DynamicZ: A Robust Evaluation Framework for Text-to-SQL Systems

Daniel Zobl[1], Lukas Kuhn[1] and Christian Ploder[1]

[1]MCI Internationale Hochschule GmbH, Universitätsstraße 15, 6020 Innsbruck / Austria

Abstract

Natural language database interfaces have been a subject of scholarly pursuit for several decades. Recent advancements in large language models have fundamentally shifted the capabilities of analytical systems. These technologies change the paradigm of data accessibility. They enable non-technical stakeholders to interact with complex data structures through intuitive queries. This process democratizes data access within organizational contexts. However, current evaluation methodologies remain largely fragmented. They are frequently disconnected from the practical requirements of enterprise environments. This research addresses the opaque nature of modern AI systems by introducing DynamicZ. This robust evaluation framework was developed through a synthesis of technical performance metrics and multi-dimensional stakeholder perspectives. The framework moves beyond the traditional reliance on binary accuracy scores. It incorporates measures of test set diversity via a metric for the randomness of category distribution. It also includes statistical reliability through a dedicated measure for confidence. These elements are consolidated into a composite metric known as the DZ-Score. This allows for a balanced assessment of a system's readiness for production. A critical finding is that the framework is fundamentally industry and technology agnostic. While validated within a Text-to-SQL context, its underlying logic is applicable to any natural language-driven data analysis approach. The evaluation categories must simply be formulated accordingly. This adaptability ensures the solution provides a replicable methodology for organizations evaluating evolving systems in real-world scenarios. The framework was validated in collaboration with a large food distribution enterprise operating a fleet of more than 12,000 vehicles and employing over 20,000 staff. Currently the framework is deployed in multiple large enterprises, including a building material and an auto-parts company which will enable us to validate its applicability across additional domains and industries. This paper extends previous work by one of the authors, in which the framework was originally introduced [1].

Keywords

Text-to-SQL, Large Language Model Evaluation, Case Study Research

CEUR-WS.org/Vol-4336/paper4.pdf

1. Introduction

The demand for database interfaces that process natural language has endured for years [2]. This need persists despite historical hesitations regarding system consistency. Earlier iterations of technology relied heavily on human-instructed rule settings or rigid ontologies. The current landscape is being reshaped by generative artificial intelligence [3]. These modern systems offer the ability to democratize data access for non-specialist users. They can automate the creation of complex queries that would otherwise require substantial technical expertise. However, the transition to widespread industrial implementation is hindered by a gap in implementation readiness. A notable contradiction is evident in current industry statistics. A vast majority of the global workforce expresses interest in utilizing AI to automate tasks. However, few organizational leaders feel confident in measuring the productivity benefits or the reliability of these systems. Executives frequently express concern regarding the inherent opacity of these systems. They often lack the transparency and auditability required for high-stakes business environments [4]. Existing systems often fail to quantify their own confidence in a generated output. This leads to speculative results that compromise data integrity and user trust. The DynamicZ framework was designed within a case study context to bridge this gap. It provides an objective evaluation that accounts for technical accuracy and practical organizational needs. The framework is designed to remain relevant as technological paradigms evolve. It offers a methodology that is

SQAMIA 2026

daniel.zobl@mci.edu (D. Zobl); lukas.kuhn@mci.edu (L. Kuhn); christian.ploder@mci.edu (C. Ploder)

0000-0002-7064-8465 (C. Ploder)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-19

not restricted to any single technology. This paper makes three primary contributions to the field of natural language database interfaces. First, it introduces DynamicZ, a practical and architecture-agnostic evaluation framework built to close the gap between academic benchmarks and the real-world requirements of enterprise Text-to-SQL deployments. Second, it establishes the DZ-Score, a composite metric that moves beyond accuracy by combining empirical execution performance, statistical confidence bounds, and the categorical diversity of the test set into a single, interpretable measure. Third, the research validates these contributions through an actual enterprise deployment within a large-scale logistics operation, providing concrete evidence of the framework's practical applicability, the scalability of its evaluation pipeline, and the operational independence it enables for organizations managing evolving AI systems.

2. Theoretical Background

The theoretical foundation of natural language data analysis has evolved toward advanced transformer-based architectures [5] [6]. Modern models utilize generic representations that are transferable across various applications. This enables a new paradigm of prompt engineering and in-context learning [7]. Despite these technological leaps, real-world deployment faces hurdles. These include semantic errors and flawed logic in data aggregation. Contextual misinterpretation can lead to misleading business insights [8].

2.1. Shortcomings of Traditional Evaluation Metrics

Conventional metrics used to assess system performance have significant limitations. Most assessments in the Text-to-SQL domain rely on indicators like Execution Accuracy and Exact Matching Accuracy. Execution Accuracy verifies whether the predicted query returns the correct result set. However, it is susceptible to false positives. Semantically distinct queries can yield identical output data depending on the state of the database. Exact Matching Accuracy compares the syntactic structure of the queries. This often results in false negatives. Semantically equivalent queries can be written in multiple syntactically valid ways [9]. Current research indicates that models may perform well on academic benchmarks [10] but fail against real-world queries containing joins [11]. Real-world tasks involve complex operations like joins and nested subqueries. This discrepancy highlights the necessity for evaluation frameworks that go beyond binary success rates. Assessments must include considerations of complexity and user-related requirements. There is also a growing emphasis on Explainable AI. Users require intuitive justifications for system conclusions to build the trust necessary for adoption [12].

2.2. Statistical Reliability and Test Set Diversity

Evaluation frameworks must incorporate statistical rigor to bridge the gap between technical results and user expectations. A robust measure for confidence is essential. It provides a conservative estimate of system performance. This is particularly important when dealing with limited sample sizes. This measure ensures that stakeholders are informed about the lower bounds of expected reliability. The diversity of the test set itself is also a critical factor. A dedicated measure for the randomness of category distribution is needed. It analyzes how well query types are distributed within a test set. This prevents the inflation of performance scores through the use of non-representative test data. This integration of accuracy and diversity forms the conceptual core of the DynamicZ framework.

3. Methodology

This research utilizes a mixed-methods case study approach. The case study method is appropriate because system performance is influenced by many factors. These include data schema complexity and varying levels of user expertise. The study was conducted in collaboration with a commissioning company. This provided a real-world environment to test the framework's logic against practical

constraints. The methodology combines quantitative system metrics with qualitative insights and archival records. This produces a convergent line of evidence. The research employs a mixed-methods approach combining explanatory and exploratory methodologies based on case study research by [13] synthesizing technical metrics with qualitative stakeholder feedback. The primary research question for this thesis is: How can a robust and practically applicable evaluation framework for Text-to-SQL systems be developed that considers both performance and stakeholder feedback?

3.1. Inductive Development and Triangulation

The development of the DynamicZ framework was exploratory. It aimed to uncover the types of errors and performance factors that emerge in a practical environment. A key component was the inductive formation of evaluation categories. Rather than imposing predefined classifications, categories emerged from a systematic analysis of literature and empirical observations. This approach ensures that the framework is closely aligned with actual workflows. Data triangulation was achieved through the collection of multiple sources. These included archival records of system logs and technical test results. Qualitative stakeholder consultations were also conducted. These involved guided conversations with internal stakeholders across various roles. Participants included product managers, engineers, and sales consultants. These conversations focused on perceptions of system consistency and adoption barriers. This iterative approach allowed for the refinement of the framework's logic. Technical metrics remained contextualized within the broader perspective of user experience.

3.2. Category Development

The framework was designed to be adapted to various technical implementations. This was achieved by focusing on the logic of category formation for test set questions as well as the answers. The methodology accounts for varying levels of query difficulty and complexity. The framework pinpoints specific architectural weaknesses regardless of the underlying technology. This granular approach provides an actionable assessment that remains valid as new analytical methods emerge.

4. Evaluation Framework Development

The framework developed aims to be applicable across all domains and to enable users in companies to systematically evaluate their individual Text-to-SQL systems and places particular emphasis on practical applicability. The metrics used are suggested and supported by current and relevant technical literature. The framework can be used regardless of whether a gold standard of SQL queries already exists or not. System log files are analyzed as part of this approach. Question and answer categories are derived in a way that may allow the framework to be applied to other domains. In addition, measures are combined that not only take into account the performance of the model, i.e. the proportion of correct queries, but also evaluate the scope and distribution of the questions contained in the test set. This is intended to provide a holistic view of the model to be evaluated. Depending on the size of the dataset and the resources available, classification can be performed via vectorization or manually by mutual agreement.

4.1. Classification of questions

The questions are divided into different categories to subsequently evaluate the distribution of the questions to ensure that the test set is robust and that the statements on the accuracy of the model are also valid. The questions were divided into the following categories:

Basic Data Retrieval describes queries that retrieve simple questions from one or more tables and use conditions such as `SELECT` statements, `WHERE` clauses for filtering, `JOIN` operations to connect tables, and `LIMIT` to restrict results. No advanced transformations are executed.

Aggregations & Rankings involve aggregations using `SUM()`, `AVG()`, `COUNT()`, `MIN()`, `MAX()`, or window functions using `ROW_NUMBER()`, `RANK()`, `DENSE_RANK()` often combined with `GROUP BY`, `HAVING`, `ORDER BY`, or ranking functions.

Advanced Analytics describes queries with multiple filtering conditions, conditional logic (`CASE WHEN`), and complex subqueries (`EXISTS`, `IN`). These queries may involve multiple levels of nesting, complex joining patterns, and advanced filtering logic. Predictive queries are also summarized in this category.

Time-Based & Sequential Queries use date and time information with functions like `DATE`, `TIMESTAMP`, `DATEDIFF`, and `EXTRACT(YEAR/MONTH)`. These queries may employ `LIMIT` and `OFFSET` for pagination and implement ordering based on temporal attributes. This category enables trend analysis over defined periods and chronological sequencing of records.

Metadata & Structural Queries request information about the database architecture rather than its content. These queries examine database schema, table structures, column definitions, relationships, and constraints. This category facilitates database exploration and structural analysis tasks.

Out of Scope queries attempt to access data that does not exist within the available dataset. These queries request information outside the database's content domain or try to perform operations on non-existent tables or columns. For a practical use-case it is particularly important to add a category with unanswerable questions, as questions of real-world users would be up to 20% of the overall volume [14].

Entity Referencing covers queries that identify database entities through different naming conventions, synonyms, or descriptive phrases. These queries require semantic understanding to recognize when different terms refer to the same database object.

Specialized Output Formats include queries that specify the output for returned data. These queries define output specifications such as PDF generation, CSV formatting, or custom report layouts, focusing on how data should be formatted and presented.

4.2. Policy on question classification

While for the majority of questions it will be evident which category should be assigned, a systematic approach is required for handling ambiguous cases to ensure mutual exclusivity across classifications. Derived from the structured model to eliminate ambiguities related to Text-to-SQL query generation [15], a structured policy was developed. The methodology operates on a straightforward principle: When a query satisfies the criteria for multiple categories simultaneously, it is exclusively assigned to the category with the highest priority in the established sequence, and all lower-priority categories are automatically disregarded. This cascading decision architecture creates a filtering mechanism where each query is evaluated first against the highest-priority category criteria, and only proceeds to lower-priority evaluations if it fails to meet the requirements of higher ones.

1. Out of Scope: First, determine if the query is answerable via the database. If a question cannot be answered with the given data, as additional knowledge may be needed, or it requests an unsupported operation, it is immediately categorized as Out of Scope. Rationale: Out of scope queries must be caught first to exclude them from the evaluation of test set distribution as they don't belong in any content category.

2. Specialized Output Formats: If in scope, check for special output requests. If the request asks for a particular output format or presentation (for example, asking for results in JSON, CSV or PDF), classify it under Specialized Output Formats. Rationale: Formatting requests override content-based categorization because they require special handling beyond standard SQL query generation.

3. Entity Referencing: If no special format, check for entity references or context dependence. If the query includes references that require resolving an entity or value from context (e.g. referring to a person using only a part of the name), assign it to Entity Referencing.

4. Metadata & Structural Queries: If no entity/reference issue, check if it is a metadata inquiry. If the question is about the database itself or its structure (schema, tables, columns) rather than the data content, categorize it as Metadata & Structural Queries.

5. Time-Based & Sequential Queries: If not a structural query, check for time-based or sequential aspects. If the query's primary focus is on time, chronology, or sequence (e.g. trends over time, sequential order, comparisons across dates, or questions like "first/last occurrence"), classify it as Time-Based & Sequential Queries.

6. Advanced Analytics: If not time-based, check for advanced analytical complexity. If the query demands complex analysis or multi-step reasoning beyond a straightforward aggregation, assign it to Advanced Analytics. This category covers cases like nested queries, sub-queries, window functions, calculations of ratios or differences, predictive or statistical questions, and multi-faceted analysis (e.g. "Find the correlation between X and Y" or "For each category, what is the percentage of total sales?").

7. Aggregations & Rankings: If none of the above categories apply, check for aggregation or ranking. If the query asks for any form of summary statistic or ranking without the complexities above, categorize it under Aggregations & Rankings. This includes questions involving SQL aggregate functions (COUNT, SUM, AVG, MIN, MAX) or requests for top/bottom results, or sorted order by some metric (e.g. "Who is the highest paid employee?", "List the top 5 products by revenue"). Rationale: These are very common query types focusing on summarized data or ordered results.

8. Basic Data Retrieval: If none of the above conditions were met, the query is a simple data lookup or retrieval. This is the default category for straightforward queries that only retrieve data with minimal processing. Examples include fetch requests like "Show all records from table X", "Find employees where department = 'Sales', or simple joins ("List customers with their orders"). Rationale: Any query that isn't caught by a more specific category is, by elimination, a basic retrieval.

4.3. Classification of Answers

To calculate model accuracy while also identifying and addressing common error patterns, responses require categorization beyond simple right-or-wrong classifications. This approach enables specific analysis of frequently occurring mistakes, similar to the framework presented by Shen et al. [16], which established seven overarching error categories for comprehensive evaluation. During the creation of the evaluation framework, 15 different categories emerged, with the first three being classified as correct and the other 12 as incorrect.

Correct Answer - The query is recorded semantically correctly and the query created returns the desired result. The syntax of the query may differ, provided this has no impractical side effects on the computation time or data retrieval volume. The query retrieves the requested data without errors or extraneous operations.

Correct After Reasonable Follow-up question - This category represents cases where the system required additional information or feedback to produce the proper query, but ultimately delivered the correct solution. As is also the case with the correct queries, the syntax of the query may differ, provided this has no impractical side effects on the computation time or data retrieval volume.

Recognized Out of Scope - The system correctly identifies that the question cannot be answered using the available database schema and responds appropriately. The response includes an explanation of why the request exceeds the capabilities of the current database structure, potentially suggesting alternatives if applicable. If the system correctly recognizes a query as out of scope, the answer should be rated as correct in the evaluation.

Unrecognized Out of Scope - The system fails to recognize that the question references data not present in the database, producing inappropriate responses. This results in incorrect queries that attempt to access non-existent data or perform operations that cannot be completed with the available schema.

Semantically Correct with Impractical Side Effects - The SQL query returns the correct results but causes inefficiencies or impractical side effects, which may include but are not limited to excessive

computation time or data retrieval volume. These queries may use suboptimal join methods, unnecessary table scans, or fail to utilize available indexes, leading to performance issues despite technical correctness. This also includes cases where the system creates a query for a question that could be answered without one.

Syntax Error - The generated SQL contains syntax mistakes that prevent it from executing in a database system. These errors include malformed statements, incorrect keyword usage, missing parentheses, or other structural issues that violate SQL rules.

In Scope but Misleading / Logical Error - The SQL executes without error but returns incorrect data due to logical flaws in query construction. These queries may have incorrect join conditions, filtering criteria, or aggregation logic that leads to results that do not match the intended request.

In Scope but Obviously Wrong / Logical Error - The SQL executes but produces obviously incorrect results due to a fundamental misunderstanding of the request. These queries display a significant conceptual error in interpreting what information was sought, resulting in incorrect output.

Incomplete Query - The SQL contains some correct elements but lacks essential components needed to fulfill the request. This includes queries missing crucial join conditions, filtering criteria, or output specifications that would be necessary to produce the expected results.

Reasonable Follow-up Question but wrong result - The system asks a relevant clarifying question that helps refine the query requirements. These questions address genuine ambiguities in the original request and seek information that would improve the system's ability to generate the correct query. Nevertheless, the final result is incorrect, which indicates that the system understands the context. If the result is incorrect, the entire query is classified as incorrect.

Unreasonable Follow-up Question - The system asks questions that are irrelevant or demonstrate a misunderstanding of the user's needs. These questions may seek information already provided, address aspects unrelated to the query, or show confusion about fundamental database concepts. As soon as the system asks an unreasonable follow-up question, the output is classified as wrong regardless of the final result.

Invalid Reference - The SQL refers to tables or columns that do not exist in the database schema. The query attempts to access database objects that are not present in the available tables, demonstrating a misunderstanding of the available data structure.

No Response - The system fails to generate any SQL output for the given request. This represents a failure to process the input or produce any query attempt, leaving the user without any response.

Error - The system experiences an unexpected failure or crash during query generation. This category covers technical failures within the system itself rather than incorrect query generation.

Hallucination - The system generates fabricated results or data that have no correlation to the actual database content.

4.4. Accuracy evaluation

The next step is to evaluate the accuracy of the model by analyzing how the response categories are distributed.

$$\text{Accuracy} = \frac{Q_{\text{correct}}}{Q_{\text{total}}}$$

- Q_{correct}: The number of queries that are correct.
- Q_{total}: The total number of queries.

In a further step, the accuracy of the results is measured against the question categories. From this, it can be deduced whether the model has weaknesses in relation to certain types of questions.

$$\text{Share}_{\text{desired}} = \frac{Q_{\text{desired}}}{Q_{\text{total}} - Q_{\text{OOS}}}$$

- Q_{desired}: The number of queries (or questions) that belong to the desired category.
- Q_{total}: The total number of queries (or questions).
- Q_{OOS}: The number of Out of scope queries (or questions) that should be excluded.

4.5. Normalized entropy for distribution analysis

Entropy is defined as a measure of the uncertainty or the average information content in the message selection process [17].

By calculating entropies in machine learning models, the accuracy of the model can be put into relation and statements can be made about its generalizability. Furthermore, this can help to understand the dataset and make the inequality and uncertainty of the dataset quantifiable. In reinforcement learning, entropy acts as a key regulator of an agent's decision-making strategy. It balances exploration and exploitation by measuring the randomness of the agent's policy, particularly crucial in uncertain or dynamic environments. Higher entropy indicates more exploratory behavior, which is especially valuable during the initial stages of learning [18]. Entropy also has practical applications in the training of deep learning models, as it can reduce redundancies and increase the efficiency of the model [19].

The Shannon entropy H for the evaluation framework is used as follows:

$$H = -\sum_{i=1}^{n} p_i \log_b(p_i),$$

where:

- n is the number of categories,
- p_i is the probability of the ith category,
- $\log_b$ denotes the logarithm with base b (commonly, $b = 2$ for bits, $b = e$ for natural logarithms, or $b = 10$).

Normalized entropy is used to make the results of LLMs comparable across different generations [20]. Values between 0 and 1 are always obtained by dividing the result of the calculated entropy by the maximum entropy $\log_b(n)$. Of the three metrics tested, normalized entropy consistently proved to be the best confidence indicator in certain domains such as relational reasoning tasks. A maximum entropy score measures sequence uncertainty, making it particularly well-suited for sequential NLP tasks [21]. The idea is also taken up in Confidence Estimation for EHRSQL [22], a benchmark for Text-to-SQL in the area of electronic healthcare data. Here, the maximum possible entropy is taken as a reference to obtain a normalized uncertainty per model response. This makes it possible to define a score always in the range from 0 to 1, where 0 corresponds to no uncertainty and 1 corresponds to the maximum uncertainty.

The normalized Shannon entropy H_{norm} for the evaluation framework would be the following:

$$H_{\text{norm}} = \frac{H}{\log_b(n)} = \frac{-\sum_{i=1}^{n} p_i \log_b(p_i)}{\log_b(n)},$$

4.6. Confidence Intervals & Wilson Confidence Interval (CI) for test set adequacy

A point estimate of performance lacks meaningful insight when isolated from its statistical context. To properly understand and evaluate such an estimate, one must accompany it with a measure of precision, typically expressed through confidence intervals (CI) [23].

Normal Approximation Confidence Interval

$$\text{CI} = p \pm z\sqrt{\frac{p(1-p)}{n}}$$

- p: The observed accuracy of correct results,

- n: The total number of observations or queries,
- z: The critical value from the standard normal distribution corresponding to the desired confidence level.

Wilson developed a new calculation method of the confidence interval which improved its reliability [24]. Let

$$p_0 = \frac{S}{n}$$

be the observed proportion of successes (with S successes in n trials) and define

$$q_0 = 1 - p_0.$$

for a chosen critical value X (analogous to the z-value for a given confidence level), set

$$t = \frac{X^2}{n}.$$

Then, the Wilson CI is given by

$$\left[\frac{p_0 + \frac{t}{2} - \sqrt{p_0 q_0 \, t + \frac{t^2}{4}}}{1+t}, \quad \frac{p_0 + \frac{t}{2} + \sqrt{p_0 q_0 \, t + \frac{t^2}{4}}}{1+t} \right].$$

The center of the interval is

$$\frac{p_0 + \frac{t}{2}}{1+t},$$

and the half-width is

$$\frac{\sqrt{p_0 q_0 \, t + \frac{t^2}{4}}}{1+t}.$$

The Wilson CI is especially useful, as it is more robust when testing smaller sample sizes or rather extreme possibilities [25][26].

If we now apply the Wilson CI for the Evaluation Framework the formula would look like this:

$$\text{Wilson CI} = \frac{p + \frac{z^2}{2n} \pm z\sqrt{\frac{p(1-p)}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}}$$

- p: The observed accuracy of correct results.
- n: The total number of observations or queries.
- z: The critical value from the standard normal distribution corresponding to the desired confidence level.

By computing confidence intervals, the relationship between sample size, accuracy, and confidence in results becomes clear: larger samples and moderate accuracy proportions yield narrower intervals, indicating greater confidence, while small samples or extreme proportions produce wider intervals, indicating less confidence.

4.7. DZ-Score

The DZ-Score is a composite metric that provides a comprehensive assessment of Text-to-SQL systems by combining three essential dimensions of performance through accuracy, scope via normalized entropy measuring dataset diversity, and statistical confidence using Wilson CI lower bounds that account for sample size and variability. The DZ-Score weights these components together, balancing accuracy with

Wilson CI lower bounds while incorporating a robustness term from test set diversity. This structure rewards models for high accuracy while ensuring consistent performance across diverse queries under statistically reliable conditions. This comprehensive approach proves particularly valuable for robust evaluation in critical deployments and benchmarking scenarios where generalizability matters. It is defined as:

$$\text{DZ-Score} = w_{ae} \cdot (\lambda \cdot \text{Accuracy} + (1 - \lambda) \cdot \text{CI}_{\text{LB}}) + w_{te} \cdot H_{\text{norm}}$$

Where:

- Accuracy is the observed accuracy of the model (correct queries / total queries).
- CI_{LB} is the lower bound of the Wilson CI for the accuracy at a chosen confidence level (e.g., 95%).
- Normalized Entropy is the normalized Shannon entropy of the test set, measuring diversity in query types.
- $\lambda \in [0, 1]$ balances optimism and conservatism:
  - $\lambda = 1$: only use raw accuracy.
  - $\lambda = 0$: only use the conservative estimate (CI lower bound).
- w_{ae} and w_{te} are weights for predictive performance weighing the accuracy evaluation part and dataset robustness weighing the evaluation evaluation part, respectively, where:

$$w_{ae} + w_{te} = 1$$

Suggested default values:

$$\lambda = 0.5, \quad w_{ae} = 0.7, \quad w_{te} = 0.3$$

Accuracy represents the most extensively researched and routinely assessed characteristic in artificial intelligence systems. AI implementations lack practical value when they fail to meet adequate accuracy thresholds. To establish model robustness, incorporating worst-case accuracy measurements is recommended, as models typically encounter greater challenges during real-world deployment scenarios compared to controlled testing environments [27]. Using the lower confidence limit for performance assessment provides very high certainty that a model will perform at least at that minimum level in real-world implementation [28]. This is especially applicable if the confidence intervals are wider. Throughout the research, the implementation of a 95% confidence interval aligns with established practices documented in LLM and natural language processing literature [29]. The evaluation framework also provides dynamic adaptation capabilities tailored to specific user requirements and priority areas. For high-risk operational environments, the evaluation framework is adjustable to support 99% confidence intervals with corresponding z-values for DZ-Score calculations. This elevated confidence level is appropriate for applications where model failures could result in significant business impact, regulatory non-compliance, or safety concerns, providing more conservative performance estimates and robust reliability guarantees. The framework's flexibility in confidence interval selection enables organizations to align statistical rigor with their specific risk tolerance levels and operational requirements.

The lambda parameter, initially set to 0.5, represents balanced trust between raw accuracy measurements and conservative confidence interval estimates. This parameter controls the relative weight assigned to observed performance versus statistical uncertainty in the final DZ-Score calculation. For safety-critical deployments, lower lambda values of 0.3 or 0.0 can be used, prioritizing trust in the lower bound of the Wilson CI. This conservative approach implements worst-case performance assumptions, minimizing the risk of overestimating model capabilities in critical applications. For early-stage exploration environments, higher lambda values of 0.7 or 0.8 maintain emphasis on raw accuracy while accounting for some statistical uncertainty.

Weighting factors w_{ae} and w_{te} govern the relative importance of different evaluation components within the composite scoring system. These parameters enable customization based on organizational

priorities regarding test set comprehensiveness and normative entropy considerations. When comprehensive coverage across diverse question types is critical, w_{te} can be increased to emphasize normative entropy importance. For specialized applications where specific analytical capabilities outweigh broad coverage, w_{ae} can be adjusted to prioritize raw performance metrics over distributional considerations. Organizations should consider interaction effects between different parameter settings and their cumulative impact on DZ-Score interpretation. In summary, the proposed weightings were therefore used for this use case to slightly emphasize accuracy relative to the test while ensuring a balanced ratio of raw accuracy and confidence.

5. Results and Interpretation

The primary output is the architectural logic of the DynamicZ evaluation framework. This centers on the composite DZ-Score balancing accuracy with statistical certainty and the representativeness of the test data. An isolated accuracy percentage is insufficient for evaluating enterprise readiness. The framework proposes a multi-dimensional assessment instead.

5.1. Empirical Evaluation and DZ-Score Calculation

To evaluate the operational readiness of the developed Text-to-SQL system within the enterprise infrastructure, the evaluation framework was applied to a test set consisting of 84 queries. This test set of questions was created based on the needs of enterprise customers and is intended to be as comprehensive and production-relevant as possible. The baseline structural diversity of the evaluation dataset was calculated using Shannon entropy (H) across the defined question classifications, yielding $H = 2.11$. To ensure comparability, this was scaled against the maximum possible entropy to produce a Normalized Entropy metric of 0.75.

Out of the 84 queries processed, 79 responses satisfied the criteria for correct classification, resulting in an Accuracy of 94.05%. To account for finite sample constraints, a Wilson Confidence Interval (CI) analysis was conducted at a 95% confidence level with a corresponding z-Value of 1.96. This analysis yielded an upper confidence limit of 97.43% and a conservative lower bound of 86.81%.

The mathematical parameters established for the aggregation include an execution weight of $w_{ae} = 0.7$, a dataset robustness weight of $w_{te} = 0.3$, and a calibration parameter of $\lambda = 0.5$. Overall, this resulted in a DZ-Score value of 0.86 for the company model, which points to some resilience in the enterprise Text-to-SQL system, though production deployment would likely require further testing and validation. While the raw accuracy of 94.05% offers an optimistic approach, the final score is anchored by a statistical safety margin via the Wilson lower bound of 86.81%, protecting the organization against worst-case performance drops under finite sample sizes. Crucially, because this level of dependability is achieved over a diverse test set with a Normalized Entropy of 0.75, the 0.86 score demonstrates that the system's precision is architecturally robust across varied business scenarios rather than overfitted to repetitive query structures. This score was calculated using the suggested weightings presented in Chapter 4.7, but may vary if the weightings are adjusted, as shown in the table below.

Table 1
Sensitivity Analysis of the DZ-Score across Parameter Configurations ($n = 84$, Accuracy = 0.94, $\text{CI}_{\text{LB}} = 0.87$, $H_{\text{norm}} = 0.75$)

Deployment Profile	λ	w_{ae}	w_{te}	Accuracy Evaluation	Test Evaluation	DZ-Score
High-Rigor / Risk-Averse	0.00	0.50	0.50	0.87 (CI_{LB})	0.75 (H_{norm})	0.81
Conservative Operational	0.00	0.70	0.30	0.87 (CI_{LB})	0.75 (H_{norm})	0.83
Suggested Default	**0.50**	**0.70**	**0.30**	**0.90 (Balanced)**	**0.75 (H_{norm})**	**0.86**
Execution-Prioritized	0.50	0.90	0.10	0.90 (Balanced)	0.75 (H_{norm})	0.89
Optimistic Benchmarking	1.00	0.70	0.30	0.94 (Raw Acc.)	0.75 (H_{norm})	0.88
Max Raw Performance	1.00	0.90	0.10	0.94 (Raw Acc.)	0.75 (H_{norm})	0.92

5.2. Practical Validation and Scalability

To substantiate the practical efficacy and scalability of the DynamicZ evaluation framework, an empirical validation was conducted in collaboration with a enterprise operating within the food distribution sector. The case study organization maintains a large-scale food distributor in Mexico, managing a workforce of over 20,000 employees and coordinating a fleet exceeding 12,000 vehicles. From a business perspective, the goal was to democratize data access by enabling fleet managers to perform independent data analysis, moving beyond rigid, standardized BI reports. An LLM-driven Text-to-SQL interface was identified as a suitable solution, aligning with literature on enterprise demand for natural language database interfaces. However, deploying such systems in production requires rigorous output validation to ensure data integrity. Accordingly, DynamicZ was iteratively refined to reconcile corporate requirements with the academic and statistical benchmarks from the methodology chapter. Once deployed, it gave the organization operational autonomy, allowing fleet managers to benchmark new LLM iterations against a gold-standard dataset without recurring manual calibration. Expanding test coverage also revealed a plateau effect, where broader test sets initially improve evaluation fidelity, but marginal gains diminish as the Wilson Confidence Interval narrows and the performance lower bound solidifies. The deployment further revealed optimization potential through an LLM-as-a-judge paradigm, which flags systemic errors and enables a feedback loop for targeted reinforcement learning to keep the system aligned with evolving data schemas.

5.3. Industry and Technology Agnosticism

The initial validation was conducted within the context of Text-to-SQL systems. However, the framework is designed to be technology agnostic. The core logic applies to any system where natural language interfaces with data. Balancing accuracy with confidence and category distribution is a universal requirement. Technology agnosticism may apply, as suggested by this case study across different data analysis methods. The agnosticism of the DynamicZ framework was explored through this case study that assessed three different models with different architectures and data processing strategies. To this end, a standardized set of questions was posed to three different models, and the responses were subsequently evaluated. This study involved a company proprietary model, ChatGPT o4-mini-high, and Gemini 2.5 Flash, which both represented the most recent versions available while the research was conducted.

Because the publicly available versions of ChatGPT and Gemini lack direct database connectivity, unlike the enterprise system's native integration, an alternative testing setup was needed. Data was extracted from the company's Snowflake warehouse and downsampled into a CSV file with an accompanying JSON schema, with sensitive entities anonymized via hashing so that queries could reference these values directly. After an initial model choice proved unworkable and was replaced with a more stable alternative, queries were processed in batches, with each model producing both generated code and a final answer. Responses were then reviewed by two independent evaluators and cross-checked through re-execution against the live database.

One of the most relevant findings of the case study concerns the divergent data analysis methods employed by the evaluated models. The proprietary model operated as a native Text-to-SQL system, translating natural language into SQL queries executed directly against the Snowflake warehouse engine. In contrast, ChatGPT o4-mini-high and Gemini 2.5 Flash functioned as Text-to-Answer agents, using Python's pandas library to derive results in-memory rather than querying the database directly. This distinction reflects a fundamentally different execution paradigm: one approach offloads computation to a relational database engine, while the other performs programmatic data manipulation within the language model's own runtime environment. Despite this architectural divergence, the DynamicZ framework applied its evaluation logic uniformly across all three systems. Because the framework's two-component structure, comprising test evaluation and accuracy evaluation was deliberately designed to avoid dependence on SQL-specific execution mechanics, it remained indifferent to whether the underlying target language was SQL or Python. The same question categories, answer classification

rubric, confidence interval calculations, and DZ-Score formula were applied without modification. The framework evaluated what each system produced, not how it produced it, and this constitutes direct empirical evidence of its technological adaptability.

The framework's ability to normalize results across these systems regardless of the technology used provides empirical evidence of its technological adaptability. The "DZ-Score" successfully captured the distinction between enterprise-grade performance and publicly available model capabilities, providing a quantitative basis for comparing model performance across different evaluation scenarios and implementation contexts.

The HELM Holistic Evaluation of Language Models framework [27] is one of the most comprehensive benchmarking suites, adopting a multi-metric approach to measure accuracy, calibration, robustness, fairness, and efficiency across multiple scenarios. While HELM established a clear standard for comparing foundation models, it often relies on fixed prompts that may fail to generalize across all LLMs. DynamicZ builds upon the principles of holistic evaluation but focuses specifically on the practical requirements of data interfaces within enterprise environments, utilizing normalized entropy to ensure more targeted domain coverage than the broad scenario taxonomy of HELM.

6. Limitations

The DynamicZ framework provides a robust conceptual foundation. It is important to acknowledge certain limitations related to the framework's current stage of development. The validation presented in this paper was conducted within a focused organizational scope, which served as a targeted environment for establishing the framework's underlying logic. The empirical datasets used in this phase provide a preliminary baseline for accuracy and confidence measures, rather than a comprehensive validation. Although DynamicZ is conceptualized as a holistic and generalizable framework, its empirical validation to date is constrained by a limited sample size of 84 queries within a single enterprise setting. Further evaluation across larger and more diverse datasets, spanning multiple organizational contexts, will be necessary to fully verify the framework's scoring dynamics and statistical stability. The initial formulation of categories involves a deliberate process to ensure context-specific relevance. This ensures a high level of adaptability. These factors represent expected stages in the deployment of a new evaluative tool. They do not detract from the core architectural strength of the logic. While the framework demonstrated promising results in a real-world enterprise setting, future research should validate its applicability across additional domains, industries, and natural-language-driven analytical architectures.

7. Conclusion

This paper detailed the development of DynamicZ. This finalized evaluation tool addresses the gaps between academic benchmarks and enterprise requirements. The framework shifts the focus from binary success metrics to a multi-dimensional DZ-Score. This incorporates statistical reliability and test set diversity. This provides the transparency necessary for the responsible adoption of AI-driven interfaces. The development process emphasized an inductive approach to category formation. It used mixed-methods triangulation to stay grounded in technical reality. DynamicZ is intended to be an industry and technology agnostic solution. It provides a standardized methodology for organizations to assess model performance. This assessment is conservative and aligned with business goals. Natural language processing is being integrated into essential business operations. Robust and adaptable evaluation frameworks are indispensable. They foster informed decision-making. They also ensure the secure democratization of data access. Future work will focus on testing the framework across a broader range of industries. This will continue to validate its universal applicability across the evolving landscape of artificial intelligence.

Declaration on Generative AI

The authors used Claude (Anthropic, claude.ai) as an AI-assisted writing tool to improve the manuscript's clarity, readability, and grammatical correctness. All scientific content, data interpretation, conclusions, and intellectual contributions are solely the work of the authors. The authors take full responsibility for the integrity and accuracy of the published work.

References

[1] D. Zobl, DynamicZ: A Robust Evaluation Framework for Text-to-SQL Systems Balancing Technical Metrics and Stakeholder Perspectives, Master's thesis, Management Center Innsbruck, 2025.

[2] A.-M. Popescu, O. Etzioni, H. Kautz, Towards a theory of natural language interfaces to databases, in: Proceedings of the 8th international conference on Intelligent user interfaces, ACM, Miami Florida USA, 2003, pp. 149–157. URL: https://dl.acm.org/doi/10.1145/604045.604070. doi:10.1145/604045.604070.

[3] A. Shankar, Architectural Patterns for Text-to-SQL: Leveraging LLMs for Enhanced BigQuery Interactions, 2023. URL: https://medium.com/google-cloud/architectural-patterns-for-text-to-sql-leveraging-llms-for-enhanced-bigquery-interactions-59756a749e15.

[4] Microsoft, LinkedIn, AI at Work Is Here. Now Comes the Hard Part, 2024. URL: https://www.microsoft.com/en-us/worklab/work-trend-index/ai-at-work-is-here-now-comes-the-hard-part#:~:text=The%20data%20is%20in%3A%202024,the%20face%20of%20AI%20inevitability.

[5] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, 2020. URL: https://arxiv.org/abs/2005.11401. doi:10.48550/ARXIV.2005.11401, version Number: 4.

[6] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, A. Mian, A Comprehensive Overview of Large Language Models, 2023. URL: https://arxiv.org/abs/2307.06435. doi:10.48550/ARXIV.2307.06435, version Number: 10.

[7] B. Zhang, Y. Ye, G. Du, X. Hu, Z. Li, S. Yang, C. H. Liu, R. Zhao, Z. Li, H. Mao, Benchmarking the Text-to-SQL Capability of Large Language Models: A Comprehensive Evaluation, 2024. URL: http://arxiv.org/abs/2403.02951. doi:10.48550/arXiv.2403.02951, arXiv:2403.02951 [cs].

[8] J. Choi, Fact-Consistency Evaluation of Text-to-SQL Generation for Business Intelligence Using Exaone 3.5, 2025. URL: http://arxiv.org/abs/2505.00060. doi:10.48550/arXiv.2505.00060, arXiv:2505.00060 [cs].

[9] B. G. Ascoli, Y. S. R. Kandikonda, J. D. Choi, ETM: Modern Insights into Perspective on Text-to-SQL Evaluation in the Age of Large Language Models, 2024. URL: https://arxiv.org/abs/2407.07313. doi:10.48550/ARXIV.2407.07313, version Number: 3.

[10] G. Katsogiannis-Meimarakis, G. Koutrika, A survey on deep learning approaches for text-to-SQL, The VLDB Journal 32 (2023) 905–936. URL: https://link.springer.com/10.1007/s00778-022-00776-8. doi:10.1007/s00778-022-00776-8.

[11] A. Mitsopoulou, G. Koutrika, Analysis of Text-to-SQL Benchmarks: Limitations, Challenges and Opportunities (2025).

[12] T. Weber, Explainability for Embedding AI: Aspirations and Actuality, 2025. URL: https://arxiv.org/abs/2504.14631. doi:10.48550/ARXIV.2504.14631, version Number: 1.

[13] R. K. Yin, Case study research and applications: design and methods, sixth edition ed., SAGE, Los Angeles London New Delhi Singapore Washington DC Melbourne, 2018.

[14] B. Wang, Y. Gao, Z. Li, J.-G. Lou, Know What I don't Know: Handling Ambiguous and Unanswerable Questions for Text-to-SQL, 2022. URL: https://arxiv.org/abs/2212.08902. doi:10.48550/ARXIV.2212.08902, version Number: 2.

[15] I. Saparina, M. Lapata, Disambiguate First Parse Later: Generating Interpretations for Ambiguity Resolution in Semantic Parsing, 2025. URL: https://arxiv.org/abs/2502.18448. doi:10.48550/ARXIV.2502.18448, version Number: 1.

[16] J. Shen, C. Wan, R. Qiao, J. Zou, H. Xu, Y. Shao, Y. Zhang, W. Miao, G. Pu, A Study of In-Context-Learning-Based Text-to-SQL Errors, 2025. URL: https://arxiv.org/abs/2501.09310. doi:10.48550/ARXIV.2501.09310, version Number: 2.

[17] C. E. Shannon, A Mathematical Theory of Communication, Bell System Technical Journal 27 (1948) 379–423. URL: https://ieeexplore.ieee.org/document/6773024. doi:10.1002/j.1538-7305.1948.tb01338.x.

[18] O. Cherednichenko, D. Chernyshov, D. Sytnikov, P. Sytnikova, Generalizing Machine Learning Evaluation through the Integration of Shannon Entropy and Rough Set Theory (2024). URL: https://arxiv.org/abs/2404.12511. doi:10.48550/ARXIV.2404.12511, publisher: arXiv Version Number: 1.

[19] M. Sung, C. Park, S. Ha, M. Ha, H. Lee, J. Kim, J. Kang, Entropy-based sampling for efficient training of deep learning on CNC machining dataset, Electronics Letters 60 (2024) e13308. URL: https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/ell2.13308. doi:10.1049/ell2.13308.

[20] Q. Lyu, K. Shridhar, C. Malaviya, L. Zhang, Y. Elazar, N. Tandon, M. Apidianaki, M. Sachan, C. Callison-Burch, Calibrating Large Language Models with Sample Consistency, 2024. URL: https://arxiv.org/abs/2402.13904. doi:10.48550/ARXIV.2402.13904, version Number: 1.

[21] Y. Yang, S. Kim, S. Kim, G. Lee, S.-Y. Yun, E. Choi, Towards Unbiased Evaluation of Detecting Unanswerable Questions in EHRSQL, 2024. URL: https://arxiv.org/abs/2405.01588. doi:10.48550/ARXIV.2405.01588, version Number: 1.

[22] G. Lee, H. Hwang, S. Bae, Y. Kwon, W. Shin, S. Yang, M. Seo, J.-Y. Kim, E. Choi, EHRSQL: A Practical Text-to-SQL Benchmark for Electronic Health Records, 2023. URL: https://arxiv.org/abs/2301.07695. doi:10.48550/ARXIV.2301.07695, version Number: 5.

[23] H. Schulz-Kümpel, S. Fischer, R. Hornung, A.-L. Boulesteix, T. Nagler, B. Bischl, Constructing Confidence Intervals for 'the' Generalization Error – a Comprehensive Benchmark Study, 2024. URL: https://arxiv.org/abs/2409.18836. doi:10.48550/ARXIV.2409.18836, version Number: 2.

[24] E. B. Wilson, Probable Inference, the Law of Succession, and Statistical Inference, Journal of the American Statistical Association 22 (1927) 209–212. URL: http://www.tandfonline.com/doi/abs/10.1080/01621459.1927.10502953. doi:10.1080/01621459.1927.10502953.

[25] Y. Zhao, C. Wu, Q. Jia, H. Zhu, J. Yan, L. Zong, L. Zhang, Z. Dong, M. Zhang, Confidence-Aware Multi-Field Model Calibration, 2024. URL: https://arxiv.org/abs/2402.17655. doi:10.48550/ARXIV.2402.17655, version Number: 2.

[26] G.-S. Ying, M. G. Maguire, R. J. Glynn, B. Rosner, Calculating Sensitivity, Specificity, and Predictive Values for Correlated Eye Data, Investigative Opthalmology & Visual Science 61 (2020) 29. URL: https://iovs.arvojournals.org/article.aspx?articleid=2770806. doi:10.1167/iovs.61.11.29.

[27] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, B. Newman, B. Yuan, B. Yan, C. Zhang, C. Cosgrove, C. D. Manning, C. Ré, D. Acosta-Navas, D. A. Hudson, E. Zelikman, E. Durmus, F. Ladhak, F. Rong, H. Ren, H. Yao, J. Wang, K. Santhanam, L. Orr, L. Zheng, M. Yuksekgonul, M. Suzgun, N. Kim, N. Guha, N. Chatterji, O. Khattab, P. Henderson, Q. Huang, R. Chi, S. M. Xie, S. Santurkar, S. Ganguli, T. Hashimoto, T. Icard, T. Zhang, V. Chaudhary, W. Wang, X. Li, Y. Mai, Y. Zhang, Y. Koreeda, Holistic Evaluation of Language Models (2022). URL: https://arxiv.org/abs/2211.09110. doi:10.48550/ARXIV.2211.09110, publisher: arXiv Version Number: 2.

[28] M. Brocidiacono, K. I. Popov, A. Tropsha, An Improved Metric and Benchmark for Assessing the Performance of Virtual Screening Models, 2024. URL: https://arxiv.org/abs/2403.10478. doi:10.48550/ARXIV.2403.10478, version Number: 1.

[29] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language Models are Few-Shot Learners, 2020. URL: https://arxiv.org/abs/2005.14165. doi:10.48550/ARXIV.2005.14165, version Number: 4.