ScienceBenchmark: A Complex Real-World Benchmark for Evaluating Natural Language to SQL Systems

Yi Zhang
Zurich University of Applied Sciences
Switzerland

Jan Deriu
Zurich University of Applied Sciences
Switzerland

George Katsogiannis-Meimarakis
Athena Research Center
Greece

Catherine Kosten
Zurich University of Applied Sciences
Switzerland

Georgia Koutrika
Athena Research Center
Greece

Kurt Stockinger
Zurich University of Applied Sciences
Switzerland

ABSTRACT

Natural Language to SQL systems (NL-to-SQL) have recently shown improved accuracy (exceeding 80%) for natural language to SQL query translation due to the emergence of transformer-based language models, and the popularity of the Spider benchmark. However, Spider mainly contains simple databases with few tables, columns, and entries, which do not reflect a realistic setting. Moreover, complex real-world databases with domain-specific content have little to no training data available in the form of NL/SQL-pairs leading to poor performance of existing NL-to-SQL systems.

In this paper, we introduce *ScienceBenchmark*, a new complex NL-to-SQL benchmark for three real-world, highly domain-specific databases. For this new benchmark, SQL experts and domain experts created high-quality NL/SQL-pairs for each domain. To garner more data, we extended the small amount of human-generated data with synthetic data generated using GPT-3. We show that our benchmark is highly challenging, as the top performing systems on Spider achieve a very low performance on our benchmark. Thus, the challenge is many-fold: creating NL-to-SQL systems for highly complex domains with a small amount of hand-made training data augmented with synthetic data. To our knowledge, *ScienceBenchmark* is the first NL-to-SQL benchmark designed with complex real-world scientific databases, containing challenging training and test data carefully validated by domain experts.

PVLDB Reference Format:
Yi Zhang, Jan Deriu, George Katsogiannis-Meimarakis, Catherine Kosten, Georgia Koutrika, and Kurt Stockinger. ScienceBenchmark: A Complex Real-World Benchmark for Evaluating Natural Language to SQL Systems. PVLDB, 17(4): 685-698, 2023.
doi:10.14778/3636218.3636225

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://sciencebenchmark.cloudlab.zhaw.ch/.

Proceedings of the VLDB Endowment, Vol. 17, No. 4 ISSN 2150-8097.
doi:10.14778/3636218.3636225

1 INTRODUCTION

Enabling users to query structured data using natural language is considered the key to data democratization. Natural Language Interfaces for Databases (or NL-to-SQL systems) emerged in the 1970s [1, 4]. Early systems relied on the database schema to build a SQL query from a natural language (NL) query (e.g., SODA [5], Precis [40]) or focused on understanding the structure of the natural language query to map it to SQL (e.g., ATHENA [37], NaLIR [24]). As early as 1995, the lack of benchmarks was apparent: "No standard benchmarks have yet been developed [...], any appraisal of the current state of the field must be impressionistic" [4]. This situation changed recently, when the first large-scale benchmarks, WikiSQL [53] and Spider [51], emerged. These allowed for training and evaluating *neural machine translation (NMT) approaches* (e.g., [44, 48, 53]). These approaches formulate the NL-to-SQL problem as a language translation problem, and train neural networks with large amounts of NL/SQL-pairs.

While the first deep learning approaches [48, 53] only worked for single tables and failed to generate complex SQL queries spanning multiple tables (e.g., including nested queries and complex clauses), recent systems [7, 8, 38, 44] work on complete databases and achieve high performance scores on the Spider benchmark [51]. The top NL-to-SQL systems reach accuracies up to 85% on Spider. However, the majority of databases present in Spider were created specifically for this benchmark and are not representative of the difficulties that arise when creating an NL interface for a real-world database. Among the current best-performing, open-source, systems on Spider, T5-Large [32] (with Picard [38] for constrained decoding), SmBoP [36] and RESDSQL with NatSQL [25], are already achieving over 70% performance. Applying a system trained on the Spider dataset to a new domain such as astrophysics or cancer research, yields poor results, making the adoption of such systems to real-life applications extremely far-fetched.

The problem is that, while WikiSQL and Spider provide a common training and evaluation tool that has been a game changer for the development of NL-to-SQL systems, they have serious limitations. Each WikiSQL question is directed to a single table and not to a relational database. The low complexity of the WikiSQL queries makes its value for real-life applications practically limited. Spider contains 200 relational databases from 138 different domains along with over 10,000 natural language questions and over 5,000 complex

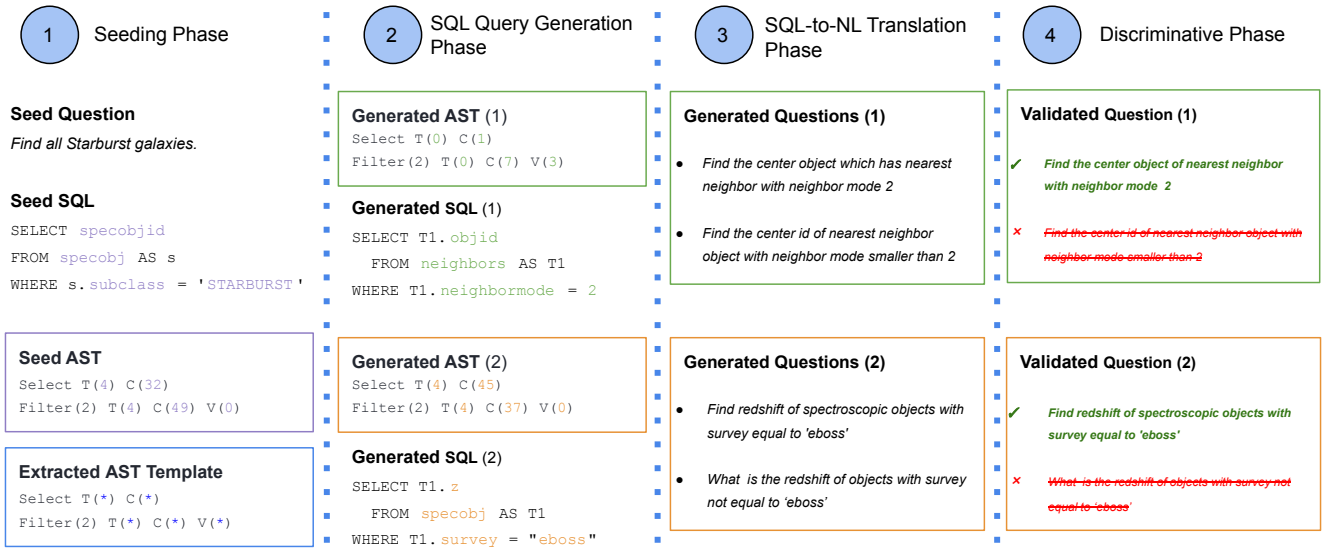

Figure 1: End-to-end architecture for automatic training data generation consisting of four different phases, namely (1) Seeding Phase, (2) SQL Generation Phase, (3) SQL-to-NL Translation Phase and (4) Discriminative Phase. The approach is used to produce our novel benchmark dataset called ScienceBenchmark. The details of these phases are described in Section 3.3.

SQL queries, hence it targets the query complexity problem. At the same time, it also attempts to solve the transfer learning problem by providing a dev set with novel databases.

However, the majority of databases in the Spider benchmark *are rather simple*, i.e., of low complexity, and containing only what is considered general knowledge. Thus, we need specialized, domain-specific benchmarks for training and evaluating NL-to-SQL systems in scientific domains. Manually crafting such a benchmark is prohibitive due to the volume of data needed and the expertise required in scientific domains. *Data augmentation*, i.e. automatic benchmark generation is the only feasible solution.

Our Approach. In this paper, we introduce *ScienceBenchmark*, a complex real-world benchmark for evaluating NL-to-SQL systems. It is the first of its kind to be developed collaboratively with SQL experts and researchers from the fields of research policy making, astrophysics, and cancer research. We *combine* the knowledge derived from *manual training data collection with an automatic training data generation system* that enables NL-to-SQL systems that require large amounts of data for training to be bootstrapped on complex, scientific databases where training data would otherwise be scarce or unavailable.

The main architecture of the data augmentation system is described in Figure 1. The system is fed a *small set* of manually generated NL/SQL-pairs to provide accurate and highly semantically relevant information about a novel domain. *SQL templates are extracted* from these seed queries, and are subsequently used to generate SQL queries over a specific database taking into account the database schema and contents. These SQL queries are then *back-translated to natural language* using the large language model (LLM), GPT-3 [6]. For the SQL-to-NL component used in this work, we experimented with state-of-the-art transformer-based pre-trained models that have shown their NL generation capabilities by achieving state-of-the-art scores in multiple related tasks. Our evaluation showed that the GPT-3 [6] model achieves the best performance and is able to generalize to new domains with very few samples (in our case, these are the seed NL/SQL-pairs), which is why it was integrated in our architecture. The resulting natural language questions are then filtered using a *critic model* to select the most relevant NL question for the corresponding SQL query. The resulting augmented dataset of *NL/SQL-pairs* can be fed into any NL-to-SQL system for training. Our approach is generic and can boost the accuracy of any NL-to-SQL system as demonstrated in our experiments in Table 5.

Contributions. The major contributions of our work are the following:

- We contribute *ScienceBenchmark - a new benchmark for evaluating NL-to-SQL systems against complex, scientific databases. ScienceBenchmark* contains more than 6,000 NL/SQL-pairs to help researchers address the complex challenges of real-world databases, overlooked by popular benchmarks.
- We have built *ScienceBenchmark* using a *novel approach for automatically generating training data for databases* where little to no training data exists. Unlike previous work that focused on rather simple databases, we *concentrate on complex, real-world scientific databases*, an area where popular NL-to-SQL systems typically falter due to their lack of domain knowledge and training data.
- We evaluated three state-of-the-art NL-to-SQL systems as well as two LLMs on our benchmark. Although these systems achieve top scores on the Spider leaderboard (above 82% accuracy), none achieves a satisfactory score on our benchmark (only in the range of 25-60% accuracy depending on the domain), showcasing the difficulty of *ScienceBenchmark*.

2 THE NEED FOR A NEW BENCHMARK - MOTIVATING EXAMPLE: ASTROPHYSICS

A NL-to-SQL system needs to address many challenges [1, 3, 18, 20]. On the one hand, a natural language question may be vague, contain references that are even hard for a human to understand, and use a different vocabulary from the one used in the DB. On the other hand, the respective SQL query needs to adhere to a strict syntax and to the underlying DB schema in order to be syntactically and semantically correct. When applying an NL-to-SQL system to a real-world scientific database, additional challenges arise that stem from the nature and the domain of these databases. While the Spider benchmark is the first large-scale dataset with complex SQL queries, its databases cannot be considered complex. Their subject-matter is very generalized, covering topics such as pets and entertainment (concerts, orchestras, musicals etc.). The majority of these databases were created by students specifically for Spider and are not representative of real-world databases.

In what follows, we motivate the need for a novel design and training of NL-to-SQL systems for complex, scientific databases. We will focus our discussion on astrophysics, a very data-intensive and highly complex scientific discipline with a long tradition of using relational databases [41]. The challenges described here are not only relevant for astrophysics but also for other scientific disciplines, such as cancer research, also included in our experimental evaluation.

As our running example, we use the astrophysics database called Sloan Digital Sky Survey (SDSS)[1]. This database stores information about stars and galaxies at specific locations in the sky. Further details on the complexity of this database are specified in Section 3.

Let us consider three different representative astrophysics queries that serve as running examples for our paper.

- *Q1: Find all Starburst galaxies.*
- *Q2: What is the object id, right ascension, declination and redshift from spectroscopically observed galaxies with redshift greater than 0.5 but less than 1?*
- *Q3: Find the photometric objects with object ids and spectroscopic object id whose spectroscopic class is 'GALAXY', with the difference of magnitude u and magnitude r less than 2.22 and the difference of magnitude u and magnitude r greater than 1.*

Their corresponding SQL queries are as follows:

```
Q1: (Spider hardness: Easy)
SELECT s.specobjid
FROM specobj AS s
WHERE s.subclass = 'STARBURST'

Q2: (Spider hardness: Medium)
SELECT s.bestobjid, s.ra, s.dec, s.z
FROM specobj AS s
WHERE s.class = 'GALAXY' AND s.z > 0.5 AND s.z < 1

Q3: (Spider hardness: Extra hard)
SELECT p.objid, s.specobjid
FROM photoobj AS p
  JOIN specobj AS s ON s.bestobjid = p.objid
WHERE s.class = 'GALAXY'
  AND p.u - p.r < 2.22 AND p.u - p.r > 1
```

As shown by these queries, the major challenges for NL-to-SQL systems for complex, scientific databases are as follows:

- *Unseen domains*: To understand a *challenging domain* like astrophysics and write meaningful queries - in natural language and SQL - extensive domain knowledge is required. Hence, neural machine translation systems pre-trained on common knowledge datasets, like Spider, typically fail in complex domains due to the large disparity in subject matter.
- *Complex, often cryptic, database schemas*: Scientific databases often store large amounts of data with *hundreds of attributes*. Moreover, attributes may have extremely short names, such *ra* or *z* (referring to right ascension and redshift, respectively, in astrophysics) or store numerical measurements. Hence, additional ontologies that describe the meaning or the scientific interpretation of these attributes - which can be complex astrophysics descriptions including mathematical equations and natural language texts - are required. Finally, learning the mapping of a token from a natural language question to the relevant database attribute is non-trivial.
- *Sophisticated queries*: Domain-specific queries may be more elaborate than the ones in Spider. For instance, astrophysics analysis requires the use of *functions* and *mathematical operators* between attributes, such as the difference in magnitude between ultraviolet (u) and infrared filters (r), e.g., $u - r < 2.22$.

These observations expose the need for specialized benchmarks designed to capture the particularities and semantics of the domain at hand as well as the types of queries that need to be understood by an NL-to-SQL system. These requirements, in combination with the size that such a benchmark necessitates, prohibit its manual construction.

Since large and complex schemas are hard (even for experts) to understand and query, the question that naturally arises is *how to build such benchmarks?* The answer is data augmentation, which in turn brings its own challenges: *How to build representative SQL queries for a new database? How to come up with their NL descriptions? How to do it with minimal, if any, human involvement?*

3 SCIENCE BENCHMARK: A NEW BENCHMARK FOR COMPLEX DATABASES

In this section, we present *ScienceBenchmark*, which is composed of three domain-specific databases, namely research policy making, astrophysics and cancer research. First, we introduce the databases, showing their complexity and the characteristics of each domain. Second, we describe the manual data collection, which includes SQL experts writing queries with the involvement of domain experts as part of multi-year research project called INODE - Intelligent Open Data Exploration [2]. Third, we describe our automatic data augmentation approach for generating synthetic training data using GPT-3 [6]. Finally, we show the training and evaluation datasets of our novel benchmark *ScienceBenchmark*.

3.1 Complex, Real-World Databases

Here, we introduce complex, real-world databases from three scientific domains, which are all of significantly greater size and complexity than the databases found in Spider [51]. Table 1 provides an overview of the complexity of the Spider, KaggleDBQA[21], BIRD[26] and ScienceBenchmark databases. The table shows that

[1] https://www.sdss.org/

Table 1: Complexity of the Spider, KaggleDBQA and BIRD databases compared with the databases of our new benchmark dataset *ScienceBenchmark*. The table shows the number of databases (DBs), tables, columns, rows per DB, average number of rows per table and DB size. The scientific domains of the databases contained in the ScienceBenchmark are *research policy making* (CORDIS), *astrophysics* (SDSS) and *cancer research* (OncoMX).

Dataset	DBs	Tbls.	Cols.	Rows	Avg. R/T	Size (GB)	#NL/SQL (man.+syn.)
Spider	186	641	4,268	1.6M	2.5K	0.51	8,053 + 0
(Avg / DB)		3.5	23	8.6K		0.03	43 + 0
KaggleDBQA	8	17	179	4.7M	280K	0.4	272 + 0
(Avg / DB)		2.1	22.3	595K		0.05	34 + 0
BIRD	81	604	4,456	3.7B	608.8K	33.4	12,751 + 0
(Avg / DB)		7.5	55	4.5M		0.35	135 + 0
Science-Benchmark							
CORDIS	1	19	82	671K	35K	1	200 + 1306
SDSS	1	6	61	86M	14M	6.1	200 + 2061
OncoMX	1	25	106	65M	2.6M	12	200 + 1065

BIRD is significantly larger and more complex than KaggleDBQA (i.e. number of tables, columns and rows). We also show that BIRD has a larger number of cross-domain databases than ScienceBenchmark. However, the complexity of each individual database of ScienceBenchmark is much higher than BIRD in the number of tables and size per database. In this sense, BIRD and ScienceBenchmark are complementary.

Research Policy Making: The CORDIS database, i.e. *Community Research and Development Information Service*[2], serves as the European Commission's primary source of results from the projects funded by the EU's framework programs for research and innovation. The database contains very detailed hierarchical information about the framework of funding calls and the network of industrial and academic institutions, all of which is coded in highly specific enigmatic EU terminology.

An example of this is the acronym NUTS, which stands for *nomenclature of territorial units for statistics*. Even the long form does not necessarily give the casual user a clear idea of what kind of information might be stored in such a column. Another challenging aspect of this database is the amount of text (e.g. descriptive project objectives averaging 1,821 characters) and the diversity of topics in the database ranging from *Information and Media* to *Nuclear Fission*. For *ScienceBenchmark*, we use version 2022-08 of the database, as shown in Table 1, which comprises 19 tables and 82 columns and has an average of 35,355 rows per table. The database size is 1 GB.

Astrophysics: The SDSS (*Sloan Digital Sky Survey*) – introduced in Section 2 – is a database containing the most detailed three-dimensional map of the universe ever made. The data collection began in 2000 and continues today. The database has 10 tables that contain disparate numbers of columns varying between 3 and 804 columns each. The tables contain various measurements and information about the type of observed object (e.g. a star or a galaxy), distances between observed objects, and various parameters (e.g. right ascension, declination, photometric system filters (u, g, r, i and z)) that have been measured in photometric or spectroscopic observations.

In order to study star-forming galaxies, the sky is measured in several color bands such as infrared or ultraviolet resulting in different spectra, i.e. alternative numerical measurements and thus various interpretations of galaxies. Unlike on Earth, the location of celestial objects is defined by their right ascension and declination. Moreover, literally, hundreds of different attributes are collected for each object, such as size, redshift, brightness, magnitudes of color bands measurements, etc. This database contains many column names and values that are labeled with abbreviations (rather than natural language) that are familiar to astrophysicists but indecipherable for non-specialists. In order to query this database with an NL-to-SQL system, it was necessary to add natural language labels for abbreviated columns (e.g. ra = right ascension). In addition to the lack of natural language labels for columns, much of the information in the database is numerical and is used in complex queries with mathematical operations. Querying numerical data in natural language is significantly more complex than querying a database comprised of mostly textual values.

Due to the limitation of the input tokens of the language model used in our experiments, we use a subset of the database comprised of 5 original tables and 1 additional table for photometrically observed astronomical objects. As described in Table 1, there are 61 columns, averaging about 10 columns per table and an average of 14,462,875 rows per table. The size of the database is 6.1 GB.

Cancer research data: OncoMX[3] is a database funded by the U.S. National Institute of Health (NIH) that integrates knowledge from several different sources about cancer biomarkers. The version of OncoMX used in *ScienceBenchmark* contains information from cancer biomarker databases (EDRN[4], FDA[5]), gene expressions in healthy anatomical entities (Bgee[6]), differential gene expressions between healthy and cancerous samples (BioXpress[7]) and cancer mutations (BioMuta[8]). As shown in Table 1, the database comprises 25 tables that have 2 to 14 columns each, for a total of 106 columns and has an average of 2,636,771 rows per table. The size of the database is 12 GB.

The complexity in this database lies in the heavily domain-specific information it contains as well as the complex queries that researchers use when exploring this database. For example, those unfamiliar with cancer research will not know that "BRCA1" or "BRCA2" refers to *BReast CAncer gene 1* and *BReast CAncer gene 2*, respectively. In addition to the domain-specific information, even a seemingly simple query in natural language such as "Show biomarkers for breast cancer" requires a SQL query with a multi-relational join and several filters.

[2]https://cordis.europa.eu/

[3]https://www.oncomx.org/
[4]https://edrn.nci.nih.gov/
[5]https://www.fda.gov/
[6]https://bgee.org/
[7]https://bioxpressbiosimilars.com/
[8]https://hive.biochemistry.gwu.edu/biomuta/norecord

3.2 Manual Data Collection

In this section, we detail the manual data collection for all three databases. All of the data was generated and reviewed by an expert group consisting of at least one SQL expert and one domain expert in research policy-making, astrophysics, or cancer research. In total, the team consisted of about 20 domain and SQL experts of various age ranges and genders. All of the experts were members of the multi-year research project INODE [2], including partners from academia and industry.

Before starting data collection, the domain experts such as astrophysicists and cancer researchers introduced the SQL experts to the domain-specific knowledge within the database. At the data generation stage, the domain experts developed the *natural language questions*, while the SQL experts were responsible for writing the corresponding *SQL queries*.

It is important to note that the domain experts were given the task of solving realistic science questions rather than simply generating complex questions based on the database. During the review and validation phase, domain experts used their expertise to verify the SQL queries together with the output of the SQL queries. For each domain, we generated a *training set* of 100 NL/SQL pairs as well as a *test set* of 100 NL/SQL pairs.

In contrast, for each database in the Spider dev set, there are far fewer questions, 50 on average per database, ranging from 63 to just 4 questions per database. We have double the amount of dev set queries for our databases than in Spider.

3.3 Automatic NL-to-SQL Training Data Generation

In this section, we present our automatic training data generation approach. This approach is generalizable and can be applied to any domain. We will use our running example for astrophysics introduced in Section 2. A concrete example of the end-to-end pipeline is depicted in Figure 1.

The process of automatic training data generation involves four phases: 1) the *Seeding Phase*, where SQL templates are extracted from the manually written seed queries, 2) the *SQL Query Generation Phase*, where the templates are filled with the database content, and schema are used to create a readable version of the query, 3) the *SQL-to-NL Translation Phase*, where GPT-3 generates a set of candidate questions, and 4) the *Discriminative Phase* (candidate selection phase) that selects the top two NL questions per SQL query.

3.3.1 Phase 1: Seeding Phase. The seeding phase ingests the manually created SQL queries (as discussed in Section 3.2) and extracts *query templates*. For this, the manually created queries are transformed into an *Abstract Syntax Tree* (AST) representation called *SemQL* [13]. The leaf nodes of the AST, i.e., tables (T), columns (C) and values (V) are replaced with placeholders (denoted as (*) in Figure 1). The resulting AST is used as a *query template*, which is filled with database content in the next phase.

3.3.2 Phase 2: SQL Query Generation Phase. When populating the query templates, the usage of randomly sampled tables and columns without any constraints might lead to meaningless or unrunnable queries. In order to ensure that the generated SQL queries are meaningful and useful to researchers in each field, we automatically create an *enhanced schema*. The enhanced schema can also be refined manually by domain experts to offer more meta-information about tables and columns. The manual work, if needed, is one-shot. This enhanced schema enables the exposure of the following information to the system.

- *Non-Aggregatable Columns:* These columns should not be allowed to appear in an attribute with an aggregation function like `SUM`, `AVG`, `MIN`, `MAX` because such an operation is not meaningful. An example below shows a query for getting an average of all IDs of spectroscopic objects, which is executable but not meaningful.

  ```
  SELECT AVG(s.specobjid) FROM specobj as s
  ```

- *Columns with Categorical Values:* Typical examples are *gender* or *number of languages spoken* by a person. These columns usually have a low cardinality and are more likely applied to a `GROUP BY` clause.

  The example below stands for *How many spectroscopic objects are there for each right ascension?* This question will hardly be asked by anyone knowing the basics of astrophysics. The SQL-statement may return millions of rows with useless information, because of the very high cardinality of right ascension.

  ```
  SELECT COUNT(*), s.ra
  FROM specobj as s GROUP BY s.ra
  ```

  With this constraint, the sampling procedure ensures that more meaningful queries are generated. The SQL-statement below retrieves the often asked question: *Find the count of spectroscopic objects grouped by their corresponding class.*

  ```
  SELECT COUNT(*), s.class
  FROM specobj as s GROUP BY s.class
  ```

- *Columns relevant for Applying Math Operators:* These columns are chosen by our sampling algorithm to apply certain math operators. Identifying these columns ensures that there will be no unexpected randomness among the operands, such as *T1.length - T2.area* – which is not meaningful.
- *Semantically Meaningful Table and Column Names.* Since complex, scientific databases often contain table and column names that are not easily interpretable for humans (e.g., *ra* stands for *right ascension*), we introduce human-readable aliases, which spell out the abbreviated table and column names. This facilitates both the automatic SQL-to-NL generation as well as the manual creation of NL/SQL pairs since the domain experts are aided with more meaningful names.

  Let us revisit the SQL-statement query Q2 of our running example in Section 2. It is not clear, what attribute *s.z* is referring to, since there is no extra information about the column *z* in table *specobj*. However, given the logical alias of *s.z*, we are able to see that *s.z* stands for redshift, which can then be rewritten as *spectroscopic_object.redshift*. Using the same transformation on all tables and columns, we obtain the readable and semantically meaningful SQL query which facilitates the development of the corresponding NL questions.

In the next step of the second phase, *query templates* are filled with the contents of the database (i.e., table names, column names,

and values) using the enhanced schema. To increase the diversity of the queries in the synthetic dataset, we apply *random sampling* to the AST representation of the template. The random sampling only *changes the leaf nodes in the AST*, which represent the corresponding columns, tables, and values. For instance, the projection column *specobjid* from table *specobj* may be changed to column *objid* from table *neighbors*, as shown by *Generated AST (1)* and *Generated SQL (1)* in Figure 1. Another result of the random sampling is represented by *Generated AST (2)* and *Generated SQL (2)*. In this case, the table *specobj* is still used, but the projection column *z*, as well as the filter condition on the column *survey*, are new.

Algorithm for SQL Query Generation Algorithm 1 details the step by step process of automatically generating a SQL query using the AST templates and enhanced schema. We explain the algorithm with the example shown in Figure 2. In particular, we analyze the generation of the following SQL query which is also shown in Figure 1:

```
SELECT T1. objid FROM neighbors AS T1
WHERE T1. neighbormode = 2
```

The algorithm starts with extracting leaf nodes from the AST templates and initializes a new temporary set of empty hash-maps, including Tables, Columns and Values (see lines 1 to 6).

Each set of *Leaf nodes* can be represented as a quadruple for a given attribute (see line 7) which consists of the aggregator function position, table position, column position and value position. An example of such a quadruple is shown in Figure 2 (see the right side of the Root-node indicated as `Filter(2)`). We will focus on the leaf nodes surrounded by dotted-lines and green backgrounds. For instance, `Filter(2)` refers to the filter with position 2, which is equivalent to a query with an exact match filter. `A(0)` refers to an attribute without an aggregation function. `T(0)` and `C(1)` refer to the table with position 0, i.e. *neighbors*, column with position 7 without aggregation function, i.e. *neighbor_mode*. Finally, `V(3)` refers to a value with position 3, i.e. the value 2.

This quadruple needs to be unpacked to extract the information about tables, columns and values as described informally above. Formally, the extraction of the tables, columns and values using the enhanced schema is described in lines 8 to 20 of Algorithm 1. If a position of a certain table, column or value is not found in the keys of the hash map, the respective sampling function will select a new value within the constraints of the enhanced schema, e.g., `sampleTable()` for table sampling (see line 9). Then the corresponding hash map will add the new position-value pair. At the end of the loop, all hash maps are filled with required position-value pairs for tables, columns and values.

Finally, the AST is created on the fly and the corresponding SQL is returned (see lines 21 and 22 of Algorithm 1).

3.3.3 Phase 3: SQL-to-NL Translation Phase. In the third phase of our pipeline shown in Figure 1, we generate the natural language questions (NLQs) that correspond to the newly generated SQL queries. To achieve this, we use *GPT-3* [9] [6]. We present details on

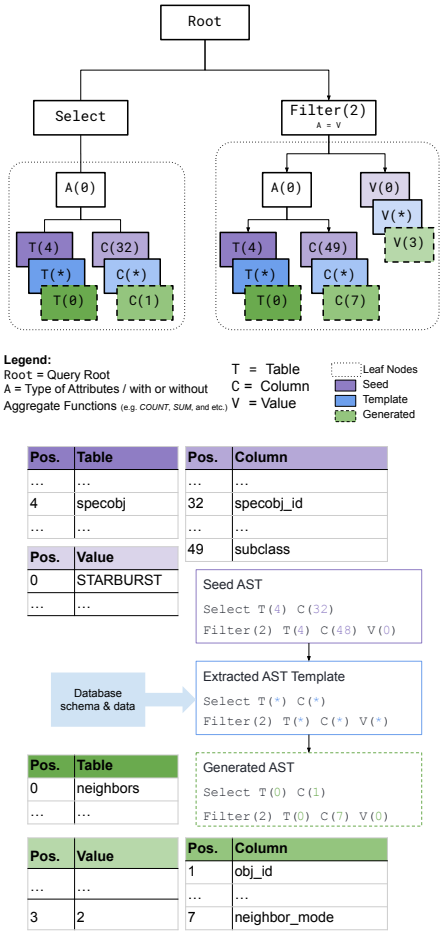

Figure 2: Example of extracting & applying query templates for automatically generating SQL queries as shown in Algorithm 1. The top part shows the abstract syntax tree (AST) of a specific query. The lower part shows how query templates are applied for generating the SQL queries based on database schema and data.

the evaluation of LLMs for generating NL questions given a SQL statement in Section 4.1.

We fine-tuned GPT-3[10] on a subset of 468 samples of the Spider training data for four epochs. This alleviates the need for prompt engineering. Thus, the input to GPT-3 is a SQL query, and we let GPT-3 generate 8 natural language question-candidates to increase the linguistic diversity. Since there is no additional input required beyond the SQL query (e.g., database schema, extra information about the DB, DB contents), this approach easily transfers to any new database without any manual effort or need for extra data.

For the more domain-specific databases, we conduct fine-tuning on GPT-3 with the manually created seed NL/SQL pairs. Afterwards we apply the fine-tuned LLMs to translate SQL to NL. As in the

[9]Note that we also experimented with DBPal [46] as an alternative but we opted for a custom pipeline using GPT-3 since the generated natural language questions are more fluent. However, DBPal can easily be integrated in our pipeline to further extend ScienceBenchmark with additional training data.

[10]We use a fine-tuned version of GPT-3 to generate NL questions because fine-tuning GPT-4 is not available yet https://platform.openai.com/docs/guides/fine-tuning/what-models-can-be-fine-tuned.

Algorithm 1: Generating SQL queries from an AST template and the enhanced schema

```
Input  : An AST Template Ast
Input  : An enhanced schema of the target database
         EnhancedSchema
Output : A new generated SQL Sql
begin
    LeafNodes ← ExtractLeafNodes (Ast);
    Tables ← EmptyHashMap;
    Columns ← EmptyHashMap;
    Values ← EmptyHashMap;
    foreach LeafNode ∈ LeafNodes do
        AggregatorPosition, TablePosition, ColumnPosition,
          ValuePosition ← LeafNode ;          // as quadruple
        if ∃TablePosition : (TablePosition, TableValue) ∉ Tables
          then
            TableValue ← SampleTable (Tables, EnhancedSchema);
            Tables.update(TablePosition, TableValue);
        end
        if ∃ColumnPosition : (ColumnPosition, ColumnValue) ∉
          Columns then
            ColumnValue ← SampleColumn (AggregatorPosition,
              TableValue, Columns, EnhancedSchema);
            Columns.update(ColumnPosition, ColumnValue);
        end
        if ∃ValuePosition : (ValuePosition, ValueValue) ∉ Values
          then
            ValueValue ← SampleValue (TableValue,
              ColumnValue);
            Values.update(ValuePosition, ValueValue);
        end
    end
    Sql ← Transform (Ast, Tables, Columns, Values) ;
     // Generated AST created on-the-fly
    return Sql;
end
```

Spider dataset, to obtain a larger variety of questions and achieve higher linguistic diversity, we generate several candidate NLQs per query. This approach also approximates the Spider dataset where each SQL query has multiple semantically equivalent natural language questions.

3.3.4 Phase 4: Discriminative Phase. The last phase of our data generation pipeline, as shown in Figure 1, selects the one or two best NL questions from the set of candidates generated in the previous phase.

Consider, for instance, the following two NL questions depicted in Figure 1: *"Find the center object which has nearest neighbor with neighbor mode 2"* and *"Find the center id of nearest neighbor object with neighbor mode smaller than 2"*. The discriminative phase aims at deciding, which one better represents the SQL query `"SELECT T1.objid FROM neighbors AS T1 WHERE T1.neighbormode = 2"`.

Inspired by the centroid-based text summarization method [35], the best NLQs are those, whose word embeddings are closest to the centroid of all sample questions. To find these points, we select the candidates that are closest to the centroid, i.e. the geometric median of all embeddings. For this, we apply *SentenceBERT*[33], to generate a set of sentence embeddings for all candidates: $x_i \in \mathbb{R}^m$ and $1 \leq i \leq n$. The best NLQs are computed by taking the geometric median and selecting the closest embedding.

Consider a set $X \in \mathbb{R}^m$ which contains n embeddings of generated NL questions, $x_1, x_2, ..., x_n$, where m denotes the dimension of embedding space. By the definition of geometric median, we can find the closest embedding $y \in X$ with respect to the centroid vector in the space by *solving the optimization problem* formalized as $f(y)$:

$$f(y) = \arg\max_{y \in \mathbb{R}^m} \sum_{i=1}^{n} CosSim(x_i, y) \tag{1}$$

That is, finding the candidate NLQ whose embedding has the highest cosine similarity to the centroid. We perform this process k times on the set $X \setminus \{y\}$ until we have the top k natural language candidates. We choose one or two best NLQs, i.e., $k \in \{1, 2\}$.

3.4 ScienceBenchmark Statistics

Table 2 gives an overview of our new benchmark dataset called *ScienceBenchmark* which we constructed using the automatic data generation pipeline shown in Figure 1. Note that for each of the three domain-specific databases described in Section 3.1, we present two manually created subsets (Seed and Dev) and one automatically generated subset (Synth). The manually created Seed and Dev queries were created by a team of 20 domain and SQL experts as described in Section 3.2, while the Synth queries we produced by our data generation pipeline described in Section 3.3. The Seed queries are used for the automatic data generation pipeline to generate synthetic data (Synth), while the Dev queries are used to evaluate NL-to-SQL systems.

Table 2 also shows the query difficulty (a metric defined by the creators of Spider [51]) distribution for each dataset. For the CORDIS and SDSS datasets, we note that the complexity of the queries is higher than the queries in the Spider Dev Set. For OncoMX, the complexity of the queries is closer to that of the distribution of the Spider dataset. This is due to the database featuring recursive traversals of complex hierarchies of anatomical entities which is outside of the scope of current NL-to-SQL systems. Note that the complexities of the queries generated by our pipeline are generally lower than the complexity of the manually created training data, or the complexity of the Spider dataset. The reason is that with more complex templates the generated queries tend to be semantically incorrect.

4 EVALUATING THE QUALITY OF SCIENCEBENCHMARK

In this section, we evaluate the *quality* of our new benchmark dataset *ScienceBenchmark*. The main objective is to answer the following two research questions:

- *Research question 1: How well do current methods work for translating SQL to NL?*
- *Research question 2: What is the quality of the automatically generated synthetic data, i.e. NL/SQL pairs?*

Table 2: New benchmark dataset called *ScienceBenchmark* which we constructed using the automatic training data generation pipeline shown in Figure 1. The size and complexity of the queries in the 3 databases of ScienceBenchmark are according to the Spider [51] hardness classification scheme. The datasets Seed and Dev are manually generated by domain and SQL experts. The datasets Synth are automatically generated. In the bottom part we also include the equivalent statistics of the Spider dataset for comparison.

Dataset	Easy	Medium	Hard	Extra Hard	Total
CORDIS Seed	4 (4%)	15 (15%)	38 (38%)	43 (43%)	100
CORDIS Synth	726 (55.59%)	494 (37.83%)	66 (5.05%)	20 (1.53%)	1306
CORDIS Dev	25 (25%)	35 (35%)	19 (19%)	21 (21%)	100
SDSS Seed	20 (20%)	54 (54%)	2 (2%)	24 (24%)	100
SDSS Synth	326 (15.82%)	1396 (67.73%)	138 (6.7%)	201 (9.75%)	2061
SDSS Dev	12 (12%)	28 (28%)	20 (20%)	40 (40%)	100
OncoMX Seed	34 (34%)	33 (33%)	19 (19%)	14 (14%)	100
OncoMX Synth	464 (43.57%)	601 (56.43%)	0 (0%)	0 (0%)	1065
OncoMX Dev	21 (21%)	32 (32%)	27 (27%)	20 (20%)	100
Spider Train	1944 (22.45%)	2831 (32.7%)	1758 (20.3%)	2126 (24.55%)	8659
Spider Dev	250 (24.22%)	440 (42.64%)	174 (16.86%)	168 (16.28%)	1032

In order to answers these questions, we first present our evaluation of four different LLMs for translating SQL to NL. The best-performing LLM will then be used to generate the synthetic data. Afterwards, we evaluate the correctness of the synthetic data for each of the three databases of ScienceBenchmark by performing an expert evaluation.

4.1 Evaluation of LLMs for SQL-to-NL Translation

This section describes the experiments we performed in order to decide which LLM to incorporate into our automatic data generation pipeline. We evaluate the accuracy of each LLM in isolation. The best LLM is then used in Phase 3 "SQL-to-NL Translation" of our automatic data generation pipeline shown in Figure 1.

We analyze the performance of four different LLMs, which are all based on large-scale transformer language models [43]. We use these LLMs for *translating the SQL queries* in the Spider Dev set *to natural language*. We apply various automated metrics to these results as well as an expert evaluation.

Large Language Models. We have chosen the following four LLMs for our SQL-to-NL translation:

- GPT-2: A fine-tuned GPT-2-large model [31] with an auto-regressive decoder-only large pre-trained language model, which is well suited for text generation.
- GPT-3-zero: A zero-shot GPT-3 Davinci model [6], which is a larger version of the GPT-2 model pre-trained on even more data.
- GPT-3: A fine-tuned GPT-3 Davinci model, which is GPT-3 fine-tuned on NL/SQL pairs.
- T5: A fine-tuned T5-base model [32], which is an encoder-decoder-based pre-trained language model developed for machine translation.

We fine-tuned a GPT-2-large language model on the Spider training data for 20 epochs. The GPT-3 model was fine-tuned on a subset of Spider for 4 epochs[11]. For this, we sampled three NL/SQL-pairs from each database in the Spider training set at random, which resulted in a training set of 468 NL/SQL-pairs. We used a simple prompt to trigger the translation from SQL to NL. The T5-base model was fine-tuned on the entire Spider dataset, for 10 epochs.

Table 3: Evaluation of various LLMs for generating natural language questions given a SQL query. The goal is to validate Phase 3 "SQL-to-NL Translation" of our automatic data generation pipeline shown in Figure 1. The evaluation is performed on the Spider Dev set using two different automatic performance metrics (SacreBLEU and SententeceBERT) as well as human experts.

Metric	GPT-2	GPT-3-zero	GPT-3	T5
SacreBLEU	33.85	30.36	**38.55**	31.79
SentenceBERT	0.840	0.870	**0.888**	0.864
Human Expert	0.629	**0.765**	0.731	0.645

Metrics. Each LLM is evaluated using the SacreBLEU score [28, 30] and the SentenceBERT score [33] automatic metrics. SacreBLEU is an instantiation of the BLEU score which measures the word overlap between two sentences. However, word overlap metrics do not capture semantically equivalent natural language questions.

For instance, consider the following two sentences (1) "*Find all Starburst galaxies?*" and (2) "*Return all the spectroscopically observed galaxies that lie in the starburst class.*". Both statements describe the same information request, however, they have a low BLEU score. Thus, we also use SentenceBERT, which measures the semantic similarity of sentences.

Additionally, since automated evaluations are not perfectly reliable, we also ran an expert evaluation where 7 SQL experts rated the generated questions. For each expert, we randomly sampled 25

[11]We decided to use only a subset of Spider to keep the costs low, since fine-tuning on all Spider data for 20 epochs would cost 600$ (we only payed 10$). Note that 4 epochs is the default value provided by GPT-3 for fine-tuning.

SQL queries from the Spider Dev set and let each of the four LLMs generate the corresponding natural language question. Thus, each expert annotated 100 SQL/synthetic question-pairs. In other words, for each LLM, we have 175 expert annotations.

4.1.1 Results for Spider Datasets. The evaluation results on the Spider Dev Set using various metrics are summarized in Table 3. The first two lines show the scores given by the automatic metrics SacreBLEU and SentenceBERT. The third line shows the evaluation by human experts. This metric shows the ratio of samples that human experts regarded as being correct.

We observe that the GPT-3 model outperforms the other models by a large margin in terms of SacreBLEU score. The average SentenceBERT similarity is also highest for the fine-tuned version of GPT-3. The human expert evaluation shows that both versions of GPT-3 achieve significantly higher scores than GPT-2 and T5. However, the difference between the two versions of GPT-3 are not significant, i.e. 76.5% vs. 73.1%. Thus, we opt to use the fine-tuned version of GPT-3 since it achieved the highest scores on ScareBLEU and SentenceBERT.

4.1.2 Results for ScienceBenchmark. We also ran expert evaluations for each of the three domains contained in the *ScienceBenchmark*. For this, we translated 100 manually generated SQL queries (called dev queries) to NL questions using a GPT-3 model, which was fine-tuned on the specific database. For each database, we used the manually created training queries and the same 468 Spider queries used above to fine-tune GPT-3. We then performed the expert evaluation for the domain-specific GPT-3 models.

For the CORDIS dataset, GPT-3 correctly translates SQL to NL in 82% of cases, for OncoMX 73%. For SDSS, the ratio is lower at 53%, which is mostly due to the higher complexity of the dev queries.

Answer to Research Question 1: We have shown that LLMs are powerful enough to generate good NL questions for a variety of domains, be it common knowledge or highly domain-specific.

4.2 Evaluation of Synthetic Datasets (Silver Standard) of ScienceBenchmark

We now analyze the quality of the synthetic datasets (or silver standard) for the novel domains of our *ScienceBenchmark* via an expert evaluation. Note that in the previous section we only evaluated the translation of SQL to NL for the *manually written* Dev Set SQL queries. Now we evaluate the synthetic datasets of CORDIS, SDSS and OncoMX, where *both the SQL queries and the corresponding NL questions are automatically generated* using the pipeline in Figure 1.

Distantly labelled data, also known as "silver standard" data has been used as a resource for reliably training neural networks when manually labelled data or "gold standard" data is scarce or unavailable. As shown in previous work on distant supervision [34], training data does not have to be perfect and neural networks can learn from noisy or partially incorrect training data.

Many training data generation systems such as DBPal [46] are based on the principal that silver standard data (possibly noisy data), is sufficient for training. Although DBPal provides an end-to-end systems analysis to show the effectiveness of the generated data, they do not provide any manual analysis of the quality or accuracy of the training data itself.

Table 4: Manual evaluation of 100 randomly chosen synthetic NL/SQL-pairs of ScienceBenchmark. The results show both the semantic equivalence of the automatically generated NL questions with their corresponding, automatically generated SQL queries and the semantic meaningfulness of these automatically generated SQL queries.

Dataset	Semantic Meaningfulness of SQL	Semantic Equivalence of NL and SQL
CORDIS	97%	83%
SDSS	97%	76%
OncoMX	91%	75%

Because the SQL queries in our data generation pipeline are generated using rule-based algorithms and filtered with heuristics crafted by domain experts to ensure the domain relevance of the SQL queries, we evaluate the *semantic equivalence* of the NL questions generated in our pipeline i.e. we check if the NL question matches the meaning of the SQL query. First, we randomly sampled 100 NL/SQL-pairs from each synthetic dataset (CORDIS, SDSS, OncoMx) proportionally in line with the Spider hardness classification schema. Afterwards, we manually evaluated the NL questions against the matching SQL query.

Table 4 shows the results of our manual evaluation of the synthetic NL/SQL pairs. First, we analyzed the *semantic meaningfulness* of the generated SQL queries (see second column of Table 4). For instance, a query that applies an aggregation over numeric data values is considered meaningful, while applying an aggregation over string values is not. Our results demonstrate that we generated meaningful queries in 91 to 97% of the cases in all three datasets.

We also analyzed the *semantic equivalence* of the generated NL question and the respective generated SQL query. The results show that in 75 to 83% of the cases we observe a semantic equivalence.

In summary, the synthetic queries are automatically generated and can be considered *silver standard data*. Previous experiments have shown that (even noisy) silver standard data outperform curated datasets (see [16]). These silver standard data can even be false. Hence, we apply the same approach and do not filter out any query.

Answer to Research Question 2: This analysis demonstrates that the quality of the synthetically generated or "silver standard" data for the three novel domains of ScienceBenchmark is high. Moreover, as we will show in Section 5.3, using the synthetic datasets for training also significantly improves the performance of NL-to-SQL systems evaluated on ScienceBenchmark.

5 BASELINE EXPERIMENTS: USING SCIENCEBENCHMARK TO EVALUATE NL-TO-SQL SYSTEMS

In this section, we perform baseline experiments to evaluate the performance of popular NL-to-SQL systems on *ScienceBenchmark*. The main research questions we want to address are as follows:

- *Research question 3: How well do current NL-to-SQL systems perform on complex, real-world scientific databases?*

- *Research question 4: How much can NL-to-SQL systems be improved using data augmentation when little to no domain-specific training data is available?*

5.1 NL-to-SQL Systems

To test the performance of NL-to-SQL systems on *ScienceBenchmark*, we selected state-of-the-art fine-tuned and few-shot systems. The fine-tuned systems meet the following criteria: 1) Access to the open source model and the pre-trained model weights; 2) Access to bidirectional conversion code between SQL and intermediate representations (IR) of the systems, if any IR is used in the model[12].

Therefore, for our experiments with fine-tuned models we use three different state-of-the-art NL-to-SQL systems: the only two completely open source state-of-the-art NL-to-SQL systems from the Spider leaderboard[13] (T5-Large, SmBoP) and an industrial-strength NL-to-SQL system, which has recently been extended to handle complex, real-world datasets (ValueNet).

- T5-Large [32] (with Picard [38] for constrained decoding). T5-Large is a language model, which is pre-trained on large amounts of text data[14].
- SmBoP [36] (with GraPPa [50]). SmBoP implements a novel autoregressive bottom-up decoder, enhanced with the GraPPa, which is a language model specifically pre-trained for the NL-to-SQL task.
- ValueNet [7]. ValueNet is based on the IRNet [13] architecture and extends the SemQL grammar by adding values to the generated queries to make them executable. To handle the SDSS astrophysics data, we extended the SemQL grammar for ValueNet to incorporate mathematical operations [15].

To evaluate the difficulty of ScienceBenchmark, we also include baseline experiments with two LLMs, GPT-3.5 from OpenAI and for comparison a newly released open source model, LLaMA2 70B from Meta[16].

For reproducibility reasons, we provide both the source code of our automatic training data generation approach and the datasets of *ScienceBenchmark*[17].

5.2 Experimental Setup

For each of the 3 databases of *ScienceBenchmark*, we ran four experiments:

- *Spider Train (Zero-Shot)*: Here, we train the NL-to-SQL systems on the *Spider Train Set*, and run the evaluation on the *Dev Set* of the respective new domain, i.e. on the evaluation set of CORDIS, SDSS and OncoMX.
- *Spider Train + Domain Train*: We train the NL-to-SQL systems on a mix of the Spider Train Set and the manually created training data, e.g. *CORDIS Train*. The goal is to understand how much impact manually generated, domain-specific training queries have on the performance of the NL-to-SQL system.
- *Spider Train + Domain Synth*: We train the systems on a mix of Spider training data and the synthetic data, which we automatically generated using our training data generation pipeline. For instance, *CORDIS Synth* is the automatically generated (synthetic) training data for the domain *research policy making*.
- *Spider Train + Domain Train + Synth*: Here, we train the systems on a mix of the Spider training data, manually created training data, and synthetic data from each domain. This shows the impact of using both synthetic and manually curated data.

The left-most column of Table 5 shows all possible settings, i.e., the four experiments per Dev Set (database), which is shown in the second column. The evaluation is performed using *Execution Accuracy*, which is the metric used by the Spider benchmark. In other words, we measure in how many cases the result sets of the predicted SQL queries correspond to the result sets of the Dev Set SQL queries.

5.3 Experimental Results

Table 5 is divided in two parts: The left most parts show the execution accuracy for NL-to-SQL systems with fine tuning using *ScienceBenchmark* (for the experiments described in the previous section). The right most parts of the table show results for LLMs with prompt-engineering.

Let us first discuss the results of the NL-to-SQL systems that we fine-tuned, i.e., ValueNet, T5-Large and SmBoP. Our results show that *ScienceBenchmark* poses a challenge to current NL-to-SQL approaches. As expected, the performance of the various systems is very low in the zero-shot setting, as the domains covered in Spider are hardly transferable to the novel complex domains. The results also highlight that simply augmenting the number of training samples does not enable the models to achieve high accuracy (80% and above) on the benchmark. In fact, adding the manual and synthetic training data of our domains, increases the scores by a large margin, however, the absolute scores remain low. For instance, the score increases by 23% on CORDIS when training ValueNet on all the data. For T5-Large, data augmentation improves the zero-shot case by 45% resulting in a maximum execution accuracy of 56%. However, with an execution accuracy of 56% the task is far from solved. This trend is noticeable for each of the three novel domains. We also observe that the particularities of each domain create a different degree of difficulty to each system as witnessed by the different scores each system achieves in each domain with zero-shot learning.

Let us now analyze the performance of LLMs where we use zero-shot and few-shot prompting as shown in the right part of Table 5. In general, we can observe that GPT-3.5 performs better than LLaMA2, since the former has significantly more parameters (175B vs. 70B). Moreover, we can see that GPT 3.5 performs best on CORDIS reaching a maximum execution accuracy of 60%. However, for SDSS, the dataset with the most numerical values and mathematical operators, the highest execution accuracy reaches only 33%.

[12]Since the SQL-to-NatSQL [11] conversion code is not available, as announced by the author in their code repository,https://github.com/ygan/NatSQL, all systems integrated with NatSQL have been excluded from our experiments.
[13]https://yale-lily.github.io/spider
[14]Due to compilation issues with Picard's decoder architecture implemented in Haskell, we only used T5-Large without the Picard version of the code provided by Picard.
[15]We only incorporated the mathematical operations for ValueNet as it was straightforward to extend SemQL. The T5 architecture handles mathematical operations out-of-the-box as we use the unconstrained version.
[16]https://ai.meta.com/llama/
[17]The source code, the datasets, the hyperparameters, and the model-specific experiments' hardware specifications for ScienceBenchmark can be found at: https://sciencebenchmark.cloudlab.zhaw.ch/

Table 5: Putting *ScienceBenchmark* into practice: Evaluation of NL-to-SQL systems without and with data augmentation (left most parts of the table). Execution Accuracy is evaluated on the Dev Set of three novel datasets. The numbers in brackets refer to the relative improvements with respect to the zero-shot baseline. The right most parts of the table show results for LLMs with prompt-engineering.

Dataset		Fine-Tuning with NL-to-SQL Systems			Zero & Few-shot-Prompting with LLMs		
Train Set	**Dev Set**	**ValueNet**	**T5-Large w/o Picard**	**SmBoP+GraPPa**	**Prompting***	**GPT-3.5 (175B)**	**LLaMA2 (70B)**
Spider Train (Zero-Shot)	CORDIS	0.12	0.16	0.16	Zero-shot	0.57	0.06
Spider Train + Seed CORDIS	CORDIS	0.20 (+0.08)	0.20 (+0.04)	0.20 (+0.04)	One-shot	0.60	0.22
Spider Train + Synth CORDIS	CORDIS	0.31 (+0.19)	0.30 (+0.14)	0.23 (+0.07)	-	-	-
Spider Train + Seed CORDIS + Synth CORDIS	CORDIS	0.35 (**+0.23**)	0.29 (+0.13)	0.21 (+0.05)	Five-shot	0.57	0.18
Spider Train (Zero-Shot)	SDSS	0.08	0.05	0.06	Zero-shot	0.29	0
Spider Train + Seed SDSS	SDSS	0.11 (+0.03)	0.06 (+0.01)	0.10 (+0.04)	One-shot	0.32	0.06
Spider Train + Synth SDSS	SDSS	0.18 (+0.10)	0.12 (+0.07)	0.13 (+0.07)	-	-	-
Spider Train + Seed SDSS + Synth SDSS	SDSS	0.21 (**+0.13**)	0.15 (+0.10)	0.15 (+0.09)	Five-shot	0.33	0.03
Spider Train (Zero-Shot)	OncoMx	0.13	0.11	0.19	Zero-shot	0.50	0.11
Spider Train + Seed OncoMx	OncoMx	0.44 (+0.31)	0.35 (+0.24)	0.32 (+0.13)	One-shot	0.51	0.21
Spider Train + Synth OncoMx	OncoMx	0.23 (+0.10)	0.27 (+0.14)	0.20 (+0.01)	-	-	-
Spider Train + Seed OncoMx + Synth OncoMx	OncoMx	0.49 (+0.36)	0.56 (**+0.45**)	0.35 (+0.16)	Five-shot	0.55	0.32

Again, the particularities of each domain play a significant role in determining the performance of each model.

In summary, these results demonstrate that ScienceBenchmark is a highly challenging dataset and thus confirms that translating NL-to-SQL is far from being solved.

Answer to Research Question 3: The results show that the current state-of-the-art approaches, which work exceptionally well on Spider, do not perform well on real-world databases. Thus, we pose this dataset as a challenge, which requires more than an increase in training data size. It requires novel approaches that are able to handle all the new complexities introduced by these domains.

Answer to Research Question 4: The results also show that for each domain and NL-to-SQL system, the combination of seed and synthetic queries yields an *improvement over the zero-shot baseline of up to 45%*. The magnitude of the improvements varies depending on the NL-to-SQL system and domain.

5.4 Discussion of the Experiments

Our experiments show that our automated data augmentation pipeline creates training data which is well suited for bootstrapping novel domains. The magnitude of the improvement depends on the NL-to-SQL systems themselves. However, for all highly domain-specific databases, the performance of the systems trained with synthetic data improved. This is useful when adapting an NL-to-SQL system trained on Spider data for a novel and more complex domain for a database with more tables, columns and rows.

The results highlight the necessity to work on real-world applications. In the zero-shot setting all the state-of-the-art systems achieved poor performance. For instance, in the SDSS domain none of the systems achieved an accuracy of even 10%. Even with the fine-tuning data, the performance of the systems are far from the 70% achieved in the Spider setting. Thus, the results show the need of a more complex and real-world oriented benchmark.

Furthermore, our results reveal that the usage of the synthetic data is useful to increase the performance of fine-tuned models. In most cases using the large set of synthetic data alone yields better results than using the small manual dataset for training. The mix of both the synthetic and the manual data yield the best results.

Thus, *ScienceBenchmark* is highly challenging as it requires to adapt most systems to domain-specific knowledge and to handle complex, real-world scientific databases – which are in stark contrast to the relatively simple databases of the Spider benchmark.

6 RELATED WORK

In this section, we review the related work regarding *data augmentation* and *NL-to-SQL benchmarks*.

6.1 Data Augmentation

Due to the need of deep learning models for high volumes of training examples, combined with sparsity of training data and the cost of manually creating it, a lot of research has been done in the area of data augmentation.

Previous work on data augmentation for NL-to-SQL systems mainly focuses on generating SQL queries that run over a single table rather than over multiple tables of a complete relational database. One such example is DBPal [46], a template-based approach for generating NL/SQL-pairs, which uses manually-crafted templates of NL/SQL-pairs, which can be filled with the names of tables and columns in order to create training instances. Additionally, the authors propose NL augmentations such as paraphrasing, random deletions and synonym substitutions. However this approach might create "robotic" NLQs whose quality might be reduced by the proposed augmentations, since designing rules that can consistently work across all possible questions is notoriously hard.

Another approach [12] creates SQL queries by using simple SQL templates and sampling column names and values from a given table and then applies Recurrent Neural Networks (RNNs) to generate the equivalent NLQ. Some key differences to our work are that: (i) we can generate augmented data without completely relying on manually created NL/SQL-pairs or templates and that (ii) our NLQ augmentation step is much more robust and can generate completely new and realistic NL utterances.

Another approach [27] uses Metamorphic Rules (MRs) to create equivalent alterations of NLQs and database schemas from given

NL/SQL/DB-triplets. Even though this work mainly focuses on providing a more robust evaluation framework for NL-to-SQL systems, it also proposes a methodology for data augmentation that takes advantage of the MRs used for the evaluation. More specifically, the authors present a set of MRs that can be used to create an alteration of either the NLQ or the database schema, while keeping them semantically equivalent to the original. However, the proposed NLQ transformations are relatively simple (i.e., synonym substitution and prefix insertion/deletion/substitution) compared to our approach for generating novel and fluent NLQs. Additionally, compared to our work, this approach is not capable of augmenting the SQL part of the training examples and requires a hand-crafted set of NL/SQL-pairs in order to work for a new database.

The more recent work presented in [47] is one of the few proposed architectures that can generate examples that cover multiple tables of a relational database. This work generates SQL queries by creating templates using an abstract syntax tree grammar and filling them with attributes from the database. The NLQs are then generated using a hierarchical, RNN-based neural model, that recursively generates explanations for all parts of the queries and then concatenates them. Our work differs from the previous because we consider much more complex and robust SQL-to-NL models that can create NLQs with much higher variety and fluency, since they are generated by taking the entire SQL query into account.

Finally, our work differs from all previous work in the sense that instead of simply increasing the performance on a generic dataset like Spider [52], we focus on adapting an NL-to-SQL system on new, unseen and complex databases, with little to no manual effort.

6.2 NL-to-SQL Benchmarks

Progress in NL-to-SQL systems was systematically impeded by the lack of a common, large-scale benchmark dataset. The introduction of WikiSQL [53] and Spider [51], has drastically changed the landscape, allowing for the introduction of deep learning techniques to tackle the problem, as well as providing a common benchmark for comparing different approaches. These two benchmark datasets remain the main point of reference for NL-to-SQL systems despite much criticism (e.g., WikiSQL has low complexity and multiple errors [17] and Spider databases are not realistic [14]).

NL-to-SQL benchmarks can be classified into: *domain-specific* and *cross-domain* datasets. Domain-specific datasets focus on a single database from a specific domain, such as: movies and television series (IMDb [49]), restaurant and shop reviews (Yelp [49] and Restaurants [29, 42]), academic research (Scholar [19] and Academic [23]), financial data (Advising [9] and FIBEN [39]), medical data (MIMICSQL [45]), and questions and answers from Stack Exchange (SEDE [15]).

In contrast, cross-domain datasets contain multiple databases, taken from different domains. Spider-DK [10] and Spider-Syn [10], are extensions of Spider which explore system capabilities at cross-domain generalization and synonym robustness. KaggleDBQA [22] is another cross-domain dataset, although of much smaller size, that has been extracted from Kaggle and features databases taken from the Web. Another cross-domain dataset is OTTA [8], which uses an inverse annotation procedure, whereby automatically generated queries, which are visually displayed are annotated with natural language questions by non-SQL experts.

The main difference of our novel benchmark *ScienceBenchmark*, compared to all aforementioned datasets, is twofold: (i) it contains scientific domains that use domain-specific vocabulary, and (ii) it was developed by scientists and domain-experts over the course of a multi-year research project including partners from both academia and industry. Hence, the deep interactions between scientists and domain experts ensured high quality examples that reflect queries posed by actual users of these complex, scientific databases.

7 CONCLUSIONS

In this work, we introduce the novel benchmark *ScienceBenchmark* for evaluating NL-to-SQL systems as well as LLMs against complex, real-world scientific databases. We also show the end-to-end pipeline for automatically generating large synthetic training datasets for highly domain-specific databases, which are more complex both in terms of the subject matter and in terms of the number of tables, columns and rows than the databases used in the Spider benchmark.

Our experimental results show that *ScienceBenchmark* poses a significant challenge to current NL-to-SQL approaches as well as LLMs. While these systems work exceptionally well on the Spider dataset which has relatively simple databases, they do not perform well on complex, real-world scientific databases. Thus, we argue that **ScienceBenchmark can serve as a new baseline benchmark for evaluating NL-to-SQL systems as well as LLMs and thus sets the stage for novel research efforts to handle the complexities introduced by these real-world challenges**.

ACKNOWLEDGMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 863410. We also thank Jonathan Fürst and Farhad Nooralahzadeh for their contributions in evaluating large language models.

REFERENCES

[1] Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. 2019. A Comparative Survey of Recent Natural Language Interfaces for Databases. *The VLDB Journal* 28, 5 (oct 2019), 793–819. https://doi.org/10.1007/s00778-019-00567-8

[2] 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.

[3] I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. 1995. Natural language interfaces to databases – an introduction. *Natural Language Engineering* 1, 1 (1995), 29–81. https://doi.org/10.1017/S135132490000005X

[4] Ion Androutsopoulos, Graeme D. Ritchie, and Peter Thanisch. 1995. Natural language interfaces to databases - an introduction. *Nat. Lang. Eng.* 1, 1 (1995), 29–81. https://doi.org/10.1017/S135132490000005X

[5] Lukas Blunschi, Claudio Jossen, Donald Kossman, Magdalini Mori, and Kurt Stockinger. 2012. Soda: Generating sql for business users. *arXiv preprint arXiv:1207.0134* (2012).

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[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] Jan Deriu, Katsiaryna Mlynchyk, Philippe Schläpfer, Alvaro Rodrigo, Dirk von Grünigen, 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*. Association for Computational Linguistics, Online, 897–911. https://doi.org/10.18653/v1/2020.acl-main.84

[9] 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)*. Association for Computational Linguistics, Melbourne, Australia, 351–360. https://doi.org/10.18653/v1/P18-1033

[10] Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. 2021. Towards Robustness of Text-to-SQL Models against Synonym Substitution. 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, 2505–2515. https://doi.org/10.18653/v1/2021.acl-long.195

[11] 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

[12] Daya Guo, Yibo Sun, Duyu Tang, Nan Duan, Jian Yin, Hong Chi, James Cao, Peng Chen, and Ming Zhou. 2018. Question Generation from SQL Queries Improves Neural Semantic Parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 1597–1607. https://doi.org/10.18653/v1/D18-1188

[13] 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*. Association for Computational Linguistics, Florence, Italy, 4524–4535. https://doi.org/10.18653/v1/P19-1444

[14] Moshe Hazoom, Vibhor Malik, and Ben Bogin. 2021. Text-to-SQL in the Wild: A Naturally-Occurring Dataset Based on Stack Exchange Data. https://doi.org/10.48550/ARXIV.2106.05006

[15] 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

[16] Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2022. Unnatural instructions: Tuning language models with (almost) no human labor. *arXiv preprint arXiv:2212.09689* (2022).

[17] Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. 2019. A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization. https://doi.org/10.48550/ARXIV.1902.01069

[18] Radu Cristian Alexandru Iacob, Florin Brad, Elena-Simona Apostol, Ciprian-Octavian Truică, Ionel Alexandru Hosu, and Traian Rebedea. 2020. Neural Approaches for Natural Language Interfaces to Databases: A Survey. In *Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, Barcelona, Spain (Online), 381–395. https://doi.org/10.18653/v1/2020.coling-main.34

[19] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a Neural Semantic Parser from User Feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, Regina Barzilay and Min-Yen Kan (Eds.). Association for Computational Linguistics, 963–973.

[20] George Katsogiannis-Meimarakis and Georgia Koutrika. 2023. A survey on deep learning approaches for text-to-SQL. *The VLDB Journal* (2023). https://doi.org/10.1007/s00778-022-00776-8

[21] 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

[22] 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

[23] Fei Li and H. V. Jagadish. 2014. Constructing an Interactive Natural Language Interface for Relational Databases. *PVLDB* 8, 1 (Sept. 2014), 73–84.

[24] Fei Li and Hosagrahar V Jagadish. 2014. NaLIR: an interactive natural language interface for querying relational databases. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 709–712.

[25] Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023. RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, Brian Williams, Yiling Chen, and Jennifer Neville (Eds.). AAAI Press, 13067–13075. https://doi.org/10.1609/aaai.v37i11.26535

[26] 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. arXiv:2305.03111 [cs.CL]

[27] Pingchuan Ma and Shuai Wang. 2021. MT-Teql: Evaluating and Augmenting Neural NLIDB on Real-World Linguistic and Schema Variations. *Proc. VLDB Endow.* 15, 3 (nov 2021), 569–582. https://doi.org/10.14778/3494124.3494139

[28] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 311–318. https://doi.org/10.3115/1073083.1073135

[29] Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a Theory of Natural Language Interfaces to Databases. In *Proceedings of the 8th International Conference on Intelligent User Interfaces* (Miami, Florida, USA) *(IUI '03)*. Association for Computing Machinery, New York, NY, USA, 149–157. https://doi.org/10.1145/604045.604070

[30] Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*. Association for Computational Linguistics, Brussels, Belgium, 186–191. https://doi.org/10.18653/v1/W18-6319

[31] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. "Language Models are Unsupervised Multitask Learners". (2019).

[32] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683* (2019).

[33] 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, Hong Kong, China, 3982–3992. https://doi.org/10.18653/v1/D19-1410

[34] Roland Roller and Mark Stevenson. 2015. Making the most of limited training data using distant supervision. In *Proceedings of BioNLP 15*. Association for Computational Linguistics, Beijing, China, 12–20. https://doi.org/10.18653/v1/W15-3802

[35] Gaetano Rossiello, Pierpaolo Basile, and Giovanni Semeraro. 2017. Centroid-based Text Summarization through Compositionality of Word Embeddings. In *Proceedings of the MultiLing 2017 Workshop on Summarization and Summary Evaluation Across Source Types and Genres*. Association for Computational Linguistics, Valencia, Spain, 12–21. https://doi.org/10.18653/v1/W17-1003

[36] Ohad Rubin and Jonathan Berant. 2021. SmBoP: Semi-autoregressive Bottom-up Semantic Parsing. In *Proceedings of the 5th Workshop on Structured Prediction for NLP (SPNLP 2021)*. Association for Computational Linguistics, Online, 12–21. https://doi.org/10.18653/v1/2021.spnlp-1.2

[37] Diptikalyan Saha, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R Mittal, and Fatma Özcan. 2016. ATHENA: an ontology-driven system for natural language querying over relational data stores. *Proceedings of the VLDB Endowment* 9, 12 (2016), 1209–1220.

[38] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. arXiv:2109.05093 [cs.CL]

[39] Jaydeep Sen, Chuan Lei, Abdul Quamar, Fatma Ozcan, Vasilis Efthymiou, Ayushi Dalmia, Greg Stager, Ashish Mittal, Diptikalyan Saha, and Karthik Sankaranarayanan. 2020. ATHENA++: Natural Language Querying for Complex Nested SQL Queries. *Proc. VLDB Endow.* 13, 11 (2020), 2747–2759.

[40] Alkis Simitsis, Georgia Koutrika, and Yannis Ioannidis. 2008. Précis: from unstructured keywords as queries to structured databases as answers. *The VLDB Journal* 17, 1 (2008), 117–149.

[41] Alexander S Szalay, Jim Gray, Ani R Thakar, Peter Z Kunszt, Tanu Malik, Jordan Raddick, Christopher Stoughton, and Jan vandenBerg. 2002. The SDSS skyserver: public access to the sloan digital sky server data. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. 570–581.

[42] Lappoon R. Tang and Raymond J. Mooney. 2000. Automated Construction of Database Interfaces: Intergrating Statistical and Relational Learning for Semantic Parsing. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural*

Language Processing and Very Large Corpora. Association for Computational Linguistics, Hong Kong, China, 133–141. https://doi.org/10.3115/1117794.1117811

[43] 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).

[44] 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. arXiv:1911.04942 [cs.CL]

[45] 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.

[46] Nathaniel Weir, Prasetya Utama, Alex Galakatos, Andrew Crotty, Amir Ilkhechi, Shekar Ramaswamy, Rohin Bhushan, Nadja Geisler, Benjamin Hättasch, Steffen Eger, Ugur Cetintemel, and Carsten Binnig. 2020. DBPal: A Fully Pluggable NL2SQL Training Pipeline. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 2347–2361. https://doi.org/10.1145/3318464.3380589

[47] Kun Wu, Lijie Wang, Zhenghua Li, Ao Zhang, Xinyan Xiao, Hua Wu, Min Zhang, and Haifeng Wang. 2021. Data Augmentation with Hierarchical SQL-to-Question Generation for Cross-domain Text-to-SQL Parsing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 8974–8983. https://doi.org/10.18653/v1/2021.emnlp-main.707

[48] Xiaojun Xu, Chang Liu, and Dawn Song. 2017. SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning. arXiv:1711.04436 [cs.CL]

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

[50] Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2021. GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing. In *International Conference on Learning Representations*. https://arxiv.org/abs/2009.13845

[51] 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*. Association for Computational Linguistics, Brussels, Belgium, 3911–3921. https://doi.org/10.18653/v1/D18-1425

[52] 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. 2019. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. arXiv:1809.08887 [cs.CL]

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