Exploring the Feasibility of Crowd-Powered Decomposition of Complex User Questions in Text-to-SQL Tasks

Salimzadeh, Sara; Gadiraju, Ujwal; Hauff, Claudia; Van Deursen, Arie

DOI
10.1145/3511095.3531282

Publication date
2022

Document Version
Final published version

Published in
HT 2022

Exploring the Feasibility of Crowd-Powered Decomposition of Complex User Questions in Text-to-SQL Tasks

Sara Salimzadeh
Delft University of Technology
Delft, The Netherlands
s.salimzadeh@tudelft.nl

Ujwal Gadiraju
Delft University of Technology
Delft, The Netherlands
u.k.gadiraju@tudelft.nl

Claudia Hauff
Delft University of Technology
Delft, The Netherlands
c.hauff@tudelft.nl

Arie van Deursen
Delft University of Technology
Delft, The Netherlands
arie.vandeursen@tudelft.nl

ABSTRACT

Natural Language Interfaces to Databases (NLIDB), also known as `Text-to-SQL` models, enable users with different levels of knowledge in Structured Query Language (SQL) to access relational databases without any programming effort. By translating natural languages into SQL query, not only do NLIDBs minimize the burden of memorizing the schema of databases and writing complex SQL queries, but they also allow non-experts to acquire information from databases in natural languages. However, existing NLIDBs largely fail to translate natural languages to SQL when they are complex, preventing them from being deployed in real-world scenarios and generalizing across unseen complex databases. In this paper, we explored the feasibility of decomposing complex user questions into multiple sub-questions — each with a reduced complexity — as a means to circumvent the problem of complex SQL generation. We investigated the feasibility of decomposing complex user questions in a manner that each sub-question is simple enough for existing NLIDBs to generate correct SQL queries, using non-expert crowd workers in juxtaposition with SQL experts. Through an empirical study on an NLIDB benchmark dataset, we found that crowd-powered decomposition of complex user questions led to an accuracy boost of an existing `Text-to-SQL` pipeline from 30% to 59% (96% accuracy boost). Similarly, decomposition by SQL experts resulted in boosting the accuracy to 76% (153% accuracy boost). Our findings suggest that crowd-powered decomposition can be a scalable alternative to producing the training data necessary to build machine learning models that can automatically decompose complex user questions, thereby improving `Text-to-SQL` pipelines.

CCS CONCEPTS

• **Human-centered computing** → **User studies**; **Natural language interfaces**; • **Information systems** → **Structured Query Language**.

KEYWORDS

Text-to-SQL, Semantic Parsing, Natural Language Interface to Databases, Crowdsourcing, Human Computation, Corpus Annotation

ACM Reference Format:
Sara Salimzadeh, Ujwal Gadiraju, Claudia Hauff, and Arie van Deursen. 2022. Exploring the Feasibility of Crowd-Powered Decomposition of Complex User Questions in Text-to-SQL Tasks. In *Proceedings of the 33rd ACM Conference on Hypertext and Social Media (HT '22), June 28-July 1, 2022, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3511095.3531282

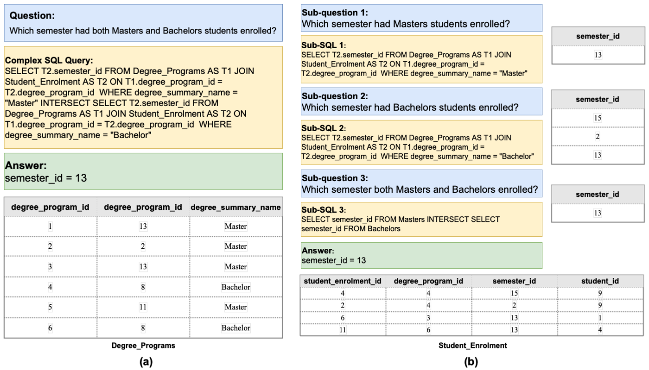
1 INTRODUCTION

Building Natural Language Interfaces to Databases (NLIDBs) has been identified as one of the most significant semantic parsing tasks for decades [3, 9, 18, 68, 73, 98]. By automatically converting text into the Structured Query Language (SQL), NLIDBs allow users to communicate with relational data in natural languages (NL) without any programming effort. These NL questions often cannot be directly answered by search engines. For example, in response to the question '*What are the total population and average area of countries in the continent of North America whose area is bigger than 3000?*', an NLIDB would return 480753000 and 1344763 for the total population and average area respectively; while a search engine would present a number of tables and leave the computation to the user. Such interfaces (also known as `Text-to-SQL` models within the NLP community) relieve users who are not proficient in query languages from the burden of learning techniques for querying databases by allowing them to pose NL questions.

Within recent years, the emergence of complex, large, and human-annotated datasets consisting of NL questions and their corresponding SQL queries has significantly developed the field. Traditionally these have included in-domain datasets such as WikiSQL [98], ATIS [15, 39], and Advising [24], more recently the family of `Spider` cross-domain datasets, including `Spider` [92], SParC [93], and CoSQL [91] challenge the generalizability of models to unseen databases. Although recent studies have demonstrated

This research has been supported by *ICAI AI for Fintech Research.*

HT '22, June 28-July 1, 2022, Barcelona, Spain
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9233-4/22/06.
https://doi.org/10.1145/3511095.3531282



Question:
Which semester had both Masters and Bachelors students enrolled?

Complex SQL Query:

```
SELECT T2.semester_id FROM Degree_Programs AS T1 JOIN
Student_Enrolment AS T2 ON T1.degree_program_id =
T2.degree_program_id  WHERE degree_summary_name =
"Master" INTERSECT SELECT T2.semester_id FROM
Degree_Programs AS T1 JOIN Student_Enrolment AS T2 ON
T1.degree_program_id = T2.degree_program_id  WHERE
degree_summary_name = "Bachelor"
```

Answer:
semester_id = 13

degree_program_id	degree_program_id	degree_summary_name
1	13	Master
2	2	Master
3	13	Master
4	8	Bachelor
5	11	Master
6	8	Bachelor

Degree_Programs

(a)

Sub-question 1:
Which semester had Masters students enrolled?

Sub-SQL 1:

```
SELECT T2.semester_id FROM Degree_Programs AS T1 JOIN
Student_Enrolment AS T2 ON T1.degree_program_id =
T2.degree_program_id  WHERE degree_summary_name = "Master"
```

semester_id
13

Sub-question 2:
Which semester had Bachelors students enrolled?

Sub-SQL 2:

```
SELECT T2.semester_id FROM Degree_Programs AS T1 JOIN
Student_Enrolment AS T2 ON T1.degree_program_id =
T2.degree_program_id  WHERE degree_summary_name = "Bachelor"
```

semester_id
15
2
13

Sub-question 3:
Which semester both Masters and Bachelors enrolled?

Sub-SQL 3:

```
SELECT semester_id FROM Masters INTERSECT SELECT
semester_id FROM Bachelors
```

semester_id
13

Answer:
semester_id = 13

student_enrolment_id	degree_program_id	semester_id	student_id
4	4	15	9
2	4	2	9
6	3	13	1
11	6	13	4

Student_Enrolment

(b)

Figure 1: (a) An example of a complex question in the `Spider` dataset. In addition to the complex question, the corresponding SQL query, the answer, and tables are shown. (b) The decomposition of the question in (a) is illustrated; Instead of feeding the complex question in (a) into `Text-to-SQL` models, we manually decompose the question into the three sub-questions. These sub-questions are classified as simpler than the original question. Executing sub-questions sequentially on the database, we can observe that answer to the complex question is the same as sub-question 3 in (a).

the high accuracy (above 70%) of state-of-the-art `Text-to-SQL` models trained and evaluated on the `Spider` dataset, the performance of these models on complex queries is rather low, as many struggle to predict complex SQL queries, **Complex SQL Generation**. Parsing a question into a SQL query with nested queries, multiple SELECT clauses, GROUP BY, ORDER BY, UNION, INTERSECT, and EXCEPT requires a model to capture the semantic dependency between the NL question, database schema, and SQL syntax. According to the `Spider` criteria, SQL queries are classified into four difficulty levels – *easy*, *medium*, *hard*, and *extra hard*. The difficulty level is determined based on the number of SQL components, selections, conditions, nested sub-queries, column selections, aggregators, etc. Further, a question is complex when the corresponding SQL query is hard or extra hard.

Evaluating the accuracy of the top-five state-of-the-art `Text-to-SQL` models only on complex questions within the development set of `Spider` as the preliminary step, we found that their performance is below 50%. On questions with corresponding SQL queries of easy and medium difficulty levels, however, such models perform with an accuracy of over 80%. Therefore, we explore to what extent the **decomposition of complex questions**, as a **novel stage** within the `Text-to-SQL` pipeline, can bring us further in the area of `Text-to-SQL`. This is guided by our intuition that by decomposing complex questions into multiple easy and medium questions, `Text-to-SQL` models can convert them into correct SQL queries with a higher accuracy, thereby circumventing the challenge of complex SQL generation, illustrated in Figure 1.

Note that the proposed decomposition stage is different from standard text simplification in NLP [59], a task in which text is rewritten to make it easier to process for a given audience. The complexity of questions in the `Spider` dataset originates from the underlying SQL query and the dependency between the text and database schema as opposed to the linguistic complexity of NL questions. To verify this, we analyzed whether metrics that are popularly used in text simplification tasks such as *Flesch-Kincaid readability score*, *Flesch’s reading ease score*, *Type-Token Ration*, and *Lexical variation* are effective in distinguishing levels of difficulty in complex user questions. We found that easy and medium questions have the same lexical complexity and lexical richness as hard and extra hard questions, confirming that the existing text simplification methods are ill-suited for decomposing complex user questions. In order to assess the feasibility of decomposition, we thereby raise the following research questions:

RQ1 To what extent can we leverage the decomposition of complex user questions as a means to circumvent the challenge of complex SQL generation facing existing `Text-to-SQL` pipelines?

RQ2 To what extent can non-expert crowd workers aid in the decomposition of complex user questions in `Text-to-SQL` tasks in comparison to SQL experts?

To assess the potential benefit of decomposing complex questions, we first manually decomposed the questions and corresponding queries within the development set of the `Spider` dataset serving as an **oracle decomposition**. We then compared the accuracy gained by `Text-to-SQL` models with the new pipeline in which the oracle decomposition was augmented, realizing an increase in accuracy by over 163% (i.e., from 30% to 79%). Despite the promise of decomposition, to develop ML models that can (semi) automate the decomposition of complex user questions in a generalizable fashion, we would require a substantial amount of training data. Since hiring groups of experts is a costly endeavour [60], the viability of decomposing complex user questions at a beneficial scale hinges on its cost-effectiveness. Crowdsourcing has proved to be a reliable, effective, and efficient approach in many tasks [31, 53, 62] and across different domains [63], including within the NLP field [36, 80, 97]. Thus, we explored whether non-expert crowd workers (recruited from the Prolific crowdsourcing platform) can power such a cost-effective alternative. In comparison to the accuracy boost of 153% as a result of the decomposition carried out by a small group of SQL experts ($N = 5$), decomposition by non-expert crowd workers ($N = 83$) led to an accuracy boost of over 96%. Our findings show that crowd workers can effectively decompose complex user questions and thereby aid in circumventing the challenge of complex SQL generation in `Text-to-SQL` pipelines.

Our experiments pave the way towards extending crowd-powered decomposition on available `Text-to-SQL` datasets to gather a substantial amount of training data. This is a crucial prerequisite for building ML-based automatic decomposition models integrated into the existing `Text-to-SQL` pipeline to circumvent the challenge of complex SQL generation.

The remainder of the paper is structured as follows. In Section 2, we describe background and related work. We then explain the gold standard created to evaluate crowd-workers performance in Section 3 followed by illustrating the steps in crowd-powered decomposition, Section 4. We answer the research questions in Section 5. We discuss the implication, outlook, and threats to validity in Section 6. Finally, in Section 7, we pinpoint conclusion.

2 BACKGROUND AND RELATED WORK

Context Independent Text-to-SQL Parsing: Generating SQL queries from natural language questions has been an active field of study for a long period in both database and NLP communities [1, 33, 46, 48, 57, 81, 84, 94, 95]. Previous `Text-to-SQL` parsers employed either expert-designed rules [72, 75, 85] or statistical techniques [42, 74, 94]. Over the past few years, driven by the development of a large in-domain context-independent dataset, WikiSQL [98], many deep learning models proposed by researchers have shown promising results for this task [30, 32, 69, 98]. All of these studies focus on mapping a single query to the corresponding SQL query which is known as context-independent parsing. Deep learning models generally adapt an encoder-decoder framework to solve the Text-to-SQL problem as a sequence-to-sequence problem [18, 20, 41, 69, 98]. To show and test the limitations of the Text-to-SQL models on generalizability on various domains and databases, Yu et al. [92] proposed a complex cross-domain dataset called Spider. In addition to the sequence-to-sequence paradigm, namely the generation-based methods, state-of-the-art neural models leverage more strategies such as sketched-based techniques (generates a SQL skeleton first and then fills the skeleton with database schema tokens) [13, 19, 32, 38, 51, 52, 86, 89, 98], data augmentation [78, 84], various attentional architectures for question/schema encoding such intermediate representation for decoding [27, 29, 34, 90], graph representation of databases in schema encoding [5, 6, 10, 12], schema linking (correctly identify column and value mentions in an natural language questions and link them to the given database schema) [6, 7, 17, 18, 21, 29, 45, 49, 50, 65, 77, 89].

While there are some attempts to tackle the complex SQL generation issue, it is still a significant challenge for Text-to-SQL models [24, 45, 92]. For instance, schema linking methods by capturing the alignment between text and table indirectly address this challenge. On the other hand, intermediate representation approaches are designed to bridge the gap between text and SQL. Furthermore, some studies have examined decomposing complex SQL queries within the decoder to generate multiple clauses and sub-queries [43, 90]. Unlike these studies, in this work, we investigate the potential performance gain by adding a decomposition stage in the Text-to-SQL pipeline to decompose complex natural language questions before submitting them to Text-to-SQL models.

Context Dependent Text-to-SQL Parsing: Recently, context-dependent Text-to-SQL parsing has drawn a lot of attention. Compared to benchmarks with single-turn questions, ATIS, a simple in-domain context-dependent benchmark, was proposed first. The models evaluated on ATIS leveraged the sequence-to-sequence framework[8]. Later, to overcome the lack of generalizability of models, two large-scale context-dependent datasets were introduced for the Text-to-SQL task, SParC [93] and CoSQL [91] modelling conversational dependencies between questions. The Text-to-SQL models, also known as conversational Text-to-SQL models, require understanding the context of sequentially related questions compared to single-turn models. Several studies were conducted on these two benchmarks that proposed EditSQL [96], IGSQL [8], IST-SQL [79], R^2SQL [37], RAT-SQL-TC [47] models. In addition to employing strategies in the previous section to tackle the problem of translating Text to SQL, these models track dialogue states to generate SQL queries according to the context. Li et al. [47] conducted an exploratory study within context-dependent parsing to determine how far we are from effective context modelling. In this work, we employed R^2SQL as the baseline to assess the accuracy gain of decomposing complex questions in the Text-to-SQL pipeline. It was the first open-source context-dependent model in the SParC leaderboard [1] at the time of carrying out the experiments in this paper.

Text-to-SQL Datasets: The growing interest in Text-to-SQL applications has led to various datasets including in-domain datasets ATIS [15, 39], GeoQuery [57, 94], Restaurants [57, 71], Scholar [39], Advising [24], Academic [46], Yelp [87], IMDB [87] which have been studied for decades. WikiSQL is among the first large-scale datasets with relatively simple questions and single tables extracted from Wikipedia. Although WikiSQL contains 80654 questions and SQL pairs for 24241 databases, it is generated from a limited set of templates and only covers the single SELECT column, aggregation, and WHERE clause. Furthermore, keywords like JOIN, GROUP BY, and ORDER BY are not included. The family of Spider datasets, Spider [92], SparC [93], and CoSQL [91] contain the most difficult questions having nested queries, covering many SQL syntaxes, and multiple table joins. These datasets evaluate the Text-to-SQL models to generalize not only new SQL queries and database schemas but also new domains. Spider contains 10,181 questions and 5,693 unique complex SQL queries on 200 databases with multiple tables, covering 138 domains. It also supports a wide range of SQL syntax. Due to these reasons, we examined our proposed solution on the development set of the Spider dataset.

Data Annotation & Crowdsourcing: Natural Language Processing research has been spurred on by the growing number of annotated corpora [36, 80, 97]. Such corpora are leveraged to train, evaluate, and compare NLP algorithms. However, annotating data is an expensive and time-consuming process [60]. The emergence of crowdsourcing [23] platforms such as MTurk [2] has led to a widespread adoption of crowd-powered workflows to create annotated corpora [4, 11, 14, 28, 44, 53–56, 61, 66]. Crowdsourcing has been shown to be a cheaper and faster alternative compared to expert annotation [25, 76]. In addition to data labelling, crowdsourcing proved to be a reliable approach in many tasks [31, 53, 62] and domains [63]. Although much research is conducted to quality control and quality assurance in crowdsourcing [16, 26, 35], several studies have also shown the benefit of employing experts to provide higher quality labels [2, 83]. Prior works have proposed augmenting crowd worker labels with those from experts to optimize the cost and quality of data labelling [40, 58, 64, 88]. We employ both domain experts and crowd workers for data annotation in this work. Our findings suggest the potential benefit of leveraging crowd workers to create training data and then build ML-based decomposition model in the future.

3 GOLD STANDARD FOR DECOMPOSITION OF COMPLEX QUESTIONS

This section introduces the steps for developing a gold standard for the decomposition, creating SpiderDec serving as the **oracle decomposition**. We then employ a Text-to-SQL model to assess the potential accuracy boost by decomposition. Note that the accuracy is measured based on comparing the execution result of each SQL query with the corresponding gold query.

SpiderDec, Extension of the Spider Dataset: In the Spider dataset, data is split into training, development, and a hidden test set.

[1] https://yale-lily.github.io/sparc
[2] https://www.mturk.com

We manually decomposed the questions, and corresponding queries within the development set of the Spider dataset on questions with hard and extra hard SQL queries, thereby creating SpiderDec[3]. State-of-the-art Text-to-SQL models have over 80% execution accuracy for SQL queries with easy and medium hardness levels, while the performance is less than 50% for hard and extra hard SQL queries. So, decomposing hard and extra hard questions into multiple easy and medium questions can lead to a higher accuracy of Text-to-SQL pipeline. For simplicity, we refer to the hard and extra hard questions of Spider dataset as complex questions. We limited our approach to the development set first to explore the potential benefit of the decomposition task; we leave the annotation of the training set as future work in case of accuracy boost in the Text-to-SQL pipeline. Our rationale behind considering the Spider dataset as a lens to circumvent the problem of complex user questions is governed by the scale and diversity of the Spider dataset compared to others. Furthermore, the cross-domain setup of Spider allows Text-to-SQL models to use different databases for training and testing.

Within the Spider dataset, in total, there are 332 complex training examples over 20 databases. Each example consists of a natural language question and its corresponding SQL gold query. In the remainder of the paper, we refer to each instance in the dataset as a pair of the NL question and the SQL query. We annotated Spider development set in two stages: sub-SQL annotation and sub-question annotation. As the complexity of the Text-to-SQL task derives from the underlying SQL queries, we created SpiderDec from a SQL-centered perspective, first annotating SQL queries and then questions.

Sub-SQL Annotation: In the SpiderDec decomposition, we first broke down each complex SQL query into multiple subsequent easy or medium SQL sub-queries. Based on our rubric inspired by prior work [92], each sub-SQL meets one of the conditions in Table 1 to be considered as easy or medium. Among 332 pairs of the NL question and the SQL query, 26.8% of SQL queries contain these keywords: *EXCEPT*, *UNION*, and *NOT IN* on which decomposition to medium or easy is not applicable. Therefore, we only decomposed their nested sub-queries into simpler ones and kept the keyword without the necessity of having all the SQL sub-queries with the easy or medium level of difficulty.

Sub-Question Annotation: Given decomposed SQL sub-queries per SQL query from the previous stage, we assigned a natural language sub-question to each of the annotated sub-queries. In order to determine whether sub-questions are semantically equivalent to their associated complex questions, two experts manually evaluated them and resolved any conflicts with each other.

Assessing Accuracy of SpiderDec: To investigate the potential accuracy boost achievable by adding the decomposition stage to the Text-to-SQL pipeline, we are required to measure the performance of Text-to-SQL models on the newly generated dataset. Instead of complex questions, we gave decomposed sub-questions to pre-trained models as input data. We then calculated the execution accuracy gained on the entire development set and separately per hardness category. To this end, we leveraged R^2SQL [37], a context-dependent BERT-based Text-to-SQL model trained on SParC [93] dataset. We then assessed the execution results of predicted sub-SQLs by R^2SQL and compared them with the result obtained from the original development set of Spider (existing Text-to-SQL pipeline). R^2SQL can effectively model contextual questions and database schemas. SParC dataset is built on top of the Spider, providing rich contextual phenomena and thematic relations between the questions. Because the sub-questions are thematically dependent on each other acting as contextual utterances, we adapted the context-dependent Text-to-SQL model, which maps the entire sub-questions to the corresponding SQL queries. Furthermore, the R^2SQL is the first open-source model on the leaderboard at the time of experimenting [4].

4 CROWD-POWERED DECOMPOSITION

We now describe our crowd-powered study in more detail. We go over the annotation tool, the task, participants, the workflow. We then explain our measurement to evaluate participants' decomposition.

Annotation Tool. We developed an annotation tool on top of the R^2SQL pre-trained model for crowd workers to decompose complex questions. We first created a Text-to-SQL API from R^2SQL, translating contextual natural language questions into SQL queries. We leveraged Vue.js JavaScript framework [5] for the frontend and Flask [6] for the backend. Within the annotation process, the question to be decomposed and its associated database are presented to participants. They can easily interact with tables, search an item, sort rows, and scroll them. They can also execute the predicted SQL corresponding to their sub-questions.

Task. In creating SpiderDec, we decomposed SQL queries. We then assigned NL questions to the queries (SQL-centered decomposition), while the crowd workers only access the NL questions and decompose them (question-centered decomposition). In the real-world scenarios, we do not necessarily have the gold SQL queries and labeled data, so we designated our crowd-powered study to investigate the feasibility of decomposing natural language questions and explore to what extent the question-centered decomposition result in accuracy boost compared to SpiderDec.

Participants. In our study, participants included SQL experts and non-expert crowd workers. We hired five computer science students with at least two years of experience with SQL as experts. Due to the high cost of hiring experts, we limited the number of experts to five students. Since the number of students was below sufficient samples to carry out statistical comparisons, we assigned them the entire set to gain more insight into their decomposition performance and quality. Each student spent between 12-30 hours for the whole corpus in the development set of Spider. According to the institutional regulations, the participants were paid between 22-30 € per hour based on their course credits. In addition to experts, 83 non-experts were employed through the Prolific Academic Platform.[7] With the Prolific platform, we required the participants to (i) have at least 100 accepted Prolific task submissions, (ii) to be native English speakers, and (iii) and have a minimum approval rate of 90%. The study took approximately 50 minutes to decompose

[3] https://github.com/sarasal/decomposition

[4] https://yale-lily.github.io/spider

[5] https://vuejs.org

[6] https://flask.palletsprojects.com/en/2.0.x/

[7] https://www.prolific.co

Table 1: Criteria to identify whether a SQL query is easy or medium used as a guideline for decomposition.

Easy or Medium SQL Query	
Condition 1	1) one SELECT column, 2)maximum one aggregator, 3)maximum one keyword from [WHERE, GROUP BY, ORDER BY, LIMIT, JOIN, OR, LIKE, HAVING], 3) no keywords from [EXCEPT, UNION, INTERSECT, IN, NOT IN]
Condition 2	1) maximum two conditions from [number of aggregator > 1, number of SELECT columns > 1, number of WHERE conditions > 1, number of GROUP BY clauses > 1], 2) maximum one keyword from [WHERE, GROUP BY, ORDER BY, LIMIT, JOIN, OR, LIKE, HAVING], and 3) no keywords from [EXCEPT, UNION, INTERSECT, IN, NOT IN]
Condition 3	1) maximum one condition from [number of aggregator > 1, number of SELECT columns > 1, number of WHERE conditions > 1, number of GROUP BY clauses > 1], 2) two keywords from [WHERE, GROUP BY, ORDER BY, LIMIT, JOIN, OR, LIKE, HAVING], and 3) no keywords from [EXCEPT, UNION, INTERSECT, IN, NOT IN]

six complex questions. We also paid our participants 7.5 £ (9 €) per hour for the experiment. For simplicity, we refer to Prolific participants as non-experts in the remainder of the paper.

SQL Knowledge. We measured the SQL knowledge of participants in a post-test conducted right after the decomposition task to avoid cognitive biases [22]. To this end, we manually designed our survey as no standard SQL assessment test is available in the literature. The survey took 10 minutes to complete and consists of 10 questions. First, participants were asked one question to self-report on their SQL proficiency, followed by seven questions regarding the key concepts of SQL [8]. Inspired by prior work [67, 70, 82], we employed the modified VKS test to measure participants' knowledge across four levels. Our questions are related to key concepts of SQL, which are used in our dataset, including relational databases, primary key, foreign key, SELECT statement, WHERE clause, JOIN tables, and Aggregate functions. Participants were asked to write their concept definitions for levels (3) and (4). Finally, participants were given a simple question, *Write a SQL query that returns the name of the 3 youngest winners across all matches found in the table matches.*, with a schema of the database to write down a SQL for. This question helps us to investigate their knowledge in practice.

(1) I don't remember having seen this term/phrase before.
(2) I have seen this term/phrase before, but I don't think I know what it means.
(3) I have seen this term/phrase before, and I think it means ___ .
(4) I know this term/phrase. It means ___.

English Proficiency. In addition to SQL proficiency, we hypothesized that the participants' proficiency in reading and writing could affect their performance. Decomposition a NL question first requires understanding the questions -associated with participants' reading skill- and then paraphrasing them in multiple sub-questions-connected with participants' writing skill. Therefore, we leveraged the self-assessment grid of CEFR scales, Common European Framework of Reference for Languages: Learning, Teaching, Assessment [9]. CEFR is an international standard describing reading, listening, speaking, and writing skills on a six-point scale, starting from A1 as a beginner to C2 as a master. Our task was only dependent on participants' reading and writing skills; we, therefore, included the self-assessment questions of these skills. As we included native English speakers in the study, we assumed their knowledge level is above A2 and excluded A1 and A2 from the options. In total, the participants answered three questions related to their reading and writing skill in English.

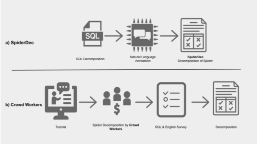


Figure 2: Overview of the flow of the user study and `SpiderDec` creation

Workflow. When participants entered the study, a 15-minute tutorial video provided information about their task and how to interact with different annotation tool components to decompose questions. All explained concepts were simplified, avoiding any technical burden for participants. Furthermore, the study was also elaborated on two examples within two stages: 1) how to decompose a complex question into multiple sub-questions and 2) how to work with the annotation tool. Subsequently, the participants moved on to the training phase, where they were given those two examples again to work with the annotation tool and learn the task in practice. Participants could stay in this stage as long as they wish to. Participants were then redirected to the actual decomposition task by clicking the respective button in the training stage— all 332 complex questions from the development set of `Spider` were randomly assigned to experts, while non-experts were given six questions. They could also skip a question if they were not certain about how to decompose it.

The experiment ended with a post-test where the SQL knowledge survey and English proficiency self-assessment were given to participants. We set these surveys as the post-test to avoid cognitive biases [22] such as *Anchoring Effect*-where the participants may overlay focus on answering the survey question rather than the actual task-, *Overconfidence or Optimism Bias*- where the participants overestimate their ability to perform the task when they can answer all the questions in the survey-, and *Loss Aversion Bias*-when the participants suspect that the answers to the questions may affect their payment. Lastly, we included five questions regarding the annotation tool and tutorial,[10] their experience working with our annotation tool, and the perceived performance. The workflow corresponding to data annotation by participants is illustrated in Figure 2.

In total, experts provided us with 1515 sub-questions. These sub-questions are associated with 623 decomposed questions, indicating that each question on average contains 2.43 sub-questions. Non-experts created 1082 sub-questions in total for 453 decomposed questions, showing on average 2.37 sub-questions per question.

Assessment of Sub-questions. We manually evaluated whether sub-questions were semantically equivalent to questions or not. Among 1515 sub-questions provided by experts, we randomly

[8] https://www.interviewbit.com/sql-interview-questions/#sql
[9] http://ebcl.eu.com/wp-content/uploads/2011/11/CEFR-all-scales-and-all-skills.pdf
[10] https://www.ueq-online.org

Table 2: Example evaluation of a participant's sub-questions

Question:	What is the country with the most number of TV channels and how many does it have?
Correct	Which country has the most number of TV channels? What is target country and how many TV channels does it have?
Partially Correct	Which country appears in the list of TV channels the most times? How many TV channels does this country have?
Incorrect	What are TV channels in the countries?

Table 3: Accuracy of the Text-to-SQL pipeline on Spider and SpiderDec reported on hard and extra hard questions.

	Dataset	Total	Hard	Extra
I	Spider	0.3	0.37	0.23
II	SpiderDec	0.79	0.82	0.76
III	Diff.	0.49	0.45	0.53

sampled 312 questions with the confidence interval of 95% from the population size while we evaluated all sub-questions generated by non-experts. As we assured the quality of data generated by experts, we randomly sampled experts' sub-questions rather than checking all of them. We labeled the entire block of sub-questions as either correct, partially correct, or incorrect. Examples of sub-questions and the labels we assigned to them are provided in Table 2. We employed the following criteria to judge the equivalency of sub-questions to the original complex question.

(2) **Correct.** If a participant's sub-questions include all concepts that appear in a question, it indicates that the question is semantically equivalent to the sub-questions. In that case, the entire block of sub-questions is assigned the highest score of **2**.

(1) **Partially Correct.** If the participant's decomposition misses one concept from the original question, a score of 1 is given to that. For example, if the question asks about the *name* and *birth date* and the participant only included *name*, we labelled the decomposition as 1.

(0) **Incorrect.** Sub-questions that are either entirely incorrect, incomplete, or missed more than one concept from the original question.

Assessment of Sub-SQLs. To examine whether participants' decomposition leads to the Text-to-SQL pipeline performance boost, we employed our baseline, R^2SQL, which is also in line with our approach in Section 3. First, the baseline predicted the sub-SQLs. We then compared the execution result of the block of sub-questions with the execution result of the gold SQL query. We labeled the sub-SQL's either correct or incorrect. The block of the sub-SQL's is correct if its execution result is equivalent to the execution result of the gold SQL; otherwise, it is incorrect.

5 RESULTS

5.1 Performance of the baseline on SpiderDec (RQ1)

In RQ1, we examine to what extent the decomposition of complex questions affects the performance of the Text-to-SQL pipeline facing the challenge of complex SQL generation. We return to Table 3 and Table 4 for insights into the performance of the R^2SQL (cf. Section 3) over the original dataset and decomposed set which is elaborated on each difficulty level.

Table 4: Number of correct SQL predictions out of 332 complex questions on Spider and SpiderDec in the Text-to-SQL pipeline.

	Dataset	Ques.	# Hard	# Extra
I	Spider	100	62	38
II	SpiderDec	265	138	127
III	Diff.	165	76	89

Table 5: Distribution of complex keywords in Spider dev. set. The number of questions predicted correctly in the Text-to-SQL pipeline w/o decomposition is reported.

	Keywords	Total	Spider	SpiderDec
I	NOT IN	46	18	38
II	EXCEPT	32	4	21
III	UNION	11	0	5
IV	Total	89	22	64

Table 6: Errors generated by experts and non-experts

Error Type	Experts	Non-Experts
Complex Sub-Questions	5	49
Missed Final Sub-Questions	2	24
Missed One Keyword	8	19
Different Interpretation of Questions	5	0
Other	7	44
Total	27	136

The baseline predicted 100 questions correctly out of 332 questions in the development set, 62 questions among the hard and 38 questions from the extra hard division. On the other hand, after decomposing complex questions, the model predicted 165 more correct questions leading to 265 questions in total, including 138 hard questions and 127 extra hard questions (cf. Table 4).

By comparing the execution accuracy of the baseline on Spider and SpiderDec, we observed that the accuracy on complex data raised from 0.3 to 0.79, (cf. Table 3). We also note that the contribution of decomposition on performance gain to each division of data is approximately the same, with 76 and 89 more correct questions for hard and extra hard, respectively.

In Section 3, we discussed that sub-questions for keywords NOT IN, EXCEPT, and UNION are still hard, being less difficult than the original question. To gain insight on the impact of decomposition on complex keywords, we focus on Table 5. For the NOT IN keyword, the baseline predicted 18 questions out of 46 from the original Spider while this number increased to 38 considering the SpiderDec. Similarly, this number raised from 4 to 21 for the EXCEPT keyword, and from 0 to 5 for the UNION. Although the decomposition did not lead to the sub-questions with an easy or medium difficulty level for these complex data, the baseline outperformed significantly on SpiderDec. Given these observations, we can see the benefit of decomposition on all types of complex questions.

Looking deeper, we also examined the cases where the Text-to-SQL model failed to predict the correct SQL query even after the decomposition had applied, which is 67 questions in total. We classified the majority of errors into two groups. Table 7 shows more examples for each category. This is understandable since the decomposition task only simplifies questions by breaking them down into multiple questions. As mentioned earlier, as it does not add any additional knowledge to sub-questions, they do not contribute to any solutions for the following issues.

- **Implicit Column Names:** Within this group of questions, some of the column names in the SQL query are implicitly mentioned in the question, so the `Text-to-SQL` model requires to infer them. For instance, we have this question. *Which airlines have departing flights from both APG and CVO airports?* The column *SourceAirport* should be inferred from the phrase *departing flights*
- **General Knowledge or Table Content:** This group of questions includes one or multiple values of the tables. Sometimes these table values are considered general knowledge. Within this example, *What is the name of a country that has the shortest life expectancy in Asia?*, *Asia* is the continent, so the model needs to know this general knowledge or recognize it as the table content.

5.2 Performance of Crowd Workers on Decomposition Task (RQ2)

RQ2 investigates to what extent crowd workers can decompose complex question compared to the oracle decomposition. We first report the result of SQL knowledge survey and English proficiency. Then, we examine to what extent the decomposition leads to an accuracy boost with decomposition compared to existing `Text-to-SQL` pipeline.

In the SQL knowledge survey, all five SQL experts assessed themselves as level (4). By manually evaluating the concept definitions, we verified that all of the answers were correct and the experts had sufficient SQL knowledge to carry out the task. In terms of reading and writing skills in English, all experts had the highest levels, C2.

In total, 83 non-experts provided us with 67 SQL concept definitions when self-assessing their knowledge as level (3) or (4). All 67 definitions were labeled as incorrect. This result suggests that our non-experts group indeed did not have any background knowledge in SQL. Evaluating non-experts' English proficiency in reading and writing, we found that their skills were distributed within the level of B1 to C2. We observed 26.8% of non-experts with level B (B1, B2) and the remaining 73.1% with level C (C1, C2) in reading. In writing, we reported these numbers as 29.2% and 70.7% with level B and level C, respectively. Regarding the demographic data, among non-experts, 35.7% were female, and 62.4% were male. The ean age of participants was 32, with a minimum of 18 years and a maximum of 55 years.

Table 8 illustrates the decomposition performance of experts. We applied our decomposition approach to the hard and extra hard division of the `Spider` development set and evaluated the experts' performance in each division separately. Experts were able to decompose 61.8% of questions correctly contributed to 64.9% on hard division and 59.5% on extra hard. Although the performance on hard division is higher than the extra hard, the low difference between these two numbers suggest that the difficulty of questions does not impact the experts' decomposition performance. On the other hand, we reported that non-experts decompose 48% of questions, with 55.6% and 40% separately on hard and extra hard questions. As the performance of non-experts on hard questions is higher than extra hard questions, we can see that non-experts perceived the extra hard question as more difficult than the hard ones.

In addition to decomposition performance, we examined the potential benefit of experts' decomposition on `Text-to-SQL` pipeline accuracy. The accuracy on the original development set of `Spider` is calculated as 0.3, particularly 0.37 and 0.23 on hard and extra hard questions. Table 9 presents the accuracy of the baseline on the `Spider` and the decomposed questions by crowd workers. In terms of the accuracy boost, experts' decomposition led to 0.76 accuracy on complex questions split to 0.75 and 0.77 for hard and extra hard. We can also see that experts contributed more to improving the accuracy on extra hard questions from 0.23 to 0.77 (+0.54). Non-experts decomposition also prompted 0.59 accuracy on the complex questions, with 0.69 and 0.5 accuracies on hard and extra hard questions. Furthermore, we found that experts outperformed non-experts (accuracy 0.76 vs. 0.59), which is also in line with our finding regarding the decomposition performance. In contrast to experts, non-experts impact more on hard data, with 0.32 and 0.27 boost on hard and extra hard, respectively. Experts decomposition remarkably increased the accuracy for extra hard questions while non-experts decomposition contributed more to hard questions. In other words, the results demonstrate that when question difficulty increases, non-experts' performance deviates from the experts.

Taking these analyses further, we can see that the performance of experts (0.76) is in line with the accuracy boost achieved by `SpiderDec` (0.79). `SpiderDec` is created according to guideline in 3 based on SQL gold query while decomposition by experts only applied on natural language questions.

In terms of English proficiency, we also found that the reading and writing skill of non-experts significantly affected their decomposition performance. As the number of experts was limited, we only analyzed the impact of reading and writing factors in the non-experts group measured by a two-way ANOVA test. The test considered reading and writing as factors; the main effects were examined where $\alpha = 0.05$. For post-hoc analysis, the Tukey HSD pairwise test was used. We found that non-experts with level B1 in writing and reading significantly had lower performance than other levels. These results suggest that we can gain higher performance if we pre-screen the participants and reject those with reading and writing skills of B1.

Among the data created by experts, 37.6% of the provided decomposition were identified as errors, 6% were partially correct, and the remaining 31.4% were incorrect. On the other hand, among the decomposed questions created by non-experts, we observed 51.8% of the decompositions were error, with 13% and 38.8% were labeled as partially correct and incorrect, respectively. Taking this analysis further, we also examined different types and frequencies of errors produced by experts and non-expert, illustrated in Table 6.

We subjectively categorized the error types into two groups: recoverable errors and costly errors.

Recoverable Errors. These are errors that can be fixed through a relatively simple post-hoc analysis without modifying the decomposed queries substantially, either through expert intervention or through algorithmic interventions.

Table 7: Example of errors remain after adding the decomposition stage to the Text-to-SQL pipeline

Implicit Column Names	
Src. or Dest. Airport	Which city has most number of *departing flights?* `SELECT T1.City FROM airports AS T1 JOIN flights AS T2 ON T1.AirportCode = T2.SourceAirport GROUP BY T1.City ORDER BY count(*) DESC LIMIT 1`
Current Address	What are the last name of the students who *live in* North Carolina `SELECT T1.last_name FROM Students AS T1 JOIN Addresses AS T2 ON T1.current_address_id = T2.address_id WHERE T2.state_province_county="NorthCarolina"`
General Knowledge or Table Content	
Continent	What is the name of country that has the shortest life expectancy in *Asia*? `SELECT Name FROM country WHERE Continent = "Asia" ORDER BY LifeExpectancy LIMIT 1`
Language	Which cities are in European countries where **English** is the *official language*? `SELECT T3.Name FROM country AS T3 JOIN countrylanguage AS T4 ON T3.Code = T4.CountryCode WHERE T4.IsOfficial = "T" AND T4.Language = "English"`

Table 8: Decomposition performance of experts and non-experts reported in percentage

		Total	Hard	Extra
I	Expert	61.8	64.9	59.5
II	Non-Experts	48	55.6	40

Table 9: Accuracy of the Text-to-SQL pipeline on Spider dev. set and decomposed data created by experts and non-experts

		Total	Hard	Extra
I	Spider	0.3	0.37	0.23
II	Experts	0.76	0.75	0.77
III	Non-Experts	0.59	0.69	0.5

Costly Errors. These errors cannot be fixed easily through post-hoc analysis without modifying the decomposed queries significantly. Experts would need to rewrite one or more complete sub-queries to fix such errors.

We also manually checked the errors and classified them into five groups. We first introduce these types. We then determine which of them are recoverable and which are costly. Examples for each categories are shown in Table 10.

Complex Sub-Questions: Sub-questions in this group have the same difficulty level as the corresponding questions. Participants cannot identify how to break down the questions to make it less difficult, so they only paraphrase the question or write down sub-questions as complex as the questions. This type of error can be easily detected automatically by comparing the difficulty level of the sub-SQL queries to the gold SQL queries. However, it is a costly error. An expert is required to revise this decomposition or rewrite it from scratch, having monetary and time costs. As expected, this type of error mainly occurred for non-experts as they do not have sufficient knowledge to determine how complex their sub-questions are. An example, in Row **I** in Table 10, sub-questions are only the paraphrased form of the question. Converting them to SQL, their SQL queries are as complex as the question. Alternative sub-questions could be (i) which student owns a cat as a pet? (ii) which students are not among them? (iii) return their age and major.

Missed Final Sub-Questions. Sub-questions in this category are nearly correct, only missing the last sub-question required to return the target result set. By comparing the execution result of the question and the sub-questions, we can determine the existence of some errors. However, it is difficult to identify whether the errors fit in this group or not. An expert is needed to identify this error category manually. Similarly, the experts should write down the final sub-question. Therefore, this error type is among costly errors, while their recovery leads to remarkable performance gain. This error is more frequent among non-experts than experts which also intuitively make sense. To resolve the sub-questions in the Row **II**, we need to add this sub-question: which airports are not among those lists? Find their name.

Missed One Keyword: Sub-questions are one keyword away from the related questions. When Participants created decomposition data, they skipped or modified one keyword of the questions in their sub-questions. This type of error could mistakenly happen, or participants may not understand the role of the keyword, so they did not take that keyword into account. This error can be automatically discovered and revised by adapting an attention model to identify keywords. So, they are recoverable errors. This group of errors is the most frequent error type among experts. The sub-questions in the example of Row **III** skipped the word *Currently.* According to the tables associated with the question, students both have a current and permanent address. So, the term *Currently* plays an important role in distinguishing which columns to return.

Different Interpretation of Questions: For some questions, participants interpret them differently from what existed in the gold standard. Although these interpretations are valid, we mark them as an error because of different execution results with the corresponding questions. As resolving this error requires rewriting the sub-questions, we classify them as a costly error. Only Experts generate this type of error. Looking into the example of Row **IV**, we can see that the word *predominant* can be interpreted differently. Does it mean that the language is official? Does it mean the language is spoken with the highest percentage? Although both interpretations could be correct, the second meaning is incorporated into the dataset.

Other: Sub-questions within this error category are partly correct or thoroughly incorrect. Participants' sub-questions are not satisfied with the condition of being semantically equivalent to the corresponding questions. Finding and resolving such errors are not only time-consuming, but also they require the cost of expert interventions.

6 DISCUSSION

This study shows that decomposition leads to the accuracy boost for the `Text-to-SQL` pipeline on complex questions. This can begin to shed light on the influence of decomposition as a promising approach in improving the accuracy of `Text-to-SQL` tasks. As the follow-up study, we can build a fully automatic ML-based decomposition model integrated into the existing `Text-to-SQL` pipeline. For training such a model, it is crucial to collect a substantial amount of labeled data, such as decomposition of the family of `Spider` dataset. Our findings support the evidence of employing crowd workers for this task as a scalable method. According to our error analysis in 5.2, a fully automatic decomposition model might face several challenges. Among five error groups discussed in 5.2, the major challenges for automatic decomposition can be error types of *Complex Sub-Questions* and *Missed One Keyword*. Circumventing the challenge of *Complex Sub-Questions* is difficult since verifying the sufficient level a question should be broken down is difficult for even a human. There is a trade-off between the granularity of sub-questions and their complexity. The fine-grained the sub-questions, the less complex sub-questions we have. However, we might over-simplify questions that are not required at all. In terms of *Missed One Keyword* challenge, we demand an attention model to identify the keywords within the questions and evaluate whether those keywords existed in the sub-questions. Many of these keywords are dependent on the schema of tables. However, the attention model in training would not be enough.

Caveats and Limitations. We had an imbalance number of SQL experts, restricting us from gaining insight into their performance individually and employing any statistical tests. We also did not consider workflows to optimize decomposition such as aggregation of crowd-workers' answers and double-checking their decomposition answers by experts, which means that it would be possible to achieve higher accuracy than what we observed in our work when optimized. Furthermore, we only leveraged one `Text-to-SQL` model in our study. Although our decomposition approach is independently defined of any `Text-to-SQL` models, a comprehensive analysis of state-of-the-arts `Text-to-SQL` models can give us a better insight into the impact of decomposition on different models.

7 CONCLUSIONS

This paper explores the feasibility of decomposing complex user questions within the `Text-to-SQL` pipeline as a means to circumvent one of the significant shortcomings of `Text-to-SQL` models in complex SQL generation (RQ1). We first adapted the decomposition on the development set of the `Spider` dataset, breaking complex questions down into simpler sub-questions in a way that `Text-to-SQL` models can convert them correctly to corresponding SQL queries. We then investigated the feasibility of leveraging crowd workers to produce sufficient training data for building a ML-based model decomposing complex questions automatically (RQ2).

We defined the decomposition task for complex questions in which a complex question is split into multiple subsequent sub-questions. Having assessed the decomposition approach on complex questions in `Spider` dev. set (`SpiderDec`), we found that the accuracy raised remarkably from 30% to 79%. Our results support the evidence of decomposition as a promising approach to boost the performance of existing `Text-to-SQL` pipelines.

We then examined the performance of 88 crowd workers on decomposing the natural language questions within the development set of `Spider`. Compared to the accuracy boost of 153% (30% to 76%) as a result of the decomposition carried out by a small group of SQL experts ($N = 5$), decomposition by non-expert crowd workers ($N = 83$) led to an accuracy boost of over 96% (30% to 59%). Our findings show that crowd workers can effectively decompose complex user questions and thereby aid in creating training data at a beneficial scale for generalization of decomposition in `Text-to-SQL` pipelines.

REFERENCES

[1] I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. 1995. Natural language interfaces to databases – an introduction. *Natural Language Engineering* 1, 1 (1995), 29–81. https://doi.org/10.1017/S135132490000005X

[2] Sinem Aslan, Sinem Emine Mete, Eda Okur, Ece Oktay, Nese Alyuz, Ergin Utku Genc, David Stanhill, and Asli Arslan Esme. 2017. Human Expert Labeling Process (HELP): Towards a Reliable Higher-order User State Labeling Process and Tool to Assess Student Engagement. *Educational Technology archive* 57 (2017), 53–59.

[3] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, 1533–1544. https://aclanthology.org/D13-1160

[4] Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. 2010. Soylent: A Word Processor with a Crowd Inside. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. Association for Computing Machinery, New York, NY, USA, 313–322. https://doi.org/10.1145/1866029.1866078

[5] Ben Bogin, Jonathan Berant, and Matt Gardner. 2019. Representing Schema Structure with Graph Neural Networks for Text-to-SQL Parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 4560–4565. https://doi.org/10.18653/v1/P19-1448

[6] Ben Bogin, Matt Gardner, and Jonathan Berant. 2019. Global Reasoning over Database Structures for Text-to-SQL Parsing. 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)*. Association for Computational Linguistics, Hong Kong, China, 3659–3664. https://doi.org/10.18653/v1/D19-1378

[7] Ruichu Cai, Jinjie Yuan, Boyan Xu, and Zhifeng Hao. 2021. SADGA: Structure-Aware Dual Graph Aggregation Network for Text-to-SQL. *CoRR* abs/2111.00653 (2021). arXiv:2111.00653 https://arxiv.org/abs/2111.00653

Table 10: Examples of error generated by experts and non-experts

Examples	
I. Complex Sub-Questions	**Question:**What major is every student who does not own a cat as a pet, and also how old are they? **Sub-Questions** • How old are the students who do not own a cat as a pet? • What major is each student which does not own a cat?
II. Missed Final Sub-Questions	**Question:** Find the name of airports which do not have any flight in and out? **Sub-Questions** • Which airports are source airports? • Which airports are destination airports?
III. Missed One Keyword	**Question:** Find the last name of the students who currently live in the state of North Carolina but have not registered in any degree program. **Sub-Questions** • List the address ids of all addresses in North Carolina. • Find the student whose address corresponds with the list in North Carolina. • Find which of these students have not enrolled in any degree program? • Find the surnames associated with these students who have not enrolled and live in North Carolina.
IV. Different Interpretation of Questions	**Question:** Count the number of countries for which Spanish is the predominantly spoken language. **Sub-Questions** • What countries speak Spanish? • Given Spanish countries, count the number of officials.

[8] Yitao Cai and Xiaojun Wan. 2020. IGSQL: Database Schema Interaction Graph Based Neural Model for Context-Dependent Text-to-SQL Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 6903–6912. https://doi.org/10.18653/v1/2020.emnlp-main.560

[9] Giovanni Campagna, Rakesh Ramesh, Silei Xu, Michael Fischer, and Monica S. Lam. 2017. Almond: The Architecture of an Open, Crowdsourced, Privacy-Preserving, Programmable Virtual Assistant. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 341–350. https://doi.org/10.1145/3038912.3052562

[10] Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. LGESQL: Line Graph Enhanced Text-to-SQL Model with Mixed Local and Non-Local Relations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 2541–2555. https://doi.org/10.18653/v1/2021.acl-long.198

[11] Tommaso Caselli and Oana Inel. 2018. Crowdsourcing StoryLines: Harnessing the Crowd for Causal Relation Annotation. In *Proceedings of the Workshop Events and Stories in the News 2018*. Association for Computational Linguistics, Santa Fe, New Mexico, U.S.A, 44–54. https://aclanthology.org/W18-4306

[12] Zhi Chen, Lu Chen, Yanbin Zhao, Ruisheng Cao, Zihan Xu, Su Zhu, and Kai Yu. 2021. ShadowGNN: Graph Projection Neural Network for Text-to-SQL Parser. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 5567–5577. https://doi.org/10.18653/v1/2021.naacl-main.441

[13] DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2021. RYANSQL: Recursively Applying Sketch-based Slot Fillings for Complex Text-to-SQL in Cross-Domain Databases. *Computational Linguistics* 47, 2 (June 2021), 309–332. https://doi.org/10.1162/coli_a_00403

[14] Jonathan Corney, Andrew Lynn, Carmen Torres, Paola Di Maio, William Regli, Graeme Forbes, and Lynne Tobin. 2010. Towards crowdsourcing translation tasks in library cataloguing, a pilot study. In *4th IEEE International Conference on Digital Ecosystems and Technologies*. 572–577. https://doi.org/10.1109/DEST.2010.5610593

[15] 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 *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*. https://aclanthology.org/H94-1010

[16] Florian Daniel, Pavel Kucherbaev, Cinzia Cappiello, Boualem Benatallah, and Mohammad Allahbakhsh. 2018. Quality Control in Crowdsourcing: A Survey of Quality Attributes, Assessment Techniques, and Assurance Actions. *ACM Comput. Surv.* 51, 1, Article 7 (jan 2018), 40 pages. https://doi.org/10.1145/3148148

[17] Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2021. Structure-Grounded Pretraining for Text-to-SQL. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 1337–1350. https://doi.org/10.18653/v1/2021.naacl-main.105

[18] 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 (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 33–43. https://doi.org/10.18653/v1/P16-1004

[19] 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)*. Association for Computational Linguistics, Melbourne, Australia, 731–742. https://doi.org/10.18653/v1/P18-1068

[20] Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence Modeling for Neural Semantic Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 743–753. https://doi.org/10.18653/v1/P18-1069

[21] Zhen Dong, Shizhao Sun, Hongzhi Liu, Jian-Guang Lou, and Dongmei Zhang. 2019. Data-Anonymous Encoding for Text-to-SQL Generation. 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)*. Association for Computational Linguistics, Hong Kong, China, 5405–5414. https://doi.org/10.18653/v1/D19-1543

[22] Tim Draws, Alisa Rieger, Oana Inel, Ujwal Gadiraju, and Nava Tintarev. 2021. A Checklist to Combat Cognitive Biases in Crowdsourcing. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing* 9, 1 (Oct. 2021), 48–59. https://ojs.aaai.org/index.php/HCOMP/article/view/18939

[23] Enrique Estellés-Arolas and Fernando González-Ladrón-De-Guevara. 2012. Towards an Integrated Crowdsourcing Definition. *J. Inf. Sci.* 38, 2 (apr 2012), 189–200. https://doi.org/10.1177/0165551512437638

[24] Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving Text-to-SQL Evaluation Methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 351–360. https://doi.org/10.18653/v1/P18-1033

[25] Karën Fort, Gilles Adda, and K. Bretonnel Cohen. 2011. Last Words: Amazon Mechanical Turk: Gold Mine or Coal Mine? *Computational Linguistics* 37, 2 (June 2011), 413–420. https://doi.org/10.1162/COLI_a_00057

[26] Edwin Gamboa, Rahul Galda, Cindy Mayas, and Matthias Hirth. 2021. The Crowd Thinks Aloud: Crowdsourcing Usability Testing with the Thinking Aloud Method. In *HCI International 2021 - Late Breaking Papers: Design and User Experience*, Constantine Stephanidis, Marcelo M. Soares, Elizabeth Rosenzweig, Aaron Marcus, Sakae Yamamoto, Hirohiko Mori, Pei-Luen Patrick Rau, Gabriele Meiselwitz, Xiaowen Fang, and Abbas Moallem (Eds.). Springer International Publishing, Cham, 24–39.

[27] Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R. Woodward, John Drake, and Qiaofu Zhang. 2021. Natural SQL: Making SQL Easier to Infer from Natural Language Specifications. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics, Punta Cana, Dominican Republic, 2030–2042. https://doi.org/10.18653/v1/2021.findings-emnlp.174

[28] Catherine Grady and Matthew Lease. 2010. Crowdsourcing Document Relevance Assessment with Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Association for Computational Linguistics, Los Angeles, 172–179. https://aclanthology.org/W10-0727

[29] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards Complex Text-to-SQL in Cross-Domain Database

with Intermediate Representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 4524–4535. https://doi.org/10.18653/v1/P19-1444

[30] Tong Guo and Huilin Gao. 2020. Content Enhanced BERT-based Text-to-SQL Generation. arXiv:cs.CL/1910.07179

[31] Christopher G. Harris. [n. d.]. You're hired! an examination of crowdsourcing incentive models in human resource tasks. In *in WSDM Workshop on Crowdsourcing for Search and Data Mining (CSDM*. 15–18.

[32] Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. 2019. X-SQL: reinforce schema representation with context. arXiv:cs.CL/1908.08113

[33] Gary G. Hendrix, Earl D. Sacerdoti, Daniel Sagalowicz, and Jonathan Slocum. 1978. Developing a Natural Language Interface to Complex Data. *ACM Trans. Database Syst.* 3, 2 (jun 1978), 105–147. https://doi.org/10.1145/320251.320253

[34] Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, and Yuan Zhang. 2021. Unlocking Compositional Generalization in Pre-trained Models Using Intermediate Representations. *ArXiv* abs/2104.07478 (2021).

[35] Tobias Hossfeld, Christian Keimel, Matthias Hirth, Bruno Gardlo, Julian Habigt, Klaus Diepold, and Phuoc Tran-Gia. 2014. Best Practices for QoE Crowdtesting: QoE Assessment With Crowdsourcing. *IEEE Transactions on Multimedia* 16, 2 (2014), 541–558. https://doi.org/10.1109/TMM.2013.2291663

[36] Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% Solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*. Association for Computational Linguistics, New York City, USA, 57–60. https://aclanthology.org/N06-2015

[37] Binyuan Hui, Ruiying Geng, Qiyu Ren, Binhua Li, Yongbin Li, Jian Sun, Fei Huang, Luo Si, Pengfei Zhu, and Xiaodan Zhu. 2021. Dynamic Hybrid Relation Exploration Network for Cross-Domain Context-Dependent Semantic Parsing. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 14 (May 2021), 13116–13124. https://ojs.aaai.org/index.php/AAAI/article/view/17550

[38] Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. 2019. A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization. arXiv:cs.CL/1902.01069

[39] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a Neural Semantic Parser from User Feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 963–973. https://doi.org/10.18653/v1/P17-1089

[40] Hiroshi Kajino, Yuta Tsuboi, Issei Sato, and Hisashi Kashima. 2012. Learning from Crowds and Experts. In *HCOMP@AAAI*.

[41] Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural Semantic Parsing with Type Constraints for Semi-Structured Tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 1516–1526. https://doi.org/10.18653/v1/D17-1160

[42] Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing Probabilistic CCG Grammars from Logical Form with Higher-Order Unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Cambridge, MA, 1223–1233. https://aclanthology.org/D10-1119

[43] Dongjun Lee. 2019. Clause-Wise and Recursive Decoding for Complex and Cross-Domain Text-to-SQL Generation. 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)*. Association for Computational Linguistics, Hong Kong, China, 6045–6051. https://doi.org/10.18653/v1/D19-1624

[44] Kenton Lee, Yoav Artzi, Yejin Choi, and Luke Zettlemoyer. 2015. Event Detection and Factuality Assessment with Non-Expert Supervision. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, 1643–1648. https://doi.org/10.18653/v1/D15-1189

[45] Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the Role of Schema Linking in Text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 6943–6954. https://doi.org/10.18653/v1/2020.emnlp-main.564

[46] Fei Li and H. V. Jagadish. 2014. Constructing an Interactive Natural Language Interface for Relational Databases. *Proc. VLDB Endow.* 8, 1 (sep 2014), 73–84. https://doi.org/10.14778/2735461.2735468

[47] Yuntao Li, Hanchu Zhang, Yutian Li, Sirui Wang, Wei Wu, and Yan Zhang. 2021. Pay More Attention to History: A Context Modeling Strategy for Conversational Text-to-SQL. *CoRR* abs/2112.08735 (2021). arXiv:2112.08735 https://arxiv.org/abs/2112.08735

[48] Percy Liang, Michael I. Jordan, and Dan Klein. 2013. Learning Dependency-Based Compositional Semantics. *Computational Linguistics* 39, 2 (06 2013), 389–446. https://doi.org/10.1162/COLI_a_00127 arXiv:https://direct.mit.edu/coli/article-pdf/39/2/389/1812365/coli_a_00127.pdf

[49] Haoyan Liu, Lei Fang, Qian Liu, Bei Chen, Jian-Guang Lou, and Zhoujun Li. 2019. Leveraging Adjective-Noun Phrasing Knowledge for Comparison Relation Prediction in Text-to-SQL. 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)*. Association for Computational Linguistics, Hong Kong, China, 3515–3520. https://doi.org/10.18653/v1/D19-1356

[50] Qian Liu, Dejian Yang, Jiahui Zhang, Jiaqi Guo, Bin Zhou, and Jian-Guang Lou. 2021. Awakening Latent Grounding from Pretrained Language Models for Semantic Parsing. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics, Online, 1174–1189. https://doi.org/10.18653/v1/2021.findings-acl.100

[51] Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, and Zheng Chen. 2020. Hybrid Ranking Network for Text-to-SQL. *ArXiv* abs/2008.04759 (2020).

[52] Jianqiang Ma, Zeyu Yan, Shuai Pang, Yang Zhang, and Jianping Shen. 2020. Mention Extraction and Linking for SQL Query Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 6936–6942. https://doi.org/10.18653/v1/2020.emnlp-main.563

[53] Bart Mellebeek, Francesc Benavent, Jens Grivolla, Joan Codina, Marta R. Costa-jussà, and Rafael Banchs. 2010. Opinion Mining of Spanish Customer Comments with Non-Expert Annotations on Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk (CSLDAMT '10)*. Association for Computational Linguistics, USA, 114–121.

[54] Matteo Negri, Luisa Bentivogli, Yashar Mehdad, Danilo Giampiccolo, and Alessandro Marchetti. 2011. Divide and Conquer: Crowdsourcing the Creation of Cross-Lingual Textual Entailment Corpora. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Edinburgh, Scotland, UK., 670–679. https://aclanthology.org/D11-1062

[55] Gabriel Parent and Maxine Eskenazi. 2010. Clustering Dictionary Definitions Using Amazon Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk (CSLDAMT '10)*. Association for Computational Linguistics, USA, 21–29.

[56] Massimo Poesio, Jon Chamberlain, Udo Kruschwitz, Livio Robaldo, and Luca Ducceschi. 2013. Phrase Detectives: Utilizing Collective Intelligence for Internet-Scale Language Resource Creation. *ACM Trans. Interact. Intell. Syst.* 3, 1, Article 3 (apr 2013), 44 pages. https://doi.org/10.1145/2448116.2448119

[57] 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 (IUI '03)*. Association for Computing Machinery, New York, NY, USA, 149–157. https://doi.org/10.1145/604045.604070

[58] Alexander J. Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid Training Data Creation with Weak Supervision. *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases* 11 3 (2017), 269–282.

[59] Horacio Saggion and Graeme Hirst. 2017. *Automatic Text Simplification.* Morgan &amp; Claypool Publishers.

[60] Mike Schaekermann, Carrie J. Cai, Abigail E. Huang, and Rory Sayres. 2020. *Expert Discussions Improve Comprehension of Difficult Cases in Medical Image Assessment.* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376290

[61] Arno Scharl, Marta Sabou, Walter Rafelsberger, and Albert Weichselbraun. 2012. Leveraging the Wisdom of the Crowds for the Acquisition of Multilingual Language Resources. 379–383.

[62] Eric Schenk and Claude Guittard. 2009. Crowdsourcing: What can be Outsourced to the Crowd, and Why ?

[63] Armin Schwienbacher and Benjamin Larralde. 2010. Crowdfunding of Small Entrepreneurial Ventures. *The Oxford Handbook of Entrepreneurial Finance* (09 2010). https://doi.org/10.2139/ssrn.1699183

[64] Burr Settles. 2009. Active Learning Literature Survey.

[65] Tianze Shi, Chen Zhao, Jordan Boyd-Graber, Hal Daumé III, and Lillian Lee. 2020. On the Potential of Lexico-logical Alignments for Semantic Parsing to SQL Queries. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 1849–1864. https://doi.org/10.18653/v1/2020.findings-emnlp.167

[66] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and Fast – But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Honolulu, Hawaii, 254–263. https://aclanthology.org/D08-1027

[67] K. Stahl and M. Bravo. 2010. Contemporary Classroom Vocabulary Assessment for Content Areas. *READ TEACH* 63 (04 2010), 566–578.

[68] Yu Su, Ahmed Hassan Awadallah, Madian Khabsa, Patrick Pantel, Michael Gamon, and Mark Encarnacion. 2017. Building Natural Language Interfaces to Web APIs *(CIKM '17)*. Association for Computing Machinery, New York, NY, USA, 177–186. https://doi.org/10.1145/3132847.3133009

[69] Yibo Sun, Duyu Tang, Nan Duan, Jianshu Ji, Guihong Cao, Xiaocheng Feng, Bing Qin, Ting Liu, and Ming Zhou. 2018. Semantic Parsing with Syntax- and Table-Aware SQL Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 361–372. https://doi.org/10.18653/v1/P18-1034

[70] R. Syed and K. Collins-Thompson. 2017. Retrieval Algorithms Optimized for Human Learning. In *Proc. 40th ACM SIGIR*. 555–564.

[71] Lappoon R. Tang and Raymond J. Mooney. 2001. Using Multiple Clause Constructors in Inductive Logic Programming for Semantic Parsing. In *ECML*.

[72] Marjorie Templeton and John Burger. 1983. Problems in Natural-Language Interface to DBMS with Examples from EUFID. In *Proceedings of the First Conference on Applied Natural Language Processing (ANLC '83)*. Association for Computational Linguistics, USA, 3–16. https://doi.org/10.3115/974194.974197

[73] Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone. 2015. Learning to Interpret Natural Language Commands through Human-Robot Dialog. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)*. AAAI Press, 1923–1929.

[74] C. Thompson. 2003. Acquiring Word-Meaning Mappings for Natural Language Interfaces. *Journal of Artificial Intelligence Research* 18 (Jan 2003), 1–44. https://doi.org/10.1613/jair.1063

[75] Frederick B. Thompson, Peter C. Lockemann, Bozena Henisz-Dostert, and R. S. Deverill. 1969. REL: A Rapidly Extensible Language system. In *ACM '69*.

[76] Aobo Wang, Cong Duy Vu Hoang, and Min-Yen Kan. 2013. Perspectives on crowdsourcing annotations for natural language processing. *Language Resources and Evaluation* 47 (2013), 9–31.

[77] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. 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*. Association for Computational Linguistics, Online, 7567–7578. https://doi.org/10.18653/v1/2020.acl-main.677

[78] Bailin Wang, Wenpeng Yin, Xi Victoria Lin, and Caiming Xiong. 2021. Learning to Synthesize Data for Semantic Parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 2760–2766. https://doi.org/10.18653/v1/2021.naacl-main.220

[79] Runze Wang, Zhenhua Ling, Jing-Bo Zhou, and Yu Hu. 2021. Tracking Interaction States for Multi-Turn Text-to-SQL Semantic Parsing. In *AAAI*.

[80] Sibo Wang, Xiaokui Xiao, and Chun-Hee Lee. 2015. Crowd-Based Deduplication: An Adaptive Approach *(SIGMOD '15)*. Association for Computing Machinery, New York, NY, USA, 1263–1277. https://doi.org/10.1145/2723372.2723739

[81] David H.D. Warren and Fernando C.N. Pereira. 1982. An Efficient Easily Adaptable System for Interpreting Natural Language Queries. *American Journal of Computational Linguistics* 8, 3-4 (1982), 110–122. https://aclanthology.org/J82-3002

[82] M. B. Wesche and T. Paribakht. 1996. Assessing Second Language Vocabulary Knowledge: Depth Versus Breadth. *Canadian Modern Lang. Review* 53 (1996), 13–40.

[83] Peter Woitek, Paul Bräuer, and Holger Grossmann. 2010. A Novel Tool for Capturing Conceptualized Audio Annotations. In *Proceedings of the 5th Audio Mostly Conference: A Conference on Interaction with Sound (AM '10)*. Association for Computing Machinery, New York, NY, USA, Article 15, 8 pages. https://doi.org/10.1145/1859799.1859814

[84] William Woods, Ronald Kaplan, and Bonnie Webber. 1972. The Lunar Science Natural Language Information System: Final Report. (01 1972).

[85] W. A. Woods. 1973. Progress in Natural Language Understanding: An Application to Lunar Geology. In *Proceedings of the June 4-8, 1973, National Computer Conference and Exposition (AFIPS '73)*. Association for Computing Machinery, New York, NY, USA, 441–450. https://doi.org/10.1145/1499586.1499695

[86] Xiaojun Xu, Chang Liu, and Dawn Song. 2017. SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning. arXiv:cs.CL/1711.04436

[87] Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. SQLizer: Query Synthesis from Natural Language. *Proc. ACM Program. Lang.* 1, OOPSLA, Article 63 (oct 2017), 26 pages. https://doi.org/10.1145/3133887

[88] Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. 2021. A Survey on Deep Semi-supervised Learning. *CoRR* abs/2103.00550 (2021). arXiv:2103.00550 https://arxiv.org/abs/2103.00550

[89] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018. TypeSQL: Knowledge-Based Type-Aware Neural Text-to-SQL Generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 588–594. https://doi.org/10.18653/v1/N18-2093

[90] Tao Yu, Michihiro Yasunaga, Kai-Chou Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir R. Radev. 2018. SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task. In *EMNLP*.

[91] 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. 2019. 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)*. Association for Computational Linguistics, Hong Kong, China, 1962–1979. https://doi.org/10.18653/v1/D19-1204

[92] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 3911–3921. https://doi.org/10.18653/v1/D18-1425

[93] 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. 2019. SParC: Cross-Domain Semantic Parsing in Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 4511–4523. https://doi.org/10.18653/v1/P19-1443

[94] John M. Zelle and Raymond J. Mooney. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2 (AAAI'96)*. AAAI Press, 1050–1055.

[95] Luke S. Zettlemoyer and Michael Collins. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence (UAI'05)*. AUAI Press, Arlington, Virginia, USA, 658–666.

[96] Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions. 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)*. Association for Computational Linguistics, Hong Kong, China, 5338–5349. https://doi.org/10.18653/v1/D19-1537

[97] Yudian Zheng, Jiannan Wang, Guoliang Li, Reynold Cheng, and Jianhua Feng. 2015. QASCA: A Quality-Aware Task Assignment System for Crowdsourcing Applications. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15)*. Association for Computing Machinery, New York, NY, USA, 1031–1046. https://doi.org/10.1145/2723372.2749430

[98] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. arXiv:cs.CL/1709.00103

Visual-Meta Appendix

The data below is what we call Visual-Meta. It is an approach to add information about a document to the document itself, on the same level of the content (in style of BibTeX).
It is very important to make clear that Visual-Meta is an approach more than a specific format and that it is based on wrappers. Anyone can make a custom wrapper for custom metadata and append it by specifying what it contains: for example @dublin-core or @rdfs.
The way we have encoded this data, and which we recommend you do for your own documents, is as follows:
When listing the names of the authors, they should be in the format 'last name', a comma, followed by 'first name' then 'middle name' whilst delimiting discrete authors with ('and') between author names, like this: Shakespeare, William and Engelbart, Douglas C.
Dates should be ISO 8601 compliant.
Every citable document will have an ID which we call 'vm-id'. It starts with the date and time the document's metadata/Visual-Meta was 'created' (in UTC), then max first 10 characters of document title.
To parse the Visual-Meta, reader software looks for Visual-Meta in the PDF by scanning the document from the end, for the tag @{visual-meta-end}. If this is found, the software then looks for @{visual-meta-start} and uses the data found between these tags. This was written September 2021. More information is available from https://visual-meta.info for as long as we can maintain the domain.

```
@{visual-meta-start}

@{visual-meta-header-start}
@visual-meta{version = {1.1},
generator = {ACM Hypertext 21},
organisation = {Association for Computing Machinery}, }

@{visual-meta-header-end}

@{visual-meta-bibtex-self-citation-start}

@inproceedings{10.1145/3511095.3531282,
author = {Salimzadeh, Sara and Gadiraju, Ujwal and Hauff, Claudia and van Deursen, Arie},
title = {Exploring the Feasibility of Crowd-Powered Decomposition of Complex User Questions in Text-to-SQL Tasks},
year = {2022},
isbn = {978-1-4503-9233-4},
publisher = {Association for Computing Machinery},
address = {New York, NY, USA},
url = {https://doi.org/10.1145/3511095.3531282},
doi = {10.1145/3511095.3531282},
abstract = {Natural Language Interfaces to Databases (NLIDB), also known as Text-to-SQL models, enable users with different levels of knowledge in Structured Query Language (SQL) to access relational databases without any programming effort. By translating natural languages into SQL query, not only do NLIDBs minimize the burden of memorizing the schema of databases and writing complex SQL queries, but they also allow non-experts to acquire information from databases in natural languages. However, existing NLIDBs largely fail to translate natural languages to SQL when they are complex, preventing them from being deployed in real-world scenarios and generalizing across unseen complex databases. In this paper, we explored the feasibility of decomposing complex user questions into multiple sub-questions — each with a reduced complexity — as a means to circumvent the problem of complex SQL generation. We investigated the feasibility of decomposing complex user questions in a manner that each sub-question is simple enough for existing NLIDBs to generate correct SQL queries, using non-expert crowd workers in juxtaposition with SQL experts. Through an empirical study on an NLIDB benchmark dataset, we found that crowd-powered decomposition of complex user questions led to an accuracy boost of an existing Text-to-SQL pipeline from 30% to 59% (96% accuracy boost). Similarly, decomposition by SQL experts resulted in boosting the accuracy to 76% (153% accuracy boost). Our findings suggest that crowd-powered decomposition can be a scalable alternative to producing the training data necessary to build machine learning models that can automatically decompose complex user questions, thereby improving Text-to-SQL pipelines.},
numpages = {12},
keywords = {Text-to-SQL, Semantic Parsing, Natural Language Interface to Databases, Crowdsourcing, Human Computation, Corpus Annotation},
location = {Barcelona, Spain},
series = {HT '22},
vm-id = {10.1145/3511095.3531282} }

@{visual-meta-bibtex-self-citation-end}

@{visual-meta-end
```