StatBot.Swiss: Bilingual Open Data Exploration in Natural Language

Farhad Nooralahzadeh[1†], **Yi Zhang**[1†], **Ellery Smith**[1], **Sabine Maennel**[2]
Cyril Matthey-Doret[2], **Raphaël de Fondville**[3], **Kurt Stockinger**[1]
[1]Zurich University of Applied Sciences, Switzerland
[2]Swiss Data Science Center, Switzerland
[3]Federal Statistical Office, Switzerland
{farhad.nooralahzadeh, yi.zhang, kurt.stockinger}@zhaw.ch

Abstract

The potential for improvements brought by Large Language Models (LLMs) in Text-to-SQL systems is mostly assessed on monolingual English datasets. However, LLMs' performance for other languages remains vastly unexplored. In this work, we release the StatBot.Swiss dataset, the *first bilingual benchmark for evaluating Text-to-SQL systems* based on real-world applications. The StatBot.Swiss dataset contains 455 natural language/SQL-pairs over 35 big databases with varying level of complexity for both English and German. We evaluate the performance of state-of-the-art LLMs such as GPT-3.5-Turbo and mixtral-8x7b-instruct for the Text-to-SQL translation task using an in-context learning approach. Our experimental analysis illustrates that current LLMs struggle to generalize well in generating SQL queries on our novel bilingual dataset [1].

1 Introduction

Switzerland is a multilingual country, officially recognizing four national languages: German, French, Italian, and Romansh. Multilingualism is a pillar of the country's identity, ensuring that all citizens have equal access to public service, education, and any other information, regardless of their linguistic background. Switzerland started offering open government data, and the supply continues to grow, spurred on, among other things, by open-by-default regulations of the federal government, and initiatives from cantons and municipalities.

While the *opendata.swiss* initiative implements the once-only principle by offering a central catalog for all available Swiss open government data, these datasets are often not standardized across administration levels and neither are methods of collection, compilation, and processing. If data are found, one must then work out and understand the methodological differences in order to know which data are more suitable for the intended usage and more importantly be capable of importing and analyzing the data through statistical software such as a spreadsheet, Python, R or SAS, which all require advanced computing skills.

These challenges pose a risk for the democratic processes: the harder it is for a citizen to access an accurate source of information such as national statistics, the more likely a country is to suffer from misinformation. The StatBot.Swiss project aims to develop a Swiss statistical bot that simplifies access to open government data by allowing interaction with the data directly via natural language. More precisely, one enters a question and gets an answer built upon the result of a query on trusted datasets from the opendata.swiss platform.

The *main objectives* of this paper are (1) to provide a *real-world bilingual dataset to benchmark Text-to-SQL systems*, namely the StatBot.Swiss dataset, which consists of 455 intricate instances of querying data from 35 large databases, totaling 7.5 GB in size, available in both English and German, and (2) to *assess the performance of current state-of-the-art pre-trained LLMs* with various prompting strategies and so to establish a strong baseline on the StatBot.Swiss dataset.

To the best of our knowledge, the StatBot.Swiss dataset is the *first Text-to-SQL benchmark to incorporate English and German languages* within the context of a complex Text-to-SQL benchmark and real-world databases. We conduct comprehensive evaluations on two pre-trained LLMs, namely GPT-3.5-Turbo-16k, (Brown et al., 2020) and Mixtral-8x7B-Instruct-v0.1 (Jiang et al., 2024). Our experimental results show that current models using in-context learning strategies achieve up to 50.58% execution accuracy and struggle to generate SQL queries giving exact matches on StatBot.Swiss, which shows that multilingual Text-to-SQL systems using state-of-the-art LLMs still lack robust-

[†]Equal contribution.
[1]We release our data and code to the community at https://github.com/dscc-admin-ch/statbot.swiss

ness for reliable applications.

2 Related Work

Text-to-SQL Dataset The development of Text-to-SQL datasets and corresponding benchmarks, such as the Spider (Yu et al., 2018) and WikiSQL (Zhong et al., 2017) datasets and other domains (Deriu et al., 2020), has been instrumental in advancing natural language interfaces to databases. Although much of the initial work focused on the English language, some progress has been achieved in creating resources for other languages. DuSQL (Wang et al., 2020b) brings Text-to-SQL interpretation to Mandarin. PAUQ (Bakshandaeva et al., 2022) fills the gap in the landscape of the Russian language by introducing the complexities associated with Slavic languages to the Spider dataset. ViText2SQL (Tuan Nguyen et al., 2020) is the first public large-scale Text-to-SQL semantic parsing dataset for Vietnamese.

MultiSpider (Dou et al., 2023) generalizes the Spider benchmark to multiple languages. However, the drawbacks of MultiSpider are also obvious. As a translated work from the Spider dataset, the complexity of the database schema and the datasets are strongly limited, and the translation lacks native language expertise.

Unlike MultiSpider, the StatBot.Swiss benchmark is realistic, complex, and curated by native speakers with proper domain knowledge, while covering both English and German for datasets with trusted sources.

LLMs in Text-to-SQL Translation Recently, there has been significant development in employing Large Language Models (LLMs) for the Text-to-SQL task. Several approaches have been suggested to improve the abilities of LLMs by in-context learning techniques (Rajkumar et al., 2022; Liu et al., 2023; Nan et al., 2023) or via semantic hypothesis re-ranking (Von Däniken et al., 2022). Additionally, methods such as intermediate reasoning steps and self-correction mechanisms have been integrated to enhance the performance of LLMs in Text-to-SQL applications (Chen et al., 2024; Pourreza and Rafiei, 2023). Following these pioneering works, we combine in-context learning to LLMs for the Text-to-SQL task and conduct a comprehensive evaluation of prompt representations over the StatBot.Swiss benchmark.

3 System Overview

3.1 Database preparation

OPENDATA.SWISS[2] is the Swiss public administration's central portal for open government data. As of the date of the submission, the catalog lists more than 10,300 datasets, covering 14 categories, e.g., *health*, *environment*, and *economy*. We meticulously selected 22 German and 13 English datasets from opendata.swiss and subsequently generated corresponding PostgreSQL databases through an automated pipeline.

3.2 Data Preparation

Each database contains a *fact table* that describes the knowledge domain of the data, and a *dimension table* for the corresponding spatial information (see Database Schema Example in Appendix A). Note that some statistical information is collected at the municipality level, while other information is collected at the cantonal level (which corresponds to a state). Moreover, the content of the fact tables does not overlap. Finally, one spatial dimensional table is connected to each of the 35 fact tables via a single foreign key constraint.

Dataset preparation For each dataset, our experts analyzed the data sources and formulated natural language questions that the dataset could answer. We then crafted SQL queries to address these questions, executed them, and compared the results with the original dataset. Since every table in our database has only a single foreign key - referencing the spatial dimension - the queries that we prepared also target only a single table for each dataset with a join on the spatial unit table.

Dataset statistics After manual curation and validation, the dataset includes 455 natural language question (NL)/SQL-pairs covering a total of 35 databases. Figure 1(a) presents the distribution of the datasets by languages, spanning a wide range of domains covered by opendata.swiss. There is, however, an imbalance in the number of NL/SQL-pairs between datasets: for instance, there are 23 NL/SQL-pairs for the database `marriage_citizenship`, while only 5 were generated for the database `greenhouse_gas_emissions_through_consumption`. Figure 1 (b) gives an overview of the sample distribution for the train set and the development set.

SQL statistics In order to evaluate the complexity

[2]https://opendata.swiss

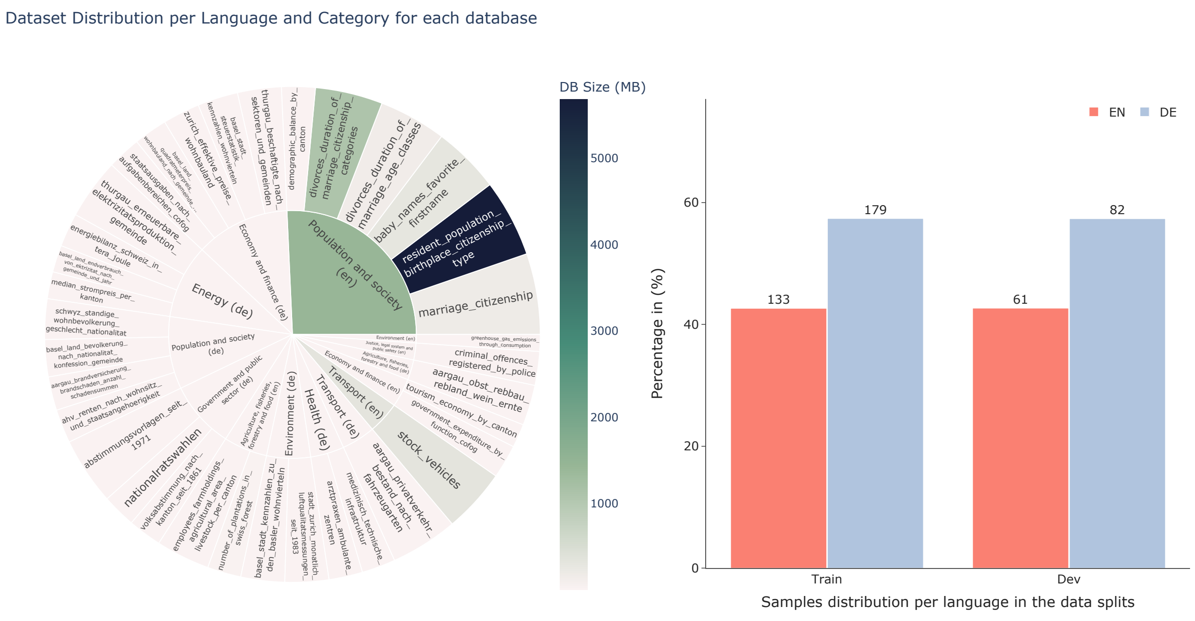


Figure 1: **Dataset distribution:** (a) Left: Knowledge domains, (b) Right: Distribution of natural language/SQL-pairs over the train and development sets. EN = English, DE = German. The numbers on top of the bars denote the number of Text-to-SQL pairs.

Dataset	#Examples (#DBs)	#Tables (#Rows)/DB	Language	Function	Granularity	Knowledge	WITH-Queries
WikiSQL(Zhong et al., 2017)	80,654 (26,521)	1 (17)	EN	✗	✗	✗	✗
SPIDER (Yu et al., 2018)	10,181 (200)	5.1 (2K)	EN	✗	✗	✗	✗
KaggleDBQA (Lee et al., 2021)	272 (8)	2.3 (280K)	EN	✗	✗	✓	✗
ScienceBenchmark (Zhang et al., 2024)	5,332 (3)	16.7 (51M)	EN	✗	✗	✓	✗
BIRD (Li et al., 2023)	12,751 (95)	7.3 (549K)	EN	✓	✗	✓	✗
StatBot.Swiss	455 (35)	2 (1.4M)	EN, DE	✓	✓	✓	✓

Table 1: Comparison between StatBot.Swiss and existing state-of-the-art Text-to-SQL datasets. `Function` refers to SQL built-in functions. `Knowledge` stands for the necessity of external knowledge reasoning from the model. `Granularity` refers to the degree of specificity or the level of details. `WITH-Queries` refer to complex sub queries that are broken up into smaller ones. EN = English, DE = German.

of the queries, we apply the Spider hardness metric (Yu et al., 2018). However, the StatBot.Swiss dataset includes additional PostgreSQL grammar and syntax features, which are not supported by the Spider hardness evaluator: our dataset includes queries with unencountered levels of complexity in the Spider dataset. For more details see Appendix B.

Thus, we *extend the Spider-hardness* to categorize the NL/SQL-pairs into five classes, namely the four classes *easy*, *medium*, *hard*, and *extra*, inherited from Spider-hardness with the addition of a fifth category *unknown*. In Table 1, we compare our StatBot.Swiss dataset with other state-of-the-art benchmarks for the Text-to-SQL task using several metrics. Although our new dataset appears simple regarding the number of databases and tables, the training and development datasets (see Section 4.1 for details) are highly complex and cover more realistic natural language questions with external knowledge and more complex SQL syntax than state-of-the-art benchmarks.

`Language`: The StatBot.Swiss dataset includes tables in two languages, that is, each table is in English or German. All other datasets are only available in English.

`Function`: Only BIRD and our dataset contain built-in functions, e.g., `CAST()` for type casting or `ROUND()` for rounding a number.

`Granularity`: It refers to the degree of specificity or the extent to which data are segmented or detailed (Rudra and Nimmagadda, 2005; Huang et al., 2023). Data may be depicted at varying levels of granularity, spanning from fine-grained (indicating a high degree of detail, e.g., price of a single item) to coarse-grained (denoting a low level of detail, e.g., subtotal or total price of all items). The choice of granularity depends on the objectives of the analysis and the information required from the data. As an illustration, the table `criminal_offences_registered_by_police` contains records at both the coarse and detailed levels querying data at the highest level of detail with `WHERE offence_criminal_code='Offence-total'` and also including detailed values for each `offence_criminal_code`.

`Knowledge`: It is defined as the external knowledge reasoning by Li et al., 2023. The reasoning types comprise *Domain knowledge*, *Numeric Computing*, *Synonym* and *Value Illustration*. Li et al., 2023 also suggests that a comprehensive understanding of the database content is imperative for addressing more complex real-world queries in Text-to-SQL applications.

`WITH-Queries`: Finally, the StatBot.Swiss records cover very complex query syntax, e.g. `WITH`-queries allowing users to break down complex queries into several smaller subqueries.

3.3 Text-to-SQL Translation

The Text-to-SQL translation task aims to match a natural language question with an SQL query that can effectively extract relevant information from a database. Its formal definition is (Wang et al., 2020a): Given a natural language question encoded as a sequence of natural language tokens $\mathcal{Q} = \{q_1, \dots, q_{|\mathcal{Q}|}\}$ and a relational database schema $\mathcal{S} = \langle \mathcal{T}, \mathcal{C} \rangle$ where $\mathcal{T} = \{t_1, \dots, t_{|\mathcal{T}|}\}$ is a series of tables, and $\mathcal{C} = \{c_1, \dots, c_{|\mathcal{C}|}\}$ denotes their corresponding columns, a Text-to-SQL system is a function $f(\mathcal{Q}, \mathcal{S})$ that outputs a correct SQL query $\mathcal{Y} = \{y_1, \dots, y_{|\mathcal{Y}|}\}$ as a sequence of tokens.

To develop a Text-to-SQL system, a prevalent approach is to collect labeled data and train a model via supervised learning (Scholak et al., 2021; Brunner and Stockinger, 2021). Although effective, this approach requires a considerable amount of training data, which is time- and resource-consuming: annotating SQL queries requires SQL-specific expertise.

As an alternative to supervised learning, *in-context learning* (ICL) (Brown et al., 2020), an emergent method of large language models (LLMs), alleviates the need for large training datasets: with only a few examples, ICL enables LLMs to demonstrate performance comparable to, if not better, than fully supervised models for many NLP downstream tasks. When applied to the task of Text-to-SQL, ICL achieves encouraging results (Liu et al., 2023; Nan et al., 2023).

Following this line of research, we formulate a Text-to-SQL task as $f(\mathcal{Q}, \mathcal{S}, \mathcal{E}, \mathcal{P})$, where f is a LLM, and $\mathcal{E}$ is a set of m in-context exemplars as $(\mathcal{Q}_i, \mathcal{Y}i)_{i<m}, \mathcal{Q}_i \neq \mathcal{Q}\}$. $\mathcal{P}$ is a textual template to represent the overall input, i.e. $\mathcal{Q}, \mathcal{S}, \mathcal{E}$, as a prompt and is fed into the LLM. We now describe the structure of our prompt design for our Text-to-SQL system.

Database information Providing prior knowledge, i.e. examples, about an underlying task can aid the generation process of LLMs. Particularly, in the Text-to-SQL task, inclusion of a database's metadata such as table relationships and variable encoding is crucial for enabling effective prompting (Chang and Fosler-Lussier, 2023; Rajkumar et al., 2022).

We opt for a *textual representation of the database information*, as suggested by Rajkumar et al. (2022). In particular, we rely on a CREATE-statement for the initial table creation; see the example prompt in Appendix D for more details. This representation encompasses specific data type information for each column and integrates all foreign key constraint details within the database. Furthermore, database content is partially included. Specifically, the strategy appends a number $r > 0$ of example rows from each table; see Appendix C for prompts with $r = 5$.

Furthermore, in line with the findings of Huang et al. (2023), and Nan et al. (2023) who demonstrated the efficacy of schema augmentation through metadata via in-context learning for the Text-to-SQL task, we *enhance the representation of the database structure*. This augmentation involves integrating metadata column information, such as column name and column description, as illustrated in Appendix D.

Selection of exemplars $\mathcal{E}$ In-context learning enables LLMs to improve their performance on the Text-to-SQL tasks with a small number of training data, with or without a small number of example pairs containing natural language questions and their corresponding SQL representations. We consider two widely used settings for in-context learning in Text-to-SQL: (i) **Zero-shot**: In this context, the evaluation focuses on the Text-to-SQL capability of pretrained LLMs, where the goal is to infer the relationship between a given question and SQL directly from unknown LLM's training dataset. This approach does not leverage any kind of task specific examples; the input prompt is limited to a natural language question along with its corresponding database metadata. The zero-shot setting is used to directly assess the Text-to-SQL capability of pretrained LLMs (Rajkumar et al., 2022; Chang and Fosler-Lussier, 2023). (ii) **Few-shot** In this scenario, the LLMs' prompts include examples from a benchmark such as Statbot.swiss. We include a small number of pairs consisting of natural language questions and their corresponding SQL examples, which are inserted between the representation of the database and the target question. The aim is to assess the Text-to-SQL performance of LLMs using a limited number of training data. When employing few-shot learning, a crucial aspect to consider is the careful selection of a subset of demonstrations from a pool of annotated examples for each test instance. This particular design choice plays a significant role and can influence the overall performance of ICL (Liu et al., 2022; Nan et al., 2023).

In the few-shot scenario, we established various subset selection methods, including *random* and *similarity-based* strategies. To implement the latter, we initially transformed both the in-domain training questions, i.e., those from the same database as the target question, and the test question into vectors using a sentence encoder. Through cosine similarity we then identified and selected the top m training examples that exhibited the highest similarity to the target question as exemplars. Appendix D displays a prompt example comprising two (i.e. $m = 2$) chosen training instances.

4 Experiments and Results

4.1 Experimental Settings

Dataset We randomly split the StatBot.Swiss dataset into training and development datasets. Specifically, the dataset is divided into a development dataset constituting 30% of the total data and a training dataset comprising the remaining 70%. The splitting is stratified to ensure that the proportion of records with varying degrees of complexity, as introduced in Section 3.2, is maintained equal for all combinations of partition elements and databases.

Table 2 in Appendix C lists the hardness distribution across the multilingual databases for each language in the train, dev, and each individual Statbot.swiss datasets.

Large Language Models We examine different strategies for in-context learning using two large language models. To accommodate for extended context lengths in some of our scenarios, i.e., more than 4096 tokens, we restrict our experiment to the GPT-3.5-Turbo-16k model[3] and the Mixtral-8x7B Instruct model[4]. To reduce the computational and memory costs of running inferences using the Mixtral model, we apply QLoRA (Dettmers et al., 2023) with 4-bit quantization and run the inferencing on two *nVidia A100-40GB* GPUs. The LLMs' API temperature is configured to 0, indicating the use of a greedy decoding strategy.

Evaluation Metrics We employ the frequently used evaluation metric *execution accuracy* (EA), which computes the percentage of the system's correctly generated SQL statements (predictions), whose execution results correspond to the results of the gold standard SQLs, i.e. the exact output of the query in the benchmark. However, Floratou et al., 2024 suggest that the original EA may underestimate the overall accuracy due the ambiguity of the queries. Therefore, the evaluation metric should ensure that the generated queries not only produce exact matches, but also meet the underlying purpose of the user's natural language query.

From a chatbot perspective, the nature of user questions can often encompass a range of acceptable queries. To illustrate this issue with a simple example, let us analyze the question "What canton got the most money from tourism in 2016?" which targets the table `tourism_economy_by_canton`. This question is by nature uncertain, and a user could be equally content with responses that include either [*canton_name*] only or [*canton_name, mio_chf_gross_value_added_of_tourism*], which also includes the corresponding funds value. The ground truth from Statbot.swiss for this question outputs only [*canton_name*]. With EA, a query output with [*canton_name, mio_chf_gross_value_added_of_tourism*] will be considered a *false* answer.

This complexity makes conventional metrics based on string-matching or execution-matching inadequate. We start by renaming EA as *strict execution accuracy* ($\text{EA}_{\text{strict}}$) and then introduce *soft execution accuracy* (EA_{soft}) and *partial execution accuracy* ($\text{EA}_{\text{partial}}$) for more effective outcome evaluation of certain ambiguous or uncertain

[3]Using the API at https://openai.com/api

[4]https://huggingface.co/mistralai. The model supports up to a 32k context window.

questions:

Given N NL/SQL-pairs, we have $\text{EA}_k = N^{-1}\sum_{n=1}^{N} I_k(r_n, \hat{r}_n)$ where r_n, respectively $\hat{r}_n$, are the result set of the ground truth, respectively the system's prediction, $k \in \{\text{strict}, \text{soft}, \text{partial}\}$, and I_k is an indicator function defined as

$$I_{\text{strict}}(r_n, \hat{r}_n) = \begin{cases} 1, & \text{if } r_n = \hat{r}_n \\ 0, & \text{otherwise} \end{cases}$$

$$I_{\text{soft}}(r_n, \hat{r}_n) = \begin{cases} 1, & \text{if } r_n \subseteq \hat{r}_n \\ 0, & \text{otherwise} \end{cases}$$

$$I_{\text{partial}}(r_n, \hat{r}_n) = \begin{cases} 1, & \text{if } r_n \subseteq \hat{r}_n \text{ or } \hat{r}_n \subseteq r_n \\ 0, & \text{otherwise} \end{cases}$$

While $\text{EA}_{\text{strict}}$ might underestimate the fraction of correct answers, its partial and soft versions tend to overestimate the overall system's performance: for the final user, the true performance is likely to lie in-between these two types of metrics.

In-context Learning (ICL) Strategies We assess the performance of the following ICL strategies for the Text-to-SQL task on the StatBot.Swiss dataset in both zero-shot and few-shot scenarios.

Zero-shot (Baseline): We use the standard prompt for the Text-to-SQL task along with the database information described in Section 3.3. The database information is stored as a text representation in the input prompt without any demonstration examples.

Few-shot: Building upon the zero-shot setting, we select $m \in \{1, 2, 3, 4, 5, 6, 8\}$ NL/SQL-pairs from the Statbot.swiss training set and insert them between the representation of the database and the target question. We pick these instances employing the following approaches: (i) **Random Selection**: Randomly selecting demonstration examples from the training data, where both the NL-questions and the SQL-queries relate to the same dataset, and (ii) **Similarity-based Selection**: Demonstrative examples are chosen based on their cosine similarity scores as described in Section 3.3 within the training set, ensuring that both the examples and the target question are collected from the same database. Since our dataset comprises both English and German NL/SQL-pairs, we employ the multilingual sentence transformers model (Reimers and Gurevych, 2020), specifically the `distiluse-base-multilingual-cased-v2` from HuggingFace model hub [5], to encode natural language

[5]https://huggingface.co/sentence-transformers

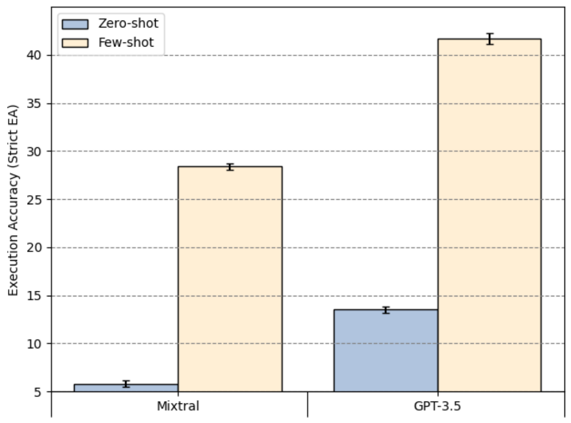


Figure 2: Mean strict execution accuracy : zero-shot and few-shot for GPT-3.5 ($m = 5$) and Mixtral ($m = 6$) models using similarity-based selection where the number of examples are chosen to maximize $\text{EA}_{\text{strict}}$.

questions.

Instruction Tuning: We further fine-tune the Mixtral model using the training dataset and the standard instructions for the Text-to-SQL task, incorporating the database information outlined in Section 3.3. Then, we assess the instruction-tuned models in a manner similar to the ICL strategies, including both zero-shot and few-shot evaluations. The model is trained for 200 epochs under Low-Rank Adaptation (LoRA) and 4-bit quantization with a learning rate of $5e^{-5}$. The batch size per device is 1, with gradient accumulation steps set to 16 and epochs set to 100. The fine-tuning is performed on a single NVIDIA A100 40GB GPU for approximately 20 hours.

4.2 Results

In this section, we present a comprehensive analysis of various prompting strategies and their effectiveness across the StatBot.Swiss datasets. To make the right conclusions when comparing different scenarios, we calculate the mean and standard deviation (std) of the evaluation metrics using 3 independent LLMs' queries over the development set with identical prompting strategies. Figures 2 and 3 give a summary of the performance results. We report the detailed estimated evaluation metrics for all models with various demonstration selection strategies and both languages in Tables 4, and 5 of Appendix F.

It is worth noting that the instruction-tuned Mixtral model significantly underperformed compared to the ICL strategies evaluation of the original Mixtral model. Consequently, we do not provide fur-

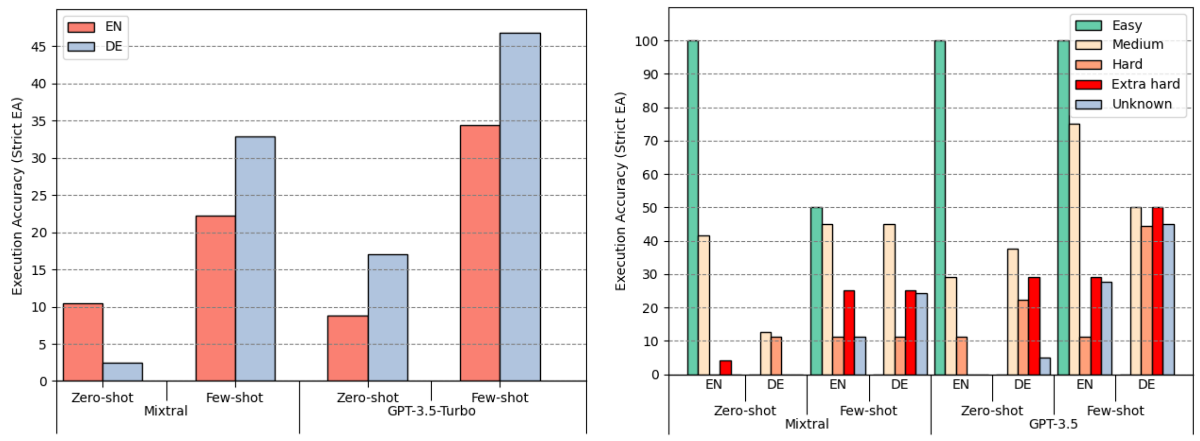


Figure 3: (Left) Strict execution accuracy (EA_{strict}) for each language. (Right) EA_{strict} for each language per query hardness level. All metrics are computed on the development set for zero-shot and few-shot prompting strategies (6-shot in Mixtral, 5-shot in GPT-3.5).

ther details on the results of this experiment.

Model Performance: Looking at the *model performance across various shot levels and selection methods*, GPT-3.5 consistently demonstrates superior performance compared to Mixtral, achieving higher mean accuracy scores. Specifically, in terms of EA_{strict}, GPT-3.5 achieves a superior result of at most 41.68% using 5 examples, whereas Mixtral is left behind with at most 28.39% using 6 shots. Taking into account the concept of correctness rather than exact matching of the generated SQL query, it can be stated that in the optimal few-shot scenario, GPT-3.5 achieves an $EA_{partial}$ of 50.07%.

Zero-shot vs. Few-shot Learning: In *zero-shot learning*, where no additional labeled examples are provided, both models perform relatively poorly. GPT-3.5 achieves an EA_{strict} of 13.52%, outperforming the result by Mixtral at 5.82%. *Few-shot learning* significantly improves model performance, demonstrating the effectiveness of providing a small number of labeled examples. Upon providing a small number of labeled examples (one-shot setting), both models show considerable improvement. GPT-3.5 achieves an EA_{strict} of 33.57%, resulting in an improvement of approximately +20.05% points compared to the zero-shot setting. Mixtral achieves an EA_{strict} of 16.92%, indicating an improvement of approximately +11% compared to the zero-shot setting.

Effect of Selection Method: The *Similarity selection* method generally outperforms the *Random* selection method, particularly in few-shot learning scenarios. Selecting examples based on their similarity to the natural language questions provides more relevant training examples in ICL for LLMs, leading to better model performance.

Impact of Exemplars Number on Model Performance: Transitioning from zero-shot to few-shot learning leads to a significant improvement in model performance across both GPT-3.5 and Mixtral models. On average, each additional shot results in approximately a 5-10% increase in mean EA_{strict}, underscoring the importance of providing labeled examples during the inference. However, this improvement is followed by a significant decrease once the number of examples reaches a certain threshold: maximal performance is achieved with 5 examples in GPT-3.5, and 6 examples in Mixtral using similarity-based selection.

Language and Hardness Level: When differentiating by language, with zero-shot learning, GPT-3.5 achieves an EA_{strict} of 8.75% for English (EN), whereas Mixtral shows a slightly higher accuracy at 10.39%. For German (DE), GPT-3.5 records an EA_{strict} of 17.07%, compared to Mixtral's 2.44%.

Transitioning to the few-shot setting, GPT-3.5 demonstrates significant superiority over Mixtral for English (EN), achieving an EA_{strict} of 34.43%, as opposed to Mixtral's 22.29%. Similarly, in German (DE), GPT-3.5 exhibits better performance with an EA_{strict} of 46.83%, contrasting Mixtral's 32.93%.

Moreover, we analyze the Type-Token Ratios (TTR) and the token length of both English and German questions respectively, revealing that the German dataset exhibits greater linguistic diversity and consequently appears more challenging. This finding is inspired by prior research on the difficulty and discrimination of natural language questions by (Byrd and Srivastava, 2022), and on cross-lingual summarization by (Wang et al., 2023), both of which assess lexical richness and diversity. TTR, defined as the ratio of unique tokens to the total number of tokens, serves as a measure of linguistic diversity. The average TTR for German questions is 0.961, compared to 0.927 for English questions. Furthermore, the average token length for German questions is 15.6, whereas for English questions, it is 15.13. These indices unveil that German questions in the dataset are more diverse and, therefore, potentially more difficult.

The results in Figure 3 also show that both models exhibit higher performance in German than in English, despite the German dataset being inherently more difficult. This trend persists even when the difficulty level of SQL within the development dataset is taken into account. This can be attributed to the fact that the German questions were written by native speakers, while the English questions were not, and thus suggesting potential avenues for future research in paraphrasing and linguistic characteristics of both datasets.

Furthermore, Figure 3 (right) and Table 5 show that GPT-3.5 is unable to answer English questions classified as extra hard or unknown in the zero-shot setting. Conversely, Mixtral succeeds in providing answers for the English questions, even when their difficulty is categorized as medium or extra hard. However, Mixtral also struggles to answer questions in the hard level category, indicating limitations in handling challenging queries despite some capability in the zero-shot scenario.

It is apparent that in the Mixtral setting with an easy difficulty level, there is a notable decline in performance when transitioning from zero-shot to few-shot scenarios. This decline stems from the scarcity of data points within our dataset under easy difficulty level. Specifically, there are only two instances present in the development set, both in English. Consequently, even a single failed prediction significantly impacts the local distribution of percentage scores, causing the performance on easy samples to drop from 100% to 50%. Upon examining the failed prediction, we identified the root cause as a key term swap between COUNT and DISTINCT during the few-shot ICL execution on Mixtral as illustrated in Appendix E.

On average, Mixtral demonstrates better performance than GPT-3.5 in the zero-shot scenario for English questions. Nevertheless, the situation is reversed in the German dataset.

In the few-shot scenario, considering the optimal prompting strategies, i.e., 5-shot with GPT-3.5 and 6-shot with Mixtral, both models exhibit the capability to provide exact matches in every category. Nevertheless, it is important to note that although both models demonstrate some proficiency in answering questions within the few-shot setup, they encounter challenges on many occasions, especially when dealing with queries of higher difficulty levels such as hard and extra hard.

5 Error Analysis

To delve deeper into the difficulties faced by state-of-the-art LLMs in the Text-to-SQL task using the StatBot.Swiss dataset, we have a closer look at the results of the best performing model, namely GPT-3.5, to perform an error analysis. We observe that (i) the model struggles significantly with inferring the queries involving `GROUP BY` multi-columns and `mixed numeric operators`. This leads to a low accuracy of 27.27% and 38.89% for these queries, respectively. These difficulties highlight the need for domain knowledge, which might not always be explicitly represented in the database schema, (ii) queries requiring proper built-in `Functions` or other general SQL keywords are answered slightly better by the LLM, which counts for 40% and 44.44% accuracy, respectively, (iii) the language model results in an accuracy of 50% for `NULL`-values and 66.67% for nested `SELECT`-alias, suggesting that the model has learned well from value illustration and grasped the use of sub-queries; see Appendix D for examples, and (iv) finally, the outcome for `WITH`-queries and `SET-operation` queries show an accuracy of 100%. However, the statistical significance of this performance is not guaranteed because of the small number of examples available in the dataset. For a more detailed understanding, we list NL/SQL-pairs for each type of complex query in Table 3 of Appendix D.

We particularly investigated datasets in which the underlying LLM fails to answer any of their questions; see Figure 6 in Appendix F. In the `tourism_economy_by_canton` English dataset, there exists a few false predictions due to the wrong understanding of data granularity. However, others are correctly labeled in terms of usefulness metrics, resulting in 66.67% of both EA_{soft} and $\text{EA}_{\text{partial}}$.

Given that each dataset was manually curated by various experts, we found that the observed disparity between $\text{EA}_{\text{strict}}$ and user's intent-based EA, namely soft and partial, is concentrated in certain datasets, which is in line with the previous assumption that semantic ambiguity is often subjective and individual. This insight of finding also demonstrates the importance of EA_{soft} and $\text{EA}_{\text{partial}}$ from the experimental perspective.

In the datasets of `staatsausgaben_nach_aufgabenbereichen_cofog` and `government_expenditure_by_function_cofog`, the knowledge misunderstanding related to the granularity of the data led to the majority of wrongly generated queries. Upon examining the similarity scores between the target question and the few-shot examples within each dataset, it becomes clear that there is a considerable level of disparity in the natural language questions found within these two datasets.

6 Conclusions

Our work highlights the potential and current limitations of Large Language Models (LLMs) using our novel bilingual Text-to-SQL benchmark StatBot.Swiss. While LLMs like GPT-3.5-Turbo and mixtral-8x7b show promising results for relatively simple queries, they face challenges with complex queries, domain knowledge inference, and handling `NULL` values. The analysis underscores the necessity for enhanced in-context learning in LLMs to incorporate a broader range of Text-to-SQL and database knowledge. The observed disparities in error analysis, especially in datasets with semantic ambiguities and data granularity issues, call for a more nuanced approach to evaluating the performance of Text-to-SQL systems, considering correctness from the user's perspective as apposed to exact matching.

Our work sets a benchmark for future developments in bilingual Text-to-SQL systems and emphasizes the importance of diverse and complex datasets in advancing LLM capabilities. As the field progresses, refining these models to better understand and execute multi-column grouping, numeric operations, and domain-specific queries will be crucial for realizing their full potential in real-world applications.

In future work, we aim to include other languages, e.g., French and Italian, in our benchmark and extend towards cross-lingual Text-to-SQL tasks.

7 Limitations

One major constraint of our study lies in the intrinsic challenges of Text-to-SQL evaluations across bilingual datasets. This includes a limited number of Large Language Models (LLMs) assessed in our experiments, the finite languages chosen, and the complexity and diversity of datasets.

Despite our comprehensive analysis, the limited dataset size, the constrained domain scope, the imbalanced query types, and the predefined hardness may not fully represent the breadth of other real-world applications.

Additionally, there is a lack of natural language/SQL-pairs across languages. That means that our benchmark dataset does not use English questions to query a database in German, or the other way around.

Lastly, our study assumes the target datasets are known beforehand, which could potentially bias the interpretation and application of the Text-to-SQL models' performance. This presupposition that the user or upstreaming network ideally grasps domain-specific knowledge may not always hold true, especially in diverse real-world scenarios where domain expertise varies widely. Such assumptions are apt to yield experimental outcomes that surpass the actual accuracy in realistic applications.

Acknowledgments

This work is the output of the INODE4StatBot.swiss project[6], funded by the Swiss Federal Statistical Office. We want to thank Patrick Arnecke from the Cantonal Statistical Office Zurich as well as Yara Abu Awad, Christine Chorat and Bertrand Loison from the Swiss Federal Statistical Office for valuable feedback during the project.

References

Daria Bakshandaeva, Oleg Somov, Ekaterina Dmitrieva, Vera Davydova, and Elena Tutubalina. 2022. PAUQ: Text-to-SQL in Russian. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2355–2376, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda

[6]https://www.zhaw.ch/en/research/research-database/project-detailview/projektid/5959/

Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Ursin Brunner and Kurt Stockinger. 2021. Valuenet: A natural language-to-sql system that learns from database information. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2177–2182.

Matthew Byrd and Shashank Srivastava. 2022. Predicting difficulty and discrimination of natural language questions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 119–130, Dublin, Ireland. Association for Computational Linguistics.

Shuaichen Chang and Eric Fosler-Lussier. 2023. How to prompt LLMs for text-to-SQL: A study in zero-shot, single-domain, and cross-domain settings. In *NeurIPS 2023 Second Table Representation Learning Workshop*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2024. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations*.

Jan Milan Deriu, Katsiaryna Mlynchyk, Philippe Schläpfer, Álvaro Rodrigo, Dirk Von Gruenigen, Nicolas Kaiser, Kurt Stockinger, Eneko Agirre, and Mark Cieliebak. 2020. A methodology for creating question answering corpora using inverse data annotation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 897–911.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Longxu Dou, Yan Gao, Mingyang Pan, Dingzirui Wang, Wanxiang Che, Dechen Zhan, and Jian-Guang Lou. 2023. Multispider: towards benchmarking multilingual text-to-sql semantic parsing. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press.

Avrilia Floratou, Fotis Psallidas, Fuheng Zhao, Shaleen Deep, Gunther Hagleither, Wangda Tan, Joyce Cahoon, Rana Alotaibi, Jordan Henkel, Abhik Singla, Alex van Grootel, Brandon Chow, Kai Deng, Katherine Lin, Marcos Campos, Venkatesh Emani, Vivek Pandit, Victor Shnayder, Wenjing Wang, and Carlo Curino. 2024. Nl2sql is a solved problem... not! In *14th Annual Conference on Innovative Data Systems Research (CIDR '24)*.

Zezhou Huang, Pavan Kalyan Damalapati, and Eugene Wu. 2023. Data ambiguity strikes back: How documentation improves GPT's text-to-SQL. In *NeurIPS 2023 Second Table Representation Learning Workshop*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. KaggleDBQA: Realistic evaluation of text-to-SQL parsers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2261–2273, Online. Association for Computational Linguistics.

Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Chenhao Ma, Kevin C. C. 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.

Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S Yu. 2023. A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability. *arXiv preprint arXiv:2303.13547*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. 2023. Enhancing text-to-SQL capabilities of large language models: A study on prompt design strategies. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14935–14956, Singapore. Association for Computational Linguistics.

Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

A. Rudra and S.L. Nimmagadda. 2005. Roles of multidimensionality and granularity in warehousing australian resources data. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, pages 216b–216b.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Anh Tuan Nguyen, Mai Hoang Dao, and Dat Quoc Nguyen. 2020. A pilot study of text-to-SQL semantic parsing for Vietnamese. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4079–4085, Online. Association for Computational Linguistics.

Pius Von Däniken, Jan Milan Deriu, Eneko Agirre, Ursin Brunner, Mark Cieliebak, and Kurt Stockinger. 2022. Improving nl-to-query systems through reranking of semantic hypothesis. In *Proceedings of the 5th International Conference on Natural Language and Speech Processing (ICNLSP 2022)*, pages 57–67.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020a. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578.

Jiaan Wang, Fandong Meng, Yunlong Liang, Tingyi Zhang, Jiarong Xu, Zhixu Li, and Jie Zhou. 2023. Understanding translationese in cross-lingual summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3837–3849, Singapore. Association for Computational Linguistics.

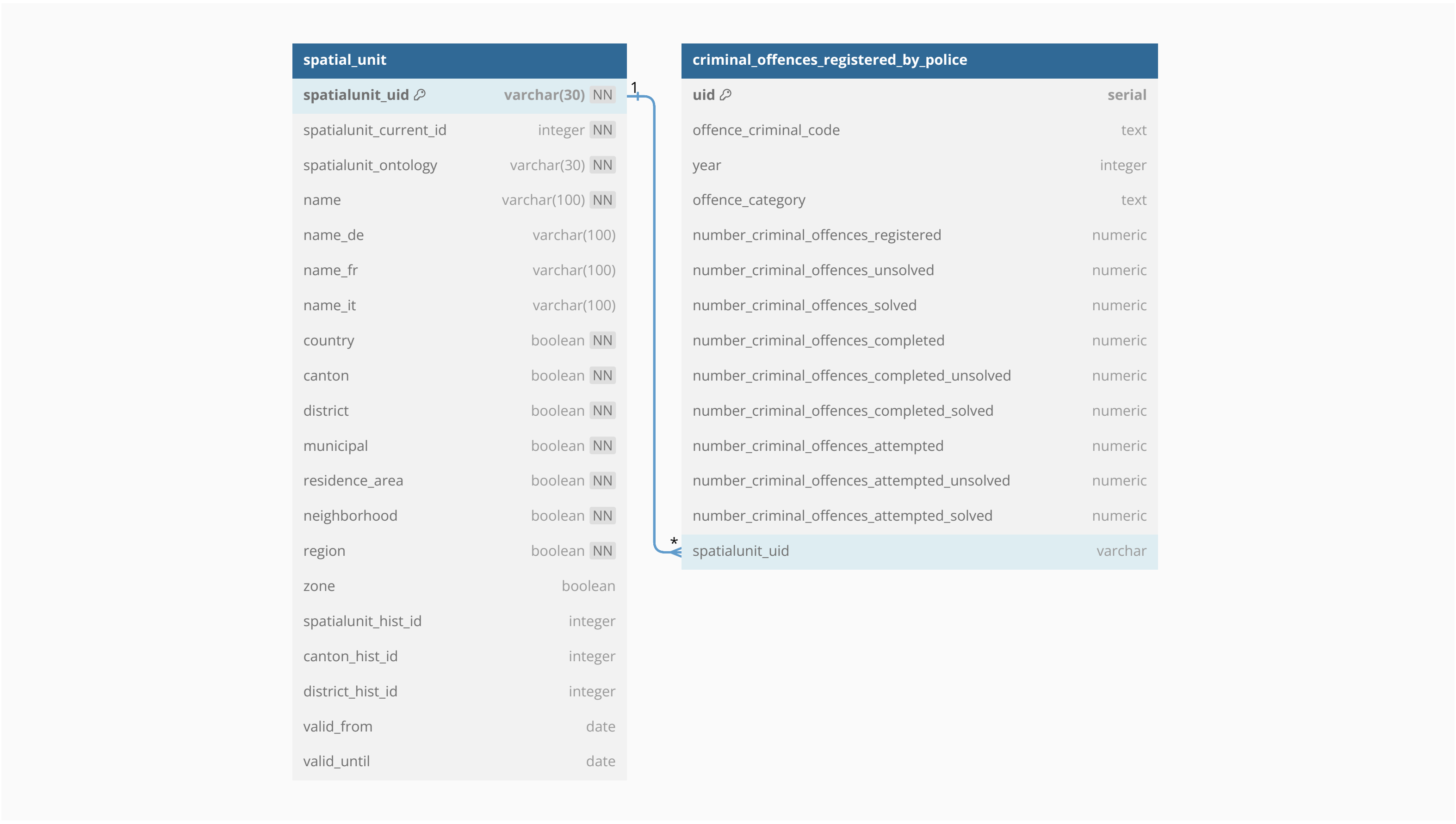
Lijie Wang, Ao Zhang, Kun Wu, Ke Sun, Zhenghua Li, Hua Wu, Min Zhang, and Haifeng Wang. 2020b. DuSQL: A large-scale and pragmatic Chinese text-to-SQL dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6923–6935, Online. Association for 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.

Yi Zhang, Jan Deriu, George Katsogiannis-Meimarakis, Catherine Kosten, Georgia Koutrika, and Kurt Stockinger. 2024. Sciencebenchmark: A complex real-world benchmark for evaluating natural language to sql systems. *Proceedings of the VLDB Endowment*, 17(4):685–698.

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

A Database Schema Example



spatial_unit		
spatialunit_uid	varchar(30)	NN
spatialunit_current_id	integer	NN
spatialunit_ontology	varchar(30)	NN
name	varchar(100)	NN
name_de	varchar(100)	
name_fr	varchar(100)	
name_it	varchar(100)	
country	boolean	NN
canton	boolean	NN
district	boolean	NN
municipal	boolean	NN
residence_area	boolean	NN
neighborhood	boolean	NN
region	boolean	NN
zone	boolean	
spatialunit_hist_id	integer	
canton_hist_id	integer	
district_hist_id	integer	
valid_from	date	
valid_until	date	

criminal_offences_registered_by_police	
uid	serial
offence_criminal_code	text
year	integer
offence_category	text
number_criminal_offences_registered	numeric
number_criminal_offences_unsolved	numeric
number_criminal_offences_solved	numeric
number_criminal_offences_completed	numeric
number_criminal_offences_completed_unsolved	numeric
number_criminal_offences_completed_solved	numeric
number_criminal_offences_attempted	numeric
number_criminal_offences_attempted_unsolved	numeric
number_criminal_offences_attempted_solved	numeric
spatialunit_uid	varchar

Figure 4: Entity-relationship diagram of the knowledge domain criminal offences. `spatial_unit` is the *dimension* table and `criminal_offenses_registered_by_police` is the *fact* table. EN = English, `NN` stands for `NOT NULL` constraint. Note that the dimension table `spatial_unit` contains information about different levels of granularity and thus enables aggregating facts by, e.g. `municipality`, `canton` and `country`. However, note that not all facts contain information about all levels of granularity. For instance, some facts are only collected at municipality level while others are collected a cantonal level.

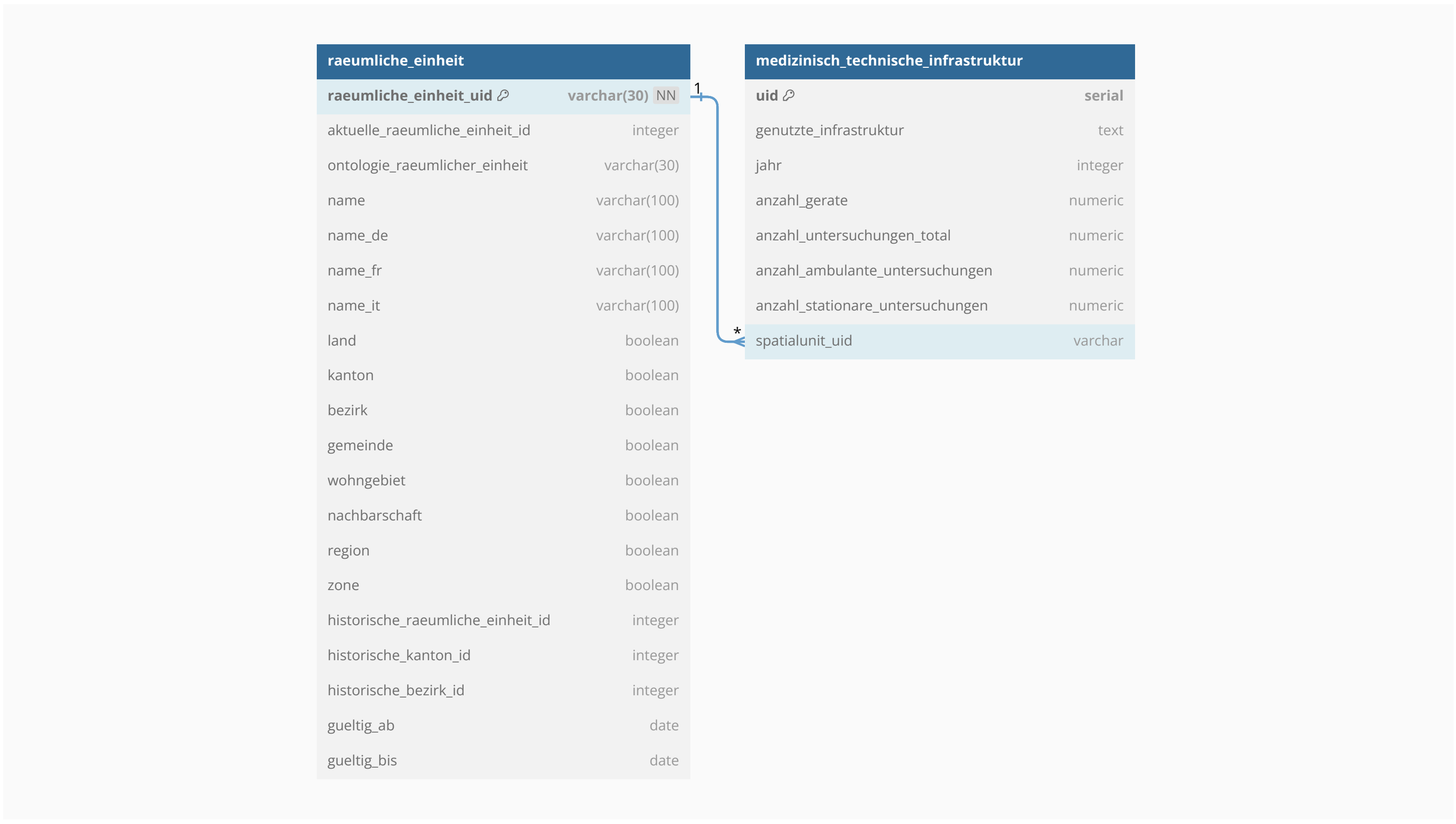

raeumliche_einheit
raeumliche_einheit_uid varchar(30) NN
aktuelle_raeumliche_einheit_id integer
ontologie_raeumlicher_einheit varchar(30)
name varchar(100)
name_de varchar(100)
name_fr varchar(100)
name_it varchar(100)
land boolean
kanton boolean
bezirk boolean
gemeinde boolean
wohngebiet boolean
nachbarschaft boolean
region boolean
zone boolean
historische_raeumliche_einheit_id integer
historische_kanton_id integer
historische_bezirk_id integer
gueltig_ab date
gueltig_bis date
1
*
medizinisch_technische_infrastruktur
uid serial
genutzte_infrastruktur text
jahr integer
anzahl_gerate numeric
anzahl_untersuchungen_total numeric
anzahl_ambulante_untersuchungen numeric
anzahl_stationare_untersuchungen numeric
spatialunit_uid varchar

Figure 5: Entity-relationship diagram of the knowledge domain `medizinisch_technische_infrastruktur` [DE] (in Eng. `medical technical infrastructure`), where `NN` stands for `NOT NULL` constraint. Note that the dimension table `raeumliche_einheit` contains information about different levels of granularity and thus enables aggregating facts by, e.g. `Gemeinde` (in English: municipality), `Kanton` (in English: canton) and `Land` (in English: country). However, note that not all facts contain information about all levels of granularity. For instance, some facts are only collected at municipality level while others are collected at cantonal level.

B Queries that Cannot be Analyzed by the Spider Hardness Evaluator

Here we show the complex query categories of our new dataset that use additional PostgresSQL features that cannot be evaluated by the Spider hardness evaluator:

1. `GROUP BY` or `ORDER BY` of more than one column, for instance, `GROUP BY offence_criminal_code, number_criminal_offences_registered`.
2. Nested `SELECT`-query alias, e.g., `SELECT a.col_1 FROM (SELECT...) AS A`. Unlike the column alias, these nested `SELECT`-queries are often required to break down complex queries into more managable sub queries and may cause errors when omitted.
3. `WITH`-queries also enable breaking down larger queries into smaller sub queries.
4. Built-in `Functions` except basic `aggregators`[7], e.g., `CAST`-function for type casting.
5. Numeric operators mixed with `aggregators`, such as `100*SUM(number_criminal_offences_solved)/SUM(number_criminal_offences_registered)`.
6. `SET`-operators to combine sub queries, e.g. `UNION ALL`, `INTERSECT ALL`, or `EXCEPT ALL`.

[7]The basic `aggregators` denotes `COUNT`, `SUM`, `MAX`, `MIN`, and `AVG`

7. Special keywords, e.g., `IN`, `CASE`.
8. `NULL` values.

C Query Hardness Distribution of the Bilingual Dataset

	English					German				
	easy	medium	hard	extra	unknown	easy	medium	hard	extra	unknown
Train	3 (2.26)	5 (3.76)	6 (4.51)	37 (27.82)	82 (61.65)	0 (0.00)	4 (2.23)	17 (9.50)	37 (20.67)	121 (67.60)
Dev	2 (3.28)	8 (13.11)	9 (14.75)	24 (39.34)	18 (29.51)	0 (0.00)	8 (9.76)	9 (10.98)	24 (29.27)	41 (50.00)
All	5 (2.58)	13 (6.70)	15 (7.73)	61 (31.44)	100 (51.55)	0 (0.00)	12 (4.60)	26 (9.96)	61 (23.37)	162 (62.07)

Table 2: Hardness distribution per language for each dataset. The values in parentheses represent the distribution of samples corresponding to the hardness level across languages and datasets in percentage.

D Example Prompt

Below we show the prompt for two examples (2-shot). First, we show the CREATE TABLE statement (database schema) followed by several example rows (data values) per table. Afterward, we illustrate natural language questions and their corresponding SQL queries. The examples are about the usage of electric cars in certain areas.

```
You are a helpful AI assistant who writes SQL query for a given question. Given
the database described by the database schema below, write a SQL query that
answers the question.
Do not explain the SQL query.
Return just the query, so it can be run verbatim from your response.
```

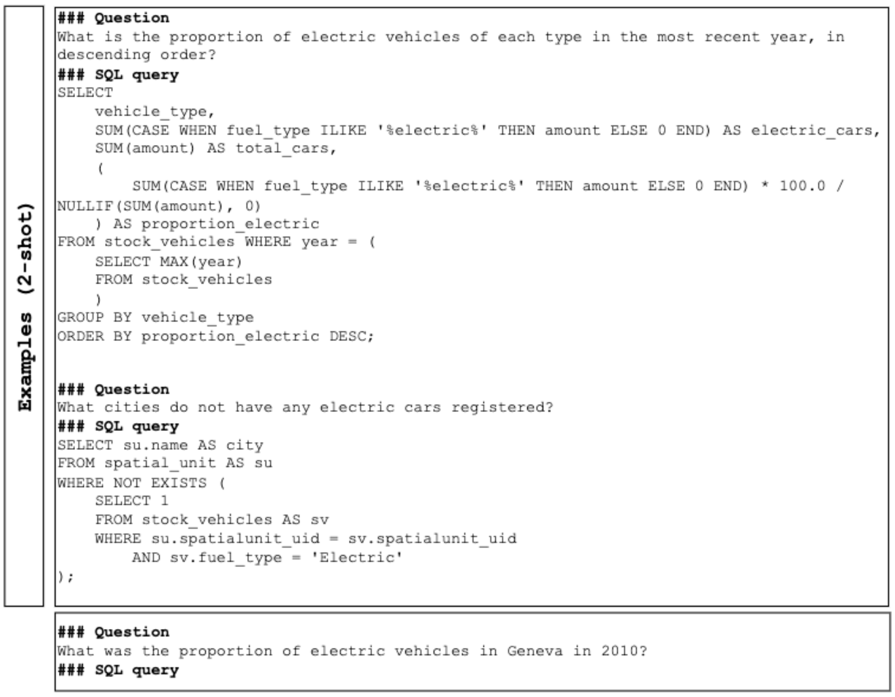
Database Information

```
### Database Schema

CREATE TABLE experiment.spatial_unit (
    spatialunit_uid VARCHAR(30) NOT NULL,
    spatialunit_current_id INTEGER NOT NULL,
    spatialunit_ontology VARCHAR(30) NOT NULL,
    name VARCHAR(100) NOT NULL,
    name_de VARCHAR(100),
    name_fr VARCHAR(100),
    name_it VARCHAR(100),
    country BOOLEAN DEFAULT false NOT NULL,
    canton BOOLEAN DEFAULT false NOT NULL,
    district BOOLEAN DEFAULT false NOT NULL,
    municipal BOOLEAN DEFAULT false NOT NULL,
    residence_area BOOLEAN DEFAULT false NOT NULL,
    neighborhood BOOLEAN DEFAULT false NOT NULL,
    region BOOLEAN DEFAULT false NOT NULL,
    zone BOOLEAN,
    spatialunit_hist_id INTEGER,
    canton_hist_id INTEGER,
    district_hist_id INTEGER,
    valid_from DATE,
    valid_until DATE,
    CONSTRAINT spatial_unit_pkey PRIMARY KEY (spatialunit_uid)
)
/*
Columns in spatial_unit and 5 examples in each column for high cardinality columns :
spatialunit_uid : 6195_A.ADM3, 695_A.ADM3, 6152_A.ADM3, 2783_A.ADM3, 4065_A.ADM3
spatialunit_current_id : 2092, 2304, 6195, 5169, 4311
name : Berlens, Uebeschi, Saint-Gingolph, Jongny, Hallwilersee (LU)
name_de : Berlens, Uebeschi, Saint-Gingolph, Jongny, Hallwilersee (LU)
name_fr : Berlens, Uebeschi, Saint-Gingolph, Jongny, Hallwilersee (LU)
name_it : Berlens, Uebeschi, Saint-Gingolph, Jongny, Hallwilersee (LU)
spatialunit_hist_id : 15230, 13245, 14740, 11434, 10088
canton_hist_id : 16, 14, 10, 23, 1
district_hist_id : 10266, 10229, 10009, 10313, 10088
valid_from : 1995-01-01, 2001-04-13, 2018-01-01, 2011-01-01, 2017-01-01
valid_until : 2010-04-24, 2017-12-31, 2012-12-31, 1967-05-31, 1971-12-31
*/

CREATE TABLE experiment.stock_vehicles (
    uid BIGINT NOT NULL,
    spatialunit_uid VARCHAR(100),
    vehicle_type VARCHAR(50),
    year INTEGER,
    amount INTEGER,
    fuel_type VARCHAR(50),
    CONSTRAINT stock_vehicles_pk PRIMARY KEY (uid),
    CONSTRAINT stock_vehicles_spatial_unit_spatialunit_uid_fk FOREIGN
KEY(spatialunit_uid) REFERENCES experiment.spatial_unit (spatialunit_uid)
)
/*
Columns in stock_vehicles and 5 examples in each column for high cardinality columns :
uid : 1601120, 1601106, 1601202, 1601155, 1601156
spatialunit_uid : 5307_A.ADM3, 5430_A.ADM3, 3237_A.ADM3, 5912_A.ADM3, 5926_A.ADM3
year : 2015, 2019, 2020, 2010, 2012
amount : 16, 41, 305, 877, 1599
fuel_type : Diesel-electric: Plug-in-hybrid, Hydrogen, total, Benzine-electric: Plug-in-
hybrid, Diesel
*/

/*
Column name, Column description, Example values
vehicle_type, Vehicle type, passenger_cars,trailers,motorcycles
amount, Amount of vehicles, 0,2,240
fuel_type, Fuel type, Hydrogen,Electric,Diesel-electric: Normal-hybrid
*/
```


Examples (2-shot)

```
### Question
What is the proportion of electric vehicles of each type in the most recent year, in
descending order?
### SQL query
SELECT
    vehicle_type,
    SUM(CASE WHEN fuel_type ILIKE '%electric%' THEN amount ELSE 0 END) AS electric_cars,
    SUM(amount) AS total_cars,
    (
        SUM(CASE WHEN fuel_type ILIKE '%electric%' THEN amount ELSE 0 END) * 100.0 /
NULLIF(SUM(amount), 0)
    ) AS proportion_electric
FROM stock_vehicles WHERE year = (
    SELECT MAX(year)
    FROM stock_vehicles
    )
GROUP BY vehicle_type
ORDER BY proportion_electric DESC;


### Question
What cities do not have any electric cars registered?
### SQL query
SELECT su.name AS city
FROM spatial_unit AS su
WHERE NOT EXISTS (
    SELECT 1
    FROM stock_vehicles AS sv
    WHERE su.spatialunit_uid = sv.spatialunit_uid
        AND sv.fuel_type = 'Electric'
);
```

```
### Question
What was the proportion of electric vehicles in Geneva in 2010?
### SQL query
```

Table 3: Complex SQL query examples with hardness "unknown". Note that these kind of queries cannot be handled by the Spider hardness evaluator due to their high complexity especially in terms of SQL features used.

Query Types	**[db_id] | *Question* | `Query`**

1. `GROUP BY` > 1 column	[volksabstimmung_nach_kanton_seit_1861]
	Welche Kantone haben 2023 gegen das Bundesgesetz über Klimaschutz gestimmt und wieviel Prozent Ja Stimmen gab es dort jeweils?
	`SELECT S.name_de AS kanton_gegen_klimaschutzgesetz,`
	`T.ja_in_prozent`
	`FROM volksabstimmung_nach_kanton_seit_1861 AS T`
	`JOIN spatial_unit AS S`
	`ON T.spatialunit_uid = S.spatialunit_uid`
	`WHERE S.canton = 'TRUE'`
	`AND LOWER(T.vorlage) LIKE '%bundesgesetz%klimaschutz%'`
	`AND T.jahr = 2023`
	`AND T.ja_in_prozent <= 50`
	`GROUP BY S.name_de, T.vorlage, T.jahr, T.ja_in_prozent``;`

2. Nested `SELECT`-Query Alias	[marriage_citizenship]
	Show me the lowest number of marriages that occurred at the canton level in 1990, where both the wife and husband were from different nationalities?
	`SELECT A.*`
	`FROM (SELECT T2.name, T1.citizenship_category_husband,`
	`T1.citizenship_category_wife, T1.amount`
	`FROM marriage_citizenship AS T1`
	`JOIN spatial_unit AS T2`
	`ON T1.spatialunit_uid = T2.spatialunit_uid`
	`WHERE T2.canton = 'True'`
	`AND T1.year = 1990`
	`AND T1.citizenship_category_husband = 'Foreign country'`
	`AND T1.citizenship_category_wife = 'Switzerland'`
	`UNION`
	`SELECT T2.name, T1.citizenship_category_husband,`
	`T1.citizenship_category_wife, T1.amount`
	`FROM marriage_citizenship as T1`
	`JOIN spatial_unit AS T2`

Table 3 (Continued): Complex SQL query examples of hardness "unknown".

Query Types	[db_id] | *Question* | `Query`

```
ON T1.spatialunit_uid = T2.spatialunit_uid
WHERE T2.canton = 'True'
AND T1.year = 1990
AND T1.citizenship_category_husband = 'Switzerland'
AND T1.citizenship_category_wife = 'Foreign country') AS A
ORDER BY A.amount ASC
LIMIT 1;
```

3. Built-in Function

[basel_land_bevolkerung_nach_nationalitat_konfession_gemeinde]

Welcher Anteil der Bevölkerung von Basel-Landschaft gehörte 2021 einer bekannten Religion an?

```
SELECT 1 - (
SUM( CAST(T.anzahl_unbekannt_konfession AS FLOAT) )
/ SUM(T.gesamt_anzahl_personen))
AS proportion_known_religion_basel_land_2021
FROM
basel_land_bevolkerung_nach_nationalitat_konfession_gemeinde
AS T
JOIN spatial_unit AS S ON T.spatialunit_uid = S.spatialunit_uid
WHERE S.municipal = 'TRUE'
AND T.jahr = 2021
GROUP BY T.jahr;
```

4. WITH-Queries

[medizinisch_technische _infrastruktur]

Welchem drei Kantone hatte den grössten Zuwachs und Untersuchungen mit medizinischen Geräten zwischen 2013 and 2021 und wieviel hoch war der Zuwachs verteilt auf ambulante und stationäre Untersuchungen und Geräte?

```
WITH Untersuchungen2013 AS (
SELECT SUM(T1.anzahl_untersuchungen_total)
AS anzahl_untersuchungen_gesamt_2013,
SUM(T1.anzahl_ambulante_untersuchungen)
AS anzahl_ambulante_untersuchungen_2013,
SUM(T1.anzahl_stationare_untersuchungen)
AS anzahl_stationare_untersuchungen_2013,
SUM(T1.anzahl_gerate) AS anzahl_gerate_2013,
S1.name_de AS kanton
FROM medizinisch_technische_infrastruktur AS T1
JOIN spatial_unit AS S1
ON T1.spatialunit_uid = S1.spatialunit_uid
WHERE S1.canton = 'TRUE' AND T1.jahr = 2013
GROUP BY S1.name_de),
Untersuchungen2021 AS (
SELECT SUM(T2.anzahl_untersuchungen_total)
AS anzahl_untersuchungen_gesamt_2021,
SUM(T2.anzahl_ambulante_untersuchungen)
AS anzahl_ambulante_untersuchungen_2021,
SUM(T2.anzahl_stationare_untersuchungen)
AS anzahl_stationare_untersuchungen_2021,
SUM(T2.anzahl_gerate) AS anzahl_gerate_2021,
S2.name_de AS kanton2021
FROM medizinisch_technische_infrastruktur AS T2
JOIN spatial_unit AS S2
ON T2.spatialunit_uid = S2.spatialunit_uid
WHERE S2.canton = 'TRUE'
AND T2.jahr = 2021
GROUP BY S2.name_de)
SELECT U2013.kanton,
U2021.anzahl_untersuchungen_gesamt_2021 -
U2013.anzahl_untersuchungen_gesamt_2013
AS zuwachs_untersuchungen_gesamt,
U2021.anzahl_ambulante_untersuchungen_2021 -
U2013.anzahl_ambulante_untersuchungen_2013
AS zuwachs_ambulante_untersuchungen,
```

Table 3 (Continued): Complex SQL query examples of hardness "unknown".

Query Types	**[db_id]** | ***Question*** | **`Query`**

	`U2021.anzahl_stationare_untersuchungen_2021 -`
	`U2013.anzahl_stationare_untersuchungen_2013`
	`AS zuwachs_statinonare_untersuchungen,`
	`U2021.anzahl_gerate_2021 - U2013.anzahl_gerate_2013 AS`
	`zuwachs_geraete`
	`FROM Untersuchungen2013 AS U2013`
	`JOIN Untersuchungen2021 AS U2021`
	`ON U2013.kanton = U2021.kanton2021`
	`ORDER BY zuwachs_untersuchungen_gesamt DESC`
	`LIMIT 3;`

5. Numeric Operators mixed with Aggregators	[resident_population_birthplace_citizenship_type]
	
	What was the percentage of the population in Switzerland who were born in a foreign country on 2017?
	
	`SELECT SUM(un.in), SUM(un.out),`
	`SUM(un.out)/(SUM(un.in)+SUM(un.out))` `AS percentage`
	`FROM (SELECT T1.year, T1.place_of_birth, T1.citizenship,`
	`SUM(T1.amount) AS in, SUM(0) AS out`
	`FROM resident_population_birthplace_citizenship_type AS T1`
	`JOIN spatial_unit AS T2`
	`ON T1.spatialunit_uid = T2.spatialunit_uid`
	`WHERE T2.country = 'True' AND T1.year = 2017`
	`AND T1.place_of_birth = 'Switzerland'`
	`AND T1.citizenship = 'Citizenship - total'`
	`GROUP BY T1.year, T1.place_of_birth, T1.citizenship`
	`UNION`
	`SELECT T1.year,T1.place_of_birth,T1.citizenship,`
	`SUM(0) AS in, SUM(T1.amount) AS out`
	`FROM resident_population_birthplace_citizenship_type AS T1`
	`JOIN spatial_unit AS T2`
	`ON T1.spatialunit_uid = T2.spatialunit_uid`
	`WHERE T2.country = 'True' AND T1.year = 2017`
	`AND T1.place_of_birth = 'Abroad'`
	`AND T1.citizenship = 'Citizenship - total'`
	`GROUP BY T1.year, T1.place_of_birth, T1.citizenship)`
	`AS un;`

6. SET-Operation Query	[marriage_citizenship]
	
	In 2000, among the municipalities, which had the highest and lowest numbers of marriages where the husbands were Swiss citizens?
	
	`(SELECT T2.name, T2.spatialunit_ontology, T1.year, T1.amount`
	`FROM marriage_citizenship as T1`
	`JOIN spatial_unit AS T2`
	`ON T1.spatialunit_uid = T2.spatialunit_uid`
	`WHERE T2.municipal = 'True'`
	`AND T1.year = 2000`
	`AND T1.citizenship_category_husband = 'Switzerland'`
	`AND T1.citizenship_category_wife = 'Citizenship of wife -`
	`total'`
	`AND T1.amount = (SELECT Max(T1.amount)`
	`FROM marriage_citizenship as T1`
	`JOIN spatial_unit AS T2`
	`ON T1.spatialunit_uid = T2.spatialunit_uid`
	`WHERE T2.municipal = 'True'`
	`AND T1.year = 2000`
	`AND T1.citizenship_category_husband = 'Switzerland'`
	`AND T1.citizenship_category_wife = 'Citizenship of wife -`
	`total')`
	`LIMIT 1)`
	`UNION ALL`
	`(SELECT T2.name, T2.spatialunit_ontology, T1.year, T1.amount`
	`FROM marriage_citizenship as T1`
	`JOIN spatial_unit AS T2`

Table 3 (Continued): Complex SQL query examples of hardness "unknown".

Query Types	**[db_id]** | ***Question*** | **`Query`**

	`ON T1.spatialunit_uid = T2.spatialunit_uid`
	`WHERE T2.municipal = ’True’`
	`AND T1.year = 2000`
	`AND T1.citizenship_category_husband = ’Switzerland’`
	`AND T1.citizenship_category_wife = ’Citizenship of wife -`
	`total’`
	`AND T1.amount = (SELECT MIN(T1.amount)`
	`FROM marriage_citizenship as T1`
	`JOIN spatial_unit AS T2`
	`ON T1.spatialunit_uid = T2.spatialunit_uid`
	`WHERE T2.municipal = ’True’`
	`AND T1.year = 2000`
	`AND T1.citizenship_category_husband = ’Switzerland’`
	`AND T1.citizenship_category_wife = ’Citizenship of wife -`
	`total’)`
	`LIMIT 1);`

7. Special Keywords	[stock_vehicles]
	
	What was the proportion of electric vehicles in Geneva in 2010?
	
	`SELECT`
	`SUM(`**`CASE WHEN`**` sv.fuel_type `**`ILIKE`**` ’%electric%’ `**`THEN`**
	`sv.amount `**`ELSE`**` 0 `**`END`**`) AS electric_vehicles,`
	`SUM(sv.amount) AS total_vehicles,`
	`(SUM(CASE WHEN sv.fuel_type `**`ILIKE`**` ’%electric%’`
	`THEN`` sv.amount `**`ELSE`**` 0 `**`END`**`) * 100.0 / NULLIF(SUM(sv.amount),`
	`0))`
	`AS proportion_electric_vehicles`
	`FROM spatial_unit AS su`
	`INNER JOIN stock_vehicles AS sv`
	`ON su.spatialunit_uid = sv.spatialunit_uid`
	`WHERE su.name `**`ILIKE`**` ’%geneva%’`
	`AND sv.year = 2010;`

8. NULL-value	[nationalratswahlen]
	
	In welchen Kantonen stand die Partei SVP 2019 nicht zur Wahl?
	
	`SELECT S.name_de AS kanton_ohne_svp_2019`
	`FROM nationalratswahlen AS T`
	`JOIN spatial_unit AS S`
	`ON T.spatialunit_uid = S.spatialunit_uid`
	`WHERE S.canton = ’TRUE’`
	`AND T.partei = ’SVP’`
	`AND T.jahr = 2019`
	`AND T.parteistarke_in_prozent IS `**`NULL`**`;`

Table 3 (End): Complex SQL query examples of hardness "unknown".

E Failure Case

Upon examining the failed prediction of easy queries, we identified the root cause as a key term swap between `COUNT` and `DISTINCT` during the few-shot ICL execution on Mixtral. As shown below, the discrepancy between the expected and predicted queries is evident for the Mixtral setting at an easy difficulty level in the few-shot scenario.

Data Schema	baby_names_favorite_firstname
Question	How many favorite baby names are registered in year 2011?
Ground Truth	`SELECT COUNT(DISTINCT first_name) FROM baby_names_favorite_firstname WHERE year = 2011`
Predicted Query	`SELECT DISTINCT COUNT(bnff.first_name) FROM baby_names_favorite_firstname as bnff WHERE bnff.year = 2011`

F Detailed Results

Experiments	Shot	GPT-3.5			Mixtral		
		EA_{strict}	EA_{soft}	$EA_{partial}$	EA_{strict}	EA_{soft}	$EA_{partial}$
Zero-shot	0	13.52 (0.33)	13.76 (0.33)	17.95 (0.33)	5.82 (0.33)	5.82 (0.33)	6.52 (0.33)
Few-shot (Random)	1	36.36 (0.00)	41.26 (0.00)	46.15 (0.00)	18.18 (0.00)	21.68 (0.00)	23.08 (0.00)
	2	36.36 (0.44)	41.68 (0.34)	46.57 (0.34)	21.68 (0.57)	27.97 (0.57)	29.37 (0.57)
	3	35.02 (0.81)	41.12 (0.52)	46.01 (0.52)	23.78 (0.44)	29.23 (1.03)	31.75 (1.75)
	4	36.36 (0.44)	41.26 (0.44)	43.50 (0.52)	**24.20** (0.56)	**30.07** (0.89)	**32.87** (0.89)
	5	36.83 (0.33)	**44.53**(0.33)	**50.58** (0.66)	23.78 (0.57)	**30.07** (0.57)	32.40 (0.87)
	6	**37.76** (0.00)	42.66 (0.00)	45.45 (0.00)	21.68 (0.00)	26.57 (0.00)	29.84 (0.33)
	8	36.36 (0.00)	41.26 (0.00)	44.76 (0.00)	21.21 (0.33)	29.60 (0.33)	31.94 (0.33)
Few-shot (Similarity)	1	33.57 (0.00)	38.46 (0.00)	41.26 (0.00)	16.92 (0.28)	23.78 (0.00)	26.57 (0.00)
	2	33.71 (0.28)	38.60 (0.28)	40.56 (0.00)	21.12 (1.12)	26.71 (1.12)	28.95 (0.95)
	3	39.02 (0.28)	45.73 (0.56)	47.83 (0.56)	22.52 (0.82)	28.81 (0.82)	30.63 (0.69)
	4	39.16 (0.44)	46.15 (0.44)	48.25 (0.44)	26.01 (0.69)	32.31 (0.69)	34.13 (0.82)
	5	**41.68** (0.56)	**48.25** (0.44)	**50.07** (0.34)	21.68 (0.44)	30.77 (0.44)	35.94 (0.71)
	6	40.56 (0.44)	45.45 (0.44)	46.85 (0.44)	**28.39** (0.34)	**35.38** (0.34)	**38.18** (0.34)
	8	39.30 (0.52)	44.90 (0.52)	46.99 (0.52)	21.40 (0.34)	28.95 (0.34)	31.47 (0.44)

Table 4: Execution accuracy for different query evaluation metrics (strict, soft and partial). The upper part shows few-shot results where the samples are *randomly* selected. The bottom part shows few-shot results where the samples are selected based on a *similarity score*. Accuracy is presented by the average and standard deviation across three separate and independent runs.

Experiment	Hadness	GPT-3.5		Mixtral	
		EN	DE	EN	DE
Zero-shot	Easy `(2,0)`	100.00 (0.00)	0.00 (0.00)	100.00 (0.00)	0.00 (0.00)
	Medium `(8,8)`	29.17 (5.89)	37.50 (0.00)	41.67 (5.89)	12.50 (0.00)
	Hard `(9,9)`	11.11 (0.00)	22.22 (0.00)	0.00 (0.00)	11.11 (0.00)
	Extra hard `(24,24)`	0.00 (0.00)	29.17 (0.00)	4.17 (0.00)	0.00 (0.00)
	Unknown `(18,41)`	0.00 (0.00)	4.88 (0.00)	0.00 (0.00)	0.00 (0.00)
Few-shot (similarity)	Easy `(2,0)`	100.0 (0.00)	0.00 (0.00)	50.00 (0.00)	0.00 (0.00)
	Medium `(8,8)`	75.00 (0.00)	50.00 (0.00)	45.00 (6.12)	37.50 (0.00)
	Hard `(9,9)`	11.11 (0.00)	44.44 (0.00)	11.11 (0.00)	33.33 (0.00)
	Extra hard `(24,24)`	29.17 (2.50)	50.00 (0.00)	25.00 (0.00)	45.83 (0.00)
	Unknown `(18,41)`	27.78 (0.00)	44.88 (1.20)	11.11 (0.00)	24.39 (0.00)

Table 5: Strict execution accuracy (EA_{strict}) of zero-shot and optimal few-shot results per query hardness (easy, medium, hard, extra hard, and unknown).

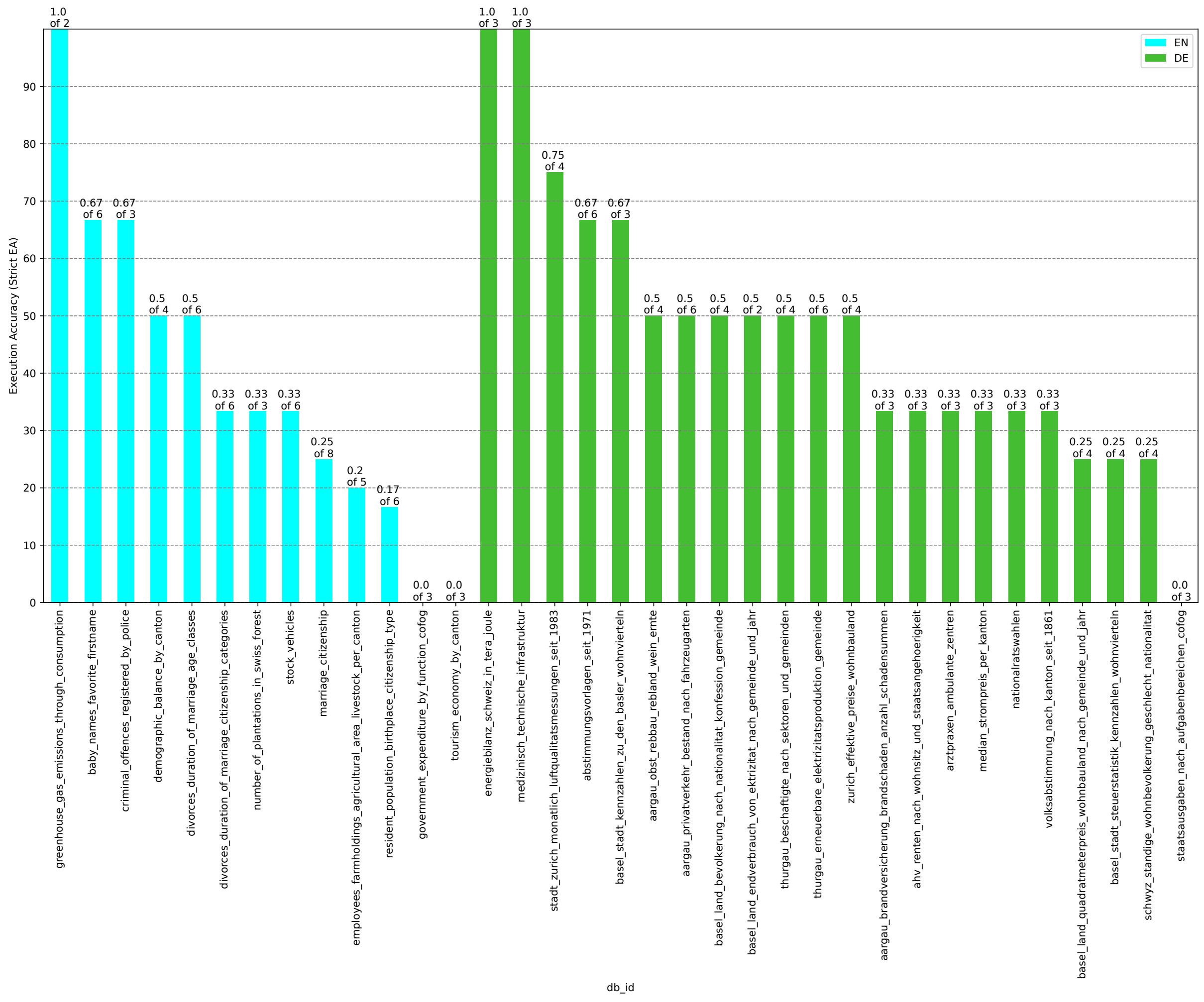


Figure 6: Strict execution accuracy (EA_{strict}) per knowledge domain and language over 35 different databases. Left hand: English. Right hand: German.

Experiment	Execution Accuracy (EA)	GPT-3.5		Mixtral	
		EN `(#82)`	DE `(#61)`	EN `(#82)`	DE `(#61)`
Zero-shot	strict	8.75 (0.77)	17.07 (0.00)	10.39 (0.77)	2.44 (0.00)
	soft	8.75 (0.77)	17.48 (0.58)	10.39 (0.77)	2.44 (0.00)
	partial	10.39 (0.77)	23.17 (0.00)	12.02 (0.77)	2.44 (0.00)
Few-shot	strict	34.43 (0.00)	46.83 (0.60)	22.29 (0.80)	32.93 (0.00)
	soft	40.00 (0.80)	54.15 (0.60)	27.21 (0.80)	41.46 (0.00)
	partial	41.67 (0.66)	57.08 (0.49)	30.49 (0.80)	43.90 (0.00)

Table 6: Mean execution accuracy (standard deviation) using three different metrics (strict, soft and partial) for zero-shot and optimal few-shot experiments. The results are shown for the two different languages English (EN) and German (DE). `#XX` represents the number of NL/SQL-pairs in the development set for each language.