CAESURA: Language Models as Multi-Modal Query Planners

Matthias Urban
Technical University of Darmstadt

Carsten Binnig
Technical University of Darmstadt & DFKI

ABSTRACT

Traditional query planners translate SQL queries into query plans to be executed over relational data. However, it is impossible to query other data modalities, such as images, text, or video stored in modern data systems such as data lakes using these query planners. In this paper, we propose Language-Model-Driven Query Planning, a new paradigm of query planning that uses Language Models to translate natural language queries into executable query plans. Different from relational query planners, the resulting query plans can contain complex operators that are able to process arbitrary modalities. As part of this paper, we present a first GPT-4 based prototype called CAESURA and show the general feasibility of this idea on two datasets. Finally, we discuss several ideas to improve the query planning capabilities of today's Language Models.

1 INTRODUCTION

Query planning, the basic process of deriving an executable query plan in response to a user query, has conceptually stayed essentially the same since IBM's System R was introduced in 1974 [14]. In traditional DBMSs, a logical plan is first obtained from parsing a SQL query and then optimized by improving the order of the query's operators. In a second step, each logical operator is mapped to a concrete implementation to obtain a physical plan, which is eventually executed. In the past decades, research has focused primarily on improving the efficiency of query plans by improving various individual aspects (e.g. the cost model [5]).

However, this traditional approach to query planning is fundamentally limited in two important aspects: Firstly, it only applies to query languages such as SQL, where semantics of queries are clear and can be easily parsed into a (at least canonical) query plan to execute the query. Secondly, it is only possible to query structured relational data stored in tables. Due to these limitations and several new trends, we argue that it is finally time to re-think how query planning is done:

Trend 1: Multi-Modal Data. In today's industries, huge amounts of non-relational multi-modal data (e.g. images, documents, sensor data, ...) need to be stored and processed, which has led to solutions that store data outside classical databases. For instance, medical clinics need to store and analyze MRI scans and patient reports along with structured patient metadata. Since these data modalities cannot be stored and queried easily in today's databases, they are usually stored in data lakes, which, however, makes gaining insights from these modalities hard. To gain insights from such multi-modal data lakes, the data needs to be made accessible first, usually by manually constructing complex data processing pipelines.

 14th Annual Conference on Innovative Data Systems Research (CIDR '24). January 14-17, 2024, Chaminade, USA

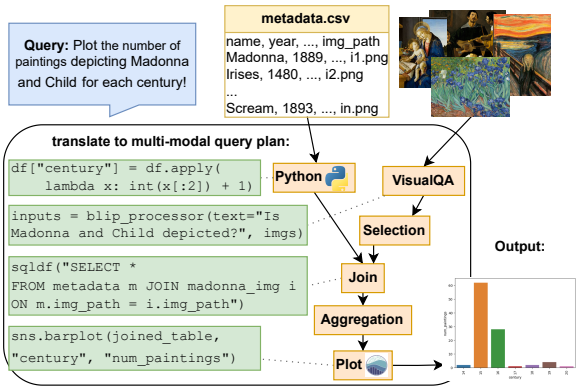

Figure 1: Example illustrating how a natural language query is automatically translated into a multi-modal query plan containing relational operators, machine generated Python UDFs, and a VisualQA Machine Learning model. The output is presented as a plot, making it easy and fast to gain insights from multi-modal data. The green boxes show code snippets that are executed when executing the plan.

While recently several pioneering systems have been proposed to ease the querying of multi-modal data in data lakes [3, 7, 16, 18, 19], these systems come with significant limitations. For instance, the queries supported by these systems are often limited in complexity (e.g. only simple queries with a single value as answer are supported [3], or non-relational modalities are only used as filters [7]), or they are limited to only a very few modalities [16, 19].

An ideal data system for multi-modal data, instead, would allow all types of queries and automatically construct the data processing pipelines, that have previously been constructed manually. This would allow users to formulate queries that combine information across modalities, and let them gain insights in a few minutes, for which they previously would have needed several days or weeks.

Trend 2: Natural Language Interfaces. SQL, with its declarative nature, has initially been designed to be understandable by laypersons. In practice, however, formulating complex queries in SQL requires profound knowledge of the language, making databases inaccessible to domain experts and management staff. Usually, these persons are required to interact with specifically trained data scientists to obtain insights from data stored in databases, a process that often requires many iterations.

Hence, in recent years, Natural Language Interfaces for databases have emerged, which would allow laypersons to query databases using natural language. However, existing approaches typically translate natural language into SQL [28, 29] and are therefore limited to relational data. Another direction are question-answering systems which work on modalities beyond tables [3]. However, question-answering systems only support queries that are much less expressive than what can be done with SQL.

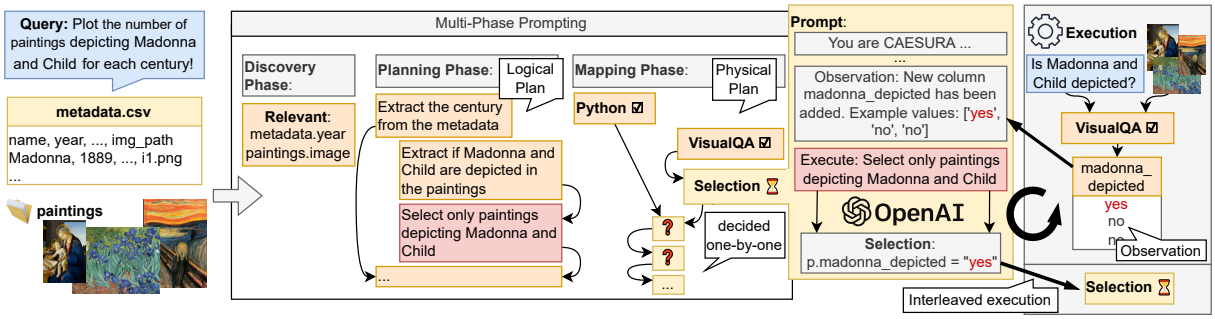

Figure 2: CAESURA transforms the query into a multi-modal query-plan using a series of prompts. In the *Discovery Phase*, the LLM is prompted to identify data items relevant for the query, such as relevant columns and datasets. In the *Planning Phase*, the LLM is prompted to construct a sequence of steps to satisfy the user request (Logical Plan). The final *Mapping Phase* is interleaved with *Execution*: a physical operator is chosen for each of the logical steps and executed incrementally. Once an operator is executed, we feed the results of the previous operator back to the LLM to choose the next operator (right). That allows the LLM to take the output of previous executions into account when choosing the physical operator and operator arguments (e.g. selection conditions as depicted in the figure) which avoids that faulty plans are generated.

Our Vision. In this paper, we thus present a vision of a data system that can be used by laypersons using natural language and can query arbitrary data modalities while enabling complex user queries way beyond classical SQL as shown in Figure 1. In the example, a user queries a data lake of a museum that stores both metadata (stored as a table) and pictures of artworks (stored as images) exhibited in the museum. To support such queries on multi-modal data, the natural language query must be translated into a complex processing pipeline that contains processing steps that can deal with multi-modal data. For instance, in Figure 1, the query is translated into a query plan that contains a *Python* operator that can run arbitrary Python code, and a *VisualQA* operator that extracts structured information from images. In particular, in the example, the *Python* operator extracts the century from a metadata column that stores the inception dates as strings and the *VisualQA* operator is used to select all pictures that depict *Madonna and Child*. An important aspect is that the result for user queries in our system can range from single values, over tables, to even a plot. Traditional query planners are clearly not able to come up with such a multi-modal query plan from a natural language query, since doing so requires **non-trivial reasoning over the user's intents, the available multi-modal data, as well as the effects of applying non-relational operators** to the data.

2 LANGUAGE MODELS AS QUERY PLANNERS

In order to enable our vision, we propose CAESURA[1], a novel query planner that leverages Large Language Models (LLMs) for compiling complex natural language queries over multi-modal data in executable plans. Recently, LLMs such as GPT-4 [11], LLaMA [17], and PaLM [4], have shown impressive results on various Natural Language Processing tasks such as translation, question answering, and reading comprehension. Most important in our context, they have also shown to possess impressive *reasoning capabilities* [2, 6, 20, 22, 27, 31, 34]. In fact, it has already been shown that LLMs can reason not only about the content of multi-modal data [10, 33, 33] but also about the effect of applying certain tools on the data sources [13, 15, 25, 25, 32], which enables e.g. a chatbot that can analyze an image that the user uploads [24]. However, to the best of our knowledge, no system has been proposed yet that operationalizes the reasoning capabilities of LLMs to construct complex query plans for multi-modal data lakes from natural language queries.

2.1 Multi-phase Query Planning

While LLMs provide reasoning capabilities as discussed before, it is still non-trivial to build a query planner that maps natural language user intents to executable query plans. As shown in Figure 2, in CAESURA we make use of a new multi-phase compilation strategy, which leverages carefully designed *prompts* that contain all the necessary information about (1) the multi-modal data sources, (2) the available operators and (3) the query, which allows the LLM to come up with a query plan. A major benefit of using prompts for query planning is that it is easy to plug in new operators (e.g. to process more modalities), as long as we provide all necessary information about their behavior in the prompt. Moreover, our query planner based on LLMs is composed of *multiple phases* which are based on the intuition of the phases of traditional query planning: first, by using a first prompt, the query planner maps the user query to a logical plan, containing a high-level step-by-step (textual) description of what needs to be done. Afterwards, using a separate prompt, each step is mapped to a concrete operator to obtain a physical plan that can be executed. As we will see in our experiments, CAESURA is thus indeed able to "reason" over user queries, data and available operators and can thus translate user queries into correct multi-modal query plans.

2.2 Challenges

Despite the capabilities of LLMs, there are many challenges when using LLMs for query planning. In the following, we discuss some of the main open research challenges for which we propose some ideas on how to tackle them in the following sections.

[1] https://github.com/DataManagementLab/caesura

Plan Executability. There are many causes why LLM-generated query plans might not be executable. For instance, the LLM might provide the wrong inputs to an operator (e.g. a collection of images as input for a traditional SQL selection). In these cases, plan execution will "crash" before the desired result can be computed. One option to fix a non-executable plan (which is integrated into CAESURA) is to use the LLM itself to fix the error by providing alternative plans as we discuss in Section 3.2. While we show that LLMs are thus often able to fix errors this way, this is by far not enough to guarantee that query plans are executable as expected by users.
Plan Correctness. Even if a query plan produced by an LLM is executable, there might still be "logical flaws" in the plan, which can lead to wrong query results. For instance, important steps (e.g., a join) might be missing. This is especially challenging since there is no feedback from the LLM on whether an executed plan is correct or not. One option is to let users inspect the final plan and let them decide whether they trust it or whether they would like to improve it. However, judging the correctness of such query plans can be difficult for laypersons. Another idea is to improve the reasoning capabilities of LLMs by fine-tuning them to avoid typical reasoning errors. See Section 5 for a discussion on this direction.
Plan Optimization. Finally, another important issue is that the plan generated by an LLM is not optimized, which is a problem since running non-optimized plans can result in huge runtime overheads. However, optimizing multi-modal query plans requires reasoning over the runtime of complex multi-modal operators, such as *VisualQA* or *Python*, which is non-trivial. One important component for optimizing multi-modal plans would be a learned cost model that captures the behavior of the multi-modal operators. We discuss some further ideas on query optimization for LLM-based query planning in Section 5.

3 OVERVIEW OF CAESURA

In essence, as discussed before, in CAESURA we orientate ourselves on the phases of traditional query planning and first generate a logical plan, which is afterwards translated to a physical plan. However, in contrast to logical plans in databases, logical plans of CAESURA consist of a description (in natural language) of the individual steps. An example of such a logical plan is shown in Figure 2 (below "Planning Phase"). Moreover, the physical plan contains operators that are very different from executable plans in databases. An example physical plan is shown in the same Figure (below "Mapping Phase").

3.1 Phases of Query Planning

Splitting the process of query planning into several phases allows us to tailor the prompts for query planning to the specific decisions of each phase. See Figure 2 for an overview of the three phases, which we elaborate in more detail in the following. In a nutshell, we first identify the relevant data sources, then in the planning phase we let the LLM generate the logical plan, and finally, in the mapping phase we let the LLM select the operators to obtain a physical plan.
Discovery Phase. In the first phase, we decide which data sources (e.g., in a data lake) provide relevant information for the current query. We only briefly describe this phase, because the focus of this paper is on query planning. In essence, CAESURA first narrows down the relevant tables, image collections, etc. using dense retrieval

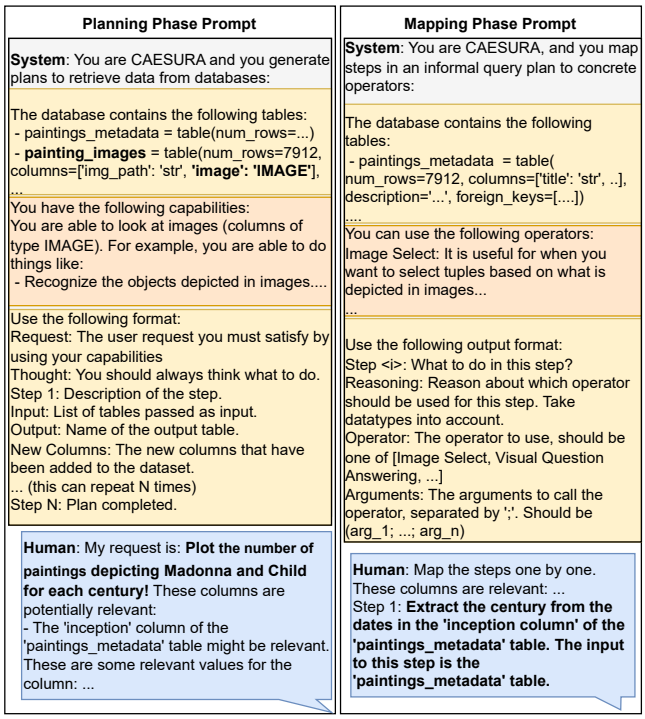

Figure 3: Example prompts for the Planning and Mapping Phase. Each prompt consists of two messages and contains all relevant information, e.g. data descriptions, operator/capability descriptions, etc. as well as an instruction telling the LLM what to do. In the planning phase, we additionally utilize in-context learning and provide a few example translations from query to logical plan for different domains at the very beginning of the prompt (not depicted).

(similar to Symphony [3]). Afterwards, for tabular data sources, we prompt the LLM to decide which columns of the retrieved data are relevant to the user query. The identified relevant data items are used to construct prompts for the next phases, e.g., to present the LLM with some relevant example values that help it to generate correct selection conditions.
Planning Phase. In the planning phase, which is at the core of CAESURA, the LLM is prompted to come up with a logical query plan that contains a natural language description of all steps necessary to satisfy the user's request. Figure 3 (left) shows an example prompt for this. The prompt consists of several parts: (1) a description of the data, (2) the capabilities of CAESURA, (3) an output format description, and (4) finally the user query and an instruction telling the model to come up with a plan. Notice how the multi-modal data is presented to the LLM: it is modeled as a special two-columned table where one column has the special datatype IMAGE. The capabilities of CAESURA describe the logical actions that CAESURA can take with the help of the available operators as can be seen in the example. Using this prompt, the LLM generates a stepwise (textual) plan which describes the logical plan in the output format specified in the input prompt. The generated stepwise plan is then parsed by CAESURA into a logical plan. Moreover, in order to improve the

quality of plans, we add a few examples of correct logical plans using few-shot prompting (not shown in the prompt in Figure 3). This helps to instruct the model to produce plans in the desired output format.

Mapping and Interleaved Execution. In the last phase, each previously determined logical step is mapped to a physical operator (and its input arguments) using a prompt similar to the one in Figure 3 (right). The prompt for this phase contains a short summary of the operators and what they can be used for, which is inspired by recent work where LLM agents can use external tools [24]. Moreover, different from traditional query planning, we do not decide on all the physical operators for all logical steps at once. Instead, we incrementally decide for each step and then execute it directly. For example, as shown in Figure 2, we first execute the *VisualQA* operator before we decide to use a SQL selection for the next step. This allows CAESURA to react on the results returned by previous operations, which leads to more plans that are in fact executable. For instance, in Figure 2 after executing the *VisualQA* operator, the LLM is able to construct a correct selection condition for the next step in the plan, based on the resulting values ("yes" and "no").

3.2 Error Handling

With CAESURA several errors can occur during query planning. In a nutshell, to deal with errors in CAESURA , we use the LLM for error handling by adding the error message to the prompt and asking the LLM to fix the error. However, this comes with the challenge that the root cause of an error is not known. In particular, the root cause can also lie in any previously executed phase. For instance, in the planning phase, the LLM could have decided to filter by a non-existent column, but the mistake is only noticed after choosing an operator and executing the step.

To fix such errors, we thus use the LLM to identify in which phase the error occurred, backtrack to it, fix the error, and rerun the subsequent phases. For this purpose, we use an additional prompt containing a set of questions that encourage the LLM to reason about the error such as: (1) What are the potential causes of this error? (2) Explain in detail how this error could be fixed. (3) Is there a flaw in my plan (Yes/No)? (4) Is there a more suitable alternative plan (Yes/No)? (5) Should a different tool be selected for any step (Yes/No)? (6) Do the input arguments of some of the steps need to be updated (Yes/No)? Parsing the responses to questions (3) + (4) allows us to determine whether to backtrack to the planning phase or if the mistake happened during the mapping phase. To finally fix the error, the ideas from questions (1) + (2) and the original error message are added to the prompt to which we backtracked to, before it (and potentially subsequent phases) is executed again. While this procedure allows CAESURA to fix non-executable plans in many cases, it clearly does not guarantee plan executability or even plan correctness. See Section 5 for further ideas on these issues.

4 INITIAL RESULTS

In our experiments, we are primarily interested in whether CAESURA is able to construct correct query plans. Our prototype of CAESURA has access to four multi-modal operators: (1) *VisualQA* based on *BLIP-2* [9], (2) *TextQA* based on *BART* [8], (3) *Python UDFs*, and (4) *Image Select*, which selects images based on a description and is also based on BLIP-2. It also has access to all *relational operators* supported by SQLite and a plotting operator based on *seaborn* [21].

Datasets. Since there does not yet exist a benchmark for the scenarios we envision for CAESURA, we constructed two multi-modal datasets. (1) The *artwork* dataset (with tables and images) resembles the example from Figures 1 and 2. The dataset contains a table about painting metadata as well as an image collection containing images of the artworks. We use Wikidata to construct both the metadata table as well as the image corpus: for the metadata table, we extract title, inception, movement, etc. for all Wikidata entities that are instances of 'painting'. (2) The second dataset is the *rotowire* dataset (with tables and text) [23] which consists of textual game reports of basketball games, containing important statistics (e.g. the number of scored points) of players and teams that participated in each game. We extend the textual reports by two tables for players and teams constructed from Wikidata. These contain general information, such as name, conference, division, etc. for every team, and name, height, nationality, etc. for every player.

4.1 Anecdotes

Before we measure the system on a broad set of queries, we first highlight a set of correctly translated queries to illustrate that using LLMs for query planning is indeed promising. We present two query plans obtained by translating two queries using CAESURA, one for each dataset. We use GPT-4 [11] as LLM for this experiment.

Query 1 (on rotowire): For every team, what is the highest number of points they scored in a game? This query involves a join, usage of the TextQA operator as well as an aggregation. Figure 4 (left) shows that CAESURA was able to come up with a correct logical plan, that it then correctly translates into a physical plan. The input arguments are chosen correctly as well. Perhaps most impressive is that CAESURA correctly uses the TextQA operator, which takes question templates as inputs. During execution, these templates are instantiated by the operator using the values from the input table to generate questions like "How many points did Heat score?", which it then answers for all reports. This allows the operator to separately extract the points scored by each team.

Query 2 (on artwork): Plot the maximum number of swords depicted on the paintings of each century. This second and more complex query on the artwork data requires the inspection of the images, as well as the visualization of the results in the end. Again, CAESURA is able to correctly translate this query using two multi-modal operators: Python and VisualQA followed by a plot operator, as can be seen in Figure 4 (right).

4.2 Plan Quality

Next, we evaluate CAESURA on a larger set of queries, 24 for each dataset. For this experiment, we are interested in the query planning abilities of LLMs. Hence, we skip the data discovery step and assume perfect retrieval (to not measure retrieval performance).

The queries used in this experiment are clustered along several aspects. In total, we have 16 queries asking for a single result value, 16 that ask for an output table, and 16 that ask for a plot. Moreover, half of the queries require multi-modal data while the other half require only relational data. Importantly, these queries were not used for tuning the prompts during the development. Table 1 shows

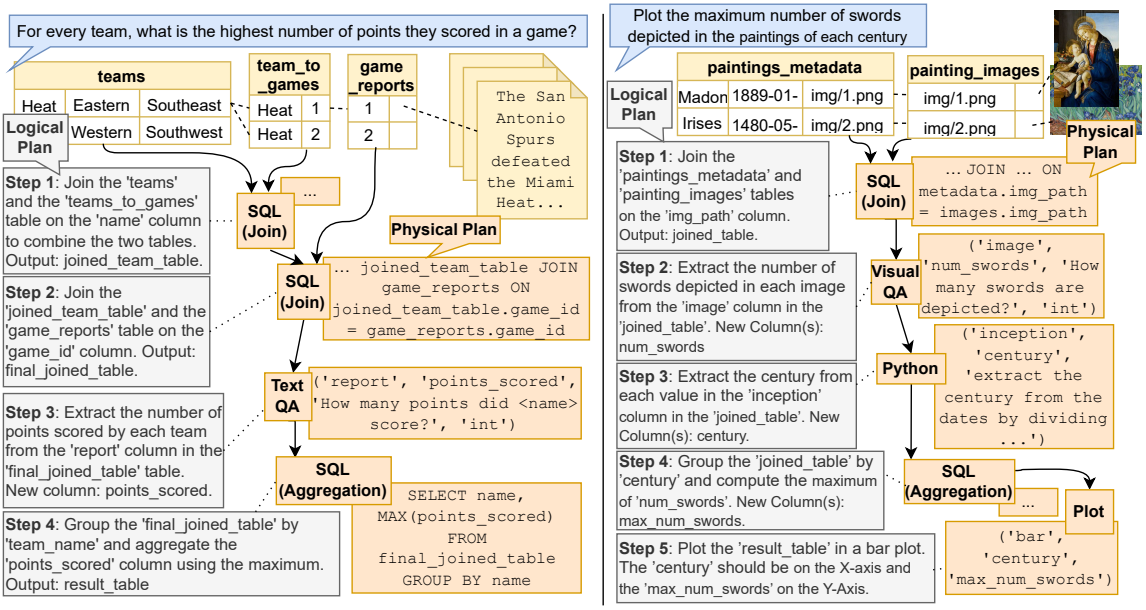

Figure 4: CAESURA using GPT-4 is able to correctly translate the user queries to multi-modal query plans that contain TextQA, VisualQA and Python operators. The final physical plan, including input arguments for the operators, is shown in orange for both queries. For each operator, we also show the corresponding step of the GPT-4 generated logical plan (in grey). CAESURA presents image and text collections as special tables (*game_reports* with columns *game_id* and *report*; *painting_images* with columns *img_path* and *image*) to the LLM, s.t. they can be the input to a regular join. The plans are presented as-is and are not optimized. The TextQA operator takes a question template as input, which is translated to questions by inserting different team names from the values in the table. The Python operator takes a description as input, which is translated to code using GPT-4.

the accuracies for the different query groups using ChatGPT-3.5 and GPT-4 as LLM.

We see that CAESURA using GPT-4 is better than ChatGPT-3.5 and is even able to correctly translate 87.5%[2] of queries despite never being fine-tuned on the queries. The approach works especially well on the artwork dataset, where CAESURA is able to translate all queries to correct query plans. However, we also see that there is still room for improvement. In particular, on the rotowire dataset, which consists of more tables and contains texts instead of images, only three-quarters of queries could be translated correctly. In the next experiment, we analyze the mistakes CAESURA makes when generating query plans.

4.3 Error Analysis

In Table 2, we categorize the errors and show the frequency of the errors for different categories. We found that the possible errors are quite diverse: sometimes wrong physical operators were chosen or important steps were missing in the final plan (e.g. CAESURA forgot to join).

The most common mistake for CAESURA powered by GPT-4 was that it chose the wrong input arguments for physical operators (e.g., wrong parameters for SQL, a wrong question for QA, usage of non-existent column names). For GPT-4, this happened for 3 out of the 48 queries. For the smaller model, ChatGPT-3.5, we see that it had some more problems understanding the data correctly

[2]plan correctness is decided by hand by a human observer

Models	ChatGPT-3.5		GPT-4	
Plan type	logical	physical	logical	physical
Artwork overall	79.2%	70.8%	100%	100%
Rotowire overall	50.0%	41.7%	87.5%	75.0%
Single modality	79.2%	75.0%	100%	92.7%
Multiple modalities	50.0%	37.5%	87.5%	83.3%
Single value	75.0%	62.5%	100%	93.8%
Table	68.8%	62.5%	87.5%	81.3%
Plot	50.0%	43.8%	93.8%	87.5%
All	64.6%	56.2%	93.8%	**87.5%**

Table 1: Correctly translated plans for the different datasets, modalities, and output formats. We show the percentage of correctly generated logical plans, as well as physical plans.

(see category *Data Misunderstanding*), a mistake that happened only once with GPT-4. In particular, ChatGPT-3.5 often tried to extract what is depicted in the image based on the title or the genre column of the metadata table. Thus, it often avoided the usage of multi-modal operators and instead tried to solve everything using SQL, resulting in flawed plans.

Interestingly, there was one query on the rotowire dataset that both models could not translate at all: *How many games did each team lose?* We speculate that this is the case because the query sounds simple and does not convey the operations necessary to

Category		ChatGPT-3.5	GPT-4
Impossible Actions	logical	4	2
Data Misunderstanding	logical	9	1
Illogical / Missing Steps	logical	3	0
Wrong Arguments	physical	3	3
Wrong Tool	physical	1	0

Table 2: Number of specific kinds of mistakes CAESURA made during query planning. In the upper three categories the mistake occurred in the planning phase (i.e. wrong logical plan), and for the lower two the mistake occurred in the mapping phase. We see that the older model often does not understand the data correctly (e.g. it tries to determine what is depicted on a painting based on its title).

answer it. One possibility to answer this query would be to join the teams table and the game reports, use the TextQA operator to ask the question "Did <name> lose?", and then aggregate the losses. Unfortunately, ChatGPT-3.5 tried to solve it using a single SQL query ignoring the text completely. GPT-4, instead, was aware to use the text but it was not able to generate the correct operator for extracting the required information from the text.

5 RESEARCH DIRECTIONS

We have seen that using today's state-of-the-art language models such as GPT-4 together with careful prompting yields promising results for multi-modal query planning. However, there is still an abundance of interesting open challenges to be solved. In particular, query planning is expected to yield correct and efficient plans, which cannot be guaranteed when LLMs are utilized. In this Section, we explain our ideas on how these challenges could be overcome.

Plan Executability and Correctness. While the reasoning capabilities of GPT-4 and similar LLMs are already impressive, they still make mistakes (see Table 2). In this regard, there are already first works that improve the reasoning capabilities of today's LLMs [1, 26, 31], and it remains to be seen if these improvements are translated to improved query planning. Nevertheless, we speculate that it might not be enough to meet the strict quality constraints on query plans. One interesting idea to push the data-reasoning capabilities of LLMs is to construct a fine-tuning dataset for query planning, similar to SPIDER [30]. SPIDER is a text-to-SQL dataset that boosted research on semantic parsing. A similar fine-tuning dataset for multi-modal query planning could lead to comparable advancements in this field. Such a fine-tuning dataset comes with several additional benefits on top of better reasoning skills and higher-quality plans. Most importantly, it could be used to fine-tune smaller, open-source models, which resolve any privacy issues currently present when using external models via an API. Moreover, the use of smaller LLMs would also reduce the computational (and monetary) cost of CAESURA.

Plan Optimization. In practice, there is also the need to generate runtime efficient query plans. However, optimizing the resulting multi-modal query plan is far from trivial, since it requires reasoning over the runtime behavior of multi-modal operators. This behavior can be hard to predict. For instance, the execution of a Python operator can lead to vastly different runtimes depending on the Python code that is executed by the operator. While there are already systems that are able to (partially) optimize UDFs in SQL queries [12], they usually do not consider multi-modal data and the use of powerful Machine Learning models. We believe an important step towards optimizing such multi-modal query plans is to learn cost models for multi-modal operators. There is already a rich line of work for learned cost models (e.g. [5]). However, so far these only capture the cost for traditional database operators and not the complex operators we consider in this paper.

Security. Since we only have limited control over what is generated by an LLM, the LLM could theoretically generate malicious or destructive code to be executed over our data. In our current prototype, we therefore limit e.g. generated SQL code to only SELECT statements and prevent running UPDATE, INSERT or DELETE statements that could maliciously manipulate data. While we did not observe such behavior during the experiments, a more extensive analysis is necessary to rule out such concerns.

6 THE ROAD AHEAD

Gaining insights from multi-modal data is a difficult endeavor because it usually involves the manual creation of complex processing pipelines. Hence, we present CAESURA, a Language-Model-driven query planner that generates complex processing pipelines automatically from queries in natural language. While CAESURA shows first promising results, there are still a plethora of open challenges as discussed before. In particular, today's language models suffer from reasoning difficulties and hallucinations, leading to query plans that crash or return wrong results, as well as plans that are sub-optimal in terms of runtime.

ACKNOWLEDGMENTS

We thank the reviewers for their feedback. This research is funded by the Hochtief project *AICO* (AI in Construction), by the BMBF and the state of Hesse as part of the NHR Program, as well as the HMWK cluster project *3AI* (The Third Wave of AI). Finally, we want to thank hessian.AI at TU Darmstadt as well as DFKI Darmstadt.

REFERENCES

[1] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2023. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. https://doi.org/10.48550/arXiv.2308.09687 arXiv:2308.09687 [cs]

[2] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of Artificial General Intelligence: Early Experiments with GPT-4. https://doi.org/10.48550/arXiv.2303.12712 arXiv:2303.12712 [cs]

[3] Zui Chen, Zihui Gu, Lei Cao, Ju Fan, Sam Madden, and Nan Tang. 2023. Symphony: Towards Natural Language Query Answering over Multi-modal Data Lakes. (2023).

[4] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr

Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. PaLM: Scaling Language Modeling with Pathways. *arXiv:2204.02311 [cs]* (April 2022). arXiv:2204.02311 [cs]

[5] Benjamin Hilprecht and Carsten Binnig. 2022. Zero-Shot Cost Models for out-of-the-Box Learned Cost Prediction. *Proc. VLDB Endow.* 15, 11 (July 2022), 2361–2374. https://doi.org/10.14778/3551793.3551799

[6] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2022. Inner Monologue: Embodied Reasoning through Planning with Language Models. arXiv:2207.05608 [cs]

[7] Saehan Jo and Immanuel Trummer. 2023. Demonstration of ThalamusDB: Answering Complex SQL Queries with Natural Language Predicates on Multi-Modal Data. In *Companion of the 2023 International Conference on Management of Data*. ACM, Seattle WA USA, 179–182. https://doi.org/10.1145/3555041.3589730

[8] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv:1910.13461 [cs, stat]* (Oct. 2019). arXiv:1910.13461 [cs, stat]

[9] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. https://doi.org/10.48550/arXiv.2301.12597 arXiv:2301.12597 [cs]

[10] Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-Play Compositional Reasoning with Large Language Models. arXiv:2304.09842 [cs]

[11] OpenAI. 2023. GPT-4 Technical Report. https://doi.org/10.48550/arXiv.2303.08774 arXiv:2303.08774 [cs]

[12] Astrid Rheinländer, Ulf Leser, and Goetz Graefe. 2018. Optimization of Complex Dataflows with User-Defined Functions. *ACM Comput. Surv.* 50, 3 (May 2018), 1–39. https://doi.org/10.1145/3078752

[13] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. arXiv:2302.04761 [cs]

[14] P. Griffiths Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. 1979. Access Path Selection in a Relational Database Management System. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data (SIGMOD '79)*. Association for Computing Machinery, New York, NY, USA, 23–34. https://doi.org/10.1145/582095.582099

[15] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. HuggingGPT: Solving AI Tasks with ChatGPT and Its Friends in Hugging Face. arXiv:2303.17580 [cs]

[16] James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Halevy. 2021. From Natural Language Processing to Neural Databases. *Proc. VLDB Endow.* 14, 6 (Feb. 2021), 1033–1039. https://doi.org/10.14778/3447689.3447706

[17] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. https://doi.org/10.48550/arXiv.2302.13971 arXiv:2302.13971 [cs]

[18] Giovanni Trappolini, Andrea Santilli, Emanuele Rodolà, Alon Halevy, and Fabrizio Silvestri. 2023. Multimodal Neural Databases. https://doi.org/10.1145/3539618.3591930 arXiv:2305.01447 [cs]

[19] Matthias Urban and Carsten Binnig. 2023. Towards Multi-Modal DBMSs for Seamless Querying of Texts and Tables. https://doi.org/10.48550/arXiv.2304.13559 arXiv:2304.13559 [cs]

[20] Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023. Describe, Explain, Plan and Select: Interactive Planning with Large Language Models Enables Open-World Multi-Task Agents. arXiv:2302.01560 [cs]

[21] Michael L. Waskom. 2021. seaborn: statistical data visualization. *Journal of Open Source Software* 6, 60 (2021), 3021. https://doi.org/10.21105/joss.03021

[22] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. *arXiv:2201.11903 [cs]* (April 2022). arXiv:2201.11903 [cs]

[23] Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in Data-to-Document Generation. *arXiv:1707.08052 [cs]* (July 2017). arXiv:1707.08052 [cs]

[24] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models. arXiv:2303.04671 [cs]

[25] Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. 2023. MM-REACT: Prompting ChatGPT for Multimodal Reasoning and Action. arXiv:2303.11381 [cs]

[26] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. arXiv:2305.10601 [cs]

[27] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629 [cs]

[28] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. https://doi.org/10.48550/arXiv.2005.08314 arXiv:2005.08314 [cs]

[29] Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2021. GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing. *arXiv:2009.13845 [cs]* (May 2021). arXiv:2009.13845 [cs]

[30] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2019. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. arXiv:1809.08887 [cs]

[31] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. STaR: Bootstrapping Reasoning With Reasoning. https://doi.org/10.48550/arXiv.2203.14465 arXiv:2203.14465 [cs]

[32] Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. 2022. Socratic Models: Composing Zero-Shot Multimodal Reasoning with Language. https://doi.org/10.48550/arXiv.2204.00598 arXiv:2204.00598 [cs]

[33] Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. 2023. Multimodal Chain-of-Thought Reasoning in Language Models. arXiv:2302.00923 [cs]

[34] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. arXiv:2205.10625 [cs]