

Towards Generalizable and Robust Text-to-SQL Parsing*

Chang Gao¹, Bowen Li², Wenzuan Zhang², Wai Lam^{1†}, Binhua Li²,
Fei Huang², Luo Si² and Yongbin Li^{2†}

¹The Chinese University of Hong Kong

²DAMO Academy, Alibaba Group

{gaochang,wlam}@se.cuhk.edu.hk, libowen.ne@gmail.com

{saike.zwx, binhua.lbh, shuide.lyb}@alibaba-inc.com

Abstract

Text-to-SQL parsing tackles the problem of mapping natural language questions to executable SQL queries. In practice, text-to-SQL parsers often encounter various challenging scenarios, requiring them to be generalizable and robust. While most existing work addresses a particular generalization or robustness challenge, we aim to study it in a more comprehensive manner. In specific, we believe that text-to-SQL parsers should be (1) **generalizable** at three levels of generalization, namely *i.i.d.*, *zero-shot*, and *compositional*, and (2) **robust** against input perturbations. To enhance these capabilities of the parser, we propose a novel TKK framework consisting of Task decomposition, Knowledge acquisition, and Knowledge composition to learn text-to-SQL parsing in stages. By dividing the learning process into multiple stages, our framework improves the parser’s ability to acquire general SQL knowledge instead of capturing spurious patterns, making it more generalizable and robust. Experimental results under various generalization and robustness settings show that our framework is effective in all scenarios and achieves state-of-the-art performance on the Spider, SParC, and CoSQL datasets. Code can be found at <https://github.com/AlibabaResearch/DAMO-ConvAI/tree/main/tkk>.

1 Introduction

Text-to-SQL parsing aims to translate natural language questions to SQL queries that can be executed on databases to produce answers (Lin et al., 2020), which bridges the gap between expert programmers and ordinary users who are not proficient in writing SQL queries. Thus, it has drawn great

*Work done when Chang Gao was an intern at Alibaba. The work described in this paper is substantially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Code: 14204418).

†Corresponding authors.

attention in recent years (Zhong et al., 2017; Suhr et al., 2020; Scholak et al., 2021; Hui et al., 2022; Qin et al., 2022a,b).

Early work in this field (Zelle and Mooney, 1996; Yaghmazadeh et al., 2017; Iyer et al., 2017) mainly focuses on *i.i.d. generalization*. They only use a single database, and the exact same target SQL query may appear in both the training and test sets. However, it is difficult to collect sufficient training data to cover all the questions users may ask (Gu et al., 2021) and the predictions of test examples might be obtained by semantic matching instead of semantic parsing (Yu et al., 2018b), limiting the generalization ability of parsers. Subsequent work further focuses on generalizable text-to-SQL parsing in terms of two aspects: *zero-shot generalization* and *compositional generalization*. Zero-shot generalization requires the parser to generalize to unseen database schemas. Thanks to large-scale datasets such as Spider (Yu et al., 2018b), SParC (Yu et al., 2019b), and CoSQL (Yu et al., 2019a), zero-shot generalization has been the most popular setting for text-to-SQL parsing in recent years. Various methods involving designing graph-based encoders (Wang et al., 2020; Cao et al., 2021) and syntax tree decoders (Yu et al., 2018a; Rubin and Berant, 2021) have been developed to tackle this challenge. Compositional generalization is the desired ability to generalize to test examples consisting of novel combinations of components observed during training. Finegan-Dollak et al. (2018) explore compositional generalization in text-to-SQL parsing focusing on template-based query splits. Shaw et al. (2021) provide new splits of Spider considering length, query template, and query compound divergence to create challenging evaluations of compositional generalization.

Another challenge of conducting text-to-SQL parsing in practice is *robustness*. Existing text-to-SQL models have been found vulnerable to input perturbations (Deng et al., 2021; Gan et al., 2021a;

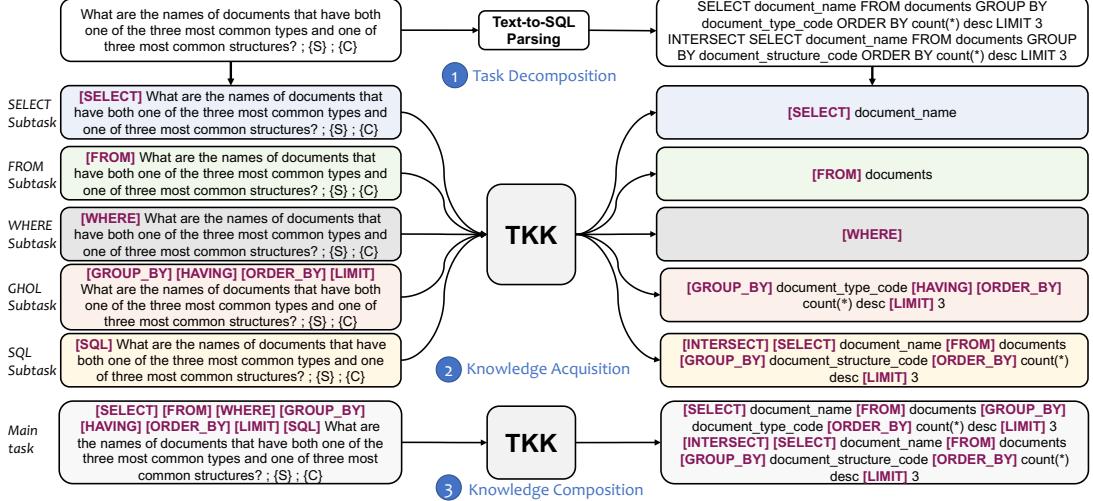


Figure 1: Overview of our TKK framework. {S} and {C} denote the database schema and context, respectively.

Pi et al., 2022). For example, Gan et al. (2021a) replace schema-related words in natural language questions with manually selected synonyms and observe a dramatic performance drop. They propose two approaches, namely multi-annotation selection and adversarial training, to improve model robustness against synonym substitution.

Although specialized model architectures and training approaches have been proposed to address a particular generalization or robustness challenge, we believe that practical text-to-SQL parsers should be built with strong generalizability in terms of **all three levels of generalization**, namely *i.i.d.*, **zero-shot**, and **compositional**, and **robustness** against input perturbations. To obtain such capabilities, it can be noticed that humans often learn to write each clause, such as `SELECT` or `WHERE`, for a basic operation, before composing them to fulfill a more challenging goal, i.e., writing the entire SQL query. In contrast, most existing methods adopt a one-stage learning paradigm, i.e., learning to write each SQL clause and the dependency between different clauses simultaneously. This may lead the model to capture spurious patterns between the question, database schema, and SQL query instead of learning general SQL knowledge.

To this end, we propose a novel framework consisting of three learning stages including **Task decomposition**, **Knowledge acquisition**, and **Knowledge composition** (TKK) for text-to-SQL parsing, which mimics the human learning procedure to learn to handle the task in stages. Specifically, in the task decomposition stage, TKK decomposes the original task into several subtasks.

Each subtask corresponds to mapping the natural language question to one or more clauses of the SQL query, as shown in the top portion of Figure 1. Afterwards, TKK features a prompt-based learning strategy to separately acquire the knowledge of subtasks and employ the learned knowledge to tackle the main task, i.e., generating the entire SQL query. In the knowledge acquisition stage, TKK trains the model with all the subtasks in a multi-task learning manner; in the knowledge composition stage, TKK fine-tunes the model with the main task to combine the acquired knowledge of subtasks and learn the dependency between them.

The advantages of our three-stage framework over previous one-stage learning methods are three-fold: (1) it reduces the difficulty of model learning by dividing the learning process into multiple easier-to-learn stages; (2) it explicitly forces the model to learn the alignment between the question, database schema, and each SQL clause as it needs to identify the intent expressed in the question based on the schema to generate a specific clause; (3) by explicitly constructing the training data for each subtask, it is easier for the model to learn the knowledge required to translate the question into each SQL clause. These advantages help the model to learn general SQL knowledge rather than some dataset-specific patterns, making it more generalizable and robust.

To verify the effectiveness of our framework, we conduct comprehensive evaluations on representative benchmarks covering all three levels of generalization and robustness scenarios with pre-trained sequence-to-sequence models. Experi-

tal results and analysis show that: (1) we achieve state-of-the-art performance on the Spider, SParC, and CoSQL datasets; (2) our method outperforms vanilla sequence-to-sequence models in all scenarios; (3) our framework significantly improves the model’s ability to generate complex SQL queries; (4) our framework is also effective in the low-resource setting.

2 Background

Notations We use the lowercase letter q to denote a natural language question and denote its corresponding database schema, context, and SQL query as s_q , c_q , and l_q , respectively. We represent the set of training examples (q, s_q, c_q, l_q) as \mathcal{D}_{train} and test set as \mathcal{D}_{test} . A perturbed test set \mathcal{D}'_{test} could be constructed by perturbations to questions such as synonym substitution to form (q', s_q, c_q, l_q) . We denote \mathcal{S}_{train} as the set of database schemas of \mathcal{D}_{train} , \mathcal{L}_{train} as the set of SQL queries of \mathcal{D}_{train} , and \mathcal{Q}_{test} as the set of questions of \mathcal{D}_{test} .

Problem Definition Given (q, s_q, c_q) , where the database schema s_q consists of tables and columns, and context c_q is the interaction history consisting of previous questions and system clarification in the multi-turn setting or empty in the single-turn setting, the goal is to generate the correct SQL query l_q .

Generalization and Robustness Following Gu et al. (2021) and Wang et al. (2022b), we formalize three levels of generalization and robustness as follows:

Zero-shot generalization: $\forall q \in \mathcal{Q}_{test}, s_q \notin \mathcal{S}_{train}$.

Compositional generalization: $\forall q \in \mathcal{Q}_{test}, s_q \in \mathcal{S}_{train}, l_q \notin \mathcal{L}_{train}$.

I.I.D. generalization: $\forall q \in \mathcal{Q}_{test}, s_q \in \mathcal{S}_{train}$. \mathcal{D}_{train} and \mathcal{D}_{test} follow the same distribution.

Robustness: training with \mathcal{D}_{train} but adopting \mathcal{D}'_{test} instead of \mathcal{D}_{test} for evaluation.

3 Our TKK Framework

TKK consists of three learning stages: task decomposition, knowledge acquisition, and knowledge composition. In this section, we first introduce each stage in detail. Then we describe the training and inference of TKK.

3.1 Three Stages of TKK

Task Decomposition As shown in Figure 1, we decompose the text-to-SQL parsing task into five subtasks, namely SELECT, FROM, WHERE, GHOL, and SQL. Basically, a subtask aims to translate the natural language question to one or more clauses of the SQL query. For example, the GHOL subtask aims to generate the GROUP_BY, HAVING, ORDER_BY, and LIMIT clauses given the question and its corresponding database schema and context. For queries involving set operators such as INTERSECT, UNION, and EXCEPT to combine two SQL queries, we treat the first query as usual and the second query as the SQL clause of the first query. The SQL subtask targets at mapping the question to the SQL clause.

There are two considerations behind constructing a subtask: (1) the number of classification examples; (2) the dependency between different clauses. First, according to the SQL syntax, every SQL has the SELECT and FROM clauses. However, clauses such as GROUP_BY and ORDER_BY appear only in relatively complicated SQL queries. It implies that the number of these clauses is much smaller than that of the SELECT or FROM clause. Trivially considering generating each clause as a subtask is problematic. If a specific clause does not exist, the generation task degenerates to a classification task because the model only needs to judge its existence. We denote these examples as *classification examples*. Too many classification examples are harmful to model learning. Second, the GROUP_BY and HAVING clauses are usually bundled together, which is also the case of the ORDER_BY and LIMIT clauses. The ORDER_BY clause is often dependent on the GROUP_BY clause if they appear in a SQL query simultaneously. Based on the above observations, combining these clauses to construct a single subtask is more appropriate. We do not further decompose the SQL clause because there will be more subtasks, and most training examples of these subtasks are classification examples.

Knowledge Acquisition In this stage, we train the sequence-to-sequence model with all subtasks using multi-task learning. We assign each SQL keyword a special token, which is also used to denote its corresponding clause. Then we construct a task prompt for each subtask based on the clauses it contains. For example,

the special token corresponding to GROUP_BY is “[GROUP_BY]” and the prompt for the GHOL subtask is “[GROUP_BY] [HAVING] [ORDER_BY] [LIMIT]”. The input for each subtask simply adds a task prompt to the input for the original task.

For constructing the target, we replace the keywords in each clause with their corresponding special tokens. If a clause is empty, we use its corresponding special token to build the target. For instance, the example in Figure 1 does not contain the WHERE clause. Thus the target of the WHERE subtask is “[WHERE]”. Those examples whose targets only contain special tokens are *classification examples*, as we mentioned earlier. For those examples whose targets contain at least one non-empty clause, we regard them as *parsing examples*. Classification examples are helpful since the model needs to learn which clauses to generate given a particular question. However, too many classification examples make it difficult for the model to learn the knowledge of subtasks. Even though we pack the GROUP_BY, HAVING, ORDER_BY, and LIMIT clauses into one subtask, the number of classification examples is still much bigger than that of parsing examples. The SQL subtask also has the problem. To tackle this problem, we downsample classification examples for each subtask to guarantee that the proportion of parsing examples is at least a ratio r .

Knowledge Composition Training the model with multiple subtasks cannot capture the interdependency between them. In this stage, we fine-tune the model with the main task, i.e., generating the entire SQL query, to capture such information. As shown in Figure 1, we combine the prompts of subtasks to construct the prompt of the main task to guide the model to composite the knowledge of subtasks.

3.2 Training and Inference

We formulate text-to-SQL parsing as a sequence-to-sequence generation problem. The input is the serialization of the question, database schema, and context, and the output is the SQL query. In the knowledge acquisition and composition stages, we adjust the input and output according to what we discussed in the last section. We adopt the pre-trained sequence-to-sequence model T5 (Raffel et al., 2020) as the backbone of TKK.

Models	EM	EX
Global-GNN (Bogin et al., 2019)	52.7	-
IRNet + BERT (Guo et al., 2019)	63.9	-
RATSQL + BERT (Wang et al., 2020)	69.7	-
RYANSQL + BERT (Choi et al., 2021)	70.6	-
RATSQL + GraPPa (Yu et al., 2021a)	73.4	-
LGEQL + ELECTRA (Cao et al., 2021)	75.1	-
T5-Base [†] (Raffel et al., 2020)	58.1	60.1
T5-Large [†] (Raffel et al., 2020)	66.6	68.3
T5-3B [†] (Raffel et al., 2020)	71.8	74.4
T5-Base + PICARD (Scholak et al., 2021)	65.8	68.4
T5-Large + PICARD (Scholak et al., 2021)	69.1	72.9
T5-3B + PICARD (Scholak et al., 2021)	75.5	79.3
TKK-Base	61.5	64.2
TKK-Large	70.6	73.2
TKK-3B	74.2	78.4
TKK-Base + PICARD	70.4	76.0
TKK-Large + PICARD	74.1	78.2
TKK-3B + PICARD	75.6	80.3

Table 1: Zero-shot generalization results on Spider.
[†]: Results are taken from (Xie et al., 2022).

Training The model is trained with a maximum likelihood objective. Given the training example (q, s_q, c_q, tp, y) , the loss function L_θ is defined as

$$L_\theta = - \sum_{i=1}^n \log P_\theta(y_i | y_{<i}, q, s_q, c_q, tp) \quad (1)$$

where θ is the model parameters, tp is the task prompt, y is the target sequence, and n is the length of y . In the knowledge acquisition stage, we mix the data of all subtasks for training. In the knowledge composition stage, we initialize the model with the weights of the model trained in the knowledge acquisition stage and use the data of the main task for training.

Inference After training, for each triple of the question, database schema, and context (q, s_q, c_q) , we generate the target sequence of the main task for obtaining the SQL query. We replace the special tokens in the target sequence with their corresponding SQL keywords.

4 Experiments

4.1 Experimental Setup

Datasets For *zero-shot generalization*, we use the original Spider (Yu et al., 2018b), SParC (Yu et al., 2019b), and CoSQL (Yu et al., 2019a)

Models	SParC		CoSQL	
	QM	IM	QM	IM
EditSQL + BERT (Zhang et al., 2019)	47.2	29.5	39.9	12.3
IGSQL + BERT (Cai and Wan, 2020)	50.7	32.5	44.1	15.8
R ² SQL + BERT (Hui et al., 2021a)	54.1	35.2	45.7	19.5
RAT-SQL + SCoRe (Yu et al., 2021b)	62.2	42.5	52.1	22.0
HIE-SQL + GraPPa (Zheng et al., 2022)	64.7	45.0	56.4	28.7
T5-Base† (Raffel et al., 2020)	50.6	31.3	42.3	12.6
T5-Large† (Raffel et al., 2020)	56.7	37.4	48.3	16.7
T5-3B† (Raffel et al., 2020)	61.5	41.9	54.1	22.8
T5-3B + PICARD (Scholak et al., 2021)	-	-	56.9	24.2
TKK-Base	52.6	32.7	46.9	17.8
TKK-Large	60.2	41.0	50.5	21.5
TKK-3B	65.5	46.7	54.9	24.9
TKK-3B + PICARD	66.6	48.3	58.3	27.3

Table 2: Zero-shot generalization results on SParC and CoSQL. [†]: Results are taken from (Xie et al., 2022).

Models	Spider-Template		Spider-Length		Spider-TMCD	
	EM	EX	EM	EX	EM	EX
T5-Base† (Raffel et al., 2020)	59.3	-	49.0	-	60.9	-
T5-3B† (Raffel et al., 2020)	64.8	-	56.7	-	69.6	-
NQG-T5-3B (Shaw et al., 2021)	64.7	-	56.7	-	69.5	-
TKK-Base	62.9	69.8	52.0	55.3	63.3	71.3
TKK-3B	70.3	77.2	58.6	63.3	71.8	79.1

Table 3: Results on three compositional splits of Spider. [†]: Results are taken from (Shaw et al., 2021).

datasets¹. Spider is a single-turn dataset, while SParC and CoSQL are multi-turn datasets. For *compositional generalization*, we use three compositional splits derived from Spider, namely template split (Spider-Template), length split (Spider-Length), and Target Maximum Compound Divergence (TMCD) split (Spider-TMCD), from Shaw et al. (2021). For *i.i.d. generalization*, we construct Spider-IID, SParC-IID, and CoSQL-IID based on Spider, SParC, and CoSQL, respectively. For example, to obtain Spider-IID, we mix the training and development sets of Spider to get the full set and then randomly sample from it to construct new training and development sets while retaining the ratio of the number of original training and development examples.

For *robustness*, we use Spider-Syn (Gan et al., 2021a) and Spider-Realistic (Deng et al., 2021) for evaluation. Spider-Syn is constructed via modifying questions in Spider using synonym substitution. Spider-Realistic selects a complex subset from the development

set of Spider and modifies the questions in this subset to remove or paraphrase explicit mentions of column names while keeping the SQL queries unchanged.

Evaluation Metrics For Spider and datasets derived from it, we use Exact Match (EM) and Execution Accuracy (EX) following Yu et al. (2018b). For SParC, CoSQL, and datasets derived from them, we use Question Match (QM) and Interaction Match (IM) following Yu et al. (2019b).

Implementation Details TKK has three model sizes: TKK-Base, TKK-Large, and TKK-3B, which are initialized with pre-trained T5-Base, T5-Large, and T5-3B models (Raffel et al., 2020), respectively. We use the same Question-Schema-Context serialization as in (Xie et al., 2022). We set the maximum input length to 512, the maximum target length to 128, and the batch size to 32. We use the Adafactor (Shazeer and Stern, 2018) optimizer for all experiments. We set the learning rate to 1e-4 for TKK-Base and TKK-Large and 5e-5 for TKK-3B and use linear

¹Since the test sets of these datasets are not public, we report results on the development sets.

learning rate decay. In addition, we choose the data balance ratio r from {0.5, 0.7, 0.9}. All experiments are done on NVIDIA Tesla A100 and V100.

4.2 Generalization Results

Tables 1 and 2 report the zero-shot generalization results on Spider, SParC, and CoSQL, respectively. Equipped with PICARD (Scholak et al., 2021), which constrains the decoders to generate valid SQL queries by rejecting inadmissible tokens, TKK-3B achieves state-of-the-art results on these three datasets, demonstrating the strong *zero-shot generalization* ability of our framework. Noticeably, TKK outperforms T5 on all datasets and all model sizes. Zero-shot generalization is challenging as it requires the model to accurately understand a question conditioned on an unseen database schema to generate the correct SQL query. As a result, the model has to acquire general SQL knowledge rather than trivially memorize seen SQL patterns. Our framework forces the model to align the question, database schema, and each SQL clause and helps the model to learn SQL knowledge, thus leading to better generalization performance. In addition, as shown in Table 1, TKK-Base with PICARD achieves comparable performance to strong specialized models such as RYANSQL, indicating the great potential of pre-trained sequence-to-sequence models for text-to-SQL parsing. Note that previous state-of-the-art models such as LGEQL and HIE-SQL heavily rely on manual design and may overfit to specific datasets. On the contrary, our framework enjoys strong generality as well as effectiveness.

Table 3 presents the results on the three compositional splits of Spider. TKK outperforms T5 on all the three splits, demonstrating its powerful *compositional generalization* ability. By comparison, NQG-T5, which combines a grammar-based approach NQG with T5, shows no gain over T5 on these datasets. Spider-Template and Spider-TMCD require the model to generalize to novel templates and atom combinations, respectively, while Spider-Length requires the model to generalize to longer outputs. By explicitly decomposing the original task into multiple subtasks and combining the knowledge of them, our framework enables the model to better learn SQL knowledge and makes it less sensitive to these changes.

Table 4 shows the results of TKK and the strong

baseline T5 model on Spider-IID, SParC-IID, and CoSQL-IID. TKK obtains better results than T5 on all three datasets, demonstrating our framework’s strong *i.i.d. generalization* ability. It can be seen that *i.i.d. generalization* is not as challenging as the other two generalization scenarios. However, the results on SParC-IID and CoSQL-IID are still not satisfactory. Enhancing the model’s ability to acquire general knowledge is also helpful and necessary in this setting.

4.3 Robustness Results

Table 5 reports the results of various models trained on Spider and evaluated on Spider, Spider-Syn, and Spider-Realistic, which measures the model’s robustness against perturbations to natural language questions. We have the following observations: (1) T5 is more robust than specialized models. For example, when evaluated on Spider-Syn, RATSQ + BERT degrades by 21.5 absolute points on EM, while T5-3B sees a performance drop of 12.2 absolute points. T5-Large, which performs worse than RATSQ + BERT on Spider, can obtain better performance than it on Spider-Syn. This indicates that models specially designed for Spider are prone to overfitting on it. Thus evaluating their robustness is important. (2) Our TKK framework can improve the robustness of T5 for text-to-SQL parsing. TKK outperforms T5 on both Spider-Syn and Spider-Realistic for all model sizes. (3) STRUG improves robustness via structure-grounded pre-training with a large-scale of text-table paired data, while TKK provides a better way for learning text-to-SQL parsing to achieve this and does not need any additional data. (4) For pre-trained sequence-to-sequence models, the larger the model is, the more robust it is. When the model becomes larger, the gap between the performance of TKK on Spider-Syn and Spider-Realistic and the performance on Spider narrows. The same trend can be seen with T5.

5 More Analysis

Is each subtask necessary in the knowledge acquisition stage? To quantify the contribution of each subtask, we examine the performance of the main task after removing a subtask for training in the knowledge acquisition stage. Table 6 shows the ablation results on Spider and CoSQL. Removing any subtask degrades the model’s performance

Models	Spider-IID		SParC-IID		CoSQL-IID	
	EM	EX	QM	IM	QM	IM
T5-Base (Raffel et al., 2020)	84.1	86.2	68.3	44.6	47.9	17.8
T5-Large (Raffel et al., 2020)	86.9	88.5	70.0	49.1	52.9	23.6
TKK-Base	86.6	88.1	70.3	46.9	51.5	22.2
TKK-Large	88.3	89.8	72.3	52.6	56.9	27.3

Table 4: Results on three datasets for i.i.d. generalization: Spider-IID, SParC-IID, and CoSQL-IID.

Models	Spider		Spider-Syn		Spider-Realistic	
	EM	EX	EM	EX	EM	EX
IRNet (Guo et al., 2019)	53.2	-	28.4	-	-	-
RAT-SQL + BERT (Wang et al., 2020)	69.7	-	48.2	-	58.1	62.1
RAT-SQL + STRUG (Deng et al., 2021)	72.6	74.9	-	-	62.2	65.3
T5-Base [†] (Raffel et al., 2020)	56.8	59.9	40.8	43.8	46.9	47.6
T5-Large [†] (Raffel et al., 2020)	66.8	70.9	53.1	57.4	57.7	60.0
T5-3B [†] (Raffel et al., 2020)	71.6	74.5	59.4	65.3	63.2	65.0
TKK-Base	61.5	64.2	44.2	47.7	53.7	53.7
TKK-Large	70.6	73.2	55.1	60.5	64.4	64.4
TKK-3B	74.2	78.4	63.0	68.2	68.5	71.1

Table 5: Results of models trained on Spider and evaluated on Spider, Spider-Syn and Spider-Realistic. [†]: We train T5 models on Spider and report evaluated results on the three datasets, which are different from Table 1.

on the main task, indicating that all subtasks are necessary for the knowledge acquisition stage. We can see that the `FROM` subtask has the largest impact on the performance. This is due to the mismatch between natural language expression and SQL syntax. User questions generally do not involve which tables to retrieve data from, while SQL requires specifying this. The `FROM` subtask allows the model to learn the alignment between the question and the `FROM` clause, thus alleviating the mismatch problem. Although other clauses are more or less mentioned in user questions, there are also alignment issues. Some previous work tackles this problem via designing intermediate representations (Guo et al., 2019; Gan et al., 2021b). Our framework provides a new perspective. The effect of the `SQL` subtask is less pronounced since the number of training examples of it is much smaller than that of the other subtasks.

How effective is knowledge acquisition? We want to investigate if adding training data in the knowledge acquisition stage will further improve the model’s performance. To this end, we first take 5%, 10%, 20%, 40%, and 100% of the data for constructing the training data in the knowledge acquisition stage and then use 5% of the data for finetuning with the main task. The results

Models	Spider		CoSQL	
	EM	EX	QM	IM
TKK-Base	61.5	64.2	46.9	17.8
w/o SELECT	60.0	63.4	43.8	15.0
w/o FROM	60.0	62.0	43.2	14.3
w/o WHERE	60.0	63.4	44.6	16.7
w/o GHOL	60.9	63.1	43.5	16.0
w/o SQL	61.2	63.4	45.3	17.1

Table 6: The effect of subtasks.

on Spider and CoSQL are shown in Figure 2. As the amount of training data in the knowledge acquisition stage increases, the performance of TKK-Base improves significantly. This suggests that pre-training the model with large-scale subtask data will be beneficial to improving the model’s performance.

How effective is knowledge composition? To answer this, we use all data in the knowledge acquisition stage for training and then take 5%, 10%, 20%, 40%, and 100% of the data for finetuning with the main task. As shown in Figure 3, training with more data in the knowledge composition stage is also helpful. Since only using subtasks for training loses the dependency information of

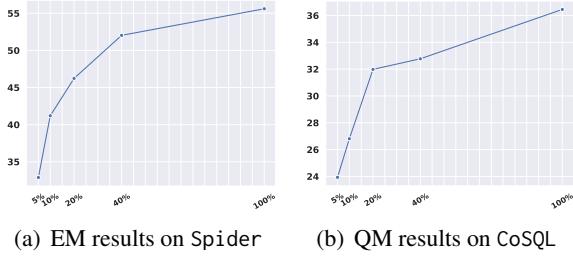


Figure 2: Results of TKK-Base as the amount of training data in the knowledge acquisition stage increases.

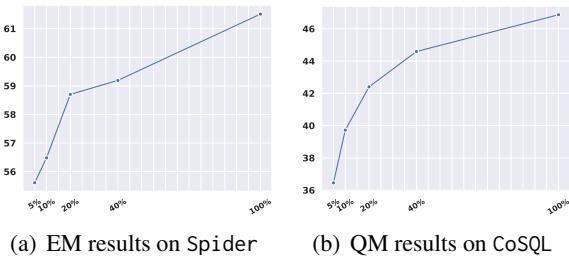


Figure 3: Results of TKK-Base as the amount of training data in the knowledge composition stage increases.

different subtasks, knowledge composition helps the model to capture this information. Moreover, fine-tuning the main task with only 5% data has already achieved 90% and 78% of the performance fine-tuned with all the data on Spider and CoSQL, respectively, showing that the model only needs a small amount of data to learn to tackle the main task.

What is the model’s performance in terms of different hardness levels? SQL queries in Spider can be divided into four hardness levels: easy, medium, hard, and extra hard (Yu et al., 2018b). Table 7 shows a comparison between TKK and T5 regarding these four hardness levels. It can be seen that the performance improvement mainly comes from hard and extra hard examples. For example, TKK-Base and TKK-Large improve T5-Base and T5-Large by 16.3 and 10.9 absolute points on extra hard examples, respectively. By dividing the learning process into multiple stages, our framework dramatically improves the model’s ability to handle complex queries, thus leading to better overall performance.

Is TKK still effective in low-resource scenarios? Another perspective to study the model’s generalization ability is to investigate its performance in the low-resource setting. To this end, we

Models	Easy	Medium	Hard	Extra
T5-Base	83.9	59.9	42.5	22.9
T5-Large	87.5	74.0	50.0	34.3
TKK-Base	83.9	63.0	47.1	39.2
TKK-Large	89.5	76.5	52.9	45.2

Table 7: EM results on Spider in terms of different hardness levels.

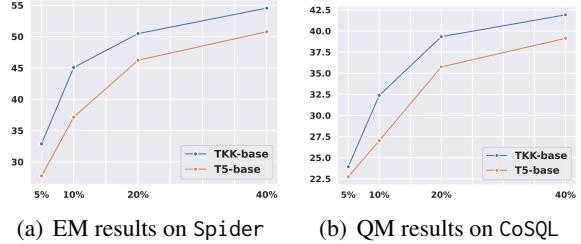


Figure 4: Results of T5-Base and TKK-Base in low-resource scenarios.

conduct experiments on Spider and CoSQL. For each dataset, we randomly shuffle the training set and then take 5%, 10%, 20%, and 40% of the data for training. The results of T5-Base and TKK-Base are shown in Figure 4. TKK-Base performs better than T5-Base no matter how much data is used for training, showing that our framework is also effective in low-resource scenarios.

Case Study We also conduct a case study to show that TKK makes fewer mistakes on SQL clauses and is more robust to synonym substitution compared with T5. Details are in Appendix A.

6 Related Work

Most previous work aims to solve a particular generalization or robustness challenge for text-to-SQL parsing. Dong and Lapata (2016) introduce a sequence-to-tree approach for traditional i.i.d. datasets such as GeoQuery (Zelle and Mooney, 1996). Yu et al. (2018a) propose a syntax tree network to tackle the zero-shot text-to-SQL problem. Later, various methods address the challenge from different perspectives such as improving schema linking (Wang et al., 2020; Hui et al., 2021b; Qin et al., 2021; Wang et al., 2022a), data augmentation (Yu et al., 2021a; Wu et al., 2021), or exploiting history information (Hui et al., 2021a; Zheng et al., 2022). Shaw et al. (2021) combines a grammar-based approach with T5 (Raffel et al., 2020) to address the compositional generalization challenge. Deng et al. (2021) develop a structure-grounded

pre-training framework for improving the model’s robustness against natural language variations. Pi et al. (2022) build an adversarial training example generation framework to bring the model better robustness against table perturbations. However, the success of specialized architectures or training approaches on one challenge cannot easily transfer to others (Herzig et al., 2021; Furrer et al., 2020). Our TKK framework, for the first time, shows improvements in all the concerned challenging scenarios for text-to-SQL parsing.

Our work is also related to the research of task decomposition in NLP (Gao et al., 2022; Nye et al., 2022; Wies et al., 2022; Wei et al., 2022; Wang et al., 2022c). For example, least-to-most prompting (Zhou et al., 2022), a method purely based on inference with a sufficiently large pre-trained language model, reduces a complex task into multiple subtasks and solves these subtasks sequentially. By comparison, TKK first learns to solve simpler subtasks and then the complex task. At inference time, the model directly tackles the complex task.

7 Conclusion

This paper proposes a general and effective TKK framework for text-to-SQL parsing, which has three stages: task decomposition, knowledge acquisition, and knowledge composition. TKK enhances the model’s ability to acquire general SQL knowledge by dividing the learning process into multiple stages. Comprehensive evaluation on three levels of generalization, namely *i.i.d.*, *zero-shot*, and *compositional*, and robustness demonstrates the effectiveness of our framework.

Limitations

Although our TKK framework is conceptually simple, it needs to decompose the task into multiple subtasks manually. It is not difficult to decompose the text-to-SQL parsing task due to the simplicity of SQL syntax. However, decomposing the complex graph structure such as Abstract Meaning Representation (AMR) is not straightforward. Therefore, a general strategy to automatically discover the meaningful substructure of the original task is needed. With such a strategy, our framework can be extended to broader research areas as long as the task can be decomposed into meaningful subtasks. We aim to address this limitation in our future work.

References

- Ben Bogin, Matt Gardner, and Jonathan Berant. 2019. [Global reasoning over database structures for text-to-SQL parsing](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3659–3664.
- Yitao Cai and Xiaojun Wan. 2020. [IGSQL: Database schema interaction graph based neural model for context-dependent text-to-SQL generation](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6903–6912.
- Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. [LGEQL: 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 (Volume 1: Long Papers)*, pages 2541–2555.
- DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2021. [RYANSQL: Recursively applying sketch-based slot fillings for complex text-to-SQL in cross-domain databases](#). *Computational Linguistics*, 47(2):309–332.
- Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2021. [Structure-grounded pretraining for text-to-SQL](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1337–1350.
- Li Dong and Mirella Lapata. 2016. [Language to logical form with neural attention](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43.
- Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. [Improving text-to-SQL evaluation methodology](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360.
- Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. 2020. [Compositional generalization in semantic parsing: Pre-training vs. specialized architectures](#). *arXiv preprint arXiv:2007.08970*.
- Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. 2021a. [Towards robustness of text-to-SQL models against synonym substitution](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language*

- Processing (Volume 1: Long Papers)*, pages 2505–2515.
- Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R. Woodward, John Drake, and Qiaofu Zhang. 2021b. **Natural SQL: Making SQL easier to infer from natural language specifications**. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2030–2042.
- Chang Gao, Wenzuan Zhang, and Wai Lam. 2022. **UniGDD: A unified generative framework for goal-oriented document-grounded dialogue**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 599–605.
- Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. **Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases**. In *Proceedings of the Web Conference 2021*, page 3477–3488.
- Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. **Towards complex text-to-SQL in cross-domain database with intermediate representation**. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535.
- Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, and Yuan Zhang. 2021. **Unlocking compositional generalization in pre-trained models using intermediate representations**. *arXiv preprint arXiv:2104.07478*.
- Binyuan Hui, Ruiying Geng, Qiyu Ren, Binhu Li, Yongbin Li, Jian Sun, Fei Huang, Luo Si, Pengfei Zhu, and Xiaodan Zhu. 2021a. **Dynamic hybrid relation exploration network for cross-domain context-dependent semantic parsing**. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):13116–13124.
- Binyuan Hui, Ruiying Geng, Lihan Wang, Bowen Qin, Yanyang Li, Bowen Li, Jian Sun, and Yongbin Li. 2022. **S²SQL: Injecting syntax to question-schema interaction graph encoder for text-to-SQL parsers**. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1254–1262.
- Binyuan Hui, Xiang Shi, Ruiying Geng, Binhu Li, Yongbin Li, Jian Sun, and Xiaodan Zhu. 2021b. **Improving text-to-sql with schema dependency learning**. *arXiv preprint arXiv:2103.04399*.
- 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 (Volume 1: Long Papers)*, pages 963–973.
- Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. **Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing**. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4870–4888.
- Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2022. **Show your work: Scratchpads for intermediate computation with language models**. In *Deep Learning for Code Workshop*.
- Xinyu Pi, Bing Wang, Yan Gao, Jiaqi Guo, Zhoujun Li, and Jian-Guang Lou. 2022. **Towards robustness of text-to-SQL models against natural and realistic adversarial table perturbation**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2007–2022.
- Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhu Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, et al. 2022a. **A survey on text-to-sql parsing: Concepts, methods, and future directions**. *arXiv preprint arXiv:2208.13629*.
- Bowen Qin, Lihan Wang, Binyuan Hui, Ruiying Geng, Zheng Cao, Min Yang, Jian Sun, and Yongbin Li. 2021. **Sdcup: Schema dependency-enhanced curriculum pre-training for table semantic parsing**. *arXiv preprint arXiv:2111.09486*.
- Bowen Qin, Lihan Wang, Binyuan Hui, Bowen Li, Xiangpeng Wei, Binhu Li, Fei Huang, Luo Si, Min Yang, and Yongbin Li. 2022b. **SUN: Exploring intrinsic uncertainties in text-to-SQL parsers**. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5298–5308.
- 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**. *Journal of Machine Learning Research*, 21(140):1–67.
- Ohad Rubin and Jonathan Berant. 2021. **SmBoP: Semi-autoregressive bottom-up semantic parsing**. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 311–324.
- 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*, pages 9895–9901.
- 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 (Volume 1: Long Papers)*, pages 922–938.
- Noam Shazeer and Mitchell Stern. 2018. [Adafactor: Adaptive learning rates with sublinear memory cost](#). In *Proceedings of the 35th International Conference on Machine Learning*, pages 4596–4604.
- Alane Suhr, Ming-Wei Chang, Peter Shaw, and Kenton Lee. 2020. [Exploring unexplored generalization challenges for cross-database semantic parsing](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8372–8388.
- 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*, pages 7567–7578.
- Lihan Wang, Bowen Qin, Binyuan Hui, Bowen Li, Min Yang, Bailin Wang, Binhu Li, Jian Sun, Fei Huang, Luo Si, and Yongbin Li. 2022a. [Proton: Probing schema linking information from pre-trained language models for text-to-sql parsing](#). In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 1889–1898.
- Xuezhi Wang, Haohan Wang, and Difyi Yang. 2022b. [Measure and improve robustness in NLP models: A survey](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4569–4586.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022c. [Self-consistency improves chain of thought reasoning in language models](#). *arXiv preprint arXiv:2203.11171*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. [Chain of thought prompting elicits reasoning in large language models](#). *arXiv preprint arXiv:2201.11903*.
- Noam Wies, Yoav Levine, and Amnon Shashua. 2022. [Sub-task decomposition enables learning in sequence to sequence tasks](#). *arXiv preprint arXiv:2204.02892*.
- Kun Wu, Lijie Wang, Zhenghua Li, Ao Zhang, Xinyan Xiao, Hua Wu, Min Zhang, and Haifeng Wang. 2021. [Data augmentation with hierarchical SQL-to-question generation for cross-domain text-to-SQL parsing](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8974–8983.
- Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. 2022. [Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models](#). *arXiv preprint arXiv:2201.05966*.
- Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. [Sqlizer: query synthesis from natural language](#). *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):1–26.
- Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, bailin wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, richard socher, and Caiming Xiong. 2021a. [Gra{pp}a: Grammar-augmented pre-training for table semantic parsing](#). In *International Conference on Learning Representations*.
- Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018a. [SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663.
- 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 Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019a. [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)*, pages 1962–1979.
- Tao Yu, Rui Zhang, Alex Polozov, Christopher Meek, and Ahmed Hassan Awadallah. 2021b. [{SC}ore: Pre-training for context representation in conversational semantic parsing](#). In *International Conference on Learning Representations*.
- Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018b. [Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.
- Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019b. [SParC: Cross-domain semantic parsing in context](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4511–4523.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. **Editing-based SQL query generation for cross-domain context-dependent questions**. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5338–5349.

Yanzhao Zheng, Haibin Wang, Baohua Dong, Xingjun Wang, and Changshan Li. 2022. **HIE-SQL: History information enhanced network for context-dependent text-to-SQL semantic parsing**. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2997–3007.

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

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. 2022. **Least-to-most prompting enables complex reasoning in large language models**. *arXiv preprint arXiv:2205.10625*.

A Case Study

Table 8 shows some real cases including questions and the SQL queries generated by T5-3B and TKK-3B. The first four cases are from the Spider dataset. It can be seen that TKK can produce correct SQL queries, while T5 makes mistakes on some clauses. The last two cases are from the Spider-Syn dataset. After synonym substitution, T5 is confused by synonyms and outputs some tables and columns that do not exist in the database, leading to incorrect SQL queries. By comparison, TKK can still identify correct tables and columns.

Spider

Question: What is the average GNP and total population in all nations whose government is US territory?

T5-3B: SELECT avg(gnp), **avg**(population) FROM country WHERE governmentform = "US Territory"

TKK-3B: SELECT avg(gnp), **sum**(population) FROM country WHERE governmentform = "US Territory"

Question: Which model of the car has the minimum horsepower?

T5-3B: SELECT model FROM cars_data ORDER BY horsepower LIMIT 1

TKK-3B: SELECT **car_names**.model FROM cars_data **JOIN** **car_names** **on** cars_data.id = **car_names**.makeid ORDER BY horsepower LIMIT 1

Question: What are the dog name, age and weight of the dogs that were abandoned? Note that 1 stands for yes, and 0 stands for no in the tables.

T5-3B: SELECT name, age, weight FROM dogs WHERE abandoned_yn = 0

TKK-3B: SELECT name, age, weight FROM dogs WHERE abandoned_yn = 1

Question: Return the different document ids along with the number of paragraphs corresponding to each, ordered by id.

T5-3B: SELECT document_id, count(*) FROM paragraphs GROUP BY document_id ORDER BY **count**(*)

TKK-3B: SELECT document_id, count(*) FROM paragraphs GROUP BY document_id ORDER BY **document_id**

Spider-Syn

Question-O: How many templates do we have?

Question-S: How many layout do we have?

T5-3B: SELECT count(*) FROM **layout**

TKK-3B: SELECT count(*) FROM **templates**

Question-O: What is the year that had the most concerts?

Question-S: What is the time that had the most shows?

T5-3B: SELECT **time** FROM concert GROUP BY **time** ORDER BY count(*) desc LIMIT 1

TKK-3B: SELECT **year** FROM concert GROUP BY **year** ORDER BY count(*) desc LIMIT 1

Table 8: Case study. Question-O is the question in the original Spider dataset. Question-S is the question in the Spider-Syn dataset, modified from Question-O using synonym substitution.