SParC: Cross-Domain Semantic Parsing in Context

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**[¶] **Dragomir Radev**[†]
[†]Department of Computer Science, Yale University
[¶]Salesforce Research
{tao.yu, r.zhang, michihiro.yasunaga, dragomir.radev}@yale.edu
{xilin, cxiong, rsocher}@salesforce.com

Abstract

We present SParC, a dataset for cross-domain **S**emantic **Par**sing in **C**ontext. It consists of 4,298 coherent question sequences (12k+ individual questions annotated with SQL queries), obtained from controlled user interactions with 200 complex databases over 138 domains. We provide an in-depth analysis of SParC and show that it introduces new challenges compared to existing datasets. SParC (1) demonstrates complex contextual dependencies, (2) has greater semantic diversity, and (3) requires generalization to new domains due to its cross-domain nature and the unseen databases at test time. We experiment with two state-of-the-art text-to-SQL models adapted to the context-dependent, cross-domain setup. The best model obtains an exact set match accuracy of 20.2% over all questions and less than 10% over all interaction sequences, indicating that the cross-domain setting and the contextual phenomena of the dataset present significant challenges for future research. The dataset, baselines, and leaderboard are released at https://yale-lily.github.io/sparc.

1 Introduction

Querying a relational database is often challenging and a natural language interface has long been regarded by many as the most powerful database interface (Popescu et al., 2003; Bertomeu et al., 2006; Li and Jagadish, 2014). The problem of mapping a natural language utterance into executable SQL queries (text-to-SQL) has attracted increasing attention from the semantic parsing community by virtue of a continuous effort of dataset creation (Zelle and Mooney, 1996; Iyyer et al., 2017; Zhong et al., 2017; Finegan-Dollak et al., 2018; Yu et al., 2018a) and the modeling innovation that follows it (Xu et al., 2017; Wang et al., 2018; Yu et al., 2018b; Shi et al., 2018).

D_1 : Database about student dormitory containing 5 tables.
C_1 : **Find the first and last names of the students who are living in the dorms that have a TV Lounge as an amenity.**

Q_1 : How many dorms have a TV Lounge?
S_1 :

```
SELECT COUNT(*) FROM dorm AS T1 JOIN has_amenity
AS T2 ON T1.dormid = T2.dormid JOIN dorm_amenity
AS T3 ON T2.amenid = T3.amenid WHERE
T3.amenity_name = 'TV Lounge'
```

Q_2 : What is the total capacity of these dorms?
S_2 :

```
SELECT SUM(T1.student_capacity) FROM dorm AS T1
JOIN has_amenity AS T2 ON T1.dormid = T2.dormid
JOIN dorm_amenity AS T3 ON T2.amenid = T3.amenid
WHERE T3.amenity_name = 'TV Lounge'
```

Q_3 : How many students are living there?
S_3 :

```
SELECT COUNT(*) FROM student AS T1 JOIN lives_in
AS T2 ON T1.stuid = T2.stuid WHERE T2.dormid IN
(SELECT T3.dormid FROM has_amenity AS T3 JOIN
dorm_amenity AS T4 ON T3.amenid = T4.amenid WHERE
T4.amenity_name = 'TV Lounge')
```

Q_4 : Please show their first and last names.
S_4 :

```
SELECT T1.fname, T1.lname FROM student AS T1 JOIN
lives_in AS T2 ON T1.stuid = T2.stuid WHERE
T2.dormid IN (SELECT T3.dormid FROM has_amenity
AS T3 JOIN dorm_amenity AS T4 ON T3.amenid =
T4.amenid WHERE T4.amenity_name = 'TV Lounge')
```

D_2 : Database about shipping company containing 13 tables
C_2 : **Find the names of the first 5 customers.**

Q_1 : What is the customer id of the most recent customer?
S_1 :

```
SELECT customer_id FROM customers ORDER BY
date_became_customer DESC LIMIT 1
```

Q_2 : What is their name?
S_2 :

```
SELECT customer_name FROM customers ORDER BY
date_became_customer DESC LIMIT 1
```

Q_3 : How about for the first 5 customers?
S_3 :

```
SELECT customer_name FROM customers ORDER BY
date_became_customer LIMIT 5
```

Figure 1: Two question sequences from the SParC dataset. Questions (Q_i) in each sequence query a database (D_i), obtaining information sufficient to complete the interaction goal (C_i). Each question is annotated with a corresponding SQL query (S_i). SQL token sequences from the interaction context are underlined.

While most of these work focus on precisely mapping stand-alone utterances to SQL queries, generating SQL queries in a context-dependent scenario (Miller et al., 1996; Zettlemoyer and

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4511–4523
Florence, Italy, July 28 - August 2, 2019. ©2019 Association for Computational Linguistics

Collins, 2009; Suhr et al., 2018) has been studied less often. The most prominent context-dependent text-to-SQL benchmark is ATIS[1], which is set in the flight-booking domain and contains only one database (Hemphill et al., 1990; Dahl et al., 1994).

In a real-world setting, users tend to ask a sequence of thematically related questions to learn about a particular topic or to achieve a complex goal. Previous studies have shown that by allowing questions to be constructed sequentially, users can explore the data in a more flexible manner, which reduces their cognitive burden (Hale, 2006; Levy, 2008; Frank, 2013; Iyyer et al., 2017) and increases their involvement when interacting with the system. The phrasing of such questions depends heavily on the interaction history (Kato et al., 2004; Chai and Jin, 2004; Bertomeu et al., 2006). The users may explicitly refer to or omit previously mentioned entities and constraints, and may introduce refinements, additions or substitutions to what has already been said (Figure 1). This requires a practical text-to-SQL system to effectively process context information to synthesize the correct SQL logic.

To enable modeling advances in context-dependent semantic parsing, we introduce SParC (cross-domain **S**emantic **Par**sing in **C**ontext), an expert-labeled dataset which contains 4,298 coherent question sequences (12k+ questions paired with SQL queries) querying 200 complex databases in 138 different domains. The dataset is built on top of Spider[2], the largest cross-domain context-independent text-to-SQL dataset available in the field (Yu et al., 2018c). The large number of domains provide rich contextual phenomena and thematic relations between the questions, which general-purpose natural language interfaces to databases have to address. In addition, it enables us to test the generalization of the trained systems to unseen databases and domains.

We asked 15 college students with SQL experience to come up with question sequences over the Spider databases (§ 3). Questions in the original Spider dataset were used as guidance to the students for constructing meaningful interactions: each sequence is based on a question in Spider and the student has to ask inter-related questions to obtain information that answers the Spider question. At the same time, the students are encouraged to come up with related questions which do not directly contribute to the Spider question so as to increase data diversity. The questions were subsequently translated to complex SQL queries by the same student. Similar to Spider, the SQL Queries in SParC cover complex syntactic structures and most common SQL keywords.

We split the dataset such that a database appears in only one of the train, development and test sets. We provide detailed data analysis to show the richness of SParC in terms of semantics, contextual phenomena and thematic relations (§ 4). We also experiment with two competitive baseline models to assess the difficulty of SParC (§ 5). The best model achieves only 20.2% exact set matching accuracy[3] on all questions, and demonstrates a decrease in exact set matching accuracy from 38.6% for questions in turn 1 to 1.1% for questions in turns 4 and higher (§ 6). This suggests that there is plenty of room for advancement in modeling and learning on the SParC dataset.

2 Related Work

Context-independent semantic parsing Early studies in semantic parsing (Zettlemoyer and Collins, 2005; Artzi and Zettlemoyer, 2013; Berant and Liang, 2014; Li and Jagadish, 2014; Pasupat and Liang, 2015; Dong and Lapata, 2016; Iyer et al., 2017) were based on small and single-domain datasets such as ATIS (Hemphill et al., 1990; Dahl et al., 1994) and GeoQuery (Zelle and Mooney, 1996). Recently, an increasing number of neural approaches (Zhong et al., 2017; Xu et al., 2017; Yu et al., 2018a; Dong and Lapata, 2018; Yu et al., 2018b) have started to use large and cross-domain text-to-SQL datasets such as WikiSQL (Zhong et al., 2017) and Spider (Yu et al., 2018c). Most of them focus on converting stand-alone natural language questions to executable queries. Table 1 compares SParC with other semantic parsing datasets.

Context-dependent semantic parsing with SQL labels Only a few datasets have been constructed for the purpose of mapping context-dependent questions to structured queries.

[1] A subset of ATIS is also frequently used in context-independent semantic parsing research (Zettlemoyer and Collins, 2007; Dong and Lapata, 2016).

[2] The data is available at `https://yale-lily.github.io/spider`.

[3] Exact string match ignores ordering discrepancies of SQL components whose order does not matter. Exact set matching is able to consider ordering issues in SQL evaluation. See more evaluation details in section 6.1.

Dataset	Context	Resource	Annotation	Cross-domain
SParC	✓	database	SQL	✓
ATIS (Hemphill et al., 1990; Dahl et al., 1994)	✓	database	SQL	✗
Spider (Yu et al., 2018c)	✗	database	SQL	✓
WikiSQL (Zhong et al., 2017)	✗	table	SQL	✓
GeoQuery (Zelle and Mooney, 1996)	✗	database	SQL	✗
SequentialQA (Iyyer et al., 2017)	✓	table	denotation	✓
SCONE (Long et al., 2016)	✓	environment	denotation	✗

Table 1: Comparison of SParC with existing semantic parsing datasets.

Hemphill et al. (1990); Dahl et al. (1994) collected the contextualized version of ATIS that includes series of questions from users interacting with a flight database. Adopted by several works later on (Miller et al., 1996; Zettlemoyer and Collins, 2009; Suhr et al., 2018), ATIS has only a single domain for flight planning which limits the possible SQL logic it contains. In contrast to ATIS, SParC consists of a large number of complex SQL queries (with most SQL syntax components) inquiring 200 databases in 138 different domains, which contributes to its diversity in query semantics and contextual dependencies. Similar to Spider, the databases in the train, development and test sets of SParC do not overlap.

Context-dependent semantic parsing with denotations Some datasets used in recovering context-dependent meaning (including SCONE (Long et al., 2016) and SequentialQA (Iyyer et al., 2017)) contain no logical form annotations but only denotation (Berant and Liang, 2014) instead. SCONE (Long et al., 2016) contains some instructions in limited domains such as chemistry experiments. The formal representations in the dataset are world states representing state changes after each instruction instead of programs or logical forms. SequentialQA (Iyyer et al., 2017) was created by asking crowd workers to decompose some complicated questions in WikiTableQuestions (Pasupat and Liang, 2015) into sequences of inner-related simple questions. As shown in Table 1, neither of the two datasets were annotated with query labels. Thus, to make the tasks feasible, SCONE (Long et al., 2016) and SequentialQA (Iyyer et al., 2017) exclude many questions with rich semantic and contextual types. For example, (Iyyer et al., 2017) requires that the answers to the questions in SequentialQA must appear in the table, and most of them can be solved by simple SQL queries with `SELECT` and `WHERE` clauses. Such direct mapping without formal query labels becomes unfeasible for complex questions. Furthermore, SequentialQA contains questions based only on a single Wikipedia tables at a time. In contrast, SParC contains 200 significantly larger databases, and complex query labels with all common SQL key components. This requires a system developed for SParC to handle information needed over larger databases in different domains.

Conversational QA and dialogue system Language understanding in context is also studied for dialogue and question answering systems. The development in dialogue (Henderson et al., 2014; Mrkšić et al., 2017; Zhong et al., 2018) uses predefined ontology and slot-value pairs with limited natural language meaning representation, whereas we focus on general SQL queries that enable more powerful semantic meaning representation. Recently, some conversational question answering datasets have been introduced, such as QuAC (Choi et al., 2018) and CoQA (Reddy et al., 2018). They differ from SParC in that the answers are free-form text instead of SQL queries. On the other hand, Kato et al. (2004); Chai and Jin (2004); Bertomeu et al. (2006) conduct early studies of the contextual phenomena and thematic relations in database dialogue/QA systems, which we use as references when constructing SParC.

3 Data Collection

We create the SParC dataset in four stages: selecting interaction goals, creating questions, annotating SQL representations, and reviewing.

Interaction goal selection To ensure thematic relevance within each question sequence, we use questions in the original Spider dataset as the thematic guidance for constructing meaningful query interactions, i.e. the interaction goal. Each sequence is based on a question in Spider and the an-

Thematic relation	Description	Example	Percentage
Refinement (constraint refinement)	The current question asks for the same type of entity as a previous question with a different constraint.	Prev_Q: Which **major** has the fewest students? Cur_Q: What is the most popular one?	33.8%
Theme-entity (topic exploration)	The current question asks for other properties about the same entity as a previous question.	Prev_Q: What is *the capacity* of **Anonymous Donor Hall**? Cur_Q: List *all of the amenities* which **it** has.	48.4%
Theme-property (participant shift)	The current question asks for the same property about another entity.	Prev_Q: Tell me the *rating* of **the episode named "Double Down"**. Cur_Q: How about for **"Keepers"**?	9.7%
Answer refinement/theme (answer exploration)	The current question asks for a subset of the entities given in a previous answer or asks about a specific entity introduced in a previous answer.	Prev_Q: Please list all the different **department** *names*. Cur_Q: What is the *average salary* of all instructors in the **Statistics department**?	8.1%

Table 2: Thematic relations between questions in a database QA system defined by Bertomeu et al. (2006). The first three relations hold between a question and a previous question and the last relation holds between a question and a previous answer. We manually classified 102 examples in SParC into one or more of them and show the distribution. The entities (**bold**), properties (*italics*) and constraints (underlined) are highlighted in each question.

notator has to ask inter-related questions to obtain the information demanded by the interaction goal (detailed in the next section). All questions in Spider were stand-alone questions written by 11 college students with SQL background after they had explored the database content, and the question intent conveyed is likely to naturally arise in real-life query scenarios. We selected all Spider examples classified as medium, hard, and extra hard, as it is in general hard to establish context for easy questions. In order to study more diverse information needs, we also included some easy examples (end up with using 12.9% of the easy examples in Spider). As a result, 4,437 questions were selected as the interaction goals for 200 databases.

Question creation 15 college students with SQL experience were asked to come up with sequences of inter-related questions to obtain the information demanded by the interaction goals[4]. Previous work (Bertomeu et al., 2006) has characterized different thematic relations between the utterances in a database QA system: ***refinement***, ***theme-entity***, ***theme-property***, and ***answer refinement/theme***[5], as shown in Table 2. We show these definitions to the students prior to question creation to help them come up with context-dependent questions. We also encourage the formulation of questions that are thematically related to but do not directly contribute to answering the goal question (e.g. Q_2 in the first example and Q_1 in the second example in Figure 1. See more examples in Appendix as well). The students do not simply decompose the complex query. Instead, they often explore the data content first and even change their querying focuses. Therefore, all interactive query information in SParC could not be acquired by a single complex SQL query.

We divide the goals evenly among the students and each interaction goal is annotated by one student[6]. We enforce each question sequence to contain at least two questions, and the interaction terminates when the student has obtained enough information to answer the goal question.

SQL annotation After creating the questions, each annotator was asked to translate their own questions to SQL queries. All SQL queries were executed on Sqlite Web to ensure correctness. To make our evaluation more robust, the same annotation protocol as Spider (Yu et al., 2018c) was adopted such that all annotators chose the same SQL query pattern when multiple equivalent queries were possible.

Data review and post-process We asked students who are native English speakers to review the annotated data. Each example was reviewed at least once. The students corrected any grammar errors and rephrased the question in a more natural way if necessary. They also checked if the

[4] The students were asked to spend time exploring the database using a database visualization tool powered by Sqlite Web `https://github.com/coleifer/sqlite-web` so as to create a diverse set of thematic relations between the questions.

[5] We group *answer refinement* and *answer theme*, the two thematic relations holding between a question and a previous answer as defined in Bertomeu et al. (2006), into a single *answer refinement/theme* type.

[6] The most productive student annotated 13.1% of the goals and the least productive student annotated close to 2%.

	SParC	ATIS
Sequence #	4298	1658
Question #	12,726	11,653
Database #	200	1
Table #	1020	27
Avg. Q len	8.1	10.2
Vocab #	3794	1582
Avg. turn #	3.0	7.0

Table 3: Comparison of the statistics of context-dependent text-to-SQL datasets.

questions in each sequence were related and the SQL answers matched the semantic meaning of the question. After that, another group of students ran all annotated SQL queries to make sure they were executable. Furthermore, they used the SQL parser[7] from Spider to parse all the SQL labels to make sure all queries follow the annotation protocol. Finally, the most experienced annotator conducted a final review on all question-SQL pairs. 139 question sequences were discarded in this final step due to poor question quality or wrong SQL annotations

4 Data Statistics and Analysis

We compute the statistics of SParC and conduct a through data analysis focusing on its contextual dependencies, semantic coverage and cross-domain property. Throughout this section, we compare SParC to ATIS (Hemphill et al., 1990; Dahl et al., 1994), the most most widely used context-dependent text-to-SQL dataset in the field. In comparison, SParC is significantly different as it (1) contains mode complex contextual dependencies, (2) has greater semantic coverage, and (3) adopts a cross-domain task setting, which make it a new and challenging cross-domain context-dependent text-to-SQL dataset.

Data statistics Table 3 summarizes the statistics of SParC and ATIS. SParC contains 4,298 unique question sequences, 200 complex databases in 138 different domains, with 12k+ questions annotated with SQL queries. The number of sequences in ATIS is significantly smaller, but it contains a comparable number of individual questions since it has a higher number of turns per sequence[8].

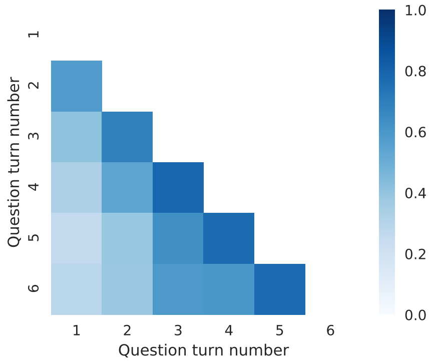

Figure 2: The heatmap shows the percentage of SQL token overlap between questions in different turns. Token overlap is greater between questions that are closer to each other and the degree of overlap increases as interaction proceeds. Most questions have dependencies that span 3 or fewer turns.

On the other hand, SParC has overcome the domain limitation of ATIS by covering 200 different databases and has a significantly larger natural language vocabulary.

Contextual dependencies of questions We visualize the percentage of token overlap between the SQL queries (formal semantic representation of the question) at different positions of a question sequence. The heatmap shows that more information is shared between two questions that are closer to each other. This sharing increases among questions in later turns, where users tend to narrow down their questions to very specific needs. This also indicates that resolving context references in our task is important.

Furthermore, the lighter color of the lower left 4 squares in the heatmap of Figure 2 shows that most questions in an interaction have contextual dependencies that span within 3 turns. Reddy et al. (2018) similarly report that the majority of context dependencies on the CoQA conversational question answering dataset are within 2 questions, beyond which coreferences from the current question are likely to be ambiguous with little inherited

[7] https://github.com/taoyds/spider/blob/master/process_sql.py

[8] The ATIS dataset is collected under the Wizard-of-Oz setting (Bertomeu et al., 2006) (like a task-oriented sequential question answering task). Each user interaction is guided by an abstract, high-level goal such as "plan a trip from city A to city B, stop in another city on the way". The domain by its nature requires the user to express multiple constraints in separate utterances and the user is intrinsically motivated to interact with the system until the booking is successful. In contrast, the interaction goals formed by Spider questions are for open-domain and general-purpose database querying, which tend to be more specific and can often be stated in a smaller number of turns. We believe these differences contribute to the shorter average question sequence length of SParC compared to that of ATIS.

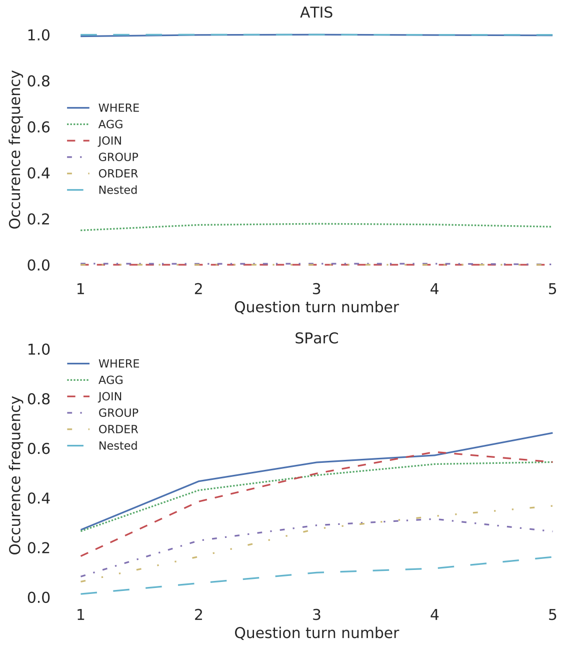

Figure 3: Percentage of question sequences that contain a particular SQL keyword at a given turn. The complexity of questions increases as interaction proceeds on SParC as more SQL keywords are triggered. The same trend was not observed on ATIS.

information. This suggests that 3 turns on average are sufficient to capture most of the contextual dependencies between questions in SParC.

We also plot the trend of common SQL keywords occurring in different question turns for both SParC and ATIS (Figure 3) [9]. We show the percentage of question sequences that contain a particular SQL keyword at each turn. The upper figure in Figure 3 shows that the occurrences of all SQL keywords do not change much as the question turn increases in ATIS, which indicates that the SQL query logic in different turns are very similar. We examined the data and find that most interactions in ATIS involve changes in the `WHERE` condition between question turns. This is likely caused by the fact that the questions in ATIS only involve flight booking, which typically triggers the use of the ***refinement*** thematic relation. Example user utterances from ATIS are *"on american airlines"* or *"which ones arrive at 7pm"* (Suhr et al., 2018), which only involves changes to the `WHERE` condition.

In contrast, the lower figure demonstrates a clear trend that in SParC, the occurrences of nearly all SQL components increase as question turn increases. This suggests that questions in subsequent turns tend to change the logical structures more significantly, which makes our task more interesting and challenging.

Contextual linguistic phenomena We manually inspected 102 randomly chosen examples from our development set to study the thematic relations between questions. Table 2 shows the relation distribution.

We find that the most frequently occurring relation is ***theme-entity***, in which the current question focuses on the same entity (set) as the previous question but requests for some other property. Consider Q_1 and Q_2 of the first example shown in Figure 1. Their corresponding SQL representations (S_1 and S_2) have the same `FROM` and `WHERE` clauses, which harvest the same set of entities – "dorms with a TV lounge". But their `SELECT` clauses return different properties of the target entity set (number of the dorms in S_1 versus total capacity of the dorms in S_2). Q_3 and Q_4 in this example also have the same relation. The ***refinement*** relation is also very common. For example, Q_2 and Q_3 in the second example ask about the same entity set – customers of a shipping company. But Q_3 switches the search constraint from "the most recent" in Q_2 to "the first 5".

Fewer questions refer to previous questions by replacing with another entity ("Double Down" versus "Keepers" in Table 2) (***theme-property***) but asking for the same property. Even less frequently, soem questions ask about the answers of previous questions (***answer refinement/theme***). As in the last example of Table 2, the current question asks about the "Statistics department", which is one of the answers returned in the previous turn. More examples with different thematic relations are provided in Figure 5 in the Appendix.

Interestingly, as the examples in Table 2 have shown, many thematic relations are present without explicit linguistic coreference markers. This indicates information tends to implicitly propagate through the interaction. Moreover, in some cases where the natural language question shares information with the previous question (e.g. Q_2 and Q_3 in the first example of Figure 1 form a ***theme-entity*** relation), the corresponding SQL representations (S_2 and S_3) can be very different.

[9] Since the formatting used for the SQL queries are different in SParC and ATIS, the actual percentages of `WHERE`, `JOIN` and `Nested` for ATIS are lower (e.g. the original version of ATIS may be highly nested, but queries could be reformatted to flatten them). Other SQL keywords are directly comparable.

SQL components	SParC	ATIS
# WHERE	42.8%	99.7%
# AGG	39.8%	16.6%
# GROUP	20.1%	0.3%
# ORDER	17.0%	0.0%
# HAVING	4.7%	0.0%
# SET	3.5%	0.0%
# JOIN	35.5%	99.9%
# Nested	5.7%	99.9%

Table 4: Distribution of SQL components in SQL queries. SQL queries in SParC cover all SQL components, whereas some important SQL components like ORDER are missing from ATIS.

One scenario in which this happens is when the property/constraint specification makes reference to additional entities described by separate tables in the database schema.

Semantic coverage As shown in Table 3, SParC is larger in terms of number of unique SQL templates, vocabulary size and number of domains compared to ATIS. The smaller number of unique SQL templates and vocabulary size of ATIS is likely due to the domain constraint and presence of many similar questions.

Table 4 further compare the formal semantic representation in these two datasets in terms of SQL syntax component. While almost all questions in ATIS contain joins and nested subqueries, some commonly used SQL components are either absent (ORDER BY, HAVING, SET) or occur very rarely (GROUP BY and AGG). We examined the data and find that many questions in it has complicated syntactic structures mainly because the database schema requires joined tables and nested sub-queries, and the semantic diversity among the questions is in fact smaller.

Cross domain As shown in Table 1, SParC contains questions over 200 databases (1,020 tables) in 138 different domains. In comparison, ATIS contains only one databases in the flight booking domain, which makes it unsuitable for developing models that generalize across domains. Interactions querying different databases are shown in Figure 1 (also see more examples in Figure 4 in the Appendix). As in Spider, we split SParC such that each database appears in only one of train, development and test sets. Splitting by database requires the models to generalize well not only to new SQL queries, but also to new databases and new domains.

	Train	Dev	Test
# Q sequences	3034	422	842
# Q-SQL pairs	9025	1203	2498
# Databases	140	20	40

Table 5: Dataset Split Statistics

5 Methods

We extend two state-of-the-art semantic parsing models to the cross-domain, context-dependent setup of SParC and benchmark their performance. At each interaction turn i, given the current question $\bar{x}_i = \langle x_{i,1}, \dots, x_{i,|\bar{x}_i|}\rangle$, the previously asked questions $\bar{I}[: i-1] = \{\bar{x}_1, \dots, \bar{x}_{i-1}\}$ and the database schema C, the model generates the SQL query $\bar{y}_i$.

5.1 Seq2Seq with turn-level history encoder (CD-Seq2Seq)

This is a cross-domain Seq2Seq based text-to-SQL model extended with the turn-level history encoder proposed in Suhr et al. (2018).

Turn-level history encoder Following Suhr et al. (2018), at turn i, we encode each user question $\bar{x}_t \in \bar{I}[: t-1] \cup \{\bar{x}_i\}$ using an utterance-level bi-LSTM, LSTM^E. The final hidden state of LSTM^E, $\mathbf{h}^E_{t,|\bar{x}_t|}$, is used as the input to the turn-level encoder, LSTM^I, a uni-directional LSTM, to generate the discourse state $\mathbf{h}^I_t$. The input to LSTM^E at turn t is the question word embedding concatenated with the discourse state at turn $t-1$ ($[\mathbf{x}_{t,j}, \mathbf{h}^I_{t-1}]$), which enables the propagation of contextual information.

Database schema encoding For each column header in the database schema, we concatenate its corresponding table name and column name separated by a special dot token (i.e., table_name.column_name), and use the average word embedding[10] of tokens in this sequence as the column header embedding $\mathbf{h}^C$.

Decoder The decoder is implemented with another LSTM (LSTM^D) with attention to the LSTM^E representations of the questions in η previous turns. At each decoding step, the decoder chooses to generate either a SQL keyword (e.g., select, where, group by) or a column header. To achieve this, we use separate layers to score SQL keywords and column headers,

[10] We use the 300-dimensional GloVe (Pennington et al., 2014) pretrained word embeddings.

and finally use the softmax operation to generate the output probability distribution over both categories.

5.2 SyntaxSQLNet with history input (SyntaxSQL-con)

SyntaxSQLNet is a syntax tree based neural model for the complex and cross-domain context-independent text-to-SQL task introduced by Yu et al. (2018b). The model consists of a table-aware column attention encoder and a SQL-specific syntax tree-based decoder. The decoder adopts a set of inter-connected neural modules to generate different SQL syntax components.

We extend this model by providing the decoder with the encoding of the previous question ($\bar{x}_{i-1}$) as additional contextual information. Both $\bar{x}_i$ and $\bar{x}_{i-1}$ are encoded using bi-LSTMs (of different parameters) with the column attention mechanism proposed by Yu et al. (2018b). We use the same math formulation to inject the representations of $\bar{x}_i$ and $\bar{x}_{i-1}$ to each syntax module of the decoder.

More details of each baseline model can be found in the Appendix. And we opensource their implementations for reproducibility.

6 Experiments

6.1 Evaluation Metrics

Following Yu et al. (2018c), we use the exact set match metric to compute the accuracy between gold and predicted SQL answers. Instead of simply employing string match, Yu et al. (2018c) decompose predicted queries into different SQL clauses such as `SELECT`, `WHERE`, `GROUP BY`, and `ORDER BY` and compute scores for each clause using set matching separately[11]. We report the following two metrics: *question match*, the exact set matching score over all questions, and *interaction match*, the exact set matching score over all interactions. The exact set matching score is 1 for each question only if all predicted SQL clauses are correct, and 1 for each interaction only if there is an exact set match for every question in the interaction.

6.2 Results

We summarize the overall results of CD-Seq2Seq and SyntaxSQLNet on the development and the test data in Table 6. The context-aware models (CD-Seq2Seq and SyntaxSQL-con) significantly outperforms the standalone SyntaxSQLNet (SyntaxSQL-sta). The last two rows form a controlled ablation study, where without accessing to previous context history, the test set performance of SyntaxSQLNet decreases from 20.2% to 16.9% on *question match* and from 5.2% to 1.1% on *interaction match*, which indicates that context is a crucial aspect of the problem.

Model	Question Match		Interaction Match	
	Dev	Test	Dev	Test
CD-Seq2Seq	17.1	18.3	**6.7**	**6.4**
SyntaxSQL-con	**18.5**	**20.2**	4.3	5.2
SyntaxSQL-sta	15.2	16.9	0.7	1.1

Table 6: Performance of various methods over all questions (*question match*) and all interactions (*interaction match*).

We note that SyntaxSQL-con scores higher in *question match* but lower in *interaction match* compared to CD-Seq2Seq. A closer examination shows that SyntaxSQL-con predicts more questions correctly in the early turns of an interaction (Table 7), which results in its overall higher *question match* accuracy. A possible reason for this is that SyntaxSQL-con adopts a stronger context-agnostic text-to-SQL module (SyntaxSQLNet vs. Seq2Seq adopted by CD-Seq2Seq). The higher performance of CD-Seq2Seq on *interaction match* can be attributed to better incorporation of information flow between questions by using turn-level encoders (Suhr et al., 2018), which is possible to encode the history of all previous questions comparing to only single one previous question in SyntaxSQL-con. Overall, the lower performance of the two extended context-dependent models shows the difficulty of SParC and that there is ample room for improvement.

Turn #	CD-Seq2Seq	SyntaxSQL-con
1 (422)	31.4	38.6
2 (422)	12.1	11.6
3 (270)	7.8	3.7
$\geq$ 4 (89)	2.2	1.1

Table 7: Performance stratified by question turns on the development set. The performance of the two models decrease as the interaction continues.

Performance stratified by question position To gain more insight into how question position affects the performance of the two models, we

[11] Details of the evaluation metrics can be found at https://github.com/taoyds/spider/tree/master/evaluation_examples

report their performances on questions in different positions in Table 7. Questions in later turns of an interaction in general have greater dependency over previous questions and also greater risk for error propagation. The results show that both CD-Seq2Seq and SyntaxSQL-con consistently perform worse as the question turn increases, suggesting that both models struggle to deal with information flow from previous questions and accumulate errors from previous predictions. Moreover, SyntaxSQLNet significantly outperforms CD-Seq2Seq on questions in the first turn, but the advantage disappears in later turns (starting from the second turn), which is expected because the context encoding mechanism of SyntaxSQL-con is less powerful than the turn-level encoders adopted by CD-Seq2Seq.

Goal Difficulty	CD-Seq2Seq	SyntaxSQL-con
Easy (483)	35.1	**38.9**
Medium (441)	7.0	**7.3**
Hard (145)	**2.8**	1.4
Extra hard (134)	**0.8**	0.7

Table 8: Performance stratified by question difficulty on the development set. The performances of the two models decrease as questions are more difficult.

Performance stratified by SQL difficulty We group individual questions in SParC into different difficulty levels based on the complexity of their corresponding SQL representations using the criteria proposed in Yu et al. (2018c). As shown in Figure 3, the questions turned to get harder as interaction proceeds, more questions with hard and extra hard difficulties appear in late turns. Table 8 shows the performance of the two models across each difficulty level. As we expect, the models perform better when the user request is easy. Both models fail on most hard and extra hard questions. Considering that the size and question types of SParC are very close to Spider, the relatively lower performances of SyntaxSQLNet on medium, hard and extra hard questions in Table 8 comparing to its performances on Spider (17.6%, 16.3%, and 4.9% respectively) indicates that SParC introduces additional challenge by introducing context dependencies, which is absent from Spider.

Performance stratified by thematic relation Finally, we report the model performances across thematic relations computed over the 102 examples summarized in Table 2. The results (Table 9) show that the models, in particular SyntaxSQL-con, perform the best on the ***answer refinement/theme*** relation. A possible reason for this is that questions in the ***answer theme*** category can often be interpreted without reference to previous questions since the user tends to state the theme entity explicitly. Consider the example in the bottom row of Table 2. The user explicitly said “Statistics department” in their question, which belongs to the answer set of the previous question [12]. The overall low performance for all thematic relations (***refinement*** and ***theme-property*** in particular) indicates that the two models still struggle on properly interpreting the question history.

Thematic relation	CD-Seq2Seq	SyntaxSQL-con
Refinement	8.4	6.5
Theme-entity	13.5	10.2
Theme-property	9.0	7.8
answer refine./them.	12.3	20.4

Table 9: Performance stratified by thematic relations. The models perform best on the *answer refinement/theme* relation, but do poorly on the *refinement* and *theme-property* relations.

7 Conclusion

In this paper, we introduced SParC, a large-scale dataset of context-dependent questions over a number of databases in different domains annotated with the corresponding SQL representation. The dataset features wide semantic coverage and a diverse set of contextual dependencies between questions. It also introduces unique challenge in mapping context-dependent questions to SQL queries in unseen domains. We experimented with two competitive context-dependent semantic parsing approaches on SParC. The model accuracy is far from satisfactory and stratifying the performance by question position shows that both models degenerate in later turns of interaction, suggesting the importance of better context modeling. The dataset, baseline implementations and leaderboard are publicly available at `https://yale-lily.github.io/sparc`.

Acknowledgements

We thank Tianze Shi for his valuable discussions and feedback. We thank the anonymous reviewers for their thoughtful detailed comments.

[12] As pointed out by one of the anonymous reviewers, there are less than 10 examples of *answer refinement/theme* relation in the 102 analyzed examples. We need to see more examples before concluding that this phenomenon is general.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association forComputational Linguistics*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland. Association for Computational Linguistics.

Núria Bertomeu, Hans Uszkoreit, Anette Frank, Hans-Ulrich Krieger, and Brigitte Jörg. 2006. Contextual phenomena and thematic relations in database qa dialogues: results from a wizard-of-oz experiment. In *Proceedings of the Interactive Question Answering Workshop at HLT-NAACL 2006*, pages 1–8. Association for Computational Linguistics.

Joyce Y. Chai and Rong Jin. 2004. Discourse structure for context question answering. In *HLT-NAACL 2004 Workshop on Pragmatics in Question Answering*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184. Association for Computational Linguistics.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: The atis-3 corpus. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, Stroudsburg, PA, USA. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742. Association for Computational Linguistics.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan Dhanalakshmi Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *ACL 2018*. Association for Computational Linguistics.

Stefan L. Frank. 2013. Uncertainty reduction as a measure of cognitive load in sentence comprehension. *Topics in cognitive science*, 5 3:475–94.

John Hale. 2006. Uncertainty about the rest of the sentence. *Cognitive science*, 30 4:643–72.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014. The second dialog state tracking challenge. In *SIGDIAL Conference*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. *CoRR*, abs/1704.08760.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831. Association for Computational Linguistics.

Tsuneaki Kato, Jun'ichi Fukumoto, Fumito Masui, and Noriko Kando. 2004. Handling information access dialogue through qa technologies-a novel challenge for open-domain question answering. In *Proceedings of the Workshop on Pragmatics of Question Answering at HLT-NAACL 2004*.

Roger P. Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106:1126–1177.

Fei Li and HV Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *VLDB*.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1456–1465. Association for Computational Linguistics.

Scott Miller, David Stallard, Robert J. Bobrow, and Richard M. Schwartz. 1996. A fully statistical approach to natural language interfaces. In *ACL*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788. Association for Computational Linguistics.

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 of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31,* *2015, Beijing, China, Volume 1: Long Papers*, pages 1470–1480.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543. ACL.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157. ACM.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2018. Coqa: A conversational question answering challenge. *CoRR*, abs/1808.07042.

Tianze Shi, Kedar Tatwawadi, Kaushik Chakrabarti, Yi Mao, Oleksandr Polozov, and Weizhu Chen. 2018. Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *arXiv preprint arXiv:1809.05054*.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2238–2249. Association for Computational Linguistics.

Chenglong Wang, Po-Sen Huang, Alex Polozov, Marc Brockschmidt, and Rishabh Singh. 2018. Execution-guided neural program decoding. In *ICML workshop on Neural Abstract Machines and Program Induction v2 (NAMPI)*.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of NAACL*. Association for Computational Linguistics.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018b. Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. In *Proceedings of EMNLP*. Association for Computational Linguistics.

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. 2018c. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *EMNLP*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, pages 1050–1055, Portland, OR. AAAI Press/MIT Press.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *UAI*.

Luke S. Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *ACL/IJCNLP*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *ACL*.

A Appendices

A.1 Additional Baseline Model Details

CD-Seq2Seq We use a bi-LSTM, LSTM^E, to encode the user utterance at each turn. At each step j of the utterance, LSTM^E takes as input the word embedding and the discourse state $\mathbf{h}^I_{i-1}$ updated for the previous turn $i-1$:

$$\mathbf{h}^E_{i,j} = \text{LSTM}^E([\mathbf{x}_{i,j}; \mathbf{h}^I_{i-1}], \mathbf{h}^E_{i,j-1})$$

where i is the index of the turn and j is the index of the utterance token. The final hidden state LSTM^E is used as the input of a uni-directional LSTM, LSTM^I, which is the interaction level encoder:

$$\mathbf{h}^I_i = \text{LSTM}^I(\mathbf{h}^E_{|x_i|}, \mathbf{h}^I_{i-1}).$$

For each column header, we concatenate its table name and its column name separated by a special dot token (i.e., `table_name.column_name`), and the column header embedding $\mathbf{h}^C$ is the average embeddings of the words.

The decoder is implemented as another LSTM with hidden state $\mathbf{h}^D$. We use the dot-product based attention mechanism to compute the context vector. At each decoding step k, we compute attention scores for all tokens in η previous turns (we use $\eta = 5$) and normalize them using softmax. Suppose the current turn is i, and consider the turns of $0, \ldots, \eta - 1$ distance from turn i. We use a learned position embedding $\phi^I(i-t)$ when computing the attention scores. The context vector is the weighted sum of the concatenation of the token embedding and the position embedding:

$$\begin{aligned} s_k(t,j) &= [\mathbf{h}^E_{t,j}; \phi^I(i-t)]\mathbf{W}_{\text{att}}\mathbf{h}^D_k \\ \alpha_k &= \textbf{softmax}(s_k) \\ \mathbf{c}_k &= \sum_{t=i-h}^{i} \sum_{j=1}^{|x_t|} \alpha_k(t,j)[\mathbf{h}^E_{t,j}; \phi^I(i-t)] \end{aligned}$$

At each decoding step, the sequential decoder chooses to generate a SQL keyword (e.g., `select`, `where`, `group by`, `order by`) or a column header. To achieve this, we use separate layers to score SQL keywords and column headers, and finally use the softmax operation to generate the output probability distribution:

$$\begin{aligned} \mathbf{o}_k &= \tanh([\mathbf{h}^D_k; \mathbf{c}_k]\mathbf{W}_o) \\ \mathbf{m}^{\text{SQL}} &= \mathbf{o}_k\mathbf{W}_{\text{SQL}} + \mathbf{b}_{\text{SQL}} \\ \mathbf{m}^{\text{column}} &= \mathbf{o}_k\mathbf{W}_{\text{column}}\mathbf{h}^C \\ P(y_k) &= \textbf{softmax}([\mathbf{m}^{\text{SQL}}; \mathbf{m}^{\text{column}}]) \end{aligned}$$

It's worth mentioning that we experimented with a SQL segment copying model similar to the one proposed in Suhr et al. (2018). We implement our own segment extraction procedure by extracting `SELECT`, `FROM`, `GROUP BY`, `ORDER BY` clauses as well as different conditions in `WHERE` clauses. In this way, we can extract 3.9 segments per SQL on average. However, we found that adding segment copying does not significantly improve the performance because of error propagation. Better leveraging previously generated SQL queries remains an interesting future direction for this task.

SyntaxSQL-con As in (Yu et al., 2018b), the following is defined to compute the conditional embedding $\mathbf{H}_{1/2}$ of an embedding $\mathbf{H}_1$ given another embedding $\mathbf{H}_2$:

$$\mathbf{H}_{1/2} = \textbf{softmax}(\mathbf{H}_1\mathbf{W}\mathbf{H}_2^\top)\mathbf{H}_1.$$

Here $\mathbf{W}$ is a trainable parameter. In addition, a probability distribution from a given score matrix $\mathbf{U}$ is computed by

$$\mathcal{P}(\mathbf{U}) = \textbf{softmax}\left(\mathbf{V}\textbf{tanh}(\mathbf{U})\right),$$

where $\mathbf{V}$ is a trainable parameter. To incorporate the context history, we encode the question right before the current question and add it to each module as an input. For example, the COL module of SyntaxSQLNet is extended as following. $\mathbf{H}_{\text{PQ}}$ denotes the hidden states of LSTM on embeddings of the previous one question and the $\mathbf{W}_3^{\text{num}}\mathbf{H}_{\text{PQ/COL}}^{\text{num}}{}^\top$ and $\mathbf{W}_4^{\text{val}}\mathbf{H}_{\text{PQ/COL}}^{\text{val}}{}^\top$ terms add history information to prediction of the column number and column value respectively.

$$P_{\text{COL}}^{\text{num}} = \mathcal{P}\left(\mathbf{W}_1^{\text{num}}\mathbf{H}_{\text{Q/COL}}^{\text{num}}{}^\top + \mathbf{W}_2^{\text{num}}\mathbf{H}_{\text{HS/COL}}^{\text{num}}{}^\top + \mathbf{W}_3^{\text{num}}\mathbf{H}_{\text{PQ/COL}}^{\text{num}}{}^\top\right)$$

$$P_{\text{COL}}^{\text{val}} = \mathcal{P}\left(\mathbf{W}_1^{\text{val}}\mathbf{H}_{\text{Q/COL}}^{\text{val}}{}^\top + \mathbf{W}_2^{\text{val}}\mathbf{H}_{\text{HS/COL}}^{\text{val}}{}^\top + \mathbf{W}_3^{\text{val}}\mathbf{H}_{\text{COL}}{}^\top + \mathbf{W}_4^{\text{val}}\mathbf{H}_{\text{PQ/COL}}^{\text{val}}{}^\top\right)$$

A.2 Additional Data Examples

We provide additional SParC examples in Figure 4 and examples with different thematic relations in Figure 5.

D_3 : Database about wine.

C_4 : **Find the county where produces the most number of wines with score higher than 90.**

Q_1 : How many different counties are all wine appellations from?

S_1 : `SELECT COUNT(DISTINCT county) FROM appellations`

Q_2 : How many wines does each county produce?

S_2 : `SELECT T1.county, COUNT(*) FROM appellations AS T1 JOIN wine AS T2 ON T1.appellation = T2.appellation GROUP BY T1.county`

Q_3 : Only show the counts of wines that score higher than 90?

S_3 : `SELECT T1.county, COUNT(*) FROM appellations AS T1 JOIN wine AS T2 ON T1.appellation = T2.appellation WHERE T2.score > 90 GROUP BY T1.county`

Q_4 : Which county produced the greatest number of these wines?

S_4 : `SELECT T1.county FROM appellations AS T1 JOIN wine AS T2 ON T1.appellation = T2.appellation WHERE T2.score > 90 GROUP BY T1.county ORDER BY COUNT(*) DESC LIMIT 1`

D_5 : Database about districts

C_5 : **Find the names and populations of the districts whose area is greater than the average area.**

Q_1 : What is the total district area?

S_1 : `SELECT sum(area_km) FROM district`

Q_2 : Show the names and populations of all the districts.

S_2 : `SELECT name, population FROM district`

Q_3 : Excluding those whose area is smaller than or equals to the average area.

S_3 : `SELECT name, population FROM district WHERE area_km > (SELECT avg(area_km) FROM district)`

D_6 : Database about books

C_6 : **Find the title, author name, and publisher name for the top 3 best sales books.**

Q_1 : Find the titles of the top 3 highest sales books.

S_1 : `SELECT title FROM book ORDER BY sale_amount DESC LIMIT 3`

Q_2 : Who are their authors?

S_2 : `SELECT t1.name FROM author AS t1 JOIN book AS t2 ON t1.author_id = t2.author_id ORDER BY t2.sale_amount DESC LIMIT 3`

Q_3 : Also show the names of their publishers.

S_3 : `SELECT t1.name, t3.name FROM author AS t1 JOIN book AS t2 ON t1.author_id = t2.author_id JOIN press AS t3 ON t2.press_id = t3.press_id ORDER BY t2.sale_amount DESC LIMIT 3`

Figure 4: More examples in SParC.

D_7 : Database about school departments

C_7 : **What are the names and budgets of the departments with average instructor salary greater than the overall average?**

Q_1 : Please list all different department names.

S_1 : `SELECT DISTINCT dept_name FROM department`

Q_2 : Show me the budget of the Statistics department. ***(Theme/refinement-answer)***

S_2 : `SELECT budget FROM department WHERE dept_name = "Statistics"`

Q_3 : What is the average salary of instructors in that department? ***(Theme-entity)***

S_3 : `SELECT AVG(T1.salary)FROM instructor as T1 JOIN department as T2 ON T1.department_id = T2.id WHERE T2.dept_name = "Statistics"`

Q_4 : How about for all the instructors? ***(Refinement)***

S_4 : `SELECT AVG(salary) FROM instructor`

Q_5 : Could you please find the names of the departments with average instructor salary less than that? ***(Theme/refinement-answer)***

S_5 : `SELECT T2.dept_name FROM instructor as T1 JOIN department as T2 ON T1.department_id = T2.id GROUP BY T1.department_id HAVING AVG(T1.salary) < (SELECT AVG(salary) FROM instructor)`

Q_6 : Ok, how about those above the overall average? ***(Refinement)***

S_6 : `SELECT T2.dept_name FROM instructor as T1 JOIN department as T2 ON T1.department_id = T2.id GROUP BY T1.department_id HAVING AVG(T1.salary) > (SELECT AVG(salary) FROM instructor)`

Q_7 : Please show their budgets as well. ***(Theme-entity)***

S_7 : `SELECT T2.dept_name, T2.budget FROM instructor as T1 JOIN department as T2 ON T1.department_id = T2.id GROUP BY T1.department_id HAVING AVG(T1.salary) > (SELECT AVG(salary) FROM instructor)`

Figure 5: Additional example in SParC annotated with different thematic relations. Entities (purple), properties (magenta), constraints (red), and answers (orange) are colored.