Deep Query Explorer: Automated Data Augmentation for Table Question Answering using LLM Agents

Anonymous ACL submission

Abstract

Models for table question answering (TableQA) often fail to generalize under minor perturbations to questions or table structures. While this brittleness is somewhat expected for foundation models lacking explicit structural pre-training, it remains surprising for fine-tuned models that undergo task-specific adaptation. We argue that a key cause lies in the limited semantic and syntactic diversity of existing intermediate pre-training datasets. To address this, we introduce a novel multi-agent framework that autonomously generates a diverse and progressively challenging curriculum of natural language questions and corresponding SQL queries. Our system comprises a curriculum agent that proposes new questions, a translator agent that produces executable SQL, and a diversifier agent that enhances linguistic variety, all coordinated through a central library ensuring non-redundancy and gradual complexity. Pre-training a BART-based model on the resulting dataset, DQE, yields substantial robustness gains on the ROBUT benchmark—improving by +6.8% on synonym-based perturbations over strong baselines such as TAPEX. Further analysis shows that our dataset exhibits markedly greater SQL syntactic diversity, confirming the effectiveness of our agent-based data generation approach.

1 Introduction

Table Question Answering (TableQA) aims to answer natural language questions over structured tables (Pasupat and Liang, 2015). Recent large language models (LLMs) (Brown et al., 2020) have achieved remarkable performance on standard TableQA benchmarks (Dou et al., 2023), yet they still exhibit poor generalization under minor perturbations to questions or table structures (Zhao et al., 2023; Zhou et al., 2024; Li et al., 2025). While such brittleness is expected for foundation models not explicitly trained for table reasoning, it remains surprising for fine-tuned models that undergo domain adaptation (Liu et al., 2021; Jiang et al., 2022; Herzig et al., 2020). Most TableQA systems adapt text-based architectures (Devlin et al., 2019; Lewis et al., 2019) through intermediate pre-training on table-related tasks before fine-tuning on downstream datasets (Pasupat and Liang, 2015; Chen et al., 2019). However, existing pre-training corpora lack semantic and syntactic diversity, being largely template-based or manually annotated (Yu et al., 2020; Shi et al., 2022). Consequently, models overfit to specific question patterns and fail to handle linguistic or structural variations (Sui et al., 2024). We argue that this limited diversity is a central cause of TableQA brittleness. To address it, we introduce a multi-agent framework that autonomously generates diverse and progressively challenging question–SQL pairs [1]. Our data distillation framework (Radosavovic et al., 2018) comprises a *curriculum agent* proposing new questions, a *translator agent* generating executable SQL validated by an execution engine, and a *diversifier agent* enhancing linguistic variety. Pre-training a BART-based model on the resulting dataset, Deep Query Explorer **DQE**, substantially improves robustness on the ROBUT benchmark—achieving significant gains over strong baselines such as TAPEX—demonstrating that richer, automatically generated pre-training data leads to stronger generalization in table reasoning.

2 Related Work

2.1 SQL Generation and Semantic Parsing

Generation-based methods use encoder–decoder architectures to translate natural language questions into executable SQL queries (Wang et al., 2019; Li et al., 2023). Recent work shows that large language models can perform text-to-SQL generation

[1] Code and data will be released publicly on GitHub upon paper acceptance to ensure reproducibility.

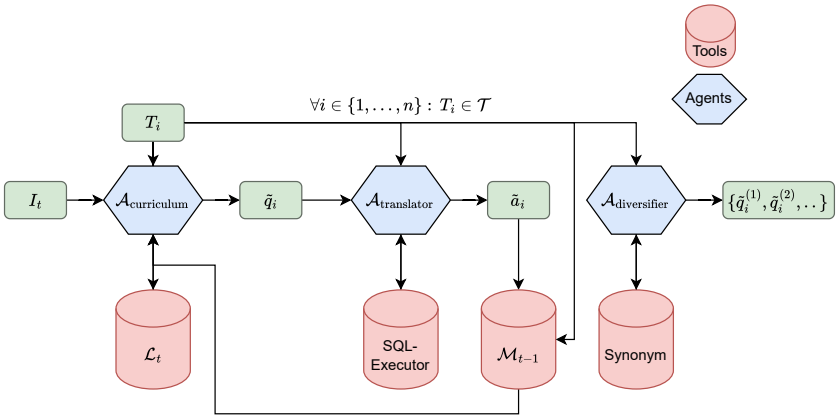

Figure 1: Overview of our agent-based data generation pipeline. An instruction I_t prompts the **Curriculum Agent** to generate a new question $\tilde{q}_i$ for a given table T_i, guided by the state of the Library $\mathcal{L}_t$ and the performance of the current model $\mathcal{M}_{t-1}$. The **Translator Agent** converts $\tilde{q}_i$ into an executable SQL query, producing an answer $\tilde{a}_i$. Finally, the **Diversifier Agent** generates multiple linguistic variations of the question. The new data is used to fine-tune the next iteration of the model.

in zero- or few-shot settings (Zhang et al., 2023; Pourreza and Rafiei, 2023; Li and Xie, 2024). In our approach, we leverage such LLMs as generators and employ a SQL executor as a verifier to retain only executable, semantically valid queries for data augmentation.

2.2 Table QA Fine-tuning

Fine-tuning adapts pre-trained models to table reasoning tasks, often through intermediate pre-training to inject tabular inductive biases (Eisenschlos et al., 2020). Prior approaches such as TaBERT (Eisenschlos et al., 2020), TAPEX (Liu et al., 2021), and OmniTab (Jiang et al., 2022) rely on synthetic or template-based SQL supervision and expert-crafted data generation (Yu et al., 2020; Yoran et al., 2021). While effective, these methods require costly manual design and offer limited linguistic diversity. In contrast, we leverage LLM as automatic data generators, coupled with a SQL executor for verification, enabling scalable and diverse data augmentation without human or template-based intervention.

2.3 Emergent Properties.

Scaling laws show that larger models exhibit emergent capabilities such as zero-shot reasoning, instruction following, and tool use (Brown et al., 2020; Wei et al., 2022; Ouyang et al., 2022; Schick et al.). These abilities enable LLMs to act as autonomous agents that plan, verify, and collaborate to solve complex tasks (Yang et al., 2024; Sapkota et al., 2025). Building on this paradigm, we design an agent-based architecture in which multiple LLM agents interact with tools (Wang et al., 2023) such as a SQL executor to generate more diverse and complex question SQL pairs for data augmentation.

3 Methodology

The core of our approach is an agent-based pipeline designed to autonomously generate a diverse and high-quality dataset for Table QA. The system is composed of three main LLM agents Curriculum, Translator, and Diversifier that interact with a central Library and the model being trained.

3.1 Curriculum Agent

The curriculum agent, $A_{curriculum}$, is responsible for driving the exploration process by proposing new, relevant and progressively complex questions. At each step t, for a given table T_i, the agent receives an instruction I_t and generates a candidate question $\tilde{q}_i$. This process is guided by two critical feedback loops:

Library-based Novelty Check: To avoid redundancy and encourage diversity, the generated question is compared against the existing questions in our library, $\mathcal{L}_t$. We use a pre-trained sentence encoder "Alibaba-NLP" (Zhang et al., 2024) to compute embeddings for all questions. A new question $\tilde{q}_i$ is rejected if its maximum cosine similarity with any question in the library exceeds a threshold $\gamma_{max} = 0.9$. This ensures that each new question is sufficiently distinct from all prior ones.

Model-based Difficulty Check: To create a cur-

riculum that is tailored to the current model's weaknesses, we use the model from the previous iteration, $\mathcal{M}_{t-1}$, as a discriminator. We execute $\mathcal{M}_{t-1}(T_i, \tilde{q}_i)$ to see if the model can already answer the proposed question correctly. If it can, the question is rejected as it provides no new learning signal. This mechanism steers the agent towards generating examples that are challenging for the current model, creating a dynamic and adaptive curriculum.

3.2 Translator Agent

Once the curriculum agent proposes a valid natural language question $\tilde{q}_i$, the translator agent, $A_{translator}$, is tasked with converting it into an executable SQL query, $\tilde{s}_i$. This process may involve several rounds of iterative prompting. The agent's initial SQL output is passed to a SQL execution engine, $\mathcal{E}_{SQL}$. If the query fails to execute or produces an incorrect result, the error feedback is provided back to the agent, which then attempts to correct its output. This verified pair $(\tilde{s}_i, \tilde{a}_i)$, where $\tilde{a}_i = \mathcal{E}_{SQL}(\tilde{s}_i, T_i)$, ensures the logical and syntactic correctness of the generated data.

3.3 Diversifier Agent

To enhance the linguistic diversity of our dataset, the diversifier agent, $A_{diversifier}$, takes the accepted question $\tilde{q}_i$ and generates multiple paraphrases, e.g., $\{\tilde{q}_i^{(1)}, \tilde{q}_i^{(2)}, ...\}$. These variants retain the original semantic intent (and thus the same SQL query and answer) but use different lexical and syntactic structures. This step helps the final model become robust to superficial linguistic variations, a common failure point for Table QA systems.

3.4 The Library

The library, $\mathcal{L}$, is the central repository of knowledge in our system. It stores successful (question, SQL) pairs generated throughout the process. For efficient retrieval and novelty checking, we store the dense vector embedding of each SQL query's abstract syntax tree. This allows us to use cosine similarity to enforce two constraints on any newly proposed query template $q^{(t+1)}$:

1. **Novelty**: The query must be sufficiently distinct from all previously stored queries: $\max_{i \in \{0..t\}} \cos(\phi(q^{(t+1)}), \phi(q^{(i)})) \leq \gamma_{max}$.

2. **Continuity**: To maintain a smooth curriculum, the query should not be excessively different from the most recent query $q^{(t)}$: $\cos(\phi(q^{(t+1)}), \phi(q^{(t)})) \geq \gamma_{min}$.

These constraints provide structured feedback E_t to the curriculum agent, ensuring a balanced exploration that is both novel and coherent.

4 Experiments

We conduct experiments to validate our core hypothesis: that intermediate pre-training (Eisenschlos et al., 2020) on our autonomously generated dataset (DQE) improves model robustness compared to existing datasets.

4.1 Training Pipeline.

Each agent in our framework is instantiated as a Qwen2.5-14B (Hui et al., 2024) model. A single step in the pipeline corresponds to processing one table from WikiTableQuestions through the multi-agent architecture illustrated in Figure 1. During each step, the agents collaboratively generate a new SQL query and its corresponding natural language question, with the curriculum designed to progressively increase task difficulty. Each tool (e.g., the SQL executor) can be invoked up to three times by the agents before the generation process terminates. After each data generation cycle, a BART-based model is pre-trained on the newly produced dataset. In the subsequent step, this updated model is reused as a controller to guide the curriculum, steering the generation toward more complex and diverse examples. For computational efficiency, we report results obtained after two iterations of this pipeline due to time and resource constraints.

4.2 Experimental Setup

Evaluation Benchmark: We evaluate our approach on the ROBUT benchmark (Zhao et al., 2023), a standard for measuring the robustness of Table QA models. ROBUT introduces various adversarial perturbations to questions and tables, such as replacing words with synonyms, reordering rows/columns, and adding distracting information.
Metrics: We use the official ROBUT metrics **Robust Accuracy**, the percentage of examples where the model is correct both before and after perturbation, measuring true robustness.
Models and Baselines. We use `bart-large` as our base model. We compare two variants: one pre-trained on the TAPEX dataset (Liu et al., 2021), a widely used TableQA benchmark, and another

pre-trained on our generated DQE dataset. We use same quantity of data as tapex for fair comparison.

4.3 Robustness Evaluation Results

As shown in Table 1, the model pre-trained on our DQE dataset consistently outperforms the TAPEX-trained baseline across nearly all perturbation types in terms of robust accuracy. The improvements are particularly notable for perturbations that target semantic understanding, such as header 'synonym' (+6.8%), header 'abbreviation' (+3.9%), and 'add'ing distractor columns (+3.6%). This suggests that the diversity of our dataset forces the model to learn deeper semantic relationships rather than relying on superficial keyword matching. Figure 3 visualizes this performance gap. It is also important to note that the TAPEX pre-training data are built around the target distribution of WikiTableQuestions, as they rely on handcrafted SQL templates from SQUALL (Shi et al., 2020). In contrast, our dataset is generated without such constraints, allowing the agent framework to explore a broader and more diverse space of question–SQL pairs.

Perturbation Type	TAPEX Robust Acc (%)	DQE Robust Acc (%)
synonym (header)	86.5	**93.3 (+6.8)**
abbreviation (header)	86.9	**90.8 (+3.9)**
word (question)	84.3	**85.2 (+0.9)**
sentence (question)	81.3	**82.4 (+1.1)**
masked (content)	**81.2**	80.6 (-0.6)
column (content)	79.1	**79.2 (+0.1)**
add (content)	72.4	**76.0 (+3.6)**
row (content)	68.8	**70.2 (+1.4)**
combined	68.5	**68.7 (+0.2)**
extend (content)	64.9	**66.1 (+1.2)**

Table 1: Robust accuracy on the ROBUT benchmark. Our model (DQE) shows significant gains over the TAPEX baseline, especially on semantic perturbations.

4.4 Dataset Analysis

To understand why our dataset yields more robust models, we analyzed its internal properties compared to TAPEX.

SQL Syntactic Diversity. We use the SQLGlot library[2] to parse each SQL query and extract its syntax tree, allowing us to analyze the distribution of node types (i.e., SQL keywords and operators). As shown in Annexe Figure 4, our DQE dataset covers a substantially broader range of SQL functionalities. DQE includes 69 unique node types compared to only 38 in TAPEX, with 34 node types exclusive to our dataset. Table 2 quantitatively summarizes these differences, demonstrating the agent framework's capacity to explore and generate more complex and diverse SQL structures.

[2] https://github.com/tobymao/sqlglot

Table 2: Comparison of SQL graph statistics between our generated dataset and TAPEX.

Metric	DQE	TAPEX	Difference
Number of query	10,000	10,000	0
Unique Node Types	69	38	31
Total Node Count	185,550	77,260	108,290
Rare Node Types ($\geq$10)	12	6	6
Common Node Types	35	35	0
Exclusive Node Types	34	3	31

Semantic Space Coverage: We projected 10,000 SQL queries from each dataset into a 2D space using UMAP on their embeddings. Figure 2 visually confirms the greater diversity of DQE. While the TAPEX samples are concentrated in a dense core, the DQE samples cover a significantly broader area, indicating that our agent-based exploration method successfully discovers more diverse regions of the semantic space.

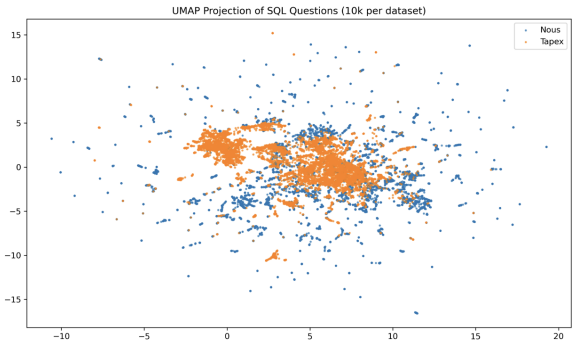

Figure 2: UMAP projection of SQL query embeddings. DQE (blue) covers a much wider semantic space than TAPEX (orange), indicating greater diversity.

5 Conclusion

We have presented a novel agent-based framework that autonomously generates a diverse, high-quality dataset for Table QA. By employing a curriculum-driven approach with distinct agents for question proposal, translation, and diversification, our system explores the SQL and natural language space more effectively than traditional static dataset creation methods. Our experiments show that pre-training on our generated data leads to models with substantially improved robustness to adversarial perturbations.

6 Limitations

While our agent-based framework shows strong potential for improving robustness and data diversity, a few limitations remain.

Adaptive Curriculum. The curriculum currently relies on heuristic similarity thresholds to enforce novelty. Future work could integrate adaptive or reinforcement-based strategies that dynamically adjust question difficulty based on the model's evolving performance.

Computational Efficiency. The sequential nature of the multi-agent pipeline increases computational cost due to repeated LLM invocations and SQL verification. Developing lighter or parallelized variants could retain diversity while improving scalability.

Evaluation Scope. Our evaluation focuses on the ROBUT benchmark. Extending experiments to additional text-to-SQL datasets (e.g., WikiSQL, Spider) would provide a more comprehensive assessment of cross-domain generalization.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2019. Tabfact: A large-scale dataset for table-based fact verification. *arXiv preprint arXiv:1909.02164*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Longxu Dou, Yan Gao, Mingyang Pan, Dingzirui Wang, Wanxiang Che, Dechen Zhan, and Jian-Guang Lou. 2023. Multispider: towards benchmarking multilingual text-to-sql semantic parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12745–12753.

Julian Martin Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. Understanding tables with intermediate pre-training. *arXiv preprint arXiv:2010.00571*.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Zhengbao Jiang, Yi Mao, Pengcheng He, Graham Neubig, and Weizhu Chen. 2022. Omnitab: Pre-training with natural and synthetic data for few-shot table-based question answering. *arXiv preprint arXiv:2207.03637*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Ce Li, Xiaofan Liu, Zhiyan Song, Ce Chi, Chen Zhao, Jingjing Yang, Zhendong Wang, Kexin Yang, Boshen Shi, Xing Wang, Chao Deng, and Junlan Feng. 2025. Treb: A comprehensive benchmark for evaluating table reasoning capabilities of large language models. *arXiv preprint arXiv:2506.18421*.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13067–13075.

Zhenwen Li and Tao Xie. 2024. Using llm to select the right sql query from candidates. *arXiv preprint arXiv:2401.02115*.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2021. Tapex: Table pre-training via learning a neural sql executor. *arXiv preprint arXiv:2107.07653*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems*, 36:36339–36348.

Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He. 2018. Data distillation: Towards omni-supervised learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4119–4128.

Ranjan Sapkota, Konstantinos I Roumeliotis, and Manoj Karkee. 2025. Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges. *arXiv preprint arXiv:2505.10468*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Peng Shi, Patrick Ng, Feng Nan, Henghui Zhu, Jun Wang, Jiarong Jiang, Alexander Hanbo Li, Rishav Chakravarti, Donald Weidner, Bing Xiang, and 1 others. 2022. Generation-focused table-based intermediate pre-training for free-form question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11312–11320.

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. *arXiv preprint arXiv:2010.11246*.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 645–654.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2019. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *arXiv preprint arXiv:1911.04942*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, and 1 others. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652.

Ori Yoran, Alon Talmor, and Jonathan Berant. 2021. Turning tables: Generating examples from semi-structured tables for endowing language models with reasoning skills. *arXiv preprint arXiv:2107.07261*.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. Grappa: Grammar-augmented pre-training for table semantic parsing. *arXiv preprint arXiv:2009.13845*.

Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. 2023. Act-sql: In-context learning for text-to-sql with automatically-generated chain-of-thought. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3501–3532.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, and 1 others. 2024. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. *arXiv preprint arXiv:2407.19669*.

Yilun Zhao, Chen Zhao, Linyong Nan, Zhenting Qi, Wenlin Zhang, Xiangru Tang, Boyu Mi, and Dragomir Radev. 2023. Robut: A systematic study of table qa robustness against human-annotated adversarial perturbations. *arXiv preprint arXiv:2306.14321*.

Wei Zhou, Mohsen Mesgar, Heike Adel, and Annemarie Friedrich. 2024. FREB-TQA: A fine-grained robustness evaluation benchmark for table question answering. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2479–2497, Mexico City, Mexico. Association for Computational Linguistics.

A Supplemental Figures

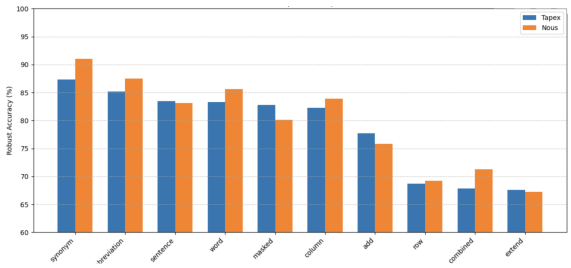

Figure 3: Robustness comparison on ROBUT. DQE consistently outperforms TAPEX.

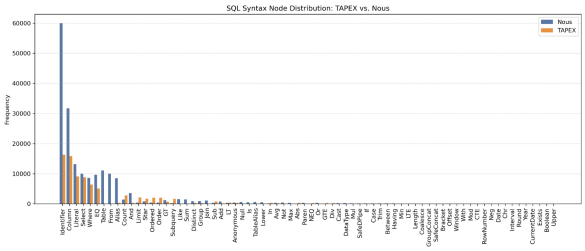

Figure 4: SQL syntax node distribution. DQE exhibits a far greater variety of SQL keywords and structures than TAPEX.