Querying Large Language Models with SQL

Mohammed Saeed
mohammed.saeed@eurecom.fr
EURECOM
France

Nicola De Cao
ndecao@google.com
Google AI
UK

Paolo Papotti
papotti@eurecom.fr
EURECOM
France

ABSTRACT

In many use-cases, information is stored in text but not available in structured data. However, extracting data from natural language (NL) text to precisely fit a schema, and thus enable querying, is a challenging task. With the rise of pre-trained Large Language Models (LLMs), there is now an effective solution to store and use information extracted from massive corpora of text documents. Thus, we envision the use of SQL queries to cover a broad range of data that is not captured by traditional databases (DBs) by tapping the information in LLMs. This ability would enable the hybrid querying of both LLMs and DBs with the SQL interface, which is more expressive and precise than NL prompts. To show the potential of this vision, we present one possible direction to ground it with a traditional DB architecture using physical operators for querying the underlying LLM. One promising idea is to execute some operators of the query plan with prompts that retrieve data from the LLM. For a large class of SQL queries, querying LLMs returns well structured relations, with encouraging qualitative results. We pinpoint several research challenges that must be addressed to build a DBMS that jointly exploits LLMs and DBs. While some challenges call for new contributions from the NLP field, others offer novel research avenues for the DB community.

1 INTRODUCTION

Declarative querying is one of the main features behind the popularity of database systems. However, SQL can be executed only on structured datasets with a well defined schema, leaving out of immediate reach information expressed as unstructured text.

Several technologies have been deployed to extract structured data from unstructured text and to model such data in relations or triples [11, 58]. While these methods have been studied for more than 20 years, creating well-formed structured data from text is still time consuming and error prone. Existing tools require engineers to manually prepare extraction pipelines, which are typically static and can only extract fixed pairs of attributes [58]. Creating such pipelines is expensive, as training examples must be defined for every relation to extract. Indeed, the precise extraction of typed data in a tuple format (n-ary relations) is still an unsolved task [1, 41].

Querying vs QA. While declarative querying of text is a big challenge, there has recently been incredible progress in *question answering* (QA) over text [47]. In this setting, a question in natural language (NL) is answered by gathering information from a corpus of text documents. Transformers enable the creation of Large Language Models (LLMs), neural networks that are used in a wide variety of NL processing tasks. LLMs, such as those in the GPT family [9, 43, 44], have been trained on large data, such as the entire Web textual content, and can answer complex questions in a closed-book fashion [46] (example (2) in Figure 1). Question answering is reaching new state of the art performance with the release of new LLMs, but it is still not possible to query, in a SQL-like declarative fashion, such models. While it has been shown that such models store high quality factual information [30, 42], they are not trained to answer complex SQL queries and may fail short with such input.

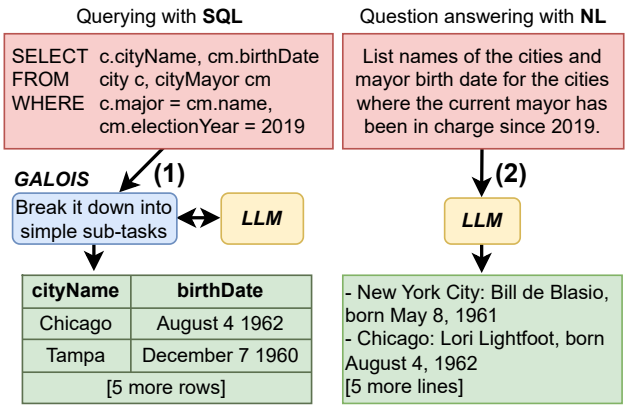

Figure 1: Querying a pre-trained LLM with SQL is different from question answering (QA). We assume a user SQL query as input. GALOIS executes the query, and obtains relations, by retrieving data from a LLM (1). The corresponding QA task consumes and produces natural language text (2).

SQL for LLMs. We envision querying pre-trained LLMs with SQL scripts. As depicted in example (1) in Figure 1, the pre-trained LLM can act as the data storage containing the information to answer the query. We argue that a solution should preserve the main characteristics of SQL when executed over this new source of data: (i) queries are written in arbitrary SQL over a user defined relational schema, enabling a precision and a complexity in contrast with the limitation of NL prompts; (ii) answers are *correct* and *complete* w.r.t. the information stored in the LLM. This last point requires the correct execution of the queries and does not assume that LLMs always return perfect information. In contrast to generation of images or fiction, where small errors are unnoticeable by users in most cases, any error in data can be critical for the target application. While LLMs still make factual mistakes, this work shows that it is already possible to collect tuples from them with promising results. With the ongoing efforts in LLMs, with new training architectures and increasing amount of text used as input, there is evidence that their factuality and coverage is quickly improving [17, 51].

Applications. We envision the execution of SQL queries to obtain relations from the information stored in LLMs. Given the increasing adoption of proprietary LLMs by companies, domain-specific textual information is getting stored in such models [12, 56]. This is a promising solution for several applications. In any domain, data is scattered across different modalities such as email,

Figure 2: With the increasing number of enterprise LLMs trained with proprietary data, hybrid SQL querying enables to extract structured data from heterogeneous sources. The DB models the relational data, while the LLM exposes the data from unstructured sources. This paradigm can query data in text without human preprocessing.

text, and PDF files. Querying their representation in a LLM enables the retrieval of information that is out of reach by accessing only the DB, as depicted in Figure 2. We envision the ability to query data beyond what is already modeled in a structured schema, for example by designing a polystore system on heterogeneous storage engines [2, 16]. An example of jointly querying with one user-provided script a DB and a LLM is the following

```
q: SELECT c.GDP, AVG(e.salary)
   FROM LLM.country c, DB.Employees e
   WHERE c.code = e.countryCode
   GROUP BY e.countryCode
```

where *c* iterates over the tuples in the LLM and *e* over the tuples in the DB. With hybrid querying, the data from the LLM can be used as a source in metadata inference [14], data integration [19], augmentation [60], imputation[33], and cleaning [36]. This paper does not claim to detail a concrete solution to all these applications, but rather to show a possible path to combine traditional DBMSs and LLMs in novel hybrid query execution plans [26].

Which Architecture? Being able to SQL query LLMs is appealing, but it is not clear on which architecture a solution should pivot on. Looking at the architectures for LLMs and DBMSs, there are different paths to explore. One is *LLM-first*, where external information (including structured data) is accessed by the LLM [8, 40]. While this approach is gaining visibility, the limited context size in LLMs does not allow yet to execute queries that require a large number of tuples as input, such as aggregate queries over tables with thousands of rows. The alternative path is *DB-first*, where LLMs are used as a component in a traditional DB query processing architecture, which is what we envision in this work.

Contributions. In this paper, we focus on how to query pre-trained LLMs (in isolation) and preliminary empirical evidence of its potential. We present one possible way to implement a DB-first architecture. The core idea is that the query plan is a natural decomposition of the (possibly complex) process to obtain the result, in analogy with the recent approaches in NLP showing that breaking a complex task in a chain of thoughts is key to get the best results [28, 54]. To bridge the gap between a logical query plan and its execution on a LLM, we suggest new physical operators for such plan.

Our main ideas are summarized in the following points:

- We introduce the problem of querying with SQL existing pre-trained LLMs. To ground our vision, we built Galois[1], a DB-first prototype that executes SPJA queries under assumptions that enable a large class of applications (code available at https://gitlab.eurecom.fr/saeedm1/galois).
- The logical query plan breaks down the complex task into simpler steps that can be handled effectively by the LLM. Physical operators in the query plan are implemented as textual prompts for LLMs. Such prompts are automatically generated from the information in the input schema and the logical operators.
- We show that the results obtained by Galois on 46 queries on top of popular LLMs are (1) comparable to those obtained by executing the same queries on DBMS and (2) better than those obtained by manually rewriting the queries (and parsing the results) in NL for QA over the same LLM.

Outline. Section 2 covers recent progress in NLP and compares Galois to prior work. Section 3 discusses the challenges in querying LLMs. Section 4 describes the architecture of our DB-first prototype. Section 5 reports preliminary experimental results from datasets in the Spider corpus. Section 6 discusses open problems and research directions. Section 7 concludes the paper.

2 BACKGROUND

Our vision is inspired by recent advances in the domain of natural language processing (NLP). Progress in this field has been driven by two major concepts: the Transformer neural network architecture and the application of transfer learning [15]. One of the transformer's benefits is its suitability for parallelization w.r.t. previous approaches, which has enabled the creation of massive LLMs [9]. These models are pre-trained on tasks, such as predicting the next word in a sentence, for which large amounts of data are easily accessible. Although pre-training is costly, the models can then be adapted to new tasks. Traditionally, "fine-tuning" with annotated examples for a target task has been the main way of customizing pre-trained LLMs. However, the latest generation of pre-trained models has opened up new possibilities. Models of sufficient size complete new tasks without any additional training, simply by being given NL descriptions of the task ("instruction tuning"). Precision is improved by incorporating a limited number of examples (e.g., five to ten) that pair the input for the task with its solution ("few-shot learning"). An example of a prompt for GPT-3 is a question in natural language ("what is the capital of USA?") or a request ("The EU state capitals are:").

Our effort is different from the problem of semantic parsing, i.e., the task of translating NL questions into SQL [27, 38, 57]. Our goal is also different from querying an existing relational database to answer a NL question [22]. We are interested in retrieving data from the LLM with SQL queries, with the traditional semantics and with the output expressed in the relational model, as if the query were executed on a DBMS. While some of these facts can be retrieved with QA, (i) the SQL query must be rewritten as an equivalent question in NL, which is not practical for complex scripts, (ii) the textual result must be parsed into a relation, (iii) current LLMs in some cases fail in answering complex queries expressed as NL. Indeed, QA systems are optimized for answering questions with a text, while SQL queries return results in the form of tuples, possibly with complex operations to combine intermediate values, such as aggregates, where LLMs fail short [45]. To overcome some of these limits, it has recently been shown that a series of intermediate reasoning steps ("chain of thought" and question decomposition [55]) improve LLMs' ability in complex tasks [54].

[1]Évariste Galois (rhymes with French word *voilà*) was a 19^{th} century mathematician.

Our work is also different from the recent proposal for Neural DBs [52], where textual facts are encoded with a transformer and queries are posed with short NL sentences. We do not assume facts as input and we focus on traditional SQL scripts executed on LLMs.

Using an LLM query as a component in SQL query answering has analogies with other DBMS extensions in the literature, such as involving crowed workers to answer open world questions [18].

3 DESIGN CONSIDERATIONS

Our goal is to execute SQL query over the data stored into LLMs. When we look at these models from a DB perspective, they (i) have extensive coverage of facts from massive textual sources; (ii) have perfect availability; (iii) directly query a very compressed version of the data, as facts are stored effectively in the parameters of the model: the CommonCrawl+ text corpus takes 45TB, while GPT-3 only 350GB. However, LLMs have their shortcomings, as we discuss next, including poor data manipulation skills, e.g., they fail with numerical comparisons. Conversely, traditional query operators are great at processing data with rich operators, such as joins and aggregates, but only within the data available in the given relation.

The combination of LLMs and traditional DBMSs shows the potential for a hybrid system that can jointly query existing relational tables and facts in the LLM. However, it is crucial to consider the limitations and challenges in querying LLMs. We now delve into three key issues that have impacted the design of Galois.

1. Tuples and Keys. As far as we know, LLMs do not have a concept of schema or tuple, but they model existing relationships between entities ("Rome is located in Italy") or between entities and their properties ("Rome has 3M residents"). However, a query asking for city names may assume that a name identifies a city, which is not the case in reality, e.g., there is a Rome city in Georgia, USA.

In some cases, key attributes exist in the real world. For example, LLMs contain keys such as IATA codes for airports. , e.g., 'JFK'. However, in general, we do not have a universal global key for several entities, such as cities, and the default semantics for the LLM is to pick the most popular interpretation, with popularity defined by occurrences of terms in the original pre-training data.

In general, this problem can be solved with keys defined with multiple attributes, i.e., the *context* in NLP terminology. For example a composite key defined over (name, state, country) enables us to distinguish the Rome in Italy from the one in Georgia. In our initial prototype, we assume that every relation involved in the query has a key and that the key can be expressed with one attribute, e.g., its name. This constraint can be relaxed by handling composite keys.

2. Schema Ambiguity. A major challenge in language is ambiguity. Similarly to the issue with entities, several words, including attribute labels, can have multiple meanings. These alternatives are represented differently in the parameters of LLMs. In our setting, a given attribute label in the query can be mapped to multiple "real world" attributes in the LLM, e.g., *size* for a city can refer to population or urban area [53].

In this initial effort, we assume that meaningful labels for attributes and relations are used in the queries. This allows the system to obtain prompts of good quality automatically.

3. Factual Knowledge in LLMs. LLMs do *not know what they know*. This is an intrinsic challenge in the transformer architecture and the decoder, specifically. The decoder returns the next token in a stream. Such token may be based on either reliable acquired knowledge, or it may be a guess. For this reason, a query result obtained LLMs is not 100% reliable and cannot be immediately verified as LLMs do not expose their sources with the results. However, with Galois, we experimentally demonstrate that it is possible to extract factual information from LLMs to answer SQL queries. Moreover, new models keep increasing the factuality of their answers[2]. In this work, we do not tackle the general problem of separating the *knowledge about language and reasoning* from *factual knowledge*, which is an ongoing NLP research topic as we discuss in Section 6.

4 OVERVIEW

The high-level architecture of Galois is presented in Figure 1. We assume that the schema (but no instances) is provided together with the query. The system processes SQL queries over data stored in a pre-trained LLM. This design enables developers to implement their applications in a conventional manner, as the complexities of using an LLM are encapsulated within Galois.

Operators. The core intuition of our approach is to use LLMs to implement a set of specialized physical operators in a traditional query plan, as demonstrated in Figure 3. As tuples are not directly available, we implement the access to the base relations (leaf nodes) with the retrieval of the key attribute values. We then retrieve the other attributes as we go across the plan. For example, if the selection operator is defined on attribute A different from the key, the corresponding implementation is a prompt that filters every key attribute based on the selection condition, e.g., "Has city *c.name* more than 1M population?", where *c.name* iterates over the set of key values. If a join or a projection involve an attribute that has not been collected for the tuple, this is retrieved with a special node injected right before the operation. For example, if a join involves an attribute "currentMayor", the corresponding attribute values are retrieved with a prompt that collects it for every key, such as "Get the current mayor of *c'.name*". Once the tuples are completed, regular operators, implemented in Python in our prototype, are executed on those, e.g., joins and aggregates.

On one hand, the query plan acts as a chain of thought decomposition of the original task, i.e., the plan spells outs intermediate steps. On the other hand, the operators that manipulate data fill up the limitations of LLMs, e.g., in computing average values or comparing quantities [31]. Together, these two features make the LLM able to execute complex queries.

Prompts. Figure 3 shows how prompts, suitable for execution on LLMs, implement logical operators in Galois. A prompt is obtained for each operator by combining a set of operator-specific prompt templates with the labels/selection conditions in the given SQL query. In the example query q', politicians (`Politicians p`) are filtered according to their age (`p.age<40`); this corresponds to retrieving a set of tuples (P) with one key attribute (name), which is then followed by a prompt that for each politician checks in the LLM its age. For example, we instantiate the template "Has *relationName keyName attributeName operator value*?", with 'politician' , 'B. Obama', 'age', 'less than', '40', respectively. Templates are a simple solution for prompting. However, DB-first

[2]For example, "GPT-4 scores 40% higher than our latest GPT-3.5 on our factuality evaluations" - https://openai.com/research/gpt-4 - published on March 15^{th} 2023

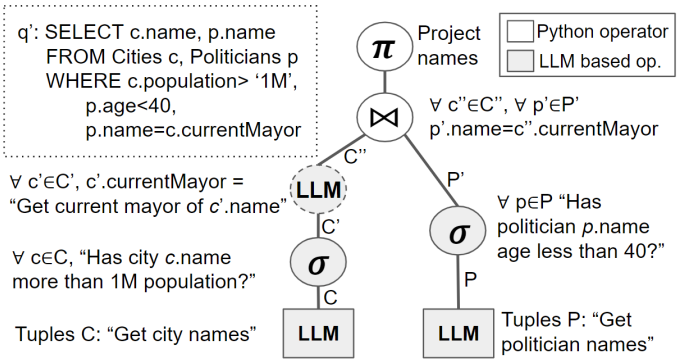

Figure 3: Logical plan for query q'. Base relations are accessed by retrieving sets of tuples (*C*, *P*) with one key attribute (*name*) from the LLM. Other LLM operators consume and produce tuple sets, retrieving for every tuple the required attributes, if not in the tuple yet. The last two operators do not involve the LLM.

```
I am a highly intelligent question answering bot. If you ask
me a question that is rooted in truth, I will give you the short
answer. If you ask me a question that is nonsense, trickery,
or has no clear answer, I will respond with "Unknown". If the
answer is numerical, I will return the number only.
Q: What is human life expectancy in the United States?
A: 78.
Q: Who was president of the United States in 1955?
A: Dwight D. Eisenhower.
Q: What is the capital of France?
A: Paris.
Q: What is a continent starting with letter O?
A: Oceania.
Q: Where were the 1992 Olympics held?
A: Barcelona.
Q: How many squigs are in a bonk?
A: Unknown
```

Figure 4: Few shot examples for the GPT-3's prompt.

approaches such as Galois can directly exploit results in prompt engineering [32], as those could be plugged in step (2) of the workflow described next.

Workflow. The main operations in Galois's query processing are:

(1) Obtain a logical query plan for a query q and the source schema. We assume the label of the key attribute is given.
(2) Access the LLM to retrieve the tuples composed of the key attribute and to gather more attributes in case of selection, join and projection. Each operation is done with a prompt template filled up with the labels and conditions at hand.
(3) Convert the string of answers from the LLM to a set of CELL values in the attribute.
(4) Use traditional algorithms for any operator involving attributes that have already been retrieved.

Two critical steps enable the practical use of Galois. First, as relations can be large, we iterate with the a prompt until we stop getting new results. For example, we ask for city names, collect the answer in a set, and keep asking for more names with another prompt ("Return more results"). The termination condition could be replaced by a user-specified threshold. A second issue is the cleaning of the data gathered from the LLM. For example, numerical data can be retrieved in different formats. We normalize every string expressing a numerical value (say, 1k) into a number (1000). The enforcing of type and domain constraints is a simple but crucial step to limit the incorrect output due to model hallucinations. More LLM-specific methods can be plugged for this cleaning step. For example, the generated output can be critiqued by another model [35, 49], or the LLM can be augmented with external information [25].

In contrast with the traditional approach based on extracting structured data from text, LLMs and prompts enable the extraction of such data without human annotations. In a DB-first approach, prompts are automatically generated, thanks to the logical plan.

5 EXPERIMENTS

All experiments are executed on popular LLMs for a set of SQL queries for which we have the ground truth according to a database.

Implementation. Galois is written in Python and all LLMs have been executed locally with the exception of ChatGPT, for which we used the API. Query plans are obtained from DuckDB. Code and datasets are available at https://gitlab.eurecom.fr/saeedm1/galois. This initial implementation serves as the experimental platform to show the promise of the vision, rather than a full-fledged solution.

Dataset. Spider is a Text2SQL dataset with 200 databases, each with a set of SQL queries [57]. For each query, it provides its paraphrases as a NL question. We focus on a subset of *46 queries for which we expect to obtain answers from an existing LLM*. More precisely, we leave out queries that are specific to the relational dataset provided by Spider (e.g., "How many heads of the departments are older than 56?") and use in our evaluation only queries about generic topics, such as world geography and airports ("What are the names of the countries that became independent after 1950?"). If there are multiple paraphrases for a question, we pick the first one.

Setup. We test four LLMs. *Flan-T5-large* (**Flan**): T5 fine-tuned on datasets described via instructions (783M parameters). *TK-instruct-large* (**TK**): T5 with instructions and few-shot with positive and negative examples (783M parameters). *InstructGPT-3* (**GPT-3**): fine-tuned GPT-3 using instructions from humans [37] (175B parameters). *GPT-3.5-turbo* (**ChatGPT**): chat model in the OpenAI API (175B parameters). We construct prompts appropriately for each model, we report the one for GPT-3 in Figure 4, showing the instruction in the prompt followed by some few-shot examples.

For a given LLM M and a SQL query q with its Spider relation D and the corresponding NL question t, we collect four results: (a) relation R_M from Galois executing q over M, (b) relation R_D by executing q over D, (c) text T_M by asking t over M, (d) text T_M^C by asking t over M using a chain-of-thought prompt. The last result (d) explores the middle ground between standard questions answering (c) and Galois (a). For NL question t, an engineered prompt contains a complete example of a manually crafted chain-of-thought (CoT), similar to the logical plan execution for the query, followed by t and instructions to reason step by step. In this method, the CoT example in the prompt is fixed as how to derive a decomposition automatically from t is an open problem.

	Flan	TK	GPT-3	ChatGPT
Difference as % of R_D size	-47.4	-43.7	+1.0	+19.5

Table 1: Average difference in the cardinality of Galois's output relations (R_M) w.r.t. the ground truth results $|R_D|$ for the 46 Spider queries. Closer to 0 is better.

	All	Selections only	Aggregates	Joins
R_M (SQL Queries)	50	80	29	0
T_M (NL Questions)	44	71	20	8
T_M^C (NL Quest.+CoT)	41	71	13	0

Table 2: Cell value matches (%) between the result returned by a method and the same query executed on the ground truth data (R_D) for the 46 Spider queries. Averaged results for ChatGPT.

Only (b) uses the relations from Spider, (a), (c) and (d) get the data from the LLM.

Evaluation. We analyze the results across two dimensions.

1) Cardinality. First, we measure to which extent Galois returns correct results in terms of number of tuples. As NL questions always return text paragraphs, we cannot include their results in this analysis. For Galois, all output relations have the expected schema, this is obtained by construction from the execution of the query plan, i.e., every R_M has the same schema as every R_D. However, in terms of number of tuples there are differences. We compute the ratio of the sizes as $f = \frac{|2*R_D|}{|R_D+R_M|}$, where the interval for f is [0,2] and best result occurs when $R_D == R_M$ (f=1). Consider expected Relation R_D with size (3,2), i.e., 3 tuples and 2 columns. Assume Galois produced R_M = (1,2). In this case, f = |2*3| / (3+1) = 6/4 = 1.5.

In Table 1, we report the difference as percentage (averaged over all queries with non-empty results) with the formula 1-f.

Results show that smaller models do worse and miss lots of result rows, up to 47.4% w.r.t. the size of results from the SQL execution R_D. For GPT models, almost all queries return a number of tuples close to R_D. Most differences are explainable with errors in the results of the prompts across the query pipeline, as we discuss next.

2) Content. Second, we measure the quality of the results by comparing the content of each cell value after manually mapping tuples between R_D on one side (ground truth) and (R_M, T_M, T_M^C) on the other. As T_M, T_M^C contain NL text, we manually postprocess them to extract the values as records. In our manual mapping, we split comma-separated values, remove repeated values and punctuation, and map the resulting tuples to the ground truth records - how to automate this mapping process is an open problem. We consider a numerical value in (R_M, T_M, T_M^C) as correct if the relative error w.r.t. R_D is less than 5%.

As this analysis requires to manually verify every result, we conduct it only for one LLM. Results in Table 2 show that Galois executes the queries on ChatGPT with a better average accuracy in the results compared to the same queries expressed as questions in NL. We believe this is a very promising result, as one can think that the results coming from the NL QA task are the upper bound for what the LLM knows. For the easiest subclass of queries, selection-only, the query approach returns correct values in 80% of the cases. Joins are the most problematic, as we observe failure in the join step due to different formats of the same text, e.g., an attempt to join the country code "IT" with "ITA" for entity Italy. This challenging subclass of queries clearly requires more work to drastically increase the homogeneity of the intermediate results. The results also show that well-engineered chain-of-thought NL prompts (T_M^C) do not lead to better results than Galois, confirming the quality of the chain of prompts obtained automatically.

As we do not control the infrastructure of OpenAI, we do not report API execution times. On average, GPT-3 takes ~20 seconds to execute a query (~110 batched prompts per query). Distributions for these metrics are skewed as they depend on the result sizes.

6 RESEARCH DIRECTIONS

Galois aims at creating a system that can push the boundaries of declarative query execution over LLMs, while achieving comparable accuracy and performance to queries executed on a traditional DBMS. While the current prototype does not yet meet these goals, we discuss the main next steps in this vision, including open research questions and associated challenges.

Query optimization. As in a traditional DBMS, optimization can be organized according to the logical and physical plans.

For the *logical* plan, an advantage of the DB-first approach is the automatic generation of chain-of-thought prompts. However, the ability to combine in one query plan operators over traditional storage and LLMs is a vision that go beyond the scope of chain-of-thought. For example, an optimizer may be able to recognize when the execution of the (more expensive) LLM is needed at runtime. We also need optimization heuristics to obtain equivalent logical plans that reduce the number of prompt executions (which can be large) over the LLM. In the example in Figure 3, pushing down the selection over city population to the data access call (leaf) requires to combine the prompts, e.g., "get names of cities with > 1M population". This simple change removes the prompt executions for filtering the list of all cities. However, the optimization decision is not trivial as combining too many prompts lead to complex questions that have lower accuracy than simple ones.

For the *physical* plan, interesting problems arise around the textual prompts. Research questions include how to generate them automatically given only the attribute labels, especially when those are ambiguous or cryptic. The rule of thumb is that the more precise the prompt, the better will be the accuracy of its results. One direction is to make use of data samples, when available. Giving examples of the desired output would guide the LLM to the right format and encoding, which is an issue in our current implementation. Another approach is to optimize the prompt for the retrieval task, with some fine tuning or by exploiting pre-defined embeddings for the desired attribute types [61]. For fine tuning, reinforcement learning from human feedback (RLHF) can be explored to better stir the generation towards the factual values that are needed for query processing [5]. For the second approach, given a library of type embeddings such as "Person" and "City", those can be added to prompts for accurate retrieval [48].

Knowledge of the Unknown. To overcome the problem of the results mixing real facts and hallucinations, one direction is to verify generated query answers by another model, possibly also build on LLMs. In most cases, verification is easier than generation, e.g., it is easier to verify a proof rather than generate it. Our enforcing of simple domain constraints shows benefit, but there is the need to adapt more general data cleaning techniques [24].

Another direction is retrieval augmented language models, where they design modules that separate the "language understanding and reasoning" part and "factual knowledge" part [13]. Our prompting is a basic approach to surface facts, but more principled solutions are needed to obtain reliable results [23].

Provenance. Retrieval augmented models are also a promising direction to address the fact that LLMs cannot always precisely cite the sources, or provenance [59], of their output. This is an issue, because it is not possible to judge correctness without the origin of the information. With prompt engineering, LLMs can produce an explanation supporting their output and there are ongoing efforts on linking generated utterances, or values in our case, to sources [7]. This can also be done through the generation process or in a post-processing step [50].

Schema-less querying. We currently assume the SQL schema as given by the user. An interesting extension is to allow users to query without providing a schema. This removes friction from the user, but raises new challenges. Consider the following two queries.

```
Q1: SELECT c.cityName, cm.birthDate
    FROM city c, cityMayor cm
    WHERE c.mayor=cm.name
Q2: SELECT cityName, mayorBirthDate
    FROM city
```

Both of them collect the names of cities with the birth date of the mayor. As the LLMs have no schema, both queries should give the same output when executed, i.e., two SQL queries that are both correct translation of the same NL question should give equivalent results. How to guarantee this natural property (for DBs) is a challenge that requires to combine the new challenges in the LLM setting with results on SQL query equivalence [20].

Portability. As SQL queries are *portable* across DB engines, the same SQL script executes on different LLMs. Differently from the schema-less query case, in this case the query q if fixed, but the LLM changes. If two LLMs are trained on the same data, ideally they should return the same answer for q. However, this requirement is hard to achieve because of the non deterministic learning process for LLMs. As a consequence, the same prompt does not give equivalent results across LLMs.

Architecture. Galois is based on a DB-first architecture, where the LLM is plugged in the operators. The alternative LLM-first architecture is also promising, but with different challenges. An open question is if LLMs can replace DBMSs by consuming the structured data in a training process or as part of the input context. Research on tabular language models show that we are far from this scenario, mostly due to the limitation to the size of the context [4], but recent research is addressing this issue [3]. However, despite the progress in LLM research, legal and economic hurdles (e.g., the need for formal guarantees in transactions) would still affect an LLM-first solution and may ultimately limit its impact.

Updates and Cost. In the DB-first approach, we envision that querying LLMs will be less common than querying traditional DBMSs; LLMs are a source for some use cases, but not a replacement. However, training and using LLMs is expensive and energy consuming. Given the cost of training, it is not clear how to deal with the continuous creation of new information [29]. One short term solution is to update LLMs without retraining [10, 34]. In the long term, cost will be reduced by cheaper training and inference[3].

Coverage and Bias. LLMs focus on common and probable cases by design. We found that, for some queries, missing results are due to their lower popularity, compared to those surfaced by the LLM. Researchers are focusing more on this challenge [17, 51]. However, LLMs do well with huge amount of data, which is available only for few languages. While the problem is mitigated with machine translation, i.e., by translating from English to a target language, in terms of factual knowledge there is no clear solution. In general, the impact of training data and how to select high quality sources is getting more attention with proprietary LLMs [21, 39].

LLMs encode biases and stereotypes that are present in observed human language, we therefore must be careful when applying these models in real-world applications [6].

7 CONCLUSION

This paper presents a vision for the DB community by highlighting the potential of querying Large Language Models with SQL, thereby opening up novel research avenues and opportunities. We report an example for a DB-first approach that leverages the power of LLMs in combination with traditional DBMSs to create a hybrid query execution environment. As LLM factuality and coverage continues to improve, the integration of these models into database systems will not only enable a wide range of data applications, but also inspire new contributions from the NLP field. By showcasing a prototype and envisioning the usage of SQL for querying LLMs, we hope to stimulate further exploration and collaboration between DB and NLP communities, ultimately leading to innovative solutions that unlock previously untapped information from unstructured text data in various domains.

REFERENCES

[1] Daniel Abadi, Anastasia Ailamaki, David Andersen, Peter Bailis, Magdalena Balazinska, Philip A. Bernstein, Peter Boncz, Surajit Chaudhuri, Alvin Cheung, Anhai Doan, Luna Dong, Michael J. Franklin, Juliana Freire, Alon Halevy, Joseph M. Hellerstein, Stratos Idreos, Donald Kossmann, Tim Kraska, Sailesh Krishnamurthy, Volker Markl, Sergey Melnik, Tova Milo, C. Mohan, Thomas Neumann, Beng Chin Ooi, Fatma Ozcan, Jignesh Patel, Andrew Pavlo, Raluca Popa, Raghu Ramakrishnan, Christopher Re, Michael Stonebraker, and Dan Suciu. 2022. The Seattle Report on Database Research. *Commun. ACM* 65, 8 (jul 2022), 72–79. https://doi.org/10.1145/3524284

[2] Divy Agrawal, Sanjay Chawla, Bertty Contreras-Rojas, Ahmed K. Elmagarmid, Yasser Idris, Zoi Kaoudi, Sebastian Kruse, Ji Lucas, Essam Mansour, Mourad Ouzzani, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, Saravanan Thirumuruganathan, and Anis Troudi. 2018. RHEEM: Enabling Cross-Platform Data Processing - May The Big Data Be With You! -. *Proc. VLDB Endow.* 11, 11 (2018), 1414–1427. https://doi.org/10.14778/3236187.3236195

[3] Anthropic. 2023. Introducing 100K Context Windows. https://www.anthropic.com/index/100k-context-windows. (2023).

[4] Gilbert Badaro, Mohammed Saeed, and Papotti Paolo. 2023. Transformers for Tabular Data Representation: A Survey of Models and Applications. *Transactions of the Association for Computational Linguistics* 11 (2023), 227–249. https://doi.org/doi.org/10.1162/tacl_a_00544

[5] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas

[3] For example, "Through a series of system-wide optimizations, we've achieved 90% cost reduction for ChatGPT since December" - https://openai.com/blog/introducing-chatgpt-and-whisper-apis - published on March 1^{st} 2023

Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. (2022). arXiv:cs.CL/2204.05862

[6] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?. In *FAccT*. ACM, 610–623. https://doi.org/10.1145/3442188.3445922

[7] Bernd Bohnet, Vinh Q. Tran, Pat Verga, Roee Aharoni, Daniel Andor, Livio Baldini Soares, Massimiliano Ciaramita, Jacob Eisenstein, Kuzman Ganchev, Jonathan Herzig, Kai Hui, Tom Kwiatkowski, Ji Ma, Jianmo Ni, Lierni Sestorain Saralegui, Tal Schuster, William W. Cohen, Michael Collins, Dipanjan Das, Donald Metzler, Slav Petrov, and Kellie Webster. 2022. Attributed Question Answering: Evaluation and Modeling for Attributed Large Language Models. (2022). https://doi.org/10.48550/ARXIV.2212.08037

[8] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. 2022. Improving Language Models by Retrieving from Trillions of Tokens. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.), Vol. 162. PMLR, 2206–2240. https://proceedings.mlr.press/v162/borgeaud22a.html

[9] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *NeurIPS*. https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[10] Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing Factual Knowledge in Language Models. In *EMNLP*. Association for Computational Linguistics, 6491–6506. https://doi.org/10.18653/v1/2021.emnlp-main.522

[11] Laura Chiticariu, Marina Danilevsky, Yunyao Li, Frederick Reiss, and Huaiyu Zhu. 2018. SystemT: Declarative Text Understanding for Enterprise. In *NAACL*. Association for Computational Linguistics, 76–83. https://doi.org/10.18653/v1/N18-3010

[12] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM. (2023). https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm

[13] Nicola De Cao. 2023. *Entity Centric Neural Models for Natural Language Processing*. University of Amsterdam, Institute for Logic, Language and Computation Thesis Series.

[14] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. TURL: Table Understanding through Representation Learning. *Proc. VLDB Endow.* 14, 3 (2020), 307–319. https://doi.org/10.5555/3430915.3442430

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[16] Jennie Duggan, Aaron J Elmore, Michael Stonebraker, Magda Balazinska, Bill Howe, Jeremy Kepner, Sam Madden, David Maier, Tim Mattson, and Stan Zdonik. 2015. The bigdawg polystore system. *ACM Sigmod Record* 44, 2 (2015), 11–16.

[17] Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard H. Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and Improving Consistency in Pretrained Language Models. *Trans. Assoc. Comput. Linguistics* 9 (2021), 1012–1031. https://doi.org/10.1162/tacl_a_00410

[18] Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. 2011. CrowdDB: Answering Queries with Crowdsourcing. In *ACM SIGMOD*. 61–72. https://doi.org/10.1145/1989323.1989331

[19] Behzad Golshan, Alon Halevy, George Mihaila, and Wang-Chiew Tan. 2017. Data integration: After the teenage years. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems*. 101–106.

[20] Paolo Guagliardo and Leonid Libkin. 2017. A formal semantics of SQL queries, its validation, and applications. *Proceedings of the VLDB Endowment* 11, 1 (2017), 27–39.

[21] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. 2023. Textbooks Are All You Need. (2023). arXiv:cs.CL/2306.11644

[22] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. TaPas: Weakly Supervised Table Parsing via Pre-training. In *ACL*. ACL, 4320–4333. https://doi.org/10.18653/v1/2020.acl-main.398

[23] Or Honovich, Leshem Choshen, Roee Aharoni, Ella Neeman, Idan Szpektor, and Omri Abend. 2021. Q^2: Evaluating Factual Consistency in Knowledge-Grounded Dialogues via Question Generation and Question Answering. In *EMNLP*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 7856–7870. https://doi.org/10.18653/v1/2021.emnlp-main.619

[24] Ihab F. Ilyas and Xu Chu. 2019. *Data Cleaning*. ACM. https://doi.org/10.1145/3310205

[25] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of Hallucination in Natural Language Generation. *ACM Comput. Surv.* 55, 12, Article 248 (mar 2023), 38 pages. https://doi.org/10.1145/3571730

[26] Zoi Kaoudi and Jorge-Arnulfo Quiané-Ruiz. 2022. Unified Data Analytics: State-of-the-art and Open Problems. *Proc. VLDB Endow.* 15, 12 (2022), 3778–3781. https://www.vldb.org/pvldb/vol15/p3778-kaoudi.pdf

[27] George Katsogiannis-Meimarakis and Georgia Koutrika. 2021. A Deep Dive into Deep Learning Approaches for Text-to-SQL Systems. In *SIGMOD*. ACM, 2846–2851.

[28] Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. Demonstrate-Search-Predict: Composing Retrieval and Language Models for Knowledge-Intensive NLP. *arXiv preprint arXiv:2212.14024* (2022).

[29] Angeliki Lazaridou, Adhiguna Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomás Kociský, Sebastian Ruder, Dani Yogatama, Kris Cao, Susannah Young, and Phil Blunsom. 2021. Mind the Gap: Assessing Temporal Generalization in Neural Language Models. In *NeurIPS*. 29348–29363. https://proceedings.neurips.cc/paper/2021/hash/f5bf0ba0a17ef18f9607774722f5698c-Abstract.html

[30] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *NeurIPS*, Vol. 33. 9459–9474. https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

[31] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving Quantitative Reasoning Problems with Language Models. In *NeurIPS*.

[32] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput. Surv.* 55, 9, Article 195 (jan 2023), 35 pages. https://doi.org/10.1145/3560815

[33] Yinan Mei, Shaoxu Song, Chenguang Fang, Haifeng Yang, Jingyun Fang, and Jiang Long. 2021. Capturing Semantics for Imputation with Pre-trained Language Models. In *ICDE*. IEEE, 61–72. https://doi.org/10.1109/ICDE51399.2021.00013

[34] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022. Mass Editing Memory in a Transformer. *arXiv preprint arXiv:2210.07229* (2022).

[35] Niels Mündler, Jingxuan He, Slobodan Jenko, and Martin Vechev. 2023. Self-contradictory Hallucinations of Large Language Models: Evaluation, Detection and Mitigation. (2023). arXiv:cs.CL/2305.15852

[36] Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *Proc. VLDB Endow.* 16, 4 (dec 2022), 738–746. https://doi.org/10.14778/3574245.3574258

[37] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv* abs/2203.02155 (2022).

[38] Simone Papicchio, Paolo Papotti, and Luca Cagliero. 2023. QATCH: Benchmarking Table Representation Learning Models on Your Data. In *NeurIPS (Datasets and Benchmarks)*.

[39] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only. (2023). arXiv:cs.CL/2306.01116

[40] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and Jianfeng Gao. 2023. Check Your Facts and Try Again: Improving Large Language Models with External Knowledge and Automated Feedback. (2023). arXiv:cs.CL/2302.12813

[41] Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics* 5 (2017), 101–115.

[42] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases?. In *EMNLP*. 2463–2473. https://doi.org/10.18653/v1/D19-1250

[43] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[44] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[45] Marco Túlio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In *ACL*. Association for Computational Linguistics, 4902–4912. https://doi.org/10.18653/v1/2020.acl-main.442

[46] Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How Much Knowledge Can You Pack Into the Parameters of a Language Model?. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 5418–5426. https://doi.org/10.18653/v1/2020.emnlp-main.437

[47] Anna Rogers, Matt Gardner, and Isabelle Augenstein. 2023. QA dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension. *Comput. Surveys* 55, 10 (2023), 1–45.

[48] Mohammed Saeed and Paolo Papotti. 2022. You are my type! Type embeddings for pre-trained language models. In *EMNLP 2022, Conference on Empirical Methods in Natural Language Processing*, ACL (Ed.). Abu Dhabi.

[49] Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. (2023). arXiv:cs.AI/2303.11366

[50] Congzheng Song and Vitaly Shmatikov. 2019. Auditing Data Provenance in Text-Generation Models. In *KDD (KDD '19)*. Association for Computing Machinery, New York, NY, USA, 196–206. https://doi.org/10.1145/3292500.3330885

[51] Derek Tam, Anisha Mascarenhas, Shiyue Zhang, Sarah Kwan, Mohit Bansal, and Colin Raffel. 2022. Evaluating the Factual Consistency of Large Language Models Through Summarization. (2022). https://doi.org/10.48550/ARXIV.2211.08412

[52] James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Y. Levy. 2021. From Natural Language Processing to Neural Databases. *Proc. VLDB Endow.* 14, 6 (2021), 1033–1039.

[53] Enzo Veltri, Donatello Santoro, Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. 2022. Pythia: Unsupervised Generation of Ambiguous Textual Claims from Relational Data. In *SIGMOD*. ACM, 2409–2412. https://doi.org/10.1145/3514221.3520164

[54] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. In *NeurIPS*.

[55] Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break It Down: A Question Understanding Benchmark. *Transactions of the Association for Computational Linguistics* 8 (04 2020), 183–198. https://doi.org/10.1162/tacl_a_00309

[56] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. BloombergGPT: A Large Language Model for Finance. (2023). arXiv:cs.LG/2303.17564

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

[58] Ce Zhang, Christopher Ré, Michael J. Cafarella, Jaeho Shin, Feiran Wang, and Sen Wu. 2017. DeepDive: declarative knowledge base construction. *Commun. ACM* 60, 5 (2017), 93–102. https://doi.org/10.1145/3060586

[59] Yi Zhang, Zachary Ives, and Dan Roth. 2020. "Who said it, and Why?" Provenance for Natural Language Claims. In *ACL*. Association for Computational Linguistics, Online, 4416–4426. https://doi.org/10.18653/v1/2020.acl-main.406

[60] Zixuan Zhao and Raul Castro Fernandez. 2022. Leva: Boosting Machine Learning Performance with Relational Embedding Data Augmentation. In *SIGMOD*. 1504–1517.

[61] Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual Probing Is [MASK]: Learning vs. Learning to Recall. In *North American Association for Computational Linguistics (NAACL)*.