

BLENDSQL: A Scalable Dialect for Unifying Hybrid Question Answering in Relational Algebra

Parker Glenn, Parag Pravin Dakle, Liang Wang, Preethi Raghavan

Fidelity Investments, AI Center of Excellence
{parker.glenn, paragpravin.dakle,
liang.wang, preethi.raghavan}@fmr.com

Abstract

Many existing end-to-end systems for hybrid question answering tasks can be boiled down to a “prompt-and-pray” paradigm, where the user has limited control and insight into the intermediate reasoning steps used to achieve the final result. Additionally, due to the context size limitation of many transformer-based LLMs, it is often not reasonable to expect that the full structured and unstructured context will fit into a given prompt in a zero-shot setting, let alone a few-shot setting. We introduce BLENDSQL, a superset of SQLite to act as a unified dialect for orchestrating reasoning across both unstructured and structured data. For hybrid question answering tasks involving multi-hop reasoning, we encode the full decomposed reasoning roadmap into a single interpretable BLENDSQL query. Notably, we show that BLENDSQL can scale to massive datasets and improve the performance of end-to-end systems while using 35% fewer prompt tokens. Our code is available and installable as a package at github.com/parkervg/blendsql.

1 Introduction

Problem Decomposition In settings involving both fine-tuning and in-context learning, it has been demonstrated that generating explicit intermediate steps for complex problems can enhance accuracy. In their aptly titled “Scratchpad” framework, Nye et al. (2021) revealed that predicting the complete execution trace led to better results when fine-tuning language models to generate execution output. More recent work has shown that prompting LLMs to decompose complicated problems into smaller reasoning steps before generating the final answer can improve results (Wei et al., 2022; Yao et al., 2023; Wang et al., 2022). However, at their core, these methods use natural language as their intermediate reasoning representation. With BLENDSQL, we argue that natural language alone is inherently a lossy and ambiguous intermediate



Figure 1: Example BLENDSQL representation for an OTT-QA dev example.

reasoning representation. Instead, by decomposing a problem into a SQL-like syntax, we allow for more deterministic reasoning capabilities and better interpretability of intermediate steps by leveraging the compositional nature of relational algebra (Codd, 1970).

Agents Previous works have explored agents: an experts-based framework where the main task is broken down into parts that each agent or expert¹ carries out (Parisi et al., 2022; Liang et al., 2022; Schick et al., 2023; Shen et al., 2023; Liang et al., 2023; Lu et al., 2023). Furthermore, these works are based on the presence of a ‘routers’ or ‘planners’ that understand the given task and break it down into smaller tasks. Although rule-based routers have been widely used, the ability of LLMs to generalize to complex texts has enabled their use as

¹Here an agent or an expert can be anything from a Python script, an API call, a tool or a language model.

planners. LLMs, as planners, process the input and as output generate text containing the list of sub-tasks (Shen et al., 2023), the agent(s) to use from a given set, what are the inputs to the agent(s) (Schick et al., 2023; Qin et al., 2023; Cheng et al., 2023), what is expected as response from the agent(s) (Parisi et al., 2022). With the scope of the tasks carried out by the agents being very narrow, recent works have primarily focused on the planner component of the framework. For the same, harnessing the in-context learning capabilities of the LLM (Shen et al., 2023; Qin et al., 2023) or fine-tuning the planner LLM (Parisi et al., 2022; Schick et al., 2023; Patil et al., 2023; Tang et al., 2023; Wang et al., 2023; Lu et al., 2023) have been the two main methods. Additionally, works have used LLMs’ in-context learning capability for dataset creation (Patil et al., 2023; Tang et al., 2023). The parser in BLENDSQL acts as a planner and generates the APIs or Ingredients to use in answering the input question. However, it differs from the previous works as it goes beyond generating a single string representation of complicated problem decomposition and simplifies the agent framework using robust and complicated abstract syntax tree (AST) structures to be encoded into a single string. It is also important to note that BLENDSQL does not just rewrite the input statement using formal constructs (Lu et al., 2021; Wu et al., 2022), it also adds information using the reference database, input prompt, and internal knowledge of the language models.

Text-to-SQL On the widely used text-to-SQL dataset Spider (Yu et al., 2018), many works demonstrate impressive performance in the few-shot or zero-shot setting (Gao et al., 2023; Pourreza and Rafiei, 2023; Dong et al., 2023; Liu et al., 2023). Even in settings where text-to-SQL is not the focus, existing work details the effectiveness of a common and well-understood intermediate representation like SQL for other reasoning tasks. Hu et al. (2022) frames dialogue state tracking (DST) as a text-to-SQL task by encoding domains and slots from MultiWOZ (Budzianowski et al., 2018) into a serialized database format and using SQL as an intermediate representation, showing significant improvement over the “traditional” key-value style prediction format.

Hybrid Question Answering Unlike text-to-SQL, hybrid question answering involves an im-

PLICIT decision to access tabular data, unstructured data, or both (Chen et al., 2020; Zhu et al., 2021; Chen et al., 2021). Li et al. (2021) make this more explicit by routing between end-to-end models and a parser that generates an intermediate SQL representation. Their implementation frames the two as isolated, specialized systems which are unable to pass information to one another.

For the scope of this work, we define *Hybrid Question Answering* as answering questions using a corpus of tabular and unstructured multi-modal content. While this multi-modal content can encompass text, images, audio, and video, the primary focus of this paper is on unstructured text in conjunction with tabular data. The potential to adapt this process to other formats, such as images, is also discussed in Appendix C.

One prior work, UniK-QA (Oguz et al., 2022), proposes converting structured tabular data into unstructured text data. Our work does the opposite, and instead converts all unstructured text data into tabular data. Empirically, UniK-QA improves on several knowledge-base (KB) QA benchmarks using a KB-to-text preprocessing model. For example, for the KB-to-text model mentioned in the UniK-QA paper, the relationship between different entities is assumed to have a simple triplet representation: <subj, pred, obj> where pred (predicate) defines the relationship from the table name. However, relationships between <subj> and <obj> are often complicatedly embedded in heterogeneous information sources (e.g., audio, documents, or column names of the table), which would then demand more efforts in developing dedicated KB-to-text models.

Neuro-Symbolic Frameworks Most similar to our work is the BINDER method presented in Cheng et al. (2023), which integrates LLM reasoning into symbolic languages like SQL and Python. While we take inspiration from their neuro-symbolic framework for reasoning, BLENDSQL differs in the following ways. (1) We create an API specialized for SQL, enabling reasoning over multi-table databases containing unstructured content, and are capable of executing and optimizing queries containing JOIN statements, aliases, conditional table expressions, and most other SQLite operations (2) We enable constrained decoding according to the database schema (3) We design our API such that users are able to easily create any number of custom functions themselves to use

within a BLENDSQL script.

1.1 Main Contributions

In summary, we make the following contributions.

- To the best of our knowledge, we are the first to propose framing the context of hybrid question answering as a relational database.
- We introduce a new open-source query language, BLENDSQL, to orchestrate and optimize hybrid functions across SQL logic and LLM reasoning.
- We demonstrate that with only a small number of in-context exemplars, BLENDSQL can outperform end-to-end methods using 35% fewer tokens and without direct access to unstructured context.

2 BLENDSQL: Overview

We implement BLENDSQL as a superset of the SQLite relational database management system (RDBMS)². An executed BLENDSQL script returns a `smoothie` object as output, containing the final result and intermediate reasoning steps taken. To make a `smoothie`, we need some ingredients and a blender. We describe these components below.

2.1 Blender vs. Parser

We use the term “blender” to describe the LLM which receives the prompts used to perform each ingredient function within a BLENDSQL script. Prior to execution, the “parser” receives a set of few-shot examples and generates a BLENDSQL query given a question and database context. We use GPT-4-0613 (Achiam et al., 2023) as both the blender and the parser for our core experiments.

2.2 Ingredients

Ingredients are at the core of a BLENDSQL script. They are subprograms used to pass certain logical operations through LLM-based functions, and are denoted by double-curly braces (“{ { ” and “} } ”). BLENDSQL syntax is represented as a parsing expression grammar, implemented via PyParsing³ (McGuire, 2007). Like vanilla SQL, ingredient calls are fully recursive, and the context passed to

one can include operations invoking other BLENDSQL ingredients.

LLMMap The *LLMMap* ingredient is a unary scalar function, much like *LENGTH* or *ABS* in standard SQLite. The output of this operation is then set as a new column in a temporary table, for later use within the wider query. Specifically, it accepts table and column arguments in the string format “*table::column*”. It then creates a new column c' with mapped values v' , which are the output of the scalar function. With n_c as the number of values in the target column c , we get the following.

$$\mathcal{V} = \{v_i, \forall_{i \in \{1, \dots, n_c\}}\}$$
$$f_{map}(\mathcal{V}) \rightarrow \{v'_1, \dots, v'_{n_c}\}$$

For example, Figure 2 depicts the *LLMMap* ingredient in action. For each value, an LLM is prompted to answer the question: “*Is this a team event?*” Given the diversity in how team events are denoted in this column (“team event”, “4x100 medley relay”, etc.), it becomes difficult to write SQLite logic for a generalized solution across the entire table via pure string parsing. *LLMMap*, instead, implements a generalized solution to transform values based on their semantic denotations.

LLMQA The *LLMQA* ingredient is an aggregate function that transforms a table subset \mathcal{T} into a single scalar value v' .

$$f_{qa}(\mathcal{T}) \rightarrow v'$$

As shown in Figure 1, this ingredient may be restricted with the *options* argument, which ensures that the output of the function will be an existing value in a specific column c , $v' \in \{v_i, \forall_{i \in \{1, \dots, n_c\}}\}$ ⁴. In Figure 2, the *LLMQA* function receives a piece of unstructured document context, and the question: “*Which NBA season was suspended due to COVID-19?*” Additionally, it receives a set of column values to use in constraining the language model’s generation.

For fact verification tasks, we implement a modified version of this ingredient, *LLMValidate*, which always returns either a *true* or *false* boolean.

LLMJoin The *LLMJoin* ingredient creates a custom mapping between a pair of value sets V_1 and

²<https://www.sqlite.org/>

³<https://github.com/pyarsing/pyarsing>

⁴This constrained decoding is implemented using `guidance`, which manipulates the `logit_bias` argument in the `OpenAI API`.

name	position	school
joshua fields	right-handed pitcher	university of georgia
dennis raben	outfielder	university of miami
matt jensen	second basemen	clovis east high school

title	content
kenn kasperek	kenn anthony kasperek (born 1985-9-23) is...
pitcher	in baseball , the pitcher is the player who throws the baseball from...
josh fields (pitcher)	joshua david fields (born 1985-8-19) is an american professional baseball pitcher...

Table 1: Example database configuration from a HybridQA example, aligned with 3. The top table w contains the structured data corresponding to a table found on a Wikipedia page, while the bottom table $documents$ contains unstructured data found in the article content of a Wikipedia page, indexed with FTS5⁶.

V2. This mapping is then used to create an auxiliary table to carry out a SQL *INNER JOIN* operation.

$$f_{join}(\mathcal{V}1, \mathcal{V}2) \rightarrow \{(v1'_1, v2'_1), \dots, (v1'_{n_v,1}, v2'_{n_v,2})\}$$

This ingredient is particularly useful in situations when proper foreign keys do not exist, but some semantic alignment is still possible. In Figure 2, “joshua fields” and “josh fields (pitcher)” share the same referent. We use the *LLMJoin* ingredient to align the two distinct references, effectively performing a form of entity linking.

2.3 Query Optimizations

When possible, we execute all SQL predicates in a subquery first and assign their outputs to a temporary session table. Since native SQLite operations are relatively inexpensive, we ensure that the expensive LLM-based ingredient functions receive no more (and no less) the subset of data required to generate a faithful execution output. As demonstrated below, we only pass the subset of rows from w where $w.school = 'university of georgia'$ to the LLM-based ingredients in 3. We validate these optimizations with an extensive test suite⁵.

3 Datasets

Due to the joint usage of SQL-like syntax and LLM-based ingredients, we expect our BLEND-

⁵<https://github.com/parkervg/blendsql/tree/main/tests>

SQL to perform well on hybrid question answering datasets. To validate our hypothesis, we experimented on several challenging benchmark datasets requiring reasoning over both structured and unstructured knowledge.

3.1 HybridQA

Compared to previous QA tasks utilizing only passages (unstructured) or tabular information (structured), HybridQA contains challenging questions whose answers demand heterogeneous forms of information collected from Wikipedia tables and passages (Chen et al., 2020). In an ablation study, the original authors show that a table-only model achieves an accuracy of 8.4 on the dev set, and a passage-only model achieves an accuracy of 19.5. Combining both sources, their baseline model improves drastically to 44.0, highlighting the importance of a model adept at combining both hybrid data sources.

3.2 OTT-QA

The Open Table-and-Text Question Answering dataset (OTT-QA) is a “decontextualized” variant of HybridQA, which requires both text and table retrieval over a large corpus (Chen et al., 2021). On the OTT-QA dataset, we demonstrate the potential of BLENDSQL to act as both the retriever and reader within a unified query language. In the multi-hop reasoning OTT-QA requires, the relevant passages can only be found after completing some operation on the structured tables, and vice versa. We enable this communication between the reader and retriever components via BM25 full-text search enabled with the built-in FTS5 extension⁷ in SQLite. For example, in Figure 1, we first retrieve unstructured text from the $documents$ table with the highest BM25 relevancy to the query “*nba OR covid*”. Then, given the structure of the table “*Lebron James Career Statistics*”, we constrain the possible generations given our unstructured context to only a value occurring in the “*Year*” column.

3.3 FEVEROUS

The FEVEROUS (Fact Extraction and VERification Over Unstructured and Structured information) dataset contains claims accompanied by context sentences and tables from Wikipedia (Aly et al., 2021). Each claim is classified as either “supports”, “refutes”, or “not enough info”. By using BLEND-

⁷<https://www.sqlite.org/fts5.html>

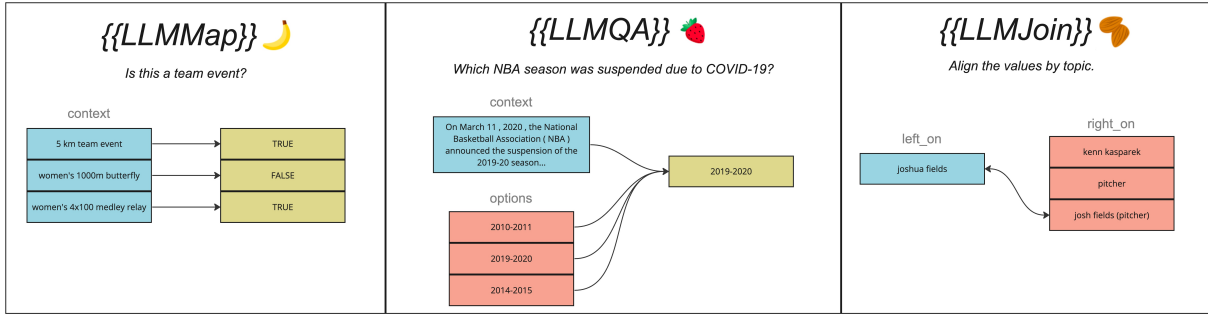


Figure 2: Visualizing built-in BLENDSQL ingredients.

```

{{
  LLMQA(
    'Which teams drafted this player?',
    (
      SELECT
        documents.title AS 'Player',
        documents.content
      FROM documents
      JOIN {{
        LLMJoin(
          left_on='w::name',
          right_on='documents::title'
        )
      }}
      WHERE w.school = 'university of georgia'
    )
  )
}}

```

Figure 3: BLENDSQL for “Which teams has the player drafted by the Seattle Mariners in 2008 out of University of Georgia played for in the MLB ?”, aligned with Table 1

SQL as an intermediate representation, we are able to frame the notion of “truth” as a function over facts found within a database. Additionally, by generating an intermediate representation of the FEVEROUS claim, we not only produce an interpretable decomposition of the implicit truth claims, but we also have a reusable blueprint for future fact verification even if the values in our underlying database are updated. This differs from the traditional end-to-end approach, where the full hybrid context must be passed each time a prediction is made.

BLENDSQL as Predicate Logic BLENDSQL transforms the propositional claims of FEVEROUS into predicate logic, providing a new language to evaluate the truth value of a statement given world knowledge in a relational database. For example, we can take the following (abbreviated) example from the FEVEROUS dev set.

Pesamino Victor (an association footballer) and his team lost in all their in-

ternational matches.

Given the constant p for Pesamino Victor and two-place predicates PLAYSON , PLAYEDBY , and WON , this becomes the following.

$$\begin{aligned}
 & \exists t (\text{ASSOCIATIONTEAM}'(t) \wedge \text{PLAYSON}'(p, t)) \\
 & \wedge \neg \exists m (\text{INTERNATIONALMATCH}'(m) \\
 & \quad \wedge \text{PLAYEDBY}'(t, m) \wedge \text{WON}'(t, m))
 \end{aligned}$$

Below, we display the BLENDSQL program corresponding to the same statement, given the underlying structured (the *Pesamino Victor* table) and unstructured (*documents*) context.

```

SELECT (
  {{
    LLMValidate(
      'Is Pesamino Victor an association footballer?',
      (
        SELECT * FROM documents
        WHERE title = 'pesamino victor'
      )
    )
  }}
) AND NOT EXISTS (
  SELECT * FROM "Pesamino Victor" AS w
  WHERE {{
    LLMMap(
      'Did they win this match?',
      'w::opposition (result)'
    )
  }} = TRUE
)

```

4 Experiments

4.1 Dataset Pre-processing

In all the datasets we evaluate, we place both structured and unstructured contexts into a SQLite database as shown in Table 1. We use the FTS5 extension⁸ to create a full-text index with BM25 ranking over the *title* and *content* columns in our *documents* table.

⁸<https://www.sqlite.org/fts5.html>

Additionally, we use the fuzzy string matching described in Lin et al. (2020) to align a given question to relevant values in the underlying database, and provide these “bridge” matches as hints in our prompt. We serialize our databases as code, as proven effective for text-to-SQL in the study by Nan et al. (2023). Specifically, we include the *CREATE* clause for each table in the database, along with n example rows for every table except *documents*. In many datasets, the unstructured text includes lengthy passages that are often irrelevant to the given question. By withholding the unstructured content from the parser, we are able to scale to massive datasets such as OTT-QA.

End-to-End Prompting on HybridQA On the HybridQA dataset, many data points contain context data that is too large to fit into a single GPT-4 prompt. To solve for this, we include all tabular data, and truncate the values of the unstructured content to a max of 400 characters.

4.2 Teaching BLENDSQL via In-Context Learning

We implement our few-shot prompting via the guidance toolkit⁹, which supports handlebars-style syntax to control the prompting workflow¹⁰. For each dataset, we randomly sample ~ 10 examples from the train set and annotate their BLENDSQL representation. For HybridQA and FEVEROUS, we use 12 examples; for OTT-QA, we use 9.

4.3 Metrics

In an attempt to measure the true performance of our approach without overfitting to spurious answer formats present in the free-text question-answering datasets, we employ the denotation accuracy metric used by Cheng et al. (2023), from the script here. This metrics measures semantic “denotation accuracy”, normalizing different output formats (such as “seven” and “7”). On the HybridQA dev set, for example, 101 gold answers contain natural language numbers (*five million, three campuses*), and 697 gold answers contain numeric representations of numbers (*4 million, 6*). The official HybridQA evaluator would judge these different answer formats as incorrect, and therefore, we opt for measuring semantic denotation instead.

⁹<https://github.com/guidance-ai/guidance/tree/0.0.64>

¹⁰For the full prompt, see Appendix B

5 Results

We direct the curious reader to Table 6 in the Appendix for example predictions and more analysis.

5.1 HybridQA

Unlike in intermediate representations based on natural language, a BLENDSQL script may result in an execution error and fail to produce a response. Additionally, the returned *LLMQA* ingredient may deem the question as unanswerable given the provided context and respond with something in the spirit of “This table does not provide the necessary info.” In these cases, we experiment with falling back to the end-to-end prompting style as a last resort, as denoted by “BlendSQL + End-to-End” in Table 2.

With 12 few shot exemplars, the parsed BLENDSQL query was unable to generate a prediction on 33% (1,173 examples out of 3,466) of the dev set. Even with this shortcoming, it outperformed the truncated context, end-to-end method by 3.76%. By using BLENDSQL first, and falling back to end-to-end prompting, we boost performance by 8.63% to **57.76%**, outperforming the oracle document retriever approach of Sui et al. (2023). Notably, on the subset of 2,293 questions that generated a valid BLENDSQL script, we reached an accuracy of 64.43%, suggesting that improved few-shot examples and perhaps more refined ingredients may have the potential to further boost performance.

5.2 OTT-QA

Depicted in Table 3, BLENDSQL allows for competitive performance among existing finetuned benchmarks, with only 9 few-shot examples. Even more promising is BLENDSQL’s use of a naive BM25 document retriever, and ability to encode the unified reasoning roadmap into a single, debuggable query. While we use the predictions from the table retriever described in Ma et al. (2022), it is up to our parser to write a satisfactory FTS5 query for retrieving documents from the BM25 index store. For these reasons (in-context learning, with a naive BM25 retriever), it is difficult to compare our results to existing literature directly. Future work involving a more robust retrieval algorithm (e.g., retrieval with vector embeddings) has the potential to further improve results.

^{10‡} We follow the previous work in Sui et al. (2023) and only evaluate on claims which include table-based evidence. However, we discard data points with the label “not enough

Method	Accuracy
<i>Oracle Document Retriever</i>	
End-to-End (Sui et al., 2023)	56.68
<i>Searching Entire Context</i>	
End-to-End	49.13
BLENDSQL	52.89
BLENDSQL + End-to-End	57.76

Table 2: HybridQA dev set results using GPT-4-0613. *End-to-End* methods are zero-shot, whereas BLENDSQL uses 12 few-shot exemplars to teach the model this new SQL dialect as an intermediate representation.

Method	Accuracy
End-to-End (Sui et al., 2023)	83.21
BLENDSQL (3 ROWS) [‡]	65.7
BLENDSQL (Entire table) [‡]	68.03

Table 4: FEVEROUS dev set results using GPT-4-0613.

5.3 FEVEROUS

As shown in Table 4, BLENDSQL does not outperform the end-to-end prompting style on the FEVEROUS dataset. As described in 3.3, this is likely due to the complexity and ambiguity of outlining predicate logic to evaluate the truth value of a given claim, compared to the other hybrid QA tasks. Not only have we prompted our parser to compose a script that evaluates to the correct judgment (“supports” or “refutes”), but we have implicitly asked it to identify the various atomic truth claims made within a claim, and set boundaries for each with respect to database context. Additionally, the table structures in FEVEROUS tend to deviate far from the traditional relational model, with many subtables and empty values.

6 Error Analysis

To better understand the advantages and limitations of BLENDSQL, we annotate randomly sampled 50 BLENDSQL question-answer pairs from HybridQA mistakes. Our chosen denotation accuracy metric judges each datapoint as either a 1 or 0. Out

enough”, making the results not directly comparable.

* To normalize different output formats (such as “seven” and “7”), we employ the denotation accuracy metric used by Pasupat and Liang (2015), from the script [here](#). It is unclear if the previous methods use the same evaluation script.

Method	Accuracy
<i>Transformer-based Doc Retriever</i>	
FR+CBR (Chen et al., 2021)	28.1
CARP (Zhong et al., 2022)	33.2
CORE (Ma et al., 2022)	49.0
<i>BM25 Doc Retriever</i>	
BLENDSQL*	34.15

Table 3: OTT-QA dev set results using GPT-4-0613. The previous systems are all finetuned, using dedicated transformer-based retriever components for both the 400k tables and 5M passages. BLENDSQL uses the top-4 table retriever predictions from Ma et al. (2022), but encodes both the BM25 passage retrieval + reader steps in a unified program.

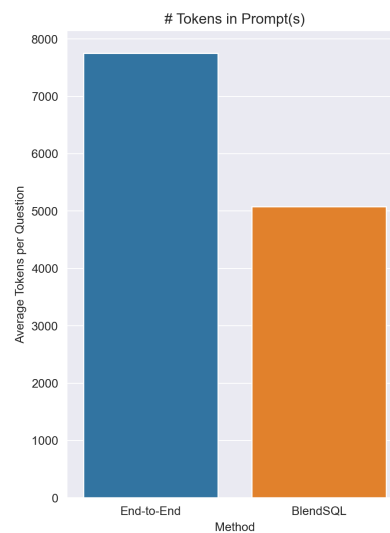


Figure 4: Average prompt tokens per question on the HybridQA dev set. BLENDSQL enables efficient filtering of large context databases to decrease data passed to the LLM by 35%.

of the 3,466 datapoints in the HybridQA dev set, we see 1,464 with an accuracy score of 0. Although the number 1,464 seems intimidating, it is important to remember that we achieved a literature-comparable score of 57.76 and to realize that there are many false negatives from those errors. To distinguish those false negative errors, we describe **Annotation Categories** as the parent categories containing false negative errors and the **BlendSQL Error Categories** as the child categories containing only true negative errors.

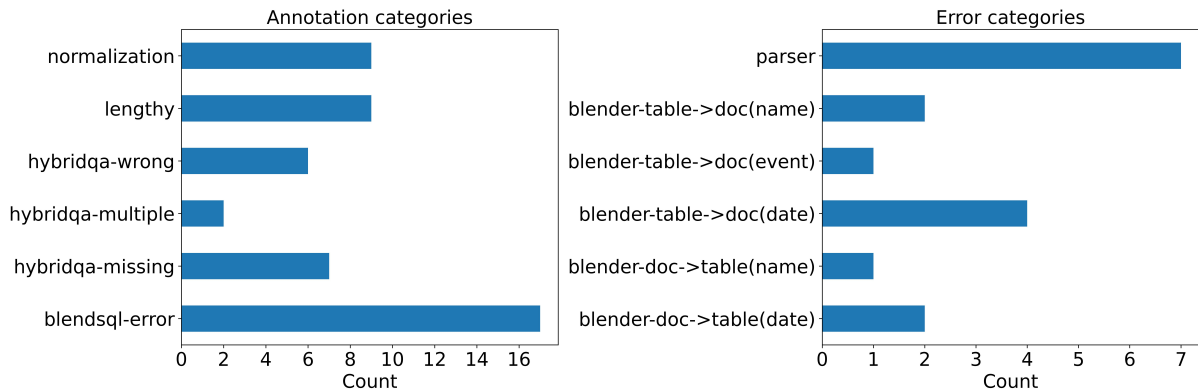


Figure 5: Error analysis on a random 50 samples of the HybridQA dev set. As described in Section 6.1, left shows 17 (34%) of the error are True Negative Errors for BlendSQL (*blendsql-error*). Right shows the causes of those True Negative Errors.

6.1 Annotation Categories

Figure 5 shows an overview of the annotation categories, among the 50 mistakes, there are considerable amount of mistakes are false negatives because those BLENDSQL answers are semantically equivalent to the HybridQA’s ground-truth answer. The annotation categories, along with their definitions, are described below:

Lengthy Our *LLMQA* ingredient tends to provide a more elaborate answer instead of naively an address, name, or date. These elaborate but semantically equivalent answers are annotated as *LENGTHY*.

Normalization Even with the normalization script of Cheng et al. (2023), some unwanted formatting variations remain. This second source of false negative is due to text normalization such as "\$20 millions" versus "20,000,000", "Belize City" versus "belize" and "METREX" versus "Metrex Network". We annotate this second source of false negative as *NORMALIZATION*.

HybridQA-Centric There are also some ground-truth answers from HybridQA that we believe are partially wrong, ambiguous with multiple answers, or require additional background information not present in the document or the table. We annotate those observations into *HYBRIDQA-WRONG*, *HYBRIDQA-MULTIPLE*, and *HYBRIDQA-MISSING*.

BlendSQL The mistake caused by BLENDSQL only occupies 17 out of the 50 sampled mistakes, amounting to 34% of the true negative rate.

6.2 BlendSQL Error Categories

Focusing on those 17 true negative BLENDSQL mistakes, we further break down the error into 6 major categories with Figure 5.

Parser The *PARSER* category indicates the incorrect answer is caused by a mistake in BLENDSQL syntax. For example, the “greater than” syntax (*WHERE "date" > {{LLMQA(...)}}*) is incorrectly generated as “equals” (*WHERE "date" = {{LLMQA(...)}}*).

Blender-Centric Besides the most common *PARSER* error, we see five additional categories indicating the direction of the multi-hop reasoning and the topic of the mistakes. Take the following question as an example.

What is the difference in time between Jose Reliegos of Spain and the person born 5 September 1892 who competed at the 1928 Olympics?

The answer requires a full-text search of the athlete name within the document via *documents MATCH 'born 5 september 1892'*, then a SQL query to calculate the time difference based upon the searched athlete name. In this given example, the error is annotated as *BLENDER-DOC->TABLE(DATE)*.

6.3 Error Causes

A majority of the BLENDSQL mistakes came from the parser generation. We find that sometimes the generated BLENDSQL syntax uses *LLMQA* to answer questions that are better suited for the *LLMJoin*

Model	Accuracy	F1	% Bad Syntax
DeepSeek-Coder-6.7b-Instruct	26.90	14.05	0.13
StarCoder2-15b	26.23	11.53	0.21
GPT-4-0613	52.89	45.02	0.03

Table 5: Comparing open code-finetuned LLMs against GPT-4-0613 on the HybridQA dev set. Despite its relatively small size, DeepSeek-Coder-6.7b-Instruct outperforms StarCoder2-15b. Both open models, however, perform far below GPT-4-0613.

operation. Those inappropriate parses hence incur incorrect answers related to date and numerical operation.

Among those blender mistakes, we find the multi-hop reasoning from DOC->TABLE is usually more challenging than TABLE->DOC. The reason is that searching for a potential filter over an unstructured document is more difficult and open-ended than a column filter over a structured table.

7 Open Model Evaluations

Closed models like GPT-4-0613, while powerful, lack a level of transparency and interpretability that is valuable to the research community. To this end, we also evaluate two open-source language models built for coding tasks: DeepSeek-Coder-6.7b-Instruct¹¹ (Guo et al., 2024) and StarCoder2-15b¹² (Lozhkov et al., 2024). Both models are autoregressive decoders with a context window of 16k tokens. We run experiments on the HybridQA dataset, and use each respective model as both the parser and blender to best reproduce the core experiments with GPT-4-0613.

7.1 Open Model Error Analysis

As shown in Table 5, the two open models fail to match the performance of GPT-4-0613. This can be partly attributed to the ability of the parser to generate valid BLENDSQL syntax: while GPT-4-0613 only generated bad syntax on 3% of the HybridQA dev set, DeepSeek-Coder and StarCoder2 saw rates of 13% and 21%, respectively. We explore specific execution errors yielded by different parser models in Figure 6. In this Figure, we consider everything to the right of “No Results” to be a syntax error. Referring back to section 6.2, these errors are subsets of the PARSER category.

¹¹<https://huggingface.co/deepseek-ai/deepseek-coder-6.7b-instruct>

¹²<https://huggingface.co/bigcode/starcoder2-15b>

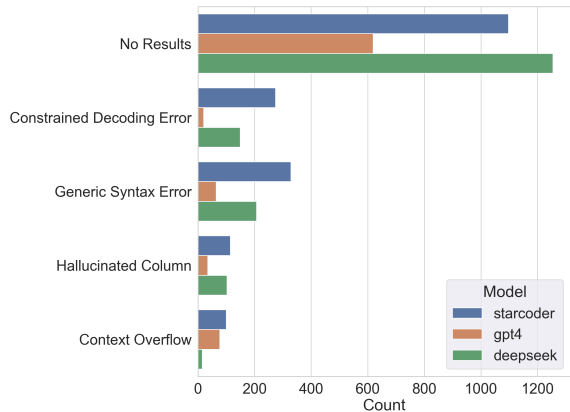


Figure 6: Detailing specific execution errors on the HybridQA dev set with various models as the parser. Most commonly, the *LLMQA* subprogram receives an empty table (e.g., due to an overly specific FTSS query) and fails to produce a response.

8 Future Work

We hope to study the ability of LLMs not only to use functions from a previously unseen SQL dialect but also to create functions that generalize certain logical patterns, as described in Cai et al. (2023). Additionally, while instruction fine-tuned LLMs show impressive abilities in executing BLENDSQL scripts, we hope to experiment with more specialized and inexpensive models for modular tasks such as fact verification and DPR retrieval.

Finally, as many errors came from BLENDSQL scripts that were unable to execute properly, we hope to explore hybrid QA tasks as a form of interactive semantic parsing (Elgohary et al., 2020; Glenn et al., 2023).

9 Conclusion

We introduce BLENDSQL, a scalable dialect for problem decomposition and hybrid question-answering. Results show competitive performance on popular benchmarks while using only ~ 10 few-shot examples. Additionally, on the HybridQA dataset, we improve the performance of a naive end-to-end system by 8.63%, while using 45% fewer prompt tokens. We open-source all code and present an installable Python package for future researchers to further explore BLENDSQL as an intermediate representation for hybrid question-answering.

10 Limitations

As our core experiments were conducted using GPT-4-0613, the API costs associated with reproducing the experiments may be a limiting factor. In an attempt to remedy this, we make our work open source and share all evaluation outputs.

Additionally, while we show that our approach can minimize prompt tokens and improve performance, we require an extra step of pre-processing all context into a SQLite database. We aim to streamline this pre-processing by adding support for more database management systems, and creating helper scripts for transforming a hybrid question-answering context into the appropriate database format.

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. [The fact extraction and VERification over unstructured and structured information \(FEVEROUS\) shared task](#). In *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pages 1–13, Dominican Republic. Association for Computational Linguistics.
- Seongsu Bae, Daeun Kyung, Jaehee Ryu, Eunbyeol Cho, Gyubok Lee, Sunjun Kweon, Jungwoo Oh, Lei Ji, Eric Chang, Tackeun Kim, et al. 2024. Ehrxqa: A multi-modal question answering dataset for electronic health records with chest x-ray images. *Advances in Neural Information Processing Systems*, 36.
- Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. [MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.
- Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2023. Large language models as tool makers. *arXiv preprint arXiv: 2305.17126*.
- Wenhu Chen, Ming wei Chang, Eva Schlinger, William Wang, and William Cohen. 2021. Open question answering over tables and text. *Proceedings of ICLR 2021*.
- Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020. [HybridQA: A dataset of multi-hop question answering over tabular and textual data](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036, Online. Association for Computational Linguistics.
- Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding language models in symbolic languages. *ICLR*.
- Edgar F Codd. 1970. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387.
- Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, Jinshu Lin, Dongfang Lou, et al. 2023. C3: Zero-shot text-to-sql with chatgpt. *arXiv preprint arXiv:2307.07306*.
- Ahmed Elgohary, Saghar Hosseini, and Ahmed Hassan Awadallah. 2020. Speak to your parser: Interactive text-to-sql with natural language feedback. In *Association for Computational Linguistics*.
- Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *CoRR*, abs/2308.15363.
- Parker Glenn, Parag Pravin Dakle, and Preethi Raghavan. 2023. [Correcting semantic parses with natural language through dynamic schema encoding](#). In *Proceedings of the 5th Workshop on NLP for Conversational AI (NLP4ConvAI 2023)*, pages 29–38, Toronto, Canada. Association for Computational Linguistics.
- Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y Wu, YK Li, et al. 2024. Deepseek-coder: When the large language model meets programming—the rise of code intelligence. *arXiv preprint arXiv:2401.14196*.
- Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A. Smith, and Mari Ostendorf. 2022. [In-context learning for few-shot dialogue state tracking](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2627–2643, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Alexander Hanbo Li, Patrick Ng, Peng Xu, Henghui Zhu, Zhiguo Wang, and Bing Xiang. 2021. [Dual reader-parser on hybrid textual and tabular evidence for open domain question answering](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4078–4088, Online. Association for Computational Linguistics.

- Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2022. Code as policies: Language model programs for embodied control. In *arXiv preprint arXiv:2209.07753*.
- Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, et al. 2023. Taskmatrix. ai: Completing tasks by connecting foundation models with millions of apis. *arXiv preprint arXiv:2303.16434*.
- Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, November 16-20, 2020*.
- Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S Yu. 2023. A comprehensive evaluation of chatgpt’s zero-shot text-to-sql capability. *arXiv preprint arXiv:2303.13547*.
- Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, et al. 2024. Starcoder 2 and the stack v2: The next generation. *arXiv preprint arXiv:2402.19173*.
- Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. 2021. [Inter-GPS: Interpretable geometry problem solving with formal language and symbolic reasoning](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6774–6786, Online. Association for Computational Linguistics.
- Yining Lu, Haoping Yu, and Daniel Khashabi. 2023. Gear: Augmenting language models with generalizable and efficient tool resolution. *arXiv preprint arXiv:2307.08775*.
- Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2022. [Open-domain question answering via chain of reasoning over heterogeneous knowledge](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5360–5374, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Paul McGuire. 2007. *Getting started with pyparsing*. "O’Reilly Media, Inc."
- Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. 2023. [Enhancing text-to-SQL capabilities of large language models: A study on prompt design strategies](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14935–14956, Singapore. Association for Computational Linguistics.
- Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. 2021. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*.
- Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2022. [UniK-QA: Unified representations of structured and unstructured knowledge for open-domain question answering](#). In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1535–1546, Seattle, United States. Association for Computational Linguistics.
- Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*.
- Panupong Pasupat and Percy Liang. 2015. [Compositional semantic parsing on semi-structured tables](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.
- Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.
- Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *arXiv preprint arXiv:2304.11015*.
- Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. 2023. Tool learning with foundation models. *arXiv preprint arXiv:2304.08354*.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.
- Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. [Hugging-GPT: Solving AI tasks with chatGPT and its friends in hugging face](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2023. [Gpt4table: Can large language models understand structured table data? a benchmark and empirical study](#).
- Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, and Le Sun. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv e-prints*, pages arXiv-2203.

Zekun Wang, Ge Zhang, Kexin Yang, Ning Shi, Wangchunshu Zhou, Shaochun Hao, Guangzheng Xiong, Yizhi Li, Mong Yuan Sim, Xiuying Chen, et al. 2023. Interactive natural language processing. *arXiv preprint arXiv:2305.13246*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Brandon T Willard and Rémi Louf. 2023. Efficient guided generation for llms. *arXiv preprint arXiv:2307.09702*.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. *Advances in Neural Information Processing Systems*, 35:32353–32368.

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 preprint arXiv:2305.10601*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. [Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Wanjun Zhong, Junjie Huang, Qian Liu, Ming Zhou, Jiahai Wang, Jian Yin, and Nan Duan. 2022. Reasoning over hybrid chain for table-and-text open domain qa. *arXiv preprint arXiv:2201.05880*.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. [TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3277–3287, Online. Association for Computational Linguistics.

A Appendix

A.1 Inferring Outputs from SQL Syntax

Ingredients in BLENDSQL are inherently stateless; they do not receive any external information other

than that which is explicitly passed in as arguments. This approach leads to a view of problem decomposition where each ingredient call is only responsible for the specialized task it’s been setup to solve for.

However, oftentimes, an ingredient’s placement within the larger SQL syntax can provide useful signals for the downstream LLMs. For example, consider the BLENDSQL query below.

```
SELECT * FROM w WHERE {{
  LLMMap(
    'What state is this city in?',
    'w::city'
  )}} = 'CA'
```

We have a single LLMMAP ingredient call, which will receive all distinct values from the `city` column. Examining the query, we also understand that we expect a returned string that looks like `'CA'`, as opposed to `'California'`, or any other equivalent denotation. By traversing the AST and extracting the arguments of predicates like `=`, `>`, and `<`, we are able to inject this prior knowledge into the LLMMAP call with a simple flag *“Here is an example output: CA”*. Additionally, by utilizing open models with publicly accessible logits, we are able to directly enforce string patterns at the decoding level (Willard and Louf, 2023). For example, take the below query.

```
SELECT * FROM w WHERE {{
  LLMMap(
    'Is this a team event?',
    'w::event'
  )}} = TRUE
```

By using the same AST traversal logic described above, we can infer that we expect the LLMMAP to return a boolean datatype. Using constrained generation, we are able to only produce a response from a language model that matches the regular expression `((true|false);)`¹³.

B Few-Shot Prompt

Below we show the few-shot prompt we used for our core experiments, along with an example database serialized in the “code” format.

```
{{#system~}}
Generate BlendsQL given the question to answer
the question correctly.
BlendsQL is a superset of SQLite, which adds
external function calls for information not
found within native SQLite.
These external functions should be wrapped in
double curly brackets.

If question-relevant column(s) contents are not
suitable for SQL comparisons or
calculations, map it to a new column using
the new function:
`LLMMAP('question', '{table}::{column}')
```

¹³We use a semicolon as a separator here


```

If mapping to a new column still cannot answer
the question with valid SQL, turn to an
end-to-end solution using the new function:
`LLMQA('{question}', ({blendsql}))`

If we need to do a `join` operation where there
is imperfect alignment between table values,
use the new function:
`LLMJoin(({blendsql}),
options='{table}::{column}')`

ONLY use these BlendSQL functions if necessary.
Answer parts of the question in vanilla SQL, if
possible.
{{~/system}}

{{#user~}}
{{few_shot_examples}}

{{serialized_db}}
Question: {{question}}
BlendSQL:
{{~/user}}

{{#assistant~}}
{{gen "result" temperature=0.0}}
{{~/assistant}}

```

```

CREATE TABLE "w" (
  "index" INTEGER,
  "no" INTEGER,
  "rider" TEXT,
  "team" TEXT,
  "motorcycle" TEXT
)
/*
3 example rows:
SELECT * FROM w LIMIT 3
  index  no      rider          team
  -----
0       1      carl fogarty    ducati performance
         ducati 996
1       4      akira yanagawa kawasaki racing team
         kawasaki zx-7rr
2       5      colin edwards   castrol honda
         honda rc45
*/

CREATE VIRTUAL TABLE "documents" USING
fts5(title, content, tokenize = 'trigram')

```

text can easily be extended for images for Visual Question Answering (VQA) task¹⁴, as shown in Bae et al. (2024). This section presents an example of how adding of a VQA Ingredient to BlendSQL allows us to do QA over a different unstructured media (images). In this example, a table and the corresponding image from a Wikipedia page¹⁵ are used for answering a question. Figure 7 shows the table from the Wikipedia page containing information about different fountains along with the corresponding image of the same. A tiny VQA model¹⁶ is used for answering the question once the correct image has been retrieved. The byte representation of the images is stored in the SQLite database and used for the question answering task.

Below, we show an example natural language to BLENDSQL pairing for this database context. The below query corresponds to the question “How many animals are in the fountain designed by Georgia Gerber?”. When executed, it invokes the VQA model on the retrieved database content to return “There are three animals in the fountain”.

```

SELECT
  {{
    VQA(
      'How many animals are in this fountain?',
      'w::Image'
    )
  }}
FROM w WHERE "Designer(s)" LIKE '%Georgia Gerber%'

```

C Hybrid QA over Images

The main paper discusses the Hybrid Question Answering task with text as the unstructured media in addition to tabular data. The methodology used for

¹⁴<https://github.com/parkervg/blendsql/blob/main/examples/vqa-ingredient.ipynb>
¹⁵https://en.wikipedia.org/wiki/Fountains_in_Portland,_Oregon
¹⁶<https://huggingface.co/bczhou/tiny-llava-v1-hf>




Title	Designer(s)	Year	Description	Image
<i>Animals in Pools</i>	Georgia Gerber	1986	Georgia Gerber's <i>Animals in Pools</i> includes ten trough-style fountains. The fountains contain 25 bronze sculptures of animals found in the Pacific Northwest. The pieces were installed in 1986 as part of the Local Improvement District affiliated with TriMet's MAX Light Rail.	
Bill Naito Legacy Fountain				
"The Car Wash" (officially <i>Untitled</i>)	Carter, Hull, Nishita, McCulley and Baxter	1977		

Figure 7: Sample of the hybrid table from the *Fountains in Portland, Oregon* Wikipedia page used for demonstrating Visual Question Answering using BlendSQL

Q & A

BlendSQL

Q: The 1995 Tooheys 1000 driver who was second-to-last in the Tooheys Top 10 was born where ?

A: Sydney

```

{{
  LLMQA(
    'Where was the driver born?',
    (
      WITH t AS (
        SELECT driver FROM "./1995 Tooheys 1000 (0)"
        ORDER BY tt10 DESC LIMIT 1 OFFSET 1
      ), d AS (
        SELECT * FROM documents
        JOIN t
        WHERE documents MATCH '' || t.driver || '' || ' OR
born'
        ORDER BY rank LIMIT 5
      ) SELECT d.content, t.driver AS 'Driver' FROM d JOIN t
    )
  )
}}

```

Output: Sydney ✓

Q: What is the title for the Taiwanese television series where Jin Chao-chun plays a Chinese politician who was born in the year 1090 ?

A: Eight Thousand Li of Cloud and Moon

```

SELECT "title"
FROM "./List of Taiwanese dramas from 2011 to present (3)" AS w
WHERE "cast" LIKE '%' || {{
  LLMQA(
    'Who is the Chinese politician born in 1090?',
    (
      SELECT * FROM documents
      WHERE documents MATCH 'chinese OR politician OR
1090'
      ORDER BY rank LIMIT 5
    ),
    options='w::cast'
  )
}} || '%'

```

While the logic for querying the table of unstructured content is correct, the model grounds the response to the wrong table. The 'role' column of ".Jin Chao-chun (0)" should have been used instead.

Output: My Hero , My Daddy ✗

Q: Abdul Hai Neamati was a member of a political party and was succeeded by Bashir Baghlani.

A: SUPPORTS

```

SELECT EXISTS(
  SELECT * FROM "Abdul Hai Neamati"
  WHERE "succeeded by" = 'bashir baghlani'
) AND (
  {{
    LLMValidate(
      'Was Abdul Hai Neamati a member of a political party?',
      (SELECT * FROM documents)
    )
  }}
)

```

Output: SUPPORTS ✓

Table 6: Example outputs from OTT-QA and FEVEROUS.