SUQL: Conversational Search over Structured and Unstructured Data with Large Language Models

Anonymous ACL submission

Abstract

While most conversational agents are grounded on either free-text or structured knowledge, many knowledge corpora consist of hybrid sources. This paper presents the first conversational agent that supports the full generality of hybrid data access for large knowledge corpora, through a language we developed called SUQL (Structured and Unstructured Query Language). Specifically, SUQL extends SQL with free-text primitives (summary and answer), so information retrieval can be composed with structured data accesses arbitrarily in a formal, succinct, precise, and interpretable notation. With SUQL, we propose the first semantic parser, an LLM with in-context learning, that can handle hybrid data sources.

Our in-context learning based approach when applied to the HybridQA dataset comes within 9.2% exact match and 6.8% F1 to the SOTA on the dev set trained on 62K data samples. More significantly, unlike previous approaches, our technique is applicable to large databases and free-text corpora.

We introduce a dataset consisting of crowdsourced questions and conversations on Yelp, a large, real restaurant knowledge base with structured and unstructured data. We show that our few-shot conversational agent based on SUQL finds an entity satisfying all user requirements 90.3% of the time, compared to 63.4% for a baseline based on linearization.[1]

1 Introduction

Large Language Models (LLMs) have shown exceptional performance on numerous downstream tasks. A range of recent works focus on improving their factuality by grounding responses in external resources including structured data (Hu et al., 2022; An et al., 2023; Nan et al., 2023; Poesia et al., 2022; Arora et al., 2023; Xu et al., 2023) and free text (Khattab et al., 2023; Jiang et al., 2023; Semnani et al., 2023; Gao et al., 2023).

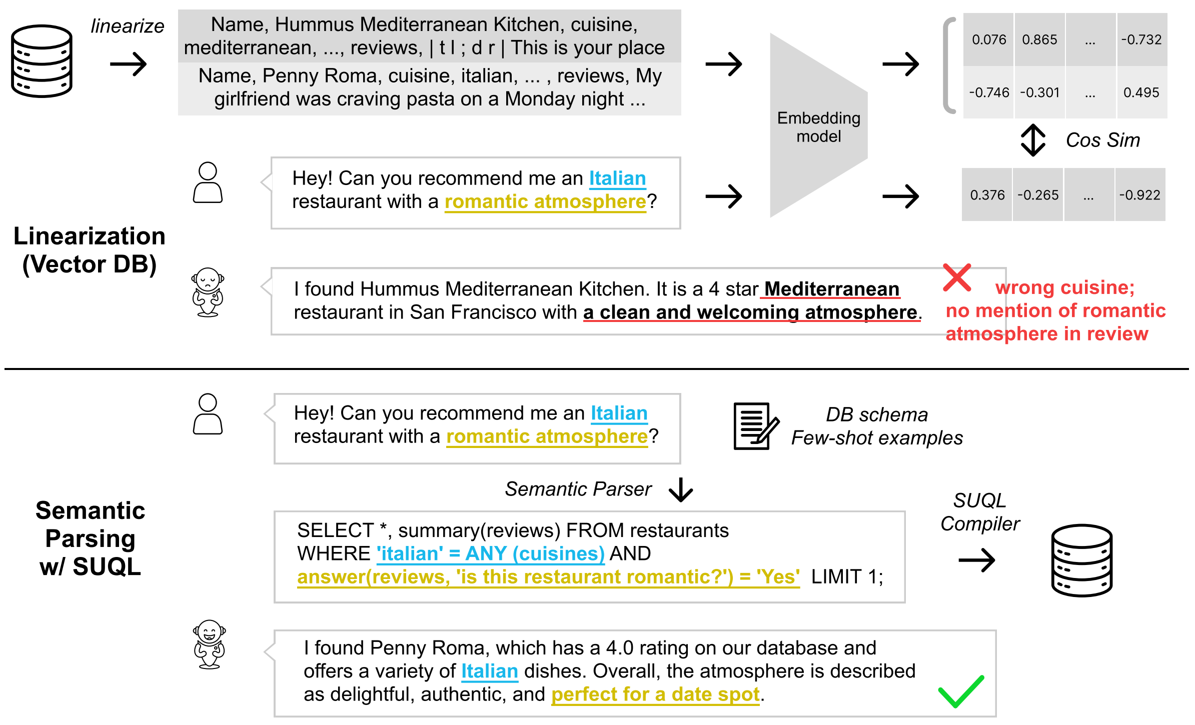
However, many data sources contain both structured data and free text: patient records, financial databases, and review websites, to name a few. Figure 2 in the appendix shows the running example of an application used in this paper. Each row in this table represents a unique restaurant, with information such as its name, type of cuisine, and rating as structured data. In addition, each row includes popular dishes and customer reviews in free text. To answer a question like "Can you find me an Italian restaurant with a romantic atmosphere?", an agent needs to combine the structured attribute cuisines and the free-text attribute reviews.

To handle the combination of structured and unstructured data, many previous chat systems use a *classifier* to assign queries to one of its specialized modules that is designed to handle structured data, unstructured data, or chitchat (Jin et al., 2021; Chi et al., 2022; Zhao et al., 2023). Unfortunately, this approach is inadequate for questions that need both free-text and structured data.

Another popular approach is to convert, or *linearize*, the structured data into free text (Oguz et al., 2022), as shown in Figure 1. With this approach, we can no longer wield the power of SQL to query the database, and free text retrievers are not good at handling complex questions.

The need of composing hybrid data source queries is highlighted by the HybridQA dataset, which collects many natural questions whose answers include information from both structured data and free text (Chen et al., 2020). Previous attempts trying to ground question-answering systems on hybrid data (Lei et al., 2023a; Wu et al., 2023; Kumar et al., 2023; Lee et al., 2023a) either work on only small data sets, or forego the expressiveness of structured data queries, or support limited compositions of structured and unstructured knowledge queries.

[1] Data and code will be released upon publication.



linearize
Name, Hummus Mediterranean Kitchen, cuisine, mediterranean, ..., reviews, | t l ; d r | This is your place
Name, Penny Roma, cuisine, italian, ... , reviews, My girlfriend was craving pasta on a Monday night ...
Embedding model
0.076 0.865 ... -0.732
-0.746 -0.301 ... 0.495
Cos Sim
0.376 -0.265 ... -0.922
Hey! Can you recommend me an Italian restaurant with a romantic atmosphere?
Linearization (Vector DB)
I found Hummus Mediterranean Kitchen. It is a 4 star Mediterranean restaurant in San Francisco with a clean and welcoming atmosphere.
wrong cuisine; no mention of romantic atmosphere in review
Hey! Can you recommend me an Italian restaurant with a romantic atmosphere?
DB schema
Few-shot examples
Semantic Parser
Semantic Parsing w/ SUQL
SELECT *, summary(reviews) FROM restaurants WHERE 'italian' = ANY (cuisines) AND answer(reviews, 'is this restaurant romantic?') = 'Yes' LIMIT 1;
SUQL Compiler
I found Penny Roma, which has a 4.0 rating on our database and offers a variety of Italian dishes. Overall, the atmosphere is described as delightful, authentic, and perfect for a date spot.

Figure 1: Comparison of traditional approach (lineralization) with our approach (semantic parsing with SUQL). Top: In the linearization approach, database entries are linearized and converted to embedding vectors. At run-time, a user request is converted to an embedding vector, which is used to find the closest embedding from the stored vectors. The results are then supplied to LLM for response generation.
Bottom: In our approach (semantic parsing with SUQL), a user utterance is parsed into formal SUQL by a few-shotted LLM, which is then executed by the SUQL compiler to fetch results from the database. The results are then supplied to LLM for response generation.

This paper proposes an approach to grounding conversational agents in hybrid data sources that take full advantage of both structured data queries and free-text retrieval techniques.

Our first contribution is to **demonstrate empirically that in real-life conversations, it is natural for users to ask questions that span both structured and unstructured data**. Through crowdsourcing, we obtain questions that users ask and conversations that they have with a restaurant chatbot. Results show that more than 49% of those questions require knowledge from both structured and unstructured knowledge.

To leverage the expressiveness and precision of formal query languages, **we propose SUQL, a precise, succinct, compositional, expressive, and executable formal language.** SUQL augments SQL with several primitives for processing free text. At a high level, SUQL combines an off-the-shelf retrieval model (for unstructured data) with the SQL semantics and operators (for structured data).

We validate our approach using the HybridQA data set. Experiments on HybridQA show that a few-shot, SUQL-based QA system comes within 9.2% exact match and 6.8% F1 to the SOTA model trained on over 60K data samples.

We have developed a fully operational conversational agent with a few-shot LLM-based semantic parser with SUQL, shown in the bottom part of Figure 1. We create a new single-turn user question data set and a conversational dataset on Yelp, a large, real knowledge corpus. Our chatbot using SUQL finds an entity satisfying all user requirements 90.3% of the time, compared to 63.4% for a baseline based on linearization.

2 Related Work

Text-to-SQL Semantic Parsing. Text-to-SQL systems have been built for single-turn question answering tasks (Guo and Gao, 2020; Wang et al., 2020a; Scholak et al., 2021; Zhong et al., 2017) as well as multi-turn, conversational tasks (Yu et al., 2019a,b; Wang et al., 2020b; Liu et al., 2022). Recently, LLMs have shown promising results on the text-to-SQL semantic parsing problem via in-context learning (Brown et al., 2020), with a range of work focusing on various prompting strategies (Hu et al., 2022; Poesia et al., 2022; An et al., 2023;

Nan et al., 2023; Arora et al., 2023; Guo et al., 2023; Sun et al., 2023; Zhang et al., 2023b).

This line of work is only applicable to structured data sources without any free text. When it comes to free text, SQL is limited to basic pattern-matching on strings, hindering the application of text-to-SQL where deeper support is needed.

Specialized Modules Using a Classifier. One approach to building a conversational interface to hybrid sources is to classify each question and assign it to one of the specialized modules. For instance, Chirpy (Chi et al., 2022) implements different modules to handle different kinds of questions. Jin et al. (2021) and Zhao et al. (2023) implement a "Turn Detection" module, to determine whether a user turn involves unstructured data access or should be handled by APIs/DBs. However, real user questions naturally span across both structured and free text columns, which cannot be answered by systems built with separate modules.

Linearize structured data. Another popular approach is to turn structured data into a linear form, which can then be directly used by a language model. A common approach is to linearize raw tables row-by-row and feed linearized content into a Tabular Language Model (TaLM) (Herzig et al., 2020; Yin et al., 2020; Eisenschlos et al., 2021; Deng et al., 2022; Iida et al., 2021; Sun et al., 2022).

In particular, Oguz et al. (2022) linearizes Wikidata and Wikipedia tables, combines them with Wikipedia text, and applies a retrieval model to open-domain question-answering. However, this approach is inherently limited. Many queries are challenging to answer through free text alone, such as "What are the number of deaths due to Covid in August and September 2020 in New York?". These types of inquiries can be easily addressed using structured data, which supports comparisons and calculations across a big database. Moreover, linearization complicates the unification of different parts of the database.

3 Design and Rationale of SUQL

We present the design and rationale of the SUQL language in this section. The design objectives of the representation are *expressiveness*, *accuracy*, and *efficiency*.

3.1 Design Rationale

Expressiveness. The design must be expressive, supporting the full generality of queries of hybrid knowledge corpus. It must handle arbitrary compositions of (1) relational operators in databases and (2) queries on free-text documents. Note that such a design automatically subsumes the *multi-hop retrieval* in NLP literature, where the results of a retrieved answer are used to retrieve another.

Formal languages, such as SQL, have been proven to be *complete* with respect to relational algebra (Codd, 1972). It can handle arbitrary compositions by virtue of its grammar rules, which for example, can be used to produce an unbounded number of nested subqueries.

Instead of linearization, which turns all structured data into text, we propose the opposite. SUQL is an extension of SQL with two NLP operators, SUMMARY and ANSWER: SUMMARY produces the summmary of a given text, and ANSWER returns the answer to a given text. These operators can be used anywhere with text values in the grammar, no different from numeric operators with numeric values. The advantage of this design is that SUQL is a succinct, formal representation that is complete with respect to relational algebra and NLP operations.

Accuracy of Translation from Natural Language. LLMs have been shown to be capable of translating complex text in one natural language to another. They can translate complex sentences into SQL queries for albeit small databases with compound operations, such as the use of group-by, ranking, and subqueries.

We posit that SUQL will give LLMs a succinct notation to express complex queries involving hybrid data sources. Leveraging LLMs familiarity with SQL, we hypothesize that we can create a semantic parser for translating user queries in a conversation into SUQL queries with a LLM via in-context learning.

Efficiency. SUQL queries can be executed by the SQL compiler requiring no modifications, as the SUMMARY and ANSWER primitives can be provided simply as user-defined functions. However, such an SQL compiler will perform very poorly. A naive implementation of these textual primitives would require retrieving and applying the NLP operation one record at a time, which is prohibitively expensive for large tables. Naive execution of the ANSWER function will not be effective.

Note that unlike previous methods such as retrieval-based semantic parsing where queries are constructed as results are retrieved (Cao et al., 2022; Gu and Su, 2022), SUQL expresses the query

Operator	Description	Example
ANSWER (t : `text` | `text` [], q : `text`) → `text`	return the answer to question q on value t	ANSWER (reviews, "is this restaurant family-friendly?")
SUMMARY (t : `text` | `text` []) → Text	return the summary of t	SUMMARY (reviews)

Table 1: Free Text primitives in SUQL

in its entirety. This makes it possible for us to develop an optimizing compiler, as described in Section 5.

3.2 Design of SUQL

We introduce two operations for text values in SQL. In this paper, we use `text` to represent any of the text types in SQL (CHAR, VARCHAR, TEXT, ...).

We define ANSWER (t,q) to return an answer to question q on text input t. For instance,

> ANSWER (reviews, 'is this restaurant family-friendly?')

will return yes if the reviews indicate that the restaurant is family-friendly, and no otherwise. The result is a text value that can be used anywhere it is allowed. For instance,

> ANSWER (reviews, 'is this restaurant family-friendly?') = 'Yes'

can be used as a filter to select family-friendly restaurants.

ANSWER is a universal function that can be used to derive any information from a text value by supplying the right question. However, for convenience, we introduce SUMMARY (t) as syntactic sugar for

> ANSWER (t, "what is the summary of this document").

We posit it that the semantic parser can easily learn to use SUMMARY. The formal definitions of ANSWER and SUMMARY are shown in Table 1.

The ANSWER and SUMMARY operations can be applied to any text arguments and their results can be used where a text value is expected, resulting in compositions of hybrid data accesses. Complex compositions of free text primitives and other SQL operators are highlighted by questions in the HybridQA dataset. In HybridQA, each cell in a column C is potentially linked to some passages, which we store in a separate column called C_Info. All questions from the dataset can be represented in SUQL. We show 6 representative examples of how each type of question can be represented in SUQL in Table 2.

4 Conversational Agent

Using SUQL as the formal representation, the architecture of a conversational agent with a hybrid knowledge corpus is relatively straightforward.

The Dialogue State Tracking problem (Cheng et al., 2020; Andreas et al., 2020; Campagna et al., 2022) for the SUQL-based conversational agent of a given schema S is defined as follows. We define the dialogue history to consist of a sequence of utterances between the user and the agent, $A_1, U_1, \cdots, A_n, U_n$, where A_i and U_i denotes an agent utterance and user input at turn i, respectively. Each $U_i = (t_i, q_i)$ where t_i is the natural text input, and q_i is a SUQL query for schema S if t_i carries a query. Given schema S, (A_i, U_i) for all previous turns $1 \leq i < n$ and the latest user utterance t_n, dialogue state tracking predicts q_n if t_n carries a query.

The semantic parser for the dialogue state tracking consists of two stages, both implemented with an LLM using in-context learning. The first classifies if the knowledge corpus needs to be consulted. For user utterances like greetings or general questions, it skips the knowledge corpus access. If consulting is needed, the second stage predicts q_n. The prompt includes the schema definition and few-shot examples demonstrating SUQL free-text primitives.

If the user utterance corresponds to a query, then the predicted SUQL query is executed. Because the semantic parser may have translated the user query incorrectly, the agent is instructed via a prompt to explicitly state to the user what it searched, based on the predicted SUQL at this turn (e.g., "I searched for Italian restaurants with a romantic atmosphere."). If the search returns a result, we ask the LLM to formulate the response based on the result; otherwise, we *explicitly* ask it to indicate that no results are found. The latter is important because LLMs tend to hallucinate whenever no answers to the user question are supplied.

Question Type	Exemplar Question	SUQL query
Type I	Where was the XXXl Olympic held?	`SELECT answer`(“Event year Info”, ‘where is this event held?’) `FROM` table `WHERE` “Name” `=` ‘XXXI’;
Type II	What was the name of the Olympic event held in Rio?	`SELECT` “Name” `FROM` table `WHERE answer`(“Event year Info”, ‘is this event held in Rio?’) `=` ‘Yes’;
Type III	When was the flag bearer of Rio Olympic born?	`SELECT answer`(“Flag Bearer Info”, ‘when is this person born?’) `FROM` table `WHERE answer`(“Event year Info”, ‘is this event held in Rio?’) `=` ‘Yes’;
Type IV	Which male bearer participated in Men’s 100kg event in the Olympic game?	`SELECT` “Flag Bearer” `FROM` table `WHERE` “Gender” `=` ‘Male’ `AND answer`(“Flag Bearer Info”, ‘did this person participate in Men’s 100kg event?’) `=` ‘Yes’;
Type V	For the 2012 and 2016 Olympic Event, when was the younger flag bearer born?	`SELECT MAX`(`answer`(“Flag Bearer Info”, ‘when is this person born?’)::date) `FROM` table `WHERE` “Event year” `IN` (‘2016’, ‘2012’);
Type VI	When did the youngest Burmese flag bearer participate in the Olympic opening ceremony?	`SELECT` “Event year” `FROM` table `ORDER BY answer`(“Flag Bearer Info”, ‘when is this person born?’)::date `DESC LIMIT` 1;

Table 2: The question types in HybridQA with exemplar questions (Figure 3 of Chen et al. (2020)) translated to the corresponding SUQL queries.

5 An Optimizing SUQL Compiler

Here, we describe the key optimizations we implemented in the SUQL compiler.

5.1 Search and Filter Optimization

When ANSWER is used as a filter in the query, the naive implementation would require a call to the LLM for every record in the database, which is infeasible. Just like how database indexing is used to optimize queries, we need to take advantage of dense retrieval models to quickly identify the relevant records, instead of operating on them one by one. In addition, if only a few results are needed, it is unnecessary to evaluate the filter on all the records.

First, our SUQL optimizing compiler identifies filters that use the ANSWER functions. It uses pre-computed embeddings from a dense retrieval model for similarity matching with the questions to identify top candidates. Note that the retrieved answers are relevant, but they may not satisfy the filtering constraints. We invoke the LLM on the entire clause to determine if the filter is successful. For example, suppose we are interested in a restaurant with parking, the dense retriever may return a review that says that it is hard to find parking. In this case, the filter ANSWER (reviews, "is parking easy") = "yes" is provided to the LLM to determine if the top candidates pass the filter. If multiple free text constraints are present, the SUQL compiler uses an aggregated similarity score based on each constraint to retrieve results that most likely satisfy all constraints.

5.2 Enumerated Types

Enumerated types (ENUM) are widely used in structured attributes to restrict the values of a text type to carry only one or more of a pre-defined set of permitted values. ENUM standardizes the values of the attributes so a filter based on the variable can be performed as a simple string match between the attribute values and permitted literals.

The challenge is how to ensure that the semantic parser will map ENUM attribute values to a permitted one. For all ENUM type declarations with no greater than $N = 10$ values, we include all the permitted values in the schema declarations supplied as a prompt to the LLM. The LLM is observed to be capable of automatically generating the ENUM values. For larger ENUM types, we do not include the permitted values, and the parser may generate an unexpected value. For example, the user utterance “Where can I find coffee” is likely to be translated to the filter clause ‘coffee’ `= ANY`(cuisines). However, the Yelp database only has ‘coffee & tea’ or ‘cafe’ cuisines, and not ‘coffee’.

Our solution is to redefine the semantics of the `=` operator for enumerated types. This is well known in the compiler literature as *overloading*. We first define the CLASSIFY function:

Definition 5.1.

$$\text{CLASSIFY}(t : \texttt{text}, S : v_1, \ldots, v_n) \\ = \{v_{i_1}, \ldots, v_{i_m}\}, \forall v_{i_k} \in S \text{ similar to } t.$$

Here, we say two strings are similar if they have similar meanings. It is possible the value of interest is not included in the set of permitted values. CLASSIFY returns $\emptyset$ if that is the case. We use a 0-shot LLM to implement CLASSIFY. It is also possible to use other methods such as cosine-similarity of the string embeddings.

Definition 5.2. The equal operator `=` is overloaded such that

$$t_1 = t_2 \text{ iff } t_2 \in \text{CLASSIFY}(t_1, E),$$
$$\text{where } t_1 : \texttt{text} \text{ and } t_2 : \text{ENUM}(E)$$

For instance, given a clause 'coffee' **= ANY**(cuisines), where CLASSIFY ('coffee', cuisines) = {'coffee & tea', 'cafe'}, then the clause will match any records whose cuisine attribute contains either 'coffee & tea' or 'cafe'.

5.3 Query Order Optimizations

Since ANSWER and SUMMARY involve LLM calls, it is important to minimize the execution of such functions.

Predicate Ordering. As discussed above, ANSWER functions in filters are expensive, compared to other predicates. Thus, whenever possible, the SUQL compiler would prioritize executing the other predicates so ANSWER is applied to fewer records.

Specifically, the SUQL compiler converts **SELECT** clauses with filter predicates into disjoint normal form (DNF), i.e., an OR of ANDs. For each AND clause, it prioritizes filters not using the ANSWER function so ANSWER calls are applied only to the filtered records.

Lazy Evaluation. Lazy evaluation, the concept of evaluating only when the result is needed, is a long-standing concept in programming languages (Hudak, 1989). The SUQL compiler adopts this concept to minimize execution cost. Specifically, when a **LIMIT** clause is present, it stops the evaluation once the required number of rows is filled.

6 Experiments

To evaluate SUQL, we perform two experiments. The first is on HybridQA, a popular academic question answering dataset as discussed above. Tables in HybridQA are small enough to be provided as input to a neural model. To perform a more comprehensive experiment on *conversations* with *large*, *real* data bases, we introduce a new benchmark based on the real restaurant data corpus from Yelp.com.

6.1 HybridQA Experiment

The HybridQA dataset consists of roughly 70K question-answering pairs aligned with 13,000 Wikipedia tables, whose entities are linked to multiple free-form corpora. Every question can be answered correctly only by referring to both the structured and unstructured data. To test out SUQL, we create the following system:

1. Use LLM with in-context learning (with less than 10 examples) to parse natural language and a given database schema into a SUQL query (Prompt 9).
2. Execute the generated SUQL to retrieve results from the database. If no results are returned, repeat this process by generating a different SUQL query, with up to 2 tries (Prompt 10).
3. Use LLM to convert the retrieved database result to a succinct answer (Prompt 11) since the gold labels in HybridQA are short. Because the gold labels have only one entity, even though the full answer may include multiple entities, we just pick one out of the possibly many results returned by SUQL.

`GPT-4-1106-preview` is used in all steps, except that `GPT-3.5-turbo-0613` is used in Step 3.

Our in-context learning-based QA system achieves 59.2 Exact Match and 68.5 F1 on the dev set of HybridQA and 57.7 EM and 67.1 F1 on the held-out test set, as shown in Table 3. Our method uses only 3 simple prompts, achieving within 9.2 EM and 6.8 F1 to the SOTA on the dev set, which has been trained on the HybridQA training set with over 62K examples.

Most significantly, unlike our approach, these models do not generalize beyond small tables. Techniques based on feeding the entire table into a Transformer (DocHopper, Mate, MITQA, DEHG, and MAFID) cannot be applied to large data corpora that exceed their input token limit. Neither can techniques based on retrieving entire columns (MuGER^2) and feeding into a reader model. The SOTA model S^{3}HQA separately retrieves rows in the table and passages. It then feeds the top results of each to the final reader. It needs to feed the whole column to the reader if the query involves sorting. In contrast, our approach has full compositional generality and can handle arbitrarily large datasets. We are the first to apply semantic parsing techniques to HybridQA since no prior formal representations could accurately capture hybrid queries.

Recently, Zhang et al. (2023a) applied LLaMA-based techniques to HybridQA. They fine-tuned LLaMA2 (7B) on their TableInstruct dataset with

Trained on (Size)	Method	Dev		Test	
		EM	F1	EM	F1
HybridQA (62k)	HYBRIDER (Chen et al., 2020)	44.0	50.7	43.8	50.6
	DocHopper (Sun et al., 2022)	47.7	55.0	46.3	53.3
	MuGER^2 (Wang et al., 2022)	57.1	67.3	56.3	66.2
	Mate (Eisenschlos et al., 2021)	63.4	71.0	62.8	70.2
	DEHG (Feng et al., 2022)	65.2	**76.3**	63.9	75.5
	MITQA (Kumar et al., 2023)	65.5	72.7	64.3	71.9
	MAFiD (Lee et al., 2023b)	66.2	74.1	65.4	73.6
	S^{3}HQA (Lei et al., 2023b)	**68.4**	75.3	**67.9**	**75.5**
TableInstruct (2.6M)	TableLlama (Zhang et al., 2023a)	27.61	-	-	-
Zero-shot	LLaMa2 (7B) (Zhang et al., 2023a)	20.72	-	-	-
Few-shot ($\leq$ 10 example)	SUQL	59.2	68.5	57.7	67.1

Table 3: Performance of few-shot-based SUQL and related work on the HybridQA dataset.

more than 2.6M samples and achieved only 27.61 EM on Hybrid QA. They also reported a baseline of LLaMa2 (7B) on HybridQA directly, which resulted in just 20.72 exact match.

Sui et al. (2023) also experimented using in-context learning with GPT-4 on HybridQA. However, they only reported the result of 1,000 randomly sampled questions from the dev set. For each question, they experiment with different formats (JSON, HTML, Markdown, etc.) of feeding the entire table and question to GPT-4. Then, according to one of the authors, they use regular expressions to match a GPT-4 prediction into one of the 1,000 gold labels to circumvent format-related issues. Their best-reported result is GPT-4 with HTML format at 56.68% with this metric. As a simplified approximation of their procedure, we experiment with checking if the gold label to a particular question is contained as a sub-string of our prediction, or vice versa. With this format matching heuristic, our SUQL-based system achieves an EM of 72.8% on questions in the entire dev set. This shows the effectiveness of SUQL on hybrid question-answering tasks.

Error Analysis. From analyzing 72 randomly sampled error cases, we found:

- 37.5% are due to format mismatches, e.g. “Johnson City, Tenessee” versus “Johnson City”. Similar issues related to evaluating LLM-generated responses have been noted by Kamalloo et al. (2023).
- 23.6% are due to the gold label being either wrong or incomplete. Incomplete cases exist because only one gold answer is permitted in HybridQA, while in fact for some cases, multiple possible correct answers could be found.
- 22.2% are due to semantic parsing errors.
- 11.1% are due to errors from the SUQL execution involving the LLM-based ANSWER function and ENUM classifier.
- the remaining 5.6% are due to type-related conversion errors, since HybridQA tables do not have annotated types while SUQL expects a typed schema.

6.2 Conversational Agent on Restaurants

To experiment with real-life datasets, we collect a total of 1828 restaurants from Yelp.com across 4 cities, alongside the top 20 reviews and top 20 popular dishes for each restaurant. The columns of our database are name, cuisines, price, rating, num_reviews, address, phone_number, opening_hours, location, reviews, and popular_dishes.

We use an off-the-shelf dense retriever model (Yu et al., 2022) as the retriever in SUQL. We use `gpt-3.5-turbo-0613` as the LLM for all systems in this section.

6.2.1 Collecting User Queries

We solicit user queries via crowdsourcing on Prolific (Prolific, 2023). We do not disclose to the workers what fields are available in the database so as to not bias their queries. We ask them to come up with 100 questions about restaurants. Separately, we also ask crowd workers to interact with our conversational agent (described in Section 4) and collect 96 turns across 20 conversations.

The setting of restaurants in real-life use cases requires a user to first specify a location, a structured column in the database. We annotate whether a user question only involves structured information or a combination with free text in Table 4. In single turns, all collected user queries involve searching for a restaurant. Out of the 96 dialogue turns, 62

	Single-turn	Conversation
Structured-only	45	37
Combination	55	25
Total	100	62

Table 4: Statistics on whether a search question requires only structured data or a combination.

	Single-turn	Conversational
Linearization @ 1	57.0 %	63.4 %
Linearization @ 3	49.7 %	61.9 %
SUQL	**93.8 %**	**90.3 %**

Table 5: Turn accuracy measurement on linearized system versus SUQL system.

involve searching for restaurants. In total, over 49% of user queries require knowledge from both structured and unstructured columns.

6.2.2 Turn Accuracy

We experiment with the linearization technique proposed by Oguz et al. (2022) for relational tables, using again the same dense retriever model (Yu et al., 2022). Specifically, we concatenate cell values on the same row and separate them by commas. Based on the conversation history, these systems use a few-shot LLM to extract a succinct search query for the retrievers.

For each user input, we manually inspect whether the restaurants retrieved by a system satisfy all criteria specified by the user and respond with correct and relevant information. Concretely, given a user utterance u and a list of returned restaurants $\mathcal{R} = \{r_1, r_2, \cdots, r_m\}$, we evaluate whether each r_i is a true positive or false positive. We calculate the turn accuracy as the number of true positives divided by the number of true and false positives for all the queries in the dataset.

For the SUQL system, the queries are limited to return at most 3 results. The accuracy is 93.8% for single-turn questions and 90.3% for conversational queries, as shown in Table 5.

We compare our results with two linearization-based systems, where $m = 1$ ("Linearization 1") and $m = 3$ ("Linearization 3"). SUQL improves the answer accuracy, by up to 36.8% in single-turn settings and up to 26.9% in conversations. This shows that the conversational agent with SUQL can provide much more accurate results.

Our system returns no answers to 21 of the 100 user questions and 8 of the 62 queries in the conversations. Manual inspection reveals that 7 out of the 21 and 2 out of the 8 truly have no answers. Thus, our system has a false negative rate of 14% and 9% for user questions and conversational turns, respectively.

6.2.3 User Feedback

We solicit feedback from our crowdsource users after they talk to our restaurant chat-bot with three free-form questions shown in Figure 5. Overall, the feedback was positive: "There's actually nothing I didn't like about this chatbot. I would honestly use this chatbot on a regular basis if it were available to the public", "I liked that the chatbot was fast in responses and it gave very detailed responses and I hardly had any questions about a restaurant after the option was given", and "Shocked at how good the restaurant suggestions were. I even asked for something with better prices and got that too. Now I'm hungry. I asked to define a cuisine style and it was able to do that".

Negative comments include: occasional slowness of the chatbot; "it didn't provide any links or pictures"; "It did not sound friendly and sometimes the responses were too long. Bullet point outputs would be much more helpful."

7 Conclusion

We introduce SUQL, the first formal query language for hybrid knowledge corpora, consisting of structured and unstructured data. The key novelty of SUQL is the incorporation of free-text primitives into a precise, succinct, expressive, and interpretable query language.

Our in-context learning based approach when applied to the HybridQA dataset comes within 9.2% exact match and 6.8% F1 to the SOTA on the dev set trained on 62K data samples. More significantly, unlike previous approaches, our technique is applicable to large databases and free-text corpora.

Our experiment on the real Yelp knowledge base with crowdsourced questions and conversations show that our in-context learning conversational agent based on SUQL finds an entity satisfying all user requirements 90.3% of the time, compared to 63.4% for a baseline based on linearization. The empirical findings underscore SUQL's applicability and its potential for future research directions such as domain-specific applications in biomedical, legal, and financial spheres.

Ethical Considerations

LLMs and formal languages such as SQL have been used by an increasingly large population of technical developers as well as everyday users. We propose to combine them in the hope of bringing the best of both sides to create a expressive, accurate, and efficient language that facilitates conversational search over structured and unstructured data. We do not foresee this work to result in any form of harm or malicious misuse.

Data. The data used in this work is an open-sourced research dataset (HybridQA) and a Yelp-based restaurant conversation dataset (Restaurant). During the curation process of the Restaurant dataset, we used a certified online research crowdsourcing platform Prolific to make sure that we respected worker's privacy and paid them at fair rates. Our procedure has been approved by an IRB from our institution.

Compute. The models used herein are existing pretrained retriever models and LLM API services provided by OpenAI. We did not additionally pretrain or finetune any compute-intensive models, therefore avoiding a significant carbon footprint in the experiments herein.

License. Our code will be released publicly and licensed under Apache License, Version 2.0. Our data will be made available to the community.

Limitations

Being LLM-based, SUQL can be subject to vulnerabilities that are intrinsic to LLMs. These intrinsic weaknesses can negatively affect SUQL's effectiveness, posing limitations on the overall pipeline performance. We highlight two aspects of limitation in the current version of SUQL methodology.

Performance Limitation. In this work, the LLM's semantic understanding capability upperbounds the semantic and syntactic correctness of parsed SUQL queries. The ANSWER and SUMMARY functionalities in SUQL can also be affected by the underlying LLM, resulting in potentially erroneous filtering evaluation during the execution of the SUQL queries.

Reliability Limitation. The applicability of the method can also be affected by the reliability of the underlying LLM. In our pipeline, the semantic parser may hallucinate database contents in a non-interpretable manner, even when explicitly instructed not to. Other caveats include non-deterministic behavior between LLM API calls and potential vulnerabilities against LLM-oriented adversarial attacks.

References

Shengnan An, Bo Zhou, Zeqi Lin, Qiang Fu, Bei Chen, Nanning Zheng, Weizhu Chen, and Jian-Guang Lou. 2023. Skill-based few-shot selection for in-context learning.

Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

Aseem Arora, Shabbirhussain Bhaisaheb, Harshit Nigam, Manasi Patwardhan, Lovekesh Vig, and Gautam Shroff. 2023. Adapt and decompose: Efficient generalization of text-to-sql via domain adapted least-to-most prompting.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Giovanni Campagna, Sina Semnani, Ryan Kearns, Lucas Jun Koba Sato, Silei Xu, and Monica Lam. 2022. A few-shot semantic parser for Wizard-of-Oz dialogues with the precise ThingTalk representation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 4021–4034, Dublin, Ireland. Association for Computational Linguistics.

Shulin Cao, Jiaxin Shi, Zijun Yao, Xin Lv, Jifan Yu, Lei Hou, Juanzi Li, Zhiyuan Liu, and Jinghui Xiao. 2022. Program transfer for answering complex questions over knowledge bases. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8128–8140, Dublin, Ireland. Association for Computational Linguistics.

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.

Jianpeng Cheng, Devang Agrawal, Héctor Martínez Alonso, Shruti Bhargava, Joris Driesen, Federico Flego, Dain Kaplan, Dimitri Kartsaklis, Lin Li, Dhivya Piraviperumal, Jason D. Williams, Hong Yu, Diarmuid Ó Séaghdha, and Anders Johannsen. 2020. Conversational semantic parsing for dialog state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8107–8117, Online. Association for Computational Linguistics.

Ethan A. Chi, Ashwin Paranjape, Abigail See, Caleb Chiam, Trenton Chang, Kathleen Kenealy, Swee Kiat Lim, Amelia Hardy, Chetanya Rastogi, Haojun Li, Alexander Iyabor, Yutong He, Hari Sowrirajan, Peng Qi, Kaushik Ram Sadagopan, Nguyet Minh Phu, Dilara Soylu, Jillian Tang, Avanika Narayan, Giovanni Campagna, and Christopher Manning. 2022. Neural generation meets real people: Building a social, informative open-domain dialogue agent. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 376–395, Edinburgh, UK. Association for Computational Linguistics.

E. F. Codd. 1972. Relational completeness of data base sublanguages. *Research Report / RJ / IBM / San Jose, California*, RJ987.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2022. Turl: Table understanding through representation learning. *ACM SIGMOD Record*, 51(1):33–40.

Julian Eisenschlos, Maharshi Gor, Thomas Müller, and William Cohen. 2021. MATE: Multi-view attention for table transformer efficiency. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7606–7619, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yue Feng, Zhen Han, Mingming Sun, and Ping Li. 2022. Multi-hop open-domain question answering over structured and unstructured knowledge. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 151–156, Seattle, United States. Association for Computational Linguistics.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling large language models to generate text with citations.

Yu Gu and Yu Su. 2022. ArcaneQA: Dynamic program induction and contextualized encoding for knowledge base question answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1718–1731, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Chunxi Guo, Zhiliang Tian, Jintao Tang, Pancheng Wang, Zhihua Wen, Kang Yang, and Ting Wang. 2023. Prompting gpt-3.5 for text-to-sql with desemanticization and skeleton retrieval. In *Pacific Rim International Conference on Artificial Intelligence*, pages 262–274. Springer.

Tong Guo and Huilin Gao. 2020. Content enhanced bert-based text-to-sql generation.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

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.

Paul Hudak. 1989. Conception, evolution, and application of functional programming languages. *ACM Comput. Surv.*, 21(3):359–411.

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. TABBIE: Pretrained representations of tabular data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3446–3456, Online. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation.

Di Jin, Seokhwan Kim, and Dilek Hakkani-Tur. 2021. Can I be of further assistance? using unstructured knowledge access to improve task-oriented conversational modeling. In *Proceedings of the 1st Workshop on Document-grounded Dialogue and Conversational Question Answering (DialDoc 2021)*, pages 119–127, Online. Association for Computational Linguistics.

Ehsan Kamalloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. 2023. Evaluating open-domain question answering in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5591–5606, Toronto, Canada. Association for Computational Linguistics.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2023. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp.

Vishwajeet Kumar, Yash Gupta, Saneem Chemmengath, Jaydeep Sen, Soumen Chakrabarti, Samarth Bharadwaj, and Feifei Pan. 2023. Multi-row, multi-span distant supervision for Table+Text question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8080–8094, Toronto, Canada. Association for Computational Linguistics.

Sung-Min Lee, Eunhwan Park, Daeryong Seo, Donghyeon Jeon, Inho Kang, and Seung-Hoon Na. 2023a. MAFiD: Moving average equipped fusion-in-decoder for question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2337–2344, Dubrovnik, Croatia. Association for Computational Linguistics.

Sung-Min Lee, Eunhwan Park, Daeryong Seo, Donghyeon Jeon, Inho Kang, and Seung-Hoon Na. 2023b. MAFiD: Moving average equipped fusion-in-decoder for question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2337–2344, Dubrovnik, Croatia. Association for Computational Linguistics.

Fangyu Lei, Xiang Li, Yifan Wei, Shizhu He, Yiming Huang, Jun Zhao, and Kang Liu. 2023a. s^3 hqa: A three-stage approach for multi-hop text-table hybrid question answering. *arXiv preprint arXiv:2305.11725*.

Fangyu Lei, Xiang Li, Yifan Wei, Shizhu He, Yiming Huang, Jun Zhao, and Kang Liu. 2023b. S3HQA: A three-stage approach for multi-hop text-table hybrid question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1731–1740, Toronto, Canada. Association for Computational Linguistics.

Qi Liu, Zihuiwen Ye, Tao Yu, Linfeng Song, and Phil Blunsom. 2022. Augmenting multi-turn text-to-SQL datasets with self-play. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5608–5620, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. 2023. Enhancing few-shot text-to-sql capabilities of large language models: A study on prompt design strategies.

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.

Gabriel Poesia, Oleksandr Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. 2022. Synchromesh: Reliable code generation from pre-trained language models.

Prolific. 2023. `https://www.prolific.com`. Acessed: June 2023.

Torsten Scholak, Raymond Li, Dzmitry Bahdanau, Harm de Vries, and Chris Pal. 2021. DuoRAT: Towards simpler text-to-SQL models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1313–1321, Online. Association for Computational Linguistics.

Sina Semnani, Violet Yao, Heidi Zhang, and Monica Lam. 2023. WikiChat: Stopping the hallucination of large language model chatbots by few-shot grounding on Wikipedia. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2387–2413, Singapore. Association for Computational Linguistics.

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. *Proceedings of WSDM 2024*.

Haitian Sun, William W. Cohen, and Ruslan Salakhutdinov. 2022. Iterative hierarchical attention for answering complex questions over long documents.

Ruoxi Sun, Sercan Ö. Arik, Rajarishi Sinha, Hootan Nakhost, Hanjun Dai, Pengcheng Yin, and Tomas Pfister. 2023. Sqlprompt: In-context text-to-sql with minimal labeled data.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020a. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Runze Wang, Zhenhua Ling, Jing-Bo Zhou, and Yu Hu. 2020b. Tracking interaction states for multi-turn text-to-sql semantic parsing. *ArXiv*, abs/2012.04995.

Yingyao Wang, Junwei Bao, Chaoqun Duan, Youzheng Wu, Xiaodong He, and Tiejun Zhao. 2022. MuGER2: Multi-granularity evidence retrieval and reasoning for hybrid question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6687–6697, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jian Wu, Yicheng Xu, Yan Gao, Jian-Guang Lou, Börje F Karlsson, and Manabu Okumura. 2023. Tacr: A table-alignment-based cell-selection and reasoning model for hybrid question-answering. *arXiv preprint arXiv:2305.14682*.

Silei Xu, Shicheng Liu, Theo Culhane, Elizaveta Pertseva, Meng-Hsi Wu, Sina Semnani, and Monica Lam. 2023. Fine-tuned LLMs know more, hallucinate less with few-shot sequence-to-sequence semantic parsing over Wikidata. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5778–5791, Singapore. Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019a. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979, Hong Kong, China. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019b. SParC: Cross-domain semantic parsing in context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4511–4523, Florence, Italy. Association for Computational Linguistics.

Yue Yu, Chenyan Xiong, Si Sun, Chao Zhang, and Arnold Overwijk. 2022. COCO-DR: Combating distribution shift in zero-shot dense retrieval with contrastive and distributionally robust learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1462–1479, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2023a. Tablellama: Towards open large generalist models for tables.

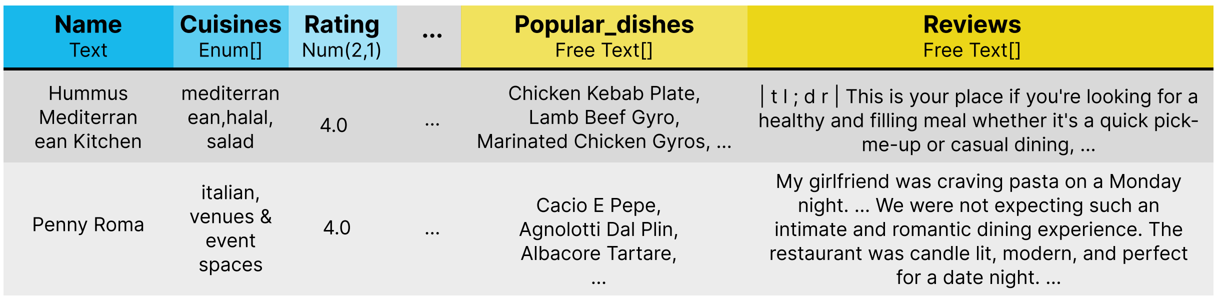
Yunjia Zhang, Jordan Henkel, Avrilia Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M. Patel. 2023b. Reactable: Enhancing react for table question answering.

Chao Zhao, Spandana Gella, Seokhwan Kim, Di Jin, Devamanyu Hazarika, Alexandros Papangelis, Behnam Hedayatnia, Mahdi Namazifar, Yang Liu, and Dilek Hakkani-Tur. 2023. "what do others think?": Task-oriented conversational modeling with subjective knowledge.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning.

A Appendix

A.1 Examples of the restaurant dataset



Name Text	**Cuisines** Enum[]	**Rating** Num(2,1)	**...**	**Popular_dishes** Free Text[]	**Reviews** Free Text[]
Hummus Mediterranean Kitchen	mediterranean,halal, salad	4.0	...	Chicken Kebab Plate, Lamb Beef Gyro, Marinated Chicken Gyros, ...	| t l ; d r | This is your place if you're looking for a healthy and filling meal whether it's a quick pick-me-up or casual dining, ...
Penny Roma	italian, venues & event spaces	4.0	...	Cacio E Pepe, Agnolotti Dal Plin, Albacore Tartare, ...	My girlfriend was craving pasta on a Monday night. ... We were not expecting such an intimate and romantic dining experience. The restaurant was candle lit, modern, and perfect for a date night. ...

Figure 2: `restaurants` table with both structured and unstructured data.

A.2 Hyperparameters

For all our experiments, we set a temperature of 0 in calls to OpenAI's LLMs, and we directly use the retriever provided by Yu et al. (2022), with the default parameters.

A.3 Prompts in our experiments

We provide the prompts mentioned in this paper. The syntax used is the Jinja2 template language, which supports Python-like loops (`{% for %}{% endfor %}`), conditions (`{% if %}{% endif %}`), variables (`{{ var }}`) and comments (`{# #}`).

A.4 Our crowdsourcing process on Prolific

We utilize Prolific (Prolific, 2023) to curate our Restaurant dataset. The crowdsourcing interface is presented in Figure 3, after starting the crowdsourcing task, the crowdsourcing workers will be prompted with questions shown in Figure 4. After they finish conversing with the chatbot, they will be shown three questions shown in Figure 5.

Among the 50 crowdsourcing workers who consented to reveal their demographic information, 33 are female and 17 are male. All 50 crowdsourcing workers reside in the United States. We paid the crowdsourcing workers 12.30 USD per hour. The average expected duration is 8 minutes. The pay rate is higher than the federal minimum wage in the United States, which is 7.25 USD per hour. Our crowdsourcing process asked for user consent in using their conversation with the chatbot for research purposes. No personal identifiable information was collected.

```
In a database, the {{ field_name }} field has the following set of options, separated by new lines. "{{ predicted_field_value
      }}" is not one of the possible choices. You need to classify "{{ predicted_field_value }}" into one or more of the
      values below:

{% for choice in field_value_choices %}
{{ choice }}
{% endfor %}

You can only select from the above choices. Your response should be a list of comma separated index numbers.
Your answer:
```

Table 6: ENUM classifier prompt used in SUQL compiler. This is a zero-shot prompt.

```
Answer a question based on the following text.{{ type_prompt }}

Question: {{ question }}. If there is no information, say "no info".

Documents:
{% for review in reviews %}
{{ review }}
{% endfor %}

Provide a concise answer in a few words:
```

Table 7: The ANSWER function prompt used in SUQL compiler. This is a zero-shot prompt.

```
`answer(document, query)` takes in a document and a query. It asks `query` on `document` and outputs the answer.

Now, let's look at this use case. Your task is to determine whether the output is correct.

answer({{ field }}, "{{ query }}") {{ operator }} {{ value }}

{{ field }} = ["{{ document }}"]

Choose from one of the following choices:
- the output is correct.
- the output is incorrect.
```

Table 8: The ANSWER function prompt as filter used in SUQL compiler. This is a zero-shot prompt.

```
You are a semantic parser. Generate a query for a database with given signature. Do not generate fields beyond the given
      fields.

1929_International_Cross_Country_Championships_0
CREATE TABLE validation_table_7 ("Rank" INT, "Athlete" TEXT, "Athlete_Info" TEXT[], "Nationality" TEXT, "Nationality_Info"
      TEXT[], "Time" TEXT);
User: What is the difference in time between Jos\'e Reliegos of Spain and the person born 5 September 1892 who competed at
      the 1928 Olympics ?
Target: SELECT a."Time"::INTERVAL - b."Time"::INTERVAL FROM "validation_table_7" a, "validation_table_7" b WHERE a."Athlete"
      = 'Jos\'e Reliegos' AND a."Nationality" = 'Spain' AND answer(b."Athlete_Info", 'is this athlete born 5 September
      1892?') = 'Yes';
--
List_of_cities_in_Somalia_by_population_0
CREATE TABLE validation_table_8 ("Rank" INT, "City" TEXT, "City_Info" TEXT[], "Region" TEXT, "Region_Info" TEXT[],
      "Population" INT);
User: Which gulf is north of the Somalian's city with 550,000 residents ?
Target: SELECT answer("City_Info", 'Which gulf is north of this Somalian''s city ?') FROM "validation_table_8" WHERE
      "Population" = '550,000';
--
List_of_the_mothers_of_the_Ottoman_Sultans_0
CREATE TABLE validation_table_14 ("Name" TEXT, "Name_Info" TEXT[], "Titles" TEXT, "Titles_Info" TEXT[], "Maiden Name" TEXT,
      "Origin" TEXT, "Origin_Info" TEXT[], "Death" DATE, "Son ( s )" TEXT, "Son ( s )_Info" TEXT[]);
User: Who was the husband of the mother of Ottoman sultan Suleiman I ?
Target: SELECT answer("Name_Info", 'Who is her husband?') FROM "validation_table_14" WHERE "Son ( s )" = 'Suleiman I';
--
List_of_Mohun_Bagan_A.C._managers_0
CREATE TABLE validation_table_10 ("Name" TEXT, "Name_Info" TEXT[], "Nationality" TEXT, "Nationality_Info" TEXT[], "FROM"
      DATE, "TO" DATE);
User: What is the nationality of the manager who was born on 15 February 1968 ?
Target: SELECT "Nationality" FROM "validation_table_10" WHERE answer("Name_Info", 'is this manager born on 15 February
      1968?') = 'Yes';
--
Grammy_Award_for_Best_Jazz_Vocal_Performance,_Male_0
CREATE TABLE "validation_table_2615" ("Year" INT, "Year_Info" TEXT[], "Performing artist ( s )" TEXT, "Performing artist ( s
      )_Info" TEXT[], "Work" TEXT, "Work_Info" TEXT[], "Nominees" TEXT, "Nominees_Info" TEXT[])
User: How many people performed on the most recent song to win ?
Target: SELECT answer("Work_Info", 'how many people performed on this song?') FROM "validation_table_2615" ORDER BY "Year"
      DESC LIMIT 1;
--
List_of_flag_bearers_for_Myanmar_at_the_Olympics_0
CREATE TABLE validation_table_67 ("Name" TEXT, "Event Year" INT, "Year_Info" TEXT[], "Season" TEXT, "Flag Bearer" TEXT, "Flag
      Bearer_Info" TEXT[]);
User: When did the youngest Burmese flag bearer participate in the Olympic opening ceremony?
Target: SELECT "Event Year" FROM validation_table_67 ORDER BY answer("Flag Bearer_Info", 'when is this person born?')::date
      DESC LIMIT 1;
--
List_of_museums_in_Atlanta_0
CREATE TABLE validation_table_3 ("Name" TEXT, "Name_Info" TEXT[], "Area" TEXT, "Area_Info" TEXT[], "Type" TEXT, "Summary"
      TEXT, "Summary_Info" TEXT[]);
User: What is that address of the museum located in a Victorian House in an area whose Architectural styles within the
      district include Craftsman Bungalow , Queen Anne , Stick style , Folk Victorian , Colonial Revival , American
      Foursquare and Neoclassical Revival ?
Target: SELECT answer("Name_Info", 'what is the address?') FROM "validation_table_19" WHERE answer("Area_Info", 'is this an
      area whose Architectural styles within the district include Craftsman Bungalow , Queen Anne , Stick style , Folk
      Victorian , Colonial Revival , American Foursquare and Neoclassical Revival ?') = 'Yes';
--
2007_in_Canadian_music_0
CREATE TABLE "validation_table_26" ("Rank" INT, "Artist" TEXT, "Artist_Info" TEXT[], "Album" TEXT, "Album_Info" TEXT[], "Peak
      position" INT, "Sales" INT, "Certification" TEXT)
User: How many purchases of albums by the musician with the record Call Me Irresponsible have occurred ?
Target: SELECT answer("Artist_Info", 'How many albums has this artist sold?') FROM "validation_table_26" WHERE
      answer("Album_Info", 'is this record Call Me Irresponsible?') = 'Yes';
--
List_of_Indian_state_flowers_0
CREATE TABLE "validation_table_74" ("State" TEXT, "State_Info" TEXT[], "Common name" TEXT, "Common name_Info" TEXT[],
      "Binomial name" TEXT, "Binomial name_Info" TEXT[])
User:What is the state flower of the smallest state by area ?
Target: SELECT "Common name" FROM "validation_table_74" WHERE answer("State_Info", 'is this the smallest state by area?') =
      'Yes';
--
List_of_Turner_Prize_winners_and_nominees_0
CREATE TABLE "validation_table_78" ("Year" INT, "Winner" TEXT, "Winner_Info" TEXT[], "Format" TEXT, "Nominees" TEXT,
      "Nominees_Info" TEXT[], "Notes" TEXT, "Notes_Info" TEXT[])
User: In what year did the 1999 Turner Prize winner win the Academy Award for his film , 12 Years a Slave ?
Target: SELECT answer("Winner_Info", 'in what year did he win the Academy Award for his film, 12 Years a Slave?') FROM
      "validation_table_78" WHERE answer("Winner_Info", 'did he win the Academy Award for his film, 12 Years a Slave?') =
      'Yes' AND "Year" = '1999';
--
{{ table_original_name }}
{{ create_cmd }}
User: {{ query }}
Target:
```

Table 9: HybridQA semantic parser prompt. This prompt contains 10 examples, each with a (1) short table description, (2) table schema shown as a CREATE command, (3) the input query, and (4) the target SUQL

```
You are a SQL semantic parser. In a prior turn, you have predicted a SQL, which returned no results. Your job now is to
      generate a new SQL to try again.
In addition to the standard SQL syntax, you can make use of the `answer` function.

In general, you should try to RELAX constraints.

Table description: Doping_at_the_Olympic_Games_15
Schema: CREATE TABLE "validation_table_56" ("Name" TEXT, "Name_Info" TEXT[], "Country" TEXT, "Country_Info" TEXT[], "Sport"
      TEXT, "Sport_Info" TEXT[], "Banned substance" TEXT, "Banned substance_Info" TEXT[])
Question: What substance was the athlete born in Bugulma banned in 2002 for using ?
Previously-generated SQL: SELECT "Banned substance" FROM "validation_table_56" WHERE answer("Name_Info", 'is this athlete
      born in Bugulma?') = 'Yes' AND "Country_Info" @> ARRAY['2002'];
This SQL returned no result.
New SQL: SELECT "Banned substance" FROM "validation_table_56" WHERE answer("Name_Info", 'is this athlete born in Bugulma and
      banned in 2002?') = 'Yes';
--
Table description: Sweden_at_the_1932_Summer_Olympics_0
Schema: CREATE TABLE "validation_table_1" ("Medal" TEXT, "Name" TEXT, "Name_Info" TEXT[], "Sport" TEXT, "Sport_Info" TEXT[],
      "Event" TEXT, "Event_Info" TEXT[])
Question: What was the nickname of the gold medal winner in the men 's heavyweight greco-roman wrestling event of the 1932
      Summer Olympics ?
Previously-generated SQL: SELECT answer("Name_Info", 'What was his nickname?') FROM "validation_table_1" WHERE "Medal" =
      'Gold' AND "Event" = 'Men''s heavyweight Greco-Roman wrestling';
This SQL returned no result.
New SQL: SELECT answer("Name_Info", 'What was his nickname?') FROM "validation_table_1" WHERE "Medal" = 'Gold' AND "Event" =
      'Men''s heavyweight' AND "Sport" = 'Greco-Roman wrestling';
--
Table description: 2011_Berlin_Marathon_0
Schema: CREATE TABLE "validation_table_4" ("Position" INT, "Athlete" TEXT, "Athlete_Info" TEXT[], "Nationality" TEXT,
      "Nationality_Info" TEXT[], "Time" TIME)
Question: What place was achieved by the person who finished the Berlin marathon in 2:13.32 in 2011 the first time he
      competed in a marathon ?
Previously-generated SQL: SELECT "Position" FROM "validation_table_4" WHERE "Time" = '2:13:32' AND answer("Athlete_Info", 'is
      this the first time this person competed in a marathon?') = 'Yes';
This SQL returned no result.
New SQL: SELECT "Position" FROM "validation_table_4" WHERE "Time" = '2:13:32';
--
Table description: List_of_Pi_Kappa_Alpha_brothers_5
Schema: CREATE TABLE "validation_table_37" ("Name" TEXT, "Name_Info" TEXT[], "Original chapter" TEXT, "Original chapter_Info"
      TEXT[], "Notability" TEXT, "Notability_Info" TEXT[])
Question: What year was the brother from Beta Omicron born ?
Previously-generated SQL: SELECT answer("Name_Info", 'what year was this brother born?') FROM "validation_table_37" WHERE
      "Original chapter" = 'Beta Omicron';
This SQL returned no result.
New SQL: SELECT answer("Name_Info", 'what year was this person born?') FROM "validation_table_37" WHERE "Original chapter" =
      'Beta Omicron';
--
Table description: List_of_radio_stations_in_the_United_Kingdom_15
Schema: CREATE TABLE "validation_table_55" ("Name" TEXT, "Name_Info" TEXT[], "Licence area" TEXT, "Licence area_Info" TEXT[],
      "Analogue frequencies" FLOAT, "Notes" TEXT)"
Question: Which station broadcasts to a civil parish in north west Dorset sited on the River Yeo ?
Previously-generated SQL: SELECT "Name" FROM "validation_table_55" WHERE answer("Licence area_Info", 'does this station
      broadcast to a civil parish in north west Dorset sited on the River Yeo?') = 'Yes';
This SQL returned no result.
New SQL: SELECT "Name" FROM "validation_table_55" WHERE answer("Licence area_Info", 'is this a civil parish in north west
      Dorset sited on the River Yeo?') = 'Yes';
--
Table description: {{ description }}
Schema: {{ schema }}
Question: {{ question }}
Previously-generated SQL: {{ previous_sql }}
This SQL returned no result.
{% if second_previous_sql is not none %}
    You also generated: {{ second_previous_sql }}
    This SQL also returned no result.
{% endif %}
New SQL:
```

Table 10: HybridQA no result recovery prompt. This prompt contains 5 examples, each with a (1) short table description, (2) table schema shown as a CREATE command, (3) the input query, (4) a previously generated SUQL which returned no results, and (5) the target SUQL.

```
You are a good answer extractor. Given a detailed answer to a question, you always extract an succinct answer. If no valid
      answers can be extracted, answer with "No Info". Do not generate answers that is not from the original detailed answer.
      The succinct answer should be the minimum span from the passage without modification. When copying the answer, do not
      use a half word.

Question: The driver who finished in position 4 in the 2004 United States Grand Prix was of what nationality ?
Detailed Answer: The driver, Jenson Alexander Lyons Button, is British.
Succinct Answer: British
--
Question: What is that address of the museum located in a Victorian House in an area whose Architectural styles within the
      district include Craftsman Bungalow , Queen Anne , Stick style , Folk Victorian , Colonial Revival , American
      Foursquare and Neoclassical Revival ?
Detailed Answer: The address of the Hammonds House Museum is 503 Peeples Street SW in the West End neighborhood of Atlanta,
      Georgia.
Succinct Answer: 503 Peeples Street SW
--
Question: What is the area of the national park whose terrain is extremely rugged and consists of sandstone peaks , narrow
      gorges , ravines and dense forests , in kilometers ?
Detailed Answer: 524 km
Succinct Answer: 524
--
Question: Which gulf is north of the Somalian city with 550,000 residents ?
Detailed Answer: The Gulf of Aden is north of this city.
Succinct Answer: Gulf of Aden
--
Question: Who was the husband of the mother of Ottoman sultan Suleiman I ?
Detailed Answer: Her husband is Selim I.
Succinct Answer: Selim I
--
Question: What are the symptoms of the titular syndrome in his 2009 movie ?
Detailed Answer: The text does not provide information on the symptoms of any syndrome.
Succinct Answer: No Info
--
Question: {{ query }}
Detailed Answer: {{ detailed_answer }}
Succinct Answer:
```

Table 11: HybridQA format extractor prompt. This prompt contains 6 examples, each with a (1) input query, (2) a detailed answer from SUQL, and (3) a target succinct answer.

```
You are a restaurant virtual assistant chatting with a user.
You can access a restaurant database to retrieve information about restaurants' cuisine, price (cheap, moderate, expensive,
      luxury), rating (1-5), num_reviews, location, popular_dishes, reviews, phone_number, and opening_hours.

=====
{# basic #}
You: Hi! How can I help you?
They: what is a good place to get brunch in Chicago?
[Check the database? Yes]
=====
{# ask for rating #}
They: show me a Chinese restaurant in upper east side, NY
You: I found the 4.5 star Calle Dao Chelsea. It is a Chinese and Asian Fusion restaurant that serves brunch and has a menu
      that reflects the history of Chinese migration to Cuba.
They: is it better than panda express?
[Check the database? Yes]
=====
{# follow up on phone number #}
They: what is a good seafood restaurant in Seattle?
You: I found The Pink Door, a 4.5 star seafood restaurant in Seattle. Reviews mention the grilled whole fish and clam pasta
      as seafood dishes served there.
They: Can you find their phone number?
[Check the database? Yes]
=====
{# have you heard? #}
They: have you heard of girl and the goat at Chicago?
[Check the database? Yes]
=====
{# do not check db when asking about general knowledge #}
They: I want a Spanish restaurant in Kansas City
You: I found the 4 star La Bodega. It has one of the best happy hours in the city and a cozy atmosphere that feels like
      you're in the countryside of Spain.
They: Do you speak Spanish?
[Check the database? No]
=====
{# do not check db when asking about general knowledge about dishes #}
They: can you find me a place that serves peking duck?
You: Sorry, I don't have that information.
They: ok, how about orange chicken instead?
You: Sorry, I don't have that information.
They: how about chicken?
You: I found the 4 star Roost & Roast. It is a take-out place with no indoor dining, but there are tables outside. The menu
      is simple and ordering is done via an iPad kiosk. They offer Thai-inspired dishes such as Hat Yai Fried Chicken, Pad
      Thai, and several other stir frys.
They: what is hat yai fried chicken?
[Check the database? No]
=====
{# show another one #}
They: hey! show me something in Washington D.C.
You: I found the 4 star Old Ebbitt Grill. It is a historic restaurant located in Washington D.C. that has been serving locals
      and tourists since 1856. The food is very good and the service is great, with a friendly and knowledgeable staff.
They: is there another one?
[Check the database? Yes]
=====
{# keep on checking database even if there are errors #}
They: have you heard of girl and the goat at Chicago?
You: Sorry. I don't have that information
They: have you heard of girl and the goat at Chicago?
[Check the database? Yes]
=====
You: Hi! How can I help you?
{% for dlg_turn in dlg %}
They: {{ dlg_turn.user_utterance }}
{% if dlg_turn.agent_utterance is not none %}
You: {{ dlg_turn.agent_utterance }}
{% endif %}
{% endfor %}
[Check the database?
```

Table 12: Input Classifier prompt for the restaurant experiment. This prompt contains 8 examples.

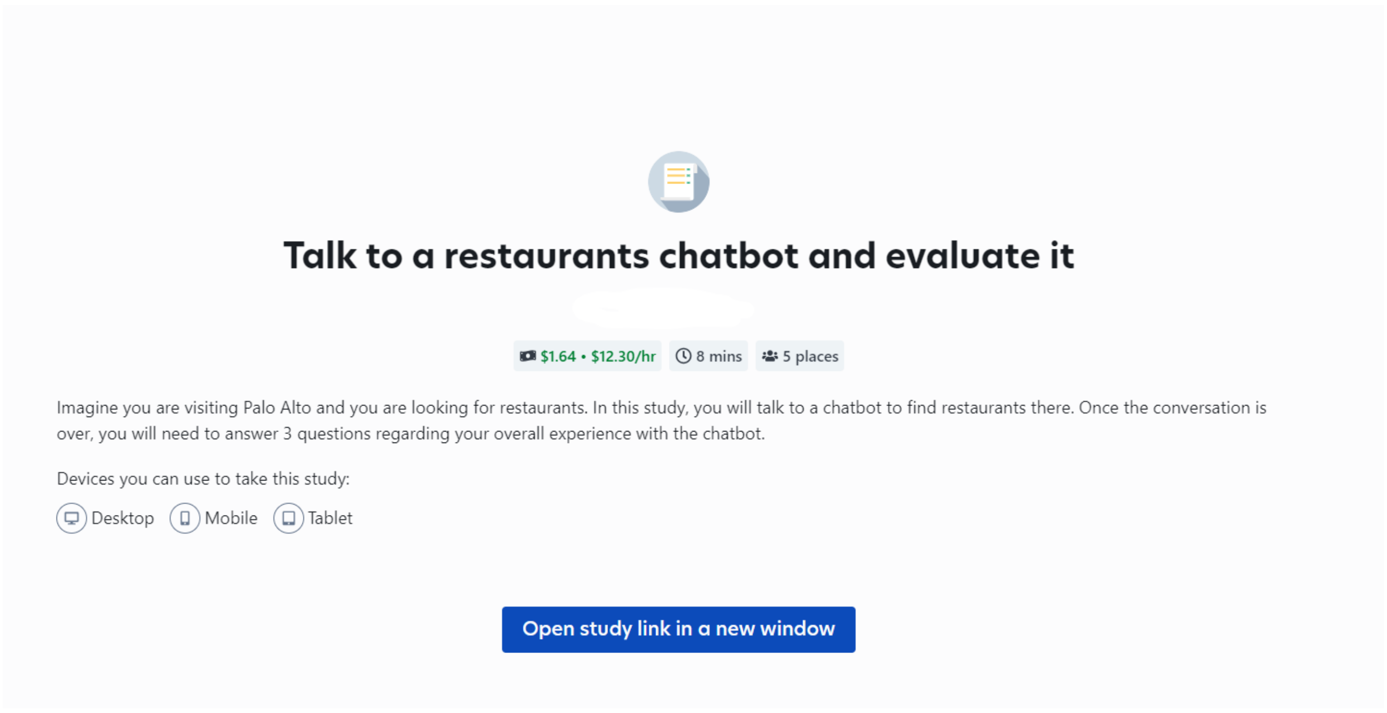
```
You are a semantic parser. Generate a query for a restaurant database with the following signature:

CREATE TABLE restaurants (
    name TEXT,
    cuisines TEXT[],
    price ENUM ('cheap', 'moderate', 'expensive', 'luxury'),
    rating NUMERIC(2,1),
    num_reviews NUMBER,
    address TEXT,
    popular_dishes FREE_TEXT,
    phone_number TEXT,
    reviews FREE_TEXT,
    opening_hours TEXT,
    location TEXT
);

Do not generate fields beyond the given fields. The `answer` function can be used on FREE_TEXT fields.

{# Basic example #}
User: Where is Burguer King?
Target: SELECT address, summary(reviews) FROM restaurants WHERE name ILIKE '%Burguer King%' LIMIT 1;
--
{# Basic example for cuisine, and follow up with restaurant names #}
User: what are some good-reviewed japanese restaurants in Kansas City?
Target: SELECT *, summary(reviews) FROM restaurants WHERE 'japanese' = ANY (cuisines) AND location = 'Kansas City' AND rating
      >= 4.0 LIMIT 3;
Agent: I found Sakura Sushi, Nami Ramen, and Kaze Teppanyaki.
User: What are their prices?
Target: SELECT name, price FROM restaurants WHERE (name ILIKE 'Sakura Sushi' OR name ILIKE 'Nami Ramen' OR name ILIKE 'Kaze
      Teppanyaki') AND location = 'Kansas City';
--
{# Usage of `answer` function on FREE TEXT field in both projection and filter #}
User: Show me a family-friendly restaurant that has burgers in D.C.
Target: SELECT *, summary(reviews), answer(reviews, 'is this restaurant family-friendly?') FROM restaurants WHERE
      answer(reviews, 'do you find this restaurant to be family-friendly?') = 'Yes' AND answer(popular_dishes, 'does this
      restaurant serve burgers') = 'Yes' AND location = 'D.C.' LIMIT 1;
Agent: I found Jason's steakhouse. Reviews mention kids love going there with their parents. It should be a great weekend
      dinner for you and your family.
User: What do the reviews say about the atmosphere in the restaurant?
Target: SELECT answer(reviews, 'What is the atmosphere?') FROM restaurants WHERE name ILIKE 'Jason''s steakhouse' AND
      location = 'D.C.' LIMIT 1;
--
{# Usage of `answer` function on popular_dishes #}
User: Find me a place with pasta in Nashville.
Target: SELECT *, summary(reviews) FROM restaurants WHERE answer(popular_dishes, 'does this restaurant serve pasta') = 'Yes'
      AND location = 'Nashville' LIMIT 1;
--
{# Usage of `answer` function on reviews #}
User: I love Chinese food. Find me a restaurant that doesn't have a long wait time.
Target: SELECT *, summary(reviews), answer(reviews, 'what is the wait time?') FROM restaurants WHERE 'chinese' = ANY
      (cuisines) AND answer(reviews, 'does this restaurant have short wait time?') = 'Yes' LIMIT 1;
--
{# Meaning of the word "popular", and follow up on fetching reviews #}
User: I want a popular restaurant in Napa, CA.
Target: SELECT *, summary(reviews) FROM restaurants WHERE rating >= 4.5 AND location = 'Napa, CA' ORDER BY num_reviews DESC
      LIMIT 1;
Agent: I found the 5.0 star Gui's vegan house. It has 2,654 reviews and reviews mention great atmosphere, quick and good
      service, and good food quality.
User: Give me the review that talk about good food quality.
Target: SELECT single_review FROM restaurants AS r, unnest(reviews) AS single_review WHERE name ILIKE 'Gui''s vegan house'
      AND answer(single_review, 'does this review mention good food quality?') = 'Yes' AND r.location = 'Napa, CA' LIMIT 1;
--
{# Usage of `answer` function on reviews #}
User: Which restaurants have a happy hour in Bakersfield?
Target: SELECT *, summary(reviews), answer(reviews, 'what is the happy hour here?') FROM restaurants WHERE location =
      'Bakersfield' AND answer(reviews, 'does this restaurant have a happy hour?') = 'Yes' LIMIT 1;
--
{# Usage of `answer` function on reviews #}
User: i'm hungry, what should i have for lunch? I am looking for salmon in Chicago.
Target: SELECT *, summary(reviews) FROM restaurants WHERE answer(popular_dishes, 'does this restaurant serve salmon?') =
      'Yes' AND location = 'Chicago' LIMIT 1;
Agent: I found the 4.5 star Daigo. It is a family-owned business that serves traditional Japanese cuisine.
User: Show me something else.
Target: SELECT *, summary(reviews) FROM restaurants WHERE NOT(name = 'Daigo') AND answer(popular_dishes, 'does this
      restaurant serve salmon?') = 'Yes' AND location = 'Chicago' LIMIT 1;
--
{% for dlg_turn in dlg[:-1] %}
{% if dlg_turn.genie_utterance is not none %}
User: {{ dlg_turn.user_utterance }}
Target: {{ dlg_turn.user_target }}
Agent: {{ dlg_turn.agent_utterance }}
{% endif %}
{% endfor %}
User: {{ query }}
Target:
```

Table 13: The semantic parser prompt for the restaurant experiment. This prompt contains 8 examples.



Talk to a restaurants chatbot and evaluate it

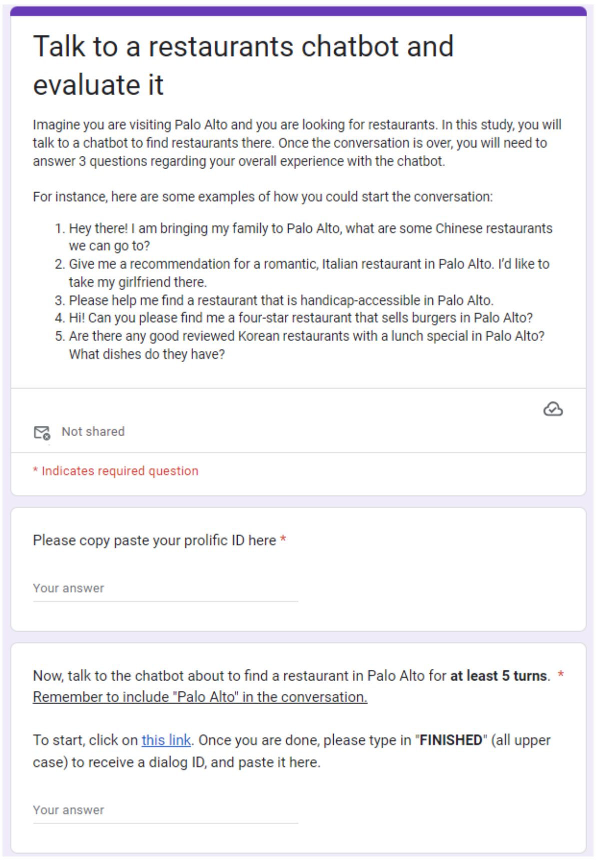
$1.64 • $12.30/hr · 8 mins · 5 places

Imagine you are visiting Palo Alto and you are looking for restaurants. In this study, you will talk to a chatbot to find restaurants there. Once the conversation is over, you will need to answer 3 questions regarding your overall experience with the chatbot.

Devices you can use to take this study:

Desktop · Mobile · Tablet

Figure 3: The crowdsourcing interface that our user sees



Talk to a restaurants chatbot and evaluate it

Imagine you are visiting Palo Alto and you are looking for restaurants. In this study, you will talk to a chatbot to find restaurants there. Once the conversation is over, you will need to answer 3 questions regarding your overall experience with the chatbot.

For instance, here are some examples of how you could start the conversation:

1. Hey there! I am bringing my family to Palo Alto, what are some Chinese restaurants we can go to?
2. Give me a recommendation for a romantic, Italian restaurant in Palo Alto. I'd like to take my girlfriend there.
3. Please help me find a restaurant that is handicap-accessible in Palo Alto.
4. Hi! Can you please find me a four-star restaurant that sells burgers in Palo Alto?
5. Are there any good reviewed Korean restaurants with a lunch special in Palo Alto? What dishes do they have?

Not shared

* Indicates required question

Please copy paste your prolific ID here *

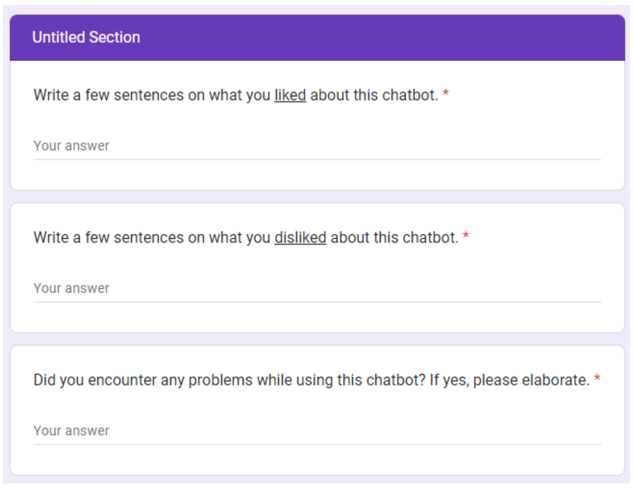
Your answer

Now, talk to the chatbot about to find a restaurant in Palo Alto for **at least 5 turns**. *
Remember to include "Palo Alto" in the conversation.

To start, click on this link. Once you are done, please type in "**FINISHED**" (all upper case) to receive a dialog ID, and paste it here.

Your answer

Figure 4: The prompts we give crowdsourcing workers before they start conversing with our chatbot.


Untitled Section

Write a few sentences on what you liked about this chatbot. *

Your answer

Write a few sentences on what you disliked about this chatbot. *

Your answer

Did you encounter any problems while using this chatbot? If yes, please elaborate. *

Your answer

Figure 5: The questions crowdsourcing workers are asked after they finish talking to the chatbot.