Querying Large Language Models with SQL [Vision]

Mohammed Saeed
mohammed.saeed@eurecom.fr
EURECOM

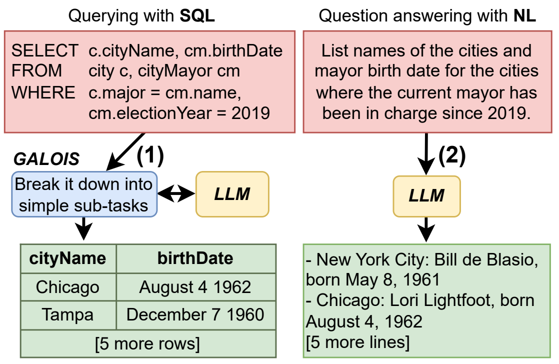
Nicola De Cao
ndecao@google.com
Google AI

Paolo Papotti
papotti@eurecom.fr
EURECOM

ABSTRACT

In many use-cases, information is stored in text but not available in structured data. However, extracting data from natural language 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 by tapping the information in LLMs. To ground this vision, we present Galois, a prototype based on a traditional database architecture, but with new physical operators for querying the underlying LLM. The main idea is to execute some operators of the 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. Preliminary experimental results make pre-trained LLMs a promising addition to the field of database systems, introducing a new direction for hybrid query processing. However, we pinpoint several research challenges that must be addressed to build a DBMS that exploits LLMs. While some of these challenges necessitate integrating concepts from the NLP literature, others offer novel research avenues for the DB community.

ACM Reference Format:
Mohammed Saeed, Nicola De Cao, and Paolo Papotti. 2023. Querying Large Language Models with SQL [Vision]. In *Proceedings of VLDB Conference (Conference'23)*. VLDB, Vancouver, BC, CAN, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn


Querying with SQL
Question answering with NL
GALOIS
(1)
Break it down into simple sub-tasks
LLM
(2)
LLM

```
SELECT  c.cityName, cm.birthDate
FROM    city c, cityMayor cm
WHERE   c.major = cm.name,
        cm.electionYear = 2019
```

List names of the cities and mayor birth date for the cities where the current mayor has been in charge since 2019.

cityName	birthDate
Chicago	August 4 1962
Tampa	December 7 1960
[5 more rows]	

- New York City: Bill de Blasio, born May 8, 1961
- Chicago: Lori Lightfoot, born August 4, 1962
[5 more lines]

Figure 1: We assume a SQL query as input. Galois **executes the SQL query, and obtains relations, by retrieving data from a pre-trained LLM (1). The corresponding question answering task requires the translation of the SQL query in natural language and the parsing of the output into a relation (2).**

1 INTRODUCTION

Declarative querying is recognized as the main feature behind the popularity of database systems. Users specify *what* they want to obtain, leaving the *how* to the system. 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 [7, 42]. 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 prepare extraction pipelines, which are typically static and can only extract fixed sets of attributes/tables. Indeed, the precise extraction of typed data in a coherent tuple format is a task still unsolved [1].

While declarative querying of text is a big challenge, there has recently been incredible progress in *question answering* (QA) over text [33]. In this setting, a question in natural language (NL) is answered by gathering information from large corpora of text documents. Transformers have enabled 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 [5, 29, 30], have been trained on large data, such as the entire Web textual content, and can answer complex questions in a closed-book fashion [32] (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 [22, 28], they are not trained to answer complex SQL queries and may fail short with such input.

In this work, 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 database that contains the information to answer the query. To ground our vision, we built Galois[1], a prototype that tackles this problem while aiming at preserving three main characteristics of SQL when executed over this new source of data: (i) queries are written in SQL over a user defined relational schema, despite the target LLM does not come with a schema; (ii) queries are *portable*: we enable any application to query with standard SQL existing pre-trained LLMs that expose textual prompts; (iii) answers are *correct* and *complete* w.r.t. the information stored in the LLM. Indeed, we focus on the correct execution of the queries

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

and do not argue that LLMs always contain complete and correct information. In contrast to generation of images, where small errors are unnoticeable by users in most cases, in querying any data error can be critical for the target application. While LLMs still make factual mistakes, we believe 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 improving over time [12, 36].

Our vision enables the execution of SQL queries to obtain relations from the information stored in LLMs, without having to manually rewrite the query as a NL question and then parsing the results into a structured format. This is a promising solution for several applications. In any domain-specific setting, such as medical or enterprise, data can be scattered across different modalities such as email, text, and PDF files. Their representation in a LLM enables information extraction queries, such as the retrieval of the proteins to treat a variant of a virus. Galois offers the ability to query data beyond what is already modeled in a structured schema. The data from the LLM can be used as a source in metadata inference [9], data integration [13], augmentation [44], imputation[24], and cleaning [26]. Looking forward, we envision the use of LLMs within a polystore system sitting on top of heterogeneous storage engines [2, 11]. We introduce an LLM interface compatible with such vision and show a path to combine traditional DBMSs and LLMs in novel hybrid query execution plans [18].

This paper describes how to query pre-trained LLMs and preliminary empirical evidence of its potential. 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 [20, 39]. To bridge the gap between a logical query plan and its execution on a LLM, we introduce new physical operators for the plan of the SQL query. Each of these new operators implements a textual prompt that is executed over the pre-trained LLM to gather the data.

Our contributions may be summarized in the following points:

- We introduce the problem of querying with SQL existing pre-trained LLMs with user-provided schemas. We present a prototype, Galois, that executes SPJA queries under assumptions that enable a large class of applications[2].
- 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. For this task, we introduce prompting methods that preserve traditional pipelining.
- We report results on 46 queries executed on top of four popular LLMs and show how their results compare w.r.t. the same queries executed on traditional DBMS. We also show that Galois's results are better than those obtained by manually rewriting the query (and parsing the results) in NL for question answering over the same LLM.
- We outline next steps and future research directions.

The remainder of this paper is organized as follows. Section 2 covers recent progress in natural language processing (NLP) and compares Galois to prior work. Section 3 discusses the challenges in querying LLMs. Section 4 describes the architecture of the first prototype. Section 5 reports preliminary experimental results from datasets in the Spider corpus. Section 6 discusses future research.

[2]Code available at https://gitlab.eurecom.fr/saeedm1/galois

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 in training [10]. The Transformer has now become the standard architecture in NLP. One of its benefits is its suitability for parallelization w.r.t. previous approaches, which has enabled the creation of massive pre-trained LLMs [5]. 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 a number of 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 for the capital cities in EU ("The EU state capitals are:").

Our effort is different from the problem of semantic parsing, i.e., the task of translating NL questions into a formal representation such as SQL [19, 41]. Our goal is also different from querying an existing relational database to answer a NL question [15]. Instead, we tackle the problem of querying 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, (ii) the textual result must be parsed into a relation, (iii) our experiments show that current LLMs in some cases fail in answering complex queries expressed as NL. 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 [31]. To overcome some of these limits, it has recently been shown that a series of intermediate reasoning steps ("chain of thought" and question decomposition [40]) improve LLMs' ability in complex tasks [39].

Our work is also different from the recent proposal for Neural DBs [37], 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.

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, as they are immediately available to all; (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 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 to distinguish the Rome city 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 [38].

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. We discuss in Section 6 the impact of prompt quality on the query results.

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 answers SQL queries. Moreover, new models keep increasing the factuality of their answers[3]. In this work, we do no 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.

[3]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 15th 2023

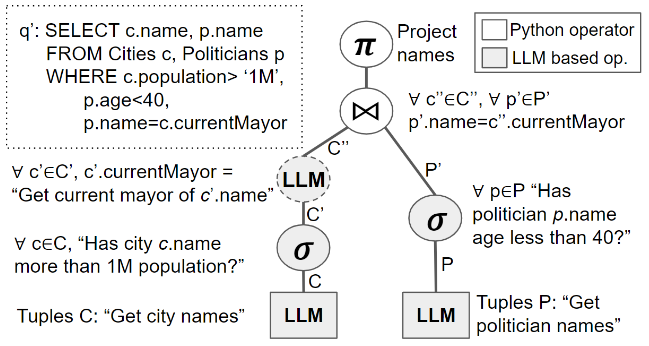

q': SELECT c.name, p.name
FROM Cities c, Politicians p
WHERE c.population> '1M',
p.age<40,
p.name=c.currentMayor
π
Project names
Python operator
LLM based op.
⋈
∀ c''∈C'', ∀ p'∈P'
p'.name=c''.currentMayor
C''
P'
∀ c'∈C', c'.currentMayor =
"Get current mayor of c'.name"
LLM
C'
σ
∀ c∈C, "Has city c.name more than 1M population?"
C
σ
∀ p∈P "Has politician p.name age less than 40?"
P
Tuples C: "Get city names"
LLM
LLM
Tuples P: "Get politician names"

Figure 2: Logical plan for query q' with notes about its execution. 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, such as *population* in the selection condition for *Cities*. The last two operators do not involve the LLM.

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 a 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 2. 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 [23]. Together, these two features make the LLM able to execute complex queries.

Prompts. Figure 2 shows how prompts, suitable for execution on LLMs, implement logical operators in Galois. In the figure, for each operator is reported a simplified prompt that is obtained automatically 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.

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 selections, joins and projections. 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 values in the attribute.
(4) Use traditional algorithms for any operator involving attributes that have already been retrieved.

In this minimalist yet general design, there are two critical steps that enable the practical utilization of Galois.

First, as relations can be large, we iterate with the a prompt until we stop getting new results. For example, we first ask for city names, we 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, either on the number of result tuples or on the cost to retrieve them.

A second practical issue is the cleaning of the data gathered from the LLM. In particular, numerical data can be retrieved in different formats. We normalize every string expressing a numerical value (say, 1k) into its expression as a number (1000). The enforcing of type and domain constraints is crucial to limit the incorrect output due to hallucinations of the model.

The present iteration of Galois is crafted to show queries that effectively retrieve relations from LLMs. This initial implementation serves as the experimental platform as reported in the next section.

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.

Dataset. Spider is a Text2SQL dataset with 200 databases, each with a set of SQL queries [41]. For each query, it provides the paraphrases of the SQL query as a NL question. We focus on a subset of 46 queries for which is reasonable to obtain answers from a LLM. Indeed, some queries are specific to the given Spider dataset, e.g., "How many heads of the departments are older than 56?". We use datasets about world geography and airports, e.g., "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**): fine-tuned T5 on a collection of datasets described via instructions (783M parameters).

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.

Figure 3: Instructions at beginning of GPT-3's prompt.

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

Table 1: Cardinality of Galois's results w.r.t. the ground truth relations $|R_D|$ for the 46 Spider queries. Closer to 0 is better.

	All	Selections only	Aggregates	Joins
R_M (SQL Queries)	0.50	0.80	0.29	0
T_M (NL Questions)	0.44	0.71	0.20	0.8

Table 2: Cell values matches (%) w.r.t. the ground truth results R_D for the 46 Spider queries. Averaged results for ChatGPT.

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 [27] (175B parameters). *GPT-3.5-turbo* (**ChatGPT**), the latest model available in the OpenAI API (175B parameters). We construct prompts appropriately for each model, we report the one for InstructGPT-3 in Figure 3.

For a given LLM M and a SQL query q with its Spider relation D and the corresponding NL question t, we obtain several 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. Only (b) uses the relations from Spider, (a) and (c) 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: the output is not structured data. For Galois, all 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, as reported in Table 1. We compute the ratio of the sizes as $f = \frac{|2*R_D|}{|R_D+R_G|}$, where 1 is the best result, and report the difference as percentage (averaged over all queries with non-empty results) with the formula 1-f. The 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 of the differences are explainable with errors in the results of the prompts across the query pipeline, as we discuss in more details next.

2) Content. Second, we measure the quality of the cell values in the results computed by all baselines. In this test, we compare the content of each cell value after manually mapping tuples between R_D on one side (ground truth) and (R_G, T_M) on the other side. As

T_M is NL text, we manually postprocess it (by splitting comma-seperated values and removing repeated values and punctutation) to extract the values as records. We consider a numerical value in (R_G, T_M) 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.

As we do not control the infrastructure used by OpenAI, we do not report API execution times. On average, **GPT-3** takes around 20 seconds to execute a query (about 110 batched prompts per query). The distribution for these metrics are skewed as they depend heavily 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 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, we 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 2, 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 a big challenge 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 [34, 45]. Finally, a key research direction is how to enable hybrid systems that can jointly query traditional storage and LLMs, e.g., when to trigger the more expensive LLM prompts, or how to combine intermediate results.

Knowledge of the Unknown. To overcome the problem of the results mixing real facts and invented tokens (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 [17].

Another direction is retrieval augmented language models, where they design modules that separate the "language understanding and reasoning" part and "factual knowledge" part [8]. Our prompting is a basic approach to surface facts, but more principled solutions are needed to obtain reliable results [16].

Provenance. Retrieval augmented models are also a promising direction to address the fact that LLMs do not cite the sources, or provenance [43], of their output. This is an issue, because it is not possible to judge correctness without the origin of the information. There are ongoing efforts on linking generated utterances, or values in our case, to sources [4]. This can also be done through the generation process or in a post-processing step [35].

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 [14].

Updates and Cost. 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. In the hybrid case, optimization will aim at minimizing the number of prompts calls. 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 [21]. One short term solution is to update LLMs without retraining [6, 25]. Looking at the long term, cost issues will be alleviated by the fact that training and inference are getting cheaper with better technology[4].

Coverage and Bias. LLMs focus on the common and probable cases by design. We found that, for some queries, the missing results are due to their lower popularity, compared to those the surfaced by the LLM. This is likely to get better as researchers focus more on this challenge [12, 36]. 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. If facts about an African country are missing from the English corpus, it is then impossible for the LLMs to capture them.

While they can miss facts, LLMs do encode biases and stereotypes that are present in observed human language. As humans are biased

[4]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

and use stereotyping, we must be careful when applying these models in real-world applications [3].

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] 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

[4] 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

[5] 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

[6] 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

[7] 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

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

[9] 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

[10] 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

[11] 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.

[12] 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

[13] 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.

[14] 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.

[15] 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

[16] 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

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

[18] 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

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

[20] 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).

[21] 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

[22] 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

[23] 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*.

[24] 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

[25] 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).

[26] 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

[27] 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).

[28] 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

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

[30] 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.

[31] 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

[32] 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

[33] 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.

[34] 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.

[35] 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

[36] 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

[37] 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.

[38] 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

[39] 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*.

[40] 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

[41] 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

[42] 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

[43] 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

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

[45] 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)*.