Evaluating Ambiguous Questions in Semantic Parsing

Simone Papicchio
Politecnico di Torino
Turin, Italy
simone.papicchio@polito.it

Paolo Papotti
EURECOM
Sophia Antipolis, France
paolo.papotti@eurecom.fr

Luca Cagliero
Politecnico di Torino
Turin, Italy
luca.cagliero@polito.it

***Abstract*—Tabular Representation Learning and Large Language Models have recently achieved promising results in solving the Semantic Parsing (SP) task. Given a question posed in natural language on a relational table, the goal is to return to the end-users executable SQL declarations. However, models struggle to produce the correct output when questions are ambiguously defined w.r.t the table schema. Assessing the robustness to data-ambiguity can be particularly time-consuming as entails seeking ambiguous patterns on a large number of queries with varying complexity. To automate this process, we propose *Data-Ambiguity Tester*, a pipeline for data-ambiguity testing tailored to SP. It first automatically generates non-ambiguous natural language questions and SQL queries of varying complexity. Then, it injects ambiguous patterns, extracted from a human-annotated set of relational tables, in the natural language questions. Finally, it quantifies the level of ambiguity using customized performance metrics. Results show strengths and limitations of existing models in coping with ambiguity between questions and tabular data.**

***Index Terms*—Tabular Representation Learning, Semantic Parsing, Text2SQL, Data-Ambiguity, NL2SQL, Large Language Models.**

I. Introduction

State-of-the-art models for table understanding are pretrained on very large collections of tabular data to understand their schema- and instance-level properties [1]. These models are then fine-tuned to accomplish downstream tasks such as Question Answering [2], [3], Table Retrieval [4], [5], Table Comprehension [6], [7], and Table Content Prediction [8].

Given a relational table, Semantic Parsing (SP) aims to translate natural language (NL) questions to SQL declarations. SP supports end-users who are not proficient in SQL code writing and speeds up user-database interactions [9]. State-of-the-art SP approaches include Tabular Representation Learning (TRL) models fine-tuned for this task (e.g., [10], [11]) and general-purpose Large Language Models (LLMs) [12]. However, both TRL models and LLMs are challenged by the inherent ambiguity between text (NL questions) and relational data (table schema and instance).

Let us consider, as a toy example, Table I. A NL question such as *Show me the size of the Abalone fish with Id 1* is ambiguous because the concept of *size* can be arbitrarily mapped to either attributes *Length*, *Diameter*, *Height*. This is a well recognized problem for SP, but there is not systematic evaluation on how models handle these challenging cases [9], [14]. In this work, we classify as ambiguous for SP a natural language question that contains free-text that ambiguously refers to more than one attribute of the relation schema. Hence, the Text2SQL process should generate many valid SQL declarations, each one corresponding to a different attribute combination. In this work, we focus on studying how existing models handle ambiguous relationships between the text of the NL question and the relational schema.

TABLE I
Toy example extracted from the Abalone dataset [13]

AbaloneId	Sex	Length	Diameter	Height
1	F	0.40	0.32	0.13
2	M	0.39	0.32	0.11
3	M	0.32	0.26	0.09

Figure 1 shows the main steps in DAMBER[1] (*Data-AMBiguity testER*), a new pipeline for ambiguous test generation and evaluation tailored to SP on tabular data. DAMBER relies on QATCH [15], a recently proposed testing benchmark for TRL models, to initially generate a large set of Text2SQL tests over a given relational table. Each test consists of a triple with an unambiguous NL question, the corresponding SQL script, and the query output. Using a template-based approach, QATCH produces test queries with varying level of complexity, executes them on the SP models, and then computes a set of performance metrics over the returned tuples. To inject ambiguity in NL questions, DAMBER leverages a human-curated set of tables annotated with *schema-only ambiguous labels*, i.e., free-text labels that are deemed ambiguous with respect to at least two attributes in the table schema [16]. For instance, every attribute in {*Height*, *Length*, and *Diameter*} in the schema of Table I is paired with the ambiguous labels {*size*, *dimension*, and *measurement*} because such labels denote concepts that are related to, but not specifically mapped to a particular attribute. ⟨Label, attribute⟩ pairs such as ⟨size, Height⟩, ⟨size, Length⟩, and ⟨dimension, Height⟩ are used to modify the original QATCH queries to inject ambiguity. Notice that for each query in the original set the injection can produce several ambiguous versions. Prompting TRL models and LLMs with these ambiguous questions allows DAMBER to generate

[1]We exploit the wordplay "*damber*" vs. "*dumber*" to stress the importance of detecting LLMs/TRL models' weaknesses.

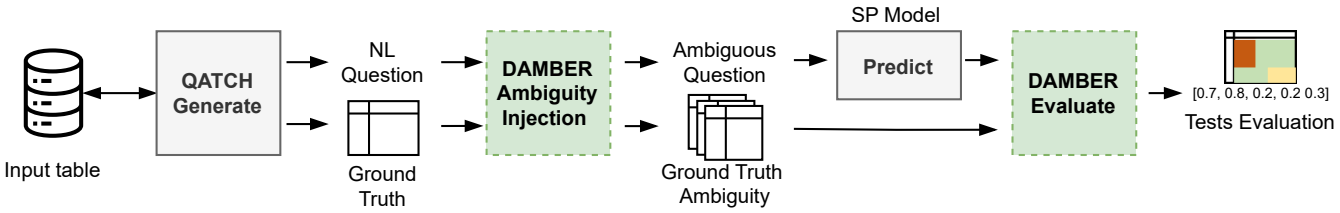

Fig. 1. *Data-AMBiguity testER*: Pipeline for ambiguous test generation and evaluation. Given a relational table, a set of unambiguous tests is generated. Then, data-ambiguity is injected in the NL questions using ground truth ambiguity information about the input table. Models are then prompted with ambiguous questions to generate SQL scripts. Finally, tuple results are compared against the results for the original SQL scripts to compute quality metrics.

SQL declarations and the corresponding output in relational form. Model outputs are then compared with the expected output of the (unambiguous) SQL query. We use the query results with new quantitative performance metrics designed to measure the robustness w.r.t. ambiguity.

The result show a clear superiority of LLMs over TRL models in handling ambiguous queries. LLMs show promising proficiency in associating ambiguous labels with the correct database attributes. This is contrasted by the struggles of TRL models, which often resulted in SQL errors while coping with ambiguous queries. However, even the best LLM model for this task (ChatGPT 3.5) cannot achieve consistent performance across all tests and tables, with results ranging between 0.98 on the easiest table and 0.60 on the hardest one in our quality metrics.

The remainder of this paper is organized as follows. Section II introduces preliminary concepts. Section III describes DAMBER. Section IV shows preliminary results, whereas Section V discusses future extensions of the present work.

II. Preliminaries

Semantic Parsing from tabular data Given a relational table $\mathcal{R}$ and a natural language (NL) question Q, Semantic Parsing (SP) aims to generate the corresponding SQL declarations $\mathrm{S_Q}$.

Data-ambiguity types Data-ambiguity refers to the case in which we have multiple interpretations of the relationship between text and data [16]. Ambiguity is due to the fact that NL used to pose a free-text question, describe metadata in the relational schema, or fill table content is inherently ambiguous. We can enumerate three main types of data-ambiguity [9]: (1) *Schema-only ambiguity*, where the NL question has an ambiguous mapping between the question and the table schema; (2) *Content ambiguity*, where there is an ambiguous mapping between the question and the table instance values; (3) *Question ambiguity*, where the ambiguity lies in the NL question only and there is a lack of contextual information to determine the query intent. In this work, we address the ambiguity between NL question and the relational schema, i.e., case (1).

Semantic Parsing with data-ambiguity Let M be a TRL model or a LLM capable to accomplishing the SP task. We prompt M with an ambiguous question A-Q to get the Text2SQL reformulation $\mathrm{S}^M_{\text{A-Q}}$ as well as its corresponding output $\mathcal{O}^M_{a-Q}$ in relational form. Let $\underline{\mathrm{S}}^1_{\text{A-Q}}$, ..., $\underline{\mathrm{S}}^n_{\text{A-Q}}$ be the alternative SQL declarations of A-Q over $\mathcal{R}$ and let $\underline{\mathcal{O}}^1_{a-Q}$, ..., $\underline{\mathcal{O}}^n_{a-Q}$ be the corresponding outputs. Our purpose is to quantify the ability of M to handle data-ambiguity by comparing the output $\mathcal{O}^M_{AQ}$ with the expected results ($\underline{\mathcal{O}}^1_{a-Q}$, ..., $\underline{\mathcal{O}}^n_{a-Q}$).

III. Data-Ambiguity Tester

SP test creator As depicted in Figure 1, DAMBER leverages QATCH [15] to generate a initial set of unambiguous SP tests to assess models on tabular data. Given a relational table $\mathcal{R}$, QATCH generates test queries consisting of triples $\langle \mathrm{Q}, \mathrm{S_Q}, O_\mathrm{Q} \rangle$ through a template-based query generator. This generator employs templates to define SQL declarations (e.g., `SELECT` *attribute* `FROM TABLE`), and the variables within these templates (e.g., *attribute*) are automatically populated by the tool based on the values in the schema and active domain of the given T. The resulting SQL test suite encompasses various levels of complexity. Examples of SQL query categories are outlined in Table II.

An example of intermediate output generated by the SP test creator of DAMBER is denoted by `Prior to ambiguity injection` in Table III.

TABLE II
EXAMPLE OF TEMPLATES FOR QUERIES IN SQL AND NATURAL LANGUAGE. T IS THE TARGET RELATIONAL TABLE. $c_i \in \mathrm{S}_R$ $(1 \leq i \leq n)$ IS AN ATTRIBUTE OF THE $\mathcal{R}$'S SCHEMA. *ord* IS THE ORDER OF VISUALIZATION OF THE TUPLES IN THE OUTPUT. NL=NATURAL LANGUAGE QUESTION.

SQL Category	Type	Content
`Project`	SQL	SELECT $\{c_1, \ldots, c_n\}$ FROM $\{T\}$
	NL	Show $\{c_1, \ldots, c_n\}$ in table $\{\mathcal{R}\}$
`Distinct`	SQL	SELECT DISTINCT $\{c_1, \ldots, c_n\}$ FROM $\{T\}$
	NL	Show the different $\{c_1, \ldots, c_n\}$ in table $\{\mathcal{R}\}$
`Select`	SQL	SELECT * FROM $\{T\}$ WHERE $\{c_i\}$ {op} {val}
	NL	Show data of table {t} where $\{c_i\}${op}{val}
`Order by`	SQL	SELECT * FROM $\{T\}$ ORDER BY $\{c_i\}$ {ord}
	NL	Show data for table $\{T\}$ in {ord} order by $\{c_i\}$
`SIMPLE AGG`	SQL	SELECT COUNT(DISTINCT $\{c_i\}$) FROM $\{\mathcal{R}\}$
`COUNT DISTINCT`	NL	How many different $\{c_i\}$ are in table $\{\mathcal{R}\}$?
`SIMPLE AGG`	SQL	SELECT MIN/MAX/AVG $\{n_i\}$ FROM $\{T\}$
`MIN/MAX/AVG`	NL	Find the min/max/avg of $\{n_i\}$ in table $\{\mathcal{R}\}$

Data-ambiguity injection To inject schema-only data-ambiguity in the generated SP tests, DAMBER relies on a human-curated collection of tables with ground truth ambiguity

information [16]. This corpus consists of a set of relational tables $\mathcal{R}_{gt}^1$, ..., $\mathcal{R}_{gt}^k$ whose schema attributes are annotated with one or more ambiguous labels. For instance, the attributes *Length*, *Diameter*, and *Height* in Table I are annotated with the ambiguous labels *Size* and *Measurement*. These labels are then used as source of ambiguity in the NL questions.

Examples of data-ambiguity injection are given in Table III. Considering each ambiguous label (e.g., *distance* associated with schema attributes *Length*, *Diameter*, and *Height*), the schema attribute in the Ambiguous Question has been replaced with the respective ambiguous label.

The next key step is the generation of the target queries, i.e., the expected output for an ambiguous question. The problem can be framed as follows: *What is the correct SP output for an ambiguous NL question?*. In this study, we define a semantics that we found useful in practice. A valid solution is *any* query that replaces the label with one of its corresponding schema attributes (e.g., `Target Queries` in Table III).

We adopt this approach due to its versatility, enabling the effective handling of all SQL granularity supported by QATCH. Indeed, one could argue that a solution to an ambiguous NL question is to include all associated attributes i.e., ⟨`SELECT` *"Length"*, *"Diameter"*, *"Height"* `FROM TABLE`⟩. However, this strategy proves impractical when confronting aggregation scenarios, such as ⟨`SELECT MAX`*("Length")* `FROM TABLE`⟩.

The above procedure enables DAMBER to automatically conduct an in-depth investigation of all the nuances of data-ambiguity in SP models. Notice that the injection of data ambiguity can be modified or personalized by the user.

TABLE III
AMBIGUOUS TEST GENERATION. AMBIGUOUS LABEL "DISTANCE" IS ASSOCIATED WITH ATTRIBUTES "LENGTH", "DIAMETER", "HEIGHT". THE LABEL REPLACES THE ORIGINAL ATTRIBUTE IN THE NEW AMBIGUOUS QUESTION. TARGET QUERIES COVER ALL ASSOCIATED ATTRIBUTES.

`Prior to ambiguity injection`	
Table name	Abalone
SQL category	Project
Query	SELECT "Length" FROM "abalone"
Question	Show all "Length" in the table abalone
`After ambiguity injection`	
Table name	Abalone
SQL category	Project
Target Queries	SELECT "Length" FROM "abalone" SELECT "Diameter" FROM "abalone" SELECT "Height" FROM "abalone"
Ambiguous Question	Show all "distance" in the table abalone

Ambiguity test evaluation We defined the expected output for an ambiguous question input. For evaluating the target queries, we leverage five established performance metrics [15] to compare model output and ground truth.

- *Cell Precision*. The proportion of table cells in the model output that match the ground truth.
- *Cell Recall*. The fraction of table cells that are present in the ground truth and in the prediction.

TABLE IV
AMBIGUOUS TEST EVALUATION. GIVEN AS INPUT THE AMBIGUOUS QUESTION IN TABLE III, WE REPORT THREE MODEL PREDICTIONS AND THE RESPECTIVE EVALUATION RESULTS.

`Model Predictions`	
Model 1	SELECT "distance" FROM abalone
Model 2	SELECT * FROM abalone
Model 3	SELECT "Length" FROM abalone

`Model Evaluation`

	Cell precision	**Cell recall**	**Tuple cardinality**	**Tuple constraint**	**Tuple order**
Model 1 evaluation	0.0	0.0	0.0	0.0	-
Model 2 evaluation	1/5	1.0	1.0	0.0	-
Model 3 evaluation	1.0	1.0	1.0	1.0	-

- *Tuple constraint*. The percentage of true tuples in the query results. It equals one when the the generated output instance shares identical schema, cardinality, and cell values with the target; otherwise, it is zero.
- *Tuple cardinality*. The ratio of output and ground truth cardinality; it ignores cell values.
- *Tuple order*. The Spearman rank correlation coefficient normalized between zero and one (zero opposite order, one otherwise). It is evaluated only for queries with the ORDER BY clause.

While these metrics are effective for general SP evaluation, they do not account for multiple ambiguity target queries. To address this issue, we compute the metrics between the predicted query and every target query (e.g., the three target queries under `After ambiguity injection` in Table III). We then "match" the predicted query with the target query with the highest mean across the metrics. In the experiments, we report this matched target query and its resulting metrics scores.

Table IV illustrates an example of the evaluation with three predicted query from different models. In the case of Model 1, the ambiguous label is mistakenly interpreted to be part of the table schema, resulting in an execution error, i.e. all metrics are scored as zero. Conversely, Models 2 and 3 do not trigger an execution error, allowing for a comparison of predictions with each target query and generating three sets of metrics (one for each target query). Model 2 has the same average value across metrics for all target queries, we therefore consider all matches equivalent. For Model 3, the highest average for the metrics is obtained with the exact match between the predicted query and one of the target queries.

The proposed heuristics rewards the target queries that either exactly match the predicted one or approximately match with highest metric scores.

IV. EXPERIMENTS

SP models. We test 5 models: three TRL models (RESDSQL [17], GAP [18], and UNIFIEDSKG [19]) and two LLMs (CHATGPT 3.5 TURBO-TURBO-0613 [20] and LLAMA-CODE [21]). RESDSQL is composed of a skeleton-aware decoding

framework and a seq2seq architecture with a ranking-enhanced encoding based on T5. UNIFIEDSKG implements an encoder-decoder (text-to-text) model based on T5. We configure both models with the *T5 large* setting. GAP employs a generative model to get pre-training data, facilitating the concurrent learning of NL utterances and table schemas.

To improve the LLMs' performance in the SP task, we leverage the few-shot learning technique; the model takes as input a prompt with examples of text-to-SQL conversions, called "shots". The prompt involves generating the corresponding SQL query based on the provided table name, table schema, and NL question. For the "shots", we use unambiguous text-to-SQL examples, exclusively employed to enhance the performance of LLMs in the SP task. These examples are intended to improve the model's ability to deal with SP, rather than its ability to deal with ambiguity. Details on the few-shot are available at https://github.com/spapicchio/QATCH.

Datasets. We rely on an existing annotated corpus, which includes 13 tables from the UCI repository and the WebTables collection [16]. Ten annotators were asked to identify ambiguous attributes and a possible label for every such pair. For example, once identified "weight" and "height" as ambiguous attributes in a table, one suggested label is "measure". Labels are added to the ground truth based on majority voting and an additional participant to solve uncertainty cases. The corpus comprises 1321 attribute pairs, with 252 ambiguous pairs and an average of 1.8 labels per pair.

How to interpret the results. The evaluation metrics are dependent on the data rather than the query itself. This ensures high results when comparing two SQL syntactically distant but with similar semantic. Consider the Table I and the query ⟨`SELECT` *"Sex"* `FROM` *"abalone"* `ORDER BY ASC`⟩ and the model prediction ⟨`SELECT DISTINCT` *"AbaloneID", "Sex"* `FROM` *"abalone"* `ORDER BY DESC`⟩, the metrics scores are as follows: (i) Cell precision: 0.5 as half of predicted attributes are correct; (ii) Cell recall: 1.0 as all target attributes are in the prediction; (iii) Tuple cardinality: 0.5 as the predicted query returns fewer tuples; (iv) Tuple constraint: 0.0 as returned schemas differs; (v) Tuple order: 0.0 order is the opposite.

Results. Table V presents the results for all models evaluated on ambiguous questions averaged over all tests and tables.

TABLE V
RESULTS FOR ALL MODELS WITH AMBIGUOUS QUESTIONS; AVERAGE OF ALL TESTS ON 13 TABLES.

Model	Cell precision	Cell recall	Tuple cardinality	Tuple constraint	Tuple order
CHATGPT 3.5 (LLM)	**0.76**	**0.78**	**0.80**	**0.63**	0.83
LLAMA-CODE (LLM)	0.52	0.54	0.58	0.39	**0.86**
RESDSQL (TRL)	0.37	0.38	0.42	0.31	0.46
UNIFIEDSKG (TRL)	0.36	0.37	0.39	0.31	0.65
GAP (TRL)	0.24	0.24	0.26	0.21	0.27

LLMs consistently outperform TRL models, suggesting that their superior reasoning capabilities make a difference for this task. LLMs associate the ambiguous label with at least one of the ambiguous attribute in most predictions. Conversely, the predictions from TRL models lead to SQL errors due to the presence of the ambiguous label in the query. Generally, both Cell precision and Cell recall exceed Tuple Constraint, indicating the models' proficiency in discerning the relevant attributes for projection. However, the preservation of the tuple structure for the returned values (Tuple Constraint) emerges as the most challenging aspect. Results for tuple order are higher compared to the other metrics as ambiguity does not affect the inclusion of "ASC" or "DESC" in the SQL query. As ChatGPT outperforms the other models significantly, we focus next on a more detailed examination of its results.

ChatGPT 3.5's results for different datasets show that it struggles to achieve consistent performance across tables, with average results ranging from 0.98 in WDC_631 to 0.60 in Abalone. An examination of the results reveals that the WDC_631 table, containing ambiguous labels closely related to schema attributes such as "price" and "pricing" with "Part #" and "List Price", aids the model. However, the model fails short in more complex scenarios, such as the ABALONE table, which involves attributes like "length", "diameter", and "height" with the ambiguous label "distance".

TABLE VI
RESULTS FOR CHATGPT-SP 3.5-TURBO-0613 FOR AMBIGUOUS QUESTIONS: AVERAGE OF ALL TESTS ON MULTIPLE TABLES.

SQL Category	Cell precision	Cell recall	Tuple cardinality	Tuple constraint	Tuple order
`Project`	0.76	0.89	0.95	0.61	-
`Order By`	0.80	0.82	0.93	0.75	0.83
`Distinct`	0.85	0.87	0.93	0.82	-
`SIMPLE-AGG AVG-MAX-MIN`	0.74	0.76	0.96	0.72	-
`SIMPLE-AGG COUNT-DISTINCT`	0.88	0.88	**1.00**	0.88	-

Table VI reports how ChatGPT performs with various types of queries containing ambiguity. ChatGPT inserts all attributes associated with an ambiguous label when faced with uncertainty, e.g., the predicted query for the ambiguous question in Table III is ⟨`SELECT` *"Length"*, *"Diameter"*, *"Height"* `FROM` *"abalone"* ⟩. This is confirmed by the high value of Cell recall. However, despite the model reliably incorporates at least one attribute associated with the ambiguous label into its predictions, prediction fails short when dealing with aggregation, as shown by lower values for `SIMPLE-AGG-AVG-MAX-MIN`.

V. FUTURE RESEARCH DIRECTIONS

As future work we plan to the generalize the concept of data-ambiguity beyond the text-schema relation. Specifically, we will also consider table content and question ambiguity (see Section II). Furthermore, we will explore the effect of data-ambiguity on other downstream tasks such as Tabular Question Answering and Tabular Computational Fact-Checking.

REFERENCES

[1] G. Badaro, M. Saeed, and P. Papotti, "Transformers for tabular data representation: A survey of models and applications," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 227–249, 2023.

[2] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. Eisenschlos, "TaPas: Weakly supervised table parsing via pre-training," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 4320–4333. [Online]. Available: https://aclanthology.org/2020.acl-main.398

[3] Q. Liu, B. Chen, J. Guo, Z. Lin, and J.-g. Lou, "TAPEX: Table pre-training via learning a neural SQL executor," *arXiv preprint arXiv:2107.07653*, 2021.

[4] B. Kostić, J. Risch, and T. Möller, "Multi-modal retrieval of tables and texts using tri-encoder models," in *Proceedings of the 3rd Workshop on Machine Reading for Question Answering*. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 82–91. [Online]. Available: https://aclanthology.org/2021.mrqa-1.8

[5] F. Pan, M. Canim, M. Glass, A. Gliozzo, and P. Fox, "CLTR: An end-to-end, transformer-based system for cell-level table retrieval and table question answering," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Aug. 2021, pp. 202–209. [Online]. Available: https://aclanthology.org/2021.acl-demo.24

[6] L. Du, F. Gao, X. Chen, R. Jia, J. Wang, J. Zhang, S. Han, and D. Zhang, "TabularNet: A neural network architecture for understanding semantic structures of tabular data," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 322–331.

[7] Y. Suhara, J. Li, Y. Li, D. Zhang, Ç. Demiralp, C. Chen, and W.-C. Tan, "Annotating columns with pre-trained language models," *arXiv preprint arXiv:2104.01785*, 2021.

[8] X. Deng, H. Sun, A. Lees, Y. Wu, and C. Yu, "Turl: Table understanding through representation learning," *SIGMOD Rec.*, vol. 51, no. 1, p. 33–40, jun 2022. [Online]. Available: https://doi.org/10.1145/3542700.3542709

[9] A. Floratou, F. Psallidas, F. Zhao, S. Deep, G. Hagleither, W. Tan, J. Cahoon, R. Alotaibi, J. Henkel, A. Singla, A. van Grootel, B. Chow, K. Deng, K. Lin, M. Campos, V. Emani, V. Pandit, V. Shnayder, W. Wang, and C. Curino, "Nl2sql is a solved problem... not!" in *14th Annual Conference on Innovative Data Systems Research (CIDR'24)*., 2024.

[10] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel, "TaBERT: Pretraining for joint understanding of textual and tabular data," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 8413–8426. [Online]. Available: https://aclanthology.org/2020.acl-main.745

[11] T. Yu, C.-S. Wu, X. V. Lin, bailin wang, Y. C. Tan, X. Yang, D. Radev, richard socher, and C. Xiong, "GraPPa: Grammar-augmented pre-training for table semantic parsing," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=kyaIeYj4zZ

[12] A. Liu, X. Hu, L. Wen, and P. S. Yu, "A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability," *arXiv preprint arXiv:2303.13547*, 2023.

[13] D. Dua and C. Graff, "Uci machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[14] Z. Huang, P. K. Damalapati, and E. Wu, "Data ambiguity strikes back: How documentation improves gpt's text-to-sql," *arXiv preprint arXiv:2310.18742*, 2023.

[15] S. Papicchio, P. Papotti, and L. Cagliero, "Qatch: Benchmarking sql-centric tasks with table representation learning models on your data," in *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[16] E. Veltri, G. Badaro, M. Saeed, and P. Papotti, "Data ambiguity profiling for the generation of training examples," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, 2023, pp. 450–463.

[17] H. Li, J. Zhang, C. Li, and H. Chen, "Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql," in *AAAI*, 2023.

[18] P. Shi, P. Ng, Z. Wang, H. Zhu, A. H. Li, J. Wang, C. N. dos Santos, and B. Xiang, "Learning contextual representations for semantic parsing with generation-augmented pre-training," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 15, 2021, pp. 13 806–13 814.

[19] T. Xie, C. H. Wu, P. Shi, R. Zhong, T. Scholak, M. Yasunaga, C.-S. Wu, M. Zhong, P. Yin, S. I. Wang *et al.*, "Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models," *arXiv preprint arXiv:2201.05966*, 2022.

[20] OpenAI, "Chatgpt 3.5," 2023. [Online]. Available: https://openai.com/blog/chatgpt

[21] B. Roziere, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin *et al.*, "Code llama: Open foundation models for code," *arXiv preprint arXiv:2308.12950*, 2023.