Learning to Recover Bridge Path for SQL Schema Linking via Tentative Answer Profiling

Wenzhong Qiu[†], **Shixin Liu**[†], **Yu Hong**[*]
School of Computer Science and Technology, Soochow University, Suzhou, China
{wenzh.qiu, sido.meet, tianxianer}@gmail.com

Abstract

Schema linking is a task of investigating relevant SQL schema (items and paths) for a specific query language. Constructing a reliable linker is crucial for Text2SQL auto-coding since it provides informative clues for directing SQL coders. Current studies have obtained considerable performance gains when independently evaluating linkers, which allow $F1$-score to be higher than 90% at best. However, our observation illustrates that the existing linkers still suffers from a high missing rate of **Bridge Path**, i.e., omitting one or a series of SQL keywords that link across SQL tables. To address the issue, we propose to enhance linkers by Bridge Path Recovering (abbr., BPR) via Tentative Answer Profiling (TAP). On the one hand, BPR is implemented by learning to rank up the better bridged schemata during finetuning. On the other hand, TAP tentatively conjectures possible answers for the query language, and generates their profiles (types and CoT steps) to guide BPR. This is inspired by the experience that answer conjecturing contributes to reversely deducing the multi-hop routing path over SQL tables. Technologically, we train LLaMA-3.1 for TAP, where teacher-student distillation is conducted via Deepseek-3. To ensure sufficient ranking data used for training BPR, we employ a multilevel retrieval technique to align profiles with diverse schemata. We experiment on the benchmark datasets BIRD and Spider. The experimental results show that our approach (BPR-TAP) consistently yields improvements for $F1$-score upon all benchmarks, and surpasses multiple competitive schema linkers. All source codes are available at https://github.com/qwz-web/BPR-TAP-Text2SQL.

1 Introduction

Text2SQL auto-coding aims to convert query language (e.g., a question) into executable SQL codes over relational databases (Zhang et al., 2024; Luo et al., 2025). Current studies (Pourreza and Rafiei, 2024; Talaei et al., 2024) leverage a decomposed pipeline for Text2SQL, which comprises two essential components, including schema linking and SQL code generation. Linker is the crucial preliminary due to its role of exploring code-friendly items and paths from an enormous scale of noisy SQL schemata (Pourreza et al., 2025). We concentrate on the enhancement of schema linking in this paper, and verify its impact on Text2SQL auto-coding.

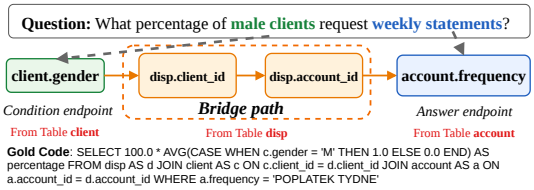


Figure 1: Example of bridge path and its effect.

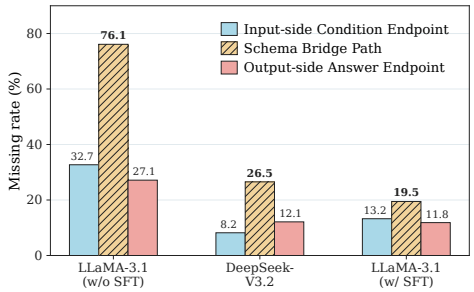


Figure 2: Missing Rates of different types of schema items. Condition-endpoint items serve as filtering constraints, such as that of WHERE and HAVING clauses. Answer endpoints are items needs to be organized, e.g., the SELECTED, GROUPED or ORDERED columns. Bridge Path are Join or Foreign keys for connection.

Linkers generally suffers from the difficulty in salvaging the bridge paths. As shown in Figure 1, a bridge path appears as one or a series of items that relationally connect SQL tables. It discloses the multi-hop routing direction, and therefore being indispensable for instructing the generation of unabridged and unhindered SQL codes. However, bridge paths are easily left out by LLM-based link-

[†] Equal contribution. [*] Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 20413–20429
October 24-29, 2026 ©2026 Association for Computational Linguistics

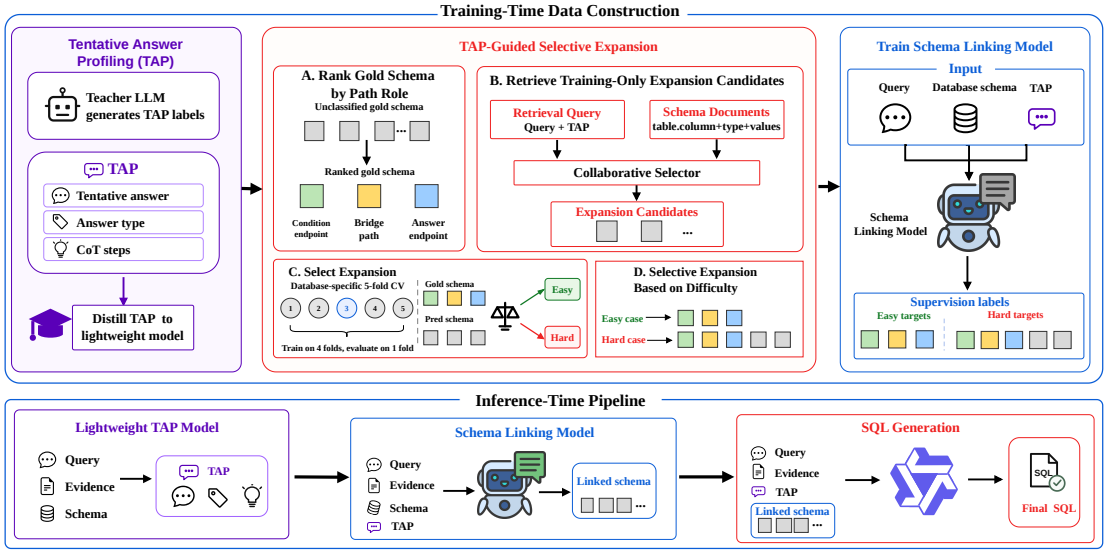


Figure 3: Overview of BPR-TAP for bridge-path-aware schema linking and SQL generation. Teacher-generated Tentative Answer Profiles are distilled into a lightweight TAP model, guide bridge-path recovery during schema-linker training, and provide inference-time context for TAP-conditioned schema linking and SQL generation.

ers. Figure 2 shows the missing rates of bridge paths caused by different LLM-based linkers. It can be found that, compared to other types of schema items, the missing rates of bridge paths are considerably higher (even if finetuning is used).

To address the issue, we propose to enhance linkers by Bridge Path Recovering (BPR) via Tentative Answer Profiling (TAP), which is uniformly denoted by BPR-TAP in experiments. BPR-TAP is essentially a "prompt enhancement" approach. Specifically, a tentative answer profile is added to prompt when LLMs are instructed for linking, where a **profile** comprises 1) a possible answer that was tentatively forecasted for SQL query, 2) answer type and 3) CoT steps produced by LLMs when predicting the tentative answer. This is motivated by the intuitions that tentative answer helps to reversely deduce relevant schema items while, more importantly, CoT steps disclose relational bridges across the items and SQL tables.

Technologically, we train an independent Question Answering (QA) model to generate answer profiles, where LLaMA-3 instruct (8B) (Grattafiori et al., 2024) is used, and teacher-student distillation is performed via Deepseek-3.2 (DeepSeek-AI et al., 2025). We adopt another LLaMA-3 to form schema linker, which is finetuned for linking given both query language and answer profile. In order to implement BPR, we promote the linker to produce better-bridged schemata during finetuning, where schema ranking is considered. To ensure diverse candidate schemata used for ranking, we use the answer profile as query to retrieve different schemata. Multi-level retrieval is utilized, including BM25-based matching (Robertson et al., 1994), Qwen3-Embedding (Zhang et al., 2025) based similarity estimation (0.6B) and reranking (0.6B).

We experiment on Text2SQL benchmarks Spider (Yu et al., 2018) and BIRD (Li et al., 2023). The test results demonstrate the effectiveness and utility of BPR-TAP. Specifically, our contributions are as follows:

- We propose to recover bridge path for schema linking using tentative answer profile, and implement a BPR-TAP optimization approach.
- Test results show that BPR-TAP consistently yields higher $F1$-scores for schema linking, and allows light SQL auto-coders (8B) to achieve higher *Execution Accuracy* (69.69%).
- Our experiments demonstrate that BPR-TAP benefits from substantial improvements in recalling schemata (higher *Recall* rates).

2 Task Formalization

Given a query language q, optional evidence e, and database schemata $\mathcal{S} = \{s_1, \ldots, s_n\}$, schema linking aims to identify the query-relevant schema subset $\hat{\mathcal{S}} \subseteq \mathcal{S}$ used for downstream SQL generation. BPR-TAP augments this task with a Tentative Answer Profile (TAP) z, constructed as described in Section 3.1, and predicts the linked schema as

$$\hat{\mathcal{S}} = f_\theta(q, e, z, \mathcal{S}). \tag{2.1}$$

3 Approach

As illustrated in Figure 3, BPR-TAP is organized into training-time data construction and an inference-time pipeline. During training, a teacher LLM first produces TAP labels, each consisting of a tentative answer, an answer type, and a reasoning clue. These labels are distilled into a lightweight TAP model. We then construct bridge-path-aware supervision for the schema linker. The gold schema is ranked by answer-path roles, namely condition endpoints, bridge paths, and answer endpoints. Training-only expansion candidates are retrieved with the question and teacher-generated TAP, and database-specific 5-fold cross-validation is used to identify easy and hard instances. Only hard instances receive selectively expanded supervision targets, while easy instances keep compact gold targets. The resulting targets train a TAP-conditioned schema-linking model. During inference, the lightweight TAP model generates the TAP, the schema linker predicts the linked schema conditioned on it, and the SQL generator uses the linked schema together with the TAP to produce the final executable SQL code.

3.1 TAP construction

To recover bridge path schema items that are weakly indicated by the query, we construct a TAP $z_i = (a_i, \tau_i, r_i)$ for each instance before schema linking. TAP is not used as a hard answer constraint. Instead, it provides tentative profiling signals that help the schema linker infer missing bridge path schema items between condition endpoints and answer endpoints. Specifically, the tentative answer a_i is tentatively forecasted for the SQL query and helps reversely deduce relevant schema items. The answer type τ_i provides type-level cues and is selected from six categories: TEXT, INTEGER, REAL, DATE, DATETIME, and OTHER. Binary-like BLOB values are mapped to OTHER. The CoT steps r_i are produced by the LLM when predicting the tentative answer. They describe a coarse derivation process from the query conditions to the tentative answer, thereby exposing relational bridges that may be weakly indicated in the query. Case study is provided in Appendix C.

We obtain teacher-generated TAP annotations in the form of $z_i = (a_i, \tau_i, r_i)$ from the input tuple $(q_i, e_i, \mathcal{S}_i)$. Specifically, DeepSeek-V3.2 is used as the teacher LLM to generate TAP labels for training. Each TAP label contains a tentative answer, an answer type, and CoT steps produced during tentative answer prediction. These annotations are then distilled into a lightweight TAP model, namely Llama-3.1-8B-Instruct. At inference time, the TAP model generates z_i from $(q_i, e_i, \mathcal{S}_i)$ and provides the tentative answer profiling for the schema linker.

3.2 Path Decomposition

To characterize schema items by their roles in the answer derivation path, we decompose the gold schema set as $G_i = \mathcal{C}_i \cup \mathcal{B}_i \cup \mathcal{A}_i$, where $\mathcal{C}_i$, $\mathcal{B}_i$, and $\mathcal{A}_i$ denote Input-side Condition Endpoint, Bridge Path, and Output-side Answer Endpoint, respectively. Each schema item is represented as a fully qualified schema unit, e.g., `table.column`. $\mathcal{C}_i$ contains schema items for filtering constraints, mainly those in `WHERE` and `HAVING` clauses. $\mathcal{B}_i$ contains join keys, foreign-key schema items, and intermediate schema items that connect condition-side and answer-side schema items. $\mathcal{A}_i$ contains schema items that determine the returned answer, including `SELECT` columns, aggregation targets, grouping columns, and ordering columns.

This decomposition is used in two ways. **Analysis.** It provides a path-wise view of schema-linking errors. This allows us to examine whether a schema linker mainly misses condition endpoints, bridge paths, or answer endpoints. We use this analysis in experiments to verify whether bridge path items are more vulnerable to local question–schema matching. **Training.** The decomposition is used to construct schema-linking SFT labels. For each instance, we rank the gold schema items into condition endpoints, bridge paths, and answer endpoints. These role-aware labels are then used to train the schema linker, so that it learns to recover schema items along the bridge path.

3.3 TAP-Guided Schema Expansion

Bridge path schema items are easy to miss during linking. To provide training supervision for Bridge Path Recovering, we retrieve answer-implied schema expansion candidates, which can include additional bridge path schema items from multiple possible paths. These candidates help the linker learn weakly aligned but important multi-hop routing paths over SQL tables.

For each training instance i, we form a retrieval query from the query language, optional evidence, and teacher-generated TAP:

$$u_i = [q_i, e_i, z_i]. \tag{3.1}$$

Each schema item $s_j \in \mathcal{S}_i$ is represented as a semi-structured text with its name, type, and sample values. We use a multi-level retrieval strategy to score each schema item. Specifically, BM25 is used for lexical matching, Qwen3-Embedding-0.6B is used to compute dense semantic similarity, and Qwen3-Reranker-0.6B is used to rerank the retrieved candidates. The combined score is used to select the top K_{ret} items as expansion candidates:

$$\mathcal{E}_i = \text{TopK}^{K_{\text{ret}}}_{s_j \in \mathcal{S}_i} \text{Score}(u_i, s_j). \quad (3.2)$$

The set $\mathcal{E}_i$ is used only for constructing SFT training targets and not for inference. Including these candidates encourages the model to output more complete schema. However, the retrieved candidates may also introduce redundant schema items. Details of integrating these candidates into difficulty-aware SFT are illustrated in Section 3.4.

3.4 Difficulty-Aware Selective SFT

The TAP-guided expansion candidates should be incorporated selectively when constructing schema-linking supervision. Easy instances keep the SQL-derived gold schema set, while hard instances receive additional TAP-guided expansion candidates. This selective strategy is used to improve Bridge Path recall while limiting redundant schema items.

Difficulty estimation. We estimate instance difficulty with database-specific five-fold cross-validation on the training databases. The folds are split by database rather than by individual question-SQL examples, so that examples from the same database do not appear in both the training and held-out folds. We construct the five folds to keep the number of training examples approximately balanced across folds, while preserving database-level separation. This prevents database leakage during difficulty estimation.

In each fold, a preliminary schema linker is trained on four folds and predicts schema sets for the held-out fold. Given the gold schema set G_i and the cross-validation prediction $\hat{G}_i^{\text{cv}}$, we set $\delta_i = \text{easy}$ if $\text{Match}(\hat{G}_i^{\text{cv}}, G_i) = 1$, and set $\delta_i = \text{hard}$ otherwise. Here, $\text{Match}(\cdot, \cdot)$ indicates whether the predicted schema set covers the gold schema set. This step is used only for training-data construction and is not required during inference.

Selective target construction. We construct the schema linking SFT data using the answer path decomposition, the TAP-guided expansion candidates, and the difficulty labels. The gold schema set G_i is ranked by its path roles, including condition endpoint, bridge path, and answer endpoint.

Given the difficulty label δ_i, the final SFT target is defined as:

$$T_i^{\text{SL}} = \begin{cases} \text{ser}(G_i), & \delta_i = \text{easy}, \\ \text{ser}(G_i \cup \mathcal{E}_i), & \delta_i = \text{hard}. \end{cases} \quad (3.3)$$

For easy instances, we keep the original gold schemata to avoid redundancy. For hard instances, we merge the TAP-guided expansion candidates with the gold schema set, where the union removes duplicated schema items. The schema linker is trained on $x_i = [q_i, e_i, z_i, \mathcal{S}_i]$ to generate T_i^{SL}.

3.5 TAP-Consistent SQL Generation

After obtaining the linked schema subset $\hat{\mathcal{S}}$, we use it together with the TAP for downstream SQL code generation. The TAP is reused as bridge path guidance, encouraging the SQL generator to remain consistent with the answer-derivation path already captured during the schema linking stage.

Following the previous works (Hao et al., 2025; Pourreza et al., 2025; Liu et al., 2026), we use multiple SQL generators to produce SQL candidates. We then execute valid SQL candidates on the target database and group them by execution results. The final SQL is selected from the top-K_{exec} execution clusters using a consistency-based ranking strategy. Implementation details of this selection procedure are provided in Appendix B.7.

4 Experiments

4.1 Experimental Setup

Datasets. We experiment on Spider (Yu et al., 2018) and BIRD (Li et al., 2023). Spider is a large-scale cross-domain dataset for complex semantic parsing. It contains 10,181 questions paired with 5,693 unique SQL queries. Moreover, BIRD is a large-scale benchmark focusing on realistic, large, and noisy databases. It includes 12,751 question-SQL pairs across 95 databases.

Evaluation. We follow the previous works (Maamari et al., 2024; Hao et al., 2025) to evaluate schema linking and SQL generation tasks. For the evaluation of schema linking, we report count-based Table Recall (TR), count-based Table Precision (TP), Table F1 (TF), count-based Column Recall (CR), count-based Column Precision (CP), Column F1 (CF), and Sample-based Recall (SR). For the evaluation of SQL generation, Execution Accuracy (EX) is used as the metric.

Dataset	Method	TR	TP	TF	CR	CP	CF	SR
BIRD-dev	CodeS-15B	99.76	36.40	53.34	98.91	13.26	23.38	**95.89**
	$\text{CHESS}_{\text{IR+SS+CG}}$	96.00	90.00	92.90	94.00	71.00	80.90	–
	$\text{GenLink}_{\text{L}}$	92.31	93.48	92.89	86.47	88.29	87.37	67.67
	$\text{GenLink}_{\text{L,Q,D}}$	96.68	88.01	92.14	93.48	81.99	87.36	81.23
	$\text{GenLink}_{\text{L,QC,Q,D,M}}$	97.22	85.50	90.98	94.56	78.52	85.80	83.96
	LLaMA-based Linker (Baseline)	93.36	96.48	94.89	88.39	91.69	90.01	85.12
	+BPR-TAP	94.69	96.03	**95.36**	91.84	93.10	**92.47**	88.16
	+BPR-TAP (Sel. exp)	96.15	93.52	94.82	95.00	84.56	89.48	92.11
Spider-test	CodeS-15B	99.97	37.40	54.44	99.98	16.69	28.60	96.60
	$\text{GenLink}_{\text{L,QC,Q,D,M}}$	99.54	90.53	94.82	98.44	89.29	93.64	92.41
	LLaMA-based Linker (Baseline)	97.45	97.81	97.63	95.36	96.72	96.04	94.95
	+BPR-TAP	98.59	97.71	**98.15**	97.01	96.88	**96.94**	96.59
	+BPR-TAP (Sel. exp)	99.12	94.25	96.62	98.77	93.35	95.98	**97.49**
Spider-dev	CodeS-15B	100.00	38.05	55.12	99.76	13.92	24.43	95.84
	$\text{GenLink}_{\text{L,QC,Q,D,M}}$	99.61	92.56	95.96	98.33	88.96	93.41	93.42
	LLaMA-based Linker (Baseline)	97.53	98.98	98.25	94.11	97.84	95.94	93.53
	+BPR-TAP	98.82	98.47	**98.64**	96.22	96.80	**96.51**	95.89
	+BPR-TAP (Sel. exp)	99.73	96.13	97.90	98.28	93.50	95.83	**97.11**

Table 1: Main schema linking results. CodeS-15B is based on CodeS (Li et al., 2024), with reproduced schema-linking results taken from GenLink (Hao et al., 2025). CHESS and GenLink results follow their reported settings (Talaei et al., 2024; Hao et al., 2025). +BPR-TAP uses TAP as guidance for schema linking, while BPR-TAP (Sel. exp) selectively expands SFT targets. Bold values represent the optimal results of TF, CF, and SR metrics.

Hyperparameter Settings. The detailed settings of hyperparameters are presented in Appendix A.

4.2 Main Results

We evaluate the effectiveness of TAP from schema linking and SQL generation perspectives. For schema linking, Table 1 compares our approach against existing linkers. For SQL generation, Table 2 measures the impact of schema linking on downstream SQL execution accuracy, where our best variant surpasses GenLink by 2.35% EX on BIRD. Detailed results are presented below.

4.2.1 Schema Linking Results

Table 1 presents schema linking results on BIRD-dev, Spider-test, and Spider-dev. our +BPR-TAP consistently achieves the strongest F1 scores.

+BPR-TAP achieves stronger F1 scores. Compared with prior schema linking methods, +BPR-TAP achieves the best F1 scores on all three datasets. On BIRD-dev, +BPR-TAP obtains 95.36 TF and 92.47 CF, outperforming CHESS and all GenLink variants in both table- and column-level F1. On Spider-test, +BPR-TAP reaches 98.15 TF and 96.94 CF, exceeding $\text{GenLink}_{\text{L,QC,Q,D,M}}$ by 3.33 TF and 3.30 CF. On Spider-dev, +BPR-TAP achieves 98.64 TF and 96.51 CF, also outperforming $\text{GenLink}_{\text{L,QC,Q,D,M}}$ by 2.68 TF and 3.10 CF.

Compared with the LLaMA-based Linker baseline, +BPR-TAP also brings consistent improvements. On BIRD-dev, it raises TF from 94.89 to 95.36 (+0.47), CF from 90.01 to 92.47 (+2.46), and SR from 85.12 to 88.16 (+3.04). Similar gains are observed on Spider-test and Spider-dev. On Spider-test, CF reaches 96.94 (+0.90) and SR reaches 96.59 (+1.64). On Spider-dev, CF reaches 96.51 (+0.57) and SR reaches 95.89 (+2.36). These gains are mainly attributed to BPR-TAP guidance, which helps the linker recover Bridge Path schema items that are not explicitly mentioned in the query.

+BPR-TAP (Sel. exp) achieves a better recall-oriented trade-off. +BPR-TAP (Sel. exp) further prioritizes recall by selectively expanding SFT targets for hard instances. Compared with +BPR-TAP, it increases CR from 91.84 to 95.00 and SR from 88.16 to 92.11 on BIRD-dev, with a moderate decrease in precision. This trade-off is desirable for downstream SQL generation, where missing required schema items is usually more harmful than retaining a small number of redundant ones. Once a required column or table is filtered out by the schema linker, the SQL generator can hardly recover it in the later generation stage.

Compared with $\text{GenLink}_{\text{L,QC,Q,D,M}}$, which also uses five models and achieves comparable recall

Method	EX
Reported prior work: In-Context Learning Methods	
$\text{CHESS}_{\text{IR+SS+CG+GPT-4-turbo}}$ (Talaei et al., 2024)	65.0
$\text{UCS-SQL}_{\text{GPT-4o}}$ (Wu et al., 2025)	68.95
$\text{CHASE-SQL}_{\text{Gemini-1.5}}$ (Pourreza et al., 2025)	73.01
Reported prior work: SFT / Training Methods	
$\text{DTS-SQL}_{\text{DeepSeek7B}}$ (Pourreza and Rafiei, 2024)	55.8
CodeS-15B (Li et al., 2024)	58.47
$\text{GenLink}_{\text{QC}}$ (w/o voting) (Hao et al., 2025)	59.32
$\text{GenLink}_{\text{L,QC,Q,D,M}}$ (w/ voting) (Hao et al., 2025)	67.34
Arctic-Text2SQL-R1-14B (Yao et al., 2026)	70.10
Ours:	
Baseline (LLaMA-3.1 Linker + Qwen-3 Coder)	56.84
+BPR	58.28
+BPR-TAP	60.04
+BPR-TAP (Sel. exp)	61.21
+BPR-TAP (Sel. exp) + TAP-Distill	61.93
+BPR-TAP (Sel. exp) + TAP-Distill (w/ voting)	**69.69**

Table 2: Impact of schema-linking variants on downstream SQL generation on BIRD-dev. Baseline uses the LLaMA-3.1 schema linker and Qwen-3 Coder. +BPR ranks gold schema targets by answer path roles. +BPR-TAP adds the distilled TAP for schema linking. +BPR-TAP (Sel. exp) selectively expands SFT targets. +TAP-Distill further uses the distilled TAP model to provide TAP context for SQL generation. "w/ voting" denotes consistency-based selection over SQL candidates.

on BIRD-dev (TR=97.22, CR=94.56), +BPR-TAP (Sel. exp) obtains substantially higher precision (TP=93.52 vs. 85.50, CP=84.56 vs. 78.52) while reaching similar recall (TR=96.15, CR=95.00). This indicates that our difficulty-aware selective expansion improves recall while avoiding the severe precision loss of aggressive multi-source fusion.

4.2.2 SQL Generation Results

Table 2 reports BIRD-dev execution accuracy against prior work and schema linking ablations. Our best setting reaches 69.69% EX with consistency-based voting. Additional SQL generation analyses are provided in Appendix B.8.

Each component progressively improves SQL generation. Baseline achieves 56.84% EX. Adding BPR improves EX to 58.28%, showing that answer path role ordering benefits downstream SQL generation. Incorporating TAP guidance further raises EX to 60.04%, and selective expansion increases it to 61.21%. Using the distilled TAP as additional SQL-generation context further improves EX to 61.93%. Overall, the single coder setting improves over the baseline by 5.09 points.

Model	Zone	Error	Rec.	Drop	Net
SLM	Cond.	743	433	146	287
	Bridge	2307	1558	88	**1470**
	Ans.	410	274	81	193
LLM	Cond.	186	96	48	48
	Bridge	804	628	66	**562**
	Ans.	183	106	33	73
SFT	Cond.	300	195	41	154
	Bridge	590	318	80	**238**
	Ans.	179	107	18	89

Table 3: Path-wise analysis of TAP-fixed schema items across three model settings: SLM (LLaMA-based Linker w/o SFT), LLM (DeepSeek-V3.2), and SFT (LLaMA-based Linker + BPR-TAP). Zone denotes answer-path roles: condition endpoints (Cond.), Bridge Path schema items (Bridge), and answer endpoints (Ans.). Error is the number of missing gold schema items before TAP guidance. Rec., Drop, and Net denote recovered items, newly dropped items, and net gain.

Our approach outperforms GenLink under comparable settings. $\text{GenLink}_{\text{L,QC,Q,D,M}}$ employs five models for voting yet achieves only 67.34% EX. In contrast, our single-model setting (+BPR-TAP (Sel. exp) + TAP-Distill) reaches 61.93%, already higher than GenLink without voting (59.32%). With five-model ensemble, we further improve to 69.69%, outperforming GenLink by 2.35 EX. This demonstrates that TAP-guided schema linking provides stronger context for SQL generation than prior methods.

4.3 Ablation Study

4.3.1 Path-wise Analysis of TAP

Table 3 reports item-level TAP effects under SLM, LLM, and SFT settings. Bridge Path schema items obtain the dominant net gains across all models: +1,470 for SLM, +562 for LLM, and +238 for SFT. They also have the highest Error counts (2,307, 804, and 590), indicating that they are the main missing schema items targeted by Bridge Path Recovering.

Condition Endpoints and Answer Endpoints also show positive but smaller gains: +287/+193 for SLM, +48/+73 for LLM, and +154/+89 for SFT, respectively. These results show that TAP guidance improves schema linking across the answer derivation path, with the largest gains on Bridge Path schema items. This confirms that Bridge Path Recovering is the primary benefit of TAP. Additional gold schema ranking improvement and instruction-only analyses in Appendices B.1 and B.2 further confirm that TAP-guided linking mainly benefits from recalling Bridge Path schemata.

Setting	Method	TR	TP	TF	CR	CP	CF	SR
DeepSeek-V3.2	Base	95.90	92.31	94.07	88.67	90.12	89.39	83.54
	+TAP-Distill	**96.60**	89.72	93.03	95.57	78.65	86.29	93.46
	+Gold Answer	**96.60**	91.07	93.75	**96.06**	79.38	86.93	**94.27**
Llama-3.1 w/o SFT	Base	90.00	90.13	90.06	63.27	89.10	74.00	52.72
	+TAP-Raw	95.29	78.83	86.28	84.10	79.06	81.50	77.11
	+TAP-Distill	96.05	80.17	87.39	86.16	79.33	82.60	80.31
	+Gold Answer	**96.65**	80.18	87.65	**86.74**	80.06	83.27	**80.89**
Llama-3.1 w/ SFT	BPR-TAP (Sel. exp)	**96.15**	93.52	94.82	95.00	84.56	89.48	92.11
	+Gold Answer	96.14	93.91	95.01	**95.25**	85.46	90.09	**92.25**

Table 4: Diagnostic ablation of replacing the tentative answer with the executed gold answer on BIRD-dev. Only the tentative answer field in the TAP is replaced, while the answer type and CoT steps remain unchanged.

4.3.2 Effect of Replacing the Tentative Answer with the Gold Answer

We conduct a diagnostic ablation to examine whether TAP mainly benefits from access to the exact answer value. For each instance, we execute the gold SQL and use the resulting values as the gold answer. We then replace only the tentative answer field in the TAP with the gold answer, while keeping the answer type and CoT steps unchanged.

Table 4 shows that using the gold answer brings only limited additional gains over TAP-Distill. For Llama-3.1 without SFT, TAP-Distill improves CR/SR from 63.27/52.72 to 86.16/80.31, while replacing the tentative answer with the gold answer further increases them only to 86.74/80.89. For DeepSeek-V3.2, CR/SR changes from 95.57/93.46 to 96.06/94.27. Under the SFT setting, the gain is even smaller, with CR/SR changing from 95.00/92.11 to 95.25/92.25.

These results indicate that TAP does not rely on exact answer values. Instead, it serves as answer path guidance for schema linking. The tentative answer, together with the answer type and CoT steps, provides a coarse answer-derivation direction, helping the schema linker recover weakly aligned Bridge Path schema items along the answer derivation path. We further support this conclusion with answer-overlap statistics in Appendix B.6, which show that predicted tentative answers rarely match the executed SQL answers.

4.3.3 Component ablation of schema linking

We conduct a component-level ablation on BIRD-dev to quantify the contribution of each variant in our schema linking framework. Table 5 reports cumulative variants from Baseline to +BPR, +BPR-TAP, full expansion (Full exp.), and difficulty-aware selective expansion (Sel. exp.).

Config.	TR	TP	CR	CP
Baseline	93.36	96.48	88.39	91.69
+BPR	94.50	95.48	90.59	92.13
+BPR-TAP	94.69	96.03	91.84	93.10
+Full exp.	96.96	89.73	96.25	70.41
+Sel. exp.	96.15	93.52	95.00	84.56

Table 5: Component ablation of schema linking variants on BIRD-dev. Rows are cumulative from Baseline to +BPR, +BPR-TAP, full expansion (Full exp.), and difficulty-aware selective expansion (Sel. exp.). Full exp. applies TAP-guided expansion to all training instances, while Sel. exp. applies it only to hard instances.

+BPR improves recall over Baseline, raising CR from 88.39 to 90.59. Adding TAP guidance further raises CR to 91.84 while preserving high precision. This suggests that answer path role supervision and TAP guidance are complementary. The former structures the target along the answer derivation path, whereas the latter provides answer oriented context for schema selection.

Full exp. achieves the highest recall, with 96.25 CR, but its CP drops to 70.41, showing that applying TAP-guided expansion to all training instances introduces redundant schema items. Sel. exp. preserves most of the recall gain, reaching 95.00 CR, while recovering CP to 84.56. This confirms that difficulty-aware selective expansion controls expansion noise and yields a better recall–precision balance. We observe the same trend on Spider-dev and Spider-test, further validating the robustness of selective expansion. As shown in Appendix B.3.

4.3.4 Effect of Candidate Set Size

We analyze how the expansion size K_{ret} affects the recall–precision trade-off on BIRD-dev. The column-level trend is shown in Figure 4, and the full TR/TP/CR/CP results for all candidate sizes are reported in Appendix B.4.

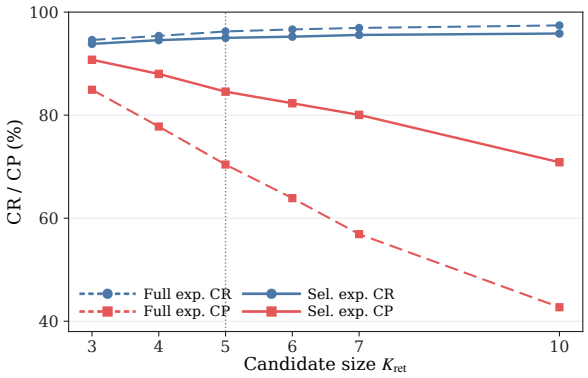


Figure 4: Candidate-size trade-off on BIRD-dev. The vertical line marks the candidate size selected for our final method, $K_{\text{ret}} = 5$. Full exp. expands all instances. Sel. exp. expands only hard instances.

As K_{ret} increases, Full exp. steadily improves CR, but it also causes a sharp CP drop. This indicates that adding all TAP-guided expansion candidates increases the chance of covering missing Bridge Path schema items, while also introducing more noisy and redundant schema items. In contrast, Sel. exp. consistently maintains much higher precision by applying expansion only to hard instances. At $K_{\text{ret}} = 5$, Sel. exp. improves CP from 70.41 to 84.56 compared with Full exp., with only a small CR decrease from 96.25 to 95.00.

The trend also shows why an intermediate candidate size is preferable. When K_{ret} is too small, the retrieved set may not contain enough useful Bridge Path schema items. When it becomes too large, the recall gain starts to saturate while expansion noise increases rapidly. We therefore use $K_{\text{ret}} = 5$ in the final method, as it provides a better balance between Bridge Path recovery and redundancy control. Spider results show the same trend and are provided in Appendix B.5.

5 Related Work

LLMs for Text2SQL. Recent Text2SQL systems improve LLM-based SQL generation through prompting, decomposition, and candidate selection. DIN-SQL (Pourreza and Rafiei, 2023) decomposes SQL generation into intermediate prompting stages, while DAIL-SQL (Gao et al., 2024) studies demonstration selection and prompt organization for in-context learning. CHESS (Talaei et al., 2024) and CHASE-SQL (Pourreza et al., 2025) further improve robustness with multi-path reasoning, and candidate selection. These methods mainly optimize SQL generation with a given schema context.

Another line of work trains specialized Text2SQL models or decomposes the pipeline into schema linking and SQL generation. DTS-SQL (Pourreza and Rafiei, 2024) studies decomposed fine-tuning with small language models, and CodeS (Li et al., 2024) builds SQL-oriented open-source language models. These studies show the value of stronger SQL generators and decomposed pipelines. Our work is complementary: we target schema linking before SQL decoding, because missing required schema items cannot be reliably repaired by downstream SQL generation.

Schema Linking for Text2SQL. Recent work revisits schema linking as a key bottleneck for LLM-based Text2SQL. Some studies analyze when explicit schema linking is still needed (Maamari et al., 2024), while others improve recall through question decomposition, extractive selection, retrieval, or large-scale schema grounding (Pradeep et al., 2025; Glass et al., 2025; Nahid et al., 2026; Wang et al., 2025). These methods reduce irrelevant schema context and improve schema recall.

Our work focuses on a different failure mode: required schema items may be only weakly implied by the question. GenLink (Hao et al., 2025) addresses implicit schema linking by generating multiple SQL candidates and extracting schema evidence from them. However, this generation-driven strategy relies on multiple SQL generators. It also obtains schema evidence only after SQL generation. This partially reintroduces the burden that decomposed Text2SQL pipelines aim to avoid. In contrast, BPR-TAP recovers weakly indicated Bridge Path schema items before SQL generation. It uses Tentative Answer Profiling as answer path guidance and difficulty-aware selective expansion to improve recall while controlling redundant schema items.

6 Conclusion

We propose BPR-TAP to enhance linkers by Bridge Path Recovering via Tentative Answer Profiling. BPR-TAP adds a tentative answer profile, including a tentative answer, answer type, and CoT steps, to guide the linker in reversely deducing multi-hop routing paths over SQL tables and ranking schemata during finetuning. Experiments on BIRD and Spider show that BPR-TAP consistently improves schema-linking F1 and downstream execution accuracy. Further analysis shows that the gains mainly come from recalling Bridge Path schemata, confirming the utility of TAP.

Limitations

BPR-TAP still has several limitations. First, TAP-guided expansion candidates are used only for constructing SFT targets on hard instances. This keeps inference lightweight, but leaves more adaptive uses of these candidates for future exploration. Second, TAP is generated by an LLM and may contain noisy or incomplete answer profiles for complex databases. Although our diagnostic analyses show that BPR-TAP does not rely on exact answer matching, improving the reliability of TAP generation remains future work.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (62376182).

References

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, and 245 others. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *Preprint*, arXiv:2512.02556.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2024. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. *Proceedings of the VLDB Endowment*, 17(5):1132–1145.

Michael Glass, Mustafa Eyceoz, Dharmashankar Subramanian, Gaetano Rossiello, Long Vu, and Alfio Gliozzo. 2025. Extractive Schema Linking for Text-to-SQL. *arXiv preprint*. ArXiv:2501.17174 [cs].

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The Llama 3 Herd of Models. *arXiv preprint*. ArXiv:2407.21783 [cs].

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. 2024. DeepSeek-Coder: When the Large Language Model Meets Programming – The Rise of Code Intelligence. *arXiv preprint*. ArXiv:2401.14196 [cs].

Zhifeng Hao, Junqi Huang, Shaobin Shi, Ruichu Cai, and Boyan Xu. 2025. GenLink: Generation-driven schema-linking via multi-model learning for text-to-SQL. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 29892–29905, Suzhou, China. Association for Computational Linguistics.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, and 5 others. 2024. Qwen2.5-Coder Technical Report. *arXiv preprint*. ArXiv:2409.12186 [cs].

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. *arXiv preprint*. ArXiv:2309.06180 [cs].

Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024. CodeS: Towards Building Open-source Language Models for Text-to-SQL. *arXiv preprint*. ArXiv:2402.16347 [cs].

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. In *Advances in Neural Information Processing Systems*, volume 36, pages 42330–42357. Curran Associates, Inc.

Yifu Liu, Yin Zhu, Yingqi Gao, Zhiling Luo, Xiaoxia Li, Xiaorong Shi, Yuntao Hong, Jinyang Gao, Yu Li, Bolin Ding, and Jingren Zhou. 2026. Xiyan-sql: A novel multi-generator framework for text-to-sql. *IEEE Trans. Knowl. Data Eng.*, 38(4):2474–2487.

Yuyu Luo, Guoliang Li, Ju Fan, Chengliang Chai, and Nan Tang. 2025. Natural language to sql: State of the art and open problems. *Proc. VLDB Endow.*, 18(12):5466–5471.

Karime Maamari, Fadhil Abubaker, Daniel Jaroslawicz, and Amine Mhedhbi. 2024. The death of schema linking? text-to-sql in the age of well-reasoned language models. *Preprint*, arXiv:2408.07702.

Md Mahadi Hasan Nahid, Davood Rafiei, Weiwei Zhang, and Yong Zhang. 2026. Rethinking Schema Linking: A Context-Aware Bidirectional Retrieval Approach for Text-to-SQL. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 4516–4546, Rabat, Morocco. Association for Computational Linguistics.

Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan Arik. 2025. Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql. In *International Conference on Learning Representations*, volume 2025, pages 60385–60415.

Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. In *Advances in Neural Information Processing Systems*, volume 36.

Mohammadreza Pourreza and Davood Rafiei. 2024. DTS-SQL: Decomposed Text-to-SQL with Small Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8212–8220, Miami, Florida, USA. Association for Computational Linguistics.

Kiran Pradeep, Kirushikesh Db, Nishtha Madaan, Sameep Mehta, and Pushpak Bhattacharyya. 2025. Divide, link, and conquer: Recall-oriented schema linking for NL-to-SQL via question decomposition. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1727–1743, Suzhou (China). Association for Computational Linguistics.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at trec-3. In *Text Retrieval Conference*.

Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. 2024. Chess: Contextual harnessing for efficient sql synthesis. *Preprint*, arXiv:2405.16755.

Yihan Wang, Peiyu Liu, and Xin Yang. 2025. LinkAlign: Scalable Schema Linking for Real-World Large-Scale Multi-Database Text-to-SQL. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 977–991, Suzhou, China. Association for Computational Linguistics.

Zhenhe Wu, Zhongqiu Li, Jie Zhang, Zhongjiang He, Jian Yang, Yu Zhao, Ruiyu Fang, Bing Wang, Hongyan Xie, Shuangyong Song, and Zhoujun Li. 2025. UCS-SQL: Uniting content and structure for enhanced semantic bridging in text-to-SQL. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8156–8168, Vienna, Austria. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Zhewei Yao, Guoheng Sun, Łukasz Borchmann, Zheyu Shen, Minghang Deng, Bohan Zhai, Hao Zhang, Ang Li, and Yuxiong He. 2026. Arctic-Text2SQL-r1: Simple rewards, strong reasoning in text-to-SQL. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 26966–26995, San Diego, California, United States. Association for Computational Linguistics.

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. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Weixu Zhang, Yifei Wang, Yuanfeng Song, Victor Junqiu Wei, Yuxing Tian, Yiyan Qi, Jonathan H. Chan, Raymond Chi-Wing Wong, and Haiqin Yang. 2024. Natural language interfaces for tabular data querying and visualization: A survey. *IEEE Trans. on Knowl. and Data Eng.*, 36(11):6699–6718.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *Preprint*, arXiv:2506.05176.

A Hyperparameter settings

Models. All small models in schema linking are initialized from Llama-3.1-8B-Instruct, including the TAP model and the student linker for difficulty estimation. TAP training data is generated on the training set by DeepSeek-V3.2, which is used only for training-data construction.

For SQL generation, Qwen3-8B (Yang et al., 2025) is used as the fixed generator in single-model and SQL-level ablations. The full pipeline performs consistency-based aggregation with five generators: Arctic-Text2SQL-R1-7B (Yao et al., 2026), DeepSeek-Coder-6.7B-Instruct (Guo et al., 2024), Llama-3.1-8B-Instruct, Qwen3-8B, and Qwen2.5-Coder-7B-Instruct (Hui et al., 2024).

Training and Inference. All small models are fine-tuned with LLaMA-Factory using LoRA-based supervised fine-tuning. The main hyperparameters are summarized in Table 6. All models are deployed with vLLM (Kwon et al., 2023) for inference, and unless otherwise specified, inference is performed with temperature 0.

Hyperparameter	Value
Fine-tuning method	LoRA SFT
LoRA rank	32
LoRA alpha	64
LoRA dropout	0.1
LoRA target modules	all
Warmup ratio	0.1

Table 6: Common training hyperparameters used in our LoRA-based SFT experiments.

Model / Stage	LR	Epochs
TAP model	5.0×10^{-5}	3
Schema linker models	5.0×10^{-5}	3
Arctic-Text2SQL-R1-7B	1.0×10^{-5}	1
Other SQL generators	5.0×10^{-5}	2

Table 7: Training hyperparameter settings.

Difficulty-Aware Selective SFT. For difficulty estimation in Selective-SFT, we use a preliminary schema linker initialized from Llama-3.1-8B-Instruct with the same LoRA configuration.

Hardware. All experiments are conducted on 4 NVIDIA GeForce RTX 3090 GPUs, each with 24GB of memory.

Path Zone	Gold Rank Gain
Input-side Condition Endpoint	+0.955
Bridge Path	+5.841
Output-side Answer Endpoint	+3.677

Table 8: Average rank gain by path zone after incorporating the TAP.

B Additional Experimental Results and Analyses

B.1 Gold-schema Ranking Improvement

To complement the ranking evidence in Section 4.3.1, we further measure the average rank improvement of all gold schema items after incorporating the TAP. While Table 3 focuses on schema items whose inclusion status changes after TAP guidance, this appendix evaluates all gold schema items within each answer-path zone. Gold Rank Gain is computed as the rank before TAP guidance minus the rank after TAP guidance, so a positive value means that a gold schema item is promoted toward the top of the ranked schema list.

As shown in Table 8, Bridge Path obtains the largest average rank gain among all path zones. This result is consistent with the path-wise analysis in the main text: TAP guidance not only recovers missing Bridge Path schema items, but also generally assigns higher ranks to gold Bridge Path items. Input-side Condition Endpoints and Output-side Answer Endpoints also receive positive rank gains, showing that the tentative answer and answer type cues help endpoint selection. Nevertheless, the much larger gain on Bridge Path indicates that the CoT steps in the TAP mainly compensate for intermediate connection paths that are weakly aligned with the question surface but essential for constructing the answer derivation path.

B.2 Effect of TAP on Large Instruction Models

We evaluate the effect of TAP on schema linking under the instruction-only setting. Base denotes direct schema linking without TAP guidance and serves as the baseline. +TAP-Distill uses the distilled TAP model described in Section 3.1: TAP labels generated by DeepSeek-V3.2 are distilled into Llama-3.1-8B-Instruct, and the distilled model generates TAP for schema linking. +TAP-Raw uses Llama-3.1-8B-Instruct itself to generate TAP without teacher-student distillation.

Dataset	Scale	Method	TR	TP	TF	CR	CP	CF	SR
BIRD-dev	Large	DeepSeek-V3.2 (Base)	95.90	92.31	94.07	88.67	90.12	89.39	83.54
	Large	DeepSeek-V3.2 (+TAP-Distill)	**96.60**	89.72	93.03	**95.57**	78.65	86.29	**93.46**
	Small	Llama-3.1-8B-Instruct (Base)	90.00	90.13	90.06	63.27	89.10	74.00	52.72
	Small	Llama-3.1-8B-Instruct (+TAP-Raw)	95.29	78.83	86.28	84.10	79.06	81.50	77.11
	Small	Llama-3.1-8B-Instruct (+TAP-Distill)	**96.05**	80.17	87.39	**86.16**	79.33	82.60	**80.31**
Spider-test	Large	DeepSeek-V3.2 (Base)	98.87	94.94	96.87	78.06	95.59	85.94	77.18
	Large	DeepSeek-V3.2 (+TAP-Distill)	**99.68**	88.97	94.02	**84.42**	81.55	82.96	**84.67**
	Small	Llama-3.1-8B-Instruct (Base)	96.62	90.20	93.30	66.67	92.94	77.64	62.27
	Small	Llama-3.1-8B-Instruct (+TAP-Raw)	99.07	72.79	83.92	79.30	78.21	78.75	77.25
	Small	Llama-3.1-8B-Instruct (+TAP-Distill)	**99.39**	73.57	84.55	**81.04**	79.35	80.19	**79.06**
Spider-dev	Large	DeepSeek-V3.2 (Base)	99.69	93.83	96.67	74.21	94.76	83.24	73.53
	Large	DeepSeek-V3.2 (+TAP-Distill)	**99.73**	89.47	94.32	**80.06**	80.71	80.38	**80.28**
	Small	Llama-3.1-8B-Instruct (Base)	96.20	90.22	93.11	62.61	91.61	74.38	58.74
	Small	Llama-3.1-8B-Instruct (+TAP-Raw)	98.18	76.71	86.13	71.85	76.92	74.30	68.96
	Small	Llama-3.1-8B-Instruct (+TAP-Distill)	**98.60**	74.74	85.03	**71.95**	75.24	73.56	**68.97**

Table 9: Effect of shared TAP on instruction-based schema linking.

TAP substantially improves recall. As shown in Table 9, TAP consistently improves recall across both large and small instruction models, with larger gains on the small model. On BIRD-dev, DeepSeek-V3.2 with +TAP-Distill raises CR/SR from 88.67/83.54 to 95.57/93.46. For Llama-3.1-8B-Instruct, +TAP-Distill improves CR/SR from 63.27/52.72 to 86.16/80.31. These results indicate that TAP provides answer path guidance for recovering weakly aligned Bridge Path schema items, especially when the base instruction model has limited linking ability.

Distilled TAP outperforms raw self-generated TAP. On BIRD-dev, TAP-Distill achieves CR/SR of 86.16/80.31 for Llama-3.1-8B-Instruct, compared to 84.10/77.11 with TAP-Raw. The distilled signal provides higher-quality guidance than the small model's own raw TAP generation, showing the benefit of teacher-student TAP distillation.

TAP introduces a recall–precision trade-off. TP and CP drop on both datasets after adding TAP. This is because the instruction-only setting does not apply the difficulty-aware selection used in our SFT framework. TAP tends to recall more potentially relevant tables and columns, and without difficulty-aware filtering, these extra schema items may reduce precision. For instance, DeepSeek-V3.2 on BIRD-dev shows TP decreasing from 92.31 to 89.72. Similar trade-offs have been observed in prior schema linking methods (Maamari et al., 2024; Pradeep et al., 2025).

Config.	TR	TP	CR	CP
Baseline	97.45	97.81	95.36	96.72
+BPR	97.62	97.37	96.20	96.38
+BPR-TAP	98.59	97.71	97.01	96.88
+Full exp.	99.50	80.86	99.23	68.24
+Sel. exp.	99.12	94.25	98.77	93.35

Table 10: Component ablation of schema linking variants on Spider-test. Rows are cumulative. Full exp. applies TAP-guided expansion to all training instances, while Sel. exp. applies it only to hard instances.

Config.	TR	TP	CR	CP
Baseline	97.53	98.98	94.11	97.84
+BPR	98.15	97.79	94.67	95.88
+BPR-TAP	98.82	98.47	96.22	96.80
+Full exp.	99.97	84.41	98.49	68.91
+Sel. exp.	99.73	96.13	98.28	93.50

Table 11: Component ablation of schema linking variants on Spider-dev. Rows are cumulative. Full exp. applies TAP-guided expansion to all training instances, while Sel. exp. applies it only to hard instances.

B.3 Additional Ablation Results on Spider

We provide additional component-level ablation results on Spider-test (Table 10) and Spider-dev (Table 11) to complement the BIRD-dev results in Table 5. The variants follow the same cumulative setting: +BPR adds answer path role supervision, +BPR-TAP further incorporates TAP guidance, Full exp. applies TAP-guided expansion to all training instances, and Sel. exp. applies the expansion only to hard instances.

The Spider-test results show the same trend as BIRD-dev. +BPR improves CR from 95.36 to

96.20, and +BPR-TAP further raises CR to 97.01 while preserving high precision. This indicates that answer-path role supervision and TAP guidance provide complementary benefits for schema linking. Full exp. achieves the highest recall, with 99.23 CR, but its CP drops to 68.24, showing that applying expansion to all instances introduces redundant schema items. In contrast, Sel. exp. preserves most of the recall gain, reaching 98.77 CR, while recovering CP to 93.35.

The same pattern is observed on Spider-dev. +BPR and +BPR-TAP progressively improve CR from 94.11 to 94.67 and 96.22. Full exp. again improves recall substantially, reaching 98.49 CR, but causes a large CP drop to 68.91. Sel. exp. keeps nearly the same recall, with 98.28 CR, while improving CP to 93.50. These results confirm that BPR, TAP guidance, and difficulty-aware selective expansion provide stable benefits across Spider.

B.4 Full Candidate-Size Results on BIRD-dev

K_{ret}	Method	TR	TP	CR	CP
3	Full exp.	96.37	93.42	94.60	84.93
3	Sel. exp.	95.75	95.40	93.83	90.75
4	Full exp.	96.73	91.09	95.38	77.79
4	Sel. exp.	95.99	94.56	94.56	88.00
5	Full exp.	96.96	89.73	96.25	70.41
5	Sel. exp.	96.15	93.52	95.00	84.56
6	Full exp.	97.15	87.37	96.64	63.89
6	Sel. exp.	96.42	93.17	95.23	82.31
7	Full exp.	97.38	84.70	96.93	56.91
7	Sel. exp.	96.60	92.24	95.57	80.05
10	Full exp.	97.73	78.81	97.42	42.72
10	Sel. exp.	96.92	90.12	95.84	70.86

Table 12: Full candidate-size results on BIRD-dev. Full exp. applies TAP-guided expansion to all training instances, while Sel. exp. applies it only to hard instances.

The BIRD-dev results show a clear recall–precision trade-off. As K_{ret} increases, Full exp. obtains higher recall but suffers a substantial precision drop, especially at the column level. For example, CR increases from 94.60 to 97.42 when K_{ret} changes from 3 to 10, while CP drops from 84.93 to 42.72. In contrast, Sel. exp. preserves most of the recall gain while maintaining much higher precision. At $K_{\text{ret}} = 5$, Sel. exp. improves CP from 70.41 to 84.56 compared with Full exp., with only a small CR decrease from 96.25 to 95.00. These results support difficulty-aware selective expansion and motivate $K_{\text{ret}} = 5$ as the default candidate size in our final setting.

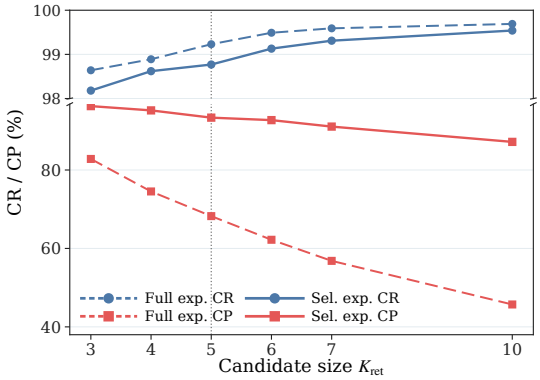


Figure 5: Candidate-size trade-off on Spider-test.

K_{ret}	Method	TR	TP	CR	CP
3	Full exp.	99.40	92.35	98.64	82.82
3	Sel. exp.	98.97	96.28	98.18	96.29
4	Full exp.	99.54	87.67	98.89	74.53
4	Sel. exp.	99.02	96.00	98.62	95.20
5	Full exp.	99.50	80.86	99.23	68.24
5	Sel. exp.	99.12	94.25	98.77	93.35
6	Full exp.	99.62	75.20	99.49	62.21
6	Sel. exp.	99.32	94.01	99.13	92.72
7	Full exp.	99.64	70.70	99.59	56.83
7	Sel. exp.	99.37	93.09	99.31	91.07
10	Full exp.	99.86	56.89	99.69	45.71
10	Sel. exp.	99.38	88.60	99.54	87.17

Table 13: Full candidate-size results on Spider-test.

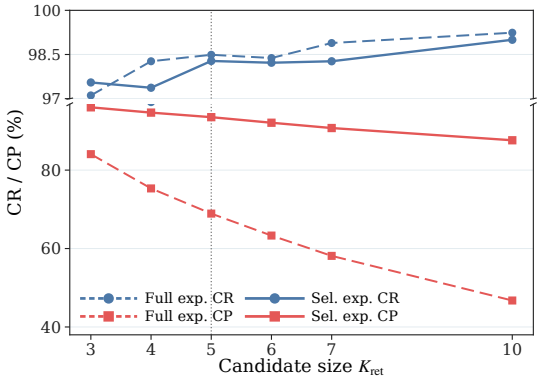


Figure 6: Candidate-size trade-off on Spider-dev.

K_{ret}	Method	TR	TP	CR	CP
3	Full exp.	99.62	93.09	97.11	84.07
3	Sel. exp.	99.48	97.55	97.55	96.01
4	Full exp.	99.77	90.48	98.27	75.33
4	Sel. exp.	99.74	97.09	97.37	94.66
5	Full exp.	99.97	84.41	98.49	68.91
5	Sel. exp.	99.73	96.13	98.28	93.50
6	Full exp.	99.97	79.14	98.38	63.31
6	Sel. exp.	99.65	95.41	98.22	92.08
7	Full exp.	99.90	75.70	98.89	58.13
7	Sel. exp.	99.42	95.13	98.27	90.73
10	Full exp.	99.97	63.15	99.24	46.75
10	Sel. exp.	99.76	92.09	99.00	87.61

Table 14: Full candidate-size results on Spider-dev.

B.5 Effect of Candidate Set Size on Spider

We further report the candidate-size analysis on Spider-test and Spider-dev. Following the BIRD-dev analysis in Figure 4, we visualize the CR–CP trade-off (Figure 5 & 6) and retain the full TR/TP/CR/CP tables (Table 13 & 14).

The Spider results are consistent with the BIRD-dev trend. As K_{ret} increases, Full exp. obtains higher recall but suffers a substantial precision drop, especially at the column level. In contrast, Sel. exp. preserves most of the recall gain while maintaining much higher precision across candidate sizes. At $K_{\text{ret}} = 5$, Sel. exp. improves CP from 68.24 to 93.35 on Spider-test and from 68.91 to 93.50 on Spider-dev compared with Full exp., with only minor CR decreases. These results further support difficulty-aware selective expansion across datasets.

B.6 Answer-Overlap Analysis

Metric	Value
Exact Match	4.43
Character Similarity	26.30

Table 15: Answer overlap statistics between tentative answers and executed SQL answers on BIRD-dev.

We further measure the overlap between the predicted tentative answers and the executed SQL answers. This analysis checks whether the TAP model implicitly copies or reconstructs the ground-truth answer. As shown in Table 15, only 4.43% of predicted answers exactly match the executed SQL answers. The average character similarity is also low, at 26.30%.

The low overlap shows that the predicted tentative answer is not an exact reproduction of the ground-truth answer. Therefore, the improvement of BPR-TAP does not come from answer leakage. Instead, the tentative answer provides approximate answer path semantics. It gives the schema linker answer-side direction about what should be returned. Together with the answer type and CoT steps, it helps guide Bridge Path Recovery without requiring access to the true SQL execution result.

B.7 Details of Consistent SQL Selection

We provide details of the consistency-based SQL selector used in our full pipeline. Given SQL candidates produced by multiple generators, we first remove invalid or unsafe SQL codes and execute the remaining candidates on the target SQLite database. Candidates with the same execution result are grouped into one execution cluster. And further use lightweight cluster-level signals, including cross-generator diversity, SQL-structure priors, result-shape priors, and empty-result penalties. The selector focuses on the top-K_{exec} execution clusters: if the top cluster has a clear margin, it is selected directly. Otherwise, the top clusters are reranked by the above signals.

Finally, we choose a representative SQL query from the selected execution cluster as the prediction. This procedure improves robustness over plain majority voting, because it accounts for both agreement among generators and basic SQL/result plausibility, while still preserving execution consistency as the main selection criterion.

Model	Simple	Mod.	Chall.	Total
DS-Coder-6.7B	68.00	49.57	35.86	59.39
Qwen2.5-Coder-7B	69.30	52.80	40.69	61.60
Llama-3.1-8B	69.62	50.00	39.31	60.82
Qwen3-8B	68.97	52.37	47.59	61.93
Arctic-R1-7B	68.22	54.31	48.28	62.13

Table 16: Single-model SQL generation results on BIRD-dev under the same schema linking setting. We report execution accuracy across difficulty levels.

B.8 SQL Generation Results

B.8.1 Single-model SQL Generation Results

We further compare the downstream SQL generation performance of different single SQL generators under the same schema linking setting. Specifically, we keep the linked schema fixed and replace only the SQL generator, so that the results reflect the effect of the generator choice rather than differences in schema linking. Table 16 reports EX on BIRD-dev across different difficulty levels.

As shown in Table 16, different SQL generators achieve comparable performance on Simple queries, while the main differences appear on Moderate and Challenging queries. Arctic-R1-7B achieves the highest Total EX of 62.13, mainly due to its stronger performance on Moderate and Challenging queries. Qwen3-8B obtains a comparable overall result and achieves the highest Challenging accuracy among the general instruction models, reaching 47.59. In contrast, DS-Coder-6.7B performs competitively on Simple queries but lags behind on more difficult queries, suggesting that generator capability becomes more important as SQL complexity increases.

These results indicate that the proposed schema

Method	Simple	Mod.	Chall.	Total
Single generator	68.97	52.37	47.59	61.93
5-model consistency	**75.24**	**63.15**	**55.17**	**69.69**
Gain	+6.27	+10.78	+7.58	+7.76

Table 17: Effect of multi-model SQL consistency on BIRD-dev. All settings use the same schema-linking output. The single-generator setting uses `Qwen3-8B`. The 5-model consistency setting selects SQL from execution clusters produced by multiple SQL generators.

Diff.	Largest	Oracle@2	Oracle@All	Headroom
Simple	72.00	79.46	81.62	+7.46
Mod.	56.90	67.24	71.98	+10.34
Chall.	48.28	61.38	67.59	+13.10
Total	65.19	74.05	77.38	+8.86

Table 18: Source analysis of multi-model SQL consistency. Largest denotes selecting the largest execution cluster. Oracle@2 denotes whether either of the two largest execution clusters contains a correct SQL. Oracle@All denotes whether any generated candidate is correct. Headroom is Oracle@2 minus Largest.

Method	Dev	Test
Baseline (LLaMA-3.1 Linker + Qwen-3 Coder)	83.95	82.63
+BPR-TAP (Sel. exp) + TAP-Distill	87.52	87.01
+BPR-TAP (Sel. exp) + TAP-Distill (w/ voting)	**89.65**	**88.64**

Table 19: Impact of schema-linking variants on downstream SQL generation, measured by execution accuracy (EX), on Spider-dev and Spider-test. Baseline uses the LLaMA-3.1 schema linker and Qwen-3 Coder. +BPR-TAP (Sel. exp) selectively expands SFT targets. +TAP-Distill uses the distilled TAP model to provide TAP context for SQL generation. "w/ voting" denotes consistency-based selection over SQL candidates.

linking setting provides stable support across different SQL generators. They also explain why the Full Pipeline in Table 2 benefits from multi-model SQL generation: while single generators achieve similar performance on simpler queries, they exhibit complementary strengths on Moderate and Challenging queries. Thus, downstream performance can be further improved by adopting stronger Text2SQL generators or by aggregating diverse SQL candidates from multiple generators.

B.8.2 Analysis of Multi-model SQL Consistency

We analyze the contribution of multi-model SQL consistency. The single generator setting uses `Qwen3-8B` to generate SQL from the linked schema and TAP. The five-model consistency setting uses multiple SQL generators to produce candidates, executes candidates on the database, and groups SQL queries with the same execution result into one execution cluster. The final SQL is selected from the execution clusters by consistency-based selector.

Table 17 shows that multi-model consistency improves execution accuracy across all difficulty levels. Compared with the single generator setting, it raises the overall EX from 61.93 to 69.69, giving a 7.76 point gain. The largest gain appears on Moderate questions, where EX increases by 10.78 points. This suggests that consistency selection is most useful when single generator decoding is unstable but at least one generator can produce a correct executable candidate.

Table 18 explains why reranking execution clusters is helpful. Selecting the largest execution cluster reaches 65.19 EX, which is already higher than the single generator setting. However, Oracle@2 reaches 74.05 EX, showing that many correct SQLs are generated by the model set but are not always in the largest cluster. The headroom is larger on harder questions, increasing from 7.46 on Simple to 13.10 on Challenging questions. This indicates that multi-model consistency benefits from candidate diversity. It further improves selection when the largest execution cluster is uncertain.

B.9 Downstream Transfer to Spider

We further evaluate the downstream SQL-generation variants from Table 2 on Spider-dev and Spider-test.

Without voting, BPR-TAP improves EX by 3.57 points on Spider-dev and 4.38 points on Spider-test. With voting, the improvements increase to 5.70 and 6.01 points, respectively. The gains on both splits show that the downstream benefit of BPR-TAP is not specific to BIRD.

B.10 Transfer to Additional LLM Linkers

Table 9 reports the results for DeepSeek-V3.2. We further evaluate TAP-Distill with two additional LLM linkers on BIRD-dev: Qwen3.5-Plus and GPT-4o. Base denotes direct schema linking without TAP, while +TAP-Distill uses the tentative answer profile generated by the distilled TAP model.

TAP-Distill increases CR and SR from 88.56 and 82.09 to 95.66 and 93.35, respectively, for Qwen3.5-Plus. For GPT-4o, it increases CR from

Model	Setting	TR	TP	TF	CR	CP	CF	SR
Qwen3.5-Plus	Base	95.51	95.22	95.36	88.56	94.01	91.20	82.09
	+TAP-Distill	**96.20**	94.05	95.11	**95.66**	87.53	91.41	**93.35**
GPT-4o	Base	95.54	93.88	94.70	89.84	93.69	91.72	89.75
	+TAP-Distill	**95.76**	94.13	94.94	**93.22**	91.22	92.21	**89.95**

Table 20: BIRD-dev schema-linking results with additional LLM linkers. Base denotes direct schema linking without TAP guidance.

Inference stage	Weights	Peak VRAM
TAP-Distill	8.67 GiB	22.62 GiB
Schema linking	8.67 GiB	23.54 GiB
SQL generation	8.97 GiB	22.12 GiB

Table 21: Model-weight footprint and peak VRAM usage during inference on BIRD-dev.

89.84 to 93.22, while CP remains above 91 and SR increases from 89.75 to 89.95. Together with the DeepSeek-V3.2 results in Table 9, these results demonstrate that TAP transfers across different LLM linkers.

B.11 Inference Memory Usage

We report inference memory usage on BIRD-dev. All models are deployed with vLLM (Kwon et al., 2023). TAP-Distill and schema linking use the same Llama-3.1-8B backbone with different LoRA adapters. The three inference stages are executed sequentially.

As shown in Table 21, the peak VRAM usage ranges from 22.12 to 23.54 GiB. The similar peak VRAM usage across stages is mainly due to the KV-cache allocation mechanism of vLLM. Since the three stages are executed sequentially, the single-generator pipeline can run on a single 24 GB GPU. The optional multi-generator consistency setting requires additional parallel GPU resources.

C Case Study

We provide two cases to illustrate how TAP guides BPR. The first case shows how CoT steps help recover Bridge Path schema items, and the second case shows how the tentative answer and type help identify a weakly signaled Answer Endpoint.

Case 1: Recovering Bridge Path schema items. **Question:** *What percentage of male clients request weekly statements?*

This case follows the motivating example in Figure 1. The question contains explicit surface clues for two endpoint schema items, `client.gender` and `account.frequency`. The phrase "male clients" is directly aligned with `client.gender`, and "weekly statements" is directly aligned with `account.frequency`. Thus, a linker can recover these endpoints through surface-level schema matching, but the endpoints alone are insufficient for constructing executable SQL.

The predicted TAP contains the tentative answer *about 55%*, the answer type REAL, and CoT steps indicating that the query should first identify weekly-statement accounts, then connect them to clients through the client–account relation, and finally compute the percentage of male clients among the matched clients. Although the CoT steps do not explicitly mention join columns, they provide the missing relational direction between the two endpoints.

Following this answer path clue, TAP recovers the Bridge Path schema items along the path `client.gender` → `disp.client_id` → `disp.account_id` → `account.frequency`. Here, `disp.client_id` and `disp.account_id` form the Bridge Path. They have weak lexical overlap with the question and are easy to omit when the linker only selects locally relevant endpoints. TAP helps linker recover the schema path needed to connect the condition side and the answer side.

Case 2: Recovering an Answer Endpoint. **Question:** *Was the total cholesterol status for the patient id 2927464 on 1995-09-04 at the normal level?*

The Condition Endpoints are `Laboratory.ID` and `Laboratory.Date`. The Answer Endpoint is weakly signaled: the question uses the phrase "total cholesterol", while the database schema uses the domain-specific abbreviation `T-CHO`.

The predicted TAP gives the tentative answer `T-CHO`=*358*. This answer clue helps the linker infer that `Laboratory.T-CHO` is the relevant Answer Endpoint. The corresponding answer derivation path is `Laboratory.ID` → `Laboratory.Date` → `Laboratory.T-CHO`. This case shows that TAP can

reduce domain-specific lexical gaps and guide the linker toward weakly indicated schema items along the bridge path.

D Prompt Templates

TAP Generation Prompt

```
Given a database schema, primary keys, foreign keys, a
natural-language query, and optional evidence, generate a Tentative
Answer Profile (TAP).

Between [SCHEMA] and [/SCHEMA] are table names, column names, value
types, and example values. primary_keys provide entity identity
clues. foreign_keys provide cross-table relation clues.

The TAP must contain:
Tentative Answer: a specific and plausible answer prior.
Answer Type: one of TEXT, INTEGER, REAL, DATE, DATETIME, OTHER.
CoT Steps: short reasoning steps for deriving the answer, especially
possible Bridge Path relations between condition-side entities and
answer-side entities.

The tentative answer does not need to be the exact gold answer, but
it must be consistent with the schema and example values. Prefer
schema evidence over generic world knowledge. If no concrete value is
provided, still infer a plausible answer prior.

Output ONLY in the following format:

Predicted Answer:
<plausible tentative answer>

Answer Type:
<TEXT|INTEGER|REAL|DATE|DATETIME|OTHER>

CoT Steps:
<short steps>

Do not output extra explanations or markdown.

[SCHEMA]
{db_schema}
[/SCHEMA]

primary_keys = {primary_keys}
foreign_keys = {foreign_keys}

[QUESTION]
{question}
[/QUESTION]

[EVIDENCE]
{evidence}
[/EVIDENCE]
```

Instruction-Model Schema Linking Prompt

```
Between [SCHEMA] and [/SCHEMA] are the table name, a column name, and
the data type of the values in that column.
Table: here it refers to the table name.
Columns: value_type respectively denote the column name and the data
type of the values in that column.

database_schema = {db_schema}

primary_keys = {primary_key}
foreign_keys = {foreign_key}

question = "{question_prompt}"
evidence = "{evidence}"

# Perform TEXT-TO-SCHEMA-LINKING instead of SQL generation.
# DO NOT modify database_schema, primary_keys or foreign_keys.

# Your goal is to decide which columns in which tables are relevant
# to answering the question.

# You must use all available information from:
# database_schema, primary_keys, foreign_keys, and evidence.

# You may also use primary_keys and foreign_keys to infer additional
# columns that are necessary to connect relevant tables.

# Please output only a JSON object:
# {{
# "pairs": [
# {{"table": "table name", "column": "column name"}},
# ...
# ]
# }}
# Do not output any extra text.
```

TAP-Guided SFT Schema Linking Prompt

```
Perform TEXT-TO-SCHEMA-LINKING.
Step1: Use ONLY question and evidence to select the minimal relevant
"table.column".
Step2: Use tentative_answer, answer_type and CoT steps to ONLY ADD
more relevant "table.column" (do NOT remove Step1).Include PKs for
selected tables and both FK sides if used.Output ONLY a JSON list of
"table.column" strings.

{db_schemat}

primary_keys = {primary_key}
foreign_keys = {foreign_key}

[QUESTION]
question = {question_prompt}
[/QUESTION]

[EVIDENCE]
evidence = {evidence}
[/EVIDENCE]

tentative_answer = "{Tentative_answer}"
answer_type = "{Answer_type}"
CoT steps = "{CoT steps}"
```

TAP-Guided SQL Generation Prompt

```
Given the following linked schema, primary keys, foreign keys,
question, evidence, and TAP, your task is to write a valid SQLite SQL
query whose execution results can accurately answer the question.
Evidence and the TAP can help you comprehend the database values and
the intended answer direction.

schema_linking = {schema_linking}

primary_keys = {primary_key}
foreign_keys = {foreign_key}

question = {question}
evidence = {evidence}

tentative_answer = "{Tentative_answer}"
answer_type = "{Answer_type}"
coT steps = "{CoT steps}"

Please only provide a valid SQLite SQL query and do not provide
unnecessary explanations.

SQL:
```