Automated Evaluation of Database Conversational Agents

Matheus O. Silva[1][a], Eduardo R. S. Nascimento[1,2][b], Yenier T. Izquierdo[2][c], Melissa Lemos[1,2][d] and Marco A. Casanova[1,2][e]

[1]*Department of Informatics, PUC-Rio, Rio de Janeiro, RJ, Brazil*
[2]*Tecgraf Institute, PUC-Rio, Rio de Janeiro, RJ, Brazil*

Keywords: Conversational Agents, Database Interfaces, ReAcT, LLM.

Abstract: Database conversational agents support dialogues to help users interact with databases in their jargon. A strategy to construct such agents is to adopt an LLM-based architecture. However, evaluating agent-based systems is complex and lacks a definitive solution, as responses from such systems are open-ended, with no direct relationship between input and the expected response. This paper then focuses on the problem of evaluating LLM-based database conversational agents. It first introduces a tool to construct test datasets for such agents that explores the schema and the data values of the underlying database. The paper then describes an evaluation agent that behaves like a human user to assess the responses of a database conversational agent on a test dataset. Finally, the paper includes a proof-of-concept experiment with an implementation of a database conversational agent over two databases, the Mondial database and an industrial database in production at an energy company.

1 INTRODUCTION

Database conversational agents support dialogues to help users interact with databases in their jargon. They typically accept user questions formulated as natural language (NL) sentences and return results phrased in NL. Such agents should also help users formulate a question step-by-step and express a new question by referring to an older one or to a previously defined context.

In general, a conversational agent must maintain the dialogue state and resolve, among other problems, anaphora, ellipsis, ambiguities, and topic shifts (Galimzhanova et al., 2023). Large Language Models (LLMs) fine-tuned to follow instructions help address these challenges, exactly because they are trained to maintain context and resolve anaphora, among other nuances of user dialogues.

In addition, a database conversational agent must translate user questions, explicitly submitted as NL sentences in the dialogue or inferred from the dialogue, into SQL queries over the underlying database. This translation task is often referred to as *text-to-SQL*. Therefore, tools that address such task help construct database conversational agents because they are designed to match user terms with the database vocabulary and perform the translation.

Evaluating database conversational agents, especially those executing tool calls based on natural language dialogues, is crucial to validate their effectiveness and reliability in real-world scenarios. It is therefore necessary to adopt a systematic evaluation process capable of measuring the alignment between user intentions and the database conversational agent's actions, including the correctness of the SQL queries generated.

This paper then focuses on the problem of evaluating a database conversational agent over a given database D.

To address this problem, the paper introduces a *dialogue generation tool* that creates a *dialogue test dataset* T_D for D, an *evaluation agent* that simulates a user interacting with the database conversational agent over D, guided by the dialogue test dataset T_D, and a collection of dialogue performance metrics. This contribution is essential to verify the performance of a database conversational agent over a

[a] https://orcid.org/0009-0009-1932-296X
[b] https://orcid.org/0009-0005-3391-7813
[c] https://orcid.org/0000-0003-0971-8572
[d] https://orcid.org/0000-0003-1723-9897
[e] https://orcid.org/0000-0003-0765-9636

Silva, M. O., Nascimento, E. R. S., Izquierdo, Y. T., Lemos, M. and Casanova, M. A.
Automated Evaluation of Database Conversational Agents.
DOI: 10.5220/0013732900003985

In *Proceedings of the 21st International Conference on Web Information Systems and Technologies (WEBIST 2025)*, pages 277-288
ISBN: 978-989-758-772-6; ISSN: 2184-3252

given database before it goes into production.

The dialogue generation tool traverses the database schema and generates a set of joins that induce a natural and coherent conversation flow. Furthermore, the set of dialogues the tool generates covers all database tables. The evaluation agent simulates realistic user interactions, incorporates iterative feedback mechanisms, and leverages LLMs to judge a database conversational agent's ability to align with the user's intentions. The dialogue performance metrics cover the database conversational agent's ability to handle realistic dialogues, and measure if the text-to-SQL tool produced correct SQL queries when compared to a set of ground-truth SQL queries.

The second contribution of the paper is a proof-of-concept experiment with an implementation of a database conversational agent over two databases, the Mondial database[1] and an industrial database in production at an energy company. The implementation uses the LangGraph ReAct (Reasoning and Acting) template, which features action planning (Yao et al., 2023) and a memory component that stores the conversation history, and LLM-based text-to-SQL tools (Oliveira et al., 2025; Nascimento et al., 2025a). The experiment suggests that the dialogue test datasets the dialogue generation tool creates and the user simulations the evaluation agent implements are valuable in assessing the performance of a database conversational agent over a given database, which is the central problem the paper addresses.

This paper is organized as follows. Section 2 covers related work. Section 3 details the dialogue generation tool. Section 4 focuses on the evaluation agent. Section 5 describes the proof-of-concept experiment. Finally, Section 6 contains the conclusions.

2 RELATED WORK

Conversational Interfaces. *Conversational interfaces* support user dialogues and may be classified as *task-oriented* or *chatbots* (Jurafsky and Martin, 2024). Task-oriented conversational interfaces support user dialogues to accomplish fixed tasks. *Chatbots* are designed to mimic the unstructured conversations characteristic of human-human interaction. The database conversational agent used in the experiment falls into the second category.

Quamar et al. (Quamar et al., 2020) proposed an ontology-driven task-oriented interface, which uses a classifier to identify user intentions and a set of templates to generate database queries. The siwarex platform (Fokoue et al., 2024) enables seamless natural language access to both databases and APIs, typical of an industrial setting, and was tested on an extension of the Spider text-to-SQL benchmark.

Wei et al. (Wei et al., 2024) explored what prompt design factors help create chatbots. The authors assessed four prompt designs in an experiment involving 48 users. The results indicated that the chatbots covered 79% of the desired information slots during conversations, and the designs of prompts and topics significantly influenced the conversation flows and the data collection performance.

There are many libraries to help build chatbots, such as Google's DialogFlow[2], the AmazonLex (LexBot) framework[3], Rasa[4], and the Xatkit Bot Platform[5]. For example, a chatbot that combines the flexibility of an LLM with the LexBot framework is described in (Boris, 2024).

The implementation of the database conversational agent used in Section 5 adopted the LangGraph ReAct agent template[6]. The ReAct ("reasoning and acting") framework (Yao et al., 2023) combines chain of thought reasoning with external tool use. ReAct served as a basis for several conversational frameworks. For example, ReAcTable (Zhang et al., 2024b) is a framework designed for the Table Question Answering task inspired by ReAct and relying on external tools such as SQL and Python code executors. It achieved an accuracy of 68.0% on the WikiTQ benchmark (Pasupat and Liang, 2015). RAISE (Liu et al., 2024) is an enhancement of the ReAct framework and incorporates a dual-component memory system to maintain context and continuity in conversations.

Text-to-SQL Tools. Comprehensive surveys of text-to-SQL strategies can be found in (Hong et al., 2025; Shi et al., 2025), including a discussion of benchmark datasets, prompt engineering, and fine-tuning methods. The Awesome Text2SQL Web site[7] lists the best-performing text-to-SQL tools on several text-to-SQL benchmarks, and the DB-GPT-Hub[8] project explores how to use LLMs for text-to-SQL. It should also be remarked that the text-to-SQL problem is considered far from solved for real-world databases (Floratou et al., 2024; Lei et al., 2025).

The text-to-SQL tools (Nascimento et al., 2025a; Oliveira et al., 2025) used in Section 5 achieved good

[1] https://www.dbis.informatik.uni-goettingen.de/Mondial/

[2] https://github.com/langchain-ai/react-agent

[3] https://aws.amazon.com/lex/

[4] https://github.com/RasaHQ/

[5] https://github.com/xatkit-bot-platform

[6] https://github.com/langchain-ai/react-agent

[7] https://github.com/eosphoros-ai/Awesome-Text2SQL

[8] https://github.com/eosphoros-ai/DB-GPT-Hub

results over challenging benchmarks based on the industrial database and on the Mondial database, respectively.

Benchmarks for Conversational Interfaces. There are currently several datasets appropriate for training and evaluating models for conversational interfaces. Examples are the TREC Conversational Assistance Track (CAsT) and the TREC Interactive Knowledge Assistance Track (iKAT) 2023 (Aliannejadi et al., 2024). MultiWOZ (Budzianowski et al., 2020) is a fully labeled collection of human-human written conversations spanning multiple domains and topics. The task-oriented dialogues were created from task templates using an ontology of the back-end database, which turns out to be very simple. The *Awesome NLP benchmarks for intent-based chatbots Web site*[9] lists benchmarks to evaluate user intent matching and entity recognition.

DialogStudio (Zhang et al., 2024a) unified several dataset collections and instruction-aware models for conversational interfaces. The authors also developed conversational AI models, using the dataset collection to fine-tune T5 and Flan-T5. They also report the use of ChatGPT to evaluate the quality of the responses generated.

Contributions. The above conversational benchmarks typically favor model fine-tuning and are not designed to test conversational agents over more complex databases. Thus, rather than relying on such generic benchmarks, Section 3 introduces a dialogue generation tool to create a dialogue test dataset T_D specifically to assess the performance of a database conversational agent over a given database D. The approach taken is rather different from any of the above efforts to create benchmarks for conversational interfaces, insofar as the dialogue generation tool starts from the selected database D for which one wants to create a conversational interface and uses the database schema and the data stored to synthesize a dataset T_D containing dialogues that traverse D. The tool automatically extends the synthetic dialogues to include the user's intentions and the expected SQL queries. The procedure is generic and automated, working with minimal human intervention.

Section 4 describes an evaluation agent to automate performance assessment, using T_D. The evaluation agent simulates realistic user behavior and systematically interacts with a database conversational agent, querying and providing feedback when necessary, following an experimental protocol that consumes the dialogues in T_D. The evaluation agent incorporates an "AI as Judge" mechanism (Zheng et al., 2023) to assess a database conversational agent's ability to align with the user's intentions.

The performance metrics cover the database conversational agent's ability to handle dialogues involving context-dependent follow-up questions and incomplete information that require inference based on dialogue memory. They also measure if the text-to-SQL tool produced correct SQL queries when compared to a set of ground-truth SQL queries. These metrics are novel and a contribution of the paper to the assessment of database conversational agents working over databases.

The procedure to generate a dialogue test dataset T_D for a given database D and the evaluation agent then help address the problem of evaluating a database conversational agent over D.

The experiment in Section 5 tested a database conversational agent over two databases, the Mondial database and an industrial database in production at an energy company. It adopted the evaluation agent and dialogue test datasets created by the dialogue generation tool specifically for these databases. Therefore, the experiment illustrates how to evaluate a database conversational agent over a given database using the techniques introduced in the paper.

3 DIALOGUE TEST DATASETS

3.1 Definitions of Dialogue and Extended Dialogue

For the purposes of this paper, a *turn* is either a *user turn* or a *system turn*, each composed of NL sentences, called *utterances*, produced by the user or by the system. An *interaction* is a sequence of two or more turns, starting with a user turn, ending in a system turn, and alternating between a user turn and a system turn. A *dialogue* is a sequence of interactions.

Intuitively, an interaction is a goal-oriented exchange between the user and the system and is typically centered around a specific question or task, which may span several turns. Figure 1 shows a dialogue with four interactions, corresponding to Lines 1–4, 5–7, 8–12, and 13–15. All interactions consist of a user turn followed by a system turn, except that in Lines 8–12, which has two pairs of a user turn followed by a system turn.

The utterances in a user turn express:

- A *greeting*, that optionally signals the start of a dialogue or an interaction.

[9] https://github.com/xatkit-bot-platform/awesome-nlp-chatbot-benchmarks

```
1  User: "Can you show me a list of the major airports,
2  along with the names and capitals of the countries
3  they are located in?"
4  System: [The first 10 major airports are...]
5  User: "Now, can you show me the border details for
6  these countries?"
7  System: [A description of the data requested]
8  User: "I will now focus on The Ville Lumière."
9  System: "I am sorry. I have no location called
10 Ville Lumière."
11 User: "I meant Paris."
12 System: "OK."
13 User: "What is the population?"
14 System: Paris has an estimated population of about
15 2 million residents.
```

Figure 1: Example of a dialogue.

- A *question*, that specifies a database request, such as that in Lines 1–3.
- A *partial question*, that partially specifies a database request and that must be expanded into a (complete) question using the dialogue context, such as that in Lines 5–6 and Line 13.
- A *context modification*, that modifies the next questions in the dialogue, such as that in Line 8.
- A *clarification*, that responds to a request from the system for information, such as that in Line 11.

The utterances in a system turn express:

- An *acknowledgment*, that signals that the system understood the user, such as that in Line 12.
- A *response*, that describes data retrieved from the database, such as that in Lines 14–15.
- A *request*, that asks the user to clarify his utterance, such as that in Lines 9–10.
- An *error message*, that signals when the user's utterance cannot be processed.

The notion of dialogue, introduced above, captures what the user observes when interacting with the interface to retrieve data from the database. Still, it is insufficient to capture what is needed to test a database conversational agent. The notion of dialogue is then extended so that a turn now includes additional fields to help assess the dialogue performance metrics introduced in Section 4.3:

- The *intention*, which is an NL sentence describing what the user's utterance expresses.
- A *ground truth SQL query*, which translates the user utterance to SQL and, therefore, describes the system response when the user utterance is a question or a partial question.

Figure 2 shows, in JSON format, the extended dialogue corresponding to the dialogue of Figure 1 (for brevity, the figure shows only the first two interactions).

3.2 Generation of Dialogue Test Datasets

A natural language database conversational agent must perform two challenging tasks: (1) Transform partial questions into complete questions; and (2) Modify questions to include explicitly defined context. A dialogue test dataset must then contain dialogues that test the ability of an interface to perform such tasks. This section explains how to create dialogues to test an interface for the first task and outlines how to test for the second task.

3.2.1 Testing the Processing of Partial Questions

The strategy to test the processing of partial questions consists of generating dialogues that traverse *valid join combinations*[10]. Given a valid join combination c, a dialogue d_c is generated so that: (1) Each question in d_c is partial, incorporates a new join in c, and has as context the previous questions; (2) The final question corresponds to an SQL query expressing c. Thus, the dialogue d_c progressively incorporates the joins in c to induce a natural and coherent conversation flow. Furthermore, the set of dialogues must cover a diverse set of valid join combinations, that is, it must traverse the database schema.

The process of creating a dialogue test dataset starts by identifying the valid join combinations, which is implemented by an LLM prompt with the following steps:

1. Define a structured format for the LLM output.
2. Instantiate an LLM model with advanced reasoning capabilities.
3. Provide the database schema.
4. Describe the task to the LLM, including examples from the database and explaining how the joins should be constructed.

This approach is based on the premise that an LLM can effectively reason about the schema and the task described to generate a list of valid join combinations, as demonstrated by the successful use of LLMs in tasks such as Named Entity Recognition (Wang et al., 2025).

However, merely selecting a fixed number of join combinations without additional constraints results in

[10] A *valid join combination* is a set of joins that induces a connected subgraph of the referential integrity graph of the database schema.

```
{   "experiment_id": "1",
    "total_expected_interactions": 3,
    "interactions": [
        {
          "utterance": "Can you show me a list of airports along with the names and
          The capitals of the countries they are located in?",
          "intention": "Show me a list of airports and  include the country's name and
          capital for each airport.",
          "ground_truth_sql":
               "SELECT A.NAME AS airport_name, A.CITY AS airport_city,
                       C.NAME AS country_name, C.CAPITAL
                FROM MONDIAL_AIRPORT A JOIN MONDIAL_COUNTRY C
                  ON A.COUNTRY = C.CODE;"
            } },
        {
          "utterance": "Now, can you show me the border details for these countries,
          such as which countries they border and the length of those borders?",
          "intention": "Using the list of countries from the previous query (those with
          airports), show me their bordering countries and the border lengths.",
          "ground_truth_sql":
                "SELECT C.NAME AS country_name,
                        B.COUNTRY2 AS neighboring_country,
                        B.LENGTH AS border_length
                 FROM MONDIAL_COUNTRY C JOIN MONDIAL_BORDERS B
                   ON C.CODE = B.COUNTRY1
                 WHERE C.CODE IN (SELECT DISTINCT COUNTRY
                                  FROM MONDIAL_AIRPORT);"
            } },
            ...
```

Figure 2: Example of an extended dialogue.

an imbalanced dataset. Some tables become overrepresented, while others remain underrepresented or absent. This imbalance can introduce bias in evaluation, affecting the database dialogue interface's ability to generalize across different SQL queries. A structured strategy for generating balanced join combinations is then introduced, ensuring that:

- Each join combination results in a unique extended dialogue.
- The number of joins per dialogue varies to cover different levels of complexity.
- The selection of join combinations maintains a balanced distribution of tables, preventing biases in dataset representation.

After implementing the balanced join selection strategy, the final join combinations are evaluated based on three main criteria: (a) ensuring full schema coverage; (b) achieving a balanced distribution of the number of joins per combination; (c) distributing table usage fairly to avoid overuse of specific tables.

The final process of creating dialogues to test the processing of partial questions is implemented by an LLM prompt with the following steps (see Figure 3):

1. Extract the required table definitions.
2. Collect sample rows to be incorporated when formulating queries.
3. Load the set C of balanced join combinations.
4. For each join combination c in C, instruct the LLM to create an extended dialogue d_c. The instructions include:
   (a) The table definitions and sample data involved in c.
   (b) A brief explanation of c.
   (c) A clarification indicating that, for each join c_i in c, there should be one interaction in d_c, with just one turn t_i, using the previous interactions as context.
   (d) Indications to include in t_i the user utterance, the intention, and the ground-truth SQL query, and their descriptions.
5. Instruct the LLM on the required output format.

3.2.2 Testing the Processing of Context Modifications

Very briefly, the strategy to test the processing of context modifications consists of generating dialogues that: (1) Define contexts by NL sentences that express SQL restrictions, such as that in Line 8 of Figure 1;

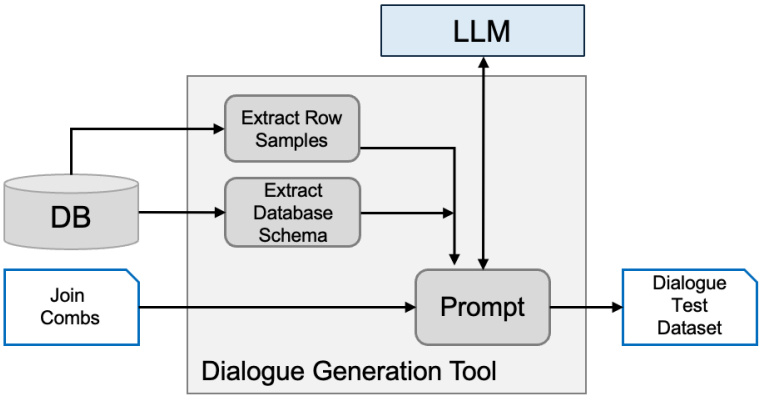

Figure 3: Architecture of the dialogue generation tool.

(2) Include subsequent partial questions that implicitly refers to the context defined, as in Line 13.

The process of creating such dialogues starts by sampling the database schema and the data values to generate restrictions. The sampling process must generate restrictions from a wide selection of tables in the schema, and not concentrate on just a few tables. The process is implemented by an LLM prompt very similar to that which creates dialogues to test the processing of partial questions, and is omitted for brevity.

3.2.3 Constructing the Final Dialogue Test Dataset

Finally, in its simplest form, the final dialogue test dataset will contain dialogues that test the processing of partial questions and dialogues that test the processing of context modifications. A more sophisticated strategy can also be developed by combining both approaches to create dialogues, each of which tests both tasks.

3.2.4 Implementation Details

The construction of the dialogue test datasets used in Section 5 adopted the OpenAI o3-mini model (with high reasoning effort mode) with a Pydantic output template requiring a list of joins. The model yields unique single- and multi-joins with an error rate. Similar structured prompting proved effective for schema reasoning tasks (Wang et al., 2025; Aquino, 2024).

A greedy heuristic incrementally selects joins while optimizing three objectives: (1) 100% table coverage; (2) near-uniform distribution over 2-, 3-, and 4-way joins; and (3) bounded per-table frequency.

For each join, a prompt was assembled containing: the join context; `CREATE TABLE` DDLs; and up to 20 sample values per column. The LLM was prompted to produce k user turns—one per hop—together with validated SQL queries. A Pydantic `Experiment` schema forces compliance.

4 EVALUATION OF DATABASE CONVERSATIONAL AGENTS

4.1 Overview

The methodology uses an *evaluation agent* that simulates realistic user behavior. This agent systematically interacts with the database conversational agent, querying and providing feedback when necessary, following an experimental protocol that consumes predefined extended dialogues from a database test dataset. Such an approach ensures the standardization of interactions, enabling controlled evaluation conditions across different queries and scenarios.

To determine whether the database conversational agent's responses fulfill the intentions, the evaluation agent incorporates an automatic judgment mechanism utilizing an LLM acting as evaluator, which is referred to as the "AI as Judge" methodology. This methodology has been shown to effectively identify alignment and correctness between generated queries and user intentions, reducing human effort and potential biases during the evaluation process (Zheng et al., 2023).

The evaluation agent leverages a state-based evaluation graph implemented using LangGraph, a framework that orchestrates complex interactions among language models, agents, and tools. LangGraph is particularly suitable for managing stateful interactions involving language models and external tools, enabling the precise orchestration of complex conversational evaluation workflows (Chase and Team, 2023).

At the core of the evaluation architecture is a care-

fully designed internal state representation, defined as a strongly typed dictionary that encapsulates all relevant information and parameters needed to manage the evaluation interactions.

To control the evaluation flow, conditional edges determine whether the evaluation process should continue or be concluded. These decisions depend on clearly defined criteria, such as exceeding the maximum allowed retries for a given interaction or completing all intended interactions within an experiment scenario.

To summarize, the implementation of the evaluation agent using LangGraph facilitates precise control over the evaluation workflow. This design decision contributed to the reproducibility, clarity, and scalability of evaluation experiments, offering valuable insights into the performance of a database conversational agent under varied interaction scenarios and conditions.

4.2 Evaluation Agent

Figure 4 shows the architecture of the evaluation agent, illustrating how each interaction progresses through specific nodes and decision points within the evaluation graph.

The evaluation agent receives as input a dialogue test dataset T_D for a database D, prefixed by control data.

It begins at the *Test Control Node*, which first initializes and configures all relevant parameters and states needed for the evaluation, using the control data in the prefix of T_D. The parameters include the maximum allowed retries per interaction, debugging preferences, and model version. The node also initializes detailed metrics and tracking fields, such as interaction metrics, timestamps, and initial debug states. After initialization, the *Test Control Node* passes each extended dialogue d_i in T_D to the *User Interaction Node*.

The *User Interaction Node* simulates user behavior by presenting the database conversational agent with a predefined natural language question from a turn of d_i (the `'utterance'` field in Figure 2). The decision on which natural language question to send to the database conversational agent depends on an internal evaluation state. Specifically, the evaluation could proceed to a new interaction, or feedback could be required, based on the previous responses. The *User Interaction Node* manages these internal decisions based on the value of a Boolean state variable, called `'state["go_next_interaction"]'`.

Upon receiving a response from the database conversational agent, the evaluation moves to the *Check Response Node* to analyze the response (see also the definition of the dialogue performance metrics in Section 4.3):

Successful Intention Alignment: the *Check Response Node* evaluates whether the natural language interpretation u' of the user input u successfully aligned with the intention u''. The interpretation u' is the NL sentence sent to be translated to SQL to create a response for u, if u is a question or partial question, or the NL sentence sent back to the user, in all other cases. The intention u'' associated with u is obtained from the `'intention'` field (see Figure 2). The NL sentences u' and u'' are compared using the "AI as Judge" approach, employing an LLM (Zheng et al., 2023).

SQL Query Correctness: the *Check Response Node* evaluates whether the SQL query Q_{SQL}, generated by the database conversational agent, reflects the SQL ground truth query Q'_{SQL}, obtained from the `'ground_truth_sql'` field (see Figure 2). The verification process compares Q_{SQL} and Q'_{SQL} as in (Nascimento et al., 2025b).

If the database conversational agent's response is inadequate, the *Check Response Node* records the cause of failure (e.g., alignment failure, SQL correctness failure, decoding errors), incrementing the retry count accordingly. At the end, the *Check Response Node* updates several variables that will define the following steps, such as `'state["proceed"]'` and `'state["go_next_interaction"]'`.

The final step of the evaluation workflow for d_i involves determining whether the evaluation of d_i should continue to the next interaction or be concluded. This decision is primarily based on two criteria:

Retries Exceeded: If the maximum number of retries for a particular interaction is exceeded without achieving a satisfactory response, the evaluation process terminates for that interaction, recording it as unsuccessful.

Completion of Interaction Sequence: If all predefined interactions within d_i have been successfully evaluated or have reached termination conditions, the evaluation of d_i is concluded.

Finally, the evaluation agent outputs an *evaluation report* with the result of processing the dialogue test dataset T_D. The report contains JSON objects, one for each extended dialogue in T_D, with `experiment_id`, `total_expected_interactions`, and an `interactions` list. Every interaction contains the user's utterance, a refined *intention*, and a gold

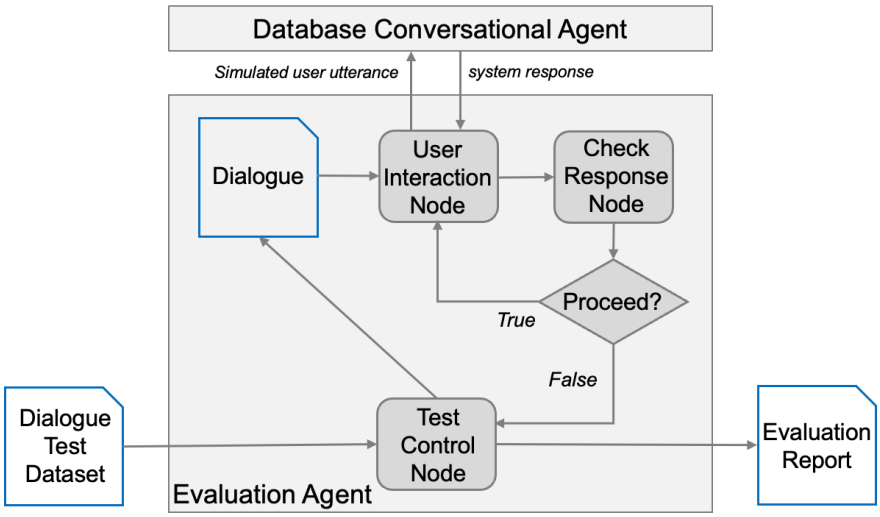

Figure 4: The architecture of the evaluation agent and its relation to the database conversational agent.

SQL query. This structure permits computing the dialogue performance metrics for T_D.

4.3 Dialogue Performance Metrics

The dialogue performance metrics should cover the database conversational agent's ability to handle realistic dialogues, especially those involving context-dependent follow-up questions and incomplete information that require inference based on dialogue memory. They should also measure if the text-to-SQL tool produced correct SQL queries when compared to a set of ground-truth SQL queries.

Given a dialogue test dataset T_D for a database D, the following metrics cover this evaluation scenario.

SQL Query Correctness Rate: The percentage of questions in the user turns of the extended dialogues in T_D that were correctly translated to an SQL query.

The notion that a user question was correctly translated into an SQL query was explained in Section 4.2.

User Turn Intention Alignment Rate: The percentage of *successfully aligned* user turns of the extended dialogues in T_D.

The notion that a user turn was successfully aligned was also explained in Section 4.2.

Dialogue Intention Alignment Rate: The percentage of *successfully aligned* extended dialogues in T_D. An extended dialogue d_i is *successfully aligned* iff every interaction in d_i has at least one successfully aligned user turn.

Note that there might be several user turns in an interaction before reaching a correctly aligned one, but these extra user turns do not affect the dialogue intention alignment rate, although they are accounted for in the user turn intention alignment rate. Therefore, a dialogue is considered successful only when all its interactions have been adequately understood and handled by the agent. This stringent measure ensures that the agent can comprehend and appropriately respond to all user requests within a dialogue.

Average Number of Turn Pairs per Interaction: The average number of turn pairs (a user turn followed by a system turn) per interaction in all dialogues in T_D.

This metric quantifies the efficiency of interactions in terms of turns, providing insight into how many exchanges are typically needed to satisfy a user request. A lower average indicates better user intention comprehension, as the agent can understand and execute the appropriate action with fewer clarifications or follow-ups.

5 A PROOF-OF-CONCEPT EXPERIMENT

5.1 A Database Conversational Agent Implementation and a Baseline Agent

Figure 5 depicts the architecture of the database conversational agent used in the experiments, organized around the following components:

- *Dialogue Control Agent*: controls the dialogue, reformulates the user question, stores the dialogue state, and interacts with the text-to-SQL tool.

- *Text-to-SQL Tool*: controls the compilation of a user question into SQL and passes the results back to the dialogue control agent. It has three major components (Shi et al., 2025): *Schema-Linking*, *SQL Compilation*, and *SQL Execution*.
- *LLM*: executes the tasks passed, guided by prompts. The dialogue control agent and the text-to-SQL tool may use different LLMs.

As mentioned in Section 2, the implementation of the dialogue control agent was based on the LangGraph ReAct agent template, which provides short- and long-term memory to retain data from recent and past user sessions. The experiment adopted a different LLM-based text-to-SQL tool tuned to each database (Coelho et al., 2024; Oliveira et al., 2025).

The experiment compared the database conversational agent implementation just described against a *baseline agent* without dialogue memory to demonstrate the critical importance of context retention in conversational interfaces.

5.2 Experiment with the Mondial Database

5.2.1 The Mondial Dialogue Test Dataset and Use of the Evaluation Agent

The experiments used a *Mondial dialogue test dataset*, generated from the Mondial database, as explained in Section 3.2, with 50 extended dialogues and a total of 149 interactions, where 17 extended dialogues have two interactions, 17 have 3, and 16 have 4. All 40 Mondial tables were covered, with a standard deviation of 1.8 in table frequency. Compared to a naive sampling baseline, coverage improved from 68% to 100% of the Mondial tables.

The test dataset was manually analyzed to verify how much its dialogues covered the Mondial schema, how realistic the automatically generated user utterances and intentions were, and whether the automatically synthesized SQL ground truth queries were correct. The analysis indicated that the dialogues covered the Mondial schema, the user utterances and intentions were adequate, but 5 of the 149 ground truth SQL queries had to be redefined. Minor edits also improved discourse coherence and question variety (aggregations, subqueries, limits).

The experiments evaluated the database conversational agent implementation and the baseline agent with the help of the evaluation agent of Section 4.2, using the Mondial dialogue test dataset. Again, the evaluation logs were manually analyzed to check if the evaluation agent properly verified the intention alignment and the SQL query correctness. The results were manually adjusted, when deemed necessary, before computing the dialogue performance metrics.

A detailed analysis of the Mondial dialogue test dataset and the evaluation agent logs can be found in (Silva, 2025).

5.2.2 Results

Table 1 summarizes the results obtained with the database conversational agent implementation, with dialogue memory, and the baseline agent, without dialogue memory.

User Turn Intention Alignment Rate and SQL Query Correctness Rate. The results demonstrate that the memory-enabled database conversational agent achieved an alignment rate of 100% to properly interpret the user's intention and an SQL query correctness rate of 90%.

In stark contrast, the baseline agent, without dialogue memory, achieved only a 59.7% user turn alignment rate and a 32.6% SQL query correctness rate, highlighting the severe degradation in performance when the dialogue context is not maintained.

Dialogue Intention Alignment Rate. The memory-enabled database conversational agent successfully processed all dialogues (100%), reinforcing its robust ability to understand and correctly process user requests across entire conversations, even in challenging scenarios involving context-dependent and follow-up questions.

By comparison, without dialogue memory, the baseline agent achieved only an 86% dialogue intention success rate, with nearly a third of all conversations failing to meet the user's information needs. This difference emphasizes how crucial conversational memory is for maintaining dialogue coherence and successfully handling multi-turn interactions.

Average Number of Turn Pairs per Interaction. The results show that the memory-enabled database conversational agent reached an average of 1.01 turns per interaction, which is close to the ideal of 1.0. This indicates that the agent almost always correctly understood user requests on the first attempt, rarely requiring additional turns for clarification or correction.

For the baseline agent, without dialogue memory, the results reveal an average of 1.59 turns per interaction. This significantly higher number of turns demonstrates how the lack of dialogue memory impacts dialogue efficiency, requiring more back-and-forth exchanges to satisfy user information needs, and sometimes not helping the agent to understand the user's real intention.

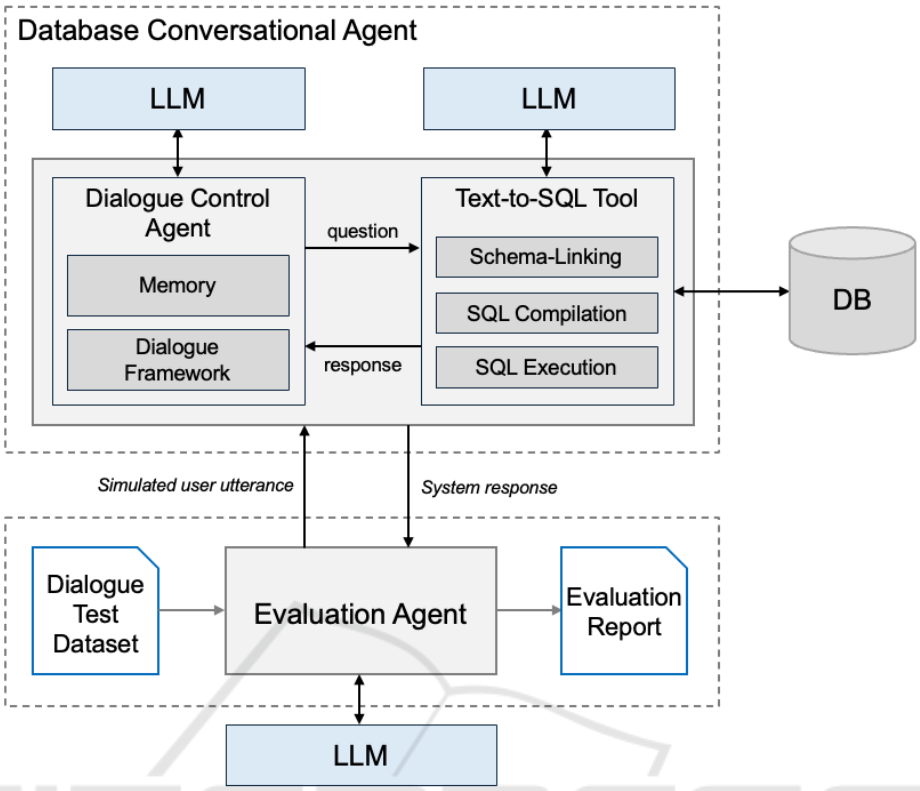

Figure 5: Experimental setup.

Table 1: Results of the database conversational agent for the Mondial database.

Metric	With Memory	Without Memory	Improvement
User Turn Intention Alignment Rate	100.0%	59.7%	+40.3%
SQL Query Correctness Rate	90.8%	32.6%	+58.2%
Dialogue Intention Alignment Rate	100%	68.0%	+32.0%
Average Number of Turn Pairs per Interaction	1.01	1.73	-42.0%

Table 2: Results of the database conversational agent for the industrial database.

Metric	With Memory	Without Memory	Improvement
User Turn Intention Alignment Rate	95.3%	69.0%	26.3%
SQL Query Correctness Rate	82.6%	41.3%	41.3%
Dialogue Intention Alignment Rate	100%	92%	8%
Average Number of Turn Pairs per Interaction	1.06	1.45	-27%

Dialogue Memory Impact Analysis. The comparative results between the memory-enabled agent and the baseline agent without dialogue memory provide clear evidence of the critical role that dialogue memory plays in text-to-SQL interfaces. Without the ability to maintain context across turns, the baseline agent struggled with:

1. Follow-up questions: Unable to connect questions to previous context, leading to misinterpreted intentions.
2. Incomplete queries: Unable to fill in missing information from previous turns.
3. Conversational references: Failed to resolve references like “their” and “that result”.
4. Query refinement: Unable to build upon or refine previous queries.

These challenges resulted in the substantially lower performance metrics observed in the baseline agent, confirming the hypothesis that dialogue memory is essential for building conversational agents.

5.3 Experiment with an Industrial Database

The industrial database chosen for the experiments (Izquierdo et al., 2024) is in production at an energy company and has a relational schema with 27 tables and a total of 585 columns and 30 foreign keys, where the largest table has 81 columns.

The experiments based on the industrial database used the database conversational agent with the KwS-assisted text-to-SQL tool described in (Nascimento et al., 2025a). They applied the evaluation agent of Section 4.2 to a dialogue test dataset generated as explained in Section 3.2, with 50 extended dialogues and a total of 140 interactions, where 10 extended dialogues had two interactions, and 40 had 3.

As in the experiment with Mondial, the dialogue test dataset was manually analyzed. The analysis found that a few golden standard SQL queries were repeated in some turns, which does not represent a problem, and two golden standard SQL queries were incorrect and had to be manually rewritten.

Also, the evaluation agent log was manually analyzed. In the experiment with the database conversational agent with memory, the evaluation agent correctly classified all alignments. In the experiments with the database conversational agent without memory, the evaluation agent correctly clarified most of the user questions in the first attempt, when the dialogue agent returned "*I am sorry, but your question seems incomplete. Could you please provide more details or rewrite the question.*", or when the alignment was false. However, in three cases, the evaluation agent made a mistake when simulating the user, and, consequently, the dialogue control agent and the text-to-SQL tool also failed.

Table 2 summarizes the results of the experiments, which were similar to those in Table 1 and, therefore, require no further comments.

6 CONCLUSIONS

This paper addressed the problem of testing a database conversational agent for a given database D. To achieve this goal, the paper introduced a tool to create a dialogue test dataset T_D for D, a collection of performance metrics, and an evaluation agent that simulates a user interacting with the database conversational agent over D, guided by the dialogue test dataset T_D. This contribution is essential to verify the performance of a database conversational agent over a given database before it goes into production.

The paper concluded with a proof-of-concept experiment with a database conversational agent over two databases, the Mondial database and an industrial database. The implementation used the LangGraph ReAct framework and LLM-based text-to-SQL tools designed for the databases.

The experiment suggests that the dialogue test datasets the dialogue generation tool creates and the user simulations the evaluation agent implements are valuable in assessing the performance of a database conversational agent over a given database, which is the central problem the paper addresses.

Lastly, as future work, experiments along the lines of the setup described in Section 5 will be conducted with other real-world databases that are in production to equip them with a natural language conversational interface.

ACKNOWLEDGEMENTS

This work was partly funded by FAPERJ under grants E-26/204.322/2024, by CNPq under grant 305.587/2021-8, and by Petrobras Contract 2018/00716-0.

REFERENCES

Aliannejadi, M., Abbasiantaeb, Z., Chatterjee, S., Dalton, J., and Azzopardi, L. (2024). TREC iKAT 2023: A test collection for evaluating conversational and interactive knowledge assistants. In *Proc. 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 819–829, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3626772.3657860.

Aquino, P. (2024). How to get structured output from llms applications using pydantic and instrutor. *Medium*. https://medium.com/@pedro.aquino.se/how-to-get-structured-output-from-llms-applications-using-pydantic-and-instrutor-87d237c03073.

Boris, F. B. (Sep 2, 2024). Say goodbye to boring chatbots by combining structure (bot frameworks) and flexibility (llms). *AI Advances*. https://ai.gopubby.com/say-goodbye-to-boring-chatbots-by-combining-structure-bot-frameworks-flexibility-llms-18ae70bc73c5.

Budzianowski, P., Wen, T.-H., Tseng, B.-H., Casanueva, I., Ultes, S., Ramadan, O., and Gašić, M. (2020). Multiwoz – a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. https://arxiv.org/abs/1810.00278.

Chase, H. and Team, L. (2023). Langgraph documentation. https://langchain-ai.github.io/langgraph/.

Coelho, G. M. C. et al. (2024). Improving the accuracy of text-to-SQL tools based on large language models for

real-world relational databases. In *Database and Expert Systems Applications: 35th International Conference, DEXA 2024, Naples, Italy, August 26–28, 2024, Proceedings, Part I*, page 93–107, Berlin, Heidelberg. Springer-Verlag. doi: 10.1007/978-3-031-68309-1_8.

Floratou, A. et al. (2024). NL2SQL is a solved problem... not! In *4th Annual Conference on Innovative Data Systems Research (CIDR '24)*. https://vldb.org/cidrdb/papers/2024/p74-floratou.pdf.

Fokoue, A. et al. (2024). A system and benchmark for LLM-based Q&A on heterogeneous data. https://arxiv.org/abs/2409.05735.

Galimzhanova, E., Muntean, C. I., Nardini, F. M., Perego, R., and Rocchietti, G. (2023). Rewriting conversational utterances with instructed large language models. In *2023 IEEE/WIC Int'l. Conf. on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 56–63. doi: 10.1109/WI-IAT59888.2023.00014.

Hong, Z., Yuan, Z., Zhang, Q., Chen, H., Dong, J., Huang, F., and Huang, X. (2025). Next-Generation Database Interfaces: A Survey of LLM-based Text-to-SQL. http://arxiv.org/abs/2406.08426.

Izquierdo, Y. et al. (2024). Busca360: A search application in the context of top-side asset integrity management in the oil & gas industry. In *Proc. 39th Brazilian Symposium on Data Bases*, pages 104–116, Porto Alegre. SBC.

Jurafsky, D. and Martin, J. H. (2024). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. Online manuscript released August 20, 2024, 3rd edition. https://web.stanford.edu/jurafsky/slp3/.

Lei, F., Chen, J., Ye, Y., Cao, R., Shin, D., SU, H., SUO, Z., Gao, H., Hu, W., Yin, P., Zhong, V., Xiong, C., Sun, R., Liu, Q., Wang, S., and Yu, T. (2025). Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=XmProj9cPs.

Liu, N., Chen, L., Tian, X., Zou, W., Chen, K., and Cui, M. (2024). From LLM to conversational agent: A memory enhanced architecture with fine-tuning of large language models. *CoRR*, abs/2401.02777. doi: 10.48550/ARXIV.2401.02777.

Nascimento, E. R., Avila, C. V., Izquierdo, Y. T., Garcia, G. M., Feijó, L., Facina, M. S., Lemos, M., and Casanova, M. A. (2025a). On the text-to-SQL task supported by database keyword search. In *Proc. 27th International Conference on Enterprise Information Systems - Volume 1: ICEIS*, pages 173–180. INSTICC, SciTePress. doi: 10.5220/0013126300003929.

Nascimento, E. R., García, G. M., Izquierdo, Y. T., Feijó, L., Coelho, G. M., Oliveira, A. R., Lemos, M., Garcia, R. S., Leme, L. A. P., and Casanova, M. A. (2025b). LLM-based text-to-SQL for real-world databases. *SN COMPUT. SCI.*, 6(2). doi: 10.1007/s42979-025-03662-6.

Oliveira, A. et al. (2025). Small, medium, and large language models for text-to-SQL. In *Conceptual Modeling*, pages 276–294, Cham. Springer Nature Switzerland. doi: 10.1007/978-3-031-75872-0_15.

Pasupat, P. and Liang, P. (2015). Compositional semantic parsing on semi-structured tables. In *Proc. 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Int'l. Joint Conf. on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics. doi: 10.3115/v1/P15-1142.

Quamar, A. et al. (2020). An ontology-based conversation system for knowledge bases. In *Proc. 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 361–376, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/3318464.3386139.

Shi, L., Tang, Z., Zhang, N., Zhang, X., and Yang, Z. (2025). A survey on employing large language models for text-to-SQL tasks. *ACM Comput. Surv.* doi: 10.1145/3737873.

Silva, M. O. (2025). Automation in the generation and evaluation of dialogues for text-to-SQL systems: An agent-based approach. Master's thesis, Department of Informatics, PUC-Rio, Rio de Janeiro, Brazil.

Wang, S., Sun, X., Li, X., Ouyang, R., Wu, F., Zhang, T., Li, J., Wang, G., and Guo, C. (2025). GPT-NER: Named entity recognition via large language models. In Chiruzzo, L., Ritter, A., and Wang, L., editors, *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4257–4275, Albuquerque, New Mexico. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.239.

Wei, J., Kim, S., Jung, H., and Kim, Y.-H. (2024). Leveraging large language models to power chatbots for collecting user self-reported data. *Proc. ACM on Human-Computer Interaction*, 8(CSCW1):1–35. doi: 10.1145/3637364.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023). REACT: Synergizing reasoning and acting in language models. In *Proc. 11th International Conference on Learning Representations (ICLR 2023)*.

Zhang, J., Qian, K., Liu, Z., Heinecke, S., Meng, R., Liu, Y., Yu, Z., Wang, H., Savarese, S., and Xiong, C. (2024a). DialogStudio: Towards richest and most diverse unified dataset collection for conversational ai. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 2299–2315, St. Julian's, Malta. Association for Computational Linguistics. https://aclanthology.org/2024.findings-eacl.152/.

Zhang, Y., Henkel, J., Floratou, A., Cahoon, J., Deep, S., and Patel, J. M. (2024b). ReAcTable: Enhancing react for table question answering. *Proc. VLDB Endow.*, 17(8):1981–1994. doi: 10.14778/3659437.3659452.

Zheng, H., Liu, Y., Ge, Y., Awadallah, A. H., and Gao, J. (2023). Judging LLM-as-a-judge: Evaluating large language models for summarization evaluation. In *Proc. 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023)*, pages 11034–11056. https://aclanthology.org/2023.acl-long.616.