}(D)$ *is converted to* $e^D = e^d \oplus e^{T_{highschooler}} \oplus e^{T_{friend}}$; $\texttt{tok}(N)$ *is converted to* e^N.

Hybrid structure prompt construction. *First, we combine e^D and e^N with the four learnable vectors $\{v^B, v^N, v^D, v^E\}$ to get $\mathbf{V} = v^N \oplus e^N \oplus v^D \oplus e^{D_{name}} \oplus e^{T_{highschooler}} \oplus e^{T_{friend}} \oplus v^E$. Second, we convert textual prompt $\mathbf{P}_s$ to a vector sequence e^S and concatenate it with V and v^B to get input I_S for predicting SQL structure: "*`SELECT [col] FROM [tab] EXCEPT SELECT [col] FROM [tab]`*".*

Hybrid content prompt construction. *This stage is very similar to the previous stage, except that the textual prompt $\mathbf{P}_c$ contains the SQL structure S generated above. we convert the textual prompt $\mathbf{P}_c$ to a vector sequence e^C and concatenate it with V to form the input I_C for predicting the SQL content: "*`[col] id [tab] highschooler [col] id [col] student_id [tab] friend`*".*

4.3 Hybrid Prompt Leaning

In this section, we discuss how to learn the parameters in our constructed hybrid prompt (*e.g.,* the learnable vectors discussed above), and the base PLM.

SQL Structure Prompt Learning. Recall that this sub-task aims to infer the SQL structure without specific values and identifiers. Next, we present how to train this sub-task through training data preparation and prompt learning.

Training data preparation. To obtain the SQL structure representation, we replace the identifiers and constant values of the ground truth SQL query with placeholders. We design three special tokens as the placeholders: `[col]`, `[tab]`, `[val]`, which represent a column name, table name, or constant value, respectively.

Using Figure 1 as an example, the ground truth SQL query "`SELECT id FROM highschooler EXCEPT SELECT student_id FROM friend`" is converted into the SQL structure representation "`SELECT [col] FROM [tab] EXCEPT SELECT [col] FROM [tab]`".

Prompt learning. As described in Section 4.2, the input I_S of this sub-task is constructed according to Equation. (4). Then, I_S is fed into the base PLM for encoding, and the expected structure

representation are generated through auto-regressive decoding. The trainable parameters here include the SQL structure generator G_s (*i.e.,* the PLM used in Stage-S) and the learnable vectors, which are denoted as Θ_s. Next, we learn Θ_s by minimizing the negative log-likelihood loss, *i.e.,*

$$\begin{aligned} L(\Theta_s) &= -\frac{1}{n}\sum_{i=1}^{n}\log P(S^i_{gold}|I^i_S;\Theta_s) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m_i}\log P(S^{i,j}_{gold}|I^i_S;S^{i,1}_{gold},\ldots,S^{i,j-1}_{gold};\Theta_s) \end{aligned} \tag{6}$$

where I^i_S is the input of of the i_{th} training sample, S^i_{gold} is the corresponding ground truth structure representation, $S^{i,j}_{gold}$ is the j_{th} token of S^i_{gold} and n is the number of training samples.

SQL Content Prompt Learning. This sub-task aims to infer specific SQL content according to the question and database schema. Again we will introduce two aspects of this module: training data preparation and prompt learning.

Training data preparation. As this sub-task needs to fill specific content for each slot in the SQL structure, to be consistent with SQL structure, we concatenate each placeholder and its corresponding identifier or constant value to construct the SQL content representation. For example, for Figure 2, the SQL content representation is "`[col] id [tab] highschooler [col] id [col] student_id [tab] friend`".

Prompt learning. The input I_C of this sub-task is constructed according to Equation (5), and then fed into the PLM for encoding, which generates output values through auto-regressive decoding. The trainable parameters here include th SQL content generator G_c (*i.e.,* the PLM used in Stage-C) and the learnable vectors, which are denoted as Θ_c. Similar to SQL structure prompt learning, we learn Θ_c by minimizing the negative log-likelihood loss.

Algorithm 1 illustrates the overall process of hybrid prompt construction and learning. We train the models for the structure and content sub-tasks separately. For each sub-task, we first construct the corresponding hybrid prompt by combining learnable vectors and textual prompt (Section 4.2), and then train the learnable parameters. In particular, we finetune the PLMs, and finetune the learnable vectors only if they are used.

5 FINE-GRAINED CONSTRAINED DECODING

In this section, we describe optimizations for the decoding phases of the two SC-Prompt stages.

Constrained Decoding. Constrained decoding approaches enable us to assert control over the result of text generation. This is valuable when we know exactly what are desired in the output. This technique is widely used for *neural program synthesis* [24, 27].

For example, when generating an SQL structure S, we know that only SQL commands, operators and placeholders should be included in the output sentence, but not any content. For populating SQL contents, we have a fixed structure, and only a few content placeholders need to be filled.

Beam Search. Beam search, which is different from greedy search, retains a list of hypotheses [33], where the length of the hypotheses list is the beam size. Consider beam search of size K: Beam search keeps the most likely K hypotheses at each time step and eventually chooses the hypothesis that has the overall highest probability (the hypothesis probability is calculated by Equation (1)). At time step i, only the top-K next possible tokens at each possible hypothesis are retained while the others are pruned. Taking the standard beam search process shown in Figure 4 as an example, the top-2 possible tokens obtained in time step 1 are {"select", "the"}, and the other tokens in the PLM's vocabulary are excluded because they are not in the top-2 possible tokens.

Algorithm 1: Hybrid Prompt Construction and Learning.

Input: a Text-to-SQL dataset $\{N_i, D_i, Q_i\}_{i=1}^{n}$

begin
  Initialize the SQL structure generator G_s
  Initialize learnable vectors $\{v^B, v^N, v^D, v^E\}$
  $\mathcal{S}_{gold} = \{S^i_{gold}\}_{i=1}^{n} \leftarrow TrainingDataPreparation(Q_i)$
  $\{I^i_S\}_{i=1}^{n} \leftarrow HybridPromtConstruction(N_i, D_i)$
  for *s-epochs* **do**
    Generate the SQL structure: $\mathcal{S}_{pred} = \{S^i_{pred}\}_{i=1}^{n} \leftarrow G_s(I^i_S)$
    Compute the negative log-likelihood loss: $L(\mathcal{S}_{gold}; \mathcal{S}_{pred}; \Theta_s)$
    // Θ_s denotes the parameters of SQL structure generator and the learnable vectors
    Update Θ_s
  end
  Initialize SQL content generator G_c
  Initialize learnable vectors $\{v^B, v^N, v^D, v^E\}$
  $C_{gold} = \{C^i_{gold}\}_{i=1}^{n} \leftarrow TrainingDataPreparation(Q_i)$
  $\{I^i_C\}_{i=1}^{n} \leftarrow HybridPromtConstruction(N_i, D_i)$
  for *c-epochs* **do**
    Generate the SQL content: $C_{pred} = \{C^i_{pred}\}_{i=1}^{n} \leftarrow G_c(I^i_C)$
    Compute the negative log-likelihood loss: $L(C_{gold}; C_{pred}; \Theta_c)$
    // Θ_c denotes the parameters of the SQL content generator and learnable vectors
    Update Θ_c
  end
end

Next, we combine constrained decoding and beam search to design decoding methods for the two stages: keyword constrained decoding for SQL structure generation and structure guided decoding for SQL content prediction.

5.1 Keyword Constrained Decoding

The SQL structure stage only needs to infer the SQL structure without specific content. Therefore, the output of this stage should only contain valid commands (*e.g.,* "SELECT"), operators (*e.g.,* ">"), and special tokens (*e.g.,* `[col]`) used for placeholders. To achieve this, we use a customized vocabulary $\mathcal{V}_{key}$ that only contains the above valid tokens, and limit the output of the decoder to not exceed the vocabulary. Technically, we leverage the vocabulary $\mathcal{V}_{key}$ to constrain beam search. At time step i, we further check the top-K possible tokens according to the vocabulary $\mathcal{V}_{key}$; that is, we filter out the tokens that are not in $\mathcal{V}_{key}$.

5.2 Structure Guided Decoding

For SQL content prediction, we use the SQL structure to guide the decoding process. Specifically, we design different pruning strategies on beam search at *slot-level* and *sentence-level*.

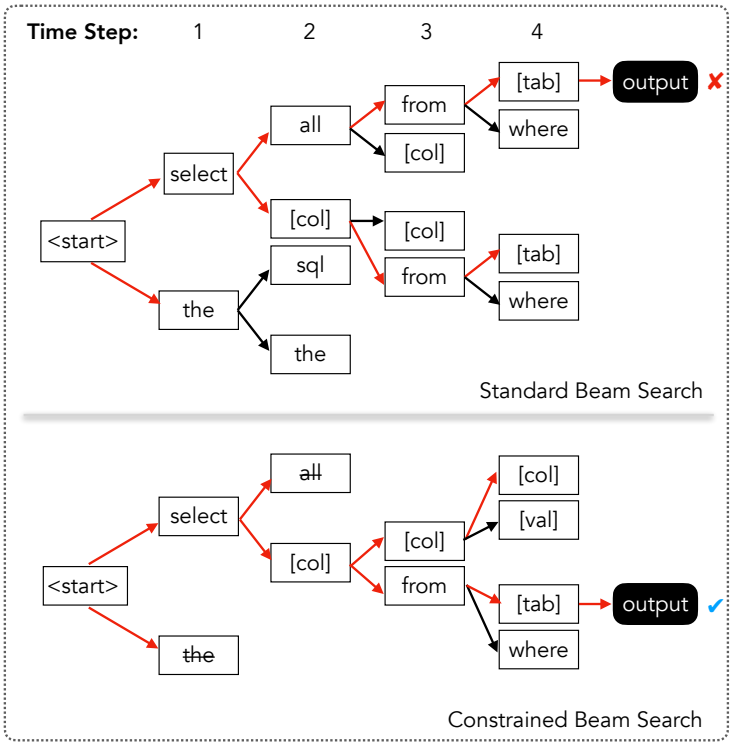


Fig. 4. Example of our fine-grained constrained decoding for the question and database schema in Figure 1.

Slot-Level. The goal of slot-level constraints is to ensure that the predicted content is consistent with the slot (*i.e.,* placeholder). For example, `[col]` can only be followed by the column name. Technically, we check each time step of beam search, and set the score of candidate tokens that do not match the slot to $-\infty$. This allows the decoder to search over only valid content.

Sentence-Level. The goal of sentence-level decoding is to ensure that the generated SQL query is semantically correct at the sentence-level. We use four consistency checks:

- ***Column-Table Consistency*** avoids the use of column names that are not in the FROM list of SQL queries.
- ***Operator-Value Consistency*** avoids operations on values of incompatible types.
- ***Aggregation-Column Consistency*** avoids aggregation operations whose type does not apply to its target column.
- ***Column-Column Consistency*** avoids operations on two columns whose value types do not match each other.

Unlike the slot-level checks, sentence-level checks are performed after the beam search. That is, after beam search outputs the K hypotheses with the highest probability score, we check their correctness. The one that is correct and has the highest probability score will eventually be selected. If all the top-K hypotheses fail to pass these checks, we default to the hypothesis with the highest probability.

Example 4 (Fine-grained constrained decoding). *Consider the NL question N and the database D in Figure 1.*

Keyword constrained decoding for the structure stage. *The target output is "*`SELECT [col] FROM [tab] EXCEPT SELECT [col] FROM [tab]`*". The SQL structure decoder uses beam search to keep the most likely K hypotheses at each time step and leverages vocabulary $\mathcal{V}_{key}$ to limit each hypothesis to contain only valid SQL commands, operators and three placeholders. As shown in the constrained beam search process in Figure 4, "the" in time step-1 is excluded from the subsequent beam search.*

Structure guided decoding for the content stage. *The output is "*`[col] id [tab] highschooler [col] id [col] student_id [tab] friend`*". The SQL content decoder first ensures that each slot is followed by the correct content. If the slot is* `[col]`*, the next possible token "friend" will be rejected because it is not a column name of the database D. Then after the beam search outputs the top-K hypotheses, the decoder checks the semantic correctness of each hypotheses and rejects the hypotheses containing the following four errors: (i) Column-Table inconsistency. The column name "id" appears in the hypothesis, but its corresponding table "highschooler" does not appear. (ii) Operator-Value inconsistency. The comparison operator < operates on a textual value. (iii) Aggregation-Column inconsistency. The aggregation operator* `SUM` *operates on the column "name". (iv) Column-Column inconsistency. The hypothesis is "*`[col] id [tab] highschooler [col] name [col] student_id [tab] friend`*", where "name" and "student_id" do not match because "student_id" is a numeric column and "name" is string.*

6 EXPERIMENTAL SETUP

6.1 Datasets

Table 1. Statistics of Text-to-SQL benchmarking datasets: #Instance, #DB, and #Domain represent numbers of annotated instances, relational databases, and domains respectively, and #Table/DB denotes the average number of tables per database.

Dataset	#Instance	#DB	#Domain	#Table/DB	ORDER BY	GROUP BY	NESTED	HAVING
Spider [39]	10181	200	138	5.1	1335	1491	844	388
CoSQL [38]	15598	200	138	5.1	2290	2171	840	688
GeoQuery [40]	877	1	1	6	20	46	167	9

We conduct experiments on three well-adopted benchmarking datasets for Text-to-SQL, with differences in question types, query complexities and database diversities. Table 1 summarizes the statistics of the three datasets.

Spider. Spider is a well-known large-scale and complex Text-to-SQL benchmark [39], which contains 200 databases in the Spider dataset and covers 138 domains, such as colleges, clubs, TV shows, government, etc. Specifically, each instance of the Spider dataset contains an NL question, a database schema and an SQL query, annotated by humans. Overall, there are 10,181 instances with 5,693 unique SQL queries. The SQL queries are divided into four difficulty levels according to the complexity, namely *easy*, *medium*, *hard*, and *extra hard*. Among them, Spider randomly selects 7000 annotated instances as the training set. In this paper, to evaluate *few-shot* Text-to-SQL, we randomly select four subsets from the training set, with the percentages of 5%, 10%, 15% and 20%.

CoSQL. CoSQL is a dialogue-based Text-to-SQL benchmark [38], where each dialogue simulates a real-world database query scenario. A user interactively explores a database with NL questions. We consider the *SQL-grounded dialogue state tracking* task in CoSQL, which aims to translate a user's NL questions into SQL queries given the *interaction* history of the user in a dialogue. In particular, CoSQL contains 200 databases from 138 different domains, and 15,598 annotated (NL, SQL) instances are divided into four difficulty levels. Compared with Spider, CoSQL demonstrates greater language diversity (refer to [38] for details). Note that, we run experiments on four sizes of *few-shot* training sets: 5%, 10%, 15% and 20%.

GeoQuery. GeoQuery is a simple Text-to-SQL benchmark with only one database about US geography [40]. It has 877 annotated instances, where each instance contains an NL question and an SQL query. For fair comparison, we follow the existing approach SeqZero [36] (which will be introduced later) to split the GeoQuery dataset based on the patterns of SQL queries, and thus obtain three training sets with sizes: 536, 150 and 50.

6.2 Evaluation Metrics

For fair comparisons, we use the *standard* evaluation processes in the three benchmarks.

(1) The standard evaluation process in Spider measures Text-to-SQL performance using two metrics, namely *exact-set-match accuracy (EM)* and *execution accuracy (EX)*. Exact-set-match accuracy measures whether a generated SQL query is *literally* equal to the ground-truth SQL query. We adopt the official evaluation script provided by Spider [39] to compute EM. However, as some approaches may generate literally different but result-equivalent SQL queries, exact-set-match may yield a problem of false negative evaluation. Thus, we also consider execution accuracy (EX), which compares execution results of the generated and the ground-truth SQL queries retrieved from the database.

(2) The standard evaluation process of CoSQL measures Text-to-SQL performance using two metrics. First, like Spider, CoSQL considers the *exact-set-match accuracy* over all instances, without considering the dialogues. Although this metric is called *question match accuracy (QM)* in its original paper [38], for consistency, we also call it as *exact-set-match accuracy (EM)*. The second metric *interaction match accuracy (IM)* is designed for the dialogue-based Text-to-SQL process. It computes the exact-set-match accuracy over all dialogue interactions. As each interaction may contain multiple (NL, SQL) pairs, we consider an interaction being correct if all the instances in the interaction are correctly predicted.

(3) The standard evaluation process of GeoQuery only considers the *exact-set-match accuracy (EM)* as evaluation metric, following the existing baseline approaches [1, 36]. The main reason is that this benchmark is relatively simple, *e.g.,* containing only one database, and exact-set-match accuracy would not incur the false negative evaluation issue. Thus, execution accuracy (EX) can be omitted without affecting the evaluation results.

6.3 Baselines

To comprehensively evaluate the SC-Prompt framework, we used 10 state-of-the-art Text-to-SQL methods at the time of writing as baselines: DT-Fixup [35], LGESQL [3], SmBoP [20] are specifically designed for the Spider dataset; R2SQL [8] is specific to the CoSQL dataset; T5 [18] and PICARD [24] can be applied to both Spider and CoSQL datasets; finally, Iyer et al. [9], GECA [1], Zheng and Lapata [42], and SeqZero [36] are for designed for and reported to perform well on GeoQuery.

DT-Fixup [35] proposes a theoretically reasonable optimization strategy. It trains a deep-transformer model on Text-to-SQL task and the reading comprehension task, which has good generalization ability and fast convergence speed.

LGESQL [3] applies a line-graph enhanced relational graph attention network (RGAT) as the encoder, and employs an additional edge-centric line graph constructed from a question-database node-centric graph.

SmBoP [20] develops a semi-autoregressive bottom-up parser. We report the results of SmBoP combined with Grappa [37], which is a BERT-like language model pre-trained on a large number of question-SQL pairs, and have achieved superior performance.

T5 [18] is a sequence-to-sequence language model pre-trained with fill-in-the-blank tasks over a large-scale general corpus. To support Text-to-SQL, similar to [24, 26], we fine-tune the model in an end-to-end manner.

PICARD [24] also uses pre-trained language models for Text-to-SQL. To improve the performance, it constrains the auto-regressive decoders of transformer models through incremental parsing. PICARD provides four decoding mode settings: off, lexing, parsing without guards, and parsing with guards. Among these modes, parsing with guards can achieve the best performance, and thus is considered in our experiments.

R2SQL [8] is a dynamic graph framework that effectively models context-dependent questions and their complicated interaction. A dynamic memory decay mechanism integrates enriched contextual relation representation, so as to further improve the performance.

Iyer et al. [9] proposes neural sequence models to map a natural language question directly to an SQL query, bypassing any intermediate representation layers.

GECA [1] proposes a data augmentation method to provide a compositional inductive bias for sequence-to-sequence PLMs, where synthetic training examples are generated by modifying real Text-SQL pairs.

Zheng [42] proposes an auto-regressive decoding framework with lexical style alignments and non-disentangled information processing capabilities to systematically generalize to unseen compositions of seen components of SQL queries.

SeqZero [36] decomposes an SQL query into a sequence of sub-clauses (*i.e.,* SELECT, FROM and WHERE sub-clauses), and uses PLMs to predict each sub-clause sequentially SeqZero adopts zero-shot models to avoid overfitting in each sub-clause.

To evaluate the performance of our hybrid prompt construction strategy, we also consider two simple end-to-end prompt construction methods, which are described in Section 4.1:

Textual Prompts + T5-Large. We use the Text-to-SQL task description as the prompt (*e.g.,* "Translate the question into a SQL query.") and fine-tune T5-Large in an end-to-end fashion.

Learnable Vectors + T5-Large. We use the learnable vectors as the prompt and fine-tune T5-Large in an end-to-end fashion. Note that we use the standard prompt tuning method [11], i.e. only the learnable vectors are trained, while parameters of PLM are fixed.

6.4 Implementation Details

We use T5 [18] as the base PLM. T5 is an encoder-decoder model pre-trained on the Colossal Clean Crawled Corpus (C4). We set the dimension of our learnable vector (see Section 4) to 1024, so as to be consistent with the base PLM. Besides, we implement the experiments with the open-source toolkit Huggingface Trainer[3] on a server with 4 NVIDIA RTX 3090 24GB GPUs. The Adafactor optimizer is used to tune the parameters in our proposed framework. The SQL parser used in the fine-grained constrained decoding phase is adapted from Picard [24], which is also our baseline model. For three Text-to-SQL datasets, in order to improve the stability of training, we first use a learning rate of 0.1 and a batch size of 32 to train the learnable vectors, and then use a learning rate of 0.00005 and a batch size of 32 to train the learnable vectors and the base PLM. We set the max input length to 512 and the inputs exceeding the length limit are truncated.

7 EXPERIMENTAL RESULTS

In this section, we first report the overall performance of our SC-Prompt framework on the three benchmarks (Section 7.1). Then, we respectively evaluate our proposed hybrid prompt construction

[3]https://huggingface.co/docs/transformers/main_classes/trainer

method (Section 7.2) and the fine-grained constrained decoding strategy (Section 7.3). Finally, we discuss the evaluation with full training data (Section 7.4), and provide a case study to qualitatively analyze the effectiveness of SC-Prompt (Section 7.5).

7.1 Overall Results

Spider: Table 2 reports the overall results of different methods on the Spider dataset. We first analyze the results of existing methods with PLMs. First, LGESQL + ELECTRA-Large [3] and SmBoP + Grappa [20] achieve good performance, due to the *task-specific* PLMs used in these methods. Specifically, LGESQL + ELECTRA-Large [3] devises graph attention networks to capture structure information of NL questions and database tables, while SmBoP + Grappa [20] utilizes Grappa [37], a BERT-like language model pre-trained on a large number of synthetically generated (NL, SQL) pairs. The results show that such task-specific strategies are helpful to few-shot Text-to-SQL. Second, general pre-trained seq2seq models also demonstrate their great potential for this task. Specifically, Picard [24] and [26] also achieve good performance in the few-shot scenario using general PLMs (*e.g.,* T5) with an effective optimization strategy, rather than relying on task-specific PLMs.

To compare with Picard [24], our proposed SC-Prompt framework also adopts T5 as our base PLM. The experimental results show that SC-Prompt significantly outperforms the baselines by a large margin. For example, compared with Picard + T5-large [24], SC-Prompt improves the exact-set-match accuracy and the execution accuracy by **5.1** and **4.6** respectively, given 5% training data. In addition, we can observe that, the less the training data is, the more significantly SC-Prompt outperforms the baseline methods. The results show that our SC-Prompt framework, which divides Text-to-SQL into two simpler sub-tasks, can better tackle the challenge of limited training data.

CoSQL: Table 3 reports the overall results on the CoSQL dataset, which is a complex dialogue-based Text-to-SQL benchmark. We have a similar observation that SC-Prompt outperforms all the baselines, *e.g.,* outperforming the previous SOTA by **3.2** on exact-set-match accuracy and **3.1** on interaction match accuracy given 5% training set. Picard + T5-Large [24] also achieves good performance as general PLMs can effectively capture interaction context of NL questions. The results show that SC-Prompt can improve the performance of dialogue-based Text-to-SQL, which is more challenging than individual Text-to-SQL translations.

GeoQuery: Table 4 reports the overall results on the GeoQuery dataset. We can see that SC-Prompt achieves the best performance among all the methods. Compared with the SeqZero [36] using manually designed SQL decomposition, SC-Prompt achieves better results under different training data sizes, even though the performance of our base PLM (T5-Large) is slightly worse than that of SeqZero (with Bart-Large as base PLM). It is worth noting that, unlike the results on both Spider and CoSQL datasets, SC-Prompt does not improve significantly compared to the previous SOTA SeqZero when there is very few training instances (*e.g.,* 50). The main reason is that SeqZero adopts manually designed SQL decomposition, *i.e.,* dividing an SQL query into SELECT, FROM and WHERE sub-clauses, which also performs well as the GeoQuery dataset only contain one simple database. In contrast, SC-Prompt does not rely on manually designed templates, but guides the PLM to predict the SQL structure as well as SQL content. Obviously, SC-Prompt would be more general in real-world scenarios, where we do not know which SQL sub-clauses a user's SQL query has.

An in-depth analysis on difficulty levels: To better understand the model performance on different queries, we perform an in-depth analysis based on the different difficulty levels (*i.e.,* Easy, Medium, Hard, and Extra Hard) on the Spider and CoSQL datasets. Recall that the difficulty levels are defined based on the number of SQL components, predicates, etc. Please refer to Section 6.1 for more details. Figure 5 shows the experimental results. We can see that SC-Prompt performs

Table 2. Overall results of few-shot Text-to-SQL on the Spider dataset. We randomly select 5%, 10%, 15% and 20% of the official training set as the few-shot training sets for evaluation. Note that the notation "-" indicates that the source code of a method does not consider the evaluation metric.

Model	EM (%)				EX (%)			
% Official training data	5%	10%	15%	20%	5%	10%	15%	20%
Existing Methods with PLMs								
DT-Fixup [35]	22.9	36.1	41.3	48.2	-	-	-	-
LGESQL + BERT-Large [3]	26.4	35.1	39.4	48.5	-	-	-	-
LGESQL + ELECTRA-Large [3]	38.3	47.8	53.7	58.2	-	-	-	-
SmBoP + Grappa [20]	36.9	48.2	56.2	58.7	-	-	-	-
T5-Large [26]	34.0	42.7	45.3	50.1	37.3	46.2	48.1	52.5
Picard + T5-Large [24]	40.1	46.4	50.2	56.4	43.7	52.7	55.4	57.9
Conventional Prompt Learning Methods								
Textual Prompts + T5-Large	34.7	43.5	46.1	50.4	37.9	47.1	48.3	53.1
Learnable Vectors + T5-Large	28.4	38.7	39.8	40.3	32.3	41.3	44.1	44.5
SC-Prompt (Our Approach)								
SC-Prompt + T5-Large	**45.2** (+5.1)	**52.8** (+4.6)	**57.2** (+1.0)	**59.5** (+0.8)	**49.1** (+5.4)	**56.9** (+4.2)	**59.3** (+3.9)	**62.0** (+4.1)
w/o Hybrid Prompt Learning	43.7	49.5	54.2	56.6	46.2	54.4	57.1	59.6
w/o Constrained Decoding	39.5	48.1	50.2	52.4	42.7	50.3	51.9	55.1

Table 3. Overall results of few-shot Text-to-SQL on the CoSQL dataset. We randomly select 5%, 10%, 15% and 20% of the official training set as the few-shot training sets for evaluation.

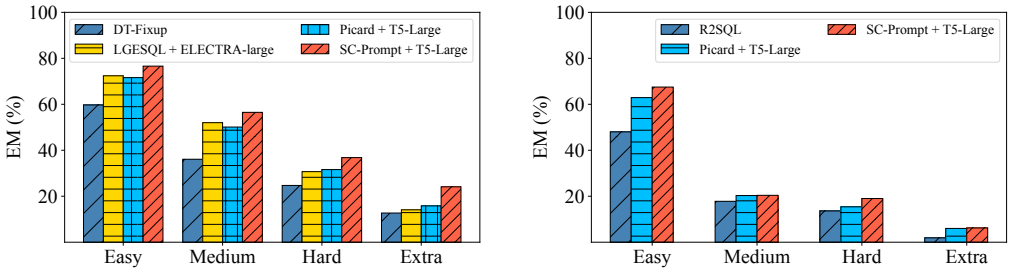
Model	EM (%)				IM (%)			
% Official training data	5%	10%	15%	20%	5%	10%	15%	20%
Existing Methods with PLMs								
R2SQL [8]	21.3	27.9	31.9	34.8	4.1	6.5	7.2	9.6
T5-Large [26]	27.4	33.1	37.5	39.7	4.1	5.8	11.0	14.0
Picard + T5-Large [24]	30.2	35.6	39.8	42.1	6.9	8.3	13.4	**16.2**
Conventional Prompt Learning Methods								
Textual Prompts + T5-Large	28.0	33.5	37.6	39.9	4.4	6.2	10.9	13.7
Learnable Vectors + T5-Large	22.9	28.1	32.5	33.1	4.1	5.1	8.6	9.9
SC-Prompt (Our Approach)								
SC-Prompt + T5-Large	**33.4** (+3.2)	**38.3** (+2.7)	**41.2** (+1.2)	**44.2** (+2.1)	**10.0** (+3.1)	**10.1** (+1.8)	**13.7** (+0.3)	16.1 (-0.1)
w/o Hybrid Prompt Learning	32.2	37.1	40.3	42.9	9.2	9.7	13.0	15.2
w/o Constrained Decoding	30.7	35.8	38.7	42.0	7.6	8.6	11.2	13.5

better than the baseline methods at all difficulty levels. In particular, we can see that on the Spider dataset, LGESQL + ELECTRA-large [3] slightly outperforms the general PLM based method Picard + T5-Large on the *Easy* and *Medium* level. This is because the task-specific schema-linking layers in LGESQL + ELECTRA-large can understand the training data more effectively. However, the performance of SC-Prompt also exceeds these task-specific methods, showing that the divide-and-conquer approach of SC-Prompt can address the problem of limited training data in few-shot Text-to-SQL.

Conclusion: (1) With less training data, SC-Prompt is more effective. The main reason is that the PLM is more prone to the poor generalization problem when there is very limited training data

Table 4. Overall results of few-shot Text-to-SQL methods on the GeoQuery dataset. Following SeqZero [36], we use 50, 150 and 536 training instances as the few-shot training sets for evaluation. The notation "-" indicates the corresponding setting is not evaluated in the original paper.

Model	**EM (%)**		
#Training data	**50**	**150**	**536**
Existing Methods with PLMs			
Iyer et al. [9]	-	-	40.0
GECA [1]	-	-	49.0
Zheng and Lapata [42]	-	-	69.6
BART-Large [12]	41.2	73.1	72.5
SeqZero + Bart-Large [36]	48.9	74.2	74.7
T5-Large	39.5	69.7	71.4
Conventional Prompt Learning Methods			
Textual Prompts + T5-Large	40.3	71.9	73.1
Learnable Vectors + T5-Large	14.7	44.3	49.5
Our SC-Prompt Framework			
SC-Prompt + T5-Large	**49.0** (+0.1)	**76.7** (+2.5)	**79.7** (+5.0)
w/o Hybrid Prompt Learning	46.0	73.4	76.5
w/o Constrained Decoding	47.5	74.9	78.1



(a) Comparison on the Spider dataset

(b) Comparison on the CoSQL dataset

Fig. 5. Approach comparisons on the Spider and CoSQL datasets with respect to difficulty levels. As our evaluation metrics have similar comparative results, we only show the results on EM here.

available (*i.e.,* 5%). (2) SC-Prompt outperforms the baseline methods at all difficulty levels in the few-shot setting.

7.2 Evaluation on Prompt Construction

This section focuses on evaluating two aspects of prompt construction, (1) comparing with conventional prompt learning methods, and (2) evaluating different initialization and combination methods for prompt construction.

Comparison with prompt construction methods. To evaluate the effectiveness of our hybrid prompt construction, we consider the two conventional prompt construction methods, Textual Prompts and Learnable Vector, as described in Section 4.1.

The experimental results on our three datasets are reported in Tables 2, 3 and 4, respectively. Comparing "T5-Large" with "Textual Prompts + T5-Large", we observe that the textual prompts improves the performance of the base PLM through an appropriate task description. In contrast, comparing "T5-Large" with "Learnable Vectors + T5-Large", we observe that due to the reduction of the trainable parameters (*i.e.,* only the learnable vectors are trainable), the standard training

Table 5. Comparison of different initialization methods for learnable vectors.

Method	Spider		CoSQL		GeoQuery
	EM (%)	EX (%)	EM (%)	IM (%)	EM (%)
Random Initialization	51.3	55.8	**38.7**	**11.0**	79.5
Vocabulary Initialization	52.6	56.4	38.4	10.2	78.9
Keyword Initialization	**52.8**	**56.9**	38.3	10.1	**79.7**

Table 6. Evaluation on prompt combination methods.

Method	Spider		CoSQL	
	EM (%)	EX (%)	EM (%)	IM (%)
Only Before the NL Question	50.9	54.2	38.1	10.5
Only Before the Database	49.8	53.4	37.7	10.0
Only At the Tail	51.1	55.2	37.9	10.4
Our Prompt Combination Method	**52.8**	**56.9**	**38.5**	**10.8**

method of the learnable vectors [11] cannot achieve comparable performance as the standard full-PLM fine-tuning method. However, since learnable vectors can learn specific semantic features for different Text-to-SQL datasets, which can compensate textual prompts, SC-Prompt proposes a hybrid prompt construction method to combine the best of these two methods. The ablation results show that, even only with our hybrid prompt, SC-Prompt can effectively improve the Text-to-SQL performance, and outperforms conventional prompt construction methods.

Evaluation on initialization methods for learnable vectors. We first compare the three initialization methods for learnable vectors (see Section 4). In particular, we conduct experiments on the Spider dataset and the CoSQL dataset using 10% of the official training data. As shown in Table 5, initialization using keywords related to our Text-to-SQL task (*e.g.,* "database", "table" and "question") is slightly better than vocabulary initialization that sampling from the PLM vocabulary. The main reason is that the vocabulary initialization strategy may introduce a lot of noises, which would affect the model's understanding on the original input. In contrast, the keyword initialization uses task related words to initialize the vectors, which can reduce the possibility of introducing noises and achieve better performance.

Evaluation on prompt combination methods. For different prompt combination methods, we perform an ablation study to compare the methods discussed in Section 4.2: only before the NL question e^N, only before the database e^D, and only at the tail. We conduct experiments on both the Spider dataset and the CoSQL dataset using 10% of the official training data. The experimental results in Table 6 show that our method performs better than other methods. The main reason is that putting the learnable vectors in different positions during the prompt construction process can obtain prompts for different purposes (*e.g.,* learning NL question specific features.), and they can better guide the PLM to output the desired results.

Conclusion: (1) The combination of the textual prompts and the learnable vectors (*i.e.,* hybrid prompt learning) achieves the best results. (2) The three initialization methods perform similarly, with the keyword initialization method slightly better. (3) Constructing the learnable vectors by putting them in different positions works better than putting them in a single position.

7.3 Evaluation on Constrained Decoding

Observed from Tables 2, 3 and 4, the ablation results validate the effectiveness of constrained decoding. For example, we can see that, in Table 2, SC-Prompt outperforms SC-Prompt w/o Constrained Decoding, *e.g.,* 45.2 vs 39.5 on EM given 5% training data. Furthermore, this section evaluates the fine-grained constrained decoding method on the Spider dataset (using 10% training

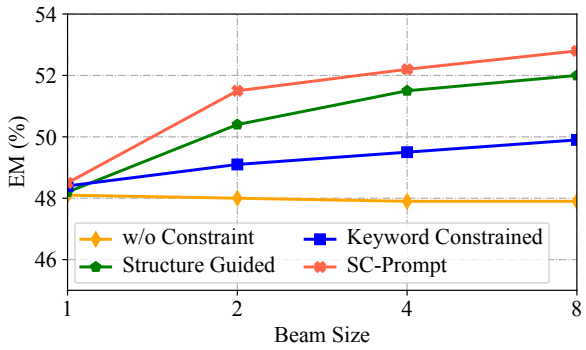


Fig. 6. The performance of different components under different beam sizes in our decoding method. This experiment is conducted on the Spider dataset, using 10% of the training data. EX Acc. (%) and EM Acc. (%) have similar comparative results, so we only show EM Acc. (%) here.

set). Specifically, we evaluate the performance of these two task-specific decoding methods under different beam sizes. Recall that the beam size refers to the number of hypotheses retained by the model during decoding. Thus, the larger the beam size is, the more likely the model can avoid the risk of missing hidden high probability sequences. Note that, when the beam size is 1, the method degenerates into a greedy search strategy. The experimental results are reported in Figure 6. We can see that the performance of the four decoding methods increase, with the increase of beam size, especially for the three decoding methods with constraints. We can also find SC-Prompt, which is equipped with both keyword-constrained and structured-guided decoding, achieves the best performance, which validates the effectiveness of our fine-grained constrained decoding strategies.
Conclusion: (1) The effect of the decoding method will increase with the increase of the beam size. (2) Our fine-grained constrained decoding method, equipped with both keyword-constrained and structured-guided decoding, achieves the best performance.

7.4 Evaluation with Full Training Data

Table 7. Method comparison given full training data.

Base PLM	Method	Spider		CoSQL	
		EM (%)	EX (%)	EM (%)	IM (%)
T5-Large	Picard	71.2	75.8	53.9	22.5
	SC-Prompt	**73.3**	**76.4**	**54.4**	**23.1**
T5-3B	Picard	75.5	79.3	56.9	24.2
	SC-Prompt	**76.9**	**81.1**	**57.6**	**26.3**

Although the goal of SC-Prompt is mainly to solve the poor generalization problem in the few-shot scenario, we also conduct experiments on the full training data setting to verify the effectiveness of our proposed framework under different training data sizes. We conduct experiments on the T5-Large model and the T5-3B model respectively, and compare them with the best baseline model Picard [24]. Table 7 shows the experimental results on the Spider dataset and the CoSQL dataset. Note that the GeoQuery dataset released by [5] only contains 536 training samples, and the results in Section 7.1 have already considered the full training data. We can see that SC-Prompt performs better than the baseline method on the two base PLMs. This shows that even if a large amount of training data is used, the poor generalization problem still exists due to the large complexity of the

Text-to-SQL task; and our divide-and-conquer framework of SC-Prompt is still effective under the full training data setting.

Conclusion: When more training data is available (*e.g.,* 100%), SC-Prompt still outperforms baselines, but the gap becomes smaller, as expected.

7.5 Case Study

This section presents five illustrative examples selected from the Spider, CoSQL, and GeoQuery datasets to qualitatively analyze the effectiveness of our proposed SC-Prompt framework. Table 8 shows the SQL queries generated using the conventional "Pre-train, Fine-tune" paradigm (*i.e.,* T5-Large) and our SC-Prompt framework. From these examples, we have the following findings.

- SC-Prompt can alleviate the issue of overfitting limited training data. For example, T5-Large incorrectly predicts the column name in Question-1 as "student_ id", and incorrectly predicts the table name in Question-5 as "city", because these two column/table names occur frequently in the training set. However, SC-Prompt can avoid such mistakes, as it first predicts the SQL structure and then infers such column/table names by predicting SQL content, which is easier for the PLM than predicting a whole SQL query.
- SC-Prompt can guide the model to better predict the SQL operators, such as `AVG` and `SUM` in Question-4, thanks to the structure and content prompt learning strategy.
- SC-Prompt can better predict the SQL structure, as shown in Question-3. This is attributed to the structure prompt and the fine-grained constraints that can effectively guide the auto-regressive decoding process.

To summarize, these examples illustrate why SC-Prompt is better than traditional end-to-end Text-to-SQL translation, in the few-shot learning setting. As mentioned in the Introduction, the structure stage focuses only on predicting an SQL structure from an NL question, which is much easier than predicting a complete SQL query. Moreover, for the content stage, the task is to fill the content placeholders, a *fill-in-the-blank* task, which is well aligned with the self-supervised mask language modeling for pre-training many PLMs (*e.g.,* BERT [4] and T5 [18]).

8 RELATED WORK

8.1 Text-to-SQL Semantic Parsing

The task of supporting Text-to-SQL with a natural language (NL) interface that allows a non-expert user to access a relational database easily has been studied by the community over the last few decades. In recent years, deep learning models using large-scale training data have substantially improved the accuracy of text-to-SQL. Several methods propose complex encoder architecture with graph-based layers to model the relationship between the natural language question and the database schema. For example, RATSQL [30] proposes a relation-aware self-attention mechanism to encode the hybrid question-schema graph. LGESQL [3] takes the topological structure of edges into account and models the 1-hop edge features in text-to-SQL with a line graph. R2SQL [8] is proposed to solve the conversational text-to-SQL task (*e.g.,* CoSQL [38]), which encodes the relationship in context-dependent questions via a dynamic graph framework. More recently, large PLMs (*e.g.,* GPT-3 [2], T5 [18]) have shown to be effective in improving text-to-SQL and achieving state-of-art performance that take advantage of the extensive knowledge in large-scale corpora. Different from the previous methods with task-specific design, PLM-based methods [24, 26] can be applied to different text-to-SQL datasets without specific modifications to the base PLMs. More importantly, many studies [2, 6, 16] have shown the good performance of large PLMs in few-shot scenarios, which is very important for Text-to-SQL that requires high-quality training data. Therefore, many works have used large PLMs for few-shot Text-to-SQL. To bridge the gap between the natural language and

Table 8. Case study of the generated SQL queries. Questions 1-2 are sampled from the Spider dataset (10%), questions 3-4 are sampled from the CoSQL dataset (10%), and question 5 is sampled from the GeoQuery dataset (536). The errors in the generated SQLs from T5-Large are highlighted in red.

Question-1: *What are the ids of high school students who do not have friends?* ★Ground Truth: SELECT id FROM highschooler EXCEPT SELECT student_id FROM friend •T5-Large: SELECT student_id FROM highschooler EXCEPT SELECT student_id FROM friend (×) •SC-Prompt: SELECT id FROM highschooler WHERE id NOT IN (SELECT student_id FROM friend) (✓)
Question-2: *What are the different years in which there were cars produced that weighed less than 4000 and also cars that weighted more than 3000?* ★Ground Truth: SELECT DISTINCT year FROM cars_data WHERE weight BETWEEN 3000 AND 4000 •T5-Large: SELECT DISTINCT year FROM cars_data WHERE weight > 3000 (×) •SC-Prompt: SELECT DISTINCT year FROM cars_data WHERE weight < 4000 INTERSECT SELECT year FROM cars_data WHERE weight > 3000 (✓)
Question-3: *Can you tell me the age of each teacher?* ★Ground Truth: SELECT name, age FROM teacher •T5-Large: SELECT t2.name, t2.age FROM course AS t1 JOIN teacher AS t2 ON t1.teacher_id = t2.teacher_id GROUP BY t2.name (×) •SC-Prompt: SELECT name, age FROM teacher (✓)
Question-4: *What is the average GNP and total population in all nations whose government is US territory?* ★Ground Truth: SELECT AVG(gnp), SUM(population) FROM country WHERE governmentform = "US Territory" •T5-Large: SELECT AVG(gnp), AVG(population) FROM country WHERE governmentform = "US Territory" (×) •SC-Prompt: SELECT AVG(gnp), SUM(population) FROM country WHERE governmentform = "US Territory" (✓)
Question-5: *What is the population of new mexico?* ★Ground Truth: SELECT statealias0.population FROM state AS statealias0 WHERE statealias0.state_name = "new mexico" •T5-Large: SELECT cityalias0.population FROM city AS cityalias0 WHERE cityalias0.city_name = "new mexico" (×) •SC-Prompt: SELECT statealias0.population FROM state AS statealias0 WHERE statealias0.state_name = "new mexico" (✓)

SQL queries, [27] and [25] first generate canonical natural language utterances with large PLMs and then transform them to the target SQL queries. Next, considering that it is difficult for the PLM to directly output long canonical utterance, SeqZero [36] decomposes the original canonical utterance into multiple sub-clauses (*e.g.,* SELECT-FROM-WHERE) through several manually designed natural language templates and guides the model to predict each sub-clause sequentially.

Different from the above methods, SC-Prompt does not rely on any manually predefined templates or grammar rules. To tackle the challenge of limited training data, SC-Prompt decomposes the complicated Text-to-SQL task into two simpler sub-tasks via structure and content prompt learning. In addition, to generate semantically correct and executable SQL queries, SC-Prompt introduces effective prompt construction and decoding strategies to enhance the performance of the base PLM on the two sub-tasks.

8.2 Few-Shot Learning with Prompt

PLMs combined with a *"pre-train, fine-tune"* paradigm have demonstrated excellent performance on a wide range of tasks in recent years. However, most existing PLMs are pre-trained with language modeling tasks (*e.g.,* masked language modeling [4]), but the objectives of fine-tuning tasks are quite different. The gap between pre-training and fine-tuning tends to make PLMs less effective, especially under few/zero shot scenarios [14]. To address this problem, a novel *"pre-train, prompt, and predict"* paradigm has been proposed [2, 19, 23] to effectively stimulate the knowledge from PLMs. Instead of designing specific objectives to adapt PLMs to downstream tasks, in this paradigm, downstream tasks are reformulated to a language modeling task by inserting language prompts, and the outputs can be easily converted to the downstream task's desired outputs. There are two main methods for prompt designing. One is to use manual design of hard prompts consisting of discrete words (*e.g.,* task description) for prompt learning [2, 22, 27]. The other is not limited to human-interpretable natural language, but uses soft prompt consisting of continuous vectors that can be optimized by the PLM [11, 13]. In this work, we design a hybrid prompt construction method to support (complicated) Text-to-SQL tasks. We first serialize the question and database schema with continuous prompts, and then combine with different textual prompts to obtain the task-specific input (structure or content).

9 CONCLUSION AND FUTURE WORK

In this paper, we have proposed SC-Prompt, a novel divide-and-conquer strategy for effectively supporting Text-to-SQL translation in the few-shot scenario. We divided the process of Text-to-SQL translation into two stages: the structure stage and the content stage. The Structure stage generates an SQL structure, while the content stage fills the placeholders in the SQL structure by SQL identifiers and constant values. We then propose a hybrid prompt to improve the encoding phase, and use fine-grained constraints to make the decoding phase more effective. Our extensive experiments conducted on multiple benchmark datasets for Text-to-SQL clearly show that SC-Prompt outperforms the SOTA method when there is no enough train data (*i.e.,* few-shot learning).

Although we are currently working on few-shot Text-to-SQL, our SC-Prompt framework can be also applied to other meaningful executable languages, such as logical forms [31]. Therefore, applying SC-Prompt to multiple semantic parsing tasks is an interesting future direction worth exploring. In addition, we can also explore how to use transformer-based models with a limited input length (*e.g.,* 512) to encode extra large databases.

ACKNOWLEDGMENTS

This work was partly supported by the NSF of China (62122090, 62072461 and 62072458), and National Key Research and Development Program of China (2020YFB2104101).

REFERENCES

[1] Jacob Andreas. 2020. Good-Enough Compositional Data Augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.). Association for Computational Linguistics, 7556–7566. https://doi.org/10.18653/v1/2020.acl-main.676

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[3] Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. LGESQL: Line Graph Enhanced Text-to-SQL Model with Mixed Local and Non-Local Relations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, 2541–2555. https://doi.org/10.18653/v1/2021.acl-long.198

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[5] Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir R. Radev. 2018. Improving Text-to-SQL Evaluation Methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, Iryna Gurevych and Yusuke Miyao (Eds.). Association for Computational Linguistics, 351–360. https://doi.org/10.18653/v1/P18-1033

[6] Heng Gong, Yawei Sun, Xiaocheng Feng, Bing Qin, Wei Bi, Xiaojiang Liu, and Ting Liu. 2020. TableGPT: Few-shot Table-to-Text Generation with Table Structure Reconstruction and Content Matching. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, Donia Scott, Núria Bel, and Chengqing Zong (Eds.). International Committee on Computational Linguistics, 1978–1988. https://doi.org/10.18653/v1/2020.coling-main.179

[7] Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. PPT: Pre-trained Prompt Tuning for Few-shot Learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, 8410–8423. https://doi.org/10.18653/v1/2022.acl-long.576

[8] Binyuan Hui, Ruiying Geng, Qiyu Ren, Binhua Li, Yongbin Li, Jian Sun, Fei Huang, Luo Si, Pengfei Zhu, and Xiaodan Zhu. 2021. Dynamic Hybrid Relation Exploration Network for Cross-Domain Context-Dependent Semantic Parsing. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 13116–13124. https://ojs.aaai.org/index.php/AAAI/article/view/17550

[9] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a Neural Semantic Parser from User Feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, Regina Barzilay and Min-Yen Kan (Eds.). Association for Computational Linguistics, 963–973. https://doi.org/10.18653/v1/P17-1089

[10] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How Can We Know What Language Models Know. *Trans. Assoc. Comput. Linguistics* 8 (2020), 423–438. https://doi.org/10.1162/tacl_a_00324

[11] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, 3045–3059. https://doi.org/10.18653/v1/2021.emnlp-main.243

[12] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.).

Association for Computational Linguistics, 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703

[13] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, 4582–4597. https://doi.org/10.18653/v1/2021.acl-long.353

[14] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *CoRR* abs/2107.13586 (2021). arXiv:2107.13586 https://arxiv.org/abs/2107.13586

[15] Oracle. 2019. https://docs.oracle.com/en/cloud/saas/service/18b/favau/natural-language-processing.html.

[16] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True Few-Shot Learning with Language Models. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 11054–11070. https://proceedings.neurips.cc/paper/2021/hash/5c04925674920eb58467fb52ce4ef728-Abstract.html

[17] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[18] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67. http://jmlr.org/papers/v21/20-074.html

[19] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67. http://jmlr.org/papers/v21/20-074.html

[20] Ohad Rubin and Jonathan Berant. 2021. SmBoP: Semi-autoregressive Bottom-up Semantic Parsing. In *Proceedings of the 5th Workshop on Structured Prediction for NLP, SPNLP@ACL-IJCNLP 2021, Online, August 6, 2021*, Zornitsa Kozareva, Sujith Ravi, Andreas Vlachos, Priyanka Agrawal, and André F. T. Martins (Eds.). Association for Computational Linguistics, 12–21. https://doi.org/10.18653/v1/2021.spnlp-1.2

[21] Salesforce. 2020. https://blog.salesforceairesearch.com/talk-to-your-data-one-model-any-database/.

[22] Timo Schick and Hinrich Schütze. 2021. Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty (Eds.). Association for Computational Linguistics, 255–269. https://doi.org/10.18653/v1/2021.eacl-main.20

[23] Timo Schick and Hinrich Schütze. 2021. It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (Eds.). Association for Computational Linguistics, 2339–2352. https://doi.org/10.18653/v1/2021.naacl-main.185

[24] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, 9895–9901. https://doi.org/10.18653/v1/2021.emnlp-main.779

[25] Nathan Schucher, Siva Reddy, and Harm de Vries. 2022. The Power of Prompt Tuning for Low-Resource Semantic Parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, 148–156. https://doi.org/10.18653/v1/2022.acl-short.17

[26] Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional Generalization and Natural Language Variation: Can a Semantic Parsing Approach Handle Both?. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, 922–938. https://doi.org/10.18653/v1/2021.acl-long.75

[27] Richard Shin, Christopher H. Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained Language Models Yield Few-Shot Semantic Parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia,

and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, 7699–7715. https://doi.org/10.18653/v1/2021.emnlp-main.608

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[29] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 3261–3275. https://proceedings.neurips.cc/paper/2019/hash/4496bf24afe7fab6f046bf4923da8de6-Abstract.html

[30] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.). Association for Computational Linguistics, 7567–7578. https://doi.org/10.18653/v1/2020.acl-main.677

[31] 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 of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. The Association for Computer Linguistics, 1332–1342. https://doi.org/10.3115/v1/p15-1129

[32] Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2019. Multi-passage BERT: A Globally Normalized BERT Model for Open-domain Question Answering. 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 2019, Hong Kong, China, November 3-7, 2019*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 5877–5881. https://doi.org/10.18653/v1/D19-1599

[33] Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-Sequence Learning as Beam-Search Optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, Jian Su, Xavier Carreras, and Kevin Duh (Eds.). The Association for Computational Linguistics, 1296–1306. https://doi.org/10.18653/v1/d16-1137

[34] Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir R. Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. UnifiedSKG: Unifying and Multi-Tasking Structured Knowledge Grounding with Text-to-Text Language Models. *CoRR* abs/2201.05966 (2022). arXiv:2201.05966 https://arxiv.org/abs/2201.05966

[35] Peng Xu, Dhruv Kumar, Wei Yang, Wenjie Zi, Keyi Tang, Chenyang Huang, Jackie Chi Kit Cheung, Simon J. D. Prince, and Yanshuai Cao. 2021. Optimizing Deeper Transformers on Small Datasets. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, 2089–2102. https://doi.org/10.18653/v1/2021.acl-long.163

[36] Jingfeng Yang, Haoming Jiang, Qingyu Yin, Danqing Zhang, Bing Yin, and Diyi Yang. 2022. SEQZERO: Few-shot Compositional Semantic Parsing with Sequential Prompts and Zero-shot Models. In *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz (Eds.). Association for Computational Linguistics, 49–60. https://doi.org/10.18653/v1/2022.findings-naacl.5

[37] Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir R. Radev, Richard Socher, and Caiming Xiong. 2021. GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=kyaIeYj4zZ

[38] Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander R. Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter S. Lasecki, and Dragomir R. Radev. 2019. CoSQL: A Conversational Text-to-SQL Challenge Towards Cross-Domain Natural Language Interfaces to Databases. 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 2019, Hong Kong, China, November 3-7, 2019*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics,

1962–1979. https://doi.org/10.18653/v1/D19-1204

[39] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 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, Brussels, Belgium, October 31 - November 4, 2018*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, 3911–3921. https://doi.org/10.18653/v1/d18-1425

[40] John M. Zelle and Raymond J. Mooney. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 2*, William J. Clancey and Daniel S. Weld (Eds.). AAAI Press / The MIT Press, 1050–1055. http://www.aaai.org/Library/AAAI/1996/aaai96-156.php

[41] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 11328–11339. http://proceedings.mlr.press/v119/zhang20ae.html

[42] Hao Zheng and Mirella Lapata. 2021. Compositional Generalization via Semantic Tagging. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, 1022–1032. https://doi.org/10.18653/v1/2021.findings-emnlp.88

Received October 2022; revised January 2023; accepted February 2023