Evaluating the Data Model Robustness of Text-to-SQL Systems Based on Real User Queries

Jonathan Fürst
Zurich University
of Applied Sciences
jonathan.fuerst@zhaw.ch

Catherine Kosten
Zurich University
of Applied Sciences
catherine.kosten@zhaw.ch

Farhard Nooralahzadeh
Zurich University
of Applied Sciences
farhad.nooralahzadeh@zhaw.ch

Yi Zhang
Zurich University
of Applied Sciences
yi.zhang@zhaw.ch

Kurt Stockinger
Zurich University
of Applied Sciences
kurt.stockinger@zhaw.ch

ABSTRACT

Text-to-SQL systems (also known as NL-to-SQL systems) have become a popular solution for bridging the gap between user capabilities and SQL-based data access. These systems translate user requests in natural language to valid SQL statements for a specific database. Text-to-SQL systems have benefited from the rapid improvement of transformer-based language models. However, while Text-to-SQL systems that incorporate such models continuously reach new high scores on—often synthetic—benchmark datasets, a systematic exploration of their robustness towards different data models in a real-world, realistic scenario is missing.

This paper provides the *first in-depth evaluation of the data model robustness of Text-to-SQL systems in practice* based on a multi-year international project on Text-to-SQL interfaces. In contrast to prior works evaluating Text-to-SQL systems on often synthetically generated generalized multi-domain benchmarks, we evaluate different data models in one domain based on *real-user questions.* Our evaluation is based on *FootballDB*, a system that was deployed over a 9-month period in the context of the FIFA World Cup 2022, during which about 6K natural language questions were asked and executed. We manually labeled a diverse subset of these questions for three different data models. *For each data model*, we explore the performance of representative Text-to-SQL systems and language models. We further quantify the impact of training data size, pre- and post-processing steps as well as inference time. Our comprehensive evaluation *sheds light on the design choices of real-world Text-to-SQL systems and their impact on moving from research prototypes to real deployments*. Last, we provide a new benchmark dataset, which is the first to *enable the evaluation of different data models for the same dataset* and is substantially more challenging than most previous datasets in terms of query complexity.

1 INTRODUCTION

Since the inception of relational databases, SQL has served as the primary query language for structured data sources, offering a declarative interface for application developers to access information. In the last decades, SQL has gained importance beyond application developers, as businesses move to data-driven decision making in the context of data warehouses and data lakes: Cities tackle complex challenges such as carbon reduction [23], and scientists in various domains, from social to natural sciences are increasingly following data-driven methods in their research. This development has led to a shift, in which non-programmers represent a large group of (potential) SQL users. For these users, the SQL interface and the required knowledge of the relational model constitute an often impervious barrier [2].

To address the gap between user capabilities and data access, a growing number of Text-to-SQL (also known as NL-to-SQL) systems have been developed in the last years [1, 26, 29, 36]. The idea is compelling: Instead of writing SQL queries, users write queries in natural language and a Text-to-SQL system, with access to the underlying database, translates them into valid SQL statements. Over the years, there have been various methods from early, keyword and rule-based systems [6] to the more recent deep learning-based approaches [7, 48] that have pushed SQL generation accuracy on common benchmarks such as Spider [57] to 91.2% (as of November 2023 [25]). At the core of this success, lie the advancements in transformer-based language models, from Bert [16] (340M parameters) and Bart [37] (148M parameters), to T5 [45] (3B parameters) to the advent of Large Language Models (LLMs), such as OpenAI's ChatGPT [42] and GPT-4 [41], Google's Bard [27] or Meta's LLaMA2 [50] (up to 100s of billions of parameters).

While Text-to-SQL systems incorporating these models continuously reach new high scores on—often synthetic—benchmark datasets, enterprise-grade Text-to-SQL is still far from being resolved as recently noted in [20]. *In fact, a systematic exploration of their design space through a real-world deployment with actual user queries does not exist.* The available survey and analysis papers on Text-to-SQL systems [1, 32] mainly provide a taxonomy, the needed steps and advantages as well as disadvantages of various approaches. However, they do not experimentally explore the design space, specifically the data model robustness of any deployed Text-to-SQL systems with real users. Likewise, many instrumental benchmarks have been proposed for evaluating Text-to-SQL over the years (e.g., Spider [57], KaggleDBQA [34], BIRD [39]), but they are not constructed from real-world user queries and do not provide the possibility to compare the impact of different data model design choices on the same set of queries.

In this paper, we provide the *first in-depth design evaluation of the data model robustness of Text-to-SQL systems* based on a multi-year international project on natural language interfaces for databases [3]. Because previous benchmarks and research have been solely fixated on the generalization capabilities of Text-to-SQL systems, in this work, we focus on a single domain *Football* to interrogate other critical aspects of the Text-to-SQL design space. Through this exploration, we: (i) *Investigate the effect of a growing amount of labeled training data on the accuracy*

of translating user questions to SQL, (ii) *Examine the impact of data model design choices on the results of translating user questions to SQL*, (iii) *Analyze the role of query complexity in the process of translating user questions to SQL*, and (iv) *Evaluate Text-to-SQL inference time for different systems*

We experimentally evaluate these questions using FootballDB—a newly created open dataset with FIFA World Cup data. FootballDB contains *information about world cup games over a 100-year time period.* We collected and curated the dataset semi-automatically from several open data sources (e.g., DBpedia [4], Wikidata [52]) and through web scraping (e.g., FIFA website). We modeled the data in three different data models: starting from a widely used football data model and two adaptations based on our deployment experience and user feedback.

Unlike other public Text-to-SQL datasets, *our labeled and evaluated queries have been asked by real users of a deployed Text-to-SQL system several months before and during the FIFA World Cup 2022.* Our system has been queried by several hundred users that issued about 6K natural language questions, 400 of which we have manually labeled for three data models (1200 NL/SQL pairs). The FootballDB dataset adds a fresh outlook to the landscape of Text-to-SQL benchmarks, providing new insights to practitioners who want to deploy these systems on their own databases and researchers who can now evaluate the data model robustness of their developed systems.

Overall, this paper makes the following <u>contributions</u>:

- It offers a *comprehensive analysis and definition of the design space for Text-to-SQL solutions*, namely (1) Data Model, (2) Language Model, (3) Training Data Size and (4) Pre-/Post-processing (Section 2).
- We provide unique insights into *the challenges and requirements of collecting real-user data as well as data model design decisions* (Sections 4 and 5) for Text-to-SQL practitioners.
- We conduct a *rigorous experimental evaluation of all design space dimensions*, employing state-of-the-art Text-to-SQL systems using small, medium, and large language models along with a realistic, novel dataset and user queries derived from several months of live deployment (Section 6).
- Drawing from our deployment experience and experimental results, we suggest promising avenues for *future research, focusing on the significance of data models and training data composition*, opposed to merely expanding the volume of labeled data (Section 7).
- Last, we *release FootballDB, a new realistic benchmark dataset* with three data model instances with end-user generated queries to the community: https://github.com/jf87/FootballDB

2 DESIGN SPACE OF TEXT-TO-SQL SYSTEMS

Task Definition. Text-to-SQL systems translate the provided natural language question into a corresponding SQL query. We formally define the process as follows [38, 44, 53]: Given a natural language question as a sequence of natural language tokens $Q : q_1 q_2 \ldots q_{|Q|}$ and a relational database schema with $S = \{\langle \mathcal{T}_1, C_1 \rangle, \ldots, \langle \mathcal{T}_n, C_n \rangle\}$ where $\mathcal{T}_i$ is a table, and C_i denotes it's corresponding columns (i.e. DB content), a Text-to-SQL system is a function $f(Q, S)$ that outputs the correct SQL query $\mathcal{Y} : y_1 y_2 \ldots y_{|\mathcal{Y}|}$ as a sequence of tokens.

2.1 Deep Text-to-SQL Systems

State of the art Text-to-SQL systems generate the SQL query $\mathcal{Y}$ with the help of Language Models (LMs) [7, 30, 56]. LMs are transformer-based [51], pre-trained sequence-to-sequence models. Generally, these models process Q, the sequence of natural language tokens through layers of self-attention and feed-forward networks to capture contextual information, and then generate SQL output tokens $\mathcal{Y}$. While, LMs constitute the core functionality of deep Text-to-SQL systems, published methods have added model fine-tuning and various pre- and post-processing steps to customize them for the SQL generation task (see Figure 1).

Pre-processing. In pre-processing, NL questions need to be encoded as tokens Q together with database schema information S. In addition, some methods also use the database content (e.g., ValueNet [7]) and/or perform input enrichment (e.g., schema linking in IRNet [30] and RAT-SQL [53]). Schema linking connects NL question entities to database tables, columns, and cell values.

Training/Fine-tuning. LMs have been pre-trained extensively on large text corpora (usually obtained from Internet sources). As such, they are not optimized for Text-to-SQL translations. To provide Text-to-SQL specialization, methods often apply two steps: (1) The model is pre-trained with a large corpus of NL/SQL-pairs (e.g., obtained from a public dataset such as Spider [57]); (2) The model is then further fine-tuned with a smaller set of database-specific NL/SQL-pairs.

Post-processing. Last, the predicted tokens $\mathcal{Y}$ are converted to a sequence of words, representing a SQL query. Some systems (e.g., IRNet and ValueNet) further use an intermediate representation (IR) such as SemQL which they then convert to SQL. IRs can help bridge the gap between the context free SQL grammar and natural language [30]. E.g., SemQL eliminates SQL `GROUPBY`, `HAVING` and `FROM` clauses, and conditions in `WHERE` and `HAVING` are uniformly expressed in the subtree of `Filter` in the SemQL query [30]. While reducing the gap between NL and SQL, the conversion from SemQL to SQL is not lossless. E.g., when a query involves multiple tables, join conditions are chosen heuristically. NatSQL [24] is another widely used IR with a wider range of supported SQL queries.

A different approach to deal with the gap between NL and SQL has been proposed by Picard [48]. As LMs have an unconstrained output space, they can produce invalid SQL statements. Picard thus introduced an incremental parsing method that constrains the decoding outputs in LMs to valid SQL.

2.2 Design Space Dimensions

Based on our observations, we derive the following important design space dimensions for Text-to-SQL systems:

[**D1**] *Data Model.* The database data model (database schema) is one of the main inputs for many Text-to-SQL systems. Furthermore, the design of the data model directly influences the complexity of the corresponding SQL queries. On the one hand, for the relational model, the database community has—through many years of theoretical and practical research—defined design practices (e.g., entity-relationship model [10]), normalization theories and techniques that reduce data redundancy and improve data integrity [13]. On the other hand, for Online Analytical Processing (OLAP) in data warehouses, the star schema [9] is a data model optimized for efficient data analysis based
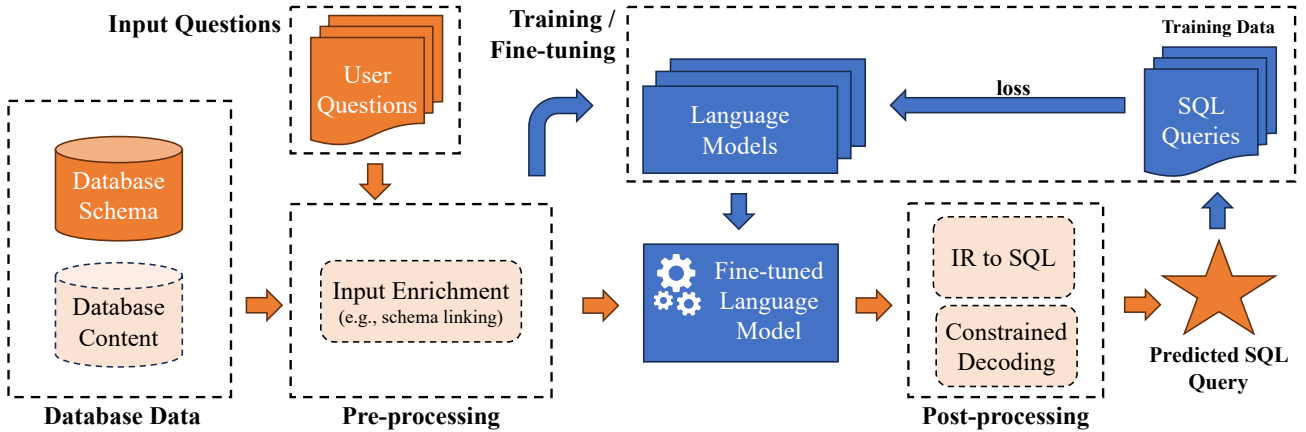
Figure 1: Design space of deep learning-based Text-to-SQL methods (fine-tuning phase in blue, prediction phase in orange). NL questions, database schema and potentially content serve as input. Pre-processing: Some methods perform input enrichment before the encoded input is fed into a fine-tuned language model. Post-processing: Some methods apply techniques such as intermediate representation (IR) or constrained decoding to improve the quality of the generated SQL query.

on de-normalization. *Currently the effect of data models on Text-to-SQL performance is completely under-explored.*

[D2] *Language Model.* A wide set of increasingly larger transformer-based language models have been proposed in recent years [1, 16, 27, 41, 42]. These pre-trained language models not only differ in size (from millions to billions of parameters), but also in terms of their availability (open-source, on premise vs. closed-source, cloud-only) and required training/fine-tuning and run-time costs. *The trade-offs of different language model sizes in terms of prediction performance and required costs have not been explored for Text-to-SQL thus far.*

[D3] *Training Data Size.* The size of the training data used for fine-tuning clearly impacts the performance of the final machine learning model. Surprisingly, the training data size dimension has not been systematically investigated for Text-to-SQL systems. While some have explored domain generalization [38], to our knowledge, all existing methods have been evaluated on fixed-sized train and test datasets (e.g., from popular benchmarks such as Spider [57] and WikiSQL [62]). However, when using *Text-to-SQL systems in practice, there is an important trade-off between the effort and cost spent creating labeled training data and the performance.*

[D4] *Pre- and Post-processing.* Various contributions have been made in the pre- and post-processing steps of Text-to-SQL systems (e.g., input enrichment and intermediate representation [7, 30] or constrained decoding [48]). *Currently it is not clear how pre- and post-processing influences the performance of Text-to-SQL translation for different data models.*

3 FOOTBALLDB OVERVIEW

FootballDB serves as a *novel, challenging benchmark for evaluating Text-to-SQL systems based on real user questions and with regards to different data models.* First, we describe how we construct FootballDB, a dataset comprising almost 100 years of football

world cup data. Second, we describe the deployment of FootballDB live before and during the FIFA World Cup 2022.
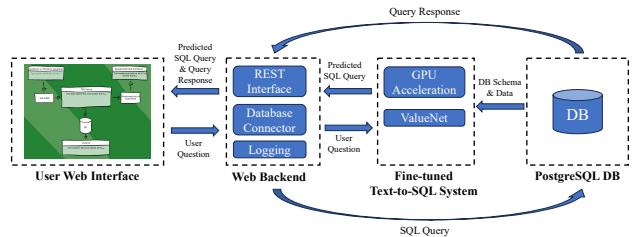

Figure 2: Implementation of the Text-to-SQL system deployed before, during and after the FIFA World Cup 2022.

3.1 FootballDB Dataset

The FootballDB dataset is an extensive world cup dataset with information about football games, players, national teams, and clubs dating from the first world cup in 1930 in Uruguay, to the most recent 2022 world cup in Qatar. Football world cups typically take place every four years over a time period of about one month. The number of participating teams has risen from 13 teams in the inaugural World Cup to 32 teams in the most recent edition. In total, we collected information about 22 world cups, 86 national teams (including former nations, e.g., the Soviet Union), 8,891 players, and 1,874 clubs.

Original Data. We sourced the initial dataset from the Kaggle FIFA World Cup dataset [5]. This initial dataset includes 3 CSV-files describing the elimination round matches, players, and final playoffs. However, it only covered the years from 1930 to 2018, contained several erroneous data points, and lacked information about players and national clubs.

Data Enrichment and Cleaning. To provide a comprehensive view of the World Cup and address frequent user queries about the clubs of players, we supplemented the dataset with additional information on leagues, clubs, and coaches from Wikidata. We also performed extensive manual checks to ensure high-quality and accurate data. For example, we included the full

names of players (previously, only nicknames of partial names were used) and their dates of birth to de-duplicate and augment player data. We also correct inaccuracies and added missing players for each participating country's team. In total, we added 1,230 new players, 89 leagues, 1,874 clubs, and 1,966 coaches to the initial dataset.

To provide an accurate and updated data source for viewers and fans of the FIFA World Cup, we continuously updated our database after each set of matches throughout the event in 2022. Specifically, we added new data points for each match, including detailed information about goals, penalties, red and yellow cards, as well as the minutes during the game when each of these events occurred. *FootballDB can be considered the most accurate, comprehensive, open-source relational database for the world cup championship to date.*

3.2 FootballDB Deployment

For our deployment, we created a modular architecture for the user web interface, including a web back-end and Text-to-SQL system. Data was stored in a PostgreSQL database (see Figure 2). During our live deployment, we employed ValueNet [7] as the Text-to-SQL system. We chose ValueNet since the source code is available and it has been used extensively in various projects between academia and industry.

Text-to-SQL System: ValueNet. We based our deployment on ValueNet, which is built on some of the components of IR-Net [30]. The main novelty of ValueNet is a pre-processing method that not only employs IRNet's schema linking mechanism but also extracts values from natural language questions and infers possible value candidates from the base data of a given database, even when not explicitly stated in the natural language question. ValueNet uses a BART encoder [37] (148M parameters) and incorporates the database schema and content as input to the neural network. For post-processing, ValueNet uses an IR (SemQL) for translating natural language to SQL. This approach is limited to the SQL language features supported by the grammar (see also Section 2.2).

Initial Data Model. We created an initial data model for our FootballDB dataset based on our domain knowledge and commonly applied database modeling practices [8, 49]. This initial data model contains information about players, matches, stadiums, national teams, clubs, coaches, etc. (see Figure 3). Overall, the resulting database schema contains 13 tables with 14 foreign key constraints

In the following two Sections, we detail the requirements, challenges, and our process of collecting real-user questions (Section 4) and then provide an in-depth discussion of the design decisions that went into the development of our three data models (Section 5).

4 REAL-USER DATA CHALLENGES

Despite the intensive research focused on improving Text-to-SQL performance on existing benchmarks, very little work on how real users interact with systems in the real world exists. This section aims to diminish this gap, by reporting the requirements, challenges, and process of collecting NL/SQL pairs through real-user interactions.

Challenge 1: Attracting Users. A major roadblock in evaluating natural language interfaces for relational databases in the real world is to attract a diverse set of users. To attract a larger

user pool, we chose the FIFA World Cup 2022 because of its international attention and diverse audience. Football is a topic that many people, regardless of their background, education, or culture, can ask questions about. We advertised the deployment within the university and with our industry partners and promoted it during citizen-research engagement events and to first-year computer science undergraduates. From the beginning of the live deployment in October 2022 until the end of the world cup in December 2022, we collected around 6K queries from several hundred users.

Challenge 2: Identifying and Fulfilling User Needs. A second challenge is to create a system that fulfills the information needs of users by providing answers to their football-related questions. Therefore, we continuously analyzed the posed questions to discover emerging topics that could not be answered by the content of the database. The questions showed a trend in user interest in the clubs that players had played in, which coaches had coached the players, and which leagues were currently division one. Based on this feedback *we added new data to our database* about clubs, leagues, players, and coaches.

Challenge 3: Harnessing User Knowledge. A third requirement is to benefit from the SQL and domain knowledge of users to receive feedback on the quality of the generated outputs. This is important for understanding if the information needs for a specific question are actually fulfilled by the provided answer. We designed a *new interface for expert users* that are either familiar enough with SQL to recognize that a SQL query is incorrect or expert enough in world cup statistics to know that the output of the query is incorrect. Users could add a thumbs up or thumbs down, indicating whether a given result is correct or not. SQL experts could edit and correct the SQL query, which was then logged by our system. These additional data points were then used to filter and clean the logged natural language questions and SQL queries.

Challenge 4: Labeling User Data. Last, data labeling is a big bottleneck in most ML systems, usually requiring manual effort [12, 21, 22]. Here, this effort translated to manually verifying and labeling all posed user questions and generated SQL outputs. Therefore, processes to reduce this manual effort are needed. For queries that had been marked correct by users (see Challenge 3), we manually verified them. For queries that had been marked either incorrect or not at all by users, we developed several automation techniques to speed up the otherwise enormous manual effort: First, we computed the cosine similarity of the embedding vectors of all natural language questions using *SentenceBERT* [47] to *compute the semantic textual similarities*. We then set a high similarity threshold ($\geq$ 96%) to automatically label queries that were very similar to previously manually verified queries. Second, we also *used this similarity measure as a labeling aid* to show labelers similar to previously validated natural language questions and SQL query pairs (< 96%). Having a similar query helped labelers to more quickly and accurately identify errors in queries and correct them, e.g., missing filters, incorrect table joins, missing columns in projections, or missing aggregations.

Overall Observations. We observed interesting patterns in the posed user questions that should be considered in the design and evaluation of future Text-to-SQL systems: 1) *unrelated questions*, 2) *unanswerable questions*, (3) ambiguous questions (4) *questions in languages other than English*, and (5) a multitude of *spelling errors* for player names. These observations are similar to the recently presented open issues in achieving enterprise-grade Text-to-SQL [20]. Specifically, their two main problems (1) natural
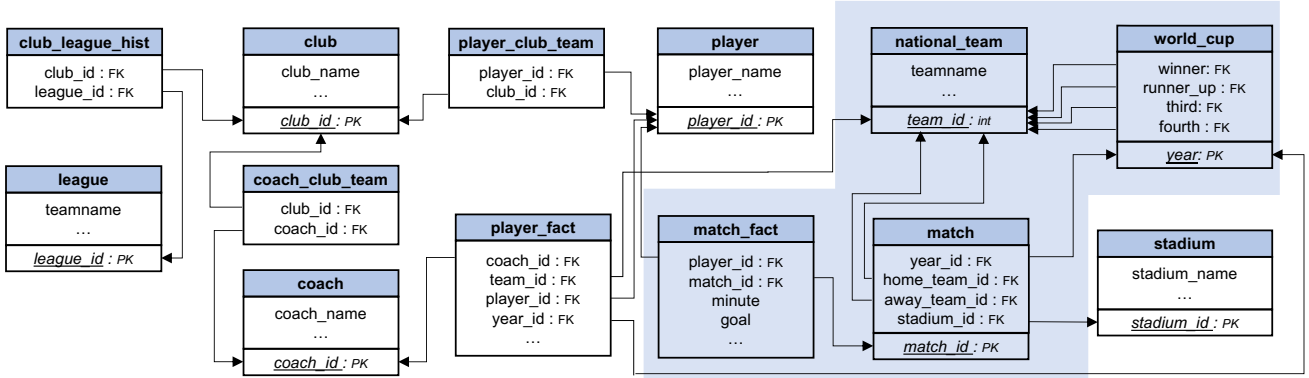
Figure 3: World Cup Schema Diagram v1. PK = Primary Key. FK = Foreign Key. The tables highlighted in blue are affected by re-designing the data model in various iterations. Note that the 1:n relationships between tables `national_team` and `match` as well as between `national_team` and `world_cup` contain multiple PK/FK references.

language ambiguity and (2) semantic mismatch (i.e., the user's intent cannot be effectively fulfilled by the provided database), can also be directly observed in our collected data. We collected around 6K user interactions (see Table 1). For these, our system was able to generate SQL queries 89% of the time. Failures to generate SQL queries are generally due to: an NL question in a different language, out of scope, lack of training data representing similar questions.

Type of User Log	Amount of Logs
#NL questions issued	5,900
#Times SQL generated	5,275
#Times no SQL generated	625
#Thumbs up	174
#Thumbs down	949
#User corrected SQL queries	1,287

Table 1: Statistics of live user logs collected during and after the world cup from November 21, 2022 to December 31, 2022.

5 DATA MODEL DESIGN

We discuss two adaptations for improving the initial data model of FootballDB. Our primary objectives in the design process have been to create a data model driven by two goals:

[**G1**] *User Goal.* We aim to reduce the complexity of SQL queries and support NL questions that are commonly asked by users in our deployment as described in Section 4.

[**G2**] *System Goal.* We aim to minimize potential errors that originate in the Text-to-SQL system's pipeline, such as in the pre- and post-processing steps or within the language model as described in Section 2.2.

5.1 Starting Point: Data Model v1

Our initial data model (see **v1** in Figure 3) consists of 13 tables with 14 foreign key constraints. Note that four tables are highlighted in blue. These are the focus of remodeling as they contain the core information about national teams and matches. This focus is supported by the fact that these tables are accessed in around 74% of user questions in our deployment.

As described in Section 4, we started with data model v1 depicted in Figure 3. Note the $1 : n$ relationship between `match` and

`national_team`. We can see that table `match` contains multiple foreign keys that reference the table `national_team`. However, this causes problems for Text-to-SQL systems that employ an intermediate representation (IR) such as SemQL as part of their post-processing step. E.g., the shortest path algorithm employed by such systems for generating SQL queries only supports a *single primary key/foreign key references between any two tables* to form the subgraph of the join path [7, 30]. Hence, the table join-path algorithm fails at the post-processing stage. E.g., the SQL query in v1 listed at the left side of Figure 4 corresponds to the natural language question *"What was the score between Germany and Brazil in 2014?"* given data model v1. The yellow components of the query represent the joins that cannot be correctly interpreted. A similar case is a 1:n relationship between `national_team` and `world_cup` in Figure 3, where `national_team` is referenced by four foreign keys.

During the initial deployment, we found that very few questions asked by users contained keywords related to the terms "home team" or "away team", defined by `home_team_id` and `away_team_id` in the table `match`. Instead, users preferred to describe matches in terms of one team against another, e.g., Brazil against Germany. As a consequence, the corresponding SQL query is too complex to conform to the intermediate representation format used in the pre-processing step shown in Figure 1. Moreover, the resulting SQL queries may be too long to fit the maximal length of the input tokens (1024) supported by the language model.

5.2 First Optimization: Data Model v2

To tackle the issue of only supporting a single primary key/-foreign key reference between two tables, we modified the data model as follows: The original $1 : n$ relationship between `national_team` and `match` is remodeled with the two bridge tables (also known as `composite entity types`[1]) `plays_as_home` and `plays_as_away` (see **v2** in Figure 5). As a result, we eliminate multiple primary/foreign key references between tables `match` and `national_team`. Note that in database schema v1 shown in Figure 3, there are also multiple primary/foreign key references between the tables `national_team` and `world_cup`. We again solve this issue by remodeling the table `world_cup` with one additional table `world_cup_result`.

---

[1] Proposed by the guide of creating correct ER diagram from [8, p.151-153]

SQL Query in v1	SQL Query in v2	SQL Query in v3

```
SELECT T2.teamname, T3.teamname,
T1.home_team_goals, T1.away_team_goals
FROM match AS T1
JOIN national_team AS T2
  ON T2.team_id = T1.home_team_id
JOIN national_team AS T3
  ON T3.team_id = T1.away_team_id
WHERE T2.teamname ILIKE '%Germany%' AND T3.teamname
ILIKE '%Brazil%' AND T1.year = 2014
UNION
SELECT T2.teamname, T3.teamname,
T1.home_team_goals, T1.away_team_goals FROM match
AS T1
JOIN national_team AS T2
  ON T2.team_id = T1.home_team_id
JOIN national_team AS T3 ON T3.team_id =
T1.away_team_id
WHERE T2.teamname ILIKE '%Brazil%' AND T3.teamname
ILIKE '%Germany%' AND T1.year = 2014;
```

```
SELECT T3.teamname, T5.teamname, T2.home_team_goals,
T4.away_team_goals
FROM match AS T1
JOIN plays_as_home AS T2 ON T1.match_id = T2.match_id
JOIN national_team AS T3 ON T2.team_id = T3.team_id
JOIN plays_as_away AS T4 ON T1.match_id = T4.match_id
JOIN national_team AS T5 ON T4.team_id = T5.team_id
WHERE T3.teamname ILIKE '%Brazil%' AND T5.teamname ILIKE
'%Germany%' AND T1.year = 2014
UNION
SELECT T3.teamname, T5.teamname, T2.home_team_goals,
T4.away_team_goals
FROM match AS T1
JOIN plays_as_home AS T2 ON T1.match_id = T2.match_id
JOIN national_team AS T3 ON T2.team_id = T3.team_id
JOIN plays_as_away AS T4 ON T1.match_id = T4.match_id
JOIN national_team AS T5 ON T4.team_id = T5.team_id
WHERE T3.teamname ILIKE '%Germany%' AND T5.teamname
ILIKE '%Brazil%' AND T1.year = 2014;
```

```
SELECT T1.teamname, T3.teamname,
T2.team_goals, T2.opponent_team_goals
FROM national_team AS T1
JOIN plays_match AS T2 ON T2.team_id = T1.team_id
JOIN national_opponent_team AS T3 ON T3.team_id =
T2.opponent_team_id
WHERE T1.teamname ILIKE '%Brazil%' AND T3.teamname
ILIKE '%Germany%' AND T2.year = 2014;
```

What was the score between Germany and Brazil in 2014?
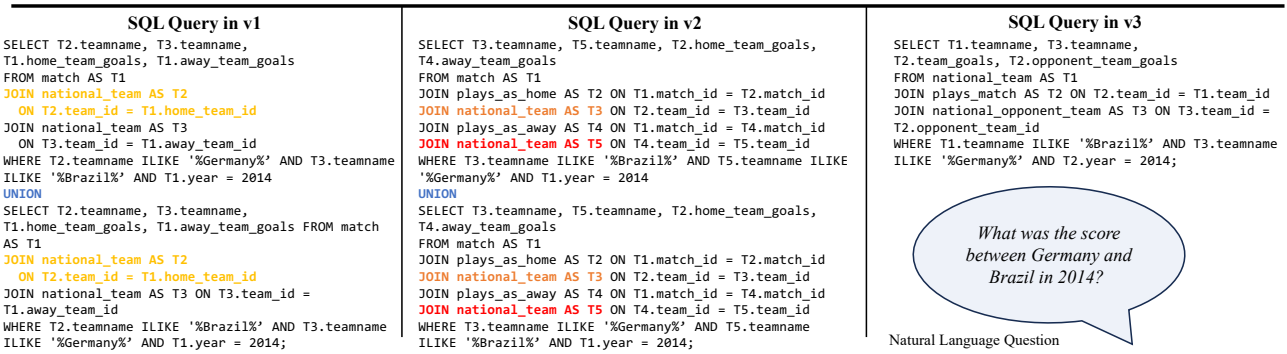
Natural Language Question

Figure 4: Example SQL query in all three data models of the natural language question: *What was the score between Germany and Brazil in 2014?* **Components shown in** yellow **indicate the unsuccessful interpretation of joins; components shown in** blue **indicate set operations; components shown in** orange **and** red **indicate the multiple instantiations of a table in a** SELECT **clause.**

Following the transition of the database schema from v1 to v2, the system now accurately handles the majority of single join paths. Nevertheless, a certain amount of queries remain unresolved as they *cannot pass through the Spider SQL parser* [57] in the stage of pre-processing. This parser is a core component of many Text-to-SQL systems such as IRNet [30], ValueNet [7] and RAT-SQL [53]. The parser does not support multiple table instances with different table aliases. These aliases result in an inferred SQL query, of which the join path may contain either match_home_team or match_away_team, but not both simultaneously.

This situation requires a workaround through the modification of SQL queries, involving the utilization of the set operation UNION (see SQL queries in v1 and v2 in Figure 4). However, this leads to a *twofold increase of the SQL query length and adds complexity to the query structure*. Further, when there is no workaround, and the occurrence of multiple instances of a specific table is inevitable, the SQL queries cannot be processed. In the center of Figure 4, the SQL query in v2 also demonstrates this case. The highlighted components (orange and red) indicate the multiple table instantiations that cause the input parser to fail during the pre-processing stage.

Another challenge is the well-known *lexical problem* [18], which is the cause of the poor performance for questions related to the prize columns in table world_cup_result. This effect was especially noticeable in the case of the prize of "runner-up". It appears that people prefer to use a more intuitive expression, such as "second place" or "lost in the final". Through analysis of all user questions, we discovered that these terms are in fact used ≈ 3 times as often.

5.3 Second Optimization: Data Model v3

To address these issues, we introduce the following hypothesis. For humans, query writing becomes easier, if the query (1) comprises fewer JOINs, which results in a shorter overall query length; (2) has a more intuitive structure with self-descriptive semantics; (3) and requires little or no implicit knowledge. Our hypothesis is that the factors in writing an easy query for humans should also influence deep Text-to-SQL systems, since their trained models aim to mimic human-like query generation capabilities. Hence, we modified the data model as follows (see **v3** in Figure 6).
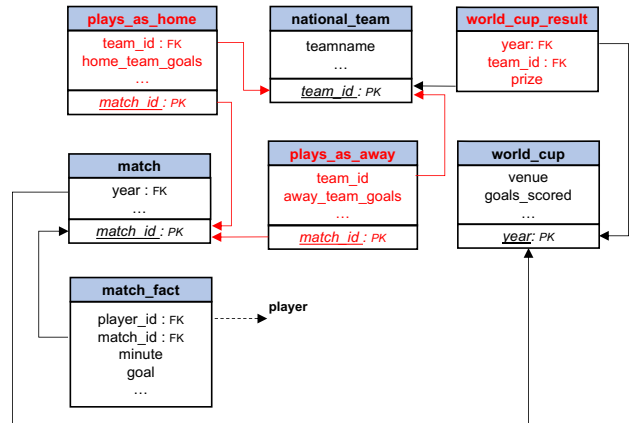

Figure 5: World Cup Schema Diagram Changes v2. Tables highlighted in red are changed with respected to data model version 1 shown in Figure 3. Original 1:n relationships containing more than one PK/FK references are remodeled to only contain a single PK/FK reference between two tables.

- To simplify writing queries about football matches between different national teams, we removed the non-intuitive tables plays_as_home as well as national_opponent_team and plays_match. Hence, the fact that, for instance, Brazil plays against Argentina can now more intuitively expressed by the query national_team ⋈ plays_match ⋈ national_opponent_team (see also right query in Figure 4).
- To implement the proper relationships adapted to the semantics of the natural language question, we redesign the relevant columns and foreign keys[1], and add the extra column team_role to indicate the role of the respective team either as home or as away.
- Since both teams in a match can be recognized as home or opponent, we adapt all the rows for the new plays_match table and add a new primary key match_team_id, which is a concatenation of match_id and team_id.

---

[1]Removing columns home_team_id and away_team_id; inserting team_id and opponent_team_id
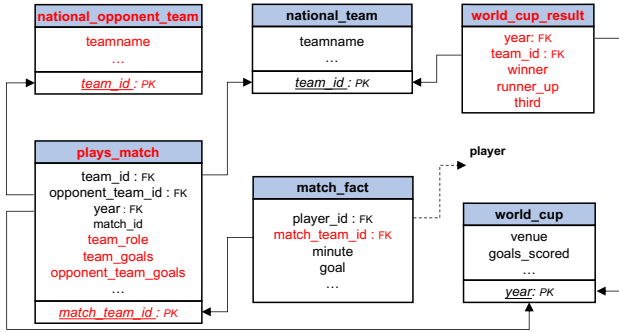
Figure 6: World Cup Schema Diagram Changes v3. Tables highlighted in red are changed with respect to data model version 2 shown in Figure 5.

	DB v1	DB v2	DB v3
#Tables	13	16	15
#Columns	97	98	107
#Rows	104,531	106,547	106,111
#FKs	14	13	16
Mean #Columns per Table	7.46	6.13	7.13
Mean #Rows per Table	8,041	6,659	7,074

Table 2: Characteristics of FootballDB across three different data models: Number of tables, columns, rows, etc.

- To enhance the ability of the language model to create links between the NL questions and the database schema, we expand the table `world_cup_result` and convert the single text column, `prize`, into four Boolean columns with more intuitive names, `winner`, `runner_up`, `third`, and `fourth`. This optimization simplifies the value extraction phase from searching for entities in both the schema and content of the database, to focus on the schema only.

Summary of Data Model Modifications. Table 2 shows the major characteristics of the three different versions of our data models and highlights all modified tables and columns. Note that data model v2 with has the highest number of tables and the lowest mean number of columns per table. In other words, this data model has a high number of tables, but a low number of columns and low mean number of columns per table. On the other hand, data model v3 has a high number of tables and columns, but it surprisingly simplifies the query writing process (see the observed example query length in Figure 4 as well as the mean query length in Table 3). We will explore the impact of different data models for Text-to-SQL systematically in the next section.

6 EXPERIMENTAL EVALUATION

We perform a detailed experimental evaluation of different Text-to-SQL systems and data models against our dataset FootballDB. Our overall goal is to investigate the affect of the identified design dimensions in Section 2.2 through the following research questions (RQs):

- RQ 1: *How do different data models change the accuracy of Text-to-SQL systems for translating natural language questions to SQL using our novel dataset collected by real users?* → Section 6.2

- RQ 2: *What is the accuracy of Text-to-SQL systems that leverage small, medium or large language models?* → Section 6.2
- RQ 3: *How does training data size impact Text-to-SQL performance?* → Section 6.2: Train Set Size
- RQ 4: *What are the systematic errors of Text-to-SQL systems based on query complexity and query characteristics?* → Section 6.3
- RQ 5: *What is the inference time of Text-to-SQL systems, and can they be used for interactive data exploration where response times of less than 3 seconds are expected?* → Section 6.4

6.1 Experimental Setup

Train and Test Set. As described in Section 4, we logged 6K real user interactions with our system. We filter these 6K questions to remove duplicates, questions in languages other than English, or questions that were entirely unrelated to the dataset, i.e., questions not about football. We then label the remaining NL/SQL pairs, i.e., we check if the SQL queries correctly expressed the NL questions, rewriting the SQL queries if not.

To create a rich and comprehensive dataset, linguistically and in terms of database coverage, we use a clustering-based approach to select a diverse set of NL questions. Specifically, we apply topic modeling using BertTopic [28] to create dense clusters of related NL questions. We then sample and label each cluster as follows. First, we sample the centroid NL question and then select all NL questions in the cluster that are below a threshold of 0.93 in terms of BERT similarity to the centroid. This threshold is chosen so that it results in approximately 1K questions, which we manually label for data model v3. By excluding samples proximal to the cluster centroid, we can effectively reduce the dataset size while minimizing the loss of diversity within the dataset. E.g., this method avoids labeling semantic and lexically similar questions such as "Who won the world cup in 2014?" and "Who won the world cup in 2018?".

Second, because manual labeling is a considerable effort (e.g., involving six expert annotators), we then only choose a subset of 400 NL/SQL-pairs to label for data models v1 and v2. These NL/SQL-pairs are chosen through uniform sampling from the 1K NL/SQL-pairs based on their SQL query hardness level proposed in the Spider paper [57]. The Spider SQL query hardness is rule-based and has 4 levels of difficulty: easy, medium, hard and extra hard. Easy queries have a single projection (this can be an aggregation or not) and no joins. Medium queries have more than 1 projection, more than 1 join and can include filters, order by, having or group by clauses. Hard queries have several projections and multi-table joins and can include set operations, multiple filters and order by, having or group by clauses. Extra hard queries have several projections, multi-table joins and can include set operations, subclauses as well as multiple filters and order by, having or group by clauses. *Overall, our method ensures a diverse sampling from the NL perspective (through topic modeling) and from the SQL complexity perspective (through Spider hardness). We release the full dataset, including the 6K raw dataset and 1K gold dataset for v3, which allows for a different sample selection strategy in the future.*

Overall this results in 1,200 manually corrected NL-SQL pairs for data model v1, v2 and v3 (400 each). Per data model, we use 100 records as the test set to evaluate the Text-to-SQL accuracy. The remaining 300 records are the train set. For our experiments, we

use 100, 200, and 300 records, respectively, to study the impact of increasing the size of the training data on the translation accuracy. The results are shown in Table 5 and Table 6.

Note that the *query characteristics vary greatly between the models* (see Table 3). For example, due to the modeling change in v3, set operations are no longer necessary. The v2 data model requires the most number of joins, because it contains the most tables. Surprisingly, the widely used Spider hardness level [57] only shows a small increase from v1 to v2, with the lowest value for v3 (we map the Spider hardness levels to numeric values from 1 for easy to 4 for extra hard.). This might indicate that the metric should be revised or expanded to better reflect other factors, such as the number of joins or the number of SQL tokens as shown in the recently released BIRD benchmark [40]. Note that FootBallDB queries have around 100% / 300% more joins and 10% / 100% more SQL tokens per query than the queries in the recent BIRD benchmark and the heavily used Spider benchmark [57], respectively.

Evaluation Metrics. A popular evaluation metric for Text-to-SQL systems is the Semantic Evaluation for Text-to-SQL with Test Suites [61]. However, this evaluation script is unable to parse some samples of our train and test sets due to the built-in limitations of its SQL parser [57]. Therefore, we apply *exact execution matching* (EX), also known as result matching [33] as the accuracy metrics instead of exact SQL component matching as in the test suite evaluation. EX denotes the fraction of questions within the evaluation set, where the outcomes of both the predicted and ground-truth queries yield identical results relative to the total number of queries.

	Train Set			Test Set		
Data Model	v1	v2	v3	v1	v2	v3
#Joins	1.68	2.23	1.45	1.78	2.63	1.45
#Projections	1.81	1.81	1.85	2.07	2.08	1.96
#Filters	1.95	2.20	1.73	2.15	2.39	1.85
#Aggregations	0.48	0.49	0.48	0.58	0.59	0.53
#Set Operations	0.14	0.14	0.00	0.17	0.19	0.00
#Subqueries	0.05	0.05	0.01	0.03	0.03	0.03
Mean Hardness	3.03	3.06	3.00	3.10	3.18	3.02
Mean Query Length	217	252	187	232	282	193

Table 3: Query characteristics: Mean values of train and test set used for training and evaluating. Hardness corresponds to the classification defined in [57]. Query length is defined as the number of characters per SQL query.

Text-to-SQL Systems. For our evaluation, we select five Text-to-SQL systems summarized in Table 4 that cover the design space presented in Section 2.2. Note that with new Text-to-SQL approaches being proposed on a monthly basis, we aim to select representative systems with the intention that our insights also apply to recent and prospective specialized approaches that might build on the language models and/or pre- and post-processing methods found in our selection.

Small Language Models. We use ValueNet [7] which improves upon IRNet [30]. ValueNet uses BART [37] as an encoder, which can be considered a first-generation, small language model (148M parameters). It also incorporates additional pre- and post-processing steps in its pipeline (see also description in Section 3.2).

Medium Language Models. We employ T5-Picard [48] as a representative second-generation model. T5-Picard comprises

sequence-to-sequence models [45] incorporating encoder and decoder components that are entirely based on transformer architectures, in contrast to the previous encoder-only language models. Picard is a method for constraining auto-regressive decoders of language models through incremental parsing to only valid SQL output sequences. Besides its original implementation, the method is used in four more Text-to-SQL systems currently found on the Spider leader board [38, 44, 58, 60]. Based on our results (see Section 6.2), we also develop one variation leveraging the base T5 model [45] and Picard: T5-Picard$_{Keys}$, which includes primary and foreign key constraints, not included in T5-Picard.

Large Language Models (LLMs). We conduct LLM experiments using OpenAI's gpt-3.5-turbo [42] and LLaMA2-70B [50], an open-source model from Meta. Currently, OpenAIs GPT models are used in the best six systems on the Spider leaderboard [17, 25, 43], while LLaMA2 is considered one of most widely used open-source models. We set the generation temperature to 0.0, frequency penalty to 0.0, and top-p to 1.0. Following the Text-to-SQL prompts proposed in [11, 46], we prepare various prompt templates for zero-shot and few-shot experiments by incorporating the DB schema including PK/FK key information. The prompt templates used in experiments for both models can be found in our code repository.

6.2 Impact of Design Dimensions

We perform a detailed evaluation of the various Text-to-SQL systems according to the following dimensions (1) Data Model, (2) Language Model, (3) Training Data Size and (4) Pre- and Post-processing. The overall results for fine-tuned models and LLMs are summarized in Table 5 and Table 6, respectively.

Data Model. How does the respective data model influence the execution accuracy of each Text-to-SQL system? For the traditional fine-tuned systems in Table 5 we see that ValueNet actually improves its performance from v1 to v3 by 5% points (for maximum training data size). However, T5-Picard exhibits little performance impact of the data model. Upon investigation, we find that the reason for this behavior is that T5-Picard does not include PK/FK relationships and thus cannot take full advantage of the simplified data model and hence the simpler queries. For verification, we create a new T5 base model using a different encoding scheme from the one proposed in [45]. The resulting T5-Picard$_{Keys}$ improves the performance greatly overall (up to 12% points for v3), but more importantly shows now an increase of 3% points from v1 to v3, supporting our hypothesis.

Though ValueNet benefits from the model development, T5-Picard with keys still surpasses it and even ties with GPT-3.5 for data model V1. In our dataset v1 and v2, we have kept samples that cannot be answered by ValueNet. Thus, our experiment aims for fairness across all systems. For the LLMs GPT-3.5 and LLaMA2-70B depicted in Table 6, we can observe no large differences between data models, with v1 showing the best overall results.

Language Model. From the perspective of LLMs, we can clearly see that GPT-3.5, the largest language model with 175 billion parameters, outperforms, LlaMA2-70B, with 70 billion parameters, across the different data models. This outcome aligns with expectations as LlaMA2-70B has significantly fewer parameters and its token limitation constrained the number of few shot examples.

T5-Picard$_{Keys}$ performs on par with GPT-3.5, both achieving 41% accuracy. This is noteworthy because of the disparity in the

---

[2]LLaMA2-70B has a limit of 4096 tokens.

Dimensions		ValueNet	T5-Picard	T5-Picard$_{Keys}$	GPT-3.5	LLaMA2-70B
Language Model	Scale (#Params)	small (148M)	medium (3B)	medium (3B)	large (175B)	large (70B)
	DB Schema w/ FK	Yes (with)	Yes (without)	Yes (with)	Yes (with)	Yes (with)
	DB Content	Yes	No	No	No	No
	Output Specification	IR	SQL	SQL	SQL	SQL
Pre-processing	Query Normalization	SQL-Parser	String Normalization	String Normalization	String Normalization	String Normalization
	Value Finder	Yes	No	No	No	No
	Conversion to IR	Yes	No	No	No	No
Post-processing		IR to SQL	Picard	Picard	N/A	N/A

Table 4: Characteristics of the Text-to-SQL systems in our experimental evaluation as well as the dimensions of the language models, pre-processing, and post-processing in different Text-to-SQL systems. FK = foreign key; IR = intermediate representation; String Normalization = set of string formatting operators which aim to remove tabs, line breaks, multiple consecutive spaces.

Data Model	Size Train Set	ValueNet	T5-Picard	T5-Picard$_{Keys}$
v1	zero	2.00%	8.00%	7.00%
	100	16.00%	22.00%	27.00%
	200	18.00%	29.00%	33.00%
	300	20.00%	29.00%	**38.00**%
v2	zero	3.00%	7.00%	7.00%
	100	14.00%	16.00%	29.00%
	200	18.00%	29.00%	33.00%
	300	20.00%	32.00%	**38.00**%
v3	zero	3.00%	6.00%	8.00%
	100	21.00%	6.00%	25.00%
	200	23.00%	27.00%	36.00%
	300	25.00%	29.00%	**41.00**%

Table 5: Execution accuracy of Text-to-SQL systems based on *small to medium-size language models*. We apply train set sizes ranging from 100 to 300 samples for each system/-data model combination.

Data Model	#Shots	GPT-3.5	#Shots	LLaMA2-70B
v1	zero	25%	zero	5.00%
	10	**41.00**%(±3.40%)	2	11.25%(±2.77%)
	20	39.00%(±4.08%)	4	10.50%(±3.35%)
	30	37.00%(±2.49%)	8	16.00%(±3.67%)
v2	zero	25%	zero	4.00%
	10	37.00%(±2.83%)	2	8.75%(±2.68%)
	20	36.00%(±1.63%)	4	8.50%(±1.50%)
	30	**37.50**%(±1.25%)	8	14.50%(±5.02%)
v3	zero	21%	zero	5.00%
	10	**38.50**%(±5.31%)	2	8.50%(±1.80%)
	20	37.00%(±2.49%)	4	8.50%(±3.20%)
	30	37.00%(±1.89%)	8	15.00%(±4.47%)

Table 6: Execution accuracy of Text-to-SQL systems based on *large language models*. For GPT-3.5, we used three different random samples of 10, 20, and 30 shots. Due to token size limits[2], for LLaMA2 we performed experiments on up to 8 random samples with multiple folds. For all systems, we report the mean accuracy and variance.

and especially the foreign keys are a crucial input for the model enabling it to understand the database schema of a given query. Overall, given that the highest execution accuracy score of any system is only 41%, the problem of translating natural language questions to SQL with large language models in a real-world application and real user queries is far from being solved.

Train Set Size. In terms of train set size, we observe that more training data typically leads to higher accuracy of the respective Text-to-SQL systems. For ValueNet, T5-Picard and T5-Picard$_{Keys}$, we used up to 300 training data samples. An important question in Text-to-SQL systems in practice is how much additional training data would actually improve their accuracy. Here, the trade-off is between manual annotation cost and the benefits of a better model. To further investigate this trade-off, we train ValueNet with a total of 895 clean training samples of the 1K labeled samples for data model v3. Note that we can only use 895 samples as the remaining 105 cannot be processed by the Spider SQL parser which is an essential part of the pre-processing component of ValueNet. For ValueNet, the accuracy increases from 25% to about 29%, i.e., tripling the amount of training samples shows an increase of 4% points. Hence, adding even more training data might not considerably increase these systems' accuracy.

For GPT-3.5, due to the token size limit, we provide a maximum of 30 random samples with each prompt. However, these 30 random samples are sufficient to perform as well as T5-Picard$_{Keys}$, which has a post-processing step, and has been fine-tuned on 300 samples. For a robust evaluation, we draw three different sample folds of up to 30 NL/SQL pairs for our few-shot experiments and report both the mean accuracy and the variance. Note the high variance, especially for the 10- and 20-shot experiments, indicates a high dependency on the specifically chosen samples.

However, adding more training data also requires manual labeling effort and more computing resources. We estimate that we used ≈ 1 person month only for manual query annotation. The training compute time of ValueNet for all experiments took 182 hours on a single nVidia v100-32GB GPU. Fine-tuning for T5 on 4 v100-32GB GPUs took 88.4 hours, which is equivalent to 354 hours on a single GPU. While GPT-3.5 does not require training, running our few-shot experiments cost about 100 US dollars. In summary, training with medium and large language models is both resource intensive but also has considerable monetary costs when performed on a large scale. Hence, deploying such systems in the real world often requires a trade-off between training data set size and the accuracy of the respective systems.

Pre- and Post-processing. The complexity of pre-processing depends on the number of input parameters of each language

number of parameters of each model, T5-Picard$_{Keys}$ only has 3 billion parameters. We also note the approximate 10% gap in the performance of the T5-Picard model (which does not use the database schema) indicating that the schema information,
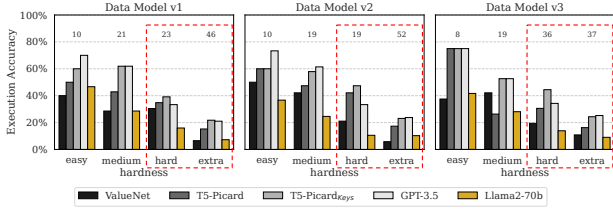
Figure 7: Execution accuracy of Text-to-SQL systems per Spider hardness level. The numbers on top of the bars represent the number of test set queries in each category.

model (see Table 4). For instance, ValueNet uses an intermediate representation, DB schema linking, and the database content as input. Hence, the pre-processing pipeline of ValueNet is more complex than the other Text-to-SQL systems, which only use conventional string normalization. Due to ValueNet's utilization of an intermediate representation layer as output, a designated post-processing step is essential for converting the intermediate representation into a structured SQL query. As we see from the performance of T5-Picard$_{Keys}$, adding FK information improves the performance of T5-Picard by 12% in the v3 version of the data model indicating that the limitations of language models can be overcome by more informative input encodings.

Pre-processing for LLMs is less relevant, because their comprehensive input prompts can include FK information and sample rows (as we demonstrate in our experimental evaluation). We leave the exploration of post-processing strategies LLMs to future work.

6.3 Impact of Query Complexity and Query Characteristics

We now investigate the impact of (1) the SQL query complexity and (2) certain query characteristics on the execution accuracy. For these experiments, we only use the models trained with the maximum training data size, i.e. 300 train set samples for ValueNet, T5-Picard and T5-Picard$_{Keys}$ as well as 30 shots for GPT-3.5 and 8 shots for LLaMA2-70B.

Query Complexity. First, we find that increasing Spider hardness reduces execution accuracy regardless of the Text-to-SQL system and data model. E.g., the accuracy for easy queries reaches up to 77%, while for extra hard queries the highest accuracy is slightly above 20% across all data models and systems (see Figure 7). We can directly *see the impact of the data model on the number of extra hard queries*, which change from 46 (v1) over 52 (v2) to only 36 in v3 (see red dashed lines). We find that the *hardness of these queries reduces from extra hard to hard (36 in v3)* with no impact on the execution accuracy for hard queries.

Query Characteristics. To further investigate the impact of query complexity, we also evaluate the accuracy of our defined query characteristics from Table 3, i.e. the number of *joins*, *projections filters*, etc. (see Figure 8). Except for projections and aggregations, we observe some interesting variances in accuracy between the data models for certain query characteristics:

Filters. Here the number of queries with only one filter reduces from 32 in v1 to 24 in v3, while the accuracy for these queries reduces slightly across all Text-to-SQL systems. Contrary to that, the number of queries with ≥ 2 filters increases from 52 to 66, while the accuracy of these queries also increases across all systems. This indicates that filters are intuitive in Text-to-SQL, i.e.

for a column like "winner" the model expects a Boolean value, (see Listing 1).

```
SELECT count(*)
FROM world_cup AS T1
JOIN national_team AS T2 ON T1.winner = T2.team_id
WHERE T2.teamname = 'England';
SELECT count(*)
FROM world_cup_result AS T1
JOIN national_team AS T2 ON T1.team_id = T2.team_id
WHERE T2.teamname = 'England' and T1.winner = 'True';
```

Listing 1: Query for "How many times did England win the worldcup?" (top for v1, bottom for v3)

Joins. We see most ≥ 2 joins for data model v2, while v3 requires the least. Investigating the affected queries, we observe that they only require a single join in v1, which is why the number of single join queries increases from v2 to v3. Unfortunately, these single-join queries are still hard for most Text-to-SQL systems. Only ValueNet and Llama2 can sustain their accuracy from v2 to v3.

Sets. The maybe most impactful effect of our data model changes can be observed from the number of set queries. Set operations, which showed poor performance across all evaluated systems, are no longer needed in the v3 data model, improving overall accuracy.

Overall, we observe that different data models can change query characteristics in various ways. Towards an explicit Text-to-SQL data model design strategy, *emphasis should be especially put on characteristics that are only poorly supported, such as set operations.* If these can be avoided in a different data model design, the overall performance of Text-to-SQL system can improve without additional training data or more advanced language models.
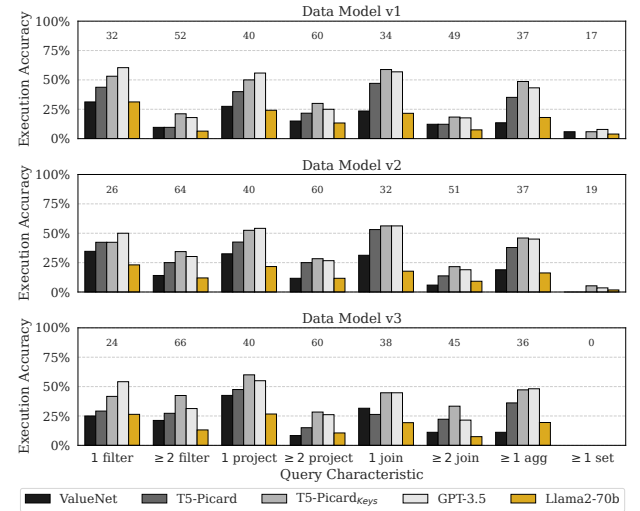

Figure 8: Execution accuracy per query characteristic. The numbers on top of the bars represent the number of test set queries in each category.

6.4 Inference Time of Language Models

Table 7 shows the mean and standard deviation of the inference time of each Text-to-SQL system in seconds per query, including the hardware specifications for each of the fine-tuned models and the open source, LlaMA2-70B. For ValueNet and all of the T5 models we used Tesla v100-SXM2-32GB GPUs. LlaMA2-70B

requires at least 35GB of GPU memory with 8bit quantizing[15], necessitating the use of the more powerful Tesla A100-SXM4-40GB GPU.

We see that for ValueNet, which is based on a small language model with 148M parameters, the mean inference time per query is about 1 sec. We note that for T5-Picard and T5-Picard$_{Keys}$, which have 3B parameters, the mean inference time per query is 652 and 294 seconds per query, respectively. In other words, *it takes more than 10 minutes (!) for T5-Picard to translate an NL question to a SQL query*. Such a response time is impractical for usage in a real-world scenario without considerable hardware ramp-up or substantial performance optimizations.

For GPT-3.5 the mean inference time per query is 2.51 seconds. However, note that GPT-3.5 is run on a massive hardware infrastructure in the cloud under the control of OpenAI.

Finally, the mean response time per query for LLaMA2-70B is 37.03 seconds. It is noteworthy to consider that this model requires a significantly more powerful hardware setup of 4 A100 GPUs compared to the setup of ValueNet and T5-Picard with only 1 v100 GPU. In summary, LLaMA2-70B's inference time of more than half a minute per query to translate Text-to-SQL cannot be used practically without a considerable hardware investment.

	ValueNet	T5-Picard	T5-Picard$_{Keys}$	GPT-3.5	LLaMA2-70B
Time (sec)	1.06 ±0.14	652.16 ±165.94	294.08 ±75.65	2.51 ±1.06	37.03 ±17.30
Hardware	v100	v100	v100	-	A100
# GPUs	1	1	1	-	4

Table 7: Mean and standard deviation of the inference unit time of each Text-to-SQL system in seconds per query.

7 DISCUSSION AND LESSONS LEARNED

From our in-depth design space evaluation, we learned several important lessons that can guide designs of Text-to-SQL systems.
Data Models Matter – Sometimes For the small- and medium-sized models, our results underscore the *significance of the keys' information, i.e., using the primary keys and foreign keys as input to the language model* to take advantage of the characteristics of the underlying data model. The results from T5-Picard and T5-Picard$_{Keys}$ show that including keys' information substantially enhances the execution accuracy across all data models in all experiments with various training data sizes. This enhancement is more significant on optimized data models with smaller training data sizes, yielding an improvement of up to 19% (see Table 5).

Although ValueNet and T5-Picard$_{Keys}$ distinguish each other in many specifications, e.g., model architectures or model sizes, they share a common requirement for keys' information as input and a common step of post-processing before prediction. Hence, both models clearly manifest the benefits from data model optimization with each training data size. In summary our results show that *LLMs are robust to the changes and design space parameters* we evaluated while small and medium sized language models are not.
Model Inference Time Matters. Even though T5-Picard$_{Keys}$, which uses a medium-size language model of 3B parameters, has a comparable execution accuracy to GPT-3.5, it's *practical usage is currently limited given an inference time of more than 5 minutes per query*. In order to use that system in practice, a considerable hardware investment is required.
Evaluating Text-to-SQL in Practice. We discover several challenges wrt evaluating Text-to-SQL systems with real user queries

in a practical application. Some of these, such as the ambiguity of natural language or the semantic mismatch between user request and database content, have also been recently discussed in [20]. This also related to the question of developing a good evaluation methodology, with important decisions such as train/test split [19]. Here, our diversity-based sampling strategy results in a challenging dataset, where even the best method achieves only 41%).
Limitations. A key reason for the changes in the schemas in v1, v2, and v3 was to facilitate query writing. As seen in Figure 4, the SQL queries for v1 and v2 are very long and difficult to write manually, while the v3 queries are much more intuitive and easier to write. While we adapted the schema design based on the questions submitted, the impact of changing the database schema could be further explored from the natural language point of view by employing lexical variation, leading to better alignment between the natural language questions and the database schema.

A possible limitation of the evaluation's validity comes from the lack of full independence between the choice of the natural language queries used in our evaluation and the three data models. First, some of the queries submitted to FootballDB were themselves used to adjust the schema in producing data models v2 and v3. Second, we used the Spider hardness of queries in data model v3 to sample our train and test sets for data models v1 and v2, influencing the distribution of queries by the query hardness in v3. Finally, applying experiments to a single database for all insights into data model refactoring might be insufficient to demonstrate the generalizability of the impact of data model optimization across diverse databases. Additional experiments on more databases are necessary to substantiate the applicability of these optimizations and draw more general conclusions.

8 RELATED WORK

Throughout the paper we have mentioned various Text-to-SQL systems. For a general overview we refer the reader to [1, 26, 33]. To the best of our knowledge there has not been any practical exploration of Text-to-SQL systems such as we presented in this paper. However, there have been several community driven datasets and benchmarks that have been crucial to the advancement in Text-to-SQL systems that can be classified into cross-domain and domain-specific:

WikiSQL [62] and Spider [57] have been two of the most widely used cross-domain benchmark datasets and helped to show the potential of deep learning for Text-to-SQL. ScienceBenchmark [59] contains databases from astrophysics, bioinformatics and policy research. The main challenge of this benchmark is the complexity of the domains which require expert knowledge. BIRD [39] includes 95 databases across 37 domains created via crowdsourcing. BIRD also introduces a new evaluation metric called Valid Efficiency Score.

Also, domain-specific datasets have been created in various domains, encompassing topics such as entertainment [14, 55], consumer reviews [55], medical data [54], and community-generated Q&A data [31]. These datasets provide a valuable play-ground for building domain specific Text-to-SQL systems.

Compared to above works, the unique value of FootballDB is that its data is based on a *months-long live deployment with ≈6K real user questions*. Moreover, we developed three data models for our dataset, for which we manually annotated 1,200 NL/SQL pairs. The effect of various data models and different sizes of

training data on the accuracy of Text-to-SQL systems has not been studied before to our knowledge. Table 8 summarizes the main relevant existing Text-to-SQL benchmark datasets. FootballDB is the only dataset that allows a multi-schema evaluation, it is the only dataset based on real-user questions posed to a live system, and exhibits the most tokens per query (an indication of query complexity).

Dataset	#Examples (#DBs)	#Tables (#Rows)/DB	#Tokens/Query	Multi-Schema	Live Users
WikiSQL[63]	80,654 (26,521)	1 (17)	12.2	✗	✗
SPIDER [57]	10,181 (200)	5.1 (2K)	18.5	✗	✗
KaggleDBQA [35]	272 (8)	2.3 (280K)	13.8	✗	✗
ScienceBench. [59]	5,332 (3)	16.7 (51M)	15.6	✗	(✓)
BIRD [40]	12,751 (95)	7.3 (549K)	30.9	✗	✗
FootballDB	1200 (3)	15 (107K)	33.7	✓	✓

Table 8: Comparison between FootballDB and existing Text-to-SQL datasets. `Multi-Schema`: multiple schemas, same data. `Live Users`: real user questions against live systems.

9 CONCLUSION

This paper provides essential insight into implementing Text-to-SQL systems in the real world including an in depth analysis of the impact of different data models.

First, we report on *over nine months of deployment with several hundred users*, the issues we faced, and the design iterations implemented to address them, providing a unique perspective for Text-to-SQL practitioners from industry and academia.

Second, we take our practical deployment as an opportunity to define and *evaluate critical design space dimensions of Text-to-SQL systems*, namely (1) Data Model, (2) Language Model, (3) Training Data Size and (4) Pre-/Post-processing. We pay particular attention to the data model impact which is a previously understudied subject. Our experimental results show that the choice of data model has a substantial impact on the accuracy of Text-to-SQL systems when using small or medium-sized language models. Our experiments also demonstrate that medium-sized and large language models show the highest execution accuracy (T5-Picard$_{Keys}$ and GPT-3.5: 41%) with little labeled data (10 few shot examples for GPT-3.5).

Third, an important aspect that has not been given much attention yet is the *inference time of Text-to-SQL systems*. Our experiments show that the mean inference time for T5-Picard$_{Keys}$ and LLaMA2-70B deployed on a reasonable GPU hardware is 294 and 37 seconds per query, respectively. These high response times make it currently impractical to use these systems in real-world scenarios where interactive query speeds of less than 3 seconds are expected. Reducing the maximum input token size has the potential to meet the inference time requirements. Nevertheless, this reduction can be accompanied by a significant decrease in the execution accuracy of the predicted SQL query due to the lossy input information. Hence, considerable hardware investment or algorithmic optimizations would be required to make these systems practical.

Last, we *release FootballDB, a new benchmark dataset* based on over 6K user queries with 1200 manual annotated NL-SQL pairs for three distinct data models. We aim to extend FootballDB with a hidden test dataset and release a public benchmark in the same vein as the Spider [57] and BIRD [39] benchmarks to support other researchers in developing and testing their systems on an internationally well-known and easily understandable domain.

REFERENCES

[1] Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. 2019. A comparative survey of recent natural language interfaces for databases. *The VLDB Journal* 28 (2019), 793–819.

[2] Alireza Ahadi, Julia Prior, Vahid Behbood, and Raymond Lister. 2015. A quantitative study of the relative difficulty for novices of writing seven different types of SQL queries. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. 201–206.

[3] Sihem Amer-Yahia, Georgia Koutrika, Martin Braschler, Diego Calvanese, Davide Lanti, Hendrik Lücke-Tieke, Alessandro Mosca, Tarcisio Mendes de Farias, Dimitris Papadopoulos, Yogendra Patil, et al. 2022. INODE: building an end-to-end data exploration system in practice. *ACM SIGMOD Record* 50, 4 (2022), 23–29.

[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *international semantic web conference*. Springer, 722–735.

[5] Andre Becklas. 2018. FIFA World Cup: All the results from World Cups. Retrieved 2023-06-26 from https://www.kaggle.com/datasets/abecklas/fifa-world-cup

[6] Lukas Blunschi, Claudio Jossen, Donald Kossmann, Magdalini Mori, and Kurt Stockinger. 2012. SODA: Generating SQL for Business Users. *Proceedings of the VLDB Endowment* 5, 10 (2012).

[7] 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)*. IEEE, 2177–2182.

[8] Hanswalter Buff. 2003. *Datenbanktheorie*. Books on Demand, Norderstedt.

[9] Surajit Chaudhuri and Umeshwar Dayal. 1997. An overview of data warehousing and OLAP technology. *ACM Sigmod record* 26, 1 (1997), 65–74.

[10] Peter Pin-Shan Chen. 1976. The entity-relationship model—toward a unified view of data. *ACM transactions on database systems (TODS)* 1, 1 (1976), 9–36.

[11] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128* (2023).

[12] Bin Cheng, Jonathan Fürst, Tobias Jacobs, and Celia Garrido-Hidalgo. 2024. Interactive Ontology Matching with Cost-Efficient Learning. *arXiv preprint arXiv:2404.07663* (2024).

[13] EF CODD. 1972. Further Normalization of the Data Base Relational Model. *Data Base Systems* (1972), 33–64.

[14] 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*. 897–911.

[15] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2022. 8-bit Optimizers via Block-wise Quantization. *9th International Conference on Learning Representations, ICLR* (2022).

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[17] Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, Jinshu Lin, Dongfang Lou, et al. 2023. C3: Zero-shot Text-to-SQL with ChatGPT. *arXiv preprint arXiv:2307.07306* (2023).

[18] Zhen Dong, Shizhao Sun, Hongzhi Liu, Jian-Guang Lou, and Dongmei Zhang. 2019. Data-Anonymous Encoding for Text-to-SQL Generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 5405–5414. https://doi.org/10.18653/v1/D19-1543

[19] Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving Text-to-SQL Evaluation Methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 351–360.

[20] 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 *Proceedings of the 24th Conference on Innovative Data Systems Research (CIDR24)*.

[21] Jonathan Fürst, Mauricio Fadel Argerich, and Bin Cheng. 2023. VersaMatch: Ontology Matching with Weak Supervision. *Proceedings of the VLDB Endowment* 16, 6 (2023), 1305–1318.

[22] Jonathan Fürst, Mauricio Fadel Argerich, Bin Cheng, and Ernö Kovacs. 2020. Towards knowledge infusion for robust and transferable machine learning in IoT. *Open Journal of Internet of Things (OJIOT)* 6, 1 (2020), 24–34.

[23] Jonathan Fürst, Bin Cheng, and Benjamin Hebgen. 2021. Realizing the digital twin transition for smart cities. *Open Journal of Internet Of Things (OJIOT)* 7, 1 (2021), 32–42.

[24] Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R. Woodward, John Drake, and Qiaofu Zhang. 2021. Natural SQL: Making SQL Easier to Infer from Natural Language Specifications. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics, Punta Cana, Dominican Republic, 2030–2042. https://doi.org/10.18653/v1/2021.findings-emnlp.174

[25] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. *arXiv preprint arXiv:2308.15363* (2023).

[26] Orest Gkini, Theofilos Belmpas, Georgia Koutrika, and Yannis Ioannidis. 2021. An in-depth benchmarking of text-to-sql systems. In *Proceedings of the 2021 International Conference on Management of Data*. 632–644.

[27] Google. 2023. Google Bard. Retrieved 2023-06-01 from https://bard.google.com/

[28] Maarten Grootendorst. 2022. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794* (2022).

[29] Zihui Gu, Ju Fan, Nan Tang, Lei Cao, Bowen Jia, Sam Madden, and Xiaoyong Du. 2023. Few-shot Text-to-SQL Translation using Structure and Content Prompt Learning. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–28.

[30] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4524–4535.

[31] Moshe Hazoom, Vibhor Malik, and Ben Bogin. 2021. Text-to-SQL in the Wild: A Naturally-Occurring Dataset Based on Stack Exchange Data. In *Proceedings of the 1st Workshop on Natural Language Processing for Programming (NLP4Prog 2021)*. Association for Computational Linguistics, Online, 77–87. https://doi.org/10.18653/v1/2021.nlp4prog-1.9

[32] George Katsogiannis-Meimarakis and Georgia Koutrika. 2023. A survey on deep learning approaches for text-to-SQL. *The VLDB Journal* (2023), 1–32.

[33] Hyeonji Kim, Byeong-Hoon So, Wook-Shin Han, and Hongrae Lee. 2020. Natural language to SQL: Where are we today? *Proceedings of the VLDB Endowment* 13, 10 (2020), 1737–1750.

[34] 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)*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, Online, 2261–2273. https://doi.org/10.18653/v1/2021.acl-long.176

[35] 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)*. Association for Computational Linguistics, Online, 2261–2273. https://doi.org/10.18653/v1/2021.acl-long.176

[36] Claude Lehmann, Dennis Gehrig, Stefan Holdener, Carlo Saladin, João Pedro Monteiro, and Kurt Stockinger. 2022. Building Natural Language Interfaces for Databases in Practice. In *Proceedings of the 34th International Conference on Scientific and Statistical Database Management*. 1–4.

[37] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 7871–7880.

[38] Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. 2023. Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. *arXiv preprint arXiv:2301.07507* (2023).

[39] Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, et al. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *arXiv preprint arXiv:2305.03111* (2023).

[40] Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, 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. arXiv:cs.CL/2305.03111

[41] OpenAI. 2023. GPT-4 Technical Report. arXiv:cs.CL/2303.08774

[42] OpenAI. 2023. OpenAI ChatGPT. Retrieved 2023-06-01 from https://openai.com/blog/chatgpt

[43] Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. *arXiv preprint arXiv:2304.11015* (2023).

[44] Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Yu Cheng, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. RASAT: Integrating Relational Structures into Pretrained Seq2Seq Model for Text-to-SQL. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 3215–3229.

[45] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

[46] Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498* (2022).

[47] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics.

[48] 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*. 9895–9901.

[49] Abraham Silberschatz, Henry Korth, and Shashank Sudarshan. 2005. *Database systems concepts*. McGraw-Hill, Inc.

[50] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[52] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[53] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. 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*. 7567–7578.

[54] Ping Wang, Tian Shi, and Chandan K Reddy. 2020. Text-to-SQL Generation for Question Answering on Electronic Medical Records. In *Proceedings of The Web Conference 2020*. 350–361.

[55] Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. SQLizer: Query Synthesis from Natural Language. *PACMPL*, Article 63 (2017), 26 pages.

[56] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018. TypeSQL: Knowledge-Based Type-Aware Neural Text-to-SQL Generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 588–594.

[57] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 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*. 3911–3921.

[58] Lu Zeng, Sree Hari Krishnan Parthasarathi, and Dilek Hakkani-Tur. 2023. N-best hypotheses reranking for text-to-sql systems. In *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 663–670.

[59] Yi Zhang, Jan Deriu, George Katsogiannis-Meimarakis, Catherine Kosten, Georgia Koutrika, and Kurt Stockinger. 2023. ScienceBenchmark: A Complex Real-World Benchmark for Evaluating Natural Language to SQL Systems. *arXiv preprint arXiv:2306.04743* (2023).

[60] Yiyun Zhao, Jiarong Jiang, Yiqun Hu, Wuwei Lan, Henry Zhu, Anuj Chauhan, Alexander Li, Lin Pan, Jun Wang, Chung-Wei Hang, et al. 2022. Importance of synthesizing high-quality data for text-to-sql parsing. *arXiv preprint arXiv:2212.08785* (2022).

[61] Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic Evaluation for Text-to-SQL with Distilled Test Suites. arXiv:cs.CL/2010.02840

[62] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR* abs/1709.00103 (2017).

[63] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. arXiv:cs.CL/1709.00103