OrQA – OpenData Retrieval for Question Answering Dataset Generation

Giovanni Malaguti
University of Modena
and Reggio Emilia, Italy
giovanni.malaguti@unimore.it

Angelo Mozzillo
University of Modena
and Reggio Emilia, Italy
angelo.mozzillo@unimore.it

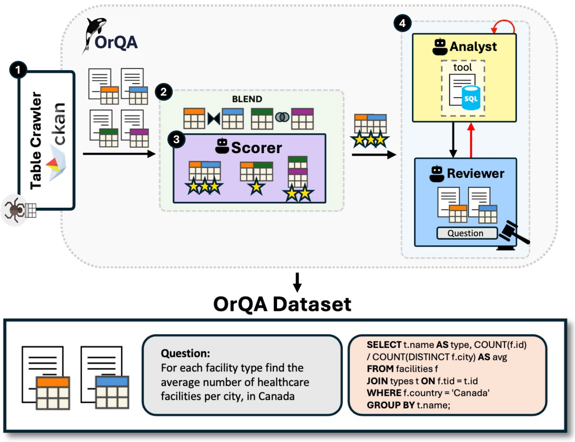
Giovanni Simonini
University of Modena
and Reggio Emilia, Italy
giovanni.simonini@unimore.it

Abstract

We present OrQA, a novel agentic framework to generate large-scale tabular question-answering (TQA) datasets based on real-world open data. Such datasets are needed to overcome the limitations of existing benchmark datasets, which rely on synthetic questions or limited web tables. OrQA employs LLM agents to retrieve related open data tables, generate natural questions, and synthesize executable SQL queries—involving joins, unions, and other non-trivial operations. By leveraging hundreds of GPU hours on four NVIDIA A100, we applied OrQA to Canadian and UK government open data to produce 1,000 question-tables–SQL triples, a representative sample of which has been human-validated. This open-source dataset is now publicly available to drive transparency, reproducibility, and progress in table-based question answering.

1 Introduction

The Open Data initiative aims to ensure transparency and foster informed civic engagement—e.g., for accessing data related to public policy outcomes or monitoring phenomena of interest. Such initiatives have significantly increased the availability of publicly accessible tabular and structured datasets, often referred to as open data lakes, many of which are accessible through web portals that facilitate discovery and reuse. However, these open datasets are typically published with highly heterogeneous schemas, making their integration into structured relational databases challenging. As a result, identifying tables that can be meaningfully joined or unioned remains a difficult task, limiting the ability to extract comprehensive insights across multiple datasets. Furthermore, to fully democratize the access to open data a Tabular Question Answering (TQA) approach is desirable, allowing users to issue queries through natural language interfaces, removing technical barriers for the user, such as query languages and data schema understanding.



Question:
For each facility type find the average number of healthcare facilities per city, in Canada

```
SELECT t.name AS type, COUNT(f.id)
/ COUNT(DISTINCT f.city) AS avg
FROM facilities f
JOIN types t ON f.tid = t.id
WHERE f.country = 'Canada'
GROUP BY t.name;
```

Figure 1: OrQA workflow: 1) Crawl tables; 2) Join/Union table-pairs discovery; 3) pairs scoring with the Scorer agent; 4) Analyst and Reviewer agents generate SQL and NL questions.

TQA has emerged as a crucial task in natural language processing, enabling models to answer questions using tabular data (Zhu et al., 2024). TQA tasks can be divided into two main categories: *(i)* the older one, fine-tuning specialized models tailored specifically for this task (Herzig et al., 2020; Yin et al., 2020; Liu et al., 2022; Zhou et al., 2022); *(ii)* the newer one, utilizes LLMs to generate code capable of manipulating tabular data (Yin et al., 2023; Liu et al., 2024; Zhang et al., 2024). While these new LLM-based approaches have shown impressive performance in reasoning over a single table—where all pertinent information is self-contained—they often struggle in more complex scenarios that require reasoning across multiple tables, including operations such as joins and unions, which are essential for handling real-world data (Zhu et al., 2024). Moreover, although LLMs have demonstrated robust capabilities across various natural language tasks, their evaluation has largely been confined to QA datasets derived from small, web-based tables (Pasupat and Liang, 2015; Iyyer et al., 2017; Zhong et al., 2017; Nan et al., 2022). This limitation arises from two key challenges. First, real-world multi-table datasets are not widely available, as many remain private due to confidentiality concerns (Hulsebos et al., 2023; Vogel et al., 2024). Second, dataset creation has traditionally relied on crowdsourcing, which, while effective, is slow, expensive, and difficult to scale (Long et al., 2024).

LLMs have shown great potential to generate synthetic datasets, providing a scalable alternative to costly human annotation. They can create diverse training data that better reflects real-world challenges, which is critical for model development (Long et al., 2024). As a pivotal application of LLMs, synthetic data generation holds significant importance for the development of new LLMs (Long et al., 2024). As of April 2025, over 519 tabular datasets on Hugging Face are labeled as *synthetic*[1] and have been employed for fine-tuning or reinforcement learning applications (Guo et al., 2025). Yet, ensuring both high accuracy and sufficient variety in these datasets is challenging. Thus, careful design and specific techniques are required to guide the generation process toward the desired outcomes.

Our Contributions

We present the **O**pen Data **r**etrieval and **Q**uestion **A**nswering (OrQA)[2] datasets generation workflow, designed to create large-scale and completely new datasets for end-to-end TQA evaluation using tabular content from Open Data sources. We also present a dataset generated with OrQA, which covers tables obtained from the Open Data portals of Canada[3] and UK[4]. The dataset includes questions expressed in natural language, each of which is associated with: *(i)* the table or set of tables containing the required information (useful for evaluating the retrieval phase of a TQA system); *(ii)* the SQL query to obtain the answer from the table(s) (useful for evaluating the generation phase of a TQA system); *(iii)* a set of statistics for analysis and inspections.

We built OrQA by designing an agentic workflow that exploits state-of-the-art data discovery techniques to select high-quality joinable and unionable tables, which are employed as seeds for generating synthetic pairs of natural language questions and SQL queries with LLMs agents—as described in the following.

2 OrQA Overview

The OrQA workflow is designed to be easily applied to any Open Data portal and allows the user to create a new dataset given a specific Open Data endpoint. OrQA is composed of four main steps, listed hereafter and explained afterward:

1. *Data Crawling*: to download both tables and metadata from a given Open Data endpoint;
2. *Candidate Table Pair Search*: to yield candidate pairs of related tables discovered through data discovery tool;
3. *Candidate Evaluation*: to evaluate the candidate table pairs with a multi-agent debate mechanism to filter casual and unmeaningful cases;
4. *Question Generation*: the accepted pairs are used as input to create the final dataset.

Data Crawling. During the first step, the user specifies the Open Data endpoint of interest exposing CKAN API [5], and from there, tables and relative metadata are downloaded and stored for the next steps.

Candidate Table Pair Search. Data discovery algorithms from the BLEND framework (Esmailoghli et al., 2024) are applied to identify candidate pairs of related tables, which could be merged with a `join` or `union` operation. BLEND is a general-purpose framework for table discovery in data lakes; after an indexing stage of the available tables, it can efficiently retrieve results, based on overlap metrics, related to a given query table. In the OrQA workflow, each column of every table is used as an input seed for BLEND, which returns K candidate tables. This search could be limited up to a user-specified budget. In initial experiments, we observed that filtering less informative columns was necessary to reduce noise Thus, for the datasets generated for this paper, we filtered out columns with less than 10 unique values and 30 rows, or more than 80% of missing values.

Candidate Evaluation. An agentic step is performed to assess for each candidate pair whether the two tables are meaningfully related or not. A team of AI agents assigns a numerical relatedness

[1] https://huggingface.co/datasets?modality=modality:tabular&other=synthetic

[2] https://github.com/dbmodena/orqa

[3] https://open.canada.ca

[4] https://www.data.gov.uk/

[5] https://docs.ckan.org/en/latest

score θ	over θ pairs	
	UK	CAN
6	952	807
7	938	755
8	902	551
9	160	256
10	0	0

Table 1: Candidate join pairs evaluated with a score equal of higher than the threshold, on a sample of 1000 pairs.

score (on a scale from 0 to 10, the highest being the most related) to each pair, using a description of each table, a small sample of rows, and other available metadata obtained from Open Data portals. Only pairs that achieve a score higher than a predefined threshold θ are retained for the final generation phase.

To assess this scoring system, user feedbacks were collected over a sample of 100 random candidate pairs from the UK dataset. The results showed an average difference of just 0.43 between human and agent team scores, with a standard deviation of 1.45, p-value 0.0038. These findings suggest that the agent-based evaluation scoring appears to be comparable to the user's when assessing how much a couple of tables is related to limited information and domain knowledge. In our experiments, we set the minimum score θ to 8—as seen in Table 1—which significantly reduces the total number of pairs that need to be processed in the final computational step.

Question Generation. An agentic workflow is implemented to generate the final output: a team of agents—composed of a natural language question generator, a text-to-SQL coder, and relative reviewers—is responsible for creating both an SQL query and the corresponding natural language question. For each pair of unionable or joinable tables validated in the previous steps, the team of agents produces queries and questions for single and multi-table cases. The coder agent receives in input the tables' metadata, a sample of their rows, and the other specifications for the current task; then it generates an SQL query, verifying its syntax through a dedicated tool. Once the query is created, the question generator agent outputs a natural language question that accurately represents the query's intent. In both these previously described stages, a reviewer evaluates the generated output: until specific requirements are not satisfied, it asks the relative generator agent to refine its output, providing suggestions for improvement. To prevent excessively long computations when the generator agent repeatedly fails, a maximum number of reviews is set. In every natural language question, it is ensured that useful references for retrieval tasks on the Open Data portals are inserted—such as remainders to significant keywords or to the organization that created the resource. We applied OrQA with a maximum of 3 review cycles, a choice that balances efficiency and accuracy, as additional cycles yielded diminishing returns. Additionally, given that many tables may contain a large number of columns, we restrict the agent's context to the first to the first 20 columns—appending any necessary columns as required. This assumes the first 20 columns contain enough information to generate meaningful queries.

source	tables	# rows		# columns	
		avg	stdev	avg	stdev
UK	24404	22747	174497	52	628
CAN	31437	141714	1405456	17	168

Table 2: Statistics of the crawled tables.

difficulty	type	#queries
simple	single-table	208
	multi-table	104
moderate	single-table	272
	multi-table	97
challenging	single-table	236
	multi-table	83

Table 3: Generated queries per difficult level and type.

3 Generated Dataset

By employing OrQA, we generated a dataset consisting of 1,000 natural language questions and corresponding ground truth, derived from both UK and Canada (CAN) open data portals—Table 2 reports statistics collected from these portals.

To generate the dataset, we employed a GPU node equipped with 4 NVIDIA A100 GPUs, each of them with 40 GB of memory. We opted for Qwen2.5 (Yang et al., 2025) family models as LLMs for the evaluation and generation steps. In particular, we used Qwen2.5-7b for the evaluation team agents and Qwen2.5-32b and Qwen2.5-coder-32b for the Natural Language and SQL generation agents, respectively. With this setup, the creation of the dataset from data crawling to the generation of the final questions required almost 100 hours of total computation, with a large part of these dedicated to crawling, indexing and candidate search.

Following (Li et al., 2024), we divided SQL queries into three main categories, *simple*, *moderate* and *challenging*, specifying to the coder agent for each category what we expect, from simple filtering clauses to window functions, grouping and

measure	group	2013	2016
canada child benefit	refundable tax credit classified as transfer payment	""	"16860"
employee benefit plans	tax expenditure	"n.a."	"n.a."
logging tax credit	tax expenditure	"15"	"25"

Table 4: Example rows from one Open Data table. The columns "2013" and "2014" are not recognized by default as numeric columns.

```sql
SELECT measure, \"group\", SUM(
    CASE WHEN regexp_matches(
            \"2013\", '^\\d+$')
        THEN CAST(\"2013\" AS INTEGER)
        ELSE 0 END
    ) AS total_2013
FROM r_df GROUP BY measure, \"group\"
```

Listing 1: Example query with data wrangling operation. label

subqueries. Table 3 reports distributions of the difficult levels for the generated collection. In addition to the question-query pairs, we provide several metadata, which are valuable for future evaluations of workflow efficiency. These include the number of review, the time taken to generate both SQL and natural language queries, and the number of tokens exchanged by the underlying LLMs. We also report the details of the final review and, in cases of SQL generation failure, the error message from the database engine to facilitate failure analysis.

Online Data Wrangling. One burdening challenge when working with Open Data is to extract meaningful information while facing data-wrangling issues. In OrQA, the agent team itself—in particular the pair of coder and code reviewer—attempts to dynamically wrangle the desired columns during query generation. As an example, the table 4 contains the columns “2013” and “2014”, whose values are initially recognized as strings, due to the presence of missing values (like *"n.a."*) and the empty string *""* in the same column. By interacting and analyzing the query output, the agents are able to generate an SQL query that solves this issue, as shown in listing 1.

4 Related Work

Text-to-SQL and Fact verification are well-known topics in the literature, and several datasets have been proposed and tested across different systems and scenarios. Notable among these are Spider (Lei et al., 2025) and BIRD (Li et al., 2024), two comprehensive Text-to-SQL benchmark datasets with a wide range of difficult tasks based on real-world data. However, they assume that the tables or databases where needed information is stored are already provided. As Retrieval Augmented Generation (RAG) systems gain relevance, there is the need to address the retrieval phase with dedicated tabular benchmark datasets. While datasets such as CRAG (Yang et al., 2024) and MTEB (Muennighoff et al., 2023) focus on text embedding, TARGET (Ji et al., 2024) represents a first step toward benchmarking table retrieval. Its evaluates model performance using embedding-based retrieval systems, assuming that tables are totally indexed. However, in many real-world scenarios with limited resources, making a complete pre-indexing stage is unfeasible. Open Data are a significant example of this case: their dynamic nature and scale make it difficult to incorporate them into static datasets, but their content could address several types of use-cases. Final users, such as public administrations or private citizens, typically need to extract information from them without performing large computations. Although Open Data have been widely used in previous years in the data discovery literature, prior work has not focused on downstream tasks, only on finding related tables. In particular, LakeBench (Deng et al., 2024) is a benchmark dataset for joinable and unionable table discovery methods, which limits its scope to identify subsets of related results given a query table. Like OrQA, during benchmark preparation it uses established data discovery tools to generate candidate pairs of related tables, but in that case a large human effort is used to evaluate them, while in OrQA this is fully automated.

5 Conclusion and Future Work

We present OrQA, an agentic workflow to generate new datasets for retrieval and question-answering model evaluation based on Open Data tables. With OrQA, we generated a dataset composed of 1,000 questions, which can be employed as realistic benchmark for RAG systems targeting TQA on Open Data.

We believe that our effort paves the way for further research, since several open challenges have not yet been addressed. For instance, semantic-aware data discovery tools could provide more interesting candidates for question generation. Additionally, the current workflow covers only questions that involve one or two tables, while users’ needs may require more complex patterns. Furthermore, different datasets might correctly address the same question and should be considered in the ground truth.

Acknowledgments

We warmly thank Leonardo S.p.A., co-funding the PhD programs of Giovanni Malaguti and Angelo Mozzillo. Further, this work was partially supported by MUR within the project "Discount Quality for Responsible Data Science: Human-in-the-Loop for Quality Data" (code 202248FWFS).

References

Yuhao Deng, Chengliang Chai, Lei Cao, Qin Yuan, Siyuan Chen, Yanrui Yu, Zhaoze Sun, Junyi Wang, Jiajun Li, Ziqi Cao, Kaisen Jin, Chi Zhang, Yuqing Jiang, Yuanfang Zhang, Yuping Wang, Ye Yuan, Guoren Wang, and Nan Tang. 2024. Lakebench: A benchmark for discovering joinable and unionable tables in data lakes. *Proc. VLDB Endow.*, 17(8):1925–1938.

Mahdi Esmailoghli, Christoph Schnell, Renée J. Miller, and Ziawasch Abedjan. 2024. Blend: A unified data discovery system. *Preprint*, arXiv:2310.02656.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4320–4333. Association for Computational Linguistics.

Madelon Hulsebos, Çagatay Demiralp, and Paul Groth. 2023. Gittables: A large-scale corpus of relational tables. *Proc. ACM Manag. Data*, 1(1).

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831.

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

Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. 2025. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. *Preprint*, arXiv:2411.07763.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, and 1 others. 2024. 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.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022. TAPEX: table pre-training via learning a neural SQL executor. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Tianyang Liu, Fei Wang, and Muhao Chen. 2024. Rethinking tabular data understanding with large language models. 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 450–482, Mexico City, Mexico. Association for Computational Linguistics.

Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. On LLMs-driven synthetic data generation, curation, and evaluation: A survey. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11065–11082, Bangkok, Thailand. Association for Computational Linguistics.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark. *Preprint*, arXiv:2210.07316.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, and 1 others. 2022. Fetaqa: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480.

Liane Vogel, Jan-Micha Bodensohn, and Carsten Binnig. 2024. Wikidbs: A large-scale corpus of relational databases from wikidata. *Advances in Neural Information Processing Systems*, 37:41186–41201.

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 23 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, Lingkun Kong, Brian Moran, Jiaqi Wang, Yifan Ethan Xu, An Yan, Chenyu Yang, Eting Yuan, Hanwen Zha, Nan Tang, and 8 others. 2024. Crag – comprehensive rag benchmark. *arXiv preprint arXiv:2406.04744*.

Pengcheng Yin, Wen-Ding Li, Kefan Xiao, Abhishek Rao, Yeming Wen, Kensen Shi, Joshua Howland, Paige Bailey, Michele Catasta, Henryk Michalewski, Oleksandr Polozov, and Charles Sutton. 2023. Natural language to code generation in interactive data science notebooks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 126–173, Toronto, Canada. Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8413–8426. Association for Computational Linguistics.

Yunjia Zhang, Jordan Henkel, Avrilia Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M. Patel. 2024. Reactable: Enhancing react for table question answering. *Proc. VLDB Endow.*, 17(8):1981–1994.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Fan Zhou, Mengkang Hu, Haoyu Dong, Zhoujun Cheng, Fan Cheng, Shi Han, and Dongmei Zhang. 2022. Tacube: Pre-computing data cubes for answering numerical-reasoning questions over tabular data. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 2278–2291. Association for Computational Linguistics.

Jun-Peng Zhu, Peng Cai, Kai Xu, Li Li, Yishen Sun, Shuai Zhou, Haihuang Su, Liu Tang, and Qi Liu. 2024. Autotqa: Towards autonomous tabular question answering through multi-agent large language models. *Proc. VLDB Endow.*, 17(12):3920–3933.

A Prompts

We provide below the main prompts passed to the agents in the different OrQA stages. The system and task prompts are concatenated.

```
evaluation_prompt = f""" \
    You are a helpful assistant in tabular data comprehension.
    Your task is to evaluate pairs of candidate tables
    for a SQL operation by providing a numerical score.
    If given, reason on other assistants observations.
    Limit your output to 50 words: your final answer should be
    a single integer number, between {self._min_score} and {self._max_score}.
    Respond with the form:
    Answer: <your numerical score here>
    Explanation: <your concise explanation>
    ---------------------------------------------------------------
    The table '{r_rsc_name}' belongs to the package '{r_pkg_name}'.
    This package is published by the organization '{r_org_name}',
    that is '{r_org_desc}', under the jurisdiction '{r_jur}'.
    The table description is: {r_pkg_notes}.
    Keywords and tags about it are: {r_pkg_keywords}, {r_pkg_tags}.
    Example rows with schema: {r_df_str}
    ---------------------------------------------------------------
    The table '{s_rsc_name}' belongs to the package '{s_pkg_name}'.
    This package is published by the organization '{s_org_name}',
    that is '{s_org_desc}', under the jurisdiction '{s_jur}'.
    The table description is: {s_pkg_notes}.
    Keywords and tags about it are: {s_pkg_keywords}, {s_pkg_tags}.
    Example rows: {s_df_str}
    ---------------------------------------------------------------
    Define a relationship quality score for the two tables.
    Focus on the meaningfulness of a potential operation between
    the given tables.
"""
```

Listing 2: Candidate table pair Evaluator agent system and initial task prompts.

```
debate_prompt = f""" \
    Using the evaluations from other agents as additional
    information, provide your score to the current table pairs.
    The original task is: {task}.
    These are the evaluations from other agents:
    One agent evaluation: {agent_evaluation}.
    ...
    One agent evaluation: {agent_evaluation}.
"""
```

Listing 3: Candidate table pair Evaluator agent inter-debate prompts.

```
query_generator_prompt = f""" \
    You are a SQL coder assistant. Your task is to generate SQL
    queries of different difficult levels.
    A 'simple' query involves just basic operations, like simple
    WHERE clauses.
    A 'moderate' query could use also casting, string replacement,
    grouping functions and other forms of aggregations.
    A 'challenging' query may require window functions, subqueries
    and other complex operations.
    You are using DuckDB: if necessary, put column names inside
    double-quotes, like "column_name".
    Do not cast FLOAT to REAL. If a VARCHAR attribute is similar
    to a datetime, try to cast it to DATE or DATETIME.
    When using regex operations, use proper options.
    Use the given tool to validate your SQL query: your response
    must be only a valid function call.
    ---------------------------------------------------------------
    Given the following information:
    Use 'R' to indicate the first table.
    Its schema is:
    {r_SQL_schema}
    Example rows of R table:
    {r_df_str}
    ---------------------------------------------------------------
    Use 'S' to indicate the second table.
    Its schema is:
    {s_SQL_schema}
    Example rows of S table:
    {s_df_str}
    ---------------------------------------------------------------
    Generate a {difficulty} SQL query based on the given tables.
    Use only 'R' and 'S' to reference the tables.
    The query must include a JOIN on the R column {r_col_name}
    and on the S column {s_col_name}.
    The new query must be different from previous queries:
    {prev_SQL}.
"""
```

Listing 4: SQL Generator agent system and task prompts to generate multi-table SQL queries involving a JOIN operation. Prompts for single and UNION queries are similar.

```
    question_generator_prompt = f""" \
        Your task is to generate natural language questions, related
        to tables from Open Data.
        Pretend to be a user that is using Open Data search portals
        and needs to get answers.
        The questions you create must be fluent and human-like: do not
        use SQL-like words, such as null or select.
        Keep focus on join and union operations between tables, if any.
        If available and meaningful, use the given keywords and tags.
        Because a common Open Data user (as you, in this case) does
        not know anything in advance about the final result, you can't
        use terms like records, data, datasets, tables, csv, packages
        and resources.
        If values are used inside the SQL query, try to
        understand what they means based on the given context: for
        example, 'ref' may mean 'refused' in a column about orders
        status.
        You must not use explicit table or column names into the
        question.
        Your response must be only the question, nothing else.
        ---------------------------------------------------------------
        Consider the following information:
        The table '{r_rsc_name}' belongs to the package '{r_pkg_name}'.
        This package is published by the organization
        '{r_org_name}', titled as '{r_org_title}' that is
        that is about '{r_org_desc}', under the jurisdiction '{r_jur}'.
        The table description is: {r_pkg_notes}.
        Keywords and tags about it are: {r_pkg_keywords}, {r_pkg_tags}.
        Example rows with schema: {r_df_str}
        ---------------------------------------------------------------
        The table '{s_rsc_name}' belongs to the package '{s_pkg_name}'.
        This package is published by the organization
        '{s_org_name}', titled as '{s_org_title}' that is
        about '{s_org_desc}', under the jurisdiction '{s_jur}'.
        The table description is: {s_pkg_notes}.
        Keywords and tags about it are: {s_pkg_keywords}, {s_pkg_tags}.
        Example rows: {s_df_str}
        ---------------------------------------------------------------
        Generate a natural language question which accurately
        represents the SQL query {sql} on the given tables
        and its aim.
        Pay attention to all the clauses used into the query.
        You must introduce into the question remainders to keywords,
        organization and other metadata.
    """
```

Listing 5: Natural Language question Generator agent system and task prompts to generate questions based on a multi-table operation.

~~~
query_reviewer_prompt = f""" \
    You are a query reviewer.
    You focus on the correctness of proposed SQL queries
    or Natural Language Questions.
    For the SQL, focus on the query syntax.
    Consider that is used DuckDB syntax.
    ---------------------------------------------------------------
    The problem statement is:
    {message.SQL_task}
    The proposed SQL query is:
    {SQL_query}
    The execution of this query is:
    {execution_result}
    Previous feedback:
    {previous_feedback}
    Revise the query if the execution was not successful.
    In the query has given an error, check if:
    - Previous feedback was not addressed.
    - The query does not involve required columns (if any).
    - The query is identical to any previously generated query.
    Respond with the following format:
    ```json
    {
        "correctness": <Your comments>,
        "approval": <APPROVE or REVISE>,
        "suggested_changes": <Your comments>
    }
    ```
"""
~~~

Listing 6: SQL Reviewer system and task prompts.

~~~
quuestion_reviewer_prompt = f""" \
    You are a query reviewer.
    You focus on the correctness of proposed SQL queries
    or Natural Language Questions.
    For the SQL, focus on the query syntax.
    Consider that is used DuckDB syntax.
    ---------------------------------------------------------------
    The problem statement is:
    {nl_task}
    The proposed Natural Language Question is:
    {nl_question}
    Previous feedback:
    {previous_feedback}
    Don't approve the question if:
    - Previous feedback was not addressed.
    - The question is too generic (like 'What is the average
      value?') or too simple (like 'Where is Canada?').
    - The question seems to be uncorrelated to the current task.
    - Columns and tables names are explicitly present into the
      question.
    - Columns required by the user are not correctly used (if any).
    - The question use too specific terms, like 'tables',
      'datasets', 'packages', 'data', 'records'.
    Respond with the following format:
    ```json
    {
        "correctness": <Your comments>,
        "approval": <APPROVE or REVISE>,
        "suggested_changes": <Your comments>
    }
    ```
"""
~~~

Listing 7: Natural Language question Reviewer system and task prompts.