2026.04.23. 세미나

Bridge for Real-world centric Text2SQL task: Schema Linking

NLP&AI 강명훈

Natural Language Processing & Artificial Intelligence

고려대학교 KOREA UNIVERSITY

What we saw in the last session..

LEARNAT: LEARNING NL2SQL WITH AST-GUIDED TASK DECOMPOSITION FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

Text2SQL task에서 **LLM의 sub-task decomposition 능력을 향상**하기 위한 합성 데이터 구축 및 특화 학습 방법론 제안

BIRD-INTERACT: RE-IMAGINING TEXT-TO-SQL EVALUATION VIA LENS OF DYNAMIC INTERACTIONS

Anonymous authors
Paper under double-blind review

Multi-turn conversation, **Agentic 상황**에서의 **Text2SQL interaction을 simulate, evaluate**할 수 있는 평가 프레임워크 제안

In this session

AutoLink: Autonomous Schema Exploration and Expansion for Scalable Schema Linking in Text-to-SQL at Scale

Ziyang Wang[1*], Yuanlei Zheng[1*], Zhenbiao Cao[1], Xiaojin Zhang[2], Zhongyu Wei[3], Pei Fu[4], Zhenbo Luo[4], Wei Chen[1†], Xiang Bai[1]

[1]School of Software Engineering, Huazhong University of Science and Technology
[2]School of Computer Science and Technology, Huazhong University of Science and Technology
[3]School of Data Science, Fudan University
[4]MiLM Plus, Xiaomi Inc.
wangziyang@hust.edu.cn, lemuria_chen@hust.edu.cn

Industrial-scale Text2SQL 을 수행하기 위해 **LLM Context 제약을 극복**하고
Exploratory Agentic Framework인 AutoLink 제안

Schema linking

AutoLink: Autonomous Schema Exploration and Expansion for Scalable Schema Linking in Text-to-SQL at Scale

Ziyang Wang[1*], Yuanlei Zheng[1*], Zhenbiao Cao[1], Xiaojin Zhang[2], Zhongyu Wei[3], Pei Fu[4], Zhenbo Luo[4], Wei Chen[1†], Xiang Bai[1]

[1]School of Software Engineering, Huazhong University of Science and Technology
[2]School of Computer Science and Technology, Huazhong University of Science and Technology
[3]School of Data Science, Fudan University
[4]MiLM Plus, Xiaomi Inc.
wangziyang@hust.edu.cn, lemuria_chen@hust.edu.cn

AAAI 2026 Main

AutoLink

- **Motivations: LLM context window is a critical bottleneck for industrial scale Text2SQL**

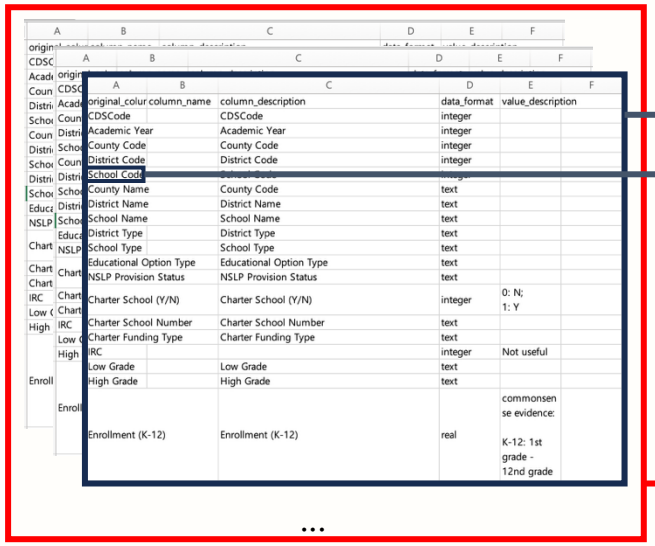


Table-level meta data

Column-level meta data

Whole table meta data per DB

Dataset	# Test Examples	# Test DB	# Col / DB	# Tok. / SQL	# Func. / SQL	External Knowledge	SQL Dialect	Project Level
WikiSQL (Zhong et al., 2017)	15,878	5,230	6.3	12.2	0.0	✘	✘	✘
Spider 1.0 (Yu et al., 2018)	2,147	40	27.1	18.5	0.0*	✘	✘	✘
KaggleDBQA (Lee et al., 2021)	272	8	23.4	13.8	0.0	✓	✘	✘
SEDE (Hazoom et al., 2021)	857	1	212.0	46.9	1.4	✘	✘	✘
BIRD (Li et al., 2024b)	1,789	15	54.2	30.9	0.4*	✓	✘	✘
Spider 2.0-lite	547	158	803.6	144.5	6.5	✓	✓	✘
Spider 2.0-snow	547	152	**812.1**	**161.8**	6.8	✓	✓	✘
Spider 2.0	632	213	743.5	148.3	**7.1**	✓	✓	✓

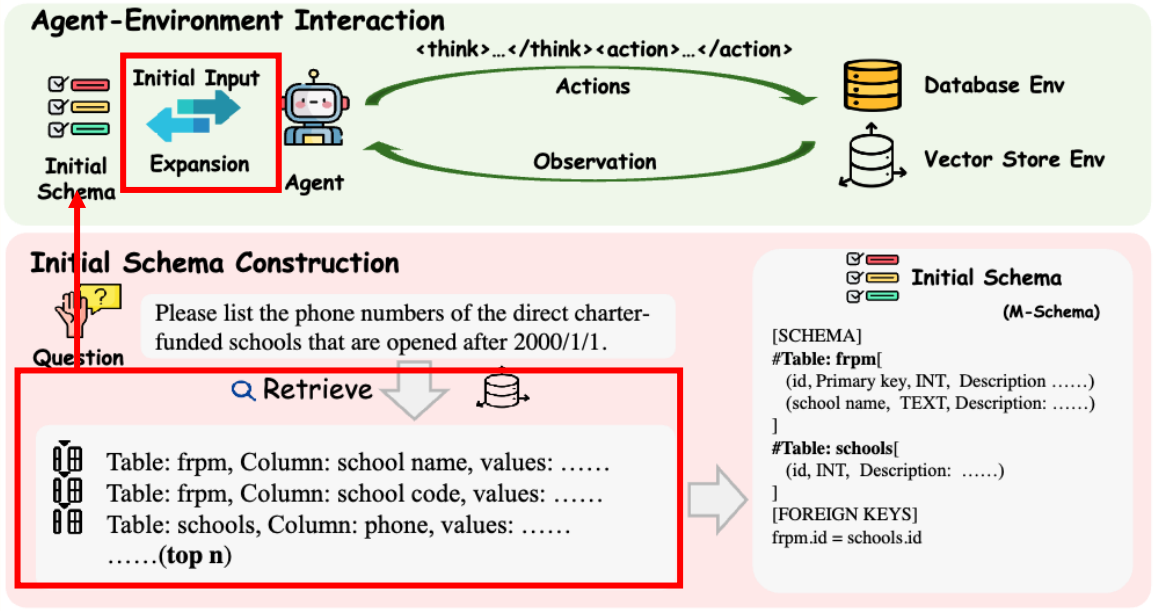
Spider 2.0 benchmark Statistics

Input Schema Burden

- 산업 현장에서 활용되는 수준의 복잡하고 거대한 DB 기반으로 작업을 수행하기 위해서는 해당 **DB와 관련된 외부 지식**과
  해당 **DB 구조 및 내용**에 대한 이해가 필수
- 다양한 Column과 DataType을 가지는 산업 수준 DB를 바탕으로 Text2SQL task를 수행하기 위해서는 DB를 설명하는 Schema(Metadata)를 주된 입력으로 활용하여 task를 수행 해야함
  ← LLM Context window를 넘어서는 DB Metadata의 길이와 복잡성 문제 직면

AutoLink

- **Idea: Expand & Explore the database with Agent-Environment Interaction**



- 질의와 관련된 Column 단위 Top-k 검색 Meta data 집합을 바탕으로 초기 질의 관련 DB 정보를 획득

- Agent가 주어진 초기 정보를 바탕으로 DB Environment와의 상호작용을 통해 추가 정보를 Explore, 활용 DB metadata를 Expansion하는 방식으로 Text2SQL에서의 LLM Context window 문제 해결

AutoLink

- **Overview of AutoLinK: Autonomous Schema Exploration and Expansion**

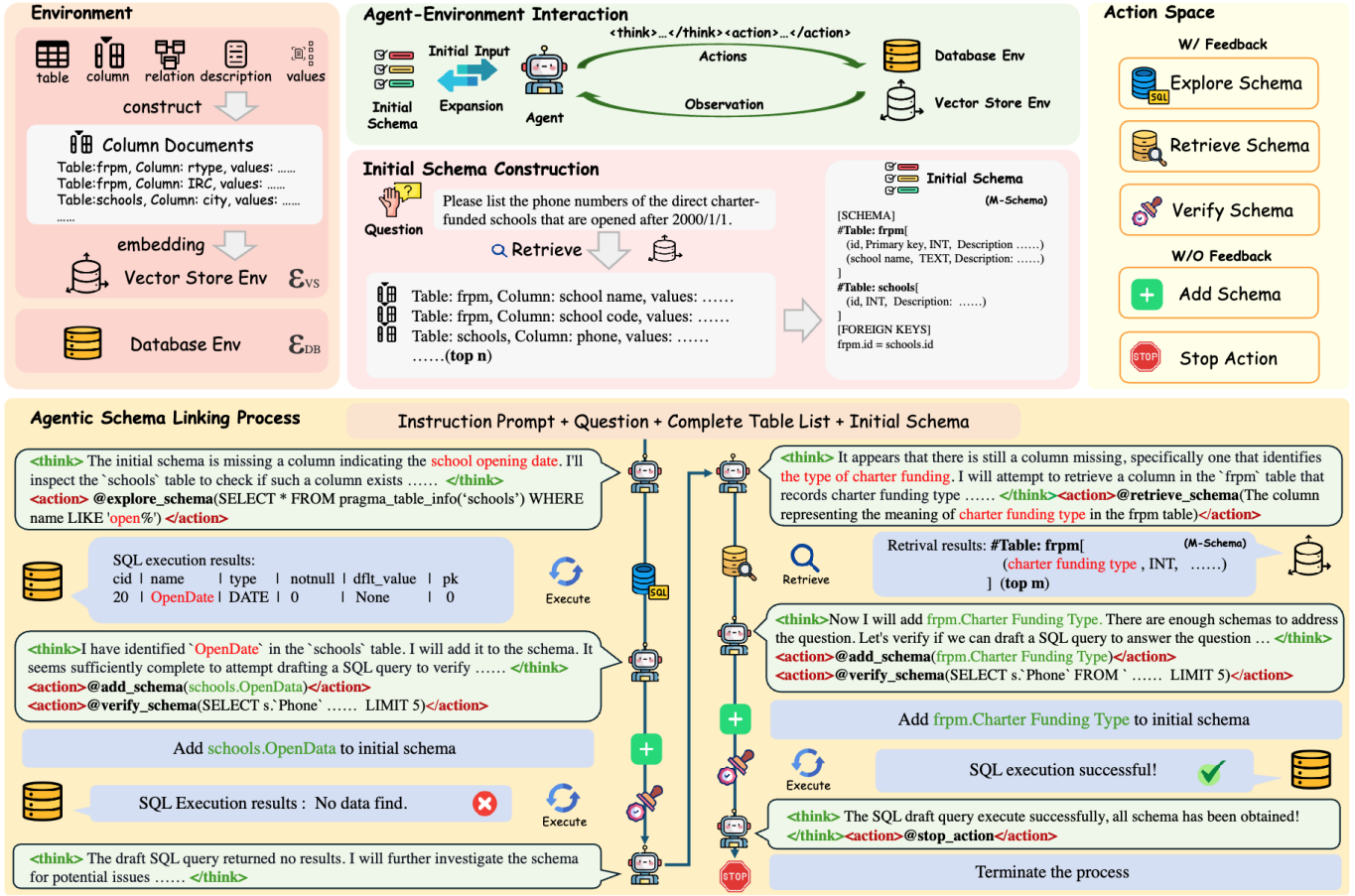


Figure 1: Overview of AutoLink: iterative interactive agentic schema linking framework.

Step 1 Initial Schema Generation

- 질의 Q와 관련된 초기 DB 정보를 얻기 위하여 $\varepsilon_{\mathrm{DB}}$, $\varepsilon_{\mathrm{vs}}$를 구축
- $\varepsilon_{\mathrm{vs}}$로부터 top-k column 정보를 검색하여 초기 질의 해결 관련 DB 정보인 $S_{\mathrm{linked}}^{(0)}$를 구성

Step 2 Agent-Environment Interaction

- $S_{\mathrm{linked}}^{(0)}$ 를 바탕으로 Agent가 환경 $\varepsilon_{\mathrm{DB}}$, $\varepsilon_{\mathrm{vs}}$ 와의 협동을 Action을 통해서 수행하여 **최종 답변 생성에 필요한 Table, Column 정보를 추출**
- 질의 해결에 추가로 **필요한 정보를 탐색 (Explore)**하고 기존의 **Meta data정보를 확대 (Expand)**하는 Multi-turn 전략으로 부족한 DB metadata 정보를 획득
- DB 정보를 획득하기 위해 **SQL 구문 생성 및 실행**을 활용하여 **DB 환경과의 직접적인 접근 & 피드백 정보를 활용**하여 한정된 정보로 질의 해결에 필요한 Table, Column 정보 획득을 목표

AutoLink - Methods

- **Problem Definition**

  - AutoLink는 사용자의 **질문 Q, Task instruction I**, 질의 해결에 필요한 **DB에 속한 모든 Table 이름 list T** 가 초기 입력으로 주어지고 이를 바탕으로 Agent와 Environment의 상호작용을 통해서 Schema linking, SQL generation을 수행

$$\mathcal{E}_{\text{DB}} : \text{SQL} \rightarrow R_{\text{SQL}}, \qquad \mathcal{E}_{\text{VS}} : (q_{\text{nl}}, K, \mathcal{C}_{\text{excl}}) \xrightarrow{\text{retrieve top-}K\text{ columns}} S_{\text{subset}},$$

Agent Action

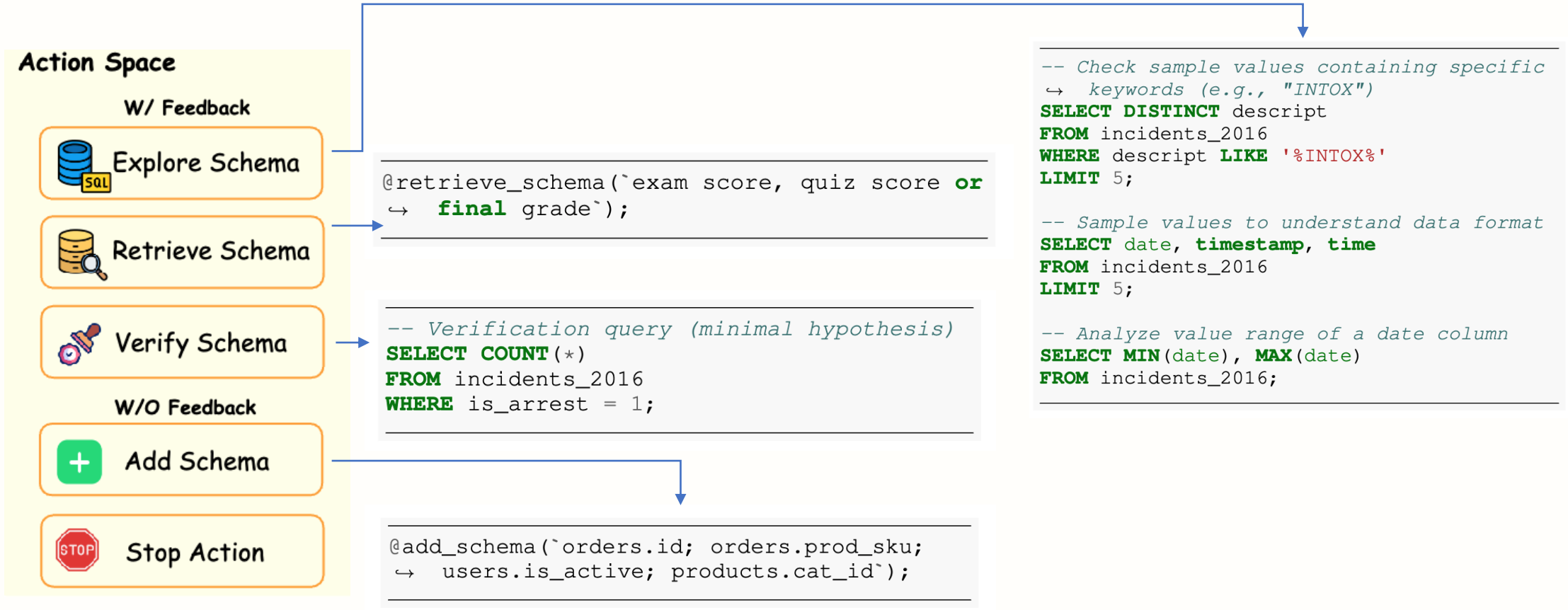
- θ_t: Reasoning Trace within <think></think> tags
- A_t: Set of actions within <actions></actions>tags
- H_{t+1}: (θ_t, A_t, O_t)

$$(\theta_t, A_t) = \pi(H_t). \qquad O_t = \mathcal{E}(A_t).$$

```
[DB_ID] ga360
# Table ga_sessions_20160810
(visitNumber: INT64,
↪  Examples:[2,1,1,1,1], The session
↪  number for this user. If this is the
↪  first session, then this is set to
↪  1)
(fullVisitorId: STRING,
↪  Examples:["1906","7880","5836",
"6743","0244"], The unique visitor ID.)
(date: STRING,
↪  Examples:["20160810","20160810",
"20160810","20160810","20160810"]: The
↪  date of the session in YYYYMMDD
↪  format.)
```

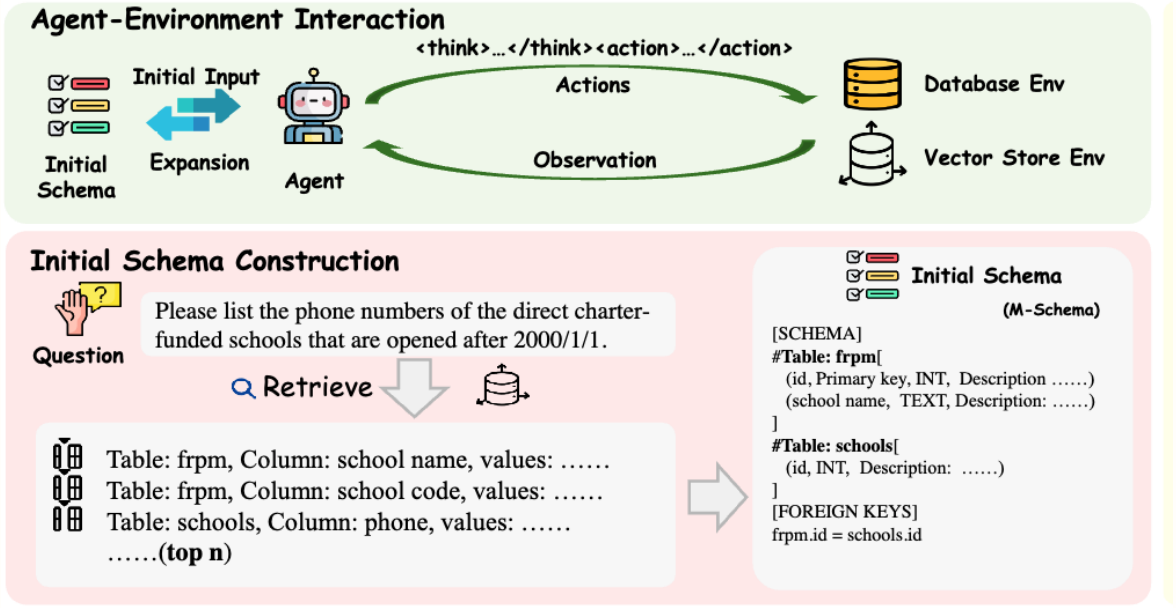
AutoLink - Methods

- **Action Space for AutoLink**



AutoLink - Methods

- **Step 1: Initial Schema Generation**



Algorithm 1: AutoLink: Autonomous Schema Exploration and Expansion

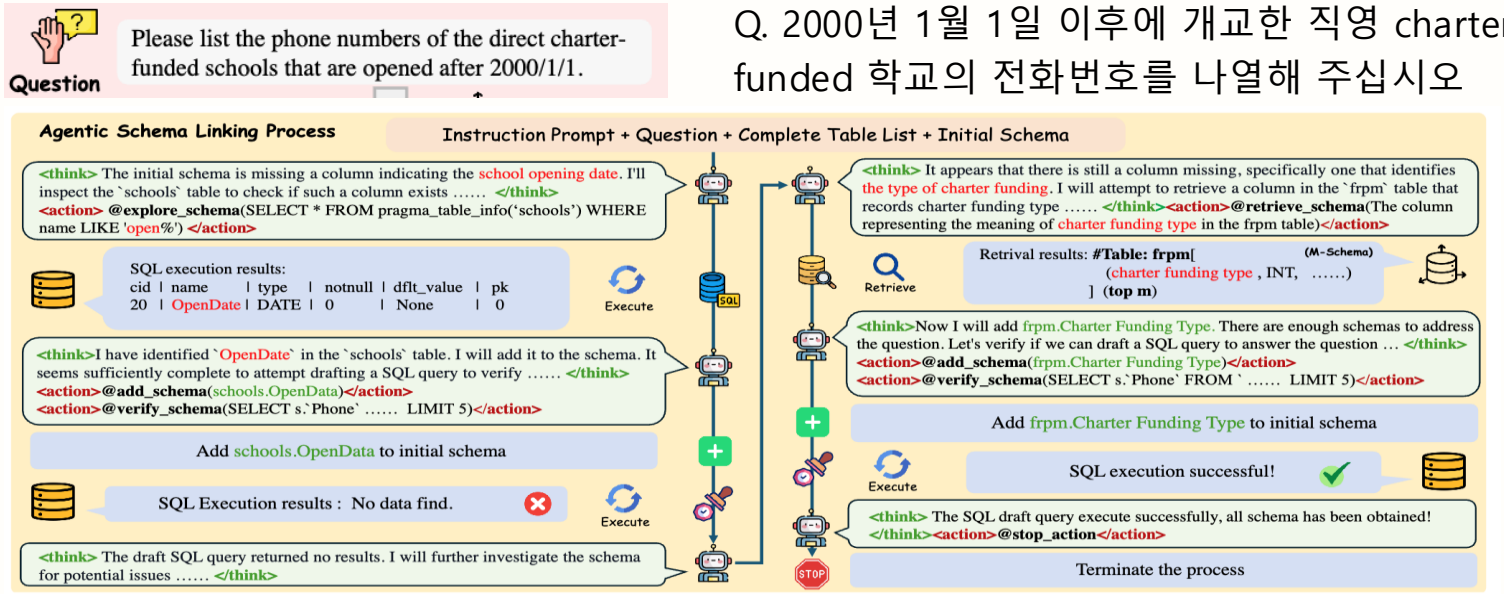
Require: Database Environment $\mathcal{E}_{\text{DB}}$, Schema Vector Store Environment $\mathcal{E}_{\text{VS}}$
Require: Instruction Prompt I, User Question Q, Full Database Schema S_{full}, All Table Names T extracted from S_{full}
Require: Maximum Interaction Turns $T_{\max}$, Initial Retrieval Count n, Targeted Retrieval Count m
Ensure: Final Linked Schema S_{linked}

Initialize:
  History $H \leftarrow$ empty string
  Current Linked Schema $S_{\text{linked}} \leftarrow \emptyset$
  Excluded Columns $\mathcal{C}_{\text{excl}} \leftarrow \emptyset$
Step 1: Initial Schema Generation
  $S_{\text{initial}} \leftarrow \mathcal{E}_{\text{VS}}(Q, n, \mathcal{C}_{\text{excl}})$ {Retrieve top-n columns from }
  $S_{\text{linked}} \leftarrow S_{\text{initial}}$
  $\mathcal{C}_{\text{excl}} \leftarrow$ columns in S_{initial}
  $H_0 \leftarrow$ Construct initial context with $I, Q, T, S_{\text{initial}}$

Step 1에서는 질의 해결에 필요한 기초 DB 정보인 $S_{\text{linked}}^{(0)}$ 를 Vector Store 검색을 통해 생성

AutoLink - Methods

- **Step 2: Agent-Environment Interaction**



Q. 2000년 1월 1일 이후에 개교한 직영 charter-funded 학교의 전화번호를 나열해 주십시오

Step 2: Agent-Environment Interaction
for $t = 1$ **to** $T_{\max}$ **do**
  Construct current prompt $P_t \leftarrow H_{t-1}$
  $(\theta_t, A_t) \leftarrow \pi(P_t)$ {LLM agent generates reasoning and actions}
  $O_t \leftarrow$ empty string {Initialize observations for current turn}
  stop_flag $\leftarrow$ FALSE
  for each action $a \in A_t$ **do**
    if $a =$ @explore_schema(sql_query) **then**
      $R_{\text{SQL}} \leftarrow \mathcal{E}_{\text{DB}}$(sql_query)
      Append R_{SQL} to O_t
    else if $a =$ @retrieve_schema(nl_query) **then**
      $S_{\text{retrieved}} \leftarrow \mathcal{E}_{\text{VS}}$(nl_query, $m, \mathcal{C}_{\text{excl}}$)
      Append $S_{\text{retrieved}}$ to O_t
    else if $a =$ @verify_schema(sql_query) **then**
      $R_{\text{SQL}} \leftarrow \mathcal{E}_{\text{DB}}$(sql_query)
      Append R_{SQL} to O_t
    else if $a =$ @add_schema(schemas) **then**
      $S_{\text{added}} \leftarrow$ extract full metadata of schemas
      $S_{\text{linked}} \leftarrow S_{\text{linked}} \cup S_{\text{added}}$
      $\mathcal{C}_{\text{excl}} \leftarrow \mathcal{C}_{\text{excl}} \cup$ columns in S_{added}
    else if $a =$ @stop_action() **then**
      stop_flag $\leftarrow$ TRUE
    end if
  end for
  $H_t \leftarrow H_{t-1} \cup \{(\theta_t, A_t, O_t)\}$ {Update history}
  Update prompt P_{t+1} with H_t
  if stop_flag is TRUE **then**
    BREAK
  end if
end for
return S_{linked}

1) @explore_schema. 학교 개교 정보를 담은 column을 탐색하는 SQL 구문 생성 및 ε_{DB} 에서실행 → OpenDate column 발견
2) @add_schema, @verify_schema. OpenDate column을 S_{linked} 에 추가, 업데이트 된 S_{linked}를 활용해 정답 도출 시도 → 에러 발생
3) @retrieve_schema. 정답 도출을 위해 추가 정보가 필요하다는 점을 확인하여 charter funding type과 관련된 column을 추출하도록 column 검색 실시 → top-m column 검색
4) @add_schema, @verify_schema. 검색된 column중 Charter Funding Type column을 S_{linked}에 추가, 업데이트 된 S_{linked}를 활용해 정답 도출 시도 → 결과 도출
5) @stop_action. @verify_schema로 결과가 도출되고 S_{linked}가 정답 생성에 충분하 정보를 담았다고 판단하여 process를 멈춤

AutoLink - Methods

- **Step 3: SQL generation**

1. **SQL Candidate Generation via LLM Policy: Self-Consistency 방식**으로 정답 도출을 위한 **N=5개의 SQL query를 생성**

$$\{\mathrm{SQL}_i\}_{i=1}^{N} = \pi_{\mathrm{SQL}}(Q, S_{\mathrm{linked}})$$

2. **Iterative Syntactic Correction:** N개 생성된 SQL 구문별로 Multi-turn 방식으로 t번의 대화동안 SQL 구문과 실행결과에서 나타나는 Error type을 입력으로 하여서 **SQL query의 Syntactic Error를 수정하는 과정 실시**. 수정된 SQL query의 실행에 문제가 없을 경우 수정을 종료

$$\mathcal{C}_j = \mathcal{C}_{j-1} \cup \{(\mathrm{SQL}_i^{(j-1)}, \mathrm{error}_j)\}. \qquad \mathrm{SQL}_i^{(j)} = \pi_{\mathrm{SQL}}(\mathcal{C}_j). \qquad \mathrm{SQL}_i^{\mathrm{corr}} = \mathrm{SQL}_i^{(k)},$$

3. Majority Voting: 수정된 SQL 구문들을 실행하여 결과값이 같은 구문들을 하나의 그룹 $\mathcal{G}$ 로 묶은 뒤, query의 개수가 가장 많은 Group을 최종 정답 생성 그룹으로 선정 <- Query Consistency가 가장

$$\mathcal{G}_{\max} = \left\{ \mathcal{G}_j \mid |\mathcal{G}_j| = \max_k |\mathcal{G}_k| \right\}$$

4. Final SQL Selection: 선정된 그룹 $\mathcal{G}$의 개수를 기준으로 아래의 규칙에 따라서 최종 평가 SQL 구문을 선정

$$\mathrm{SQL}^* = \begin{cases} \mathrm{RandomSelect}(\mathcal{G}_{\max}), & \text{if } |\mathcal{G}_{\max}| = 1 \\ \underset{\mathrm{SQL} \in \mathcal{S}_{\mathrm{tie}}}{\mathrm{argmax}}\, \mathrm{PairwiseWins}(\mathrm{SQL}), & \text{otherwise} \end{cases}$$

AutoLink – Experimental Setups

- **Experimental dataset**

Dataset	# Test Examples	# Test DB	# Col / DB	# Tok. / SQL	# Func. / SQL	External Knowledge	SQL Dialect	Project Level
WikiSQL (Zhong et al., 2017)	15,878	5,230	6.3	12.2	0.0	✘	✘	✘
Spider 1.0 (Yu et al., 2018)	2,147	40	27.1	18.5	0.0*	✘	✘	✘
KaggleDBQA (Lee et al., 2021)	272	8	23.4	13.8	0.0	✓	✘	✘
SEDE (Hazoom et al., 2021)	857	1	212.0	46.9	1.4	✘	✘	✘
BIRD (Li et al., 2024b)	1,789	15	54.2	30.9	0.4*	✓	✘	✘
Spider 2.0-lite	547	158	803.6	144.5	6.5	✓	✓	✘
Spider 2.0-snow	547	152	**812.1**	**161.8**	6.8	✓	✓	✘
Spider 2.0	632	213	743.5	148.3	**7.1**	✓	✓	✓

Spider 2.0-lite

BIRD-dev	Simple	Moderate	Challenging	Total
	925	465	144	1,534

BIRD-dev

{'question_id': 0,
'db_id': 'california_schools',
'question': 'What is the highest eligible free rate for K-12 students in the schools in Alameda County?',
'evidence': 'Eligible free rate for K-12 = `Free Meal Count (K-12)` / `Enrollment (K-12)`',
'SQL': "SELECT `Free Meal Count (K-12)` / `Enrollment (K-12)` FROM frpm WHERE `County Name` = 'Alameda' ORDER BY (CAST(`Free Meal Count (K-12)` AS REAL) / `Enrollment (K-12)`) DESC LIMIT 1",
'difficulty': 'simple'}

AutoLink– Experimental Setups

- **Implementation details**

  Policy Model (671B)
  - Schema linking policy : DeepSeek-V3
  - SQL generation policy: DeepSeek-R1, DeepSeek-V3

- **Metric**

Schema linking Evaluation Metric

- **Strict Recall Rate (SRR)**

$$\text{SRR} = \frac{\sum_{i=1}^{n} \mathbb{I}(\tilde{S}_i \supseteq S_{\text{gt},i})}{n}$$

SQL Query generation Evaluation Metric

- **Execution Accuracy (EX)**

$$\text{EX} = \frac{\sum_{i=1}^{N} \mathbb{1}(\hat{V}_i, V_i)}{N}$$

$$\mathbb{1}(\hat{V}_i, V_i) = \begin{cases} 1 & \text{if} \quad \hat{V}_i = V_i \\ 0 & \text{if} \quad \hat{V}_i \neq V_i \end{cases}$$

AutoLink – Result

- **Main result: Schema linking performance of AutoLink**

Method	Full Input	Bird Dev			Spider2.0-Lite					
		SRR$^{\uparrow}$	$\bar{C}^{\downarrow}$	Avg. Tokens$^{\downarrow}$	SRR$^{\uparrow}_{all}$	SRR$^{\uparrow}_{bq}$	SRR$^{\uparrow}_{sf}$	SRR$^{\uparrow}_{local}$	$\bar{C}^{\downarrow}$	Avg. Tokens$^{\downarrow}$
DE-SL (BGE-Large)	✓	35.5	35.7	–	43.6	28.2	57.1	87.5	153.8	–
CE-SL (BGE-reranker)	✓	72.4	35.7	–	57.6	42.3	72.6	**95.8**	153.8	–
MCS-SQL	✓	85.7	7.5	29.8K	58.9	57.8	54.8	79.2	45.1	168.9K
SQL-to-Schema	✓	92.5	13.5	19.4K	64.0	64.8	54.8	91.7	49.0	171.9K
CHESS	✓	89.7	**4.5**	–	–	–	–	–	–	–
RSL-SQL	✓	93.3	13.0	14.8K	52.0	52.8	44.1	75.0	25.8	29.2K
LinkAlign	✗	–	–	–	36.4	22.5	51.2	66.7	**21.1**	66.7K
AutoLink (ours)	✗	**97.4**	35.8	**8.0K**	**91.2**	**93.7**	**85.7**	**95.8**	159.4	**21.2K**

Table 1: **Performance comparison of schema linking on Bird Dev and Spider 2.0-Lite**. We report the overall Strict Recall Rate (SRR$_{all}$) for both datasets. For Spider 2.0-Lite, SRR is further broken down by SQL dialect: BigQuery (SRR$_{bq}$), Snowflake (SRR$_{sf}$), and SQLite (SRR$_{local}$). $\bar{C}$ denotes the average number of columns included in the simplified schema (S_{linked}) after schema linking. **Full Input** indicates whether the entire database schema is provided to the LLM or if all database schemas are iterated through.

- 제안한 AutoLink가 Baseline 대비 우수한 Schema linking 정확도를 보임. 특히 Spider2.0-lite에서 Baseline 대비 성능 차이가 두드러짐
- Spider 2.0-lite가 BIRD대비 DB metadata의 길이와 복잡도 측면에서 매우 높기 때문에 다른 Baseline이 성능이 저하되는 것으로 추정. 반면 AutoLink는 처음부터 제한된 정보를 바탕으로 Explore & Expand하기 때문에 Context Length 문제를 완화하여 Schema linking의 정확성을 담보
- Baseline 측면에서는 CHESS, LinkAlign같은 초기 주어진 정보에서 Noise를 줄이는 방향으로 접근하는 baseline은 성능이 가장 뒤떨어진다는 점을 발견

AutoLink – Result

- **Main result: SQL generation performance**

Method	Model	EX↑	Avg. Tokens↓
Spider-Agent	QwQ	11.33	–
	GPT-4o	13.16	–
	DeepSeek-R1	13.71	–
	o1-preview	23.03	–
	o3-mini	23.40	–
	Claude-3.7-Sonnet	28.52	–
CHESS	GPT-4o	3.84	–
LinkAlign	DeepSeek-R1	33.09	–
RSL-SQL†	DeepSeek-R1	30.53	50.0K
REFORCE†	DeepSeek-R1	29.62	81.1K
REFORCE	o1-preview	30.35	81.1K
REFORCE	GPT-o3	**37.84**	81.1K
AutoLink	DeepSeek-R1	34.92	**38.0K**

Table 2: Execution Accuracy (EX) comparison of different methods on the Spider 2.0-Lite dataset. † indicates results obtained through independent reproduction. Avg. Tokens indicates the average number of tokens comsumed to generate a SQL.

Method	Model	EX↑	Avg. Tokens↓
MAC-SQL	GPT-4	59.39	**6.8K**
TA-SQL	GPT-4	56.19	7.3K
RSL-SQL	GPT-4o	67.21	7.5K
CHESS	Gemini-1.5-Pro	68.31	14.5K
AutoLink	DeepSeek-v3	66.36	8.0K
AutoLink	Gemini-1.5-Pro	**68.71**	8.0K

Table 3: Execution Accuracy (EX) comparison of different methods the Bird Dev.

- SQL generation에 특별한 기여가 없는 상황에서도 Spider 2.0, BIRD 에서 AutoLink의 성능이 타 Basline 대비 우수한 SQL 정확도를 달성
- 동일 Backbone을 사용한 경우에 한하여 AutoLink는 타 baseline 대비 우수한 성능을 달성
- Avg. Tokens가 다른 baseline 대비 적음에도 높은 성능을 달성한 점을 볼 때 AutoLink의 Schema linking 정확성을 간접적으로 파악할 수 있음

AutoLink – Result

- **Ablation Study: Impact of ablating action space and Environment interaction**

Method	$SRR_{n=5}$	$SRR_{n=50}$	$SRR_{n=100}$
AutoLink	**79.2**	**88.0**	**91.2**
- w/o Verify Schema	$77.2_{\downarrow 2.0}$	$86.8_{\downarrow 1.2}$	$89.6_{\downarrow 1.6}$
- w/o Explore Schema	$76.8_{\downarrow 2.4}$	$84.8_{\downarrow 3.2}$	$88.8_{\downarrow 2.4}$
- w/o Retrieve Schema	$72.4_{\downarrow 6.8}$	$80.4_{\downarrow 7.6}$	$84.5_{\downarrow 6.7}$

Table 4: Ablation study on the impact of different schema linking actions under varying numbers of initial candidates n on Spider 2.0-Lite.

- Schema Retrieval 제거 하는게 가장 큰 성능 저하를 불러 일으킴 → 초기 검색 결과만으로 Schema linking을 성공할 수 없음을 암시
- Explore Schema 제거시 두번째로 큰 성능 저하가 발생 → 질의, DB와 관련된 Ambiguity 해소에 Explore Schema가 기여함을 암시
- Verify Schema 제거시에도 성능 저하가 발생 → Self-assessment가 성능 향상에 기여함을 암시

AutoLink – Result

- **Scalability Analysis of Different Database Scales**

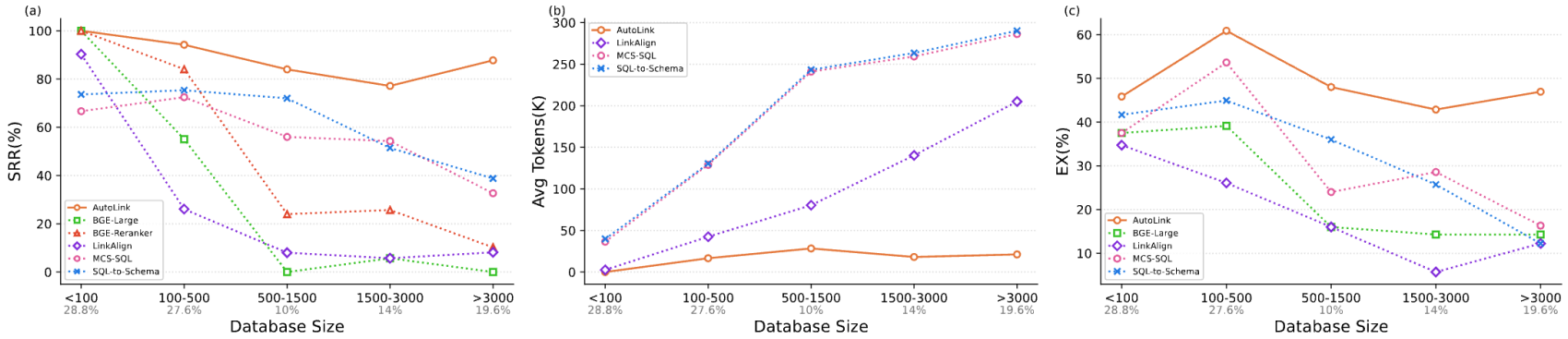


Figure 2: Scalability comparison across databases on Spider 2.0-Lite of varying sizes in terms of (a) Strict Recall Rate (SRR↑), (b) Average Tokens Consumption (Avg. Tokens↓), and (c) Execution Accuracy (EX↑). Percentages below each bin indicate the proportion of databases within each size range.

- Column의 개수에 따라서 Database size interval을 나눠서 분석을 시도
- AutoLink는 다른 Baseline 대비 SRR, Avg Tokens, EX 모든 측면에서 Database scale에 영향을 덜 받으면서 안정적인 성능을 내는 것을 확인할 수 있음

Schema Linking in the Wild: Clause-Aware Grounding with DB Disambiguation under Synthetic Distractor Pressure

Schema Linking in the Wild

- **AutoLink의 우수성**
  - 적은 초기 DB 데이터로 Agent –Environment Interaction을 통한 Schema linking을 수행하는 방법론의 효과성 증명
  - DB의 Scale에 덜 영향을 받는 강건한 방법론 제안
  - Schema linking의 performance가 최종 SQL generation 성능에 영향을 주는 점을 증명

- **AutoLink및 기존 Schema linking 방법론 한계**
  - **Model Scalability**: AutoLink의 Agent은 DeepSeek-V3이며 671B 모델, LinkAlign의 Agent는 GLM-4-air, GPT-4
    ← 제안하는 framework를 수행하기 위해서 **매우 비싼 모델**이 필요
  - **Unrealistic Settings**: Autolink의 Explore & Expand 방식 자체는 타당하나 그 방식이 작동될 수 있는 이유는 Explore & Expand의 범위가 DB 1개로 한정되는 것에 있음. 이러한 DB 정보를 주고 SQL 구문을 생성하는 기존 Text2SQL 방식의 문제점을 모두가 공유
    ← 사용자가 어떤 DB를 활용할지 모르는 상황, 즉 **Multi-DB 상황에서는 작동하기 어려운 방식**

Schema Linking in the Wild

- **How to implement Realistic Schema linking settings?**
  - RAG 상황을 가정하여 질의 해결 가능 DB 정보와 질의 해결에 관련 없는 **Distractor DB가 입력으로 포함된 평가 데이터셋**을 구축
    - Ground truth Schema Linking 결과를 기반으로 column 단위 distraction을 다양하게 주는 방식으로 Distractor DB metadata 생성 ← Contextual Error Injection
    - Distractor DB metadata의 구조적 타당성 확인을 위한 데이터 verification 방법론 구축 ← Structural error prevention
- **How to utilize weak LLM for Schema Linking?**
  - 하나의 **LLM이 Two-stage 전략으로 작동할 수 있는 framework** 제안
    - Stage 1: Mention & Entity Extraction: 자연어 질의와 Gold + Distractor DB metadata set이 주어졌을 때 관련 있는 mention, entity를 추출하는 역할
      → Schema linking에 필요한 관련성이 높은 table, column set을 최대한 많이 뽑도록 **recall을 높이는 전략을 추구**
    - Stage 2: Clause-aware Schema linking: Stage 1에서 추출된 Mention, Entity를 활용하여 최종 사용자 질의에 응답할 수 있는 table, column을 추출하는 schema linking을 실시
      → Entity를 Table/Column에 mapping할 때, **SQL의 어느 절(Clause: SELECT, WHERE, GROUP BY 등)에 속하는지**, 어떤 **연산자(Aggregation 등)와 결합되는지 까지를 예측**하면서 Schema linking을 진행

Schema Linking in the Wild

- **Distractor DB creation procedure**

내용적 중복 (Example Overlap)
정답 DB의 실제 예시 값(oakland, alameda, 2014-2015)을 가짜 DB 컬럼에 동일 주입 → 모델이 값을 보고 정답으로 착각 — VALUE

의미론적 미세 변형 (Semantic Shift)
도메인·범주는 유사하나 의도가 어긋난 컬럼 생성 (cards.name vs. set_name) → 모델이 도메인만 보고 오선택 — SEMANTIC

관계성 단절 (Relationship Break)
개별 카드를 특정할 수 없는 무의미한 FK 부여 (card_id → set_id) → FK 존재만 보고 조인 가능으로 오판 — JOIN

시간 단위 불일치 (Temporal Shift)
날짜가 필요한 질문에 연도 컬럼 제시 (races.date vs. season_year) → 시간 컬럼이라는 이유로 답변 가능하다고 속음 — TEMPORAL

집계 수준 불일치 (Granularity Shift)
집계 단위 스케일 불일치 컬럼 제시 (country vs. region, 학교 단위 vs. 학군 단위) → 더 넓은 단위를 정답으로 선택하도록 유도 — GRANULARITY

Schema Linking in the Wild

- **Distractor DB creation procedure**

```
【DB_ID】 california_schools
【Schema】
# Table: frpm
[
(CDSCode:INTEGER, Examples: [01100170118489, 01100170112607, 011
(Academic Year:INTEGER, Examples: [2014-2015]),
(County Code:INTEGER, Examples: [01]),
(District Code:INTEGER, Examples: [10017]),
(School Code :INTEGER, Examples: [0109835, 0118489, 0112607]),
(County Name:TEXT, Examples: [Alameda]),
(District Name :TEXT, Examples: [Alameda County Office of Educat
(School Name:TEXT, Examples: [Aspire California College Preparat
(District Type:TEXT, Examples: [County Office of Education (COE)
(School Type :TEXT, Examples: [K-12 Schools (Public), High Schoo
(Educational Option Type:TEXT, Examples: [Traditional]),
(NSLP Provision Status:TEXT),
(Charter School (Y/N):INTEGER, Examples: [1]),
(Charter School Number:TEXT, Examples: [0811, 1049, 0728]),
(Charter Funding Type:TEXT, Examples: [Directly funded]),
(IRC:INTEGER, Examples: [1]),
```

컬럼에

et_name

→ FK 존

→ 시간

vs. 학군

```
【DB_ID】 california_student_wellness
【Schema】
# Table: campus_directory
[
(campus_id:INTEGER, Primary Key, Examples: [120045, 120318, 1210
(reporting_year:TEXT, Examples: [2014-2015, 2013-2014]),
(district_label:TEXT, Examples: [Alameda Unified, Hayward Unifie
(site_label:TEXT, Examples: [Lincoln Middle, Fremont High, Bayvi
(campus_category:TEXT, Examples: [Elementary, Middle, High]),
(low_grade_flag:TEXT, Examples: [K, 6, 9]),
(high_grade_flag:TEXT, Examples: [5, 8, 12]),
(charter_flag:BOOLEAN, Examples: [true, false]),
(street_location:TEXT, Examples: [1250 Atlantic Ave, 4610 Foothi
(municipality:TEXT, Examples: [Alameda, Oakland, San Leandro]),
(postal_code:TEXT, Examples: [94501, 94601, 94577]),
(latitude:REAL, Examples: [37.76512, 37.77483, 37.72451]),
(longitude:REAL, Examples: [-122.24118, -122.21456, -122.15674])
(calendar_model:TEXT, Examples: [Traditional, Year-Round, Modifi
(start_month:TEXT, Examples: [August, July]),
(end_month:TEXT, Examples: [June, May]),
(active_status:TEXT, Examples: [Active, Inactive])
]
# Table: fitness_assessments
[
(assessment_id:INTEGER, Primary Key, Examples: [500901, 500944,
(reporting_year:TEXT, Examples: [2014-2015, 2013-2014]),
(campus_id:INTEGER, Examples: [120045, 120318, 121007]),
```

VALUE

EMANTIC

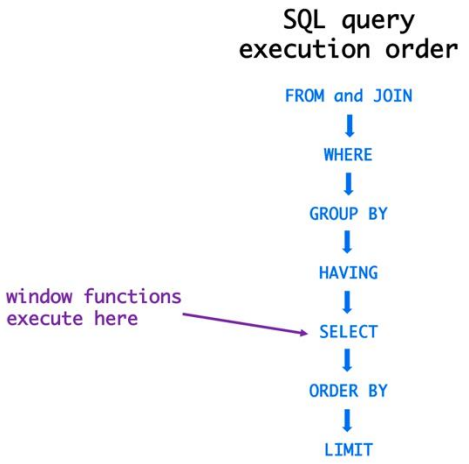
JOIN

MPORAL

ANULARITY

Schema Linking in the Wild

- **Clause-aware Schema linking**



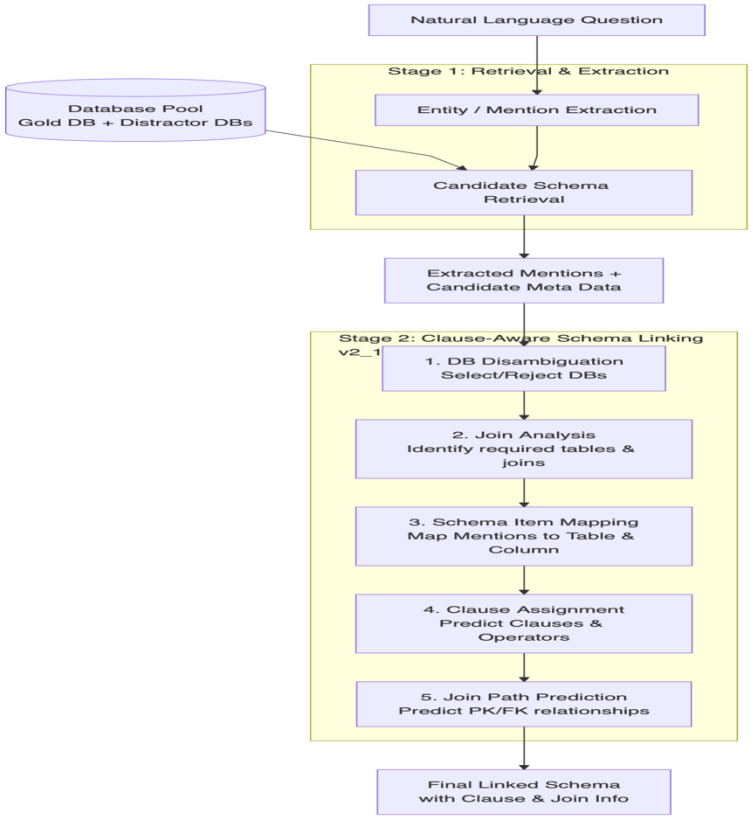

```json
{
  "join_analysis": {
    "needs_join": true,
    "tables_involved": ["<table_1>", "<table_2>"]
  },
  "schema_linking_predictions": [
    {
      "mention": "<exact mention string from the input list>",
      "schema_items": [
        {
          "table": "<table name as shown in the schema>",
          "column": "<column name as shown in the schema>",
          "clauses": ["<clause_label>", "..."],
          "elements_by_clause": {
            "<clause_label>": ["<operator_or_function>", "..."]
          }
        }
      ]
    }
  ],
  "join_path": [
    "<table_a>.<column_i> = <table_b>.<column_j>"
  ]
}
```

"question": "Provide ID, sex and age of patient who has blood glucose (GLU) not within normal range but with total cholesterol(T-CHO) within normal range."

```
"join_analysis": {
 "needs_join": true,
 "tables_involved": ["Patient", "Examination", "Laboratory"]
},
"schema_linking_predictions": [
 {
  ... {
  "mention": "blood glucose (GLU) not within normal range",
  "schema_items": [
   { "table": "Laboratory", "column": "GLU", "clauses": ["WHERE"], "elements_by_clause": { "WHERE": ["NOT_EQUAL"] } }
  ]
 },
 {
  "mention": "total cholesterol(T-CHO) within normal range",
  "schema_items": [
   { "table": "Laboratory", "column": "T-CHO", "clauses": ["WHERE"], "elements_by_clause": { "WHERE": ["EQUAL"] } }
  ]
 }
],
"join_path": [
 "Patient.ID = Examination.ID",
 "Patient.ID = Laboratory.ID"
]
}
```

Schema Linking in the Wild

- **Framework structure diagram**



Schema Linking in the Wild

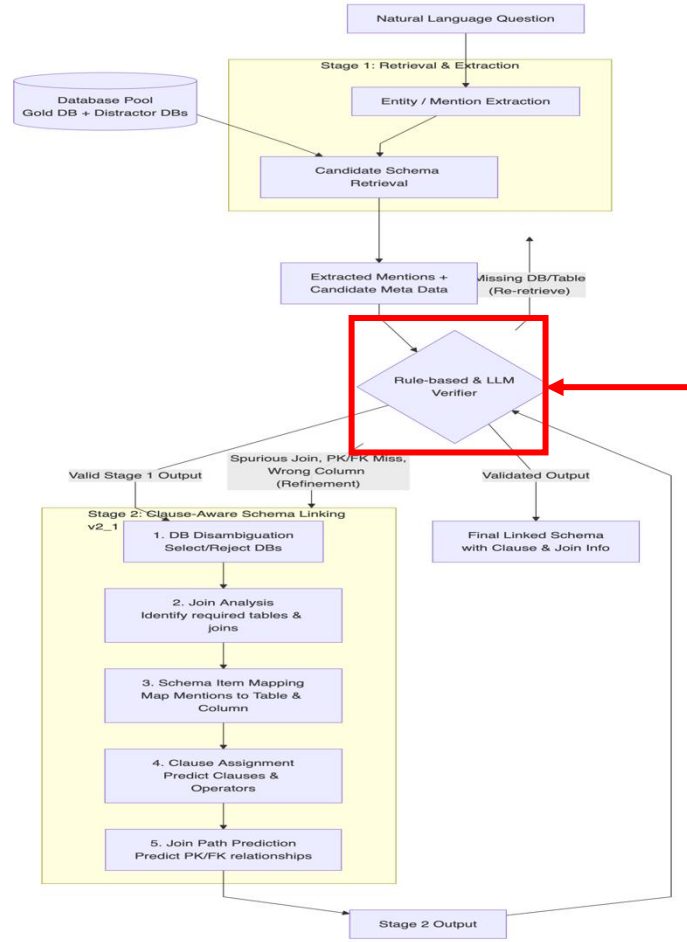
- **Experimental Results: Schema linking**

Qwen/Qwen2.5-Coder-7B-Instruct 활용 추론 실시

	DB Acc	Table Acc	Col Acc	Table Acc (cond.)	Col Acc (cond.)
real_naive	N/A	51.47%	31.93%	N/A	N/A
v2 Zero Shot	N/A	53.39%	36.96%	N/A	N/A
v2 2shot	N/A	57.661%	42.014%		
v2_1 Zero Shot	60.47%	48.60%	34.84%	**80.06%**	57.46%
v2_1 2shot	**72.791%**	**58.427%**	**43.708%**	79.837%	**59.950%**

Schema Linking in the Wild

- **Framework structure diagram TODO**



- Verifier 도입을 통한 각 Stage error mitigation 방지
- Feedback-aware 학습을 하기 위한 구조 선정

Schema Linking in the Wild

- **Experimental Results: Schema linking**

	DB Acc	Table Acc	Col Acc	Table Acc (cond.)	Col Acc (cond.)
real_naive	N/A	51.47%	31.93%	N/A	N/A
v2 Zero Shot	N/A	53.39%	36.96%	N/A	N/A
v2 2shot	N/A	57.661%	42.014%		
v2_1 Zero Shot	60.47%	48.60%	34.84%	80.06%	57.46%
v2_1 2shot	**72.791%**	**58.427%**	43.708%	79.837%	59.950%
v3_fewshot	70.46%	58.11%	**44.35%**	82.48%	**62.95%**

Schema Linking in the Wild

- **TODO**
  - Stage 1, Stage 2 특화 학습 실시
    - Stage 1은 Input Context Length를 줄이면서 부족한 Context를 Explore & Expand하는 방법론 고민 필요
    - Stage 2는 SQL Clause-aware signal을 줄 수 있는 loss + verifier feedback을 따르는 loss 설계 필요
  - 학습 데이터셋 구축
  - 비교 Baseline 선정
  - 실험 진행에 따른 분석 실시
  - Etc…

Thank you