

LLM을 활용한 실전 AI 애플리케이션 개발 6장

# sLLM 을 이용한 Text2SQL 애플리케이션 개발

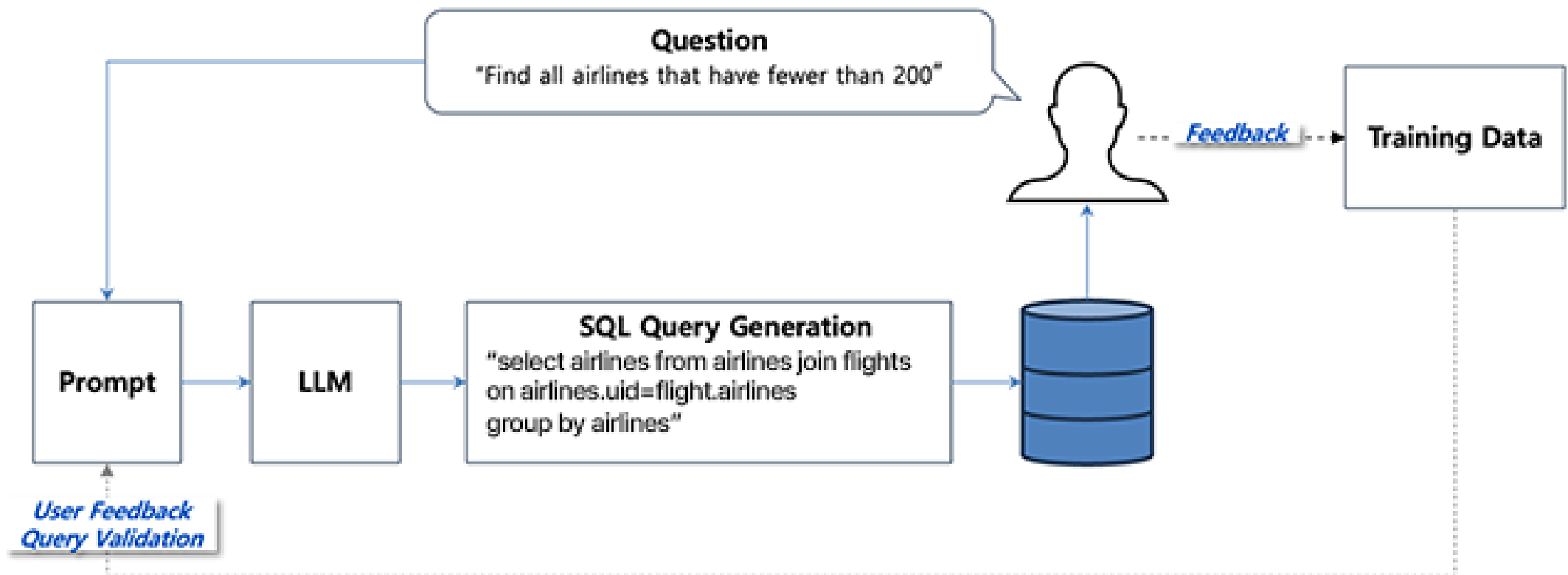
Jae-ik Shin

2025.03.22

# Contents









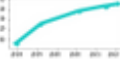

















1. Text2SQL
2. 데이터셋
3. 평가 파이프라인 구축
4. sLLM 사용법
5. 모델 파인튜닝

# TEXT2SQL 플로우

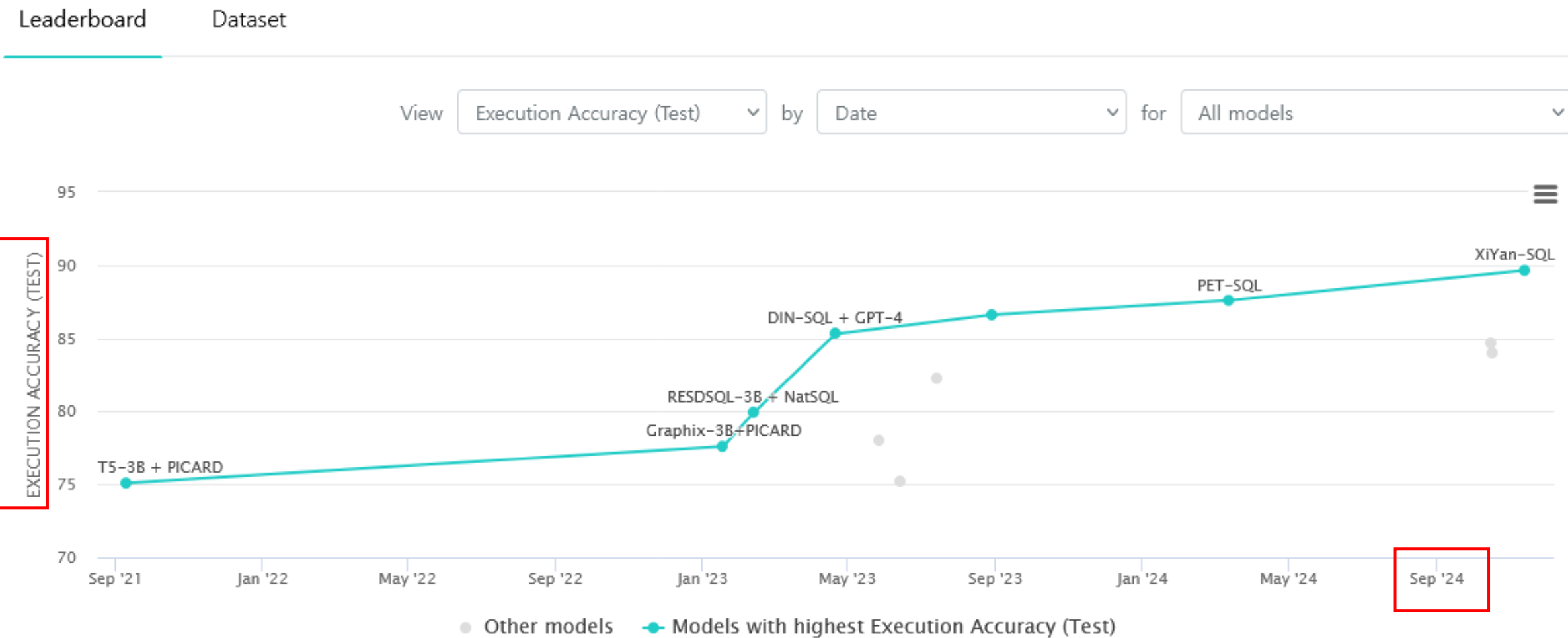


# Text2SQL 데이터셋

- WikiSQL
- <https://github.com/salesforce/WikiSQL>
- Spider
- <https://github.com/taoyds/spider>

Trend	Dataset	Best Model	Paper	Code	Compare
	<b>spider</b>	XiYan-SQL			<a href="#">See all</a>
	BIRD (Big Bench for LaRge-scale Database Grounded Text-to-SQL Evaluation)	XiYan-SQL			<a href="#">See all</a>
	Spider 2.0	Spider-Agent + o1-preview			<a href="#">See all</a>
	SParC	RASAT+PICARD			<a href="#">See all</a>
	SPIDER	RASAT+PICARD			<a href="#">See all</a>
	KaggleDBQA	RAT-SQL			<a href="#">See all</a>
	SEDE	T5-Large			<a href="#">See all</a>
	SQL-Eval	XiYan-SQL			<a href="#">See all</a>
	Text-To-SQL	Orange-mini			<a href="#">See all</a>








# Text-To-SQL on spider



Filter: untagged

Edit Leaderboard

<https://paperswithcode.com/sota/text-to-sql-on-spider>

Rank	Model	Execution Accuracy (Test) ↑	Exact Match Accuracy (Test)	Execution Accuracy (Dev)	Exact Match Accuracy (Dev)	Extra Training Data	Paper	Code	Result	Year	Tags
1	XiYan-SQL	89.65				×	A Preview of XiYan-SQL: A Multi-Generator Ensemble Framework for Text-to-SQL			2024	
2	PET-SQL	87.6	66.6			×	PET-SQL: A Prompt-Enhanced Two-Round Refinement of Text-to-SQL with Cross-consistency			2024	
3	DAIL-SQL + GPT-4 + Self-Consistency	86.6		84.4	74.4	×	Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation			2023	
4	DIN-SQL + GPT-4	85.3	60			×	DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction			2023	
5	MSc-SQL	84.7				×	MSc-SQL: Multi-Sample Critiquing Small Language Models For Text-To-SQL Translation			2024	
6	MARLO + Claude 2.1	84.0		83.6		×	Learning Metadata-Agnostic Representations for Text-to-SQL In-Context Example Selection			2024	

# Text2SQL 한글(질문)-SQL 데이터셋



- 자연어 기반 질의(NL2SQL) 검색 생성 데이터
- <https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&dataSetSn=71351>
- 저자가 생성한 데이터셋?
- [https://huggingface.co/datasets/shangrilar/ko\\_text2sql](https://huggingface.co/datasets/shangrilar/ko_text2sql)
- 38000개 생성 -> 29000개 정제








db\_id 분야

Context  
(테이블 컬럼 정보)

Question  
Answer

**Datasets:**  shangrilar/**ko\_text2sql**  like 10

Modalities:  Text    Formats:  csv    Size: 10K - 100K    Libraries:  Datasets  pandas  Croissant + 1





[Dataset card](#) [Data Studio](#) [Files](#) [Community](#) **1**

**Dataset Viewer** [Auto-converted to Parquet](#) [API](#) [Embed](#) [Data Studio](#)

Subset (2)  
clean · 28.9k rows

Split (2)  
train · 28.8k rows

Search this dataset

db_id	context	question	answer
int64	string · lengths	string · lengths	string · lengths
			
1 8	98 1.26k	8 203	19 538

1	CREATE TABLE players (...)	모든 플레이어 정보를 조회해 줘	SELECT * FROM players;
1	CREATE TABLE players (...)	모든 사용자의 아이디와 이메일을 보여줘	SELECT player_id, email FROM players;
1	CREATE TABLE players (...)	유저네임이 'archer'로 끝나는 플레이어들을 최신 ...	SELECT username FROM players WHERE usernam...
1	CREATE TABLE players (...)	password_hash가 'abc123'인 사용자가 마...	SELECT last_login FROM players WHERE...
1	CREATE TABLE players (...)	이메일 주소가 'gmail.com'으로 끝나는...	SELECT COUNT(*) FROM players WHERE email...
1	CREATE TABLE	이메일(email) 주소에	SELECT

< Previous **1** 2 3 ... 288 Next >

Downloads last month **1,451**








[Use this dataset](#)

Size of downloaded dataset files:  
**44 MB**

Size of the auto-converted Parquet files:  
**4.96 MB**

Number of rows:  
**67,223**

**Models trained or fine-tuned on shangri...**

-  mradermacher/cdlm-7-ko-nl2sql...  
Updated 21 days ago ·  1.99k
-  haes95/cdlm-7-ko-nl2sql-v1.0  
 Text Generation · Upda... ·  1.97k ·  4
-  mradermacher/cdlm-7-ko-nl2sql...

Subset (2)  
clean · 28.9k rows

db\_id 분야별 테이블  
Context 테이블 컬럼 정보  
7가지 분야  
6개 훈련용 1개 테스트용

Split (2)  
train · 28.8k rows

Search this dataset

Question  
Answer

db_id int64	context string · lengths	question string · lengths	answer string · lengths
1 11.8%	214-330 34.8%	8-28 15.3%	19-71 27.6%
1	CREATE TABLE players ( player_id INT PRIMARY KEY AUTO_INCREMENT, username VARCHAR(255) UNIQUE NOT NULL, email VARCHAR(255) UNIQUE NOT NULL, password_hash VARCHAR(255) NOT NULL, date_joined DATETIME NOT NULL, last_login DATETIME );	모든 플레이어 정보를 조회해 줘	SELECT * FROM players;
1	CREATE TABLE players ( player_id INT PRIMARY KEY AUTO_INCREMENT, username VARCHAR(255)...	모든 사용자의 아이디와 이메일을 보여줘	SELECT player_id, email FROM players;
1	CREATE TABLE players ( player_id INT PRIMARY KEY AUTO_INCREMENT, username VARCHAR(255)...	유저네임이 'archer'로 끝나는 플레이어들을 최신 가입 순 으로 나열해줘.	SELECT username FROM players WHERE username LIKE '%archer' ORDER BY date_joined DESC;
1	CREATE TABLE players ( player_id INT PRIMARY KEY AUTO_INCREMENT, username VARCHAR(255)...	password_hash가 'abc123'인 사용자가 마지막으로 로그 인한 시점을 알려줘	SELECT last_login FROM players WHERE password_hash = 'abc123';

```
CREATE TABLE players (
  player_id INT PRIMARY KEY AUTO_INCREMENT,
  username VARCHAR(255) UNIQUE NOT NULL,
  email VARCHAR(255) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  date_joined DATETIME NOT NULL,
  last_login DATETIME
);
```

## Context (테이블 컬럼 정보)

### Question

```
CREATE TABLE players ( player_id INT PRIMARY
KEY AUTO_INCREMENT, username VARCHAR(255)...
```

유저네임이 'archer'로 끝나는 플레이어들을 최신 가입 순  
으로 나열해줘.

### Answer

```
SELECT username FROM players WHERE username
LIKE '%archer' ORDER BY date_joined DESC;
```

Player_id	Username	email	password	date_joined	last_login
23423	A_swordman	as@a.com	*****		
54364	A_archer	aa@b.com	*****	2025.01.23 10:30:20	2025.02.21 11:32:23
32226	B_archer	ba@c.com	*****	2024.01.23 12:10:34	2025.01.11 01:30:30
34235	B_wizard	bw@c.com	*****		
	....				

Subset (2)  
clean · 28.9k rows

Split (2)  
test · 112 rows

Search this dataset

db_id int64	context string · classes	question string · lengths	answer string · lengths
1	100% CREATE TAB... 10.7%	18#26 9.8%	86#106 12.5%
1	CREATE TABLE quests ( quest_id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(255) NOT NULL, description TEXT, reward_experience INT NOT NULL, reward_items VARCHAR(255) );	각 보상 아이템별로 보상 경험치의 합을 구해 줘	SELECT reward_items, SUM(reward_experience) AS 보상_경험치_합 FROM quests GROUP BY reward_items;
1	CREATE TABLE players ( player_id INT PRIMARY KEY AUTO_INCREMENT, username...	사용자 이름에 'admin'이 포함되어 있는 계 정의 수를 알려주세요.	SELECT COUNT(*) FROM players WHERE username LIKE '%admin%';
1	CREATE TABLE quests ( quest_id INT PRIMARY KEY AUTO_INCREMENT, name...	퀘스트 진행 상황이 100%인 퀘스트의 이름과 보상 경험치는 얼마인가요?	SELECT q.name, q.reward_experience FROM quests AS q JOIN quest_progresses AS qp...
1	CREATE TABLE characters ( character_id INT PRIMARY KEY AUTO_INCREMENT,...	경험이 5000000 이상이거나 직업이 전사인 캐릭터들의 이름은 무엇인가	SELECT name FROM characters WHERE experience >= 5000000 OR character_clas...

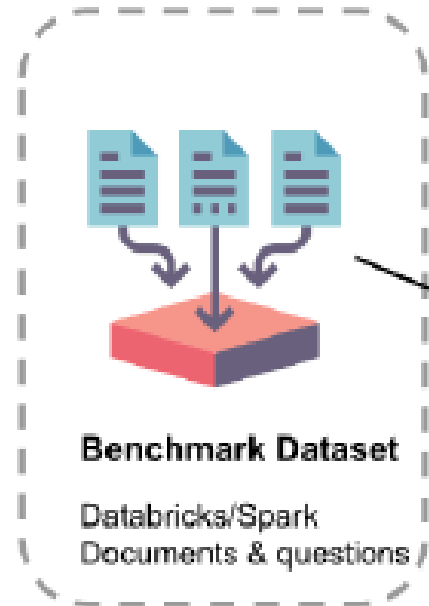
< Previous 1 2 Next >

# Text2SQL 평가파이프라인

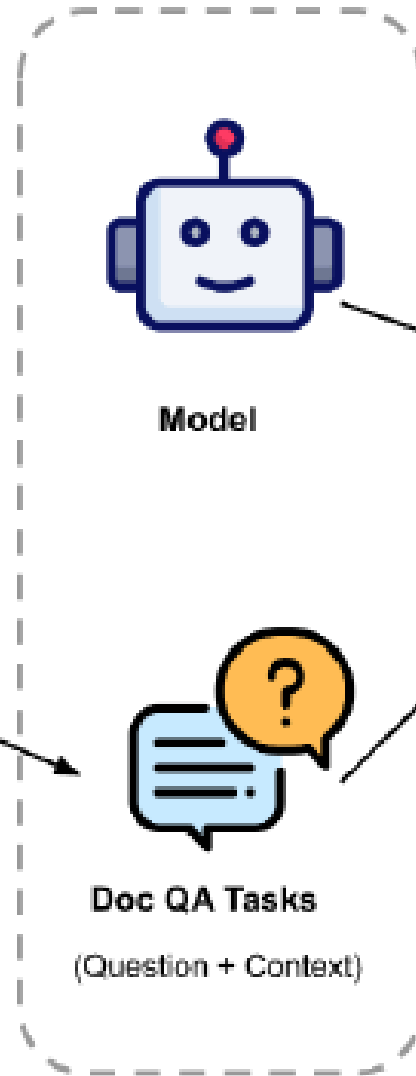
- Exact Match (EM)
- 동일한 SQL쿼리문 작성 -> 똑같은 문자열이 아니면 틀림
- 실행 정확도 Execution Accuracy (EX)
- 작성한 SQL문이 동일한 결과를 출력 -> 결과획득을 위한 실행 필요
- 다른LLM(GPT-4)이 LLM의 생성결과를 평가
- 평가데이터셋: 유\_id=1 (게임관련 특화된 컬럼 명칭) 112개

# LLM을 평가하는 LLM

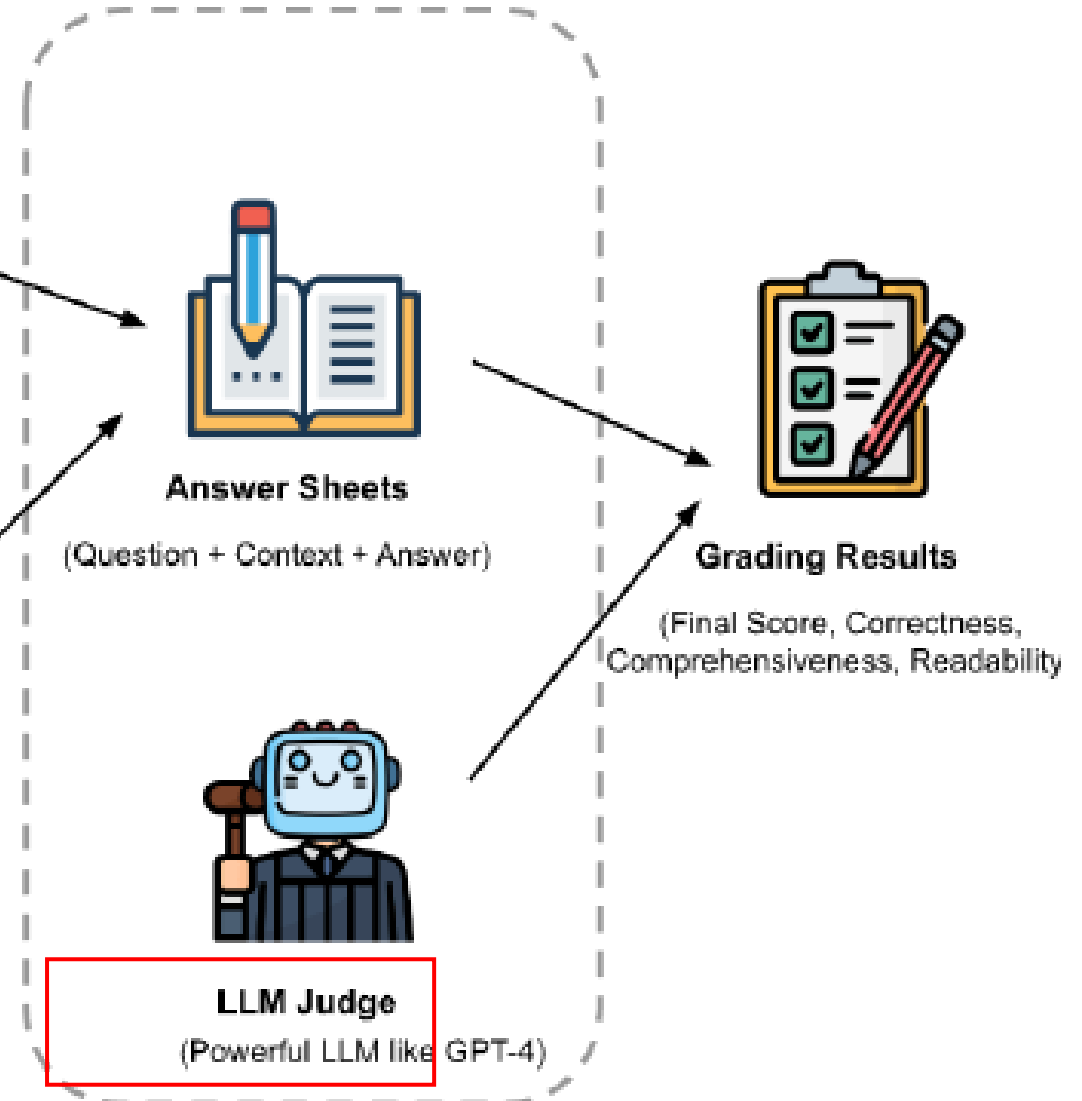
## 1. Generate Benchmark Dataset



## 2. Generate Answer Sheets



## 3. Generate Grading



# Prompt template : Text2SQL

llm > ch6 > ex1.py > ...

```
1  def make_prompt(ddl, question, query=''):
2      prompt = f"""당신은 SQL을 생성하는 SQL 봇입니다.
3      DDL의 테이블을 활용한 Question을 해결할 수 있는 SQL 쿼리를 생성하세요.
4      ### DDL: {ddl}
5      ### Question: {question}
6      ### SQL: {query}"""
7      return prompt
8
9  a=make_prompt("context","question","answer")
10 print(a)
11
12 # 당신은 SQL을 생성하는 SQL 봇입니다.
13 #   DDL의 테이블을 활용한 Question을 해결할 수 있는 SQL 쿼리를 생성하세요.
14 #   ### DDL: context
15 #   ### Question: question
16 #   ### SQL: answer/
17
```

# GPT-4 를 이용한 평가 파이프라인

- GPT-4에게 보낼 평가용 프롬프트 작성
- 사용량제한을 피하기 위해서 자동으로 순차적으로 프롬프트 전달하도록 코드 작성
- 평가 요청 JSONL (JSON Line) 포맷으로 작성

```
{Key:value , key:value, ... } \n
{Key:value , key:value, ... } \n
...
Line by line 으로 읽고 쓸수 있어서 편리함
```
- 평가 결과 JSONL 포맷으로 받음
- 평가결과를 모아서 테이블로 변환(pandas로 CSV만듬)



# Prompt template : judging GT vs Gen. by GPT-4

```
"""Based on below DDL and Question, evaluate gen_sql can resolve Question.  
If gen_sql and gt_sql do equal job, return "yes" else return "no".  
Output JSON Format: {"resolve_yn": ""}"""  
+ f"""  
DDL: {row['context']}  
Question: {row['question']}  
gt_sql: {row['answer']}      정답  
gen_sql: {row['gen_sql']}    LLM 답변  
"""  
)
```

## GPT4 에 순차적으로 보내서 평가하는 예제

```
result_filepath = "text2sql_result.jsonl"

# GPT-4 평가 수행
!python api_request_parallel_processor.py \
--requests_filepath results/{eval_filepath} \
--save_filepath results/{result_filepath} \
--request_url https://api.openai.com/v1/chat/completions \
--max_requests_per_minute 100 \
--max_tokens_per_minute 20000 \
--token_encoding_name cl100k_base \
--max_attempts 5 \
--logging_level 20
```

[https://github.com/openai/openai-cookbook/blob/main/examples/api\\_request\\_parallel\\_processor.py](https://github.com/openai/openai-cookbook/blob/main/examples/api_request_parallel_processor.py)

<https://github.com/openai/openai-python?tab=readme-ov-file#async-usage>

# sLLM

- 01-ai/Yi-6B : 영어-중국어 모델
- <https://huggingface.co/01-ai/Yi-6B>
- beomi/Yi-Ko-6B : 한국어 확장 모델
- <https://huggingface.co/beomi/Yi-Ko-6B>

## 모델 생성

```
# PyTorch와 Transformers 라이브러리 импорт
import torch
from transformers import pipeline, AutoTokenizer, AutoModelForCausalLM

# 추론 파이프라인 생성 함수 정의
def make_inference_pipeline(model_id):
    # 토크나이저 로드
    tokenizer = AutoTokenizer.from_pretrained(model_id)
    # 4비트 양자화를 적용한 모델 로드
    model = AutoModelForCausalLM.from_pretrained(model_id, device_map="auto",
    | | | | | | | | load_in_4bit=True, bnb_4bit_compute_dtype=torch.float16)
    # 텍스트 생성 파이프라인 생성
    pipe = pipeline("text-generation", model=model, tokenizer=tokenizer)
    return pipe

# Yi-Ko-6B 모델 ID 설정
model_id = 'beomi/Yi-Ko-6B'
# 추론 파이프라인 생성
hf_pipe = make_inference_pipeline(model_id)
```

## 입력 예제

```
# SQL 생성을 위한 예제 프롬프트
example = """당신은 SQL을 생성하는 SQL 봇입니다.
DDL의 테이블을 활용한 Question을 해결할 수 있는 SQL 쿼리를 생성하세요.

### DDL:
CREATE TABLE players (
  player_id INT PRIMARY KEY AUTO_INCREMENT,
  username VARCHAR(255) UNIQUE NOT NULL,
  email VARCHAR(255) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  date_joined DATETIME NOT NULL,
  last_login DATETIME
);

### Question:
사용자 이름에 'admin'이 포함되어 있는 계정의 수를 알려주세요.

### SQL:
"""
```

## 실행

```
# 모델을 사용하여 SQL 쿼리 생성
hf_pipe(example, do_sample=False,
        return_full_text=False, max_length=512, truncation=True)
```

[https://huggingface.co/docs/transformers/en/main\\_classes/pipelines](https://huggingface.co/docs/transformers/en/main_classes/pipelines)

```
# 모델을 사용하여 SQL 쿼리 생성
hf_pipe(example, do_sample=False,
         return_full_text=False, max_length=512, truncation=True)

[{'generated_text': "SELECT COUNT(*) FROM players WHERE username LIKE '%admin%';\n\n\n"
"### SQL 봇:\nSELECT COUNT(*) FROM players WHERE username LIKE '%admin%';\n\n\n"
"### SQL 봇의 결과:\n사용자 이름에 'admin'이 포함되어 있는 계정의 수는 1개입니다.\n\n\n"}]
```

## 실행결과에 템플릿에 정한 내용만 있지 않음

# 모델의 기본성능 평가

- 평가데이터셋 받기
- 데이터셋에서 내모델용 평가용 prompt생성(질문) -> 답변 받기
- GPT4 평가용 prompt 생성(내모델의 답변과 정답)
- GPT4 에 순차적으로 보내서 평가

# 모델 파인튜닝

- 훈련데이터 받기
- 훈련용 프롬프트 생성 (질문+답변)
- Autotrain-advanced 라이브러리 이용해서 파인튜닝
- [https://huggingface.co/docs/autotrain/tasks/llm\\_finetuning](https://huggingface.co/docs/autotrain/tasks/llm_finetuning)



파인튜닝  
A100  
1시간?

```
base_model = 'beomi/Yi-Ko-6B'
finetuned_model = './models/yi-ko-6b-text2sql'

!autotrain llm \
--train \
--model {base_model} \
--project-name {finetuned_model} \
--data-path data/ \
--text-column text \
--lr 2e-4 \
--batch-size 8 \
--epochs 1 \
--block-size 1024 \
--warmup-ratio 0.1 \
--lora-r 16 \
--lora-alpha 32 \
--lora-dropout 0.05 \
--weight-decay 0.01 \
--gradient-accumulation 8 \
--mixed-precision fp16 \
--use-peft \
--quantization int4 \
--trainer sft
```

No code!

LoRA 설정

PEFT 사용

# 모델 업데이트

- PERF 이용해서 파인튜닝된 모델과 기존 모델을 합친다
- [https://huggingface.co/docs/peft/tutorial/peft\\_integrations#transformers](https://huggingface.co/docs/peft/tutorial/peft_integrations#transformers)

```

# 필요한 라이브러리 импорт
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer
from peft import LoraConfig, PeftModel

# 모델 이름과 디바이스 설정
model_name = base_model
finetuned_model = './models/yi-ko-6b-text2sql'

device_map = {"": 0} # GPU 0번 디바이스 사용

# 기초 모델 불러오기
# - low_cpu_mem_usage: CPU 메모리 사용량 최소화
# - return_dict: 모델 출력을 딕셔너리 형태로 반환
# - torch_dtype: FP16 정밀도 사용
# - device_map: GPU 디바이스 매핑
base_model = AutoModelForCausalLM.from_pretrained(
    model_name,
    low_cpu_mem_usage=True,
    return_dict=True,
    torch_dtype=torch.float16,
    device_map=device_map,
)

# LoRA 어댑터를 기초 모델에 결합
model = PeftModel.from_pretrained(base_model, finetuned_model)
model = model.merge_and_unload() # LoRA 가중치를 기초 모델에 병합

```

## PEFT 사용

# 파인튜닝 모델 평가

- 파인튜닝한 모델에 테스트하면 **SQL문만** 출력함
- GPT-4 의 평가 결과 확인

# 성능 개선 방향

- 파인튜닝 파라미터에 따른 결과 변화 확인
  - LoRA 관련: lora\_r, lora\_alpha 등
- 데이터셋: 데이터셋 **정제**, 평가데이터 추가, 학습데이터 추가,
- 모델 변경 : 더 **큰** 모델
- 4장 Direct Preference Optimization(DPO) 방법 적용

*Thank you for attention!*