Advancing Reliable and Explainable Evaluation of Domain-Specific Retrieval-Augmented Language Models

Anonymous ACL submission

Abstract

The advent of Large Language Models (LLMs) has significantly advanced the capabilities of Retrieval-augmented Generation (RAG) systems, leading to their extensive research and deployment across various industries for domain-specific knowledge querying. However, evaluating these systems presents unique challenges due to the scarcity of domain-specific queries and corresponding ground truths, as well as a lack of systematic approaches to diagnosing the cause of failure cases—whether they stem from knowledge deficits or issues related to system robustness. To address these challenges, we introduce an evaluation framework comprising two key elements: 1) a data generation process that leverages relational databases and LLMs to efficiently produce scalable query-answer pairs, facilitating the separation of query logic from linguistic variations for enhanced debugging capabilities; and 2) an explainable evaluation protocol equipped with a novel metric that assesses the extent of knowledge comprehension and system robustness in both retrieval and language modeling contexts. Importantly, our empirical findings highlight the shortcomings of prevalent reference-free evaluation methods, positioning our reliable reference-based evaluation protocol as a valuable adjunct.

1 Introduction

The increasing implementation of knowledge-intensive question answering systems, particularly Retrieval-Augmented Generation (RAG) systems, in practical applications is witnessing rapid growth. This trend underscores the critical need for effective evaluation of these systems, a task that remains intricate and largely unaddressed.

One of the foremost challenges is the scarcity of domain-specific data. Typically, evaluators resort to using data designed for human assessment, as referenced in studies such as (Santurkar et al., 2023; Wang et al., 2022; Zhong et al., 2022, 2023; Hendrycks et al., 2021; Choi et al., 2023), which mainly focuses on evaluating commonsense and broad world knowledge. Yet, the assessment of industrial RAG systems demands knowledge specific to particular domains, such as details about company projects and employees. While user queries can be gathered at the time of system deployment, acquiring accurate ground truths poses a significant challenge. Recent studies have explored the use of reference-free LLM evaluators in the absence of ground truths (Chern et al., 2023; Min et al., 2023; Es et al., 2023). Nonetheless, the reliability of such reference-free evaluation methods remains questionable. To address this, our paper proposes an innovative data-generation methodology that allows for the creation of ground-truth answers and scalable queries. This method leverages relational databases and LLMs, enabling the extraction of ground truths via SQL queries. Our empirical findings highlight the limitations of reference-free evaluation methods when compared to evaluations with ground truths.

A second significant challenge relates to the robustness of models. Specifically, we observe inconsistencies in the models' responses to queries that, while semantically equivalent, differ in their linguistic presentation, as highlighted in Shen et al. (2023) and illustrated in Table 1. Such inconsistencies pose a direct challenge to the crucial requirement for consistent performance. Researchers often hypothesize about the factors influencing the robustness or susceptibility of retrieval and language models to certain attributes, questioning what impedes these models from responding accurately to queries. Yet, the current body of research, including works on adversarial robustness (Alzantot et al., 2018; Li et al., 2020, 2019), lacks the mechanisms to effectively test these hypotheses. To address this gap, we propose a novel evaluation framework designed to facilitate the modular testing of hypotheses, offering a structured approach to assess model

Query (Q)	Response (A)
Q1: Who is the project leader or director of "Project X"?	A1: The project leader or director of "Project X" is Person A.
Q2: Tell me the director of "Project X"	A2: The director of "Project X" is Person B.
Q3: Who is the director of "Project X"?	A3: The passage does not contain information about the director of "Project X".

Table 1: Inconsistencies in responses to variations in linguistic expression. This example, derived from an industrially deployed RAG system, showcases discrepancies in answers to semantically equivalent queries. Specific names and project identifiers have been anonymized.

performance in response to skeptical attributes. By taking advantage of the control offered by our proposed data generation methodology, we can create clusters of queries tied to the predefined attributes. One cluster is aligned with a specific hypothesis while another is matched with its opposing counter-hypothesis, thereby supporting a more focused and hypothesis-driven investigation into vulnerable attributes. Especially, the ablation study proves the significance of separating the assessment of the model's knowledge gaps from its robustness, a distinction that can be effectively addressed through the proposed framework.

2 Background

Retrieval-augmented Generation RAG systems augment LLMs with retrieval for domain-specific use where extensive databases can potentially provide necessary information. A sparse retrieval (Chen et al., 2017) or a dense retrieval (Mialon et al., 2023; Lewis et al., 2020; Borgeaud et al., 2022) can be used for retrieving relevant passages or documents. A dense retrieval consists of an embedding model to produce a vector v_q for any given query q. This vector is then used to identify pertinent text segments normally via dot product calculations with the vectors of text chunks. The most relevant chunks are aggregated to form an additional context c, which serves as the factual basis for the response generation, until adding another chunk would surpass the maximum allowable context length. Subsequently, an LLM will process this aggregated context alongside the original query to generate a predicted answer $\hat{a}$ for q. Our empirical study will use simple sparse retrieval since it can better demonstrate the ability of the proposed framework.

Knowledge-intensive Evaluation Many evaluation protocols of RAG and LLMs focus on open-domain knowledge. Both the queries Q and their ground-truth answers A in the open domains can be easily accessed via online forums, public surveys (Santurkar et al., 2023) and qualification and admission exams that were originally designed for humans Wang et al. (2022); Zhong et al. (2022, 2023); Hendrycks et al. (2021); Choi et al. (2023). Since LLMs are originally trained on open-domain knowledge, these evaluation protocols on RAG focuses more on whether LLMs' factuality can be improved by retrieval. However, this paper focuses on industrial scenarios where retrieval mostly provides new knowledge to LLMs and ground-truth answers are hardly available. The reference-free evaluation protocol is commonly proposed for this situation, e.g., ARES (Saad-Falcon et al., 2023) and RAGAS (Es et al., 2023). For example, instead of the correctness of answers against the users' queries, Saad-Falcon et al. (2023) assess answer faithfulness against the context from retrieval modules by profiling LLMs to generate evaluation scores. The results can be skeptical due to the reliability of retrieval and the fairness of LLMs as evaluators. Reference-free evaluation is also utilized to assess LLMs' open-domain knowledge Kadavath et al. (2022). SelfCheck (Manakul et al., 2023) relies on the stochastic nature of LLMs for validations, while FactScore (Min et al., 2023) and FactTool (Chern et al., 2023) use Retrieval+LLM as a validation system. FactScore and FactTool are naturally not suitable for evaluating RAG systems that already include retrieval as an inherent component.

3 Domain-specific Evaluation Framework for Grounded and Explainable Analysis

In this section, we delve into the intricacies of the proposed evaluation framework. Our framework aims to facilitate the analytical evaluation of RAG systems through a methodical approach. This objective is achieved through a process of scaled, grounded data generation coupled with the decoupling of query logic from linguistic variations during the generation of queries, detailed in Algorithm 1. Leveraging this approach in data

generation allows us to introduce an explainable evaluation protocol. This protocol is designed to isolate and analyze modular errors effectively, utilizing a specifically crafted robustness metric for insightful debugging.

3.1 Generating SQL Templates

A SQL template tpl_{sql} essentially represents a query logic. It can be instantiated into text templates, each denoted as tpl_t, by linguistic features and then transformed into text queries by specifying the details of entities.

Database Schema Database schema serves as the blueprint that defines how classes of entities are organized and the relations among them. Appendix A demonstrates the reasons why we use database schema for knowledge representation. Let a database schema with a collection of N tables $S = \{S_1, S_2, \ldots, S_N\}$, where each S_i is a schema for a table. A table schema S_i consists of the table name T_i, a set of attributes $\{A_1, A_2, \ldots, A_M\}$ and a set of constraints, i.e., $S_i = (T_i, \{A_1, A_2, \ldots, A_m\}, C)$, where the constraint C demonstrates the primary key PK and foreign keys FK. Specifically, $FK\,(T_i.A_k \rightarrow T_j.A_h)$ indicates a foreign key A_k in table T_i referencing the primary key A_h in table T_j. The schema defines the knowledge structure of an entity, which can be enough to generate query logic that is not tied to any specific linguistic expressions and sensitive data.

Generation of SQL Templates The creation of SQL templates can be undertaken manually, with the formulation of various query logics through the application of relational algebra operators. In our study, we concentrate on three fundamental operators:

- The Select (σ) operator determines the attributes for querying, such as requesting specific attributes of an entity or seeking entities that meet certain predicates;
- The Project (π) operator, analogous to the WHERE clause in SQL, sets the conditions for selection;
- The Join ($\bowtie$) operator facilitates the expression of query logic involving multiple types of entities.

Furthermore, we have developed an automated SQL template generator g_{sql}, powered by a generative language model. Specifically, when provided with a target schema $\mathcal{S}$target and a verbal description of the query logic $\mathcal{C}$sql, the template generator g_{sql} is tasked with producing a set of SQL query templates that align with the established criteria, drawing from the relevant domain (refer to Appendix B for details).

3.2 Generating Text Templates

Upon acquiring SQL query templates denoted as tpl_{sql}, the next phase involves the generation of corresponding text templates, represented as tpl_t. This process, driven by the text template generator g_{t}, leverages linguistic criteria $\mathcal{C}_{\text{t}}$ to transform tpl_{sql} into natural language forms that align with specified linguistic characteristics, e.g., complexity, length and stylistic nuances. The text template generator g_{t} is also operationalized using a sophisticated language model, such as GPT-4, which is adept at producing diverse linguistic variations of the same semantic content.

$$\{tpl_{\text{t}}\} = g_{\text{t}}(tpl_{\text{sql}}, \mathcal{C}_{\text{t}})$$

, where $\{tpl_{\text{t}}\}$ signifies the resulting list of text templates.

3.3 From Templates To Evaluation Data

Utilizing a database $\mathcal{D}$ that aligns with the predefined schema, the framework samples a diverse range of queries along with their corresponding ground-truth answers. This dataset, derived from SQL and text templates filled with database content, provides a rich source for evaluation.

Generating Queries Via Placeholder Fill-in The placeholder fill-in step populates the templates with actual data from the rows of tables in $\mathcal{D}$. Specifically, the approach employs a SELECT query format: "SELECT DISTINCT {column_name} FROM {table_name};" to ensure variety in the data points. It handles SQL query templates with multiple placeholders by employing combinations of placeholders and using Cartesian products to generate multiple query permutations. This approach leads to a comprehensive set of SQL queries (q_{sql}) and their natural language equivalents, the text queries (q_t).

Generating Answers Answer generation is integral to completing our dataset. Each SQL query

Algorithm 1 Grounded Data Generation Process.

Require: SQL Template Generator g_{sql}, Text Template Generator g_{t}, Semantic criteria for SQL template generation $\mathcal{C}_{\text{sql}}$, Linguistic criteria for text template generation $\mathcal{C}_{\text{t}}$, Database $\mathcal{D}$, Database Schema $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$, where each S_i is a schema for a table

$Q \leftarrow \emptyset$ ▷ Store final evaluation data
$\mathcal{S}_{\text{target}} \in \mathcal{S}$ ▷ Step 1: Define schema of a domain
$\{tpl_{\text{sql}}\} \leftarrow g_{\text{sql}}(\mathcal{S}_{\text{target}}, \mathcal{C}_{\text{sql}})$ ▷ Step 2: Generate SQL templates applying semantic criteria
for $tpl_{\text{sql}} \in \{tpl_{\text{sql}}\}$ **do**
  $\{tpl_{\text{t}}\} \leftarrow g_{\text{t}}(tpl_{\text{sql}}, \mathcal{C}_{\text{t}})$ ▷ Step 3: Generate text templates applying linguistic criteria
  $P \leftarrow$ extract placeholders from tpl_{sql}
  for each $p \in P$ **do**
    $C \leftarrow$ extract column from p
    $V_p \leftarrow$ query $\mathcal{D}$ for distinct values of C
  end for
  $Comb \leftarrow$ Cartesian product of $\{V_p : p \in P\}$
  for each $comb \in Comb$ **do**
    $q_{\text{sql}} \leftarrow$ substitute $comb$ into tpl_{sql} ▷ Step 4: Generate SQL queries
    $a \leftarrow$ query $\mathcal{D}(q_{\text{sql}})$ ▷ Step 5: Query DB for answers
    $\{q_t\} \leftarrow$ substitute $comb$ into $tpl_{\text{t}} \in \{tpl_{\text{t}}\}$ ▷ Step 6: List of text queries
    $Q \leftarrow Q \cup \{(\{q_t\}, a)\}$
  end for
end for
return Q

(q_{sql}) is executed against $\mathcal{D}$ to match with a factual answer a. Each answer is then paired with the relevant text queries $\{q_t\}$, forming the basis of our evaluation data.

3.4 Scaling Data Generation With Combinatorial Expansion

Within the data generation framework, which spans from schema definition through SQL templates and text templates to the final formulation of text queries, there exists significant combinatorial potential at each transition point, driven by a range of factors. This process of generating queries via LLM, which includes both the creation of templates and the formulation of queries, can be succinctly represented by the formula (see Figure 1 for a visual representation):

$$\text{Total Query Variations} = M \times N \times Q, \quad (1)$$

where "M" denotes the array of SQL templates that can be extracted from a given schema, "N" refers to the vast number of text templates that can be produced from a single SQL template, showcasing the flexibility of natural language, and "Q" accounts for the broad spectrum of text queries that can be generated from a single text template, with this diversity arising from different combinations of unique values filling the placeholders. A comprehensive analysis of these concepts is presented in Appendix D and C.

In this scenario, a database with a modest number of attributes and rows suffices for testing purposes, given the extensive variety of query permutations made possible through the combinatorial approach.

3.5 Explainable Evaluation

This section is dedicated to establishing a framework for explainable evaluation by distinguishing between the model's knowledge base and its robustness. To this end, we introduce two distinct group classes and a robustness metric to facilitate precise assessments of the model's knowledge and robustness.

Gap Groups And Competence Groups All textual queries Q generated from a specific SQL logic q_{sql} can be grouped semantically.[1] Depending on the model M's performance, we categorize q_{sql} (and consequently, the group Q) into three principal categories:

- **Gap Groups** emerge when the model M consistently provides incorrect answers for every text query within a subset $S \subseteq Q$ produced through the data generation process. This reflects a deficiency in M 's knowledge or capability. Formally, this scenario is described as:

$$\forall q_t \in S, \neg M\left(q_t\right)$$

  wherein $\neg M\left(q_t\right)$ signifies an inaccurate response from M to the text query q_t.

[1] Refer to Appendix E for an illustration.

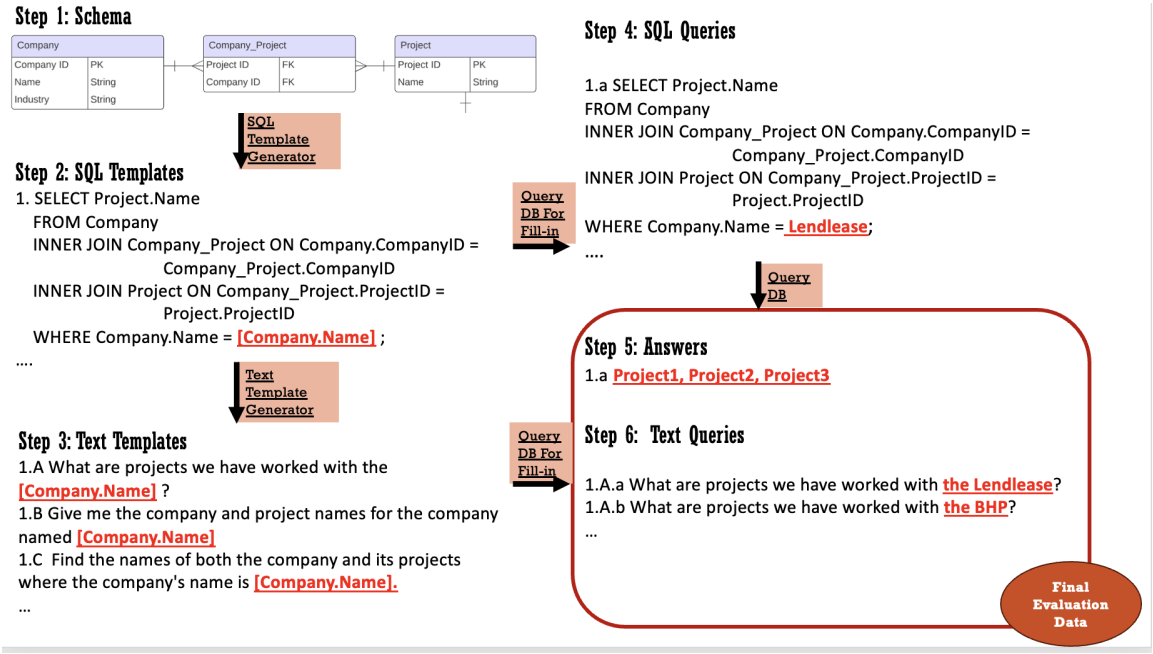

Figure 1: Scaled Generation of Query-answer Pairs With Combinatorial Power. The step 3 and step 4 are independent of each other and depend only on the SQL templates.

- **Competence Groups**, as the logical complement to gap groups, are identified when at least one query q_t within the subset S is answered correctly by the model M, indicating M 's potential competence in the underlying query logic q_{sql} . This is formally stated as:

$$\exists q_t \in S, M\ (q_t)$$

These groups are further subdivided into: 1) **Robust Groups** are established when model M accurately responds to all text queries within the subset S, evidencing a robust comprehension of the SQL logic. Formally, this is articulated as:

$$\forall q_t \in S, M(q_t)$$

. 2) **Non-robust Groups** are characterized by model M's exhibited capability to potentially address the query logic, yet there exists at least one query q_t within S that M inaccurately predicts. This is formally denoted as:

$$(\exists q_t \in S, M(q_t)) \wedge (\exists q_t \in S, \neg M(q_t))$$

.

Robustness Metric (R) Robustness is defined as the proportion of correct answers within the competence groups:

$$R = \frac{\text{\# Correct Predictions in Competence Groups}}{\text{Total \# in Competence Groups}}$$

, where "#" represents the number of. This formula intentionally excludes gap groups to avoid confusing the model's knowledge deficiencies with its actual robustness. Essentially, the robustness metric is calculated as the ratio of accuracy normalized by N_1/N_2, where N_1 is the total number of instances and N_2 is the total number of instances subtracting those in gap groups. Assuming the proportion of instances in gap groups is represented by λ, the normalization constant simplifies to $1/(1-\lambda)$. Section 4 will illustrate the importance of isolating gap groups for an explainable evaluation and introduce, in addition to the robustness metric, a stratified sampling method derived from our data generation process to align with accuracy outcomes.

3.6 Modular Evaluation

Assessing an RAG model's overall performance may obscure insights into the robustness or weaknesses of its constituent modules (refer to Section 4.2). It is essential, therefore, to evaluate the contributions of the retrieval and language modeling (LM) components individually: Does the retrieval module source and provide the sufficient context needed for a given query? With the sufficient context at its disposal, is the LM capable of drawing accurate inference? A correct final prediction implies both appropriate context retrieval and accurate LM inference. The difficulty lies in pinpointing the cause of errors in non-robust examples —

those mispredicted by the model within non-robust groups.

Delineating LM-Influenced Robustness from Retrieval-Centric Issues Utilizing a "context comparison" methodology enables pinpointing the LM's role in generating erroneous predictions when adequate context is provided. By contrasting the document indices idx_c of each non-robust instance against indices from accurate predictions within analogous semantic groups against those linked to correct predictions within the same semantic groups $I = \text{idx}_1, \text{idx}_2, \ldots$, we evaluate the sufficiency of context. This assumes the LM's reliance on external information for query responses, mirroring our scenario and prevalent industrial applications. Should any index in I match idx_c, we reclassify the outcome as a successful retrieval yet an LM shortfall. This strategy allows for precise identification of LM-related non-robust instances. Nevertheless, the absence of such matches does not conclusively indicate retrieval failures. Given our experimental setup with a synthetic database comprising 10 documents, this "context comparison" method effectively identifies all document combinations across a moderate query range (3 to 6) within semantically related groups. This framework lays the groundwork for future explorations into more intricate cases.

4 Case Study: Question Answering on Industrial Projects

This section presents a case study focusing on question answering about industrial projects. It serves to critically evaluate both existing reference-free evaluation frameworks and the newly proposed evaluation framework

System Overview Our study incorporates two distinct systems: 1) **Dense Retrieval + GPT3.5**: It is deployed in a commercial context with secure access to the retrieval module via the Qdrant API, ensuring data privacy. The system employs an OpenAI embedding model for document embedding, providing a basis to assess the reference-free evaluation framework's reliability (§4.1). 2) **Simple Sparse Retrieval + GPT3.5**:The absence of comprehensive research into the assured robustness of dense retrieval systems makes it difficult to pinpoint distinct robust or vulnerable characteristics with confidence. Consequently, we employ a straightforward keyword-matching retrieval approach. Although this method is less prevalent in real-world applications, this method's inherent shortfall in handling lengthy queries makes it exceptionally ideal to validate our evaluation framework in conducting reliable hypothesis testing (§4.2).

Synthetic Knowledge Base Concurrently, a synthetic dataset has been created, consisting of project documents from the fictitious company, Aurp, serving as a retrieval database. It facilitates a precise evaluation of the retrieval component and the language model, since the knowledge contained within the synthetic data is not present in the training datasets of large language models. In scenarios where retrieval fails to procure adequate context, the language models are definitely unable to accurately respond to queries. Additional information on this methodology can be found in Appendix G.

Mapping Domain Knowledge into Database Schema Our approach distills domain knowledge into a database schema centered around three pivotal entity types in project management: Project, Client, and Employee. [2]This schema encapsulates the quintessential entity types found within a corporate setting. With the scalability of our framework, the limited amounts of entity types and associated attributes and entities lead to large amounts of instances for evaluation: 198 examples for the commercial case (with totally 12 attributes and 11 rows) and 942 examples for the synthetic scenario (with 14 attributes and 49 rows in total). The details of the database schema and the amounts of columns and rows for each table are shown in Appendix F and H), respectively.

4.1 Is Reference-free Evaluation Reliable?

Leveraging ground truths, our evaluation framework sets a benchmark for reference-free assessments. This section delves into the reliability of reference-free evaluation methods, utilizing examples generated for the commercial case. The focus is to discern the reliability of such evaluation approaches: How trustworthy are these reference-free protocols? From what perspectives can they be relied upon in domain-specific contexts?

Reference-free Evaluation Protocols We assess two reference-free evaluation protocols: SelfCheck

[2]This triad are reflective of the core components outlined in the Project Management Body of Knowledge (PMBOK), a widely recognized standard in the field of project management (Guide, 2001).

and RAGAS. RAGAS-Fact utilizes the context-query-response triplets to assess the veracity of responses. It evaluates the faithfulness of a response by calculating the ratio of claims grounded on the context to the total claims made. This process, termed as **Ragas-Fact**, involves identifying statements that hold atomic facts, following the methodology outlined by Chern et al. (2023). SelfCheck, as introduced by Manakul et al. (2023), relies on stochastically generating responses from the model under the premise that incorrect answers are unlikely to repeatedly yield identical outcomes through random sampling. This principle, initially applied to LLMs, is adapted for our analysis of RAG systems. In a departure from its original application, we enhance the evaluation prompt to include not only stochastic responses but also the query itself. Refer to Appendix I for details. To generate four stochastic samples, we adjust the temperature setting to 1.0, contrasting with a temperature of 0.0 used to procure the primary response. A response is deemed correct if it aligns consistently across all stochastic samples.

RAGAS' Reliable Assessment of Correct Predictions Amidst Broader Unreliability in Reference-Free Evaluation To gauge reliability, we devise two metrics: the ratio of predictions judged as accurate among all those classified as correct by the evaluation, and the ratio of predictions deemed accurate within the set of instances that were indeed correctly predicted. When considering reference-free evaluation as a binary classification task, these metrics align with precision and recall, respectively. In this context, the positive class represents correct predictions, and the negative class represents incorrect ones. The ground truth corresponds to the actual accuracy of model predictions, while the evaluation task is to determine the correctness of these predictions. As shown in Table 2, RAGAS demonstrates effectiveness in assessing accurately predicted instances, yet it achieves only a 19% accuracy rate across all evaluations, highlighting a deficiency in correctly identifying erroneous predictions. This trend suggests a propensity for inflating model performance assessments, likely due to the RAG system producing responses that, while contextually relevant, fail to directly address the intended query. SelfCheck performs even worse than RAGAS in both cases. SelfCheck's performance falls short of RAGAS's, underperforming in both precision and recall. The preliminary insight suggests that reference-free evaluation methods are inherently prone to inaccuracies, underscoring the critical need for a robust, reference-based evaluation framework to ensure dependable model assessment.

	Precision	Recall
Ours (Upper Bound)	1	1
RAGAS-Fact	19%	92%
SelfCheck	15%	50%

Table 2: Reliability of reference-free evaluation protocols.

4.2 Empirical Validation of Framework's Explainability

Generating Clusters of Data for Hypothesis Testing The simple keyword-matching retrieval implemented for testing is inherently weak against long queries due to the increased likelihood of matching extraneous terms. By hypothesizing the model's inadequacy with lengthy queries, we define robust and vulnerable clusters through the controlled generation of "short and concise" versus "long and detailed" queries. This allows for the analysis of linguistic traits within uniform query logic contexts, laying groundwork for future assessments of more sophisticated RAG systems.

Stratified Sampling Our data generation strategy enables precise control over the distribution of examples across gap groups by adjusting the text template generator's inputs, ensuring an equal representation of each query logic. This method maintains a consistent λ across both robust and non-robust clusters, despite the potential imbalance between examples in gap groups and those in competence groups within each cluster. To explore the effects of unbalanced sampling, that can occur in the real world, we randomly reduce examples from query groups related to "clients" and "employees".

Enhanced Model Evaluation through Stratified Sampling and the Robustness Metric Table 3 demonstrates that traditional accuracy metrics falter in distinguishing clusters based on vulnerability attributes, particularly when λ varies between the hypothesis-driven cluster and its counterpart (Row 2 under the "Accuracy" category). Stratified sampling ameliorates this issue, enabling a precise identification of both robust and non-robust clusters (Row 4 under the "Accuracy" category).

		Robust	Non-robust	Success?
Accuracy	Baseline: Unbalanced Sampling (US)	0.31	**0.43**	No
	US+Modular Evaluation (Retrieval)	0.36	**0.46**	No
	Stratified Sampling (SP)	**0.31**	0.29	Yes
	SP+Modular Evaluation (Retrieval)	**0.38**	0.35	Yes
Robustness	Baseline: Unbalanced Sampling (UP)	0.84	**0.85**	No
	US+Modular Evaluation (Retrieval)	**0.99**	0.93	Yes
	Stratified Sampling (SP)	**0.78**	0.74	Yes
	SP+Modular Evaluation (Retrieval)	**0.94**	0.87	Yes

Table 3: Evaluating the efficacy of accuracy and designed robustness metrics in distinguishing between robust and non-robust clusters. Modifications include retrieval-specific adjustments by excluding LM-induced non-robust examples.

Importantly, the implementation of a specifically designed robustness metric ensures consistent and accurate identification of the robust cluster, irrespective of λ variations (Row 2 and 4 under the "Robustness" category). This approach, integrating stratified sampling with our bespoke robustness metric, providing a reliable analysis.

The Masking Effect of Comprehensive Assessments Analysis presented in Table 3 underscores the value of segmenting the evaluation to specifically examine the retrieval process. Such an approach distinctively highlights errors stemming from the language model's misinterpretations, as reflected by improvements in both assessed metrics. A deeper analysis reveals that groups classified as non-robust within the non-robust cluster actually surpass their equivalents in the robust cluster regarding accuracy. This underscores the importance of evaluating the retrieval function independently; failing to do so may allow language model inaccuracies to obscure the true performance metrics, potentially masking retrieval robustness issues. Although instances causing such distortions were not dominant in our dataset, they highlight a critical evaluation nuance. While our study did not explore this avenue, it is posited that a focused analysis on the language model's performance, especially against LM-vulnerable examples, might offer a more refined insight into its robustness across varied clusters.

5 Conclusions

In our work, we have developed an evaluation framework tailored for Retrieval-augmented Generation (RAG) systems within an industrial context, enhancing the reliability and interpretability of evaluations. Additionally, we have exposed the limitations of current reference-free evaluation methods, particularly their ineffectiveness in accurately assessing incorrect predictions.

Reproducibility The entire evaluation framework, encompassing both the code and the prompts for the LLM, will be made open-source and available to the academic and industrial community. This initiative aims to facilitate the wider use and adaptation of our framework for research and real-world implementations. In terms of the systems and data evaluated to enable result replication, we plan to publish the model, including the documents intended for retrieval, as well as all templates and queries developed for the hypothetical Aurp scenario. However, the model designated for commercial use and its associated knowledge base will remain confidential and unpublished to honor non-disclosure agreements and protect proprietary information.

6 Limitations

The challenges related to the presence of multiple correct answers (refer to Appendix J) and the constraints in expressiveness posed by database schema and SQL (refer to Appendix K) ultimately result in difficulties in generating queries that demand multi-step reasoning and free-form responses. Nonetheless, as demonstrated in the case study, straightforward queries prove to be sufficient for identifying and debugging the intrinsic robustness issues within the models.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

I Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, Pengfei Liu, et al. 2023. Factool: Factuality detection in generative ai–a tool augmented framework for multi-task and multi-domain scenarios. *arXiv preprint arXiv:2307.13528*.

Jonathan H Choi, Kristin E Hickman, Amy Monahan, and Daniel Schwarcz. 2023. Chatgpt goes to law school. *Available at SSRN*.

Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2023. Ragas: Automated evaluation of retrieval augmented generation.

A Guide. 2001. Project management body of knowledge (pmbok® guide). In *Project Management Institute*, volume 11, pages 7–8.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics.

Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S Yu. 2023. A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability. *arXiv preprint arXiv:2303.13547*.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models.

Grégoire Mialon, Roberto Dessi, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Roziere, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented language models: a survey. *Transactions on Machine Learning Research*. Survey Certification.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. *arXiv preprint arXiv:2305.14251*.

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2023. Ares: An automated evaluation framework for retrieval-augmented generation systems.

Shibani Santurkar, Esin Durmus, Faisal Ladhak, Cinoo Lee, Percy Liang, and Tatsunori Hashimoto. 2023. Whose opinions do language models reflect? *arXiv preprint arXiv:2303.17548*.

Xinyue Shen, Zeyuan Chen, Michael Backes, and Yang Zhang. 2023. In chatgpt we trust? measuring and characterizing the reliability of chatgpt. *arXiv preprint arXiv:2304.08979*.

Siyuan Wang, Zhongkun Liu, Wanjun Zhong, Ming Zhou, Zhongyu Wei, Zhumin Chen, and Nan Duan. 2022. From lsat: The progress and challenges of complex reasoning. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 30:2201–2216.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*.

Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, Yining Chen, Jiahai Wang, Jian Yin, Ming Zhou, and Nan Duan. 2022. Analytical reasoning of text. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2306–2319, Seattle, United States. Association for Computational Linguistics.

A Knowledge Structure and Representation

Commonly, the knowledge is formalized by entities along with their relations, e.g., a knowledge graph, ontology and a database schema. Instead of using knowledge graphs containing concrete entities and their relations, we use the schema-based definition, where the schema is characterized by entity types (classes of entities). An entity type is defined by its name, its attributes and relations with other entity types that will be concretized as components in a database, i.e., the table name, columns and foreign keys.

The definition of entity types can provide combinatorial expansion of data generation (See §3.4), protecting the leak of private information (normally stored as rows of tables) during the data generation process and the ability of retrieving ground-truths via Structured Query Language (SQL). The three advantages are the reasons why we represent knowledge in database schema rather than knowledge graphs. It disentangles the private data stored in the database with universal metadata expressed in the database schema. Since schemas and templates contain only meta-data for structuring the real data, there is no private issue to use commercial large language models for this process.

B Utilizing LLMs for Generating SQL Templates

We utilize GPT-4 as a SQL template generator for its state-of-the-art capabilities in SQL generation benchmarks (Liu et al., 2023).

Profile Prompt The profile prompt defines its purpose and establishes criteria for generating valid SQL templates. These criteria limit the types of templates that can be generated, such as requiring the selection of attributes that are understandable to humans, as detailed in Table 4.

Limitations: Scope of Use for Automated SQL Generators The effectiveness of queries generated by LLMs hinges on the meaningfulness of table and column names to accurately reflect the essence of the entities or concepts they represent. If these names lack contextual clarity, the resulting queries may be impractical. To mitigate this, practitioners can conduct an informal assessment or consult established benchmarks that evaluate LLMs' proficiency in domain knowledge and commonsense reasoning, for instance, (Zhong et al., 2023; Hendrycks et al., 2021).

C Breakdown of Scalable Data Generation

Let's break down each step of the proposed data generation process:

- 1 schema ⇒ M SQL Templates: The number of possible SQL templates (M) that can be generated from a defined schema is influenced by the SQL operators used, the number of tables, and the columns in the database schema. Each combination of tables and columns, along with different SQL operators (like SELECT, WHERE, JOIN), can lead to a unique SQL query template, e.g., the exponential complexity of the SELECT operator, and the combinatorial increase of possible predicates (See Appendix D for details). Note that while the theoretical maximum is high, practical and meaningful queries will be a smaller subset, as specified in the case study and Appendix B.

- 1 SQL Template ⇒ N Text Templates: For each SQL template, there can be an arbitrary number (N) of textual expressions. This variation arises from the different ways to linguistically express the same SQL query due to the flexibility and richness of natural language, e.g., the variation of synonyms and sentence structures.

- 1 Text Template ⇒ Q Text Queries: Each text template can lead to a number of text queries (Q), depending on the unique values available for each placeholder. With more than one placeholder, the potential for growth in the number of text queries is combinatorial, barring semantic conflicts (context conflict), where certain combinations of column values may not be semantically valid. Overall, this framework demonstrates a combinatorial expansion at each transition stage, especially notable in steps involving natural language due to its inherent variability. However, this

```
You are a SQL query Template Generator: Generate ACCEPTABLE SQL query
templates with placeholders according to the give data schema and
requirements. A simple example of an acceptable SQL query template is:
SELECT Industry FROM Company WHERE Name = '[Company.Name]';

You must follow the basic criteria below except for other requirements:
##CRITERIA##
- The placeholder format should be a combination of a table name and
a column name, enclosed within square brackets, e.g., '[User.Name]'.
- Use only 'SELECT' queries.
- Select specific column(s) instead of using '*'. Avoid projecting
attributes that appear in the predicate.
- The selected and condition columns in the query MUST BE MEANINGFUL and
DESCRIPTIVE to ensure the queries are easily understood by non-technical
users.
- Avoid using technical column names that don't clearly signify the nature
of the entities or objects involved, e.g., column for semantically void
record identifiers.
- Do not create redundant or semantically duplicated queries when translated
into natural language.
- Each query must contain at least one parameter placeholder in the WHERE
clause.
- Ensure the query yields a specific and singular answer to avoid multiplicity
issues, thus facilitating accurate chatbot evaluation.
 {SPECIFIC_REQUIREMENTS}
- If no acceptable SQL template can be generated with the given table and
column information, do not generate any text.

##RESPONSE FORMAT##
- Output each SQL template as a single line, without any prefix or suffix.
- Do not include any other text in your response, even something like
##RESPONSE_END##.

##DATA SCHEMA##
{GIVEN_SCHEMA}

##RESPONSE_START##
```

Table 4: A prompt template designed to guide the LLM in functioning as a controllable SQL template generator

growth is tempered by practical constraints such as the meaningfulness of queries (semantic validity) and the actual data distribution in the database. The exponential increase is most pronounced in the transition from text templates to text queries, where the permutations of placeholders can lead to a vast array of unique query possibilities.

D Possibilities Of SQL Templates

One Table/Entity Given a schema with only one table, the possibilities for SQL templates can be analyzed by considering the basic SQL operators like SELECT, WHERE. Here's a breakdown:

- Variation in Selected Attributes (SELECT): The number of SQL templates varies based on the combination of columns selected. If the

"Company" table has n columns, then theoretically, there are $2^n - 1$ possible combinations of columns for selection (excluding the case where no column is selected).

- Conditions in Queries (WHERE): Each SQL query can include zero or more conditions in the WHERE clause. The number of possible conditions is determined by the number of columns, the type of each column (text, numeric, date, etc.), and the range of operators applicable to these types (like =, <, >, LIKE, IN for text columns; =, !=, <, >, BETWEEN for numeric columns). The complexity increases combinatorially with multiple conditions combined using AND/OR.

Two Entities The relations between tables further amplifies the number of possible templates. For example, with two entities, "Company" and "Project," and their associative table "Company_Project", the possibilities for SQL templates expand significantly due to the introduction of joins and more complex WHERE clauses. Let's break down the possibilities.

- Selection Variations (SELECT): The number of SQL templates grows with the combination of columns selected across the three tables: "Company", "Project" and "Company_Project". If "Company" has n columns, "Project" has m columns, and "Company_Project" has p columns, the possible combinations for selection are $(2^n - 1) \times (2^m - 1) \times (2^p - 1)$.

- Join Conditions (JOIN): The introduction of the associative table "Company_Project" allows for meaningful JOIN operations between "Company" and "Project." Templates can include joins like Company JOIN Company_Project ON condition and Project JOIN Company_Project ON condition, or a multi-table join linking all three. The variety of JOIN conditions adds another layer of complexity to the possible templates.

- WHERE Clause Complexity: With more tables, the WHERE clause can include a wider range of conditions, potentially involving attributes from any of the three tables. The complexity increases with the number of columns and their types across all tables, and combinations of these conditions.

E Semantic Group

Textual queries that originate from a specific SQL logic q_{sql} are organized into a semantic group because of their underlying shared semantics. For instance, the SQL query "SELECT C.address FROM COMPANY C WHERE C.CompanyName = 'Carlton Innovation Precinct'" results in the creation of a semantic group $Q = \{q_{t1}, q_{t2}, \ldots\}$, with each q_t representing a distinct textual interpretation. Examples include q_{t1} as "What is the address of 'Carlton Innovation Precinct'?" and q_{t2} as "Where is 'Carlton Innovation Precinct' located?".

F Database Schemas Overview

Figures 2 and 3 illustrate the database schemas used for an actual industrial context and a fabricated scenario, respectively, within our research.

G Synthetic Knowledge Base for Evaluating Retrieval-Augmented Language Models

The process of generating synthetic data for a synthetic environment involves creating comprehensive, detailed, and interconnected datasets for a fictitious company named Aurp, which specializes in structural engineering consultancy services. This multi-step procedure includes:

1. **Generating Company Profile**: Initially, a company profile for Aurp is created, detailing its foundation year, headquarters location, CEO, number of employees, and the services it offers, such as bespoke architectural solutions, sustainable urban planning, and structural health monitoring.

2. **Generating Organizational Structure**: Next, a project-oriented organizational structure is established, naming key positions and employees within the company, similar to real-world firms. This includes a wide range of roles from executive positions to specialized engineers and support staff.

3. **Generating Employee Information**: For each employee listed in the organizational structure, detailed job titles, departments, and direct supervisors or managers are fabricated, creating a network of relationships and reporting lines within the company.

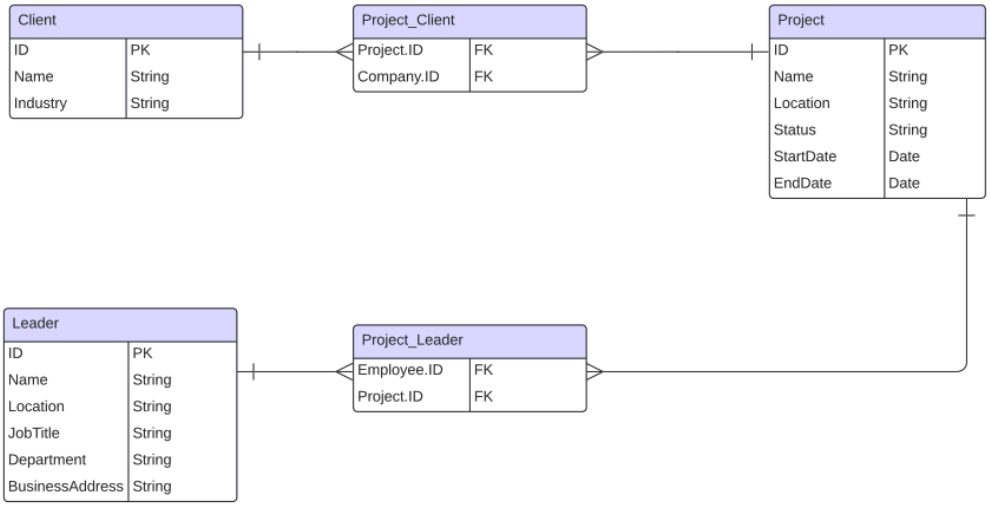

Figure 2: Entity-Relationship Diagram (ERD) showcasing the schema.

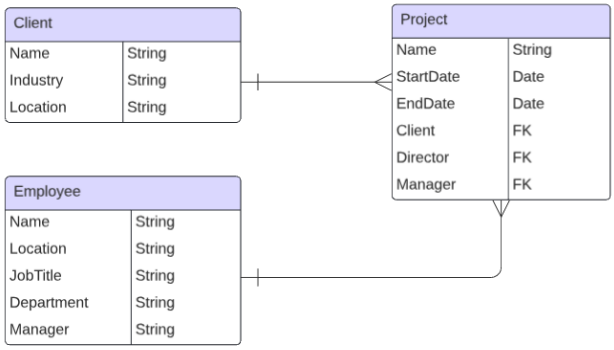

Figure 3: Entity-Relationship Diagram (ERD) depicting the database schema for a simulated setting.

	Attribute #	Template #			Row #	Queries #	
		SQL	SQL (Extra Constraints)	Text		SQL	Text
Project	5	7	3	18	4	13	78
Employee	5	12	5	30	3	14	84
Client	2	6	2	12	4	6	36
Total	/	/	/	/	/	33	198

Table 5: Statistics of DB Rows And Columns And Templates/Queries.

	Attribute #	Template #		Row #	Queries #	
		SQL	Text		SQL	Text
Project	6	5	15	10	50	300
Employee	5	4	12	29	87	522
Client	3	2	6	10	20	120
Total	/	/	/	/	157	942

Table 6: Statistics of Aurp DB Rows And Columns And Templates/Queries.

4. **Generating Client Information**: The process creates a list of ten clients across various industries—ranging from technology and real estate to hospitality and healthcare—each with specified locations, thereby illustrating the company's diverse portfolio.

5. **Generating Projects**: Specific projects are then devised, including names, locations, start and end dates, clients, project directors, and project managers. This step integrates previously generated data (client and employee information) to create realistic project scenarios.

6. **Generating Project Reports**: This step involves synthesizing data from the previous steps to produce detailed analyses, updates, and outcomes of the various projects, which may or may not contain detailed information for clients and employees.

This elaborate and interconnected data creation method is tailored for evaluating retrieval-augmented language models. It serves as a critical tool for exploring how these models handle data that is entirely fictional and thus not present in their training datasets. The synthetic nature of this environment provides a unique testbed for separating and assessing the distinct capabilities of the retrieval component and the language model component in isolation. Because the knowledge in the synthetic data does not appear in the training data of large language models.

H Assessing Scalability with Limited Entity Types, Rows, and Attributes

Our scalability assessment focused exclusively on the utilization of the SELECT operator and the WHERE clause for each table schema. As illustrated in Table 5 and 6, imposing constraints to generate relevant queries significantly reduces the potential quantity of SQL templates based on attribute count. Despite these restrictions, a minimal set of attributes and rows still yields hundreds of viable query-answer pairs.

I Prompt Templates for LLM-Based Evaluation

For SelfCheck, given that our responses are typically concise, we opt for assessing the overall response rather than rather than performing a sentence-by-sentence validation as in the original implementation.

J Presence of Multiple Correct Answers

In the evaluation of question-answering (QA) models, a unique challenge arises from the existence of multiple valid answers to a single query, which necessitates a nuanced approach to assessing model performance. Consider the question: "Get the name of the client associated with the project named Innovation Precinct" For such a question, a set of correct responses could include any combination of names from a predefined list, such as {Apple, Amazon, Meta, Facebook}. This multiplicity of correct answers underscores the complexity of evaluating QA models, as it requires the assessment mechanism to recognize and validate the full spectrum of possible correct answers rather than comparing the model's output against a single 'gold standard' answer. This scenario demands a more flexible method for answer validation that can accommodate the variability in correct responses. Also, evaluation metrics may be required to effectively measure the performance of QA models in handling diverse and equally valid answers. Such metrics must account for the exhaustive set of correct answers and evaluate the model's ability to retrieve any or all valid responses within the context of the query, thereby ensuring a comprehensive assessment of the model's understanding and response generation capabilities.

Simple Queries to Avoid Answer Multiplicity Generally, encompassing all valid responses within the database is not feasible. Hence, rather than let the non-robust results affect the accuracy of evaluation, we look for an approach to avoid it, i.e., only generating queries ensuring one ground-truth answer. Specifically, the evaluation of certain SQL queries, such as "SELECT Name, Department, BusinessAddress FROM Employee WHERE JobTitle = 'designer"', requires a complete and thorough listing of multiple answers that can be dynamically changed. To solve the issue, the solution involves adding a specific criterion to the prompt for generating SQL queries. This criterion, "Ensure the query yields a specific and singular answer", aims to produce queries that result in a single, clear answer, thereby avoiding the complexities of multiple possible correct answers. For example, with this criterion, the query "SELECT Industry FROM Company WHERE Name = '[Company.Name]';"

SelfCheck from (Manakul et al., 2023)
Context: {context}
Sentence: {sentence}
Is the sentence supported by the context above?
Answer Yes or No:

SelfCheck-QA
Query: {query}
Answer A: {answer}
Answer B: {stochastic_answer}
Do both answers address the query with equivalent meaning?
Use only "Yes" or "No" for your evaluation:

Ragas from (Es et al., 2023)
Natural language inference. Use only 'Yes' (1), 'No' (0) and 'Null' (-1) as verdict.
context: {context}
statement: {statement}
verdict:

Ours
Evaluate the accuracy of the given response in relation to the true answer for the specified query. After evaluating, provide a judgement as either "Correct" or "Incorrect" based on whether the ##Given Response## accurately matches the ##True Answer##.
##Query##: {query}
##True Answer##: {true_answer}
##Given Response##: {given_response}
##Judgement##:

Table 7: Prompt templates for LLM-based evaluation.

is preferred as it's likely to yield a singular answer about a company's industry based on a specific company name. In contrast, without this criterion, queries like "SELECT Name FROM Company WHERE Industry = '[Company.Industry]';" are acceptable but may result in multiple names, leading to evaluation difficulties due to data completeness and query multiplicity issues.

K Queries Beyond SQL Expressiveness

While SQL and relational algebra offer a wide range of operations enabling the formulation of numerous user queries, certain semantic nuances exceed the expressive capabilities of SQL. The following examples illustrate such limitations:

- Real-world queries often contain ambiguity and subjective interpretations that SQL struggles to accommodate. For instance, the term "major" in the query "What are some major rail projects we've been involved with?" does not directly translate into SQL criteria without additional interpretative steps.
- SQL queries are typically structured to elicit specific, predefined responses, contrasting with the open-ended nature of many real-world inquiries that seek exploratory or comprehensive answers.