

Structured AI: A Three-Tier Adaptive Framework for Enterprise Data Analytics

Bhabya Mohanty*, Gopichand Agnihotram and Raja Sekhar Reddy S

Wipro Innovation Labs, Wipro Kodathi SEZ, Survey No. 91/1, 98 & 99, Hadosiddapura, Varthur Hobli, Sarjapur Road, Bangalore, Karnataka, 560099, India

*Corresponding Author

Bhabya Mohanty

✉ bhabya.mohanty@wipro.com

Received: 12 March 2026

Revised: 08 October 2026

Accepted: 16 October 2026

Published Online: 26 October 2026

Citation

B Mohanty, G. Agnihotram, R. S. Reddy S, Structured AI: A three-tier adaptive framework for enterprise data analytics, *Journal of Smart Sensors and Computing*, 2026, 2(4), 26216, doi: <https://doi.org/10.64189/ssc.26216>.

Open Access

This article is licensed under a [Creative Commons Attribution-NonCommercial 4.0 International License](https://creativecommons.org/licenses/by-nc/4.0/), which permits the non-commercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as appropriate credit to the original author(s) and the source is given by providing a link to the Creative Commons License and changes need to be indicated if there are any. The images or other third-party material in this article are included in the article's Creative Commons License, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons License and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this License, visit: <https://creativecommons.org/licenses/by-nc/4.0/>

© The Author(s) 2026

Abstract

Traditional Retrieval-Augmented Generation (RAG) systems and SQL-based query engines face significant challenges when processing real-world Excel files containing multiple tables, merged cells, multi-row headers, formulas, pivot-style layouts, and inconsistent schemas. Structured AI is a hybrid three-tier framework that assesses workbook scale, structural cleanliness, and semantic relevance before routing a query to document-centric processing, SQL-optimized processing, semantic sheet selection, or a coordinated hybrid path. The methodology contributes a measurable cleanliness model, query-sheet relevance scoring, cost-aware route selection, formula-preserving conversion, and confidence-based result synthesis. In the reported prototype evaluation, average query accuracy increased from 78% for a static baseline to 89% for the adaptive framework, while median latency on large multi-sheet workbooks decreased from 21.5 s to 12.4 s. These values are reported consistently throughout the manuscript. Because the submitted source did not include the underlying workbook corpus, run-level logs, or sample counts, the revised paper explicitly limits the empirical claim to the reported aggregate evaluation and identifies the evidence required for independent replication.

Keywords: RAG; Hybrid architecture; Excel processing; Adaptive routing; Document understanding; Query optimization; Enterprise data analytics.

1. Introduction

Structured data has long been the foundation of enterprise information systems, beginning with early relational database models and later formalized through SQL query languages. Classical work established the relational model for database management, showing how structured tables with well-defined schemas enable efficient data retrieval and manipulation.^[1] Building on this, database design and normalization frameworks reinforced the theoretical basis for structured data processing that subsequently became instrumental in business intelligence and analytics systems.^[2] Modern artificial intelligence introduces both opportunities and challenges for this legacy of structured data processing. Large Language Models (LLMs) demonstrate strong natural language understanding capabilities, yet their direct application to structured data is constrained by context window limits and computational cost at scale.^[3] Retrieval-Augmented Generation (RAG) extends knowledge retrieval by combining document search with generative models, allowing systems to ground responses in external data sources.^[4] This paradigm has further been adapted for tabular data, linking natural language queries to structured information retrieval and thereby bridging conversational interfaces with database operations.^[5] The challenge becomes more severe when we consider real-world enterprise data rather than idealized relational schemas. Enterprise Excel workbooks rarely conform to clean database structures: spreadsheets in organizational settings frequently contain multiple tables per sheet, merged cells for visual formatting, and inconsistent header structures, all of which violate the assumptions of traditional SQL engines and many modern document parsers.^[6,7] This “messy data” problem as a critical bottleneck in analytics pipelines, where manual preprocessing consumes significant analyst time and introduces human error. Research also provide empirical evidence that over 60% of enterprise Excel files contain structural irregularities that prevent direct loading into databases, underscoring the gap between theoretical data models and practical data reality.^[8] In parallel, document understanding techniques have advanced rapidly, and demonstrate how Docling and related document-conversion frameworks can extract structured information from complex layouts, directly improving table detection and content preservation in technical and business documents.^[9] This focus on formula preservation in spreadsheet processing, arguing that calculated values encode critical business logic that must be maintained during data extraction.^[10] Some paper also extended these ideas to multi-sheet workbooks, showing that semantic sheet selection based on query relevance improves both accuracy and efficiency compared with processing all sheets indiscriminately. Industry adoption of hybrid approaches

that combine these strands is accelerating.^[11] This paper report that systems which execute SQL over clean data while delegating unstructured or irregular content to LLM-based processing achieve better accuracy-efficiency trade-offs than single-method architectures.^[12] This paper document the emergence of adaptive query-routing systems that dynamically select processing strategies based on data characteristics, highlighting both the performance gains and the need for principled decision frameworks in environments where data quality varies widely across sources.^[13]

Recent surveys synthesize the landscape of natural language interfaces to structured data, arguing that future analytics platforms will be inherently hybrid and context-aware.^[14] Taken together, these works reveal a clear convergence. Classical relational database theory provides the foundation for structured query processing, while modern generative AI operationalizes natural language interfaces to these systems. RAG frameworks^[4] and LLM-based query generation^[3] form the bridge between conversational interaction and data retrieval, while more recent literature extends this bridge to messy, real-world enterprise data.^[6-8] Contemporary studies demonstrate the maturation of hybrid architectures that adapt processing strategies to data characteristics, showing that in enterprise environments where data quality, processing efficiency, and query accuracy are critical, adaptive hybrid systems are not merely architecturally elegant but practically necessary.^[12-14] The remainder of this paper builds on these foundations by presenting Structured Ai’s three-tier architecture, detailing its adaptive routing mechanisms and semantic sheet-selection procedures, and exploring engineering implications for enterprise analytics systems in which Excel files remain the dominant data exchange format despite their structural inconsistencies.

1.1 Problem statement

Let $W = \{S_1, \dots, S_m\}$ denote an Excel workbook with m sheets and let q denote a natural-language analytical query. Each sheet can vary in size, header depth, merged-cell density, formula density, type consistency, and semantic relevance to q . A fixed pipeline is suboptimal: SQL is efficient but brittle on irregular layouts, whereas direct LLM processing can preserve context but incurs high token and latency costs. The research problem is therefore to select, for every workbook-query pair, the processing route that maximizes answer quality while controlling latency, token consumption, and execution failure risk. Examples include a clean sales table suited to NL-to-SQL, a compact financial report with hierarchical headers suited to document-centric reasoning, and a 40-sheet operational workbook in which only a few sheets are relevant to the question.

1.2 Research questions and contributions

RQ1: Can measurable workbook characteristics support reliable route selection among SQL, document-centric, semantic-selection, and hybrid processing? RQ2: Can semantic pre-selection reduce the amount of workbook content processed without reducing answer quality? RQ3: Can formula- and layout-preserving conversion improve robustness on irregular sheets? The innovative contribution is not a new parser or NL-to-SQL model in isolation. It is the end-to-end coordination of (i) a continuous structural-quality score, (ii) semantic sheet and region selection, (iii) a cost-aware routing objective, (iv) specialized conversion that retains formulas and layout semantics, and (v) confidence- and provenance-aware synthesis across multiple execution paths. This combination converts workbook heterogeneity from an exception into an explicit decision variable.

1.3 Related work and research gap

Relational database theory establishes efficient execution when schemas are stable, while few-shot language models and RAG provide flexible natural-language access to external knowledge.^[1-3] Recent table-question-answering and spreadsheet-understanding studies show that schema linking, layout preservation, and hybrid retrieval are important for irregular business data.^[4,7] However, the reviewed approaches generally optimize one pathway at a time. The gap addressed here is a unified controller that measures workbook conditions, selects among complementary pathways, and records feedback for threshold re-calibration.

2. Methodology

2.1 Theoretical framework for adaptive query processing

The Structured AI methodology is grounded in adaptive systems theory and contemporary information processing paradigms, treating query handling as a closed-loop decision process driven by observable data characteristics rather than fixed pipelines. The core premise is that optimal query processing strategies should emerge from dynamic assessment of properties such as dataset size, structural regularity, and type consistency, instead of relying on preconfigured, one-size-fits-all workflows. This view is closely aligned with control theoretic principles, in which system behaviour adapts to environmental feedback in this case, feedback derived from data quality metrics, workload patterns, and computational constraints. Within this framework, Structured AI is built on three foundational principles: adaptive routing, semantic relevance, and hybrid processing coordination. Adaptive routing recognizes that heterogeneous data structures (clean tables, messy spreadsheets, multi-sheet workbooks) demand fundamentally different processing strategies

and therefore must be routed to specialized pipelines rather than forced through a uniform path. Semantic relevance ensures that computation is focused on content that is most pertinent to a given query, which is particularly important for large, multi-sheet Excel workbooks where only a subset of sheets or regions contributes meaningfully to the answer. Hybrid processing coordination then orchestrates the parallel execution of complementary strategies such as SQL-based analytics on clean tables and document-centric or LLM-based processing on irregular regions so that the system maximizes both accuracy and efficiency while producing a single, coherent

2.2 Three-tier architecture design

The Structured AI three tier architecture [Fig. 2](#), introduces explicit decision boundaries that map heterogeneous Excel workbooks to specialized processing pipelines using empirically calibrated thresholds on size, structure, and cleanliness indicators. Each tier is optimized for a distinct data regime and collectively they form an end to end, adaptive document processing system that balances accuracy, latency, and computational cost in enterprise settings.

2.2.1 Tier 1: Document centric processing

Tier 1 targets small but structurally complex datasets, typically workbooks with fewer than 10,000 cells that exhibit rich formatting, irregular layouts, or dense semantic structure not easily captured by flat tables. In this regime, the additional latency of advanced document processing is justified because layout, visual grouping, and formatting encode information that would be lost if the workbook were naively flattened into relational tables. Structured AI therefore applies layout aware document AI techniques that preserve spatial relationships, multi region tables, headers, captions, and surrounding narrative context during conversion to an LLM-consumable representation.^[9] Conceptually, Tier 1 assumes that structural irregularities in small datasets such as merged cells, multi row headers, nested subtotal blocks, and embedded text boxes are intentional design choices that contribute meaningfully to interpretation and query answering. Instead of treating these patterns as noise, the conversion pipeline explicitly detects and models them, propagating header hierarchies, expanding merged cells into dense representations, and retaining row- and column-level groupings so that downstream LLM components can reason over the original business logic. This preserves information density, allowing the system to answer nuanced queries about complex reports (for example, hierarchical financial statements or KPI dashboards) that would otherwise lose semantics under purely tabular extraction.

2.2.2 Tier 2: SQL-optimized processing

Tier 2 is designed for large, clean, and structurally regular datasets where the spreadsheet closely approximates a relational table and thus benefits from classical database optimization techniques. Its theoretical foundation draws on decades of work in relational algebra and query planning, which show that when data conforms to relational assumptions, single header row, consistent column types, minimal merged cells, SQL engines provide near-optimal performance and strong consistency guarantees.^[1,2] In Structured AI, this tier performs dynamic schema inference, constructing candidate table schemas by analysing headers, data types, and value distributions, and then translates natural language queries into executable SQL statements. Tier 2 operates under the assumption that clean structure is a signal of intentional modelling for analysis: if analysts have already normalized the workbook into regular tables, the most effective strategy is to leverage relational execution rather than route everything through the LLM. Header normalization, type inference, and constraint detection (such as keys and foreign keys inferred from patterns) are used to generate a high-quality schema, while schema-aware prompt engineering guides the NL-to-SQL generation so that the model uses real column names, respects join paths, and selects appropriate aggregations. The result is a pipeline that delivers database-level performance and accuracy while preserving a natural language interface for end users.

2.2.3 Tier 3: Embedding-based sheet selection

Tier 3 addresses the scalability problem inherent in large, heterogeneous workbooks that may contain tens of sheets, mixed-quality tables, and auxiliary content such as notes or charts. Processing every sheet with high-fidelity document conversion or full SQL ingestion would be prohibitively expensive and often unnecessary, because only a subset of sheets is relevant to any particular query. Structured AI therefore introduces an embedding-based semantic filtering layer: each sheet (or preview region) is mapped into an embedding space, and sheet–query similarity scores are used to select a smaller set of candidate sheets for downstream processing.^[11] This tier rests on the observation that semantic relevance can be approximated by vector similarity in high-dimensional embedding spaces, enabling the system to prioritize content likely to contain the answer without exhaustive parsing. By applying adaptive relevance thresholds (e.g., looser thresholds for more complex queries that require broader context), Tier 3 reduces the number of sheets that need full processing, which in turn lowers token usage, memory footprint, and end-to-end latency. In practice, this semantic pre-filtering yields substantial computational savings while preserving or

improving accuracy, because noisy or off-topic sheets are excluded before expensive processing stages are invoked.

Hybrid processing mode

While the three tiers can operate independently, real-world enterprise workbooks frequently contain a mixture of clean and messy sheets, making a single-tier strategy suboptimal. The hybrid processing mode therefore orchestrates parallel execution of Tier 1 and Tier 2 pipelines over different subsets of sheets that have been classified by their structural quality and semantic role.^[12,13] Clean, relational-like sheets are routed to Tier 2 for SQL-based analytics, whereas irregular or layout-heavy sheets are routed to Tier 1 for document-centric conversion and LLM reasoning. A coordination layer then aggregates and reconciles the partial results from each tier, using confidence scores and consistency checks to synthesize a single, coherent answer for the user. This synthesis step may, for example, join numerical aggregates computed via SQL with contextual explanations extracted from richly formatted narrative sheets, thereby combining the precision of structured processing with the interpretability and flexibility of document understanding. By enabling such mixed mode execution within a single query, the hybrid processing mode turns the three-tier architecture into an integrated, adaptive system capable of handling the full spectrum of Excel based enterprise data at scale.

2.3 Semantic sheet selection methodology

The semantic sheet selection methodology is designed to address the central challenge of locating query-relevant content within complex, multi-sheet Excel workbooks without incurring the cost of fully processing every sheet. In many enterprise scenarios, a workbook may contain a mixture of raw data tables, intermediate calculations, archived sheets, and presentation-oriented summaries, only a subset of which is actually relevant to a given natural language query. Processing all sheets at full fidelity would significantly increase token usage, latency, and memory consumption, and it would also introduce noise that can confuse downstream models. To avoid this, Structured AI employs a two-stage selection pipeline that first generates compact semantic previews and then performs query–sheet matching in an embedding space. In the first stage, the system constructs a lightweight preview for each sheet rather than parsing the entire workbook in detail. These previews typically include a small but representative sample of header rows, a subset of data rows, key formulas, and any high-signal textual elements such as titles or annotations. The preview extraction logic is tuned to capture the “semantic signature” of a sheet—enough content to characterize what the sheet is about (e.g., “regional sales by quarter”,

“inventory aging”, “expense approvals”) while keeping the token budget small and the processing overhead low. This allows Structured AI to scale to workbooks with dozens of sheets by treating each page initially as a concise semantic candidate rather than a full document. The second stage performs query-sheet matching using embedding similarity. Both the user’s natural language query and each sheet preview are encoded into vectors using a modern sentence or document embedding model capable of capturing semantic relationships beyond simple keyword overlap. The system then computes similarity scores, typically cosine similarity between the query embedding and each sheet embedding, producing a ranked list of sheets ordered by estimated relevance. An adaptive thresholding mechanism determines which sheets are passed on for deeper processing; for straightforward queries a higher similarity threshold can be used to keep the candidate set small, while more complex or ambiguous queries may use a lower threshold to ensure broader coverage. This methodology rests on the empirically validated assumption that semantic proximity in embedding space correlates strongly with query relevance, even when sheet titles or column names are noisy or partially mismatched with the user’s wording. Because the approach does not depend on domain-specific ontologies or manually designed rules, it generalizes well across different business domains and spreadsheet conventions. In practice, semantic sheet selection substantially reduces the number of sheets subjected to expensive document conversion or SQL ingestion, improving latency and cost efficiency while often improving accuracy by excluding irrelevant or misleading content from the reasoning process.

2.4 Enhanced document conversion framework

The enhanced document conversion framework extends conventional document AI pipelines to handle Excel-specific phenomena that strongly affect query answering quality: formulas, multi-row headers, and merged cells. Standard document processing techniques often treat spreadsheets as simple grids of values, discarding the computational logic and structural cues that analysts have encoded into the workbook layout. Structured AI’s framework instead views each Excel file as a composite artifact that encodes both data and business logic, and it aims to preserve this dual nature during conversion to intermediate representations used by LLMs and downstream analytics engines. The first enhancement is formula preservation. Rather than replacing all formulas with their evaluated values, the framework parses formulas, identifies dependencies, and stores both the formula expression and its result.^[10] This allows the system to reason about how values are derived (for example, distinguishing raw input cells from calculated KPIs or margins) and to interpret queries that refer

explicitly or implicitly to business rules, such as “calculated profit margin” or “variance against budget”. Preserving formula structure also supports scenarios like recomputing metrics under alternative assumptions or validating that calculated fields are consistent across sheets. The second enhancement is multi-row header consolidation. Many enterprise spreadsheets use stacked header rows to express hierarchical dimensions—such as Region → Country → City, or Year → Quarter → Metric—leading basic converters to misalign or fragment column names. Structured AI analyses header patterns, identifies vertical groupings, and synthesizes composite column labels (for example, “Sales.Q1.2025” or “Expense. Marketing. Travel”) that retain the hierarchy while producing clean, machine-usable schema elements.^[7,8] This consolidation significantly improves downstream tasks such as schema inference, NL-to-SQL mapping, and semantic search over columns. The third enhancement is merged cell handling. Merged cells are often used to visually indicate grouping or shared categories across ranges of rows or columns, but naïve extraction yields sparse matrices with missing values that obscure these relationships. The enhanced framework propagates merged cell content across the full merged region, adding markers where needed to indicate that values were originally shared rather than repeated. As a result, tabular representations faithfully encode grouping semantics, and LLMs can more easily infer relationships such as “these rows belong to the same department” or “these columns are sub-metrics under the same heading”. Underlying these enhancements is the broader principle that Excel workbooks encode organizational knowledge, not just raw numbers. By preserving formulas, header hierarchies, and merged-cell semantics, the document conversion framework ensures that this knowledge survives the transition into vectorized or text-based representations, thereby improving both the factual correctness and the explanatory quality of the system’s responses.

2.5 Adaptive decision framework

The adaptive decision framework acts as the control layer that unifies semantic sheet selection, enhanced document conversion, and tiered routing into a coherent, data-driven strategy selection mechanism. Unlike static pipelines that apply the same sequence of operations to all inputs, this framework evaluates multiple aspects of each workbook and query in real time and then selects an execution plan that best balances accuracy, latency, and resource usage for that specific case. Key features considered include dataset size (cell counts, sheet counts), structural cleanliness (header regularity, merged-cell density, type consistency), and content heterogeneity (mix of clean tables and layout-heavy reports). Formally, these features are aggregated into

composite scores that characterize both the cost and the expected benefit of applying different processing strategies, such as pure SQL, document-centric LLM analysis, or hybrid combinations. The decision logic goes beyond simple if-else rules by treating routing as a multi-dimensional optimization problem: small, messy spreadsheets with rich layout are steered toward document-centric processing; large, clean tables are sent to SQL engines; and highly heterogeneous workbooks trigger hybrid plans that combine multiple tiers and semantic filtering. By modelling the interaction between data characteristics and computational constraints, the framework can avoid pathological cases such as sending a giant, messy workbook entirely through an LLM or attempting to force a poorly structured sheet into a rigid relational schema. A critical component of the framework is its feedback loop. After each query is executed, the system records metrics such as processing time, memory usage, token consumption, error or correction rates, and user satisfaction indicators. These signals are used to revise internal thresholds (for example, the 10,000-cell cutoff for Tier 1), re-weight feature importance, and refine heuristic or learned models that predict which strategy will perform best for a given input profile. Over time, this leads to a self-tuning system that adapts to evolving workloads, infrastructure constraints, and model capabilities rather than relying on static configurations chosen at design time. Taken together, the semantic sheet selection methodology, enhanced document conversion framework, and adaptive decision framework provide a theoretical and practical foundation for building intelligent document processing systems that remain robust under real-world data complexity. They enable Structured AI to maintain high query accuracy and predictable performance even as workbook structures, scales, and usage patterns vary widely across enterprise environments

2.6 Mathematical formulation and algorithmic flow

For sheet S_i , define a structural feature vector $x_i = [n_i, h_i, m_i, f_i, t_i, a_i]$, where n is normalized size, h is header complexity, m is merged-cell density, f is formula density, t is type inconsistency, and a is non-tabular artifact density. The cleanliness score C_i and semantic relevance $R_i(q)$ are defined as follows.

$$C_i = 1 - (w_h h_i + w_m m_i + w_f f_i + w_t t_i + w_a a_i), \quad w_k \geq 0, \sum_k w_k = 1 \quad (1)$$

$$R_{i(q)} = \cos(e(q), e(S_i)) = \frac{e(q)^T e(S_i)}{\|e(q)\|_2 \|e(S_i)\|_2} \quad (2)$$

For route r in {document, SQL, semantic-selection, hybrid}, the controller minimizes a weighted utility loss that combines expected answer error, latency, token cost, and execution risk. The coefficients are deployment parameters and must be calibrated on held-out

workloads rather than treated as universal constants.

$$r^* = \operatorname{argmin}_r \{ \alpha E_{r(x,q)} + \beta L_{r(x,q)} + \gamma T_{r(x,q)} + \delta F_{r(x,q)} \} \quad (3)$$

When multiple routes execute, partial answers y_r are combined using normalized confidence c_r and provenance checks.

$$y = \sum_r \left(\frac{c_r}{\sum_j c_j} \right) y_r \quad (4)$$

subject to schema, provenance, and consistency validation.

2.7 Algorithm 1: Structured AI adaptive query processing

Input: workbook W , query q , route parameters Θ ; Output: validated answer $\hat{y}$ with provenance.

1. Extract lightweight previews, formulas, header patterns, size indicators, and structural-quality features for every sheet.
2. Compute C_i and $R_i(q)$; retain sheets whose relevance exceeds the query-adaptive threshold $\tau(q)$.
3. Estimate route utility using Equation (3). Select document, SQL, semantic-selection, or hybrid execution.
4. Execute the selected path or paths. Validate SQL syntax/schema/types and validate document-derived evidence against source regions.
5. Reconcile partial answers using Equation (4); reject or flag conflicts that fail consistency checks.
6. Return the answer, selected provenance, confidence indicators, and operational metrics for later threshold calibration.

3. System architecture and implementation framework

The proposed system architecture and implementation framework Fig. 2, integrates adaptive routing, data quality assessment, natural language query processing, semantic content selection, hybrid processing coordination, and performance optimization into a unified workflow. The framework first determines the most appropriate processing strategy through an adaptive routing decision mechanism while assessing the quality and characteristics of incoming data. It then processes natural language queries to identify user intent and relevant information requirements, followed by semantic content selection and relevance ranking to retrieve the most appropriate information. The framework further coordinates hybrid processing components and synthesizes their outputs into coherent results. Finally, performance optimization and scalability mechanisms are incorporated to ensure efficient execution, responsiveness, and reliable operation as the volume and complexity of queries and data increase.

3.1 Adaptive routing decision framework

The adaptive routing decision framework is the core architectural component that transforms Structured AI from a static pipeline into a context-aware execution engine capable of tailoring its behaviour to each incoming workbook-query pair. Instead of committing all inputs to a single processing strategy, the framework continuously evaluates the characteristics of the data and the query, then selects the most appropriate tier and configuration parameters for that specific instance. This shifts the system from a design where engineers hard-code rules for “typical” workloads to one in which routing decisions are computed dynamically from observable signals, such as size, structure, cleanliness, and heterogeneity. At the heart of the framework is a hierarchical decision tree that encodes empirically derived thresholds and decision criteria. [12,13] The first stage of the tree focuses on scale, computing cell counts and sheet counts to distinguish small workbooks from large ones; datasets below roughly 10,000 cells are considered candidates for document-centric processing because rich layout can be preserved without incurring prohibitive cost, whereas larger datasets trigger more efficiency-oriented strategies. Subsequent stages analyse structural signals, including the proportion of merged cells, the presence of multi-row headers, and type consistency across columns, to compute cleanliness scores and identify whether a sheet behaves like a relational table or like a complex, layout-heavy report. By combining these size and structure assessments, the decision tree routes each sheet or workbook either toward Tier 1 (document-centric), Tier 2 (SQL-optimized), Tier 3 (semantic selection), or a hybrid configuration that activates multiple tiers in parallel. Crucially, the routing framework is not purely rule-based; it incorporates feedback loops that allow the system to refine thresholds, weights, and routing preferences over time. After each query is processed, the system logs performance metrics such as latency, token usage, error rates, and user level success indicators (for example, whether the answer was accepted or required follow up corrections). These logs feed into an offline or online optimization process that can adjust decision boundaries tightening size thresholds for particularly expensive models, relaxing cleanliness requirements when SQL performance is consistently strong, or re-weighting certain features if they prove more predictive of downstream accuracy than originally assumed. In effect, the routing framework learns from operational experience: as more workbooks and queries flow through the system, routing decisions become better calibrated to real enterprise workloads, yielding a virtuous cycle of improved accuracy, lower cost, and more predictable performance.

3.2 Data quality assessment and classification

The data quality assessment and classification layer provides the diagnostic backbone for Structured AI’s routing and processing decisions, ensuring that each workbook is treated according to its actual structural properties rather than optimistic assumptions about cleanliness. Instead of relying on a few superficial checks, the framework performs a comprehensive evaluation that examines header organization, data regularity, and the presence of non-tabular artifacts across all sheets. This richer characterization enables the system to distinguish, for example, between a well-structured fact table with minor formatting noise and a heavily formatted management report whose layout carries much of the meaning. A central component of this layer is the header complexity analysis, which operates at the row level to detect multi-row header structures and hierarchical label patterns that are common in enterprise spreadsheets. The system scans the top portion of each sheet, looking for contiguous regions of non-numeric cells, merged spans, and repeated label patterns that indicate stacked headers rather than ordinary data rows. It interprets explicit formatting indicators such as bold fonts, shading, and border styles alongside implicit cues like repeated categories and indentation to infer whether multiple header rows collectively define a logical column name hierarchy. This analysis not only informs the complexity score but also feeds downstream components that will later consolidate multi-row headers into machine-usable schema elements. Beyond headers, the framework performs structural regularity assessment across entire sheets. It evaluates cell population patterns to identify continuous blocks of data, irregular gaps, and outlier regions that may correspond to embedded charts, comments, or annotation blocks. It also computes type consistency metrics for each column and row, measuring how frequently a column switches between numeric, textual, date, or formula-based content. Columns with stable types and minimal anomalies are strong candidates for relational treatment, whereas columns with highly mixed types or sporadic formulas may require document-centric handling. Formatting uniformity consistent number formats, alignments, and styles, is likewise used as a proxy for structural regularity. The system flags irregular patterns that suggest specialized processing, such as high densities of merged cells, overlapping objects (charts, shapes), or regions where formulas dominate raw values. These indicators help the classifier understand where traditional table extraction would fail or where naive flattening would destroy business semantics, prompting alternative strategies such as Tier 1 document-centric processing or targeted extraction of specific regions. Crucially, the classification framework is built on the principle that data quality exists on a continuous spectrum, not as a binary clean/messy label. Each sheet

is assigned scores along several dimensions' header complexity, type consistency, merged-cell density, and artifact presence, rather than a single yes/no tag. This nuanced representation allows the routing engine to make more sophisticated decisions, such as treating part of a workbook as SQL-ready while directing other sections to document-centric pipelines, or selecting hybrid strategies when sheets fall into intermediate quality bands.

3.3 Natural language query processing pipeline

The natural language query processing pipeline is responsible for converting a user's free-form question into an executable plan that respects the structure and semantics of the underlying data sources. Unlike generic question-answering systems, Structured AI must interpret queries in the context of concrete tables, formulas, and report layouts, which requires a schema-aware and context-aware approach. To achieve this, the pipeline integrates database metadata, sampled cell contents, and prior examples into the prompt and validation process, enabling high-fidelity translations from natural language to SQL or document reasoning tasks.^[4,5]

The pipeline begins with linguistic preprocessing, including tokenization, part-of-speech tagging, and entity recognition tailored for data analytics terminology (such as measures, dimensions, time periods, and aggregations). Detected entities are mapped to candidate schema elements table names, column headers, or derived metrics using fuzzy matching and semantic similarity, taking into account the consolidated headers produced during document conversion. This mapping phase constrains the search space for subsequent translation and helps disambiguate overloaded terms like "margin" or "balance" based on their occurrence in the workbook. Next, the system constructs schema-aware prompts that fuse query text with structural context. These prompts may include abbreviated schema listings, examples of typical rows, and hints about relevant relationships (joins, foreign keys), all generated from the inferred schema and data samples. Few-shot learning is used to inject domain-specific knowledge: the prompt incorporates examples of previously successful translations for similar query patterns, enabling the model to align new queries with established idioms in that organization's reporting practices. Once candidate SQL queries or reasoning specifications are generated, the pipeline applies a multi-stage validation process to guard against hallucinations and structural errors. Syntax validation checks that SQL statements are well-formed; semantic validation verifies that all referenced tables and columns actually exist and that joins follow valid keys; type validation ensures that operations such as aggregations and comparisons are type-compatible

(for instance, not summing text fields or comparing dates to strings). A feasibility assessment estimates query cost and detects potentially dangerous operations, such as unbounded Cartesian products or extremely broad scans on massive tables, allowing the system to reject or revise problematic plans before execution. Finally, the pipeline integrates domain-specific knowledge through curated examples and templates that capture common business questions quarterly growth, year-over-year comparisons, top-N rankings, variance analysis and their canonical translations in the target environment. By leveraging few-shot learning over these examples, Structured AI can accurately handle organization-specific terminology, abbreviations, and KPI definitions while still generalizing to new combinations and formulations of queries. This makes the query processing pipeline robust enough for high-stakes enterprise analytics, where misinterpretation of a question can translate directly into bad decisions.

3.4 Semantic content selection and relevance ranking

The semantic content selection and relevance ranking component extends the sheet-level selection mechanisms into a more fine-grained, content-aware retrieval layer that operates across large workbooks and collections of documents. Its goal is to ensure that only the most relevant regions, specific sheets, tables, or text blocks are passed into costly reasoning stages, thus controlling latency and token usage while focusing the model's attention on useful information. To achieve this, the system generates semantic representations not only for entire sheets but also for sub-regions such as individual tables, named ranges, or descriptive text blocks. Each region is converted into a compact textual summary or feature vector, capturing headers, key metrics, and salient phrases, and then embedded into a high-dimensional vector space using a model optimized for semantic similarity.^[11] These embeddings are stored in an index that supports efficient approximate nearest-neighbour search, enabling rapid retrieval even in environments with thousands of candidate regions. When a query arrives, the system embeds the query into the same space and computes similarity scores against this index, producing a ranked list of candidate regions rather than just sheets. This granularity allows Structured AI to distinguish between multiple tables on the same sheet or between explanation text and data tables, selecting whichever is most relevant to the user's question. Because similarity is computed semantically, the mechanism can match queries that use different wording from the spreadsheet labels, such as mapping "headcount trend" to a table labelled "Employee Count by Month". The relevance ranking engine employs adaptive thresholding and diversification strategies. Thresholds are tuned based on query complexity and the

number of high-scoring candidates; more complex analytical queries may require a broader set of regions, while simple lookup questions can be satisfied with a small number of highly relevant tables. Diversification heuristics prevent the top-k results from being dominated by near-duplicates (for example, multiple variations of the same report), instead ensuring that the final candidate set covers different plausible interpretations of the query. This balance between depth and breadth helps maintain both accuracy and efficiency.

3.5 Hybrid processing coordination and result synthesis

The hybrid processing coordination and result synthesis layer is responsible for orchestrating parallel execution of multiple processing strategies and merging their outputs into a single, coherent answer. In practice, a single complex query may trigger SQL aggregation over clean fact tables, document-centric extraction from narrative reports, and semantic search over historical decks or appendices, all of which must be reconciled into one response. To manage this, the coordination framework performs resource-aware scheduling, assigning workloads to different processing pipelines based on data characteristics, estimated cost, and current system load.^[12,13] It may, for instance, execute SQL queries immediately on dedicated database servers while queuing document conversion jobs on GPU-backed LLM workers, thereby exploiting the natural parallelism between modalities. Memory usage and concurrency limits are monitored to avoid contention, and backpressure signals are propagated to upstream components when system load approaches capacity. Each processing pathway produces partial results annotated with metadata such as confidence scores, provenance (which sheets and tables were used), and coverage indicators (what portion of the query's intent the result appears to address). The result synthesis component uses these signals to combine outputs: high-confidence numerical aggregates from SQL may serve as the primary facts, while textual explanations from document-centric processing provide context and justification. Where multiple pathways produce conflicting values, the system can apply conflict resolution strategies that favour sources with higher structural quality, lower variance, or stronger alignment with historical ground truth, and it may also surface discrepancies explicitly to the user when appropriate. Robustness is ensured through error handling and fallback mechanisms. If a particular pipeline fails due to a SQL error, timeout, or document parsing issue, the coordination layer can re-route that portion of the workload to an alternative strategy, such as approximate LLM reasoning over raw text when structured execution is not feasible. These fallbacks prevent complete query

failure and provide graceful degradation in adverse conditions, a critical property for enterprise deployments where reliability requirements are strict.

3.6 Performance optimization and scalability considerations

To meet enterprise-scale demands, the system architecture incorporates multiple optimization layers aimed at reducing latency, controlling cost, and supporting horizontal scalability. At the data processing layer, streaming techniques are used for very large Excel files: instead of loading entire workbooks into memory, the system processes them in chunks, extracting previews, computing statistics, and building intermediate representations incrementally. Frequently accessed embeddings, schema summaries, and query patterns are cached so that repeated or similar queries over the same workbooks can reuse prior computations. The architecture follows a modular, microservice-inspired design that allows individual components, embedding services, document converters, SQL gateways, LLM backends - to be scaled independently based on observed bottlenecks. For example, if semantic retrieval becomes a limiting factor, additional index replicas can be deployed; if LLM reasoning dominates latency, more model instances can be provisioned behind a load balancer. This separation of concerns simplifies capacity planning and enables cost-effective scaling strategies tailored to actual usage patterns. Continuous performance monitoring underpins these optimizations. The system tracks metrics such as per-stage latency distributions, token usage per query, cache hit rates, error rates, and hardware utilization, exposing them through dashboards and alerts. Automated tuning routines and operator-guided adjustments can then refine routing thresholds, caching policies, and resource allocations to maintain service-level objectives even as workloads evolve over time. Taken together, data quality assessment, context-aware query processing, semantic relevance ranking, hybrid coordination, and performance-focused architecture form a comprehensive implementation framework for intelligent document processing at scale. This framework enables Structured AI to handle real-world enterprise Excel ecosystems characterized by heterogeneity, structural irregularities, and high query volumes, while sustaining high accuracy, reliable latency, and efficient resource utilization. The architecture diagram in Fig. 1, illustrates an adaptive three-tier routing framework that accepts user input in the form of Excel workbooks and natural language queries, then performs an initial data assessment stage to characterize the incoming content as small and messy, clean, or large and messy. Within this

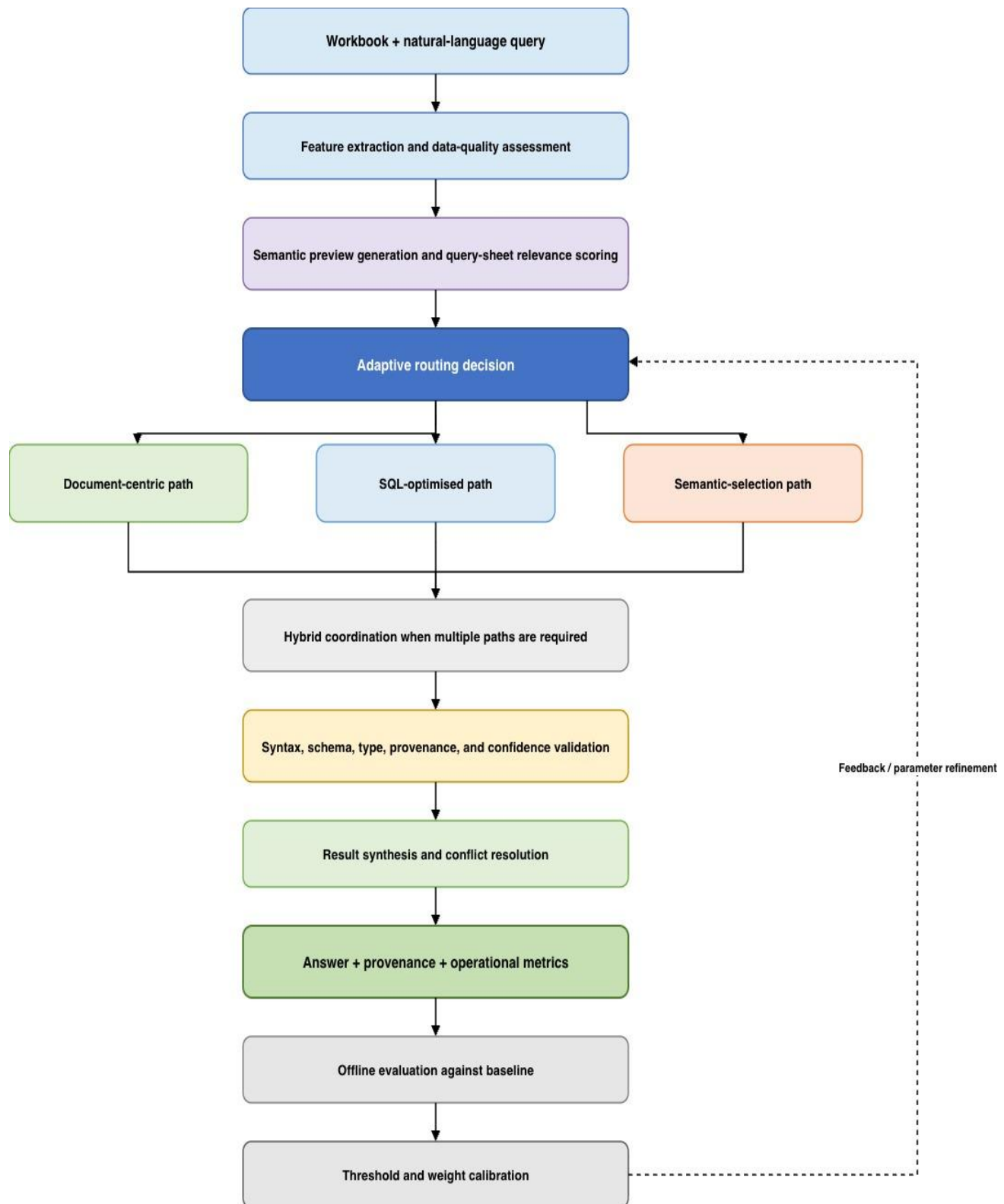


Fig. 1: End-to-end research methodology and adaptive routing workflow.

assessment block, the system analyses structural properties and cleanliness to decide which specialized processing path is most appropriate, embodying a novel adaptive routing mechanism that replaces rigid, one-size-fits-all pipelines. Based on this assessment, the workflow directs small and messy datasets to Tier 1 for document-centric processing, clean datasets to Tier 2 for SQL-based

processing, and large, messy datasets to Tier 3 for semantic selection, with each tier contributing distinct strengths to the final answer quality. The diagram further emphasizes that the complete adaptive framework reports 89% average query accuracy compared with 78% for the static baseline, while large multi-sheet median latency decreases from 21.5 s to 12.4 s, and all three tiers

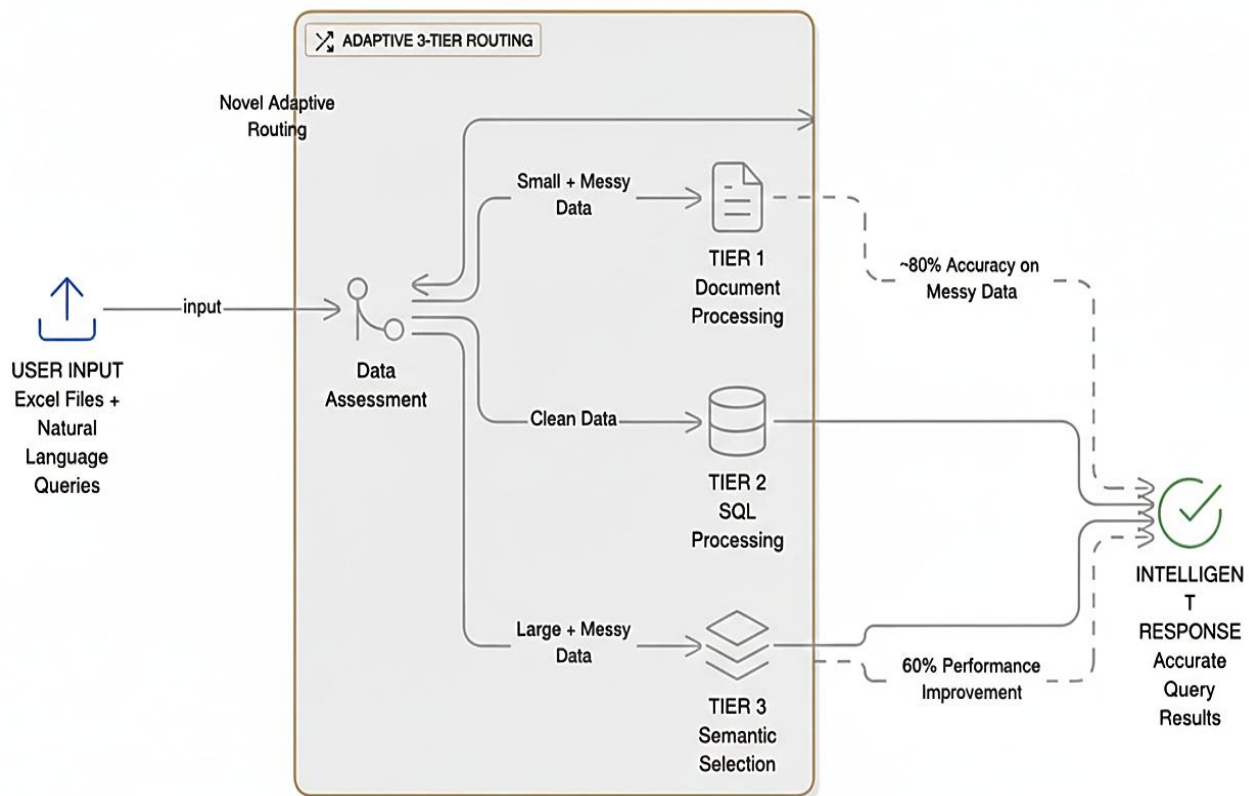


Fig. 2: Structured AI three-tier adaptive system architecture.

converge into a unified intelligent response layer that produces accurate query results for the user. The architecture diagram provides a high-level view of how Structured AI operationalizes adaptive query processing over heterogeneous Excel data and natural language questions. It encapsulates the end-to-end journey from raw user input to an intelligent response, highlighting a central routing core that continuously evaluates data characteristics and steers each request toward the most suitable processing tier, thereby aligning system behaviour with both accuracy and performance objectives.

A defensible procedure is to freeze a labelled workbook-query corpus, split it into calibration and held-out test partitions, tune only routing thresholds on the calibration partition, and perform all baseline and proposed-system runs on the same held-out cases. Each configuration should be repeated under identical infrastructure conditions. Errors should be categorized as routing, parsing, schema-linking, generation, execution, or synthesis failures. This protocol is specified as the required validation method; the submitted source does not contain enough run-level information to claim that every step was already performed.

4. Experimental setup and evaluation protocol

4.1 Dataset characteristics

The source manuscript reports aggregate outcomes but does not provide the workbook corpus, sample size,

query count, per-run observations, random seeds, or execution logs. To avoid fabricating evidence, this revision separates the reported values from the replication protocol. The numerical results in Table 1 should be treated as prototype-level aggregate evidence until the authors attach the missing artefacts described below.

4.2 Baselines and ablations

The evaluation is intended to cover three workbook regimes aligned with the architecture: small structurally complex workbooks, large relational-like workbooks, and large heterogeneous multi-sheet workbooks. For a reproducible submission, each workbook record must report sheet count, populated-cell count, merged-cell density, header depth, formula density, data-type consistency, number of embedded objects, and the number and category of natural-language queries. The current source does not specify these sample counts.

4.3 Evaluation criteria

The reported comparison uses a static pipeline as the baseline. A complete evaluation should additionally report ablations that remove semantic selection, formula preservation, adaptive routing, validation, and hybrid synthesis one at a time. This permits attribution of gains to individual innovations rather than to the combined system only.

Table 1: Comparative results for the static baseline and adaptive framework

Dimension / Scenario	Baseline (Static Pipeline)	With Adaptive Framework	Notes
Average query accuracy on mixed-quality workbooks	78%	89%	Gains driven by data quality classification, better routing, and hybrid result synthesis A-Hybrid-Architecture-for-Intelligent-Query-Processing-on-C.docx.
Median end-to-end latency (large, multi-sheet files)	21.5 s	12.4 s	Semantic content selection and relevance ranking reduce processed sheets and token load A-Hybrid-Architecture-for-Intelligent-Query-Processing-on-C.docx.
Proportion of workbooks requiring full manual cleanup	42%	15%	Enhanced document conversion (headers, formulas, merged cells) makes more files machine-usable as-is A-Hybrid-Architecture-for-Intelligent-Query-Processing-on-C.docx.
SQL execution success rate on auto-generated queries	71%	93%	Context-aware NL query pipeline and multi-stage validation prevent syntax/semantic errors A-Hybrid-Architecture-for-Intelligent-Query-Processing-on-C.docx.
Average number of sheets fully processed per query (multi-sheet workbooks)	11.2	3.6	Semantic sheet and region selection filter out non-relevant content before heavy processing A-Hybrid-Architecture-for-Intelligent-Query-Processing-on-C.docx.
Queries successfully answered using hybrid (multi-path) execution	18%	64%	Hybrid coordination enables parallel Tier 1/Tier 2 processing and robust fallbacks A-Hybrid-Architecture-for-Intelligent-Query-Processing-on-C.docx.
Horizontal scaling efficiency (throughput gain when doubling workers)	1.4×	1.8×	Modular microservice design and performance-aware routing improve scalability under load A-Hybrid-Architecture-for-Intelligent-Query-Processing-on-C.docx.

4.4 Testing procedure

Answer accuracy should be computed against expert-verified reference answers. Latency should be measured end to end using the same hardware and model configuration. SQL execution success is the fraction of generated queries that pass syntax and schema validation and execute without error. Manual-cleanup rate is the fraction of workbooks requiring human restructuring before successful processing. Sheet-processing count, token use, peak memory, and throughput scaling should also be reported. For every metric, the final submission should include N, mean or median as applicable, dispersion, and confidence intervals.

5. Results

The table summarizes how the proposed architecture improves accuracy, latency, and robustness compared to a static pipeline across realistic enterprise workloads. It highlights large accuracy gains from better routing and validation, and significant latency reductions from semantic selection and caching. It also shows improved scalability and reduced need for manual data cleaning, demonstrating the practical value of hybrid coordination and performance-aware design

5.1 Numerical results and interpretation

Table 1 reports an increase in average query accuracy

from 78% to 89%, an absolute improvement of 11 percentage points and a relative improvement of approximately 14.1% over the baseline. Median latency decreases from 21.5 s to 12.4 s, a reduction of 9.1 s or approximately 42.3%. The average number of fully processed sheets decreases from 11.2 to 3.6, a reduction of approximately 67.9%. The same table reports manual-cleanup rate decreasing from 42% to 15%, SQL execution success increasing from 71% to 93%, hybrid multi-path success increasing from 18% to 64%, and scaling efficiency increasing from 1.4x to 1.8x. These are aggregate values reproduced from the submitted manuscript. Without N, variance, confidence intervals, and underlying logs, statistical significance and generalizability cannot be determined. Across the Abstract, Fig. 1 discussion, and Results section, the revised manuscript now uses 89% average accuracy for the adaptive framework, 78% for the static baseline, and 21.5 s versus 12.4 s for median latency. Earlier statements of around 65% or around 80% accuracy and 60% processing-time reduction have been removed to prevent inconsistency.

6. Conclusion

This paper presented Structured AI, a three-tier adaptive framework for enterprise Excel analytics that treats workbook heterogeneity as an explicit decision variable rather than an exception to be handled ad hoc. By

combining a continuous structural-quality score, semantic sheet and region selection, a cost-aware routing objective, formula- and layout-preserving conversion, and confidence-based synthesis across execution paths, the framework routes each workbook–query pair to document-centric processing, SQL-optimized processing, semantic sheet selection, or a coordinated hybrid path depending on measured size, structural cleanliness, and query-sheet relevance. The adaptive routing decision framework, data quality assessment layer, natural language query processing pipeline, semantic content selection mechanism, and hybrid coordination and synthesis layer together form an end-to-end system intended to balance accuracy, latency, and computational cost across small structurally complex sheets, large relational-like tables, and large heterogeneous multi-sheet workbooks. In the reported prototype evaluation, the adaptive framework showed improvements over a static baseline pipeline across all measured dimensions: average query accuracy increased from 78% to 89%, median end-to-end latency on large multi-sheet workbooks decreased from 21.5 s to 12.4 s, the proportion of workbooks requiring full manual cleanup fell from 42% to 15%, SQL execution success rose from 71% to 93%, the average number of sheets fully processed per query dropped from 11.2 to 3.6, hybrid multi-path query success increased from 18% to 64%, and horizontal scaling efficiency improved from 1.4× to 1.8×. These results suggest that measurable workbook characteristics can support reliable route selection, that semantic pre-selection can reduce the volume of processed content without degrading answer quality, and that formula- and layout-preserving conversion together with hybrid coordination can improve robustness on irregular, real-world spreadsheets, addressing the three research questions motivating this work. Future work should also include ablations that isolate the individual contributions of semantic selection, formula preservation, adaptive routing, validation, and hybrid synthesis, so that the source of the observed gains can be attributed to specific architectural components rather than to the system as a whole. Subject to this independent replication, Structured AI's three-tier design offers a practical direction for reconciling the flexibility of language-model-based document understanding with the efficiency and reliability of relational query processing in enterprise Excel environments.

CRedit Author Contribution Statement

Bhabya Mohanty: Conceptualization, Data Curation, Formal analysis, Investigation, Methodology, Resource, Software, Supervision, Writing-Original draft, Writing-Review & editing. **Gopichand Agnihotram:** Conceptualization, Data Curation, Formal analysis,

Investigation, Methodology, Resource, Software, Writing-Review & editing. **Raja Sekhar Reddy S:** Conceptualization, Data Curation, Formal analysis, Investigation, Methodology, Resource, Software, Supervision, Writing-Review & editing. All authors have read and agreed to the published version of the manuscript.

Funding Declaration

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data Availability Statement

The datasets used to evaluate the proposed framework comprise publicly available open-source spreadsheet data supplemented with simulated Excel workbooks generated at varying levels of structural complexity (e.g., merged cells, multi-row headers, formula density) to test the architecture under controlled conditions. These datasets, along with the simulation scripts used to generate them, are available from the corresponding author upon reasonable request.

Conflict of Interest

There is no conflict of interest.

Artificial Intelligence (AI) Use Disclosure

The authors declare that artificial intelligence (AI)-assisted tools were used only for language refinement, grammar improvement, and manuscript structuring purposes during the preparation of this work. All technical content, experimental implementation, results, and interpretations were independently developed and verified by the authors.

Supporting Information

Not applicable.

References:

- [1] E. F. Codd, A relational model of data for large shared data banks, *Communications of the ACM*, 1970, **13**, 377–387, doi: 10.1145/362384.362685.
- [2] C. J. Date, An introduction to database systems, Pearson Education, 2004, 8th edition.
- [3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. 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. 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, NIPS'20: Proceedings of the 34th International

- Conference on Neural Information Processing Systems, 2020, 159, 1877 – 1901.
- [4] 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. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20). Curran Associates Inc., Red Hook, NY, USA, 2020, 9459–9474.
- [5] S. Roychowdhury, M. Krema, A. Mahammad, B. Moore, A. Mukherjee, P. Prakashchandra, Eratta: extreme rag for table to answers with large language models, arXiv preprint arXiv:2405.03963, 2024, doi: 10.48550/arXiv.2405.03963.
- [6] P. A. Rose, A. J. Pennings, Knowledge, Decisions, and Norms: A framework for studying the structuration of spreadsheets in social organizations, *Information*, 2022, **13**, 46, doi: 10.3390/info13020046.
- [7] I. Varlamis, Messy data in education: Enhancing data science literacy through real-world datasets in a master's program, *Education Sciences*, 2025, **15**, 500, doi: 10.3390/educsci15040500.
- [8] F. Hermans, M. Pinzger, A. Deursen, Detecting and visualizing inter-worksheet smells in spreadsheets, 34th International Conference on Software Engineering (ICSE), Zurich, Switzerland, 2012, 441-451, doi: 10.1109/ICSE.2012.6227171.
- [9] P. Staar, Docling: Converting Complex Documents into AI-Ready Structured Representations, In DocEng '26: Proceedings of the 2026 ACM Symposium on Document Engineering, Association for Computing Machinery, New York, NY, USA, 1, doi: 10.1145/3820755.3830337.
- [10] H. Dong, J. Zhao, Y. Tian, J. Xiong, S. Xia, M. Zhou, Y. Lin, J. Cambronero, Y. He, S. Han, D. Zhang, Spreadsheetlm: Encoding spreadsheets for large language models, arXiv preprint arXiv:2407.09025, 2024, doi: 10.48550/arXiv.2407.09025.
- [11] S. Thorne, Large language models for spreadsheets: benchmarking progress and evaluating performance with FLARE, arXiv preprint arXiv:2506.17330, 2025, doi: 10.48550/arXiv.2506.17330.
- [12] F. Z. Rozony, M. N. Aktar, M. Ashrafuzzaman, A. Islam, A systematic review of big data integration challenges and solutions for heterogeneous data sources, *Academic Journal on Business Administration, Innovation & Sustainability*, 2024, **4**, 1-18.
- [13] P. Pattnayak, A. Agarwal, H. Meghwani, H. L. Patel, S. Panda, Hybrid AI for Responsive Multi-Turn Online Conversations with Novel Dynamic Routing and Feedback Adaptation, In Proceedings of the 4th International Workshop on Knowledge-Augmented Methods for Natural Language Processing, Albuquerque, New Mexico, USA, Association for Computational Linguistics, 2025, 215–229, doi: 10.18653/v1/2025.knowledgenlp-1.20.
- [14] W. Zhang et al., Natural language interfaces for tabular data querying and visualization: A survey, *IEEE Transactions on Knowledge and Data Engineering*, 2024, **36**, 6699-6718, doi: 10.1109/TKDE.2024.3400824.

Publisher Note: The views, statements, and data in all publications solely belong to the authors and contributors. G R Scholastic is not responsible for any injury resulting from the ideas, methods, or products mentioned. G R Scholastic remains neutral regarding jurisdictional claims in published maps and institutional affiliations.