TableKV: KV Cache Compression for In-Context Table Processing

Giulio Corallo
SAP Labs, France
EURECOM, France
giulio.corallo@sap.com

Elia Faure-Rolland
EURECOM, France

Miriam Lamari
EURECOM, France

Paolo Papotti
EURECOM, France

{firstname.lastname}@eurecom.fr

Abstract

Processing large tables provided in-context to LLMs is challenging due to token limits and information overload. While Retrieval-Augmented Generation can select relevant subsets externally, this work explores Key-Value (KV) cache compression as an alternative, applied directly to the linearized table during inference. We show that the LLM's internal attention scores over the table context guides the retention of essential KV pairs, effectively compressing the processing context while preserving crucial relational information needed for complex queries. Experiments on Spider, WikitableQA, and QTSumm datasets validate the compression approach for in-context table processing, offering a promising path for improved table representation learning in LLMs.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across language tasks. A promising frontier is enabling LLMs to reason over structured data, such as tables, alongside natural language. This ability is key for applications such as table question answering (TableQA) (Chen et al., 2024) and fact-checking using relational data (Aly et al., 2021). While generating SQL queries from text (Text2SQL) is a popular approach (Yu et al., 2019), directly processing tabular data *within the LLM's context* offers a unified framework, leveraging the model's abilities to handle nuances beyond SQL's scope (Deng et al., 2024).

However, directly feeding large tables into LLMs faces significant issues. The primary challenge is *context length*: even moderately sized tables (e.g., thousands of rows and tens of columns) linearized into text easily exceed the token limits of popular models (e.g., >120,000 tokens) (Chen et al., 2024). Consequently, full-table inputs are often impractical, necessitating truncation or retrieval mechanisms (Ji et al., 2024; Badaro et al., 2023).

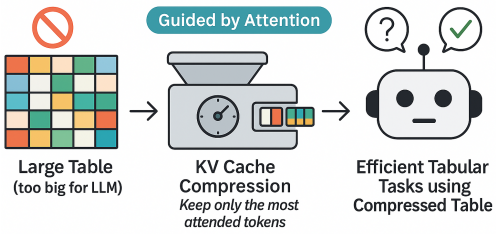


Figure 1: High-level overview of attention-guided KV compression for efficient tabular reasoning with LLMs. The model compresses the KV representation of a large table by selecting only the most attended tokens, enabling inference over a compressed table.

Even when models accommodate large contexts, their reasoning accuracy often degrades substantially (Liu et al., 2024). This phenomenon is exacerbated by tables, which inherently mix relevant cells with irrelevant information, diluting the model's attention (Sui et al., 2023; Satriani et al., 2025). Capturing relational patterns that span disparate rows or columns is particularly difficult, hindering accurate aggregation or multi-step reasoning.

Existing solutions often rely on Retrieval-Augmented Generation (RAG) (Chen et al., 2024; Lin et al., 2023). While RAG effectively reduces the input length by pre-selecting relevant table chunks (rows, columns, or cells), it introduces its own complexities. First, it requires separate retrieval modules and deciding how to optimally partition and retrieve table segments (e.g., by row, column, or semantic blocks) (Bodensohn and Binnig, 2024). Second, relying on embedding similarity for retrieval might fail to capture the fine-grained relational dependencies, shifting the bottleneck to the retriever's effectiveness.

In this paper, we explore an alternative approach: leveraging *Key-Value (KV) cache compression* techniques, originally developed for general text inference (Qin et al., 2023; Corallo and Papotti, 2024), to handle large tabular data *directly within the LLM's inference process*. Our core hypothesis is

that the LLM's own attention mechanism, as it processes the linearized table, inherently identifies the most salient information. We exploit these attention scores to dynamically prune the KV cache, retaining only the key-value pairs corresponding to the most attended-to tokens. This effectively compresses the table's representation, making the information from the original tables available *during inference*, mitigating information overload while avoiding the complexities of explicit retrieval.

Our experiments across datasets, including Spider (Yu et al., 2019), WikitableQA (Pasupat and Liang, 2015), and QTSumm (Zhao et al., 2023), demonstrate the viability of this approach.

Related Work. RAG methods retrieve relevant subsets of tabular data to reduce input complexity. TableRAG (Chen et al., 2024) employs schema and cell retrieval techniques, while TAP4LLM (Sui et al., 2023) uses sampling strategies to focus the model's attention on relevant subsets of data. Specialized encoding techniques tailored for structured inputs have also gained attention. SpreadsheetLLM (Tian et al., 2024) exploits structural redundancies within tabular data, compressing input lengths without losing semantic fidelity. However, retrieval-based methods often fall short in capturing the comprehensive relational contexts that are required to handle queries involving multiple tuples. Although KV cache compression methods (Qin et al., 2023; Corallo and Papotti, 2024) have demonstrated significant context compression by retaining subsets of relevant tokens, these techniques have yet to be adapted for structured data. This work, therefore, represents the first exploration of KV cache compression tailored to tabular inputs.

2 Background

Given a sequence of n tokens $\mathbf{x} \in \mathbb{R}^n$, each transformer layer produces hidden representations via a multi-head self-attention mechanism:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\Big(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\Big)\mathbf{V},$$

where $\mathbf{Q} = \mathbf{W}^Q\mathbf{h}, \mathbf{K} = \mathbf{W}^K\mathbf{h}, \mathbf{V} = \mathbf{W}^V\mathbf{h}$, with $\mathbf{h}$ representing the hidden states (token embeddings) for the input sequence. The dimension d_k is $\frac{d}{H}$ where d is the hidden size and H is the number of attention heads. Most LLMs organize their input as a context followed by a prompt. Let $\mathbf{x}$ denote a sequence of tokens and the input sequence: $\mathbf{x} = \Big[\ \mathbf{x}^{(\text{cont})},\ \mathbf{x}^{(\text{prompt})}\Big] \in \mathbb{R}^{n^{(\text{cont})}+n^{(\text{prompt})}}$, where $\mathbf{x}^{(\text{cont})}$ serves as the knowledge (i.e., the table) the model has access to when generating the final response. During inference, an LLM operates in two phases. In the Prefill Stage, the model processes the entire input sequence $\mathbf{x}$ and caches the KV matrices for each layer $\mathbf{K} \in \mathbb{R}^{n\times d}, \mathbf{V} \in \mathbb{R}^{n\times d}$. In the Generation Stage, for each new token y_j, the model computes autoregressively $\mathbf{q}^{\text{new}}, \mathbf{k}^{\text{new}}, \mathbf{v}^{\text{new}} \in \mathbb{R}^{1\times d}$, and updates the KV cache. With the cached KV matrices, self-attention complexity reduces from $O(n^2 d)$ to $O(nd)$. However, storing these matrices for every layer is memory intensive. To mitigate the memory load from very long contexts, one approach is *KV cache compression*. Instead of retaining $\mathbf{K}, \mathbf{V}$ for all n tokens, one compresses them into smaller matrices $\widetilde{\mathbf{K}} \in \mathbb{R}^{k\times d}$ and $\widetilde{\mathbf{V}} \in \mathbb{R}^{k\times d}$ with $k \ll n$, that preserve the information needed for generating the response, i.e., $\min_{\widetilde{\mathbf{K}},\widetilde{\mathbf{V}}}\Big[\,\text{dist}\big(\mathbf{y} \mid \mathbf{K}, \mathbf{V},\ \mathbf{y} \mid \widetilde{\mathbf{K}}, \widetilde{\mathbf{V}}\big)\Big]$, where $\mathbf{y}$ is the model's output.

To introduce compression, we detail a *query-aware* approach that compresses the KV cache by retaining only the most relevant KV vectors for the query given at inference time (Corallo and Papotti, 2024). Let m be the chunk length, and let $\{\mathbf{c}_1, \mathbf{c}_2, \dots\}$ be the segments obtained by slicing the input table $\mathbf{x}^{(\text{cont})}$. At iteration i, the method takes as input

$$\Big[\ \underbrace{\widetilde{\mathbf{K}}_{i-1}, \widetilde{\mathbf{V}}_{i-1}}_{\text{previous compressed cache}}\ ,\ \underbrace{\mathbf{c}_i}_{\text{current chunk}}\ ,\ \underbrace{\mathbf{q}}_{\text{question}}\ \Big],$$

where $\widetilde{\mathbf{K}}_{i-1}, \widetilde{\mathbf{V}}_{i-1} \in \mathbb{R}^{k\times d}$ denote the compressed cache from the previous iteration, $\mathbf{c}_i \in \mathbb{R}^{m\times d}$ is the chunk of context for the current iteration, and $\mathbf{q} \in \mathbb{R}^{q\times d}$ is the question to be answered.

During the forward pass, the multi-head attention scores are computed. The cross-attention submatrix $\mathbf{W}^{(\mathbf{q},\mathbf{c})} \in \mathbb{R}^{q\times(k+m)}$, captures how each question token attends to both the previous cache and the current chunk. The method then selects the top k token positions (according to the highest attention scores in $\mathbf{W}^{(\mathbf{q},\mathbf{c})}$) to form $\widetilde{\mathbf{K}}_i, \widetilde{\mathbf{V}}_i$. Here, k is a user-defined global budget that stays constant across iterations.

After processing all chunks, the final $\widetilde{\mathbf{K}}, \widetilde{\mathbf{V}} \in \mathbb{R}^{k\times d}$ provide a global representation of the entire context, at a reduced length. *Agnostic* methods use similar principles but in a single offline computation of the cache, thus without making use of the query. For example, Ada Expected Attention scores are computed by modeling the distribution

of queries and estimating their interaction with key vectors at future positions (Jegou et al., 2024), in conjunction with head-specific compression (Feng et al., 2025).

3 KV Compression for Tables

Handling structured data in LLMs remains a significant challenge due to the quadratic complexity of self-attention and limited context windows. Another challenge is capturing interconnections between tuples. For example, consider a table containing sales data. Answering a query such as `SELECT region, SUM(sales) FROM table GROUP BY region` requires capturing information across multiple tuples. Retrieval methods may fall short by only selecting isolated tuples or columns, missing the holistic relational context necessary for accurate aggregations.

KV cache compression, initially introduced for general LLM inference, presents a promising opportunity for tabular data. The key insight of KV compression is straightforward: after linearizing a structured table into a textual representation, standard mechanisms within transformers naturally encode relevance and information importance within attention scores. When selecting KV vectors from the cache, these vectors inherently contain latent information representing broader relational context, including information from vectors that have been evicted. Thus, rather than employing separate retrieval systems or encoding mechanisms tailored specifically for tables, we hypothesize that LLMs themselves inherently identify critical elements of linearized tables directly through attention patterns.

We explore two types of compression for linearized tabular data. *(1) Query-aware compression* dynamically compresses the cache during inference by retaining KV vectors with the highest attention scores relative to a specific question. *(2) Query-agnostic compression* pre-computes a representation of the cache independently of a specific query, capturing general information from the table.

4 Experimental Setting

Datasets. We consider three datasets. In all cases, we linearize the input table as a string with a list of lists: the first element is the table header and each subsequent sub-list is a tuple in the table. This approach outperforms or is comparable to alternative serializations. **Spider** (Yu et al., 2019) is primarily used for Text2SQL and its dev split contains 1,034 examples, each using one or more tables. We generate our ground truth by executing the SQL queries on the corresponding tables. In cases involving multiple tables, we concatenate their linearized representations sequentially, prepending the name of each table before its content. **WikitableQA** (Pasupat and Liang, 2015) and **QTSumm** (Zhao et al., 2023) focus on question answering and query-focused summarization, respectively, with answers in natural language. We use their evaluation splits. Both datasets operate on a single table at a time.

Metrics. We use different evaluation metrics depending on the task. For Spider, we assess the generated output tables with four metrics from the literature (Papicchio et al., 2023): Cell Precision and Recall, Tuple Constraint, and Execution Accuracy. For the WikitableQA dataset, we evaluate outputs with Accuracy (Pasupat and Liang, 2015). For QTSumm, we rely on ROUGE-L (Lin, 2004)

LLMs. We use **LLaMA-3.1-8B-Instruct** (Touvron et al., 2023) and **Qwen-2.5-7B** (Yang et al., 2024). Both models are used in a few-shot setting, where we prepend task-specific instructions, defining expected input and output formats along with two examples. Additionally, we enforce a fixed maximum number of output tokens (the maximum number of tokens in the ground truth), to ensure fair comparisons and prevent overly long generations.

Methods. For compression, we report results for a query-aware method, **Finch** (Corallo and Papotti, 2024), a query-agnostic one, **Ada Expected Attention**, and a **RAG** approach similar to those in (Lin et al., 2023; Sui et al., 2023). We chunk the tables into tuples and iteratively select them based on their relevance to the question, until the number of tokens aligns with the context length used in the compression methods - we use BGE-BASE-1.5-EN (Xiao et al., 2023) as embedding model. We also report results for a baseline for the full-context setting, i.e., input table without compression.[1]

5 Results

In Tables 1a and 1b, we report results for all methods on WikiTableQA and QTSumm, respectively. Based on the tables' average length in each dataset, we select different target context lengths, obtaining compression rates between 1.7x and 51.39x (average context length in Spider is 13158 tokens).

[1] We do not report results for the execution with the base model only (no tuples in the context) because of low performance, e.g., 1.80 accuracy for WikiTableQA.

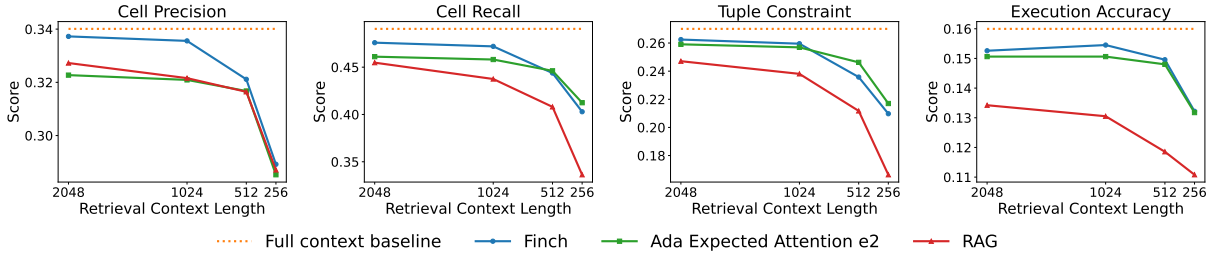
Model	Context Length	Finch	Ada EA	RAG
Llama-3.1-8B-Instruct	1024 (1.7x)	**35.11**	34.16	29.09
	512 (3.35x)	**34.92**	32.16	28.00
	256 (6.71x)	**33.59**	28.66	24.05
	128 (13.43x)	**30.02**	21.94	9.82
	Full context		35.08	
Qwen2.5-7B-Instruct	1024 (1.91x)	**29.72**	29.17	29.27
	512 (3.83x)	**28.80**	28.22	28.59
	256 (7.66x)	**27.07**	23.30	23.04
	128 (15.31x)	**23.49**	15.24	9.96
	Full context		30.04	

(a) WikiTableQA

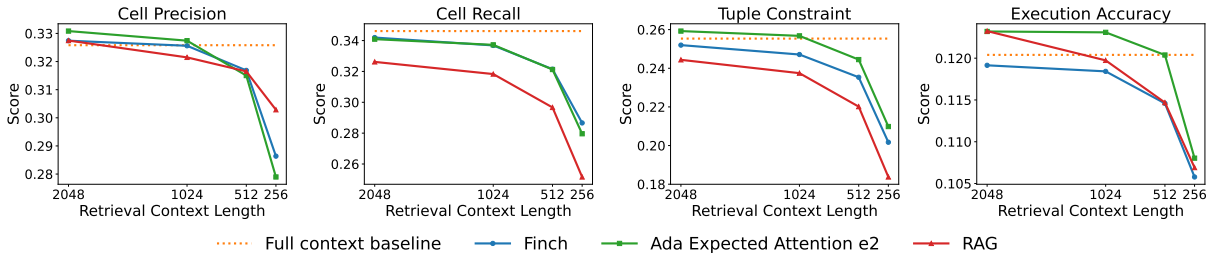
Model	Context Length	Finch	Ada EA	RAG
Llama-3.1-8B-Instruct	512 (2.5x)	**30.61**	30.59	26.67
	256 (5x)	**29.78**	29.74	24.32
	128 (10x)	**26**	25.85	19.23
	64 (20x)	**21.86**	20.75	12.14
	Full context		30.56	
Qwen2.5-7B-Instruct	512 (2.85x)	29.82	**30.01**	26.58
	256 (5.70x)	28.62	**29.39**	24.32
	128 (11.40x)	**25.75**	24.71	19.69
	64 (22.78x)	**23.54**	19.57	16.07
	Full context		29.96	

(b) QTSumm

Table 1: Performance of Finch, Ada Expected Attention, and RAG on **WikiTableQA** (left) and **QTSumm** (right) across various target context lengths; "Full context" at the bottom of each block shows the full table input result.



(a) Llama-3.1-8B-Instruct, compression rate varies between 5.66x and 45.30x.



(b) Qwen2.5-7B-Instruct, compression rate varies between 6.42x and 51.39x.

Figure 2: Performance of Finch, Ada Expected Attention, and RAG on the **Spider** dataset for two LLMs.

Overall, KV compression methods outperform the RAG-based approach in most scenarios, and in several cases, achieve better results than the full-context setup. Finch achieves strong results on WikiTableQA: with LLaMA-3.1-8B-Instruct, it obtains an Accuracy of 35.11 at a compression rate of 1.7× (1024 tokens), which is significantly higher than the other approaches, including full context. With QTSumm, compression methods yield results that are either better or very close to those of the full-context case, with compression (e.g., 30.61 for Finch with LLaMA-3.1-8B-Instruct and 30.01 for Ada with Qwen-2.5-7B).

In the Spider dataset's results in Figure 2a, query-aware compression reports promising results for Llama, outperforming the row retrieval-based strategy in all cases. Moving to Spider on Qwen in Figure 2b, RAG and Ada (agnostic) are more competitive and even surpass the full-context scenario. In this scenario, Finch reports strong performance in terms of Precision, Recall, and Tuple Constraint, but lower scores for Execution Accuracy.

In terms of execution-time, query-agnostic KV cache compression delivers faster inference than RAG, while query-aware compression similarly to full-context decoding (Corallo et al., 2025).

6 Conclusion and Future Work

This work shows that KV cache compression can outperform RAG and match full-context performance, offering a promising technique for processing tables directly within LLMs. Future research includes developing table-specific compression strategies beyond adapting existing methods and investigating the interplay between table/query complexity and compression effectiveness. Finally, we plan to examine hybrid approaches combining KV compression with lightweight retrieval.

References

Rami Aly, Zhijiang Guo, Michael Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. Feverous: Fact extraction and verification over unstructured and structured information. *Preprint*, arXiv:2106.05707.

Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. 2023. Transformers for tabular data representation: A survey of models and applications. *Trans. Assoc. Comput. Linguistics*, 11:227–249.

Jan-Micha Bodensohn and Carsten Binnig. 2024. Rethinking table retrieval from data lakes. In *Proceedings of the Seventh International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, aiDM '24, New York, NY, USA. Association for Computing Machinery.

Si-An Chen, Lesly Miculicich, Julian Eisenschlos, Zifeng Wang, Zilong Wang, Yanfei Chen, Yasuhisa Fujii, Hsuan-Tien Lin, Chen-Yu Lee, and Tomas Pfister. 2024. Tablerag: Million-token table understanding with language models. *Advances in Neural Information Processing Systems*, 37:74899–74921.

Giulio Corallo and Paolo Papotti. 2024. Finch: Prompt-guided key-value cache compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1517–1532.

Giulio Corallo, Orion Weller, Fabio Petroni, and Paolo Papotti. 2025. Beyond rag: Task-aware kv cache compression for comprehensive knowledge reasoning. *Preprint*, arXiv:2503.04973.

Naihao Deng, Zhenjie Sun, Ruiqi He, Aman Sikka, Yulong Chen, Lin Ma, Yue Zhang, and Rada Mihalcea. 2024. Tables as texts or images: Evaluating the table reasoning ability of llms and mllms. *arXiv preprint arXiv:2402.12424*.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. 2025. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *Preprint*, arXiv:2407.11550.

Simon Jegou, Maximilian Jeblick, and David Austin. 2024. kvpress.

Xingyu Ji, Aditya Parameswaran, and Madelon Hulsebos. 2024. TARGET: Benchmarking table retrieval for generative tasks. In *NeurIPS 2024 Third Table Representation Learning Workshop*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Weizhe Lin, Rexhina Blloshmi, Bill Byrne, Adria de Gispert, and Gonzalo Iglesias. 2023. An inner table retriever for robust table question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9909–9926, Toronto, Canada. Association for Computational Linguistics.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Simone Papicchio, Paolo Papotti, and Luca Cagliero. 2023. 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*.

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

Guanghui Qin, Corby Rosset, Ethan C Chau, Nikhil Rao, and Benjamin Van Durme. 2023. Dodo: Dynamic contextual compression for decoder-only lms. *arXiv preprint arXiv:2310.02409*.

Dario Satriani, Enzo Veltri, Donatello Santoro, and Paolo Papotti. 2025. Relationalfactqa: A benchmark for evaluating tabular fact retrieval from large language models. *arXiv preprint arXiv:2505.21409*.

Yuan Sui, Jiaru Zou, Mengyu Zhou, Xinyi He, Lun Du, Shi Han, and Dongmei Zhang. 2023. Tap4llm: Table provider on sampling, augmenting, and packing semi-structured data for large language model reasoning. *arXiv preprint arXiv:2312.09039*.

Yuzhang Tian, Jianbo Zhao, Haoyu Dong, Junyu Xiong, Shiyu Xia, Mengyu Zhou, Yun Lin, José Cambronero, Yeye He, Shi Han, and 1 others. 2024. Spreadsheetllm: encoding spreadsheets for large language models. *arXiv preprint arXiv:2407.09025*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

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. 2019. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *Preprint*, arXiv:1809.08887.

Yilun Zhao, Zhenting Qi, Linyong Nan, Boyu Mi, Yixin Liu, Weijin Zou, Simeng Han, Ruizhe Chen, Xiangru Tang, Yumo Xu, Dragomir Radev, and Arman Cohan. 2023. Qtsumm: Query-focused summarization over tabular data. *Preprint*, arXiv:2305.14303.