# TARGET: Benchmarking Table Retrieval for Generative Tasks

**Xingyu Ji** **Aditya Parameswaran** **Madelon Hulsebos***

**UC Berkeley**
{jixy2012,adityagp,madelon}@berkeley.edu

## Abstract

The data landscape is rich with structured data, often of high value to organizations, that drive important applications in data analysis and machine learning. Recent progress in representation learning and generative models for such data has led to the development of natural language interfaces to structured data, including those leveraging text-to-SQL. Contextualizing interactions, including conversational and agentic elements, in structured data through retrieval-augmented generation can provide substantial benefits in the form of freshness, accuracy, and comprehensiveness of answers. The key question, however, is: how do we retrieve the right table(s) for the analytical query or task at hand? To investigate this question, we introduce TARGET: a benchmark for evaluating **TA**ble **R**etrieval for **GE**nerative **T**asks. We use TARGET to analyze the retrieval performance of different retrievers in isolation, as well as their impact on downstream generators for question answering, fact verification, and text-to-SQL. We find that out-of-the-box embedding-based retrievers far outperform a BM25 baseline which appears less effective than it is for retrieval over unstructured text. We also surface the sensitivity of retrievers across various metadata (e.g., missing table titles), and illustrate a stark variation of retrieval performance across datasets and tasks. TARGET is developed for easy reuse and extension to advance research on retrieval methods and pipelines for relational data through fine-grained, comprehensive, and consistent evaluation. TARGET is available at https://target-benchmark.github.io.

## 1 Introduction

Large Language Models (LLMs) have become an indispensable tool in the knowledge worker's arsenal, providing a treasure trove of information at one's fingertips. Retrieval-Augmented Generation (RAG) [14] further extends the capabilities of these LLMs by grounding generic dialog using information from external data stores. Despite progress in long-context LLMs, RAG still provides benefits in cost and inference time [16, 25]. Moreover, it allows us to augment generic, off-the-shelf LLMs with proprietary data they haven't been trained on. Progress on RAG has largely been enabled by benchmarks that help exhaustively evaluate the effectiveness of various methods [24, 19].

While RAG has been extensively explored for free-form text, this is unfortunately not the case for structured data, stored either in relational databases or otherwise. Prior work has shown that structured data is a different story, requiring dedicated research [5]. Moreover, retrieval of structured data for RAG is important to explore further: contextualizing LLMs using frequently updated statistical data sources, such as Data Commons [9], or using proprietary relational databases, can yield rich dividends [21], all underscoring need for better models, approaches and evaluation for retrieval over structured data. Another important motivation for research on table retrieval is rooted in research on LLM-powered interfaces and agentic systems for processing and querying structured data. Most research in this direction, e.g., for question answering [20] or text-to-SQL [7], assumes that a table

*Now at CWI.

Preprint. Under review.

or relational database is provided, while identifying the relevant table is, in fact, a non-trivial task for a user (or agent). While there has been initial work exploring open-domain question answering on public table corpora such as Wikipedia [3, 10], this does not represent the full spectrum of data characteristics and tasks for structured data retrieval. The development of a broad and comprehensive benchmark covering divsere tasks and datasets of varying difficulty is therefore key in advancing retrieval systems for structured data.

We present TARGET: *the first benchmark evaluating Table Retrieval for Generative Tasks*. With TARGET we provide a consistent and comprehensive framework for evaluating models and pipelines for table retrieval in isolation, as well as end-to-end for downstream tasks. We use TARGET to analyze existing table retrievers [3], out-of-the-box LLM-based retrieval baselines [2], and industry approaches [1]. We find that BM25 is far less effective for retrieval over tabular data as it is found to be for rich free-form text [19]. In our initial exploration with TARGET, we find that out-of-the-box table embeddings with an OpenAI model [2] outperform baselines but still show high variation in performance across analytical tasks and datasets. Finally, we highlight the sensitivity of retrievers to the provided metadata inputs (e.g., web page titles) and table data availability (e.g., embedding full tables, headers only, or generated table summaries). Our findings identify a performance gap in retrieval accuracy and robustness across data and tasks, emphasizing the need for more research in this area for which TARGET can be an instrumental stepping stone.

## 2 Related Work

**Table Representation Learning and LLMs for Structured Data** Tables have recently become a modality of interest for representation learning and generative models for tasks such as table understanding [13, 6], fact verification [11, 27], and question answering [11], and more recently text-to-SQL [7]. These models either deploy LLMs out-of-the-box for tabular data, or develop tailored architectures to capture the properties of tables, which pose specific challenges [5]. These models typically take one or more tables and a query as input to generate an answer; however, the relevant tables per query can be difficult to identify. The task of open-domain question answering over tables [3, 10] has spurred research on table retrieval but existing approaches show a performance gap, as shown in Section 4. Grounding LLMs with RAG in up-to-date domain knowledge as found in (proprietary) relational databases and data lakes holds promise, but requires stronger retrievers [21].

**Benchmarks and Datasets** Benchmarks and datasets are the cornerstone to advance research on retrieval systems, as well as the corresponding LLM-driven tasks over relational data. The MTEB and CRAG benchmarks [19, 24] have been instrumental in benchmarking text embedding quality and RAG over rich text documents. We need similar benchmarks for retrieval systems and embedding models for structured data. In prior research, useful datasets were introduced to evaluate various tasks for relational data, such as TabFact [4], FeTaQA [20], GitTables [12], and Spider [26]. These datasets focus on evaluating methods for a specific downstream task only, i.e., given a table or database, answer natural language queries about it, without integrating the critical task of retrieval. TARGET addresses this gap by focusing on the evaluation of table retrieval performance while incorporating existing task-specific datasets.

## 3 The TARGET Benchmark

**Benchmark Design** The pipeline of TARGET aligns with that of typical RAG pipelines (Figure 1). TARGET takes as inputs the corpus with tables/databases and queries (e.g. a natural language question). Data loading and evaluation are abstracted away such that custom core components of RAG pipelines, i.e., the `Retriever` and `Generator` can easily be evaluated when aligned with the TARGET API (see Appendix A). The retriever, which can be basic or advanced [8], identifies the relevant table(s)/database(s) for an input query. Given the tables and query, the generator yields a response which is then evaluated with respect to the ground-truth. All resources for use and extension of TARGET are at: `https://target-benchmark.github.io`.

**Tasks, Datasets & Metrics** Per source dataset, we combine all tables and any available metadata into a retrieval corpus. For all tasks, e.g., question answering, we evaluate the retriever and generator outputs using metrics from the original papers or that are widely adopted. We use the test splits of included datasets for our evaluations. For OTTQA and BIRD, where test splits are unavailable, validation splits are used. An overview of the tasks, datasets, and metrics in the initial version of TARGET can be found in Table 1, while detailed dataset specifications are in Appendix B.

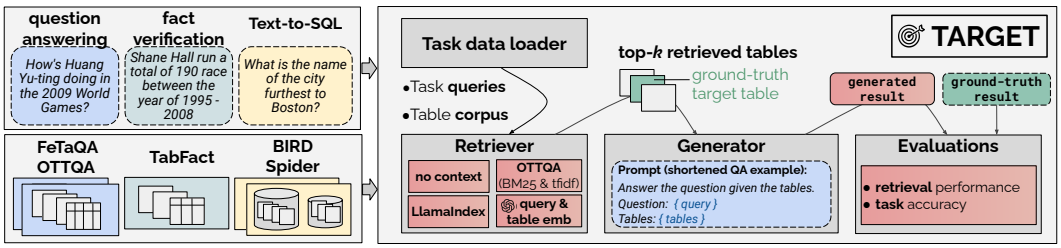


Figure 1: Overview of the TARGET benchmark for evaluating table retrieval and generation for various datasets and tasks, initially table QA, fact verification and Text-to-SQL.

Table 1: Tasks, Datasets, and Evaluation Metrics in TARGET.

| Task | Initial Datasets | Evaluation Metrics |
|---|---|---|
| Question answering | OTTQA [3], FeTaQA [20] | sacrebleu (SB) |
| Fact verification | TabFact [4] | precision (P), recall (R), f1-score (F1) |
| Text-to-SQL | Spider [26], BIRD [15] | execution accuracy (EX) |
| Table retrieval | all above datasets | recall (R@$k$), avg. retrieval time (s) |

Table Retrieval This task is used to evaluate retrieval performance in isolation and is the first step for end-to-end downstream evaluation. Retrieval performance is measured with recall@top-$k$. For Question Answering and Fact Verification tasks, a successful retrieval occurs when the ground-truth table is among the top-$k$ retrieved tables. For the text-to-SQL task, a successful retrieval occurs when the ground-truth database contains any of the retrieved tables. Additionally, we include the average retrieval time per question.

Question Answering Given the retrieved tables, an answer to the input question is generated and compared to the ground-truth natural language answer to assess its accuracy and comprehensiveness.

Fact Verification Given the retrieved tables, the generator either accepts or refutes a natural language statement, or acknowledge that insufficient information is provided.

Text-to-SQL The retrieved database tables along with the natural language question are provided to the SQL generator. The queries can be simple, medium, or difficult. The executed results from the generated SQL are compared to those of the ground-truth SQL.

**Retrievers** We present initial insights with TARGET for the following research and industry retriever baselines that reflect popular distinct methods. Text-to-SQL has a slightly different experimental setting inducing small changes to the retriever and generator, as explained in Appendix D.

No context baseline LLMs are capable of memorizing facts from the data that they were trained on [18]. To understand the influence of memorization on downstream task responses, the LLM-based generators are evaluated on the downstream task performance solely based on their internal knowledge without any retrieved tables provided, which we refer to as the "No Context" baseline.

OTTQA [3] The OTTQA retriever constructs lexical representations of tables by generating a TF-IDF matrix of the corpus, which may use TF-IDF term weights or BM25. It takes as input the table header, data rows, and, optionally, table metadata such as the (Wikipedia) page title. On retrieval, a query is converted into a TF-IDF-weighted vector for which the dot product is calculated with the table representations to find the $k$-most similar tables.

LlamaIndex [1] The retrievers of the open-source LlamaIndex library are commonly used in practice. The table retriever implements three steps, ① generate a table name and summary of each table with an LLM using the headers and first rows of the table, ② embed the table metadata with `text-embedding-ada-002` and store in an index, and ③ retrieve relevant tables based on the cosine similarity between natural language query and metadata embedding.

OpenAI embeddings [2] The input query and tables are embedded using an out-of-the-box OpenAI embedding model (`text-embedding-3-small`). We evaluate the performance for embeddings of only table headers versus headers along with 100 rows, which are formatted as a markdown string[2]. The embeddings are stored in an HNSW index[17], often used as backbone for vector stores. On retrieval, the input query is embedded with the same model after which the top-$k$ table embeddings are retrieved based on highest cosine-similarity with the input query embedding.

[2]Formatting tables as json appeared better for GPT-3.5 [22], but markdown formatting yields better results here.

Table 2: Results with TARGET for retrieval and downstream tasks. SB stands for sacrebleu, R@$k$ for recall@$k$, s for seconds, EX for execution accuracy. Best scores are in **bold**, second-best underlined.

| | Question Answering | | | | | | Fact Verification | | | Text-to-SQL | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OTTQA | | | FeTaQA | | | TabFact | | | Spider | | | BIRD | | |
| **Method** | R@10 | s | SB | R@10 | s | SB | R@10 | s | P/R/F1 | R@1 | s | EX | R@1 | s | EX |
| No context | - | - | 0.414 | - | - | 12.495 | - | - | 0.578/0.42/0.44 | - | - | 0 | - | - | 0 |
| OTT-QA BM25 | **0.955** | 0.001 | 0.606 | 0.082 | 0.001 | 1.631 | 0.338 | 0.001 | 0.75/0.26/0.39 | 0.635 | 0.001 | 0.385 | 0.709 | 0.001 | 0.181 |
| *w/o table title* | 0.443 | 0.001 | 0.529 | 0.084 | 0.001 | 1.555 | 0.331 | 0.001 | 0.75/0.26/0.38 | 0.5 | 0.001 | 0.376 | 0.535 | 0.001 | 0.164 |
| OTT-QA TF-IDF | 0.950 | 0.001 | 0.425 | 0.083 | 0.001 | 1.639 | 0.336 | 0.001 | 0.75/0.26/0.38 | 0.622 | 0.001 | 0.474 | 0.640 | 0.001 | 0.227 |
| *w/o table title* | 0.43 | 0.001 | 0.593 | 0.083 | 0.001 | 1.527 | 0.322 | 0.001 | 0.75/0.25/0.37 | 0.492 | 0.001 | 0.376 | 0.491 | 0.001 | 0.164 |
| LlamaIndex | 0.458 | 0.354 | 0.507 | 0.435 | 0.396 | 13.745 | **0.827** | 0.297 | 0.73/0.34/0.47 | 0.735 | 0.198 | 0.559 | 0.937 | 0.228 | 0.311 |
| OpenAI embedding | 0.950 | 0.190 | 0.599 | **0.722** | 0.200 | 17.64 | 0.779 | 0.189 | 0.76/0.51/0.61 | 0.768 | 0.193 | 0.602 | 0.926 | 0.199 | 0.317 |
| *header only* | 0.950 | 0.189 | 0.61 | 0.718 | 0.18 | 17.66 | 0.781 | 0.187 | 0.75/0.48/0.58 | **0.833** | 0.175 | 0.646 | **0.958** | 0.191 | 0.323 |

**Generators** We integrate generators based on basic LLM prompts for downstream tasks. The `Instruction` prompt takes in: ① task instructions, ② the top-$k$ retrieved table(s) or database schemas, and ③ the query. All prompt templates can be found in Appendix D.

## 4 Results

Table 2 presents the performances of the evaluated retrievers. Here, we set the retrieval parameter $k$ to 10 except for text-to-SQL, where $k$ is 1 for database retrieval (Section C). Figure 2 illustrates the average retrieval recall over various values of $k$ across datasets. The TF-IDF based method from the OTTQA retriever is not included as its performance is similar to the BM25 based method.

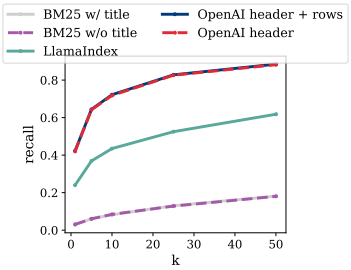


Figure 2: Influence of $k$ on retrieval performance.

**Retrieval insights** We observe a gradual increase in average recall as $k$ value rises for all retrievers, as can be expected. We find that lexical retrieval methods based on BM25 and TF-IDF are less effective than they are for text [19], even with increased $k$. The high performance of these methods on the OTTQA dataset appears to be mainly driven by the high correspondence between Wikipedia table title and queries, as the performance drops when the title is left out. We observe a similar pattern for these methods on the text-to-SQL tasks when table names are not included, which is further confirmed with results on FeTaQA, where the table titles are not descriptive and including them does not enhance performance. These findings emphasize the potential critical role of table metadata. Embeddings of table headers and rows generally yield the best performance. LLM-generated table summaries with LlamaIndex results in lower retrieval performance and efficiency than the direct table embedding pipeline, but generating descriptive table titles in place of non-descriptive ones (e.g., FeTaQA) can enhance retrieval performance. For both text-to-SQL datasets, including data rows in the embedding actually lowers retrieval performance.

**Generator insights** Unsurprisingly, we observe that providing database schemas for text-to-SQL is critical to generate accurate SQL queries, as the No Context baseline yields an accuracy of 0. The low performance of all retrievers on the OTTQA dataset is also notable, which we hypothesize is due to the relatively short answers in OTTQA versus longer generated answers despite prompting for conciseness. Overall, we find that dense embeddings yield better retrieval performance. Notably, for the fact verification task, the precision and recall with OpenAI embeddings are significantly higher than when evaluating the statements without context, i.e., using only the memory of the LLM, underlining the value of grounding LLMs conversations in factual structured data. When we exclude all "not enough information" responses, we find that the recall across all retrievers increases to approximately 0.747, which confirms the impact of incorporating relevant tables into the context.

## 5 Conclusion

Retrieval is key in LLM-powered query systems over structured data in relational databases and other data systems, as well as for grounding dialog and interactions with LLMs in up-to-date, factual data. With both categories of use-cases in mind, we present TARGET: the first benchmark for Table Retrieval for Generative Tasks. Beyond our initial evaluation that has already provided valuable insights, we are expanding it in the form of more datasets, tasks, and retrievers, as well as metrics for downstream task evaluations. Once complete, we will conduct fine-grained error analysis to identify patterns in cases of incorrect retrieval and the relationship between retrieval and downstream tasks. In future work, we plan to explore, for example, the relationship between retrieval and downstream performance and error cases, to inform future research on table retrieval for generative tasks.

## 6 Acknowledgements

We acknowledge support from the National Science Foundation (grants DGE-2243822, IIS-2129008, IIS-1940759, IIS-1940757, 1845638, 1740305, 2008295, 2106197, 2103794, 2312991), the Dutch Research Council (NWO) grant NGF.1607.22.045, a BIDS Accenture Fellowship, and EPIC lab sponsors (G-Research, Adobe, Microsoft, Google).

## References

[1] Llamaindex. https://www.llamaindex.ai/. Accessed: 09/16/2024.

[2] T. B. Brown, B. Mann, N. Ryder, and e. a. Subbiah, Melanie. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[3] W. Chen, M.-W. Chang, E. Schlinger, W. Y. Wang, and W. W. Cohen. Open question answering over tables and text. In *International Conference on Learning Representations*, 2021.

[4] W. Chen, H. Wang, J. Chen, Y. Zhang, H. Wang, S. Li, X. Zhou, and W. Y. Wang. TabFact: A large-scale dataset for table-based fact verification. *International Conference of Learning Representations*, 2020.

[5] T. Cong, M. Hulsebos, Z. Sun, P. Groth, and H. V. Jagadish. Observatory: Characterizing embeddings of relational tables. *Proceedings of VLDB*, 17(4), 2023.

[6] X. Deng, H. Sun, A. Lees, Y. Wu, and C. Yu. Turl: Table understanding through representation learning. *ACM SIGMOD Record*, 51(1), 2022.

[7] D. Gao, H. Wang, Y. Li, X. Sun, and et al. Text-to-sql empowered by large language models: A benchmark evaluation. *Proceedings of VLDB*, 2024.

[8] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[9] R. V. Guha, P. Radhakrishnan, B. Xu, W. Sun, C. Au, A. Tirumali, M. J. Amjad, S. Piekos, N. Diaz, J. Chen, et al. Data commons. *arXiv preprint arXiv:2309.13054*, 2023.

[10] J. Herzig, T. Mueller, S. Krichene, and J. Eisenschlos. Open domain question answering over tables via dense retrieval. In *Proceedings of NAACL*, 2021.

[11] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. M. Eisenschlos. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*, 2020.

[12] M. Hulsebos, Ç. Demiralp, and P. Groth. Gittables: A large-scale corpus of relational tables. *Proceedings of the ACM on Management of Data*, 1(1), 2023.

[13] M. Hulsebos, K. Hu, M. Bakker, E. Zgraggen, A. Satyanarayan, T. Kraska, Ç. Demiralp, and C. Hidalgo. Sherlock: A deep learning approach to semantic data type detection. In *Proceedings of the 25th ACM SIGKDD*, 2019.

[14] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33, 2020.

[15] J. Li, B. Hui, G. Qu, J. Yang, B. Li, B. Li, B. Wang, B. Qin, R. Geng, N. Huo, et al. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36, 2024.

[16] Z. Li, C. Li, M. Zhang, Q. Mei, and M. Bendersky. Retrieval augmented generation or long-context llms? a comprehensive study and hybrid approach. *arXiv preprint arXiv:2407.16833*, 2024.

[17] Y. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4), 2018.

[18] A. T. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

[19] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022.

[20] L. Nan, C. Hsieh, Z. Mao, X. V. Lin, N. Verma, R. Zhang, W. Kryściński, H. Schoelkopf, R. Kong, X. Tang, et al. Fetaqa: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10, 2022.

[21] P. Radhakrishnan, J. Chen, B. Xu, P. Ramaswami, H. Pho, A. Olmos, J. Manyika, and R. Guha. Knowing when to ask-bridging large language models and data. *Data Commons*, 2024.

[22] A. Singha, J. Cambronero, S. Gulwani, V. Le, and C. Parnin. Tabular representation, noisy operators, and impacts on table structure understanding tasks in llms. *Table Representation Learning workshop at NeurIPS*, 2023.

[23] S. Talaei, M. Pourreza, Y.-C. Chang, A. Mirhoseini, and A. Saberi. Chess: Contextual harnessing for efficient sql synthesis. *arXiv preprint arXiv:2405.16755*, 2024.

[24] X. Yang, K. Sun, H. Xin, Y. Sun, N. Bhalla, X. Chen, S. Choudhary, R. D. Gui, Z. W. Jiang, Z. Jiang, et al. Crag–comprehensive rag benchmark. *arXiv preprint arXiv:2406.04744*, 2024.

[25] T. Yu, A. Xu, and R. Akkiraju. In defense of rag in the era of long-context language models. *arXiv preprint arXiv:2409.01666*, 2024.

[26] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*, 2018.

[27] H. Zhang, Y. Wang, S. Wang, X. Cao, F. Zhang, and Z. Wang. Table fact verification with structure-aware transformer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1624–1629, 2020.

[28] A. Zhuang, G. Zhang, T. Zheng, X. Du, J. Wang, W. Ren, S. W. Huang, J. Fu, X. Yue, and W. Chen. Structlm: Towards building generalist models for structured knowledge grounding. *arXiv preprint arXiv:2402.16671*, 2024.

# Appendix

## A TARGET API

TARGET has been developed with ease in reuse and extensibility in mind. For example, running evaluations only requires instantiating a `TARGET` evaluator for a specified task and dataset, a `Retriever` instance (e.g., a built-in retriever), and running the evaluation with a value for $k$ and the desired dataset split, as shown below for the LlamaIndex Retriever for the Question Answering task. Data handling and task evaluations are abstracted away such that the core components of RAG pipelines, i.e., the `Retriever` and `Generator` can be easily evaluated. Custom retrievers can be created by implementing just 2 methods – `embed_corpus(corpus)` and `retrieve(query, top_k)`, which can include advanced RAG components like query enhancement, re-ranking [8].

```
from target_benchmark.evaluators import TARGET
from target_benchmark.retrievers import LlamaIndexRetriever

target    = TARGET(("Table Question Answering", "OTTQA"))
retriever = LlamaIndexRetriever()
results   = target.run(retriever, split="test", top_k=10)
```

Table 3: Specifications of current datasets in TARGET.

| Task | Dataset | Split | Corpus size | Number queries |
|---|---|---|---|---|
| Question Answering | OTTQA | train | 8.1k tables | 41.5k |
| | | validation | 789 tables | 2.2k |
| | FeTaQA | train | 7.3k tables | 7.3K |
| | | validation | 1K tables | 1k |
| | | test | 2K tables | 2k |
| Fact Verification | TabFact | train | 13.2K tables | 92.3K |
| | | validation | 1.7K tables | 12.8k |
| | | test | 1.7K tables | 12.8k |
| Text-to-SQL | BIRD | validation | 11 DBs (75 tables) | 1.5K |
| | Spider | train | 146 DBs (2K tables) | 8.7K |
| | | validation | 20 DBs (1K tables) | 1k |
| | | test | 40 DBs (1K tables) | 2.1k |

## B Dataset statistics

The available datasets in TARGET can be found in Table 3. All publicly available splits of each datasets are included except for BIRD's train split. To ensure consistency across datasets, we standardized their structures by aligning the column headers and component formats for all datasets. In addition to table contents and table identifiers, each corpus entry includes a "context" field for any metadata or relevant information to the table, if available. For instance, in the text-to-SQL datasets, the context field contains the primary key, foreign key, and other table schema information. Although BIRD's validation split contains fewer tables and databases, the large size of each table poses a significant challenge for retrieval systems. Specifically, the average number of rows per table in BIRD's validation corpus is 52.4k, compared to 5.3k rows per table in Spider's corpus.

## C Text-to-SQL specification

The text-to-SQL task has a slightly modified setup for retrievers as it involves database retrieval as well as in-database schema and table retrieval, which poses a novel challenge. As a baseline, retrievers first retrieve the tables, and find the database(s) that these tables belong to. TARGET's default generator currently take all table headers within the retrieved database, but more advanced retrievers and custom generators can extend this with in-database schema/table filtering.

## D Generator Prompts

**Question Answering** These prompts were adapted from [28].

System Prompt:

```
You are a data analyst who reads tables to answer questions.
```

Instruction Prompt:

```
Use the provided table(s) to answer the question.  Yield a concise
answer to the question.

If the tables cannot be used to answer the question, say that not
enough information is provided.

Tables:  {table_contents}
Question:  {query}
```

**Fact Verification** These prompts were adapted from [28].

System Prompt:

```
You are an expert in evaluating statements on factuality given the
provided tables.
```

Instruction Prompt:

```
Given the following evidence which may take the form of sentences or
a data table, determine whether the evidence supports or refutes the
following statement, or does not contain enough information.

Assign the statement one of three labels: True, False, Not Enough
Information. Do not include anything else in your answer.

Tables: {table_contents}
Statement: {query}
```

**Text-to-SQL Prompts** These prompts were adapted from CHESS [23].

In order to ensure the generated SQL query can be easily parsed from the generator's response (which includes both Chain of Thought Reasoning and the generated SQL), we use LangChain's `StructuredOutputParser` to enforce output in JSON format. The `{format_instructions}` parameter in the user prompt describes the expected output format to the generator.

System Prompt:

```
You are an expert and very smart data analyst.
```

Instruction Prompt:

```
Below, you are presented with a database schema and a question.

Your task is to read the schema, understand the question, and generate
a valid SQLite query to answer the question.

Before generating the final SQL query, think step by step on how to
write the query.

Database Schema: {database_schema}
This schema offers an in-depth description of the database's
architecture, detailing tables, columns, primary keys, foreign keys,
and any pertinent information regarding relationships or constraints.

Question: {query}

Please respond with a paragraph structured as follows: {instructions}

Take a deep breath and think step by step to find the correct SQLite
SQL query. If you follow all the instructions and generate the
correct query, I will give you 1 million dollars.
```

**No Context Instruction** For "No Context" evaluations, we provide the following message to the generator in place of the {table_contents} field in the instruction prompts.

```
Some or all tables are not available. Please use your best judgment
to complete the task. DO NOT RESPOND with "not enough information" or
similar answers, and don't acknowledge the lack of information in your
response. Just use your knowledge base and answer to the best of your
ability.
```