Frontiers in Business and Finance Volume 2 Issue 1, 2025
ISSN: 3079-9325

AUBIQ: A Generative AI-Powered Framework for Automating Business Intelligence Requirements in Resource-Constrained Enterprises

Ruolin Qi[1, *]

[1] Johns Hopkins Carey Business School, Johns Hopkins University, Washington.DC 20001, United States

[*] Corresponding Author: rqi4@alumni.jh.edu

Abstract

In today's ever-changing data-driven environment, obtaining requirements for business intelligence (BI) systems remains a major challenge. This pain point is particularly prominent in small and medium-sized enterprises with limited resources: weak technical teams, high external communication costs and long response cycles make traditional manual specification processes more difficult to maintain. This paper proposes a new AI-driven codeless framework AUBIQ, which uses semantic search and large language models (LLM) to automate and accelerate the specification of BI system requirements. The system converts user natural language input into outputs such as prototype analysis code, descriptions, and data dependencies through a conversational interface, and can generate detailed test case reports with visual assistance, thereby streamlining the requirements acquisition process. In addition, AUBIQ introduces a user feedback mechanism to continuously improve BI reports and system design, showing practical application potential for accelerating data-driven decision-making. This paper also explores the broad prospects of generative artificial intelligence in changing the development of BI systems and explains its role in large-scale evolving data engineering practices.

Keywords

Business intelligence; Generative artificial intelligence; Large language models; AI-driven data engineering; Requirements automation; Semantic search; Ontology-based query generation; Text-to-SQL

1. Introduction

1.1. Research Background

In today's data-driven business environment, organizations rely on business intelligence (BI) systems to extract insights from decentralized data sources such as SAP, customer relationship management (CRM) systems, and internal databases. As the business environment changes rapidly, BI systems must continue to adapt to evolving requirements, requiring continuous adjustments to analytical models and data models to support decision-making processes. However, the process of refining and specifying BI system requirements is difficult. As BI systems scale and requirements change dynamically, manually managing evolving requirements is time-consuming and error-prone, requiring a lot of coordination between data analysts, domain experts, and business stakeholders. Traditional BI requirements elicitation methods often lead to a gap between business requirements and technical implementation. This inefficient process leads to repeated design iterations, mismatches between available data and desired analysis, and outages in production environments. This process becomes more complex as enterprises increasingly move toward cloud computing, data transformation, and technology migration. Therefore, to ensure business continuity during the evolution of BI systems, it has become increasingly necessary to introduce automated tools to simplify the requirements specification process, improve accuracy, and reduce repeated manual intervention [1].

Current approaches fall short in capturing the semantic nuances of data schemas and aligning business requirements with technical solutions, especially in the context of legacy systems. Some related work has explored the use of target models for BI system design, but has not involved automated [2].

1.2. Problem Statement

For small and medium-sized enterprises, the above challenges are particularly prominent: due to the lack of sufficient technical resources, small and medium-sized enterprises face greater difficulties in deploying and maintaining complex BI systems, and have to communicate frequently with external experts, resulting in high communication costs and lengthy response cycles. This significantly raises the threshold for the implementation of BI systems and slows down the speed of data-driven decision-making. To address these issues, this paper proposes a new BI requirements automation method based on generative artificial intelligence. We designed and implemented a code-free system called AUBIQ (AI-driven Unified BI Intelligence Qualifier) , which combines large language models (LLM) and semantic search technology to automate and accelerate the process of refining and standardizing BI requirements. Through this system, users can interact with data through a conversational interface, and natural language business requirements will be automatically converted into executable queries, analysis specifications, test reports, data dependencies, and prototype analysis code. This system reduces the time and effort required to develop BI systems while ensuring the accuracy of requirement specifications, marking an important AI-driven data engineering advancement in BI requirement acquisition automation. AUBIQ uses semantic technology to align user intent with the underlying data structure to ensure the accuracy and contextual relevance of generated queries; at the same time, by introducing user feedback, it continuously optimizes output and improves the quality of BI system design and reporting. With the application of LLM and semantic search, the system streamlines the data management process and opens up new prospects for the scalability and flexibility of BI system design [3].

This paper will explore the technical architecture of the system, the integration of artificial intelligence and data discovery technology, and its practical significance for improving the efficiency of BI analysis and development. At the same time, this paper discusses the potential of generative AI in transforming the field of data engineering and proposes future research directions. The paper is organized as follows: Section 2 provides an actual operation example of the AUBIQ system to demonstrate its core functions; Section 3 introduces the key technical components that support this system; Section 4 details the system architecture and main modules of AUBIQ; Section 5 evaluates the system and summarizes the experience and inspiration based on the application of four different industries, and discusses the impact of the system on the working methods of data professionals and the development of BI systems; Section 6 reviews the relevant research work in the field of query generation; Section 7 discusses the internal, external and construct validity threats of this study and their mitigation measures; Finally, Section 8 gives the research conclusions and looks forward to future work [4].

2. Run the example

The core goal of AUBIQ is to enable users (such as data analysts, domain experts, and business managers) to interact with data systems directly and intuitively in natural language, thereby

ISSN: 3079-9325

formalizing analytical requirements more accurately and efficiently than traditional methods. The following example shows several core functions of AUBIQ: (1) business problem formulation; (2) analytical queries automatically generated by the system; (3) natural language interpretation of queries; (4) relevant data models in ontology format; (5) mapping of physical data sources involved in the query; and (6) test cases generated during the requirements specification process [5].

To demonstrate these features, we used the open source sample database AdventureWorks2014 developed by Microsoft. This database simulates the complete information system of a bicycle shop, contains 71 tables and hundreds of fields, and provides rich data for various analytical queries. In this example, the question asked by the user is: "Please provide the total sales of each product in Euros." The system then automatically generates the corresponding SQL query and gives a natural language interpretation of the query. At the same time, the system provides a sub-ontology graph of related datasets based on semantic search to show the business concepts and datasets involved in the query. Subsequently, the user directly executed the SQL query on the AdventureWorks2014 database and reported the results. It is worth noting that the word "Earnings" is used in this business question, rather than technical terms such as "Sales" or the specific "Sales Amount", and there is no direct mention of fields such as CurrencyRate or CurrencyCode in the query . This means that the tool uses semantic search to interpret the user's intention based on the data model and correctly maps "Earnings" to sales-related data. The system generates requirements specifications including: initial business questions, SQL queries for analysis, query results, supported ontology data model fragments, and binding relationships with database tables. The data model presented to users abstracts the physical database structure into conceptual relationships. For example, foreign key relationships are displayed as object attributes rather than explicit table joins, which improves the understandability of the model to non-technical stakeholders [6].

The system also uses generative AI to interpret query logic and generate reports that go beyond traditional data tables, including graphical visualizations to deepen users' understanding of query results. These reports provide a solid foundation for verifying and refining BI requirements. This innovative practice of applying advanced generative AI to BI requirement specifications shows that AI technology can significantly improve the efficiency and scalability of user interactions with massive data systems, enabling users to navigate and refine their BI requirements more accurately and clearly [7, 8].

3. Key technologies

The AUBIQ framework uses a series of core technology components to realize the automated specification of BI requirements, including OWL ontology, data binding, semantic search, and text query generation. These key technologies are introduced below.

OWL (Web Ontology Language): OWL is a powerful and expressive knowledge representation language. It provides a formal structured vocabulary that can be used to represent concepts in various fields. OWL supports the formal description of classes, properties and their instances, so that concepts and their relationships can be fully characterized. In AUBIQ, we use OWL to represent the logical data model of heterogeneous data sources in an abstract way. With OWL, data sources in different forms such as relational database schemas and JSON schemas can be uniformly represented as ontologies. The main exchange format for OWL ontologies is RDF (Resource Description Framework). In the context of this article, "ontology" refers to the logical data model, while "physical model" refers to the actual data representation in the data source system [9].

Data binding: Data binding is a formal language used to define the mapping relationship between physical data models and RDF datasets (ontologies). In AUBIQ, the OWL ontology corresponding to each data source represents its data structure. The physical data model can be mapped to the ontology structure in combination with binding languages such as R2RML. By defining data binding, the system can seamlessly switch between the ontology and the physical model, supporting query generation and interaction between the system and the underlying information system. For example, we use R2RML to express the mapping between a relational database and its OWL ontology, so that AUBIQ can execute queries against data sources on different platforms such as Databricks , Microsoft SQL Server, and Stardog [10].

Semantic search: This study integrates semantic search capabilities into the system and uses vector databases such as Milvus and Pinecone to implement semantic indexing and retrieval. Vector databases can efficiently identify items that are most "similar" to queries by representing items as fixed-length numerical vectors in high-dimensional space and combining them with the k-nearest neighbor algorithm. Modern vector databases also combine approximate nearest neighbor algorithms and hybrid retrieval techniques to improve query efficiency while ensuring retrieval accuracy. In AUBIQ, semantic search is used to match user natural language queries with ontology concepts in the knowledge graph, thereby dynamically constructing a related sub-ontology; the system then uses LLM to generate SQL queries based on the sub-ontology. In this way, AUBIQ can understand the semantics in the business language, find the corresponding data model concepts, and eliminate the query barriers for users who do not understand the details of the underlying data structure [11].

Text query generation (Text2Query): Recent studies have begun to explore methods for automatically generating queries from unstructured text. The classic Text-to-SQL problem is viewed as a mapping process from natural language business question reasoning to SQL queries. Existing research has proposed generation methods for other query languages, such as Cypher queries for graph databases and SPARQL queries for RDF knowledge bases. In AUBIQ, we draw on these recent advances and apply LLM to the task of "text to query": the model starts from the user's natural language question and automatically generates the corresponding query statement that can answer the question. This capability enables the system to handle more flexible and diverse questions and generate BI queries that meet user needs [12, 13].

4. AUBIQ system architecture

Although AUBIQ is currently used as an internal service tool of Accenture and has not been publicly released, its core components and workflow can be summarized as follows. The system architecture includes tools in two main phases: tools in the Setup phase are used to connect new data sources, and tools in the Run-Time phase are used to handle user business problems. In the Setup phase, the Ontology Discovery (OntoDis) tool first builds the ontology data model of the data source based on the data source metadata and generates mapping bindings between the ontology and the physical model. Next, the Ontology Management (OntoManager) tool is responsible for recording information such as connection parameters, derived data models and their bindings to establish a data source catalog. Finally, the Ontology Search (OntoSearch) tool creates a semantic search index for the new ontology and embeds it into a vector database to support dynamic semantic parsing and entity matching in the Run-Time phase [14].

During the runtime, when a user asks a business question, AUBIQ first generates a corresponding executable query. The query executor then executes the query and retrieves the results, while the data visualizer generates a graphical representation of data insights in parallel. The test report automation tool archives these results for subsequent testing and verification. In addition, to improve interpretability and fit user intent, the system provides an optional feature: automatically generate a natural language interpretation of the query to provide users with feedback on how the system understands their request. The following introduces the main modules in the AUBIQ architecture and their functions.

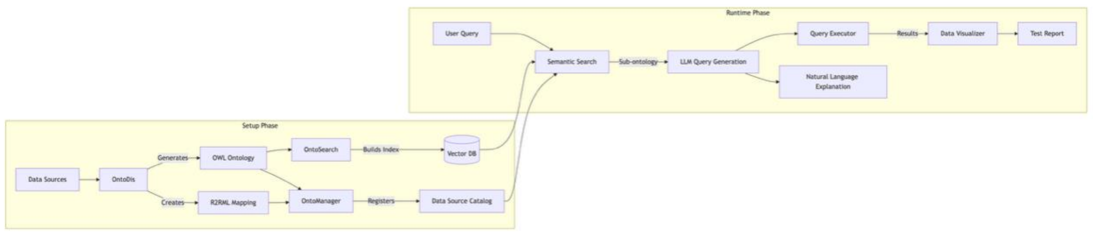
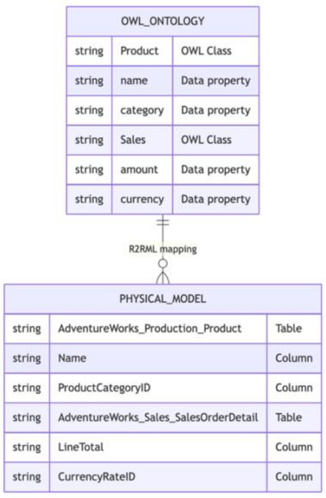


Figure 1: AUBIQ System Architecture

4.1. OntoDis : Generate Ontology and Data Binding

OntoDis tool enables business users to define analytical requirements for data sources from different sources and distributed sources. It supports on-demand data integration by abstracting the physical data model into an ontology model (OWL) to hide the underlying complex implementation details and shift the focus from the physical structure to the conceptual level. This abstraction method avoids the efficiency constraints of physical data models such as the third normal form (3NF) and allows the fusion of similar concepts from different sources at the ontology level (for example, aligning the "customer" and "client" entities in different data sources), achieving a unified representation of cross-source data.

OntoDis uses the metadata of the data source to automatically generate an OWL ontology corresponding to the data source, and establishes a mapping binding between the ontology and the physical data model. Specifically, OntoDis reads the relational database schema or other schema information, abstracts and conceptualizes the physical data schema into a comprehensive OWL ontology through iterative refinement, and then applies specifications such as R2RML to bind the ontology to the original physical model. In this process, OntoDis provides model refinement capabilities to improve the quality of the ontology by editing the model in a targeted manner. For example, users can delete or rename unnecessary classes, split or merge redundant classes, and remove irrelevant attributes based on preset strategies. These strategies consist of specific conditions and modification actions, and can iteratively adjust the model based on the name tags and relationships of the entities [15].



OWL_ONTOLOGY		
string	Product	OWL Class
string	name	Data property
string	category	Data property
string	Sales	OWL Class
string	amount	Data property
string	currency	Data property

PHYSICAL_MODEL		
string	AdventureWorks_Production_Product	Table
string	Name	Column
string	ProductCategoryID	Column
string	AdventureWorks_Sales_SalesOrderDetail	Table
string	LineTotal	Column
string	CurrencyRateID	Column

Figure 2: Ontology-Physical Model Mapping

Table 1: OntoDis schema-lifting pipeline

```
%-------------------------------------------------------------
% Listing — OntoDis Schema Lifting & Fixed-Point Refinement
\begin{table}[H]
  \caption{OntoDis Schema Lifting and Rule-Driven Ontology Refinement Pipeline Achieving Fixed-Point Termination}
  \label{table:01}
  \begin{algorithm}[H]
    \begin{algorithmic}[1]
      \REQUIRE Physical schema $P$ (e.g.\ SQL DDL, JSON Schema), refinement rule set $\mathcal{R}$
      \STATE $O \gets \textsc{LiftToOwlDraft}(P)$ \COMMENT{Initial conceptual model}
      \REPEAT
          \STATE $changed \gets \textbf{false}$
          \FORALL{$r \in \mathcal{R}$}
              \IF{$r.\textsc{Matches}(O)$}
                  \STATE $O \gets r.\textsc{Apply}(O)$
                  \STATE $changed \gets \textbf{true}$
              \ENDIF
          \ENDFOR
      \UNTIL{$\lnot changed$} \COMMENT{Fixed-point reached}
      \STATE $M \gets \textsc{GenerateR2RML}(O, P)$ \COMMENT{Create bindings}
      \RETURN $(O, M)$
    \end{algorithmic}
  \end{algorithm}
\end{table}
%-------------------------------------------------------------
```

4.2. OntoManager : Data Source and Ontology Management

OntoManager is responsible for managing and maintaining the models and ontology information related to the data source. This module catalogs the data models, binding relationships, and access rights according to the data governance strategy, and adopts a multi-tenant strategy to divide the management rights of different users or teams. OntoManager supports the management and group management of data source versions, ensuring that diverse data sources are effectively governed within the framework, while meeting different user needs and operational requirements through strict access control. With OntoManager , AUBIQ can maintain an ontology and data source library that evolves over time, providing orderly and traceable knowledge resource management for different data analysis scenarios.

4.3. OntoSearch : Ontology Catalog Search

As the number of connected data sources and ontologies continues to increase, the OntoSearch module is used to efficiently search and match in the huge ontology catalog. OntoSearch creates a semantic search index for each newly added ontology to support fast search for related concepts. Specifically, OntoSearch builds a string representation of each entity by traversing the label and annotation information of each entity in the ontology; if necessary, it can also take into account the names of related attributes of the entity. Then, OntoSearch uses a pre-trained or fine-tuned language model to convert these string representations into vector embeddings and store them in the vector database. To improve the search effect, OntoSearch adds metadata

ISSN: 3079-9325

to the entity index, such as the data source identifier of each entity source, the entity type, etc., to support hybrid search combining vectors and attributes. For example, searching for classes related to "Helmet" in the AdventureWorks2014 database, OntoSearch will return related tables such as "Product" based on semantic similarity, because the system can interpret "Helmet" as a product category.

Retrieval accuracy depends largely on the quality of entity string representation. To this end, OntoSearch provides customizable configurations that allow users to choose which annotation information to include when generating embeddings. For example, users can specify that class definition annotations are used first, and irrelevant information such as version numbers are ignored to optimize the embedding's capture of semantics. This can minimize noise while improving recall, ensuring that retrieval results are both comprehensive and accurate.

Sub-ontology retrieval and construction: When dealing with user business problems, OntoSearch is responsible for identifying sub-ontologies related to the problem from the global ontology catalog. The specific process is: First, extract keywords such as nouns from user queries, and locate the corresponding classes and related attributes in the ontology library based on them. Then, starting from these preliminary matched entities, expand the search scope along the association relationship between entities in the ontology graph until the predetermined boundary conditions (such as limiting the number of hops or relevance thresholds) are reached. During the expansion process, the algorithm determines the concept pairs that are preferentially expanded according to the descending order of the cosine similarity of the entity embedding vectors to ensure that the extracted sub-ontology covers all entities required to answer the query (to ensure recall rate) and excludes irrelevant content (to improve precision).

Note: The algorithm is designed taking into account that user queries often involve multiple entities in the data model, similar to joins in SQL. Since users may not fully understand these entities and their relationships, OntoSearch uses semantic search to automatically identify the relevant entities involved in the user's question and expand them to form a sub-ontology that contains all necessary entities. Its goal is to reduce redundancy (exclude irrelevant entities) while ensuring the correctness of the query (including the required entities) and extract sub-ontologies that are highly consistent with the user's intentions.

Table 2: OntoSearch sub-ontology extraction

```
%-----------------------------------------------------------
% Listing — OntoSearch Relevance-Guided Expansion
\begin{table}[H]
  \caption{OntoSearch Sub-Ontology Extraction Algorithm Leveraging Relevance-Guided Breadth-First Expansion with Depth and Similarity Constraints}
  \label{table:02}
  \begin{algorithm}[H]
    \begin{algorithmic}[1]
      \REQUIRE Natural-language query $q$, global ontology graph $G=(V,E)$, similarity threshold $\tau$, max depth $h$
      \STATE $Seeds \gets \textsc{KeywordMatch}(q)$
      \STATE $G_s \gets (Seeds,\varnothing)$
      \FOR{$d \gets 1$ \TO $h$}
        \STATE $Frontier \gets \textsc{Boundary}(G_s)$
        \FORALL{$v \in Frontier$}
          \STATE $sim \gets \cos\!\bigl(\textsc{Embed}(v),\textsc{Embed}(q)\bigr)$
          \IF{$sim \ge \tau$}
```

```
                \STATE $\textsc{Attach}(G_s, v)$
            \ENDIF
        \ENDFOR
    \ENDFOR
    \RETURN $G_s$
  \end{algorithmic}
 \end{algorithm}
\end{table}
%-------------------------------------------------------------
```

4.4. AUBIQ: Query Generation and Self-Tuning

The core function of the AUBIQ system is to convert business queries into executable technical queries at the semantic level. This "text-to-query" conversion process is a hot research area, especially the text-to-SQL direction, which has gained great development and widespread attention in recent years. This field is committed to accurately converting business requirements into correct query statements. To achieve this goal, AUBIQ has made many enhancements to the standard text-to-query framework, including: (1) query generation and customization for large-scale systems; (2) user feedback integration and session continuity; (3) self-debugging and error correction; (4) query logic interpretation (query to text); and (5) multi-platform query across distributed systems. The following describes these enhancements and system implementation details.

Query generation and customization: When a user asks a business question, the system first uses OntoSearch to locate the relevant sub-ontology as a semantic context. LLM then generates query statements in the target query language (such as SQL or SPARQL) based on the sub-ontology and predefined query generation rules. Since the identified sub-ontology is independent of the underlying physical data model, we map the generated queries back to the actual data source structure through pre-established bindings. The system adopts different query execution strategies for different data sources : for example, for SQL Server relational databases, LLM generates and executes SQL queries with references to physical tables; while for the Stardog knowledge graph platform that supports OWL ontologies and SPARQL queries, LLM generates SPARQL queries based on ontology subsets. At system initialization, we pre-provide corresponding guidelines, exemplary question-query pairs, and conversion methods for translating OWL ontology components and bindings into the target query language for each query language. This ensures that LLM has clear specifications to follow when generating queries in a specific language. We assume that the sub-ontologies and their bindings obtained through OntoSearch already contain all the information required for the query; this assumption has been verified and supported in our preliminary experiments using platforms such as Stardog and Databricks [5].

Conversation history and user feedback: AUBIQ extends the traditional Text-to-Query framework and introduces a dynamic conversation mechanism to continuously obtain user feedback and improve query results. The system retains the conversation history of the most recent several rounds of interactions to maintain contextual coherence, allowing LLM to refer to previous questions and answers when generating subsequent responses to understand subtle changes in user intent. This contextual information is used for subsequent questions or clarifications, helping LLM provide more relevant and accurate responses, improving the quality of conversational interactions and user experience. By retaining the conversation context, users can iteratively refine queries in a series of conversations without repeating existing information. For example, after getting a preliminary report, users can further ask for specific data details or request different visualizations, and the system will respond based on

ISSN: 3079-9325

the context. This interactive feedback mechanism enables users to understand existing data capabilities in real time, adjust queries in a timely manner to improve accuracy, and provide opportunities to improve the ontology, thereby enhancing the understanding of the data by both users and LLM.

Self-debugging and error correction: AUBIQ has the ability to self-debugging, that is, it automatically detects and fixes potential errors after query generation. The system has three types of built-in checkers: (1) Syntax checker, which ensures that the generated query complies with grammatical specifications (for example, the SQL syntax is correct); (2) Semantic checker, which verifies whether the query is consistent with the entities, types, and levels in the ontology, and checks whether the generated query meets the semantic intent of the user's business question; (3) Execution checker, which submits the query to the actual data engine for execution to detect any runtime errors or exceptions. When any checker finds a problem, the system will feed this information back to the next round of LLM prompts, guiding the LLM to correct the previous problem and generate more accurate queries. This iterative generate-verify-correct cycle improves the accuracy of query generation and simplifies the process of converting user questions into executable queries. In our preliminary experiments, these checkers effectively ensured the correctness of the query conditions, but currently they only verify the correctness of the query itself, but not the business rationality of the final query results. It should be noted that deterministic checkers (such as syntax checking) ensure security by rejecting invalid queries; while non-deterministic checkers (such as semantic checking) may occasionally mistakenly reject legal queries. Therefore, caution should be exercised during design and evaluation to minimize the false positive rate so as not to affect the system's acceptance of valid queries.

Explainability: For generated formal queries (such as SQL statements), AUBIQ provides an explanation function for non-professional users to improve the comprehensibility of the results. Specifically, after the query is generated, the system will construct a prompt for LLM, asking it to explain the query intent to non-technical users in a specified style. The explanation style can be selected by the user, including different tones such as Compact, Verbose, Formal, Simple or Precise. The prompt contains the original business question and the sub-ontology information used to generate the query, and LLM outputs a natural language explanation of the query based on this. Through this module, technical users can view technical details such as the generated SQL, while business users can read the explanatory statements given by LLM to understand the intent and logic of the query. This explanation function helps bridge the business and technology gap and ensures that users from different backgrounds can understand how BI needs are implemented.

Distributed Data Source Query: Traditional query generation is usually oriented towards a single data source. AUBIQ extends this capability with the help of knowledge graph technology, enabling it to support queries across multiple data sources. Specifically, we leverage enterprise knowledge graph platforms such as Stardog to virtually fuse multiple heterogeneous data sources into a knowledge graph. OntoSearch can not only retrieve sub-ontologies within a single ontology, but also search for related entities across multiple ontologies to build a comprehensive sub-ontology across sources. The system then generates a SPARQL query to be executed on the knowledge graph to retrieve data distributed across different sources. This approach enables AUBIQ to conduct federated queries and achieve unified access to distributed data warehouses. When integrating multiple data sources, the system needs to solve the problem of aligning similar concepts between different sources. To this end, we introduce an alignment step after OntoDis generates the initial ontology to map synonymous concepts across sources to a common ontology to ensure semantic consistency during query. Future research will explore more advanced automatic ontology alignment techniques to further improve the effectiveness and accuracy of cross-source queries.

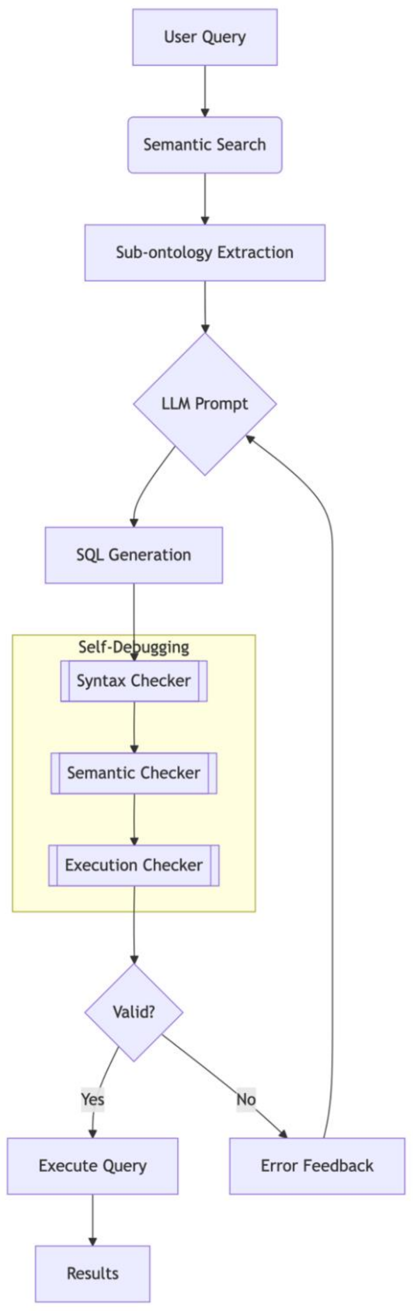


Figure 3: Query Generation Workflow

Table 3: AUBIQ self-debugging loop

```
%-------------------------------------------------------------
% Listing — Generate–Verify–Correct Self-Debugging Loop
\begin{table}[H]
  \caption{AUBIQ Generate–Verify–Correct Loop for Producing Executable, Semantically Valid SQL under Iterative Error Feedback}
  \label{table:03}
  \begin{algorithm}[H]
    \begin{algorithmic}[1]
      \REQUIRE Question $q$, sub-ontology $G_s$, bindings $M$, max retries $k$
      \STATE $prompt \gets \textsc{ComposePrompt}(q, G_s, M)$
      \FOR{$i \gets 1$ \TO $k$}
        \STATE $sql \gets \textsc{LLMGenerate}(prompt)$
        \STATE                           $pass                              \gets
\textsc{SyntaxCheck}(sql)\,\wedge\,\textsc{SemanticCheck}(sql,G_s)\,\wedge\,\textsc{DryRun}(sql)$
        \IF{$pass$}
          \RETURN $sql$
        \ELSE
          \STATE $err \gets \textsc{CollectDiagnostics}()$
          \STATE $prompt \gets prompt \cup \{err\}$ \COMMENT{Feedback for next round}
        \ENDIF
      \ENDFOR
      \RETURN \textsf{FAIL} \COMMENT{Retry budget exhausted}
    \end{algorithmic}
  \end{algorithm}
\end{table}
%-------------------------------------------------------------
```

4.5. Report Generation and Data Visualization

Once AUBIQ generates an executable query, the user can run the query on the corresponding data source. The system executes the query on the analysis platform through the query executor, which is either directly connected to the data source itself or acts as a unified platform for data access. After the query execution is completed, AUBIQ obtains the result set (supports paging by page) and displays the result data on the system dashboard.

If the user chooses to visualize the data, the system will call LLM to automatically generate the corresponding graphical representation based on the query results. Specifically, AUBIQ integrates drawing libraries such as the Plotly Express library in the data visualizer. The system provides LLM with guidance on supported visualization types, snapshots of result data subsets (such as the first 10 rows and 10 columns), expected output formats, and sample code snippets. Based on this, LLM generates the code required to draw the specified visualization (such as Plotly Express code), which is executed by the system to produce a chart. If an error occurs in this process, the system will repeatedly call LLM for adjustments, up to a predetermined number of iterations, until the chart is successfully generated and presented to the user. At present, the system uses Python's Plotly Express library for visualization by default. In the future, different visualization platforms can be supported by replacing graphics tools or providing more examples.

The generated analysis reports and visualization charts are not only used for immediate display, but also for future reference and verification. AUBIQ archives each generated report and chart together with the data snapshot on which it is based, and can also upload it to the test report automation platform as a test case when necessary. Through such a complete report retention mechanism, the system creates a traceable record of results for each analysis requirement, which is convenient for future review of analysis results or rapid adjustment when requirements change. This ensures that each visualization meets current analysis needs, and its accumulation also constitutes a knowledge base to support future decision verification and review [16].

ISSN: 3079-9325

Table 4: TableLLM-orchestrated visualization pipeline

```
%-------------------------------------------------------------
% Listing — OntoDis Schema Lifting & Fixed-Point Refinement
\begin{table}[H]
  \caption{OntoDis Schema Lifting and Rule-Driven Ontology Refinement Pipeline Achieving Fixed-Point
Termination}
  \label{table:01}
  \begin{algorithm}[H]
    \begin{algorithmic}[1]
      \REQUIRE Physical schema $P$ (e.g.\ SQL DDL, JSON Schema), refinement rule set $\mathcal{R}$
      \STATE $O \gets \textsc{LiftToOwlDraft}(P)$ \COMMENT{Initial conceptual model}
      \REPEAT
          \STATE $changed \gets \textbf{false}$
          \FORALL{$r \in \mathcal{R}$}
              \IF{$r.\textsc{Matches}(O)$}
                  \STATE $O \gets r.\textsc{Apply}(O)$
                  \STATE $changed \gets \textbf{true}$
              \ENDIF
          \ENDFOR
      \UNTIL{$\lnot changed$} \COMMENT{Fixed-point reached}
      \STATE $M \gets \textsc{GenerateR2RML}(O, P)$ \COMMENT{Create bindings}
      \RETURN $(O, M)$
    \end{algorithmic}
  \end{algorithm}
\end{table}
%-------------------------------------------------------------
```

5. Evaluation and Lessons Learned

To validate the effectiveness of AUBIQ, we implemented and iterated on the system in four different domains: Security, Air Defense, Retail, and Banking. Each domain involves its own unique database and business objectives. We invited 23 subject matter experts (SMEs) from five countries and four organizations to participate in the evaluation. These experts included data analysts, data engineers, and data scientists. They provided feedback for our qualitative evaluation as important informants. We tested AUBIQ's performance in real-world tasks by letting the experts interact with the system through structured usage sessions. Following the Design Science Research methodology, we incorporated expert feedback (including suggestions for improvements to the framework and domain-specific use cases) into the iterative development of the system and continuously adjusted AUBIQ. This evaluation method provides practical insights into the adaptability and effectiveness of the tool in different business scenarios. The following are the main lessons and inspirations we summarized during the evaluation process [17].

5.1. Improving Data Analysis: Interactive Feedback and User-centered Design

The interactive human-computer feedback mechanism implemented by AUBIQ plays a key role in connecting user questions with system-generated queries and related data models. On the one hand, it allows users to understand the supporting information that existing data can provide in real time during the query construction process, so as to calibrate and refine queries in real time to ensure accuracy; on the other hand, it provides an opportunity to improve the ontology - when LLM cannot correctly interpret user questions, user questions and corrections help us discover the deficiencies of the ontology model and improve it, thereby enhancing the understanding of the data by both the system and the user.

For a no-code BI platform, the accuracy of automatically generated queries and corresponding data models is crucial, especially when the target users include business personnel and data professionals, and the system must be easy to use and the generated results must be trustworthy. AUBIQ improves user-friendliness and efficiency in the process of building analytical specifications by combining LLM with an interactive feedback loop. For example, the system provides many practical functions: (1) Regenerate queries: When users are not satisfied with the results, they can request LLM to regenerate different versions of queries with one click; (2) Manually adjust queries: Advanced users can directly edit the SQL queries generated by the system to make minor modifications; (3) Query data models and patterns: Users can ask the system for details about the data model structure to understand the entities and relationships involved in the query; (4) Edit data visualization: Users can adjust the visualization reports generated by the system, such as changing the chart type or layout. These functions are designed to best meet user intentions, reduce the number of rounds of repeated modifications, and allow users to obtain the required analytical results in fewer steps.

During the evaluation process, a prominent use case was the scenario where data professionals used AUBIQ for data semantic interpretation and agile analysis prototypes. These users hope to use AUBIQ to accelerate data exploration and prototyping of analytical dashboards. Through interactive semantic queries and fast visualization, they can understand the key characteristics of complex data sets and verify analytical ideas in a short time. This application case demonstrates the value of this toolset in simplifying complex data analysis, improving data accessibility, and quickly generating insights. Overall, AUBIQ's human-computer interaction design fully embodies the "user-centric" concept: through real-time feedback and easy-to-use adjustment options, it allows users from different backgrounds to efficiently participate in the BI requirements specification and data analysis process.

5.2. System Technical Requirements

During our exploration, we further clarified several technical requirements for AUBIQ. In particular, we focused on the multi-purpose capabilities of LLM, hoping that it can not only resolve general user questions, but also guide complex queries raised by domain experts to be correctly mapped to the data model. Discussions with multiple domain experts led to several system features introduced in Section 4, such as adaptive query generation, session memory, and self-debugging. In addition, experts generally hope that the system can support a variety of database types and storage formats, and not be limited to SQL - that is, queries can be generated for different data query languages (such as NoSQL, GraphQL , etc.) for flexible use in distributed data source environments.

To meet the scalability requirements, we introduced an ontology semantic layer into the system to abstract the underlying heterogeneous databases. With this ontology abstraction, AUBIQ can achieve model combination and fusion by generating multiple subqueries. Specifically, the system can generate queries for different data sources separately, and then merge these query results using ontology bindings to support cross-platform and cross-language queries. The architecture is modular and flexible, enabling AUBIQ to switch smoothly between different query languages and BI engines. For example, for the same business problem, the system can generate SQL queries to execute in a relational database, or generate SPARQL queries to run on an ontology graph platform. This flexibility shows that AUBIQ's architectural design meets the technical requirements required for widespread application in an enterprise environment: compatibility with multiple data sources, multiple query languages, and multiple BI tools. In the future, we plan to further expand the technical boundaries of the system, including supporting larger data sets and more complex query scenarios [18].

5.3. The Importance of a Good Data Model

Practice has proven that high-quality logical data models are critical to the success of systems such as AUBIQ. Data engineers have long recognized the important role of logical data models in complementing physical data models, but in reality many companies lack formal logical models and rely only on physical models that are full of abbreviations, ambiguous names, and even transliterations from foreign languages. This not only makes understanding difficult, but also limits the effectiveness of automatic query parsing. LLM plays a key role in systems such as AUBIQ, and it requires a semantically clear middle layer (i.e., ontology) to fully realize its capabilities. This has led to a new research topic: how to automatically derive, evaluate, and improve formal ontology representations, and develop corresponding design and maintenance best practices. Our experience is that only by developing a logically complete and semantically rich model can we improve the human-computer interaction interface, achieve efficient connection between human cognition and machine understanding, and truly improve the effect of data interaction.

For companies moving towards the era of generative AI applications, neglecting to optimize physical data models or not building corresponding logical models may hinder their ability to adopt tools such as AUBIQ. In our project, an important advantage demonstrated by introducing the AUBIQ toolset was the ability to create a catalog of semantic data representations that can be queried by a natural language interface. Users can get a friendly overview of data integration possibilities by simply asking questions, which greatly improves the efficiency of the decision-making process. These advances enable companies to better leverage generative AI technologies and ensure that their data infrastructure is compatible and optimized with these innovations. Therefore, we recommend that companies invest in improving the semantic layer of their data models before deploying systems like AUBIQ, such as unifying naming conventions and establishing business concept ontologies. This will largely determine whether such AI-driven tools can fully play their role and create real value for the company.

5.4. Security Considerations in LLM-Driven Query Generation

While using LLM to automatically generate queries brings convenience, it also introduces several security risks. First, there is a risk of generating malicious or inappropriate queries. Without proper restrictions, LLM may generate queries that accidentally modify database contents, leak unauthorized data, or even cause security vulnerabilities such as SQL injection due to the lack of strict validation of input. Second, when fine-tuning LLM to incorporate enterprise database knowledge, the data values stored in the underlying storage may be inadvertently leaked. If the LLM service is hacked by a malicious attacker, it may output harmful results, including useless queries that consume a lot of computing resources, misleading queries with subtle errors, or directly expose sensitive data. In addition, letting LLM learn internal enterprise data itself poses the risk of information leakage - if data containing sensitive information is used during the fine-tuning process, this information is likely to appear in the model response, thereby bypassing the original data access control measures.

In response to the above issues, we have taken multiple measures in the design and use of the system. First, all data entering LLM is strictly managed to ensure that no personal sensitive information (PII) or confidential data is exposed to the model during the prompting and fine-tuning stages. Second, a protection strategy is developed for the output of LLM, such as configuring read-only database permissions, preventing the model from generating harmful queries such as update, and performing security reviews and tests on the generated queries. Third, we have established sound security best practices, including access control to LLM services, output monitoring, and early warning mechanisms for abnormal behavior. Finally, we realize that security issues not only exist at the technical level, but also depend on continuous monitoring and improvement. In the future, we will dynamically update AUBIQ's protection strategy based on the latest security research to ensure that security risks are minimized while enjoying the convenience brought by LLM.

5.5. Implications for the Design and Use of Future BI Systems

Our recent explorations have shown that AUBIQ has the potential to have a transformative impact on enterprises in terms of democratizing BI capabilities. Through discussions with several senior executives in enterprise data departments, we found that this modular architecture makes it easier for business users to trigger BI analysis processes. AUBIQ's flexible architectural design supports rapid integration with various data sources, enabling remote queries of heterogeneous systems distributed everywhere, while reducing dependence on expertise. This means that even business personnel who lack a deep technical background can build personalized BI dashboard prototypes without having to rely entirely on the data engineering team, thereby reducing the burden on the latter and improving the productivity of dashboard design. At the same time, the AUBIQ toolset catalogs analysis through natural language descriptions or shared data, which improves the discoverability of analytical assets and optimizes data structures for common analytical needs. These capabilities are crucial for developing high-value data products and building an enterprise's data mesh, bringing new ideas to the development and organization of data products.

Experts in various fields have high expectations for AUBIQ. They believe that the system has the potential to revolutionize the report and analysis design process, significantly reduce labor costs and shorten delivery cycles. For example, in terms of report generation, AUBIQ automates many tasks that require repeated manual communication and adjustment, significantly reducing the time required for reports from the proposal to the initial results. Based on this, we plan to conduct a large-scale user study to evaluate the actual effectiveness of AUBIQ in improving requirement specification tasks, including indicators such as efficiency improvement and error rate reduction compared to traditional BI development processes.

In general, our experience and research show that generative AI-driven BI systems like AUBIQ represent the future direction of BI system design and use. It promotes the sinking and popularization of BI capabilities, enabling business users to complete data analysis tasks autonomously at a higher level. This will not only increase the overall data insight speed of the enterprise, but will also force traditional data engineering processes to change to adapt to a more agile, user-led BI development paradigm.

6. Related Work on Query Generation Research

Research in the field of query generation, especially text-to-SQL, has made significant progress in recent years. This direction usually uses sequence-to-sequence models to convert natural language into SQL queries, and explores a variety of encoding techniques. For example, Katsogiannis and Koutrika et al. reviewed text-to-SQL deep learning methods, involving a variety of encoding architectures such as bidirectional LSTM models, convolutional neural networks (CNNs), BERT pre-trained models, and graph neural networks (GNNs). In addition, Rubin et al. proposed using intermediate representations to assist in the conversion of natural language to SQL. In the decoding stage, the methods are mainly divided into two categories: template-based and generative methods: the template-based method relies on a fixed query template, which is fast but limited in flexibility, while the generative method requires a large amount of training data support to learn to generate queries freely.

When Text-to-SQL is applied to real-world scenarios, a series of new challenges need to be addressed. Studies have found that on real enterprise datasets, the execution accuracy of the model is significantly lower than academic benchmarks (such as Spider). The main reasons include: actual queries may involve longer contexts, business problem statements are more ambiguous, and query structures are more complex (such as nested queries). To alleviate these challenges, AUBIQ uses ontology-based pattern abstraction and semantic search to reduce the complexity of the underlying pattern, and allows users to iteratively refine queries through interactive feedback loops, thereby improving robustness and performance in practical applications. In addition, the latest research work has also proposed new ways to improve the effectiveness of Text-to-SQL.

Query accuracy assurance and automatic debugging: The correctness of query results is crucial for automatic generation systems. Some research is devoted to ensuring the reliability of generated queries. Our work goes a step further in this idea, integrating a semantic checker (ensuring that queries use ontology knowledge correctly) as well as a syntax and execution checker for deterministic verification. We emphasize that introducing conversational debugging is very important for improving the efficiency of query verification and explanation: through interaction with users, the system can better understand the deviation of user intent and correct the generated results, thereby improving the credibility and accuracy of the final query [19].

New progress in query evaluation: The establishment of large-scale benchmarks has promoted the in-depth research of Text-to-SQL. For example, the release of the Spider dataset has greatly promoted the development of query generation technology and also spawned many ranking-oriented research. However, since Spider mainly contains single-domain patterns, its limitations are becoming increasingly apparent. To this end, the BIRD dataset was proposed, which covers 37 different domains and provides a public ranking to evaluate models closer to real-world scenarios. At the same time, the WikiSQL dataset is still widely used as an entry-level evaluation due to its single-table query characteristics. Common evaluation indicators include component matching, exact matching , and execution accuracy. In our evaluation, we also use the Spider and BIRD datasets, focusing on the exact matching and execution accuracy indicators, and found that the BIRD dataset reveals many weaknesses of existing methods. It should be pointed out that BIRD has not yet fully involved the testing of conversational systems, and the CoSQL dataset fills this gap. CoSQL simulates the database query process of human-computer dialogue, emphasizing that the model needs to maintain context, respond to user feedback, and handle more complex queries. These studies remind us that a Text-to-SQL system for practical applications should focus on the ability to maintain session context, dynamically adapt to user interactions, and parse complex queries, which is also one of the key considerations in the design of AUBIQ.

The newly released Spider 2.0 dataset introduces a new level of complexity to Text-to-SQL research, more closely aligned with real-world enterprise workflows. The databases included in Spider 2.0 often have more than 1,000 fields, distributed across platforms such as BigQuery and Snowflake, requiring models to handle massive amounts of context including metadata, and to generate multi-step queries of more than 100 rows. These challenges go beyond the scope of traditional evaluation metrics and highlight the need for more robust, scalable models that combine logical reasoning with support for diverse SQL dialects and workflows. In the case of AUBIQ, the enterprise scenarios emphasized by Spider 2.0 coincide with our goal of automating BI requirements capture in complex data environments. Preliminary evaluation results for Spider 2.0 show that existing models achieve only 17% of the task completion rate on Spider 2.0, compared to 91.2% on Spider 1.0. This significant performance gap reflects the additional challenges of translating high-level business queries into SQL workflows that require consideration of dynamic context and user-refined iterations. How to deal with these challenges will be a key step in enabling systems like AUBIQ to master the complex requirements of real BI systems. This also further proves the broad potential of generative AI in promoting data-driven decision-making. We believe that by combining more powerful semantic understanding, smarter interactions, and specialized optimizations for enterprise-level problems, the next generation of Text-to-SQL systems will achieve significant breakthroughs in complex real-world environments.

7. Validity Threats

This section discusses the main validity threats faced by this study, including internal validity, external validity, and construct validity. We also describe the mitigation measures taken to address each threat. These measures are crucial to improving the reliability and generalizability of the research findings.

7.1. Internal Validity

Internal validity refers to the extent to which a study can accurately establish the causal relationship between independent and dependent variables without being interfered by potential confounding factors. In LLM-related research, ensuring internal validity means that the model performance improvements we observe are indeed due to changes in the model or training methods, rather than external factors such as data leakage, biased datasets, or improper evaluation methods. The internal validity of this study may be threatened by the following:

Data selection bias: The AdventureWorks2014 database used in our evaluation is a public sample database and may not fully represent the diversity of real BI system data. This limitation in data selection will lead to biased experimental results, thereby weakening the validity of the conclusions in a broader context. To alleviate this problem, we introduced public query benchmark datasets such as Spider and BIRD and the corresponding OWL ontologies to cover diverse domains and data structures. These datasets include databases of different scales and modes, which helps to conduct a more comprehensive evaluation of AUBIQ's performance. In future work, we plan to expand the test scope to more heterogeneous datasets, such as the newly released and larger Spider 2.0 dataset, to further verify the universality of the system.

Generated ontology quality: The quality of the ontology generated by the OntoDis tool in AUBIQ is crucial to the accuracy of the system's semantic search and query generation. If there are problems such as incorrect mapping or missing relationships in the ontology construction, these errors may run through the system process and affect the accuracy of the final query. For example, the lack of correct association between two concepts in the ontology may cause the query generation to miss the table that should be connected. To reduce this risk, we introduced an iterative refinement mechanism in OntoDis to perform multiple rounds of improvement and verification on the initial ontology. Specific measures include: automatic verification of the consistency of the ontology with the physical schema, manual inspection of the correct mapping of key concepts by domain experts, and continuous correction of discovered problems through feedback [20]. These measures help us ensure logical consistency at the ontology level and align the ontology structure with the physical data schema as much as possible. Through these efforts, we strive to ensure that semantic search and query generation are built on high-quality ontologies and improve the reliability of system output from the source.

Integration of user feedback: We rely heavily on user feedback from key information providers (i.e., domain experts who participated in the evaluation) during the iterative process of system development. This feedback is used to guide the improvement of AUBIQ. However, the professional levels of different experts vary, or they may tend to give overly positive evaluations, which may lead to deviations in the direction of improvement. To reduce such biases, we designed a feedback standardization mechanism in the system: the natural language feedback of experts is converted into structured ontology and query improvement suggestions, which are uniformly processed by the system. At the same time, AUBIQ uses a conversational interface with a conversation history record, trying to keep each user in a similar interaction context when giving feedback, so as to reduce the deviation caused by inconsistent backgrounds between different user feedback. In the future, we also plan to conduct larger-scale and more formal usability studies, introducing users from different backgrounds for testing, in order to obtain more objective and diverse feedback information, so as to continuously optimize the system [7].

Generative model dependency: AUBIQ relies heavily on LLM to understand natural language and generate queries. If there are errors in the LLM output (e.g., incorrect SQL queries are generated), the reliability of the system results will be affected. We mitigate the risks brought by LLM dependency in a number of ways. First, the aforementioned self-debugging mechanism (see Section 4.4) is integrated into the query generation stage, including syntax, semantics, and execution checks, to automatically capture and correct errors generated by LLM. Second, the system provides a query interpretability module to explain the query logic generated by LLM in natural language to help users understand the system's parsing process. This not only increases the transparency of the results, but also allows users to question and request clarification of suspicious generated results, thereby reducing the possibility of misunderstanding and ambiguity. Finally, we conduct offline testing and continuous prompt word optimization on the LLM output to minimize the probability of obvious errors in the model. Through these measures, we minimize the impact of errors that may be introduced by LLM dependency and improve the credibility of the final output of the system [21].

7.2. External Validity

External validity refers to the generalizability of research results or model performance outside of specific research conditions. This requires that the observed effects still hold true in different data sets, environments, or user contexts, ensuring that the model is also applicable in real-world scenarios. For LLM applications, external validity reflects the accuracy or consistency of the model in benchmark tests, and whether it can be consistent in different fields and unknown tasks. The external validity threats faced by this study and their mitigation methods include:

Scalability and generalization: Although we have tested AUBIQ on small and medium-sized datasets, we have not yet fully validated its scalability in enterprise-scale massive data environments. The challenges that similar systems may encounter when scaling to larger scales have been documented, such as Spider 2.0, which was found to have significantly reduced model performance in enterprise environments. To enhance the external validity of AUBIQ in large-scale environments, we adopted a modular approach in the architectural design to support distributed data queries and scalable vector databases (such as Milvus and Pinecone) for semantic search. This design helps the system scale horizontally to more data nodes. In addition, we conducted preliminary scalability tests to simulate queries on federated data sources, and the results showed that the system has some potential for scalability. Nevertheless, we plan to further evaluate AUBIQ in larger enterprise data environments, including more complex query workloads and more concurrent users, to verify the performance of the system under scale-up and identify and resolve potential bottlenecks in a timely manner.

Applicability to non-standardized data: AUBIQ currently relies on pre-defined structured schemas and OWL ontologies to work, which limits its applicability to semi-structured or unstructured data. In other words, for data sources that lack clear schemas (such as free text, NoSQL documents, etc.), the system may be unable to achieve semantic understanding and query generation. To expand the scope of AUBIQ, we have added dynamic schema adaptation capabilities to OntoDis , which can perform schema inference on new data sources. For example, when a JSON data source is connected, OntoDis will try to automatically infer its schema structure and convert it into an OWL ontology. This dynamic adaptation allows the system to cope with non-standardized data formats to a certain extent. Of course, this is only a preliminary solution. In the future, we plan to further study ontology generation and query methods for unstructured data (such as text logs), such as combining natural language processing technology to establish an ontology representation for text data, so that AUBIQ can handle a wider range of data query tasks [13].

7.3. Construct Validity

Construct validity measures whether a study or model actually measures or reflects the theoretical concepts it is designed to examine. For LLM applications, this means whether the evaluation indicators or tasks we use actually reflect the capabilities claimed by the model, such as reasoning, coherence, or factuality. Ensuring construct validity requires that the model behavior we observe corresponds to theoretical performance improvements, rather than simply reflecting irrelevant factors or superficial patterns in the data. The construct validity threats and corresponding mitigation measures of this study include:

Evaluation Metrics: When evaluating AUBIQ, we mainly used quantitative metrics such as query accuracy and execution correctness. However, these standard metrics cannot fully capture user satisfaction or the actual practical value of AUBIQ. Even if a query generated by a model is technically accurate, it may not really meet the needs of business users. To make up for this shortcoming, we introduced qualitative analysis in the evaluation: by having 23 domain experts use the system and provide informal subjective evaluations, we collected feedback on the system's ease of use and task fit. This feedback from actual users provides us with insights into the practicality of the system as a useful supplement to purely quantitative metrics. In future evaluations, we also consider using methods such as questionnaires or usability scores to quantify user experience dimensions to fully evaluate the effectiveness of AUBIQ in real tasks.

Comparison with existing systems: To validate AUBIQ's performance, we benchmarked it against state-of-the-art text-to-SQL and semantic search systems, using datasets such as Spider and BIRD to simulate real-world enterprise scenarios. The results show that AUBIQ has competitive query accuracy and is superior in iterative query generation with user interactions. However, we have not yet compared AUBIQ with other BI dashboard generation or acceleration tools. For example, some commercial BI tools can automatically generate data visualizations and reports, which may have different strengths in interpretability and usability. The current comparison is mainly limited to systems in the field of Text-to-SQL (e.g., accuracy, error correction, etc.). Therefore, there is still some threat to construct validity: we claim that AUBIQ has advantages in accelerating BI dashboard creation, but there is a lack of direct comparison with technologies that go beyond Text-to-SQL. To this end, in the future, we plan to conduct a comprehensive comparison of AUBIQ's capabilities in explanation generation and visualization recommendation with current BI tools. This will include comparative evaluations of metrics such as user workflow efficiency, quality of generated results, and user satisfaction to ensure that our understanding of AUBIQ's strengths and weaknesses accurately reflects its true capabilities.

8. Conclusion and future work

This paper proposes a code-free framework, AUBIQ, which aims to simplify and accelerate the creation and specification of functional analytical requirements in business intelligence (BI) environments. Through a conversational natural language interface, AUBIQ can interpret business users' natural language input into query code, data models, query interpretation instructions, execution results, and corresponding data visualization representations. The tool is highly customizable, for example, it can be extended to support new query languages, adjust the data model discovery process, or replace different platforms to execute queries to meet the needs of specific BI scenarios. With a simple and intuitive prompt mechanism, both business users and data professionals can guide the system to customize outputs and adjust the visualization as needed to make the results more in line with their specific needs.

AUBIQ framework has been implemented and improved in several actual customer projects, with application goals ranging from looking ahead to future BI system blueprints, enhancing existing data products and integration processes, to prototyping new analytical solutions. By automatically generating business-oriented reports and providing detailed technical specifications, AUBIQ effectively lowers the technical threshold for business users to understand and use data analysis, allowing them to focus on business logic; at the same time, data engineers can quickly implement solutions based on the technical details provided by the system, bridging and synergizing business and technical perspectives. This bridging role helps accelerate the analytical development cycle and ensure that technical implementation is consistent with business goals.

As the system evolves, our goal is to further democratize data analytics, allowing more users to easily access and leverage data-driven insights. The code-free automation enabled by AUBIQ marks an important step forward, allowing users to define analytical requirements and explore new possibilities with little to no technical background. We will continue to introduce the latest research results and emerging technologies to improve system performance and user experience to meet current needs and anticipate future challenges. For example, we plan to integrate more advanced language models, optimize semantic search algorithms, and improve visualization recommendations to cope with more complex analytical tasks.

We have implemented mitigation measures for the various validity threats described in the previous section to ensure that AUBIQ is robust, scalable, and easy to use. Future work will build on these measures, including: expanding the size and diversity of the datasets used for training and testing, improving the methods for ontology generation and alignment, and conducting large-scale user studies and comparative experiments with other techniques. These efforts will help us continue to improve the reliability and generalization ability of AUBIQ.

This study also highlights the importance of continuing to explore advanced techniques, such as automated ontology discovery, stronger semantic search, and robust model alignment. We recognize the need to develop a methodology to enable guided, secure, and explainable query generation, and to establish evaluation benchmarks to measure the effectiveness, usability, and security of data analysis tools. Our ultimate goal is to provide a reliable, user-friendly solution that meets evolving industry standards and creates new opportunities in the BI space.

It is worth pointing out that for small and medium-sized enterprises, the method proposed in this paper has significant application value in actual operations. Small and medium-sized enterprises often lack a large professional technical team, and AUBIQ can significantly reduce the professional threshold required to deploy a BI system, allowing such enterprises to build their own data analysis capabilities without relying heavily on external experts. This not only reduces communication costs and iteration cycles, but also speeds up the implementation of data-driven decision-making, allowing small and medium-sized enterprises to obtain business insights more timely and maintain competitiveness. In addition, by automating requirements specifications and report generation, small and medium-sized enterprises can free limited human resources from tedious technical details and focus on analyzing and solving core business problems. These advantages show that the AUBIQ method has the potential to become a powerful tool for small and medium-sized enterprises on the road to digital transformation, helping them embrace data-driven business decision-making models at a lower cost and higher efficiency.

References

[1] Krishna, S.H., et al. Generative AI in Business Analytics by Digital Transformation of Artificial Intelligence Techniques. in 2024 International Conference on Communication, Computer Sciences and Engineering (IC3SE). 2024. IEEE.

[2] Hendricks, F., Business value of Generative AI use cases. Journal of AI, Robotics & Workplace Automation, 2024. 3(1): p. 47-54.

[3] Busany, N., et al., Automating Business Intelligence Requirements with Generative AI and Semantic Search. arXiv preprint arXiv:2412.07668, 2024.

[4] Yafei, X., et al., Generative AI in Industrial Revolution: A Comprehensive Research on Transformations, Challenges, and Future Directions. Journal of Knowledge Learning and Science Technology ISSN: 2959-6386 (online), 2024. 3(2): p. 11-20.

[5] Paulose, R. and V. Neelanath. Generative AI-Driven Automation of Business Process ReImagination. in 2024 IEEE Recent Advances in Intelligent Computational Systems (RAICS). 2024. IEEE.

[6] Benala, T.R. and S. Dehuri, Transforming Software Development: From Traditional Methods to Generative Artificial Intelligence, in Boosting Software Development Using Machine Learning. 2025, Springer. p. 1-13.

[7] Reznikov, R., Leveraging generative AI: Strategic adoption patterns for enterprises. Available at SSRN 4851632, 2024.

[8] Ramamoorthy, L., Evaluating Generative AI: Challenges, Methods, and Future Directions. International Journal of Future Multidisciplinary Research, 2025. 7(1).

[9] Korinek, A., Generative AI for economic research: Use cases and implications for economists. Journal of Economic Literature, 2023. 61(4): p. 1281-1317.

[10] Mudrinić, D. and I. Šoda. The Rise of Generative Artificial Intelligence in Business. in 2024 IEEE 17th International Scientific Conference on Informatics (Informatics). 2024. IEEE.

[11] Ellingrud, K., et al., Generative AI and the future of work in America. 2023.

[12] Doron, G., et al., Generative AI: driving productivity and scientific breakthroughs in pharmaceutical R&D. Drug Discovery Today, 2024: p. 104272.

[13] Health–Europe, T.L.R., Embracing generative AI in health care. The Lancet Regional Health-Europe, 2023. 30: p. 100677.

[14] Gupta, R., et al., Adoption and impacts of generative artificial intelligence: Theoretical underpinnings and research agenda. International Journal of Information Management Data Insights, 2024. 4(1): p. 100232.

[15] Su, J. and W. Yang, Unlocking the power of ChatGPT: A framework for applying generative AI in education. ECNU Review of Education, 2023. 6(3): p. 355-366.

[16] Preiksaitis, C. and C. Rose, Opportunities, challenges, and future directions of generative artificial intelligence in medical education: scoping review. JMIR medical education, 2023. 9: p. e48785.

[17] Occhipinti, J.-A., et al., The recessionary pressures of generative AI: A threat to wellbeing. arXiv preprint arXiv:2403.17405, 2024.

[18] Routray, S.K., et al. Generative Artificial Intelligence: Principles, Potentials and Challenges. in 2024 2nd International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS). 2024. IEEE.

[19] Evans, D.S., A. Hagiu, and R. Schmalensee, Invisible engines: How software platforms drive innovation and transform industries. 2008: The MIT Press.

[20] Wessel, M., et al., Generative AI and its transformative value for digital platforms. Journal of Management Information Systems, 2023.

[21] Chatterjee, S., et al., Towards new frontiers of healthcare systems research using artificial intelligence and generative AI. 2024, Taylor & Francis. p. 263-273.