Dossier Stage: text to sql

**Github repository:** https://github.com/XanderVanderLindenIntoData/Stage

**Omschrijving**: In deze 14 weken stage ga ik met behulp van generatieve AI en large language models een proof of concept ontwikkelen voor text 2 sql in functie van relationele databases.

De stage zal zich opdelen in 4 grotere delen:

1. Marktverkenning: Een analyse op de bestaande text to sql oplossingen met focus op open-source
2. Technische Analyse: Identificeren en vergelijken van de belangrijkste text to sql libraries in python
3. Proof of concept: Een proof of concept ontwikkelen die ten minste 1 van de geïdentificeerde oplossingen gebruikt en uitwerkt. Dit wordt getest en geëvalueerd, mijn bedoeling is om er een aantal uitgebreid te testen en te vergelijken.
4. Rapportage en aanbevelingen: Aan de hand van dit rapport zal ik beschrijven hoe het best wordt geïmplementeerd en zal ik aanbevelingen geven voor in de toekomst.

# Waarom huidige oplossing tekortschieten?

Looking at Spider [**leaderboards**](https://yale-lily.github.io/spider), where top models achieve 90%+ accuracy and even solutions from a year ago surpass 80%, one might assume that text-to-SQL is a solved problem. So, if that's the case, why isn’t everyone already using LLM solutions to interact with data?

The reality is that when applied to real-world business use cases, these solutions often fall short. For instance, when we tested a state-of-the-art language model, GPT-4o, using our internal evaluation set, its accuracy plummeted to 51%. This significant gap between benchmark performance and real-world application reveals several limitations of traditional benchmarks. There are four main pillars that define the gap between nice 90% benchmark numbers vs. real-world BI use cases:

(<https://www.snowflake.com/en/engineering-blog/cortex-analyst-text-to-sql-accuracy-bi/>)

4 grote problemen met huidige llm-integraties:

* **1. Question complexity**: The natural language questions used in existing benchmarks are often not very complex, industry-specific or realistic. They also fail to capture the types of questions frequently asked in BI contexts.
* **2.Schema complexity:**The databases in these benchmarks don’t always represent the complex, messy and often unclear schemas that are common in BI tasks. In addition, time-series tables — which are very common in transactional data, like sales or user web activity — are also significantly underrepresented in existing benchmarks.
* **3.SQL complexity**: These benchmarks often lack complex queries, such as those containing window functions, CTEs and complex aggregations, which are crucial for real-world BI tasks.
* **4.Measure SQL in the context of semantics:**Traditional benchmarks fail to account for the importance of aligning SQL queries with specific business definitions and metrics. For instance, how different organizations define key metrics like "Daily Active Users" can vary significantly, impacting the accuracy of SQL queries in real-world applications.

## Approaches voor text2sql met llm’s :

Prompt-engineering: gebruikt grote closed source’s llm’s zoals bijvoorbeeld GPT-4 of Gemini met enorm veel parameters.

Supervised-fine-tuning: modellen met kleiner aantal parameters zoals NL2SQL => controleerbaardere sql queries

Beste accuraatheid met combinatie van de 2

**Voorbeeld framework met 3 agents:**

Afbeelding met tekst, schermopname, Lettertype, lijn

Door AI gegenereerde inhoud is mogelijk onjuist.

## Evolutie van sql evaluatie + Hoe evalueren

Afbeelding met tekst, schermopname, diagram, software

Door AI gegenereerde inhoud is mogelijk onjuist.

**Meest gebruikte datasets:**

Spider met aanpassingen(kolomnamen weg en : <https://arxiv.org/abs/2010.12773>

## M-schema uitleg(gebaseerd op paper van Xiyan-SQL):

## Stappenplan (niet definitief)

Moet nog aantal papers per ondertitels bekijken voor de efficiëntste te vinden

### Pre-processing

**Schema-linking:** Benodigde tabellen en kolommen verbinden met de natural language query => zorgt voor accurate mapping en processing van de ids. Dit deeltje van preprocessing is zeer belangrijk geworden bij de integratie van llm’s voor txt to sql

**DB content retrieval:** Hoe je **de werkelijke data ophaalt uit de tabellen is velangrijk voor de performantie van het NL2SQL mode, daarom is het belangrijk een goede strategie te kiezen hiervoor.**

**Additional information acquisition: De algemene uitleg is vrij vaag ik leg het verder uit aan de hand van voorbeeldmethoden.**

### NL2SQL translatie-methoden

**Encoding Stategy**

1 Sequentiele encoding

2 Graaf gebaseerde encoding

3 Separate Encoding

**Decoding Strategy**

1 Gulzig zoek gebaseerd

2 Beam Search

3 Constraint aware incremental decoding

**Task-specific Prompt Strategy**

Intermediate Representation for NL2SQL Translation

### Post-processing

Guiding models to identify and find syntax errors in sql queries ( refining)

# Python libraries

Litgpt - voor volledige implementatie modellen

Hugging face -

## Enkele goede applicaties

Chat2DB ( kan technische en niet-technische response generen)

https://chat2db.ai/

## Welke LLM gebruik ik best?

GPT-4.0

## Welke benchmarks gebruik ik?

Meeste werken met:

## 

-BIRD(x):

-SPIDER:

-SQL-Eval:

~~-NL2GQL(niet relationele databanken bij ons niet van toepassingen):~~