Relational Deep Dive: Error-Aware Queries Over Unstructured Data

Daren Chao
University of Toronto
Toronto, Canada
drchao@cs.toronto.edu

Kaiwen Chen
University of Toronto
Toronto, Canada
kckevin.chen@mail.utoronto.ca

Naiqing Guan
University of Toronto
Toronto, Canada
naiqing.guan@mail.utoronto.ca

Nick Koudas
University of Toronto
Toronto, Canada
koudas@cs.toronto.edu

ABSTRACT

Unstructured data is pervasive, but analytical queries demand structured representations, creating a significant extraction challenge. Existing methods like RAG lack schema awareness and struggle with cross-document alignment, leading to high error rates. We propose ReDD (Relational Deep Dive), a framework that dynamically discovers query-specific schemas, populates relational tables, and ensures error-aware extraction with provable guarantees. ReDD features a two-stage pipeline: (1) Iterative Schema Discovery (ISD) identifies minimal, joinable schemas tailored to each query, and (2) Tabular Data Population (TDP) extracts and corrects data using lightweight classifiers trained on LLM hidden states. A main contribution of ReDD is SCAPE, a statistically calibrated method for error detection with coverage guarantees, and SCAPE-Hyb, a hybrid approach that optimizes the trade-off between accuracy and human correction costs. Experiments across diverse datasets demonstrate ReDD's effectiveness, reducing data extraction errors from up to 30% to below 1% while maintaining high schema completeness (100% recall) and precision. ReDD's modular design enables fine-grained control over accuracy-cost trade-offs, making it a robust solution for high-stakes analytical queries over unstructured corpora.

PVLDB Reference Format:
Daren Chao, Kaiwen Chen, Naiqing Guan, and Nick Koudas. Relational Deep Dive: Error-Aware Queries Over Unstructured Data. PVLDB, 19(8): 1840 - 1853, 2026.
doi:10.14778/3811243.3811256

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/daren996/ReDD.

1 INTRODUCTION

In many applications, including healthcare, finance, and engineering, the vast majority of the data produced is unstructured (in the form of reports, surveys, clinical trials, etc). Data analytics applications, however, in these domains require the data to be in a structured format. Consider, for example, analytical queries on the results of clinical trials for drug side effects or related queries in a financial domain to report on the top reasons identified for missed earnings of public companies in certain sectors. These queries may involve entity alignment, multi-hop reasoning, and statistical aggregation—tasks that are particularly difficult in the absence of structured representations. Structured data is also mandated by the strict accuracy requirements in these applications. As a result, unstructured data has to be processed to yield structured information for further downstream analytical processing.

Motivating Example. Consider the example in Figure 1, which depicts a natural language query asking for the average treatment cost by disease for hospitals in New York in January 2024, alongside a collection of documents, depicted as document chunks for illustration purposes. These document chunks vary in focus: some describe treatment details (e.g., costs and diseases) like D1, others provide hospital metadata (e.g., location) like D2, and some contain patient profiles, irrelevant to the query, like D3. Answering this query requires aligning hospital names across document chunks and aggregating costs, a process complicated by the heterogeneous and fragmented nature of the data. Moreover, the absence of schemas or explicit semantic information (e.g., entity relationships) further hinders query answering. These challenges extend across domains-beyond medical reports to financial filings, legal documents, and more. Answering query Q in Figure 1 directly from the data in the document collection is challenging.

Current methods, such as retrieval-augmented generation (RAG) [22], based on large language models (LLMs), attempt to answer queries over unstructured data by retrieving top-ranked documents based on query similarity and generating a response conditioned on a limited context. This design makes RAG optimized for precision, but offers limited control over recall—a property that is often more critical in database-style queries, such as those involving statistical aggregation and other analytical tasks, across documents as in Figure 1. Moreover, RAG lacks schema awareness and struggles with cross-document entity alignment, such as linking hospital names across documents [32, 42]. State-of-the-art approaches for text-to-SQL [12] or traditional information extraction techniques [33], rely on predefined schemas or explicit semantic information, which is generally unavailable or undefined in unstructured corpora.

Proceedings of the VLDB Endowment, Vol. 19, No. 8 ISSN 2150-8097.
doi:10.14778/3811243.3811256

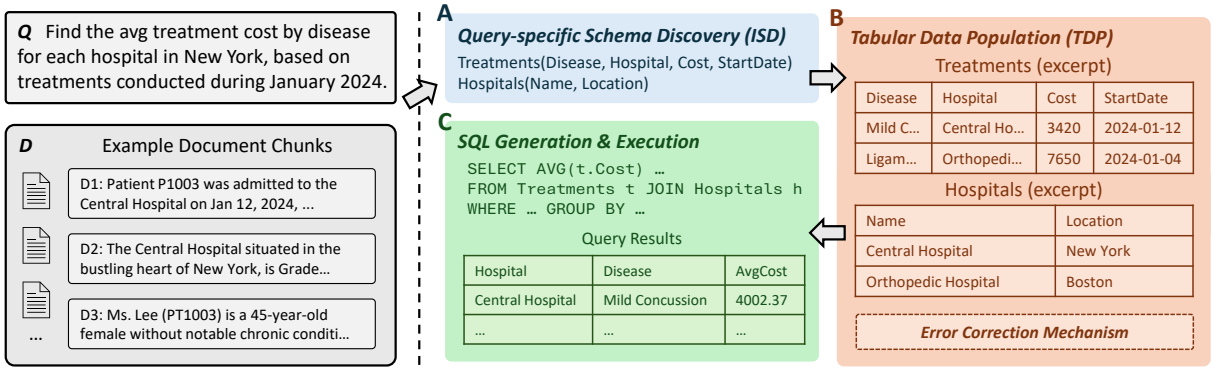


Figure 1: Overview of the query processing pipeline in REDD. Left: a natural-language query and unstructured document chunks. Right: REDD's core workflow: (A) schema discovery, (B) data population, and (C) SQL generation/execution (not the focus here). Component (B) includes error correction for controllable accuracy.

Recent agentic frameworks for unstructured data analysis, including DeepResearch [14, 19, 37], DeepSearch [58], and LangChain [34], enable query-driven document scanning and analysis. However, their correctness is typically governed by heuristic or empirically tuned mechanisms, without formal error detection or statistical guarantees. In contrast, our goal is to extract query-specific structured data and support error-aware analytical queries with explicit accuracy control.

Relational Deep Dive (REDD) Overview. In this paper, we present REDD, (pronounced 'ready') our proposed **Re**lational **D**eep **D**ive framework. We aim for fine-grained accuracy control for the specified query, bridging the gap between raw documents and structured query execution. REDD integrates schema discovery (if applicable), structured data population, with error guarantees, and result synthesis in a cohesive pipeline *driven by the input query*. REDD focuses on the backend stages illustrated in Figure 1, namely query-driven schema discovery and structured data population with error guarantees. Figure 1 provides an illustrative decomposition that highlights where REDD fits; downstream SQL execution is handled by existing database engines, and text-to-SQL translation can be supplied by any state-of-the-art approach [12, 23, 44, 54].

The pipeline begins with Iterative Schema Discovery (ISD), which processes the collection of documents at a suitable granularity (referred to as document chunks), iteratively refines a candidate set of tables and attributes, given a query Q, as more document chunks are processed. This procedure identifies the minimal schema required to answer Q, and uncovers latent semantic structures such as shared entities (join keys) across document chunks. For example, in Figure 1(A), ISD discovers two tables—Treatments and Hospitals—along with attributes both directly relevant to the query (e.g., treatment cost and disease) and indirectly necessary for alignment (e.g., hospital name), even if not mentioned in the query itself. Once the schema is in place, REDD proceeds to Tabular Data Population (TDP). Each document chunk is parsed and converted into one or more rows across the discovered tables, depending on its content and relevance to the query Q. As exemplified in Figure 1(B), chunk D1 populates the first row of the Treatments table, while D2 contributes a row to Hospitals. In contrast, D3 is irrelevant to the query and therefore does not populate any table. However, due to the ambiguity of natural language and the inherent variability of LLM outputs, extraction errors remain common—especially when attributes are implicit, phrased inconsistently, or missing altogether. To combat these challenges, we introduce a downstream correction module that proactively detects and corrects extraction errors, and supports human-in-the-loop intervention when required.

To enable this functionality, REDD supports controlled accuracy. Rather than treating the LLM as an infallible oracle, REDD quantifies extraction uncertainty and selectively corrects low-confidence outputs through lightweight classifiers trained on LLM hidden states and ensemble strategies. Specifically, REDD exploits the hidden representations (i.e., intermediate layer states) of LLMs, which encode rich contextual signals. These hidden representations are used as features for a suite of classifiers that predict extraction correctness—e.g., incorrect attribute values, wrongly assigned tables, or missing entries. These classifiers underpin a set of correction strategies, including ensemble agreement checks, conformal prediction, and fallback re-extraction. When classifier disagreement or uncertainty is high, we either reprocess the document chunk or flag it for human-in-the-loop review. This modular design enables fine-grained control over the accuracy-efficiency trade-off at query time as we detail in §5.

Finally, to remain robust when a single logical record is fragmented across multiple chunks, REDD introduces a consolidation mechanism: SCHEMAMERGE for aggregating schema evidence and ROWMERGE for document-level row construction, as detailed in §6.

In our experiments across multiple domains, we observe that direct data extraction by LLMs can yield error rates (measured as the proportion of incorrectly populated rows) of up to 30% on certain challenging datasets. With REDD's correction techniques, these error rates consistently drop below 1%, without requiring schema supervision or domain-specific rule engineering. These gains underscore the effectiveness of our end-to-end backend pipeline (schema discovery and data population) in achieving not only high accuracy but also controllable extraction quality, even over large-scale unstructured document collections. This paper makes the following contributions:

- We present REDD, a query-specific framework for executing structured queries over unstructured text. REDD dynamically discovers schemas, populates relational tables, and detects/corrects

extraction errors, enabling accurate query answering without predefined schemas or annotations.
- We develop a two-stage schema discovery pipeline that constructs minimal, joinable, and query-complete tables. Experiments show that this approach consistently outperforms single-phase alternatives in schema accuracy and coverage.
- We introduce SCAPE (Spatial Conformal Activation Partitioning for Errors), a statistically calibrated error detection method that guarantees error coverage (Theorem 5.1) while asymptotically minimizing human correction cost (Theorem 5.2). We further propose SCAPE-Hyb, which incorporates conflict-aware signals into SCAPE, preserving calibration guarantees while enabling flexible recall–cost trade-offs (Theorem 5.3).
- Extensive experiments on real-world datasets show that ReDD scales to large document collections and reduces query error rates from up to 30% to below 1%, demonstrating its effectiveness in producing query-ready structured data with provable guarantees.

2 THE REDD FRAMEWORK

ReDD proceeds in two stages and transforms unstructured textual data into query-ready structured tables. The system accepts as input a natural language query q and a collection of unstructured documents. These documents are segmented into semantically coherent *chunks*, denoted as $D = \{d_1, d_2, \ldots, d_n\}$, where each chunk d_i is a contiguous span of text bounded by semantic discontinuities (e.g., paragraph breaks). We treat each chunk as the minimal semantic unit of processing. Depending on its content, a chunk may yield one or multiple rows[1] distributed across one or more (initially unknown) tables. The goal of ReDD is to (i) discover the latent schema S_D^q required to answer q, and (ii) populate that schema with tuples extracted from D, supporting accurate and error-aware query execution.

To answer complex analytical queries over unstructured text, it is often necessary to recover latent structured representations—namely, relational tables and their schemas—that are not explicitly present in the input. In some cases (as exemplified by systems such as Galois [43] and Palimpzest [30]), the schema may be known or provided (see §8 for details). Although ReDD handles this case naturally, we take a more general and principled approach by enabling automated schema discovery. This process is complicated by several factors:

- The number of tables, their schemas, and the mapping from document chunks to tables are unknown a priori.
- The query q may require aggregating information distributed across multiple latent tables to produce a complete answer.

This setting reflects real-world scenarios where no external schema, entity linking, or domain-specific annotations are available, and the query-specific structured tables must be discovered dynamically from text.

ReDD addresses these challenges through its two-stage pipeline: Iterative Schema Discovery (ISD) and Tabular Data Population (TDP), which includes an error correction module (see §4-5). The entire pipeline is driven by the input query q, while also mining latent semantics across chunks to construct accurate, structured outputs. We first detail the Iterative Schema Discovery (ISD) phase (§3), which produces the query-specific schema S_D^q. We then describe the Tabular Data Population (TDP) phase (§4-5), which populates the schema with data extracted from documents.

3 ITERATIVE SCHEMA DISCOVERY

The goal of Iterative Schema Discovery (ISD) is to derive a relational schema S_D^q over a collection of document chunks D that contains the entities and relationships required to answer the query q. ISD proceeds in two phases: Phase I induces a general (query-agnostic) schema S_D^{gen}, and Phase II adapts it into a minimal, query-complete schema S_D^q. We defer prompt templates and implementation details to our technical report [11].

Phase I: General Schema Discovery. ISD scans chunks sequentially and incrementally revises an evolving schema state organized as relational tables (entities/relationships) with lightweight annotations (e.g., short descriptions and representative evidence). Schema updates are implemented by a prompt-based function $\mathcal{S}_G$ and follow the iterative update rule in Equation (1):

$$S_k^{\text{gen}} = \mathcal{S}_G\left(S_{k-1}^{\text{gen}}, d_k\right), \quad \text{for } k = 1..n \tag{1}$$

The resulting S_D^{gen} serves as a comprehensive abstraction over the corpus and as the structural basis for query-specific refinement.

Phase II: Query-Specific Schema Discovery. Given S_D^{gen}, Phase II constructs S_D^q by removing irrelevant tables/attributes and adding any missing query-critical elements (including implicit join keys). This phase uses a second prompt-based function $\mathcal{S}_Q$ and follows Equation (2):

$$S_k^q = \mathcal{S}_Q\left(S_{k-1}^q, d_k, q, S_D^{\text{gen}}\right), \quad \text{for } k = 1..n \tag{2}$$

We empirically validate the effectiveness of this two-phase design in §7.2.

Repair Step. As an optional safeguard, we apply a *fallback repair* after Phase II to verify that S_D^q suffices to answer q, and to trigger additional iterations when incompleteness is detected (details in [11]).

Performance Considerations. Our schema discovery pipeline consists of two sequential passes over the document collection: Phase I constructs an initial schema, and Phase II refines it to the query intent. While sampling or selective processing could reduce computation, such optimizations are orthogonal to this work. Our focus is on establishing whether accurate, query-specific schema

[1]For brevity, we assume that each chunk yields one row for illustration. In practice, ReDD naturally supports both *one-to-many* and *many-to-one* mappings without changing the downstream modules. **One-to-many** arises when a chunk contains multiple relevant facts and should populate multiple rows (possibly across multiple tables); this is naturally handled by emitting multiple row candidates from a single chunk during TDP (§4). **Many-to-one** commonly occurs under real-world *document revisions* (updates/deletes) and redundant evidence, where multiple chunks refer to the same entity key and should be consolidated into a single row; such conflicts can be resolved via simple overwrite/priority rules (e.g., latest-wins when metadata is available) or human verification when needed. We empirically validate these settings and their trade-offs in §7.4.1 (one-to-many and many-to-one) and §7.3.1 (overall data population results). Finally, a distinct source of many-to-one mappings arises from *document chunking*, where a single logical record in the original document may be split across multiple chunks, each containing only partial attribute information, even when the document-to-record relationship is conceptually one-to-one. We address this case separately in §6 by introducing a Map Reduce–style consolidation mechanism: **SchemaMerge** for document-level schema discovery and **RowMerge** for document-level data population from chunk-level fragments.

discovery and extraction are feasible independent of hard computational constraints (e.g., token usage [30, 43]). Answering this question first, instigates future research directions or engineering optimizations to derive performance efficiency. A detailed exploration of efficiency-oriented enhancements is left to future work and falls outside the scope of this paper.

4 TABULAR DATA POPULATION

Given a query-specific schema S_D^q, the Tabular Data Population (TDP) stage inserts tuples into the schema by extracting structured data from document chunks. Each chunk d_k is independently processed to yield structured rows aligned with the schema semantics. The TDP stage follows a fixed, two-step pipeline, in which lightweight, LLM-guided prompt functions extract and format relevant data[2].

- Table Resolver $\mathcal{X}_T$. For every chunk d_k, the resolver selects the most semantically compatible table $T_k \in S_D^q$:

$$T_k = \mathcal{X}_T\left(d_k, S_D^q\right), \quad \text{for } k = 1..n \tag{3}$$

  where T_k denotes the identifier of the selected target table.
- Attribute Extractor $\mathcal{X}_A$. Given the pair (d_k, T_k), the extractor iteratively fills each attribute $a_i \in T_k$:

$$v_{k,i} = \mathcal{X}_A\left(d_k, a_i, T_k\right), \quad \text{for } k = 1..n,\ a_i \in T_k \tag{4}$$

  where $v_{k,i}$ denotes the value of attribute a_i extracted from chunk d_k. The resulting tuple $(v_{k,1}, \ldots, v_{k,m})$ forms a single row to be inserted into table T_k.

This procedure extracts, for each chunk, a semantically aligned table and a complete attribute-value tuple, yielding one structured row per chunk as the final output of data population. It also generalizes to one-to-many settings: the table resolver can return multiple candidate tables per chunk, and the attribute extractor can extract multiple tuples per table accordingly.

Due to the inherent ambiguity of natural language and the stochastic behavior of LLM outputs, the chunk-wise extraction of attribute values can lead to extraction inconsistencies and errors, such as incorrect or incomplete attribute values, or even mis-assigned rows. Relying solely on an LLM in TDP during table population yields error rates (measured as the proportion of incorrectly populated rows) of up to 30% on challenging datasets (see §7.3). The TDP stage, makes local decisions: it processes each document chunk during data population, one at a time, mapping it to one or more tables and extracting tuple(s). Lacking a holistic view, TDP cannot revise earlier decisions based on broader context, making tuple extraction inherently error-prone. This limitation instigates the introduction of additional mechanisms to mitigate such errors.

4.1 Error Detection Approaches

While Tabular Data Population (TDP) extracts tuples and populates query table(s) whose schema is discovered for a query q during ISD, it may introduce extraction errors due to the inherent uncertainty of LLM outputs. In high-stakes settings, even a small number of erroneous entries may lead to incorrect conclusions in downstream analyses. REDD operation is geared towards a specific query q and it includes mechanisms to identify and correct data extraction errors, ensuring that each relational table entry is grounded in the source document. Central to our approach towards error-free data extraction is assessing the trustworthiness of each extracted value, utilizing small, open-weight LLMs.[3] Their compact size allows for local deployment, while being open-weight enables the development of specialized, tunable algorithms around them. Based on this assessment, we decide when to trigger corrective actions and/or abstain from extracting the value and trigger a human review to assist our algorithms with extraction. Thus, REDD incorporates a human-in-the-loop as a first-class primitive.

Before TDP commences, we construct a small labeled dataset, denoted as $\mathcal{D}_{\text{cls}}$, which serves as training data for lightweight classifiers that predict the correctness of extracted values during TDP. Each entry in $\mathcal{D}_{\text{cls}}$ is generated by applying the LLM prompts and using the same LLM model (denoted as M^{TDP}) as in the TDP stage (as defined in Equations (3)-(4)) to a limited set of document chunks, producing candidate table rows. The ground-truth labels for these entries can be obtained through one of the following two approaches:

(1) **Human-Verified Labeling**: A user manually verifies each extracted value against the source text, annotating whether it is correct or incorrect.
(2) **LLM-Committee-Based Labeling**: A committee of powerful LLMs (e.g., OpenAI-o3 [39] and Claude-4 [3]) independently extracts and evaluates the same tuples. The correctness of a candidate extraction is determined by the consensus of all committee members: if the initial LLM's output disagrees with the committee's assessment, the committee's label is taken as the ground truth, and the extraction is marked as incorrect[4].

Formally, let $\mathcal{M}=\{M_1, \ldots, M_K\}$ denote the committee of K powerful LLMs. For a candidate tuple t extracted by the initial LLM M^{TDP}, the label y_t is assigned as:

$$y_t = \begin{cases} 1 & \text{if } t \text{ matches majority of } \mathcal{M} \text{ or human confirms,} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

This hybrid approach ensures high-quality labeled data while minimizing reliance on manual annotation. The required size of $\mathcal{D}_{\text{cls}}$ is small, as we demonstrate in §7.3 by varying its size and measuring its impact on extraction accuracy. These classifiers (§4.2) enable error detection during the extraction process.

Following error detection, REDD applies error resolution measures, such as committee-based inference using more powerful LLMs, or escalation to human verification for low-confidence cases. Such error resolution methods incur additional cost (e.g., monetary, time burden, etc), and need to be minimized. We refer to this overhead as *human correction cost* or *cost* interchangeably. This

[2] Concrete prompt templates and implementation details are provided in the accompanying technical report [11].

[3] A key assumption of our framework is that the underlying Large Language Model (LLM) is *inspectable*. Our error detection method relies on access to the model's internal signals, in particular token-level log probabilities and (more importantly) hidden representations, which we use to train lightweight classifiers for predicting extraction correctness. Consequently, REDD requires the use of open-weight models (e.g., Llama [49], Mistral [17], Qwen [47]) or deployments that expose these internal states. Closed-source models accessed via black-box APIs, which typically do not expose hidden representations (and often do not expose token-level probabilities), are currently incompatible with our correction pipeline.

[4] To address potential labeling biases, the committee uses diverse, high-capability LLMs from different model families: this reduces dependence on M^{TDP}, avoiding a self-reinforcing loop and mitigating the risk of correlated errors [3, 39, 47].

cost is proportional to the number of entries flagged as erroneous, i.e., those with predicted error label $\hat{y}$=1. Notice that in case error detection instigates *false positives* ($\hat{y}$=1 but the true label y=0), this adds extra human correction cost (such as human validation of already correct extractions). Thus, although correct error detection is important (to minimize false negatives), reducing false positives is a major design requirement as well.

Our approach deploys a collection of lightweight classifiers utilizing the hidden states of the LLM (M^{TDP}) to quantify the correctness of each token output. We will start by introducing notation and two baseline methods. The first, MV, utilizes majority voting over the binary outputs of individual classifiers to quantify token-level correctness. The second, CF, is a conservative refinement of MV designed to identify strong disagreement among classifier predictions, thereby reducing false positives. We then introduce our main proposals SCAPE and SCAPE-Hyb in §5.

4.2 Error Detection via Latent Representations

A cornerstone of our approach are lightweight binary classifiers that determines whether each extracted attribute value is correct, using the LLM's own hidden representations. LLM internal states encode rich signals about output truthfulness and correctness [40]. ReDD leverages this property to identify extraction errors, building on prior work demonstrating that hidden states act as high-capacity features strongly correlated with factuality and correctness, even when individual dimensions are not directly interpretable [5, 8, 10, 20].

Leveraging LLM Hidden States. As described in §4, for each document chunk d_k, the TDP stage first assigns a target table T_k via the table resolver $\mathcal{X}_T$ (Equation (3)), and then extracts attribute values $v_{k,i}$ using the attribute extractor $\mathcal{X}_A$ (Equation (4)). During these steps, we have full access to the hidden states of the LLM M^{TDP}. All LLM outputs (including the assigned table name T_k and each extracted attribute value $v_{k,i}$) are generated as sequences of output tokens. Let $\mathbf{w}_{k,\text{table}} = (w^1_{k,\text{table}}, \ldots, w^m_{k,\text{table}})$ be the token sequence corresponding to the assigned table name T_k, and let $\mathbf{w}_{k,\text{attr-}i} = (w^1_{k,\text{attr-}i}, \ldots, w^{m_i}_{k,\text{attr-}i})$ be the token sequence corresponding to the extracted value $v_{k,i}$ for attribute a_i (the i-th attribute) in T_k. Let $h^{(l)}(w)$ denote the hidden state at layer l corresponding to a token w. To obtain a compact representation of the LLM extraction outputs (for both table names and attribute values), we apply mean-max pooling across the token-level hidden states at each layer, then concatenate the pooled vectors:

$$\begin{aligned} h^{(l)}_{k,\text{table}} &= \texttt{concat}\left(\operatorname*{mean}_j h^{(l)}(w^j_{k,\text{table}}), \max_j h^{(l)}(w^j_{k,\text{table}})\right), \\ h^{(l)}_{k,\text{attr-}i} &= \texttt{concat}\left(\operatorname*{mean}_j h^{(l)}(w^j_{k,\text{attr-}i}), \max_j h^{(l)}(w^j_{k,\text{attr-}i})\right), \end{aligned} \tag{6}$$

where $h^{(l)}_{k,\text{table}} \in \mathbb{R}^{2d}$ and $h^{(l)}_{k,\text{attr-}i} \in \mathbb{R}^{2d}$ denote the layer-l hidden representations for table name T_k and attribute value $v_{k,i}$, respectively. Here, d is the size of the hidden state of the underlying LLM M^{TDP}. Since both vectors are computed through identical mean-max pooling operations, share the same dimensionality, and follow the same procedure to train the per-layer classifiers (introduced in §4.3), we adopt a unified notation for simplicity. Specifically, we denote each concatenated hidden representation as $h^{(l)}_{k,\circ}$, where $\circ$ stands for either extracted table names or attribute values. This simplification relaxes notation without loss of generality. In addition, we use $y_{k,\circ}$ to refer to the ground truth label of each extraction, indicating whether the extracted item (table or attribute value) is erroneous or not. A value of $y_{k,\circ}$=1 denotes an error, while $y_{k,\circ}$=0 indicates correctness.

Let $\mathcal{L}$ denote the set of LLM layers from which hidden states are extracted. This set may include all layers or a selected subset[5], as recent studies have shown that certain layers encode richer and more informative signals than others [16, 35]. The aggregated hidden representations obtained using Equation (6) are subsequently used as input features for binary classifiers that predict whether the corresponding table assignment or extracted attribute value is likely to be incorrect. These hidden states capture rich contextual and semantic cues from both the document and the query, making them an informative signal for spotting inconsistencies or mistakes in the LLM's own outputs [10, 20, 40]. The classifiers are trained using $\mathcal{D}_{\text{cls}}$.

4.3 Voting Based Methods: MV and CF

To detect extraction errors from TDP, we begin with a straightforward but effective baseline: performing majority voting on predictions from classifiers trained on hidden states from each layer.

Layer-wise Error Classifiers. For each LLM layer $l \in \mathcal{L}$, we train a lightweight binary classifier[6] $f^{(l)}: \mathbb{R}^{2d} \to [0, 1]$ to predict if the extraction associated with hidden state $h^{(l)}_{k,\circ}$ (as in §4.2) is erroneous:

$$\pi^{(l)}_{k,\circ} = f^{(l)}\left(h^{(l)}_{k,\circ}\right), \tag{7}$$

where $\pi^{(l)}_{k,\circ}$ is the predicted probability of an error. The classifier per layer is trained independently using binary cross-entropy loss. To convert probabilities into binary decisions, a fixed threshold θ is applied:

$$\hat{y}^{(l)}_{k,\circ} = \mathbf{1}\left[\pi^{(l)}_{k,\circ} > \theta\right]. \tag{8}$$

Here, θ determines the decision boundary of each classifier. Its choice is typically set arbitrarily (e.g., 0.5) and ignores classifier-specific mis-calibrations, as well as classifier dependencies.

Majority Voting (MV). MV aggregates the binary predictions $\hat{y}^{(l)}_{k,\circ}$ ($l \in \mathcal{L}$) via majority voting. The idea is that if most classifiers agree an extraction is wrong, it's likely to be so:

$$\hat{y}^{\text{MV}}_{k,\circ} = \mathbf{1}\left[\sum_{l \in \mathcal{L}} \hat{y}^{(l)}_{k,\circ} > \frac{|\mathcal{L}|}{2}\right]. \tag{9}$$

This simple rule effectively denoises isolated errors from individual classifiers by requiring consensus across layers. However, majority voting offers no explicit control over the false positive and false negative rates, making it hard to balance detection accuracy and correction cost as application demands vary.

Conflict Filtering (CF). While MV relies on agreement, classifier disagreement can also be informative. CF builds on this idea by measuring how much conflict exists among the layer-wise predictions.

[5] When a subset of layers is used to train classifiers for error prediction, the index l refers to the l-th element in $\mathcal{L}$, not the original layer number in the LLM.
[6] Each classifier is a multilayer perceptron (MLP) with a sigmoid output.

Intuitively, if different layers disagree about whether an extraction is erroneous, it likely reflects ambiguity or model hesitation. We define the conflict score κ as the number of classifiers that disagree with the majority vote:

$$\kappa = \sum_{l \in \mathcal{L}} \mathbf{1}\left\{\hat{y}^{(l)}_{k,\circ} \neq \hat{y}^{\mathrm{MV}}_{k,\circ}\right\}, \tag{10}$$

An extraction is flagged as potentially erroneous if the conflict score exceeds a tunable threshold τ^{CF}:

$$\hat{y}^{\mathrm{CF}}_{k,\circ} = \begin{cases} \hat{y}^{\mathrm{MV}}_{k,\circ}, & \text{if } \kappa < \tau^{\mathrm{CF}} \\ 1, & \text{if } \kappa \geq \tau^{\mathrm{CF}} \end{cases}$$

Empirically, a higher κ value signifies a high-conflict case and often corresponds to true errors, making CF an effective strategy for reducing false negatives. In contrast, a small κ value signifies low disagreement and points to more confident decisions. The threshold τ^{CF} controls the sensitivity of the CF strategy: a smaller value flags potential errors even in low-conflict cases, thereby improving recall by capturing more true errors than MV, but at the cost of increased false positives and correction overhead.

Limitations. Both methods lack a principled way to control the trade-off between detection accuracy and correction cost. While CF introduces a tunable conflict threshold τ^{CF}, its impact on accuracy and cost is heuristic, with no calibrated semantics or statistical guarantees. This motivates the development of a more controlled algorithm with formal error-rate guarantees to be introduced next.

5 ENABLING ERROR DETECTION TRADEOFFS

To address the limitations of the previous methods and enhance control over the accuracy of error detection and associated error correction costs, we present two methods, SCAPE and SCAPE-Hyb, to trade off error-detection recall against human correction cost by leveraging the continuous probabilities of layer-wise error classifiers.

- SCAPE (§5.1): is a statistically calibrated method that quantifies prediction uncertainty. Its aim is to reduce false negatives (and thus undetected errors) but may increase false positives in the process and thus increase cost (i.e., imposing extra human labour to check the result of the extraction). It allows adjustment of correction aggressiveness via a coverage parameter α.
- SCAPE-Hyb (§5.2): enhances SCAPE with CF, allowing a more flexible balance between accuracy and human correction cost.

In essence, these algorithms enable a tradeoff between prediction accuracy for erroneous data extractions by the LLM and (human) correction cost, by introducing two key parameters:

- Coverage Threshold α: Higher values reduce false negatives (undetected errors) but may increase false positives, leading to higher correction costs.
- Conflict Weight λ: Controls how strongly the conflict-aware algorithm CF influences correction decisions. A higher λ gives greater weight to CF, encourages more conservative decisions, increasing the chance of flagging potential errors (thus reducing false negatives), but may also lead to more cautious behavior and a rise in false positives.

These techniques enable precise and cost-aware control during data extraction. This is essential in applications where undetected errors are unacceptable.

5.1 SCAPE: Spatial Conformal Activation Partitioning for Errors

The SCAPE framework introduces a novel approach to uncertainty quantification by leveraging a high-dimensional non-confor-mity score space [7, 52], in contrast to prior conformal methods that rely on independent thresholds for binary classifiers [2, 7, 45]. Rather than treating each classifier's decision boundary in isolation, SCAPE constructs a multi-dimensional (spatial) score by aggregating outputs from classifiers trained at different layers[7]. The key idea is to partition this high-dimensional space into adaptive cells centered on calibration samples, and to rank these cells by the empirical ratio of correct to incorrect labels observed among the calibration data they contain.

By selecting regions with low concentrations of incorrect labels, SCAPE produces tighter and more informative uncertainty sets while maintaining guaranteed coverage. This spatial partitioning avoids rigid thresholding or weighted aggregation, instead exploiting the geometric structure of the multi-dimensional score space to better separate true from false predictions. As a result, SCAPE enables coverage (or recall) control for error detection under mild distributional assumptions, and is particularly effective when classifiers provide complementary information across layers or regions.

Non-Conformity Vectors. For each extracted item (either a table name or an attribute value) with hidden representations $\{h^{(l)}_{k,\circ}\}_{l \in \mathcal{L}}$ (as per §4.2), we first obtain the sigmoid outputs $\pi^{(l)}_{k,\circ} = f^{(l)}(h^{(l)}_{k,\circ}) \in [0,1]$ from each layer-wise classifier $f^{(l)}$ at layer $l \in \mathcal{L}$ (as detailed in Equation (7)). We then define a *multi-dimensional non-conformity vector* $\mathbf{s}(c) \in \mathbb{R}^{|\mathcal{L}|}$ for each candidate label $c \in \{0,1\}$ (where c=1 denotesw erroneous extraction and c=0 denotes correct extraction) as:

$$\mathbf{s}(c) = \left[s_l(c)\right]_{l=1}^{\mathcal{L}}, \quad \text{where } s_l(c) = \begin{cases} 1 - \pi^{(l)}_{k,\circ} & \text{if } c = 1, \\ \pi^{(l)}_{k,\circ} & \text{if } c = 0. \end{cases} \tag{11}$$

which reflects how atypical the outputs of the layer-specific error classifiers are, under the assumption that the extraction is either erroneous or correct.

Cell Construction and Selection. We leverage a small labeled calibration dataset $\mathcal{D}_{\text{cal-base}} = \{(x_i, y_i)\}$ constructed as in §4.1, where $y_i \in \{0,1\}$ indicates whether an extraction is erroneous. We randomly split $\mathcal{D}_{\text{cal-base}}$ into two disjoint subsets: $\mathcal{D}_{\text{cell}}$, used to partition the non-conformity score space into K cells via k-means clustering, and $\mathcal{D}_{\text{re-cal}}$, used to rank and select cells for coverage. Each cell corresponds to a cluster in $\mathbb{R}^{|\mathcal{L}|}$, and new score vectors are assigned by nearest-centroid matching. Each cell C_m groups similar non-conformity patterns. To identify the most reliable regions of the score space for detecting true extraction errors, we rank cells based on their *false-to-true ratio* on $\mathcal{D}_{\text{re-cal}}$, that is, for each cell C_m, and for $c \in \{0,1\}$ the number of examples with y_j=c whose score vectors $\mathbf{s}(1-c)$ fall into the cell C_m (i.e., false examples), divided by the number of examples y_j=c whose $\mathbf{s}(c)$ also fall into the cell (i.e., true examples):

$$\rho_m = \frac{\sum_{c \in \{0,1\}} |(x_j, c) \in \mathcal{D}_{\text{re-cal}} : \mathbf{s}(1-c) \in C_m|}{\sum_{c \in \{0,1\}} |(x_j, c) \in \mathcal{D}_{\text{re-cal}} : \mathbf{s}(c) \in C_m|}. \tag{12}$$

[7]This design avoids relying on the assumption that any single layer has to consistently behave correctly. Instead, as long as correctness-related signals exist in a distributed manner within the network, our cross-layer aggregation effectively captures them.

Algorithm 1: SCAPE

Require: Calibration dataset $\mathcal{D}_{\text{cal-base}} = \{(x_i, y_i)\}_{i=1}^{N_{\text{cal-base}}}$, split into $\mathcal{D}_{\text{cells}}$ and $\mathcal{D}_{\text{re-cal}}$; Coverage level $\alpha \in (0, 1)$;
Ensure: Error prediction set $\hat{y}_{k,\circ}^{\text{SCAPE}} \subseteq \{0, 1\}$;
1: Compute non-conformity vectors using Eq. (11), for all $(x_i, y_i) \in \mathcal{D}_{\text{cal-base}}$.
2: Partition non-conformity space to obtain cells $C_1, \ldots, C_K$ on $\mathcal{D}_{\text{cell}}$.
3: Rank cells in ascending order of ρ_m computed using Eq. (12): $C_{(1)}, C_{(2)}, \ldots, C_{(K)}$.
4: Select top-ranked cells C_α through Eq. (13).
5: For new extraction: $\hat{y}_{k,\circ}^{\text{SCAPE}} = \{y \in \{0, 1\} : \mathbf{s}(y) \in C_\alpha\}$.

We rank the cells in ascending order of ρ_m, prioritizing those where non-error examples are least likely to be mistaken as errors.

To guarantee the desired coverage level, we select the smallest set of top-ranked cells such that the score vectors of the true labels for at least $\lceil (1-\alpha)(N_{\text{re-cal}}+1) \rceil$ examples in $\mathcal{D}_{\text{re-cal}}$ fall within the selected cells. Formally, let all cells be *re-indexed* as $C(1), C_{(2)}, \ldots, C_{(K)}$, sorted in order of increasing false-to-true ratio. We define the selected region as $C_\alpha \subset \mathbb{R}^{|\mathcal{L}|}$:

$$C_\alpha = \bigcup_{j=1}^{\eta^*} C_{(j)}, \qquad \text{where } \eta^* = \min\left\{\eta \in 1..K \mid \left|\{(x_j, y_j) \in \mathcal{D}_{\text{re-cal}} : \mathbf{s}(y_j) \in C_\alpha\}\right| \geq \lceil (1-\alpha)(N_{\text{re-cal}}+1) \rceil\right\} \tag{13}$$

where $\alpha \in (0, 1)$ is the user-specified miscoverage tolerance.

Test-Time Inference. At test time, for each new extraction with hidden representations $\{h_{k,\circ}^{(l)}\}_{l \in \mathcal{L}}$ (as per §4.2), we compute the non-conformity vectors $\mathbf{s}(y)$ for both possible labels $y \in \{0, 1\}$ (via Equation (11)), and define the conformal prediction set as:

$$\hat{y}_{k,\circ}^{\text{SCAPE}} = \{y \in \{0, 1\} \mid \mathbf{s}(y) \in C_\alpha\}. \tag{14}$$

We accept an extraction if $\hat{y}_{k,\circ}^{\text{SCAPE}} = \{0\}$; otherwise (if 1 is included) we flag it for correction.

SCAPE is presented as Algorithm 1, where line 1 corresponds to the computation of non-conformity vectors as defined in Equation (11), lines 2-3 implement the clustering and ranking procedure described in Equation (12), line 4 selects cells according to the coverage constraint in Equation (13), and line 5 defines the prediction set $\hat{y}_{k,\circ}^{\text{SCAPE}}$ for each new extraction as Equation (14).

Coverage Guarantee. For any erroneous extraction (i.e., $y_{k,\circ}=1$),

Theorem 5.1 (Coverage Guarantee under Exchangeability). *Under the assumption that calibration and test examples are exchangeable, the conformal prediction set determined by SCAPE satisfies:*

$$\mathbb{P}\left(y_{k,\circ} \in \hat{y}_{k,\circ}^{SCAPE}\right) \geq 1 - \alpha,$$

where $\hat{y}_{k,\circ}^{SCAPE} = \{y \in \{0, 1\} \mid \mathbf{s}(y) \in C_\alpha\}$.

The proof is available in the accompanying technical report [11].

Set Size Optimality. Theorem 5.2 shows that SCAPE asymptotically minimizes the expected prediction set size while maintaining $(1-\alpha)$ coverage; proof details are in [11].

The proof, which leverages the Neyman-Pearson lemma [36, 45] to show that ranking cells by false-to-true ratio ρ_m (as Equation (12)) is equivalent to optimizing the likelihood ratio $\Lambda(\mathbf{s})$, is provided in the technical report [11].

Theorem 5.2 (Optimal Set Size of SCAPE). *Assume that for each $c \in \{0, 1\}$ the label-conditional densities $p(\mathbf{s}(c) \mid y=0)$ and $p(\mathbf{s}(c) \mid y=1)$ exist and are continuous (y represents the true label). Then, SCAPE asymptotically minimizes the expected prediction set size $\mathbb{E}[|\hat{y}_{k,\circ}^{SCAPE}|]$ subject to coverage $\geq 1 - \alpha$.*

Discussion and Limitations. In practice, classifier calibration and finite-sample effects can impact performance when $|\mathcal{D}_{\text{cal-base}}|$ is small. We also find that inter-layer disagreement provides an additional signal for errors, motivating the conflict-augmented extension below (see [11] for further discussion). While the coverage guarantee of SCAPE (Theorem 5.1) relies only on the exchangeability assumption rather than the stronger i.i.d. assumption, it may still degrade under severe domain shifts [48, 52, 59] when calibration samples are not representative, which we leave as an important direction for future work. Specifically, exchangeability is weaker than i.i.d., allowing for correlated or non-independent samples as long as calibration and test data are jointly exchangeable. Accordingly, the guarantee requires only this standard assumption, rather than identical data generation mechanisms, as widely adopted in the conformal prediction literature [2, 7, 45, 52].

5.2 SCAPE-Hyb: Hybrid Method

We now introduce SCAPE-Hyb, a unified method that incorporates the inter-layer conflict signal (§4.3) into the conformal prediction framework (§5.1). The key idea is to augment the multi-dimensional non-conformity vector with a scaled conflict term, allowing the conformal predictor to respond not only to probabilistic uncertainty but also to internal disagreement among classifiers.

Conflict-Augmented Non-Conformity. The conflict score κ in CF (Equation (10)) is a coarse, thresholded measure of disagreement and often collapses to small or zero values despite substantial variation in classifier probabilities. We therefore replace κ with a more informative, real-valued *disagreement score* Δ. Let $\pi^{(l)}$ denote the predicted probability of erroneous extraction from layer l, and let $\bar{\pi} = \frac{1}{|\mathcal{L}|}\sum_{l \in \mathcal{L}} \pi^{(l)}$ be the average probability across layers. We define the disagreement score as: $\Delta = \max_{l \in \mathcal{L}} |\pi^{(l)} - \bar{\pi}|$, which reflects the maximum deviation from consensus among the classifiers. We then embed it as an additional dimension into the non-conformity vector by defining a conflict-calibrated representation:

$$\mathbf{s}_\lambda(c) = \Big[(1-\lambda) \cdot s_1(c), \ldots, (1-\lambda) \cdot s_{|\mathcal{L}|}(c), \lambda \cdot \Delta\Big], \tag{15}$$

where each $s_l(c)$ is the layer-wise non-conformity score defined in §5.1, and $\lambda \in [0, 1]$ is a tunable parameter that adjusts the relative weight of the disagreement signal inside the conformal prediction pipeline. When $\lambda=0$, the method falls back to SCAPE (§5.1); on the other hand, when $\lambda=1$, the method ignores all layer-wise non-conformity scores and relies exclusively on the disagreement score Δ, effectively acting as a calibrated variant of conflict filtering (§4.3).

Calibration and Prediction. We follow the same calibration protocol as SCAPE, but operate on $\mathbf{s}_\lambda(\cdot)$ (Equation (15)) to obtain C_α^λ; the test-time set is:

$$\hat{y}_{k,\circ}^{\text{Hyb}} = \left\{y \in \{0, 1\} \mid \mathbf{s}_\lambda(y) \in C_\alpha^\lambda\right\},$$

We accept the extraction if $\hat{y}^{\text{Hyb}}_{k,\circ}=\{0\}$; otherwise we flag it. The coverage guarantee (Theorem 5.1) continues to hold under the same exchangeability assumption.

Conflict-Aware Optimality. SCAPE-Hyb inherits the same asymptotic set-size optimality principle as SCAPE, now over the augmented score space; proof details are in [11].

Advantage of SCAPE-Hyb. If conflict is informative (errors tend to exhibit higher disagreement), SCAPE-Hyb can reduce expected correction cost while preserving coverage.

Theorem 5.3 (Optimality of SCAPE-Hyb over SCAPE). *Assume that the conflict score provides an additional signal for distinguishing erroneous from correct extractions, i.e., erroneous examples (y=1) are more likely to have a higher conflict score than correct ones. Then, under the same $(1-\alpha)$ coverage constraint, SCAPE-Hyb produces a prediction set with equal or smaller expected size compared to SCAPE:*

$$\mathbb{E}\left[\left|\hat{y}^{Hyb}_{k,\circ}\right|\right] \leq \mathbb{E}\left[\left|\hat{y}^{SCAPE}_{k,\circ}\right|\right].$$

A detailed formal proof of Theorem 5.3 is provided in the accompanying technical report [11]. Our experimental results (§7) empirically corroborate this theoretical advantage.

SCAPE-Hyb can be viewed as a *soft* generalization of conflict filtering: conflict is embedded as an extra dimension and calibrated within the conformal framework, avoiding brittle thresholds while retaining coverage.

Summary. The parameter λ interpolates between SCAPE (λ=0) and pure-disagreement filtering (λ=1). In practice, λ can be selected on a validation set; intermediate values often work well (§7.3).

6 CONSOLIDATION OVER DOC CHUNKS

Due to LLM context and cost constraints, long documents must be processed via chunking. However, the one-to-one assumption between chunks and table rows used earlier for exposition does not always hold: a logical record may be fragmented across multiple chunks, creating a mismatch between chunk-level processing and row-level table construction.

To address this, ReDD adopts a *MapReduce-style* [46] design that is robust to arbitrary chunking. Rather than designing new chunkers, ReDD treats chunking as a black box and aggregates chunk-level signals into document-level structures. For documents corresponding to a single record, multiple chunks may each contain partial attributes. ReDD resolves this fragmentation via consolidation steps for schema discovery and data population, enabling stable, query-relevant tables to be constructed from fragmented evidence. Specifically, we introduce **SchemaMerge** for document-level schema discovery and **RowMerge** for document-level data population.

- **SchemaMerge (document-level schema discovery).** Chunking may fragment schema evidence within a single document, where attributes or join keys become identifiable only when information from multiple chunks of the same document is considered jointly. SchemaMerge aggregates chunk-level schema proposals into a stable, query-relevant schema by normalizing and consolidating local candidates (Algorithm 2).
- **RowMerge (document-level row construction).** When attributes of a record are distributed across chunks, RowMerge first extracts attribute values independently from individual chunks (*map*) and then aggregates them at the document level to construct complete rows (*reduce*). By consolidating partial assignments (i.e., (T_k, a_i, v_{k,a_i}) table T_k, attribute a_i, and extracted value v_{k,a_i}) using deterministic resolution rules, as per Algorithm 3, this process can mitigate extraction failures caused by record semantics being split across multiple chunks.

Overall, this consolidation mechanism decouples ReDD from any specific chunking strategy. We empirically evaluate robustness under different chunking methods and validate the necessity of SchemaMerge and RowMerge via ablation studies in §7.4.2.

Algorithm 2: SchemaMerge

Input : document set $\mathcal{D}$; query q
Output: final schema $\mathcal{S}$

```
foreach d ∈ D do
    C ← Chunk(d)
    foreach c_k ∈ C do
        ΔS_k ← Schema(c_k, q)      // Update Schema Using Eq.1&2
        S_C ← S_C ∪ ΔS_k
S ← SchemaRepair(S_C, q)           // Schema Repair
return S
```

Algorithm 3: RowMerge

Input : document d; schema $\mathcal{S}$; query q
Output: populated tables $\mathcal{T}$

```
C ← Chunk(d);   E ← ∅
foreach c_k ∈ C do
    T_k, A_k ← Map(c_k, S, q)  // Table Resolution: Variant of Eq.3
    foreach a_i ∈ A_k do
        v_{k,a_i} ← Extract(c_k, a_i)    // Attr Extraction Via Eq.4
        E ← E ∪ {(T_k, a_i, v_{k,a_i})}
T ← MergeToTable(E, ResolveConflict)
return T
```

Table 1: Evaluation Datasets Used for Assessing ReDD.

Dataset	Dataset Source	# Queries	Avg. Result Rows
Spider	Spider [60]	86	1733
Bird	Bird [24]	36	2435
Galois	Galois [43]	10	497
FDA	FDA 510(k) [57]	6	100
CUAD	CUAD [46]	15	501

7 EXPERIMENTAL EVALUATION

7.1 Experimental Setup

7.1.1 Datasets. We evaluate ReDD on *five datasets* that simulate realistic information extraction and analytical query scenarios over unstructured or weakly structured document collections. Table 1 summarizes these datasets, detailing the number of queries per set and the average number of expected output table entries per query. The first two datasets, Spider and Bird, are derived from the Spider [60] and Bird [24] benchmarks. Using their original schemas and tabular data, following [32] we convert tabular rows

into natural language document chunks using a state-of-the-art LLM (GPT-5 [38]). While we acknowledge that generating natural language documents from structured data may reduce the difficulty of populating data back into a schema, we mitigate this by ensuring that documents are not generated from a uniform template. Instead, the datasets provided for queries often contain documents derived from diverse tables and narrative styles, thereby simulating the complexity of multi-source document aggregation. This ensures that the model used in ReDD (Qwen3-30B-A3B [47]) has not seen these documents during training, thereby mitigating data leakage concerns. For each benchmark query, we know the precise result (ground truth) to evaluate correctness. ReDD is evaluated on the original natural language queries from Spider and Bird datasets, as well as on newly introduced queries that involve multi-table joins, aggregation, and multi-hop reasoning. In our evaluation, we do not account for natural language to SQL translation; instead, we provide the correct SQL query to execute on the extracted tables. We do this to isolate our evaluation from text-to-SQL translation errors; naturally any state-of-the-art text-to-SQL technique can be adopted. While generating the datasets, we randomly shuffle the chunks to eliminate any semantic correlations. We also introduce varying degrees of information density in the document collection (ratio of relevant vs irrelevant chunks to the query) in §7.4. The prompts used to generate the documents, due to space constraints, are provided in the accompanying technical report [11]. The dataset Galois is sourced from the Fortune and Premier datasets as described in [43]. The dataset FDA is sourced from FDA 510(k) regulatory filings [4, 57], which are long-form and heterogeneous, containing narrative summaries, tabular sections, and metadata. The dataset CUAD is a legal-contract benchmark [46], whose lengthy documents make single-pass LLM processing infeasible due to context limitations [46]. For these datasets, we use both the original benchmark queries (typically involving a limited set of documents and attributes) and additional new queries proposed in this paper. The additional queries are designed to include more attributes and to incorporate cross-document aggregation as well as multi-hop reasoning.

7.1.2 Measurements. We evaluate ReDD in two stages: schema discovery (ISD) and data population (TDP). For schema discovery, we first assess whether the discovered schema for each query is sufficient to answer the query, resulting in an accuracy metric denoted as ACC_{sch}. Additionally, by comparing the discovered schema with the ground-truth schema in terms of their attributes, we compute attribute-level recall (Rec^{attr}_{sch}) and precision (Pre^{attr}_{sch}), which measure the completeness and redundancy of the discovered schema.

For data population, we evaluate the accuracy of the extracted tabular data by comparing it with the ground truth at the *cell* (attribute value) level. Specifically, each ground-truth cell (i.e., each populated value in the ground-truth table) is checked against the extracted result. We then compute the following accuracy metric [46]:

$$ACC_{pop} = 1 - \frac{\#\text{ missing cells} + \#\text{ incorrect cells}}{\#\text{ ground-truth cells}} \tag{16}$$

Here, *missing* cells are those that should have been extracted but were not, and *incorrect* cells are those that were extracted but contain erroneous values.

SCAPE and SCAPE-Hyb, identify potentially erroneous cells and use additional review steps (e.g., human inspection) to validate and fix extraction errors. If correctly extracted cells are flagged for inspection, this results in unnecessary correction efforts. To quantify this, we compute the false positive rate:

$$FPR_{pop} = \frac{FP}{FP + TN}, \tag{17}$$

where FP (false positives) denotes correctly extracted cells flagged for inspection and TN (true negatives) denotes correctly extracted cells not flagged. For a fixed ACC_{pop}, a higher FPR_{pop} implies additional wasted effort inspecting accurate extractions, while a lower FPR_{pop} reflects a more efficient process, focusing inspections primarily on erroneous extractions. Thus, FPR_{pop} quantifies the inefficiency (unnecessary inspections) in the extraction pipeline.

Finally, we evaluate end-to-end query answering by executing the ground-truth SQL over the populated tables and checking whether the query result matches the ground truth, yielding the query-level accuracy:

$$ACC_{qry} = \frac{\#\text{ correctly answered queries}}{\#\text{ queries}} \tag{18}$$

7.1.3 Experimental Settings. ReDD is implemented in Python. We use GPT-5 [38] for schema discovery and Qwen3-30B-A3B [47] for data population on an A100 (80GB) GPU. Unless noted otherwise, λ=0.5 (Eq. (15)); classifiers are trained on 50 entries and calibrated on 150 randomly sampled entries per query.

7.2 Results of Schema Discovery

We evaluate the schema discovery stage (ISD) over all datasets. As shown in Table 2, the final pipeline (Phase I & II + Repair, see §3) achieves perfect sufficiency (identifies all required attributes) on all queries, with zero invalid cases—i.e., every extracted schema contains all necessary attributes required to answer the query, resulting in an average recall of Rec^{attr}_{sch}=1.000. In addition, the extracted schemas remain concise, with an average precision of Pre^{attr}_{sch}=0.956, meaning only a few redundant attributes are occasionally included—improving the efficiency of downstream data population.

7.2.1 Ablation Study of ISD Components. We conduct an ablation study to assess the contribution of each ISD component. Table 2 compares four configurations derived from Section 3:

- Phase I Only: A query-independent document-based schema extraction approach that achieves high recall (0.989)—slightly below 1.0, as some query-relevant attributes are deeply embedded in the text and not easily surfaced by a general approach. However, it suffers from very low precision (0.522), as it tends to include many irrelevant attributes not required by the query.
- Phase II Only: A query-specific strategy that improves precision (0.968) but often misses necessary attributes due to the lack of contextual information, resulting in many invalid schemas.
- Phase I & II: Combining both phases significantly improves overall effectiveness, achieving high recall (0.991) and precision (0.952), and greatly reducing invalid extractions.
- Phase I & II + Repair: A repair step further eliminates remaining invalid cases, achieving 100% valid extractions.

These results highlight the effectiveness of the two-phase design in achieving both schema completeness and compactness, while the repair step ensures full robustness across diverse query scenarios.

Table 2: Evaluation of Schema Discovery over All Datasets.

Method	# Invalids	Rec^{attr}_{sch}	Pre^{attr}_{sch}
ISD (Phase I Only)	2	0.989	0.522
ISD (Phase II Only)	12	0.951	0.968
ISD (Phase I & II)	1	0.991	0.956
ISD (Phase I & II + Repair)	**0**	**1.000**	**0.956**

Table 3: Summary of Data Population Accuracy (ACC_{pop}).

Method	Spider	Bird	Galois	FDA	CUAD
EVAPORATE [4]	-	-	0.475	0.516	0.209
Palimpzest [30]	-	-	0.867	0.924	0.613
ReDD (No Correction)	0.938	0.917	0.926	0.908	0.813
ReDD (SC [53])	0.947	0.927	0.934	0.916	0.821
ReDD (EC [61])	0.956	0.938	0.946	0.927	0.863
ReDD (SCAPE)	0.988	0.985	0.987	0.988	0.924
ReDD (SCAPE-Hyb)	**0.990**	**0.988**	**0.988**	**0.983**	**0.979**

Table 4: Correction Overhead (FPR_{pop}) in Data Population.

Method	Spider	Bird	Galois	FDA	CUAD
ReDD (SCAPE)	0.063	0.054	0.074	0.081	0.114
ReDD (SC [53])	0.050	0.051	0.063	0.049	0.075
ReDD (EC [61])	0.051	0.052	0.068	0.056	0.081
ReDD (SCAPE-Hyb)	**0.045**	**0.048**	**0.052**	**0.049**	**0.072**

7.3 Results of Data Population and Correction

7.3.1 Accuracy and Correction Cost. Table 3 reports the data population accuracy (ACC_{pop}, defined in Eq. 16) under multiple approaches, including: (i) ReDD (No Correction), which directly outputs the extracted table; (ii) our correction modules SCAPE (SCAPE) and SCAPE-Hyb (SCAPE-Hyb) under default parameters (α=0.15); (iii) two confidence-based correction baselines, Self-Consistency Confidence (SC) [53] and Elicited Confidence (EC) [61][8]; and (iv) prior end-to-end baselines, EVAPORATE [4] and Palimpzest[9] [30, 31]. All metrics are averaged over all queries in Table 1.

Without correction, ReDD achieves reasonably high accuracy (e.g., 0.938 on Spider and 0.917 on Bird), but still leaves a non-trivial fraction of erroneous/missing cells. Since EVAPORATE and Palimpzest are not designed to handle schemas involving multiple tables, we present their evaluation only on Galois, FDA, and CUAD datasets (which do not involve multiple tables). Using the same base model (Qwen3-30B-A3B [47]), EVAPORATE performs considerably worse than other approaches, while Palimpzest achieves accuracy comparable (though slightly lower) than ReDD *without correction*. SC and EC improve accuracy modestly over No Correction across all datasets (e.g., 0.947/0.956 on Spider), with EC consistently stronger than SC. By contrast, applying our correction algorithms yields substantially larger gains: SCAPE raises accuracy to 0.988/0.985 on Spider/Bird, and SCAPE-Hyb further improves to 0.990/0.988, achieving the best overall accuracy across all datasets.

The CUAD dataset poses unique challenges due to its long-document nature: a single legal contract (corresponding to one row in the ground-truth table) can be nearly 100,000 tokens, far exceeding the context window of Qwen3-30B-A3B [47]. To address this, we adapt a map-reduce style document chunking strategy inspired by DocETL [46]. Combined with our correction algorithm, this chunk-merge (map-reduce) pipeline enables SCAPE-Hyb to reach 0.979 accuracy on CUAD.

Table 4 reports the corresponding correction overhead measured by the false positive rate (FPR_{pop}). Overall, SCAPE-Hyb keeps overhead modest and is lower than the confidence baselines (SC/EC) across all datasets, while also substantially reducing the overhead of SCAPE on long-document CUAD (0.072 vs. 0.114). These results demonstrate the effectiveness of our correction module in improving extraction quality while incurring minimal verification overhead.

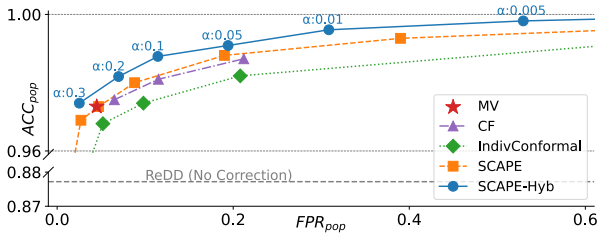


Figure 2: Trade-off between data population accuracy (ACC_{pop}) and correction cost measured by the false positive rate (FPR_{pop}).

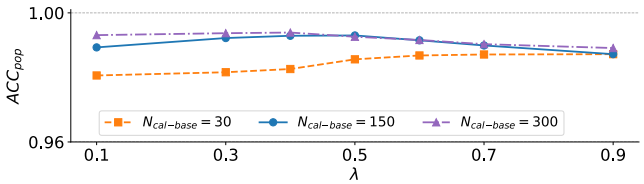


Figure 3: Data population accuracy ACC_{pop} for SCAPE-Hyb with different calibration dataset size $N_{\text{cal-base}}$ on dataset Spider, varying conflict weight λ, under FPR_{pop}=0.2.

7.3.2 Ablation: Accuracy-Cost Tradeoff. We ablate SCAPE-Hyb by comparing it with MV (§4.3), CF (§4.3), SCAPE, and *IndivConformal* (conformal-calibrated base classifiers [2] followed by majority voting). Figure 2 plots ACC_{pop} vs. FPR_{pop} averaged over all datasets (gray dashed line: ReDD without correction). All methods improve over the baseline, but MV provides only a single operating point, IndivConformal is consistently weaker, and CF is dominated by SCAPE. SCAPE and SCAPE-Hyb achieve the best accuracy–cost trade-offs, with SCAPE-Hyb consistently dominating, validating the benefit of adding conflict information on top of SCAPE.

7.3.3 Effect of Coverage Threshold α. Figure 2 (avg. across datasets) shows the expected trade-off: as α decreases, SCAPE-Hyb becomes more conservative (higher ACC_{pop} but higher FPR_{pop}). Even at large α, SCAPE-Hyb attains high initial accuracy (≥0.974), while very

[8]Both baselines operate on top of ReDD's extracted outputs, replacing SCAPE/SCAPE-Hyb with confidence scores to trigger corrections on low-confidence cells. SC estimates confidence via answer agreement across multiple LLM samples, while EC elicits uncertainty and fidelity signals directly from the LLM. We use a fixed confidence threshold to trigger corrections [53, 61]. These black-box, confidence-only baselines do not leverage hidden-state classifiers or conflict signals, enabling a direct comparison with our error-aware design; implementation details are in our technical report [11].

[9][43] in their recent evaluation, demonstrated that Palimpzest has superior accuracy over other prior baselines. For this reason we compare with Palimpzest, without including the baselines that [43] demonstrated inferior to Palimpzest. We run experiments with the same configuration parameters as in [43].

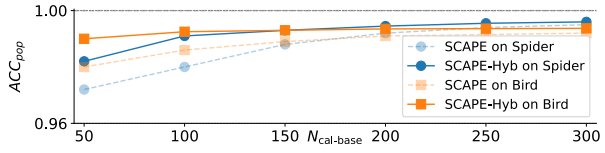


Figure 4: Data population accuracy ACC_{pop} of SCAPE and SCAPE-Hyb varying calibration dataset size $N_{\text{cal-base}}$, under FPR_{pop}=0.2.

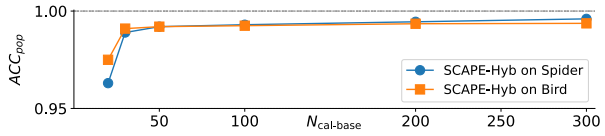


Figure 5: Data population accuracy ACC_{pop} varying training dataset size N_{cls}, under FPR_{pop}=0.2.

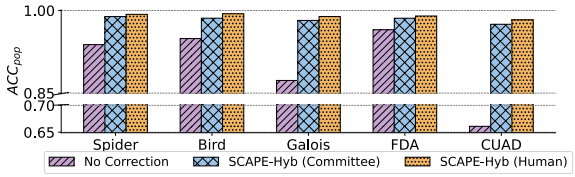


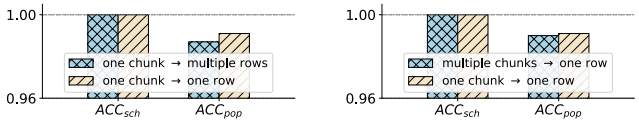
Figure 6: Data population accuracy ACC_{pop} using human-annotated vs. LLM committee-generated training data.

small α can approach perfect accuracy at higher cost. On Spider (avg. 1,733 rows/query), α=0.5 yields 0.947 accuracy with 4.9% reviewed (8 false positives), α=0.3 yields 0.975 with 9.2% reviewed (37 false positives), and α=0.01 yields 0.998 with 31% reviewed (608 false positives). Overall, modest review effort suffices for >0.97 accuracy, while achieving 1.0 requires substantially more verification (§9).

7.3.4 Effect of Conflict Weight λ. Figure 3 studies λ on Spider at FPR_{pop}=0.2. Under the default $N_{\text{cal-base}}$=150 (blue curve), accuracy peaks at $\lambda \approx 0.5$ and drops when λ is too small or too large, supporting our default λ=0.5: under-weighting misses disagreement signals, while over-weighting can suppress correct but less frequent outputs. The same pattern holds across datasets, while the optimal λ shifts with calibration size (e.g., $\approx$ 0.9 at $N_{\text{cal-base}}$=30 and $\approx$ 0.3–0.4 at $N_{\text{cal-base}}$=300).

7.3.5 Effect of Calibration Dataset Size $N_{cal\text{-}base}$. Figure 4 varies $N_{\text{cal-base}}$ (§5.1) and reports ACC_{pop} for SCAPE and SCAPE-Hyb at FPR_{pop}=0.2 (we show Spider and Bird for brevity). Accuracy improves steadily with more calibration data and plateaus once $N_{\text{cal-base}}$ exceeds roughly 100–150 examples. This indicates strong data efficiency (e.g., SCAPE-Hyb exceeds 0.99 accuracy on Spider with 150 calibration examples), and SCAPE-Hyb is especially robust when calibration data is scarce by leveraging conflict information.

7.3.6 Effect of Training Dataset Size N_{cls}. Figure 5 demonstrates that data population accuracy (ACC_{pop}) improves with larger training set size (N_{cls}), but plateaus quickly—for example, reaching >0.99 accuracy with just 50 examples. These results suggest that our approach is label-efficient: high accuracy can be achieved with limited training data and scales well as more data becomes available. In deployment, $\mathcal{D}_{\text{cls}}$ could be bootstrapped via uncertainty-driven sampling rather than uniform labeling, using human or LLM-committee



(a) One-to-many setting. **(b) Many-to-one setting.**

Figure 7: Schema discovery accuracy ACC_{sch} and data population accuracy ACC_{pop} under different chunk-to-table mapping settings.

supervision (§4.1); a full active-learning treatment is left to future work.

7.3.7 Impact of Label Source: Human vs. LLM-Synthetic. Human annotations correspond to manual verification of extraction correctness, where annotators compare the original document text with the extracted outputs to assess factual consistency. All annotations in our experiments were performed by members of our research team using unified criteria, providing a high-quality and trustworthy human reference signal rather than a large-scale crowd-sourced dataset. These labels are used only for the small supervision sets required by the correction pipeline (e.g., classifier training and calibration), and disagreements were resolved by re-examining the source text and reaching consensus.

Figure 6 compares data population accuracy (ACC_{pop}) when the training labels used in correction are sourced from either human annotations or LLM committee decisions. SCAPE-Hyb performs comparably under both label sources, with <1% difference in accuracy. The results show that both sources yield highly consistent final data extraction accuracy, demonstrating the robustness of the LLM committee in practice without introducing significant harmful bias (Figure 6). This suggests that high-quality synthetic labels from LLM committees can be a viable alternative to human annotations in training error detection classifiers. Compared to data sourced from humans, LLM-generated labels are relatively cheaper, easier to obtain, and more practical for scaling to large datasets.

7.4 Additional Analyses

7.4.1 Results under 1-to-Many and Many-to-1 Chunk-to-Table Mappings. We test ReDD beyond the default one-to-one chunk/row assumption under both *1-to-many* and *many-to-1* mappings, which commonly arise in real-world corpora. For 1-to-many, we modify Spider by merging two or three chunks with different schemas so one chunk contributes to multiple rows or tables. Figure 7a shows unchanged ACC_{sch} and only a 0.13% drop in ACC_{pop}. For many-to-1, Figure 7b injects redundancy and conflicts across chunks referring to the same logical row; both schema and population accuracy remain nearly unchanged. Overall, ReDD remains robust without requiring strict one-to-one chunk/row alignment.

7.4.2 Ablation Study under Document Chunking. Figure 8 summarizes an ablation study evaluating robustness under realistic document chunking. We evaluate ReDD on the long-document CUAD dataset under semantic chunking and fixed-size chunking, with and without SchemaMerge +RowMerge. With consolidation enabled, performance remains high under both strategies (Rec_{sch}^{attr}=1.000/0.989 and ACC_{pop}=0.978/0.969 for semantic/fixed-size chunking). Without consolidation, performance degrades, especially under fixed-size chunking (ACC_{pop} drops to 0.845). Even when

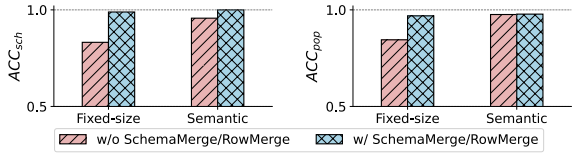


Figure 8: Ablation study comparing Rec_{sch}^{attr} and ACC_{pop} with vs. without SchemaMerge +RowMerge, varying chunking strategy.

Table 5: Query-level End-to-End Accuracy (ACC_{qry}).

Method	Spider	Bird	Galois	FDA	CUAD
EVAPORATE [4]	–	–	0.100	0.000	0.067
Palimpzest [30]	–	–	0.200	0.167	0.133
ReDD (No Correction)	0.116	0.111	0.200	0.167	0.200
ReDD (SC [53])	0.163	0.167	0.300	0.167	0.267
ReDD (EC [61])	0.209	0.222	0.300	0.333	0.333
ReDD (SCAPE-Hyb)	**0.988**	**0.972**	**1.000**	**1.000**	**1.000**

semantic chunking preserves accuracy reasonably well, RowMerge still substantially improves token efficiency, reducing end-to-end token consumption from 2.6×10^9 to 8.3×10^8. These results confirm that document-level consolidation is important for recovering complete rows and schema evidence, especially when attributes are fragmented across fixed-size chunks.

7.4.3 End-to-End Query Answering Accuracy. We report a strict end-to-end query accuracy metric (ACC_{qry}) by executing the ground-truth SQL over the populated tables and comparing results with the ground truth. As shown in Table 5, prior systems (EVAPORATE and Palimpzest), despite being provided with the ground-truth schemas, achieve low query-level accuracy, and confidence-only baselines (SC/EC) offer only modest improvements. In contrast, SCAPE-Hyb achieves near-perfect query accuracy across all benchmarks (0.988 on Spider, 0.972 on Bird, and 1.000 on Galois/FDA/CUAD), demonstrating that our extracted schemas and tables are reliably good enough to answer the target queries.

7.4.4 Runtime and Token Considerations. We report *real-time query execution* latency (wall-clock time from query submission to result generation) and token usage. In all experiments, total execution is under 4 hours: about 40 minutes for ISD, 3 hours for TDP on an 8 A100 GPU cluster, and 20 minutes for error correction. Each query processes about 1,500 document chunks on average, making TDP the dominant latency bottleneck at roughly 7.2 seconds per chunk. TDP already supports batching and parallelism with up to 8 concurrent model inferences, whereas ISD remains sequential due to schema update dependencies. On Galois, average token consumption is approximately 18M tokens per query, so reducing token cost remains important future work.

8 RELATED WORK

Document-to-Database/Table. LLMs have enabled systems that transform heterogeneous documents into structured, queryable tables or database-like representations. Prior work has explored this space from several complementary angles, including structured extraction over heterogeneous data lakes [4], ad-hoc querying over document collections [29], declarative and optimized document-processing pipelines through Palimpzest [31], its subsequent systemization [30], and DocETL [46], and abstractions for querying unstructured data such as UQE [13] and LOTUS/semantic operators [41]. Other efforts study more specialized settings, including templatized documents [28], table selection over large document collections [6], integration-aware database updating [18], query-specific schema discovery and table-based reasoning in SRAG [26] and DocSage [27], Doc2Table benchmarking [15], and schema bootstrapping for structured retrieval [25]. While effective in their target settings, these systems generally address only part of the pipeline, rather than end-to-end, query-driven relational schema construction and population over heterogeneous document collections with statistically grounded error detection and correction.

Agentic Prompt/Program Optimization Frameworks. Agentic frameworks such as DSPy [21] and LangChain/LangGraph [34] focus on orchestrating multi-step LLM calls and optimizing prompt or program structures for downstream tasks. These systems primarily address how to synthesize, tune, and execute LLM pipelines, rather than the data management challenges of schema discovery, structured data construction, and reliability. We view them as orthogonal and composable with our work: such frameworks can be used to optimize internal LLM calls, while our work targets end-to-end relational structuring with accuracy guarantees.

Text-to-SQL Models. Text-to-SQL models study the problem of translating natural language questions into executable SQL queries given a relational schema. Representative approaches include sequence-to-sequence and encoder–decoder models over database schemas [54, 60], as well as more recent LLM-based methods that leverage in-context learning or fine-tuning to improve generalization [23, 44]. These models are complementary to our setting. Once query-ready relational tables and schemas are constructed, any state-of-the-art text-to-SQL technique can be used to translate a natural language query into SQL. However, text-to-SQL methods typically assume that the target schema already exists and focus on query translation, rather than addressing query-driven schema discovery or reliable table population from unstructured documents. As such, they operate downstream of the document-to-table extraction problem studied in our work.

Query Execution and Optimization. Systems such as ELEET [51], TAG [9], Unify [55], CAESURA [50], and related LLM-powered analytics engines [1, 56] focus on query planning and execution over text or multimodal data. Our work instead targets reliable relational table construction from unstructured documents and provides formal accuracy guarantees.

9 CONCLUSION AND FUTURE WORK

This paper introduces ReDD, a framework for executing error-aware analytical queries over unstructured data by combining query-specific schema discovery, relational table population, and statistically calibrated error detection. Experiments show that ReDD reduces extraction errors from as high as 30% to less than 1% while maintaining 100% schema recall. The proposed reliability framework suggests several directions for future work, including integrating ReDD with other unstructured data query processing systems [50], improving performance and monetary efficiency, and further mitigating bounded extraction errors.

REFERENCES

[1] Eric Anderson, Jonathan Fritz, Austin Lee, Bohou Li, Mark Lindblad, Henry Lindeman, Alex Meyer, Parthkumar Parmar, Tanvi Ranade, Mehul A. Shah, Benjamin Sowell, Dan Tecuci, Vinayak Thapliyal, and Matt Welsh. 2025. The Design of an LLM-powered Unstructured Analytics System. In *15th Conference on Innovative Data Systems Research (CIDR 2025)*. CIDR, Amsterdam, The Netherlands. https://www.vldb.org/cidrdb/papers/2025/p13-anderson.pdf

[2] Anastasios N. Angelopoulos and Stephen Bates. 2022. A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification. *CoRR* abs/2107.07511 (2022). https://doi.org/10.48550/arXiv.2107.07511

[3] Anthropic. 2025. Introducing Claude 4. Retrieved 2026-04-16 from https://www.anthropic.com/index/claude-4

[4] Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. 2023. Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes. *Proc. VLDB Endow.* 17, 2 (2023), 92–105. https://doi.org/10.14778/3626292.3626294

[5] Amos Azaria and Tom M. Mitchell. 2023. The Internal State of an LLM Knows When It's Lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, Singapore, 967–976. https://doi.org/10.18653/v1/2023.findings-emnlp.68

[6] Muhammad Imam Luthfi Balaka, David Alexander, Qiming Wang, Yue Gong, Adila Krisnadhi, and Raul Castro Fernandez. 2025. Pneuma: Leveraging LLMs for Tabular Data Representation and Retrieval in an End-to-End System. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 200:1–200:28. https://doi.org/10.1145/3725337

[7] Rina Foygel Barber, Emmanuel J Candes, Aaditya Ramdas, and Ryan J Tibshirani. 2023. Conformal prediction beyond exchangeability. *The Annals of Statistics* 51, 2 (2023), 816–845. https://doi.org/10.1214/23-AOS2276

[8] Yonatan Belinkov and James Glass. 2019. Analysis Methods in Neural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics* 7 (2019), 49–72.

[9] Asim Biswal, Siddharth Jha, Carlos Guestrin, Matei Zaharia, Joseph E. Gonzalez, Amog Kamsetty, Shu Liu, and Liana Patel. 2025. Text2SQL is Not Enough: Unifying AI and Databases with TAG. In *15th Conference on Innovative Data Systems Research (CIDR 2025)*. CIDR, Amsterdam, The Netherlands. https://vldb.org/cidrdb/2025/text2sql-is-not-enough-unifying-ai-and-databases-with-tag.html

[10] Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2023. Discovering Latent Knowledge in Language Models Without Supervision. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*. OpenReview.net, Kigali, Rwanda. https://openreview.net/forum?id=ETKGuby0hcs

[11] Daren Chao, Kaiwen Chen, Naiqing Guan, and Nick Koudas. 2025. Relational Deep Dive: Error-Aware Queries Over Unstructured Data (Technical Report). Retrieved 2026-04-16 from https://www.cs.toronto.edu/~drchao/papers/2025_ReDD_TechReport.pdf

[12] Kaiwen Chen, Yueting Chen, Nick Koudas, and Xiaohui Yu. 2025. Reliable Text-to-SQL with Adaptive Abstention. *Proc. ACM Manag. Data* 3, 1, Article 69 (2025), 30 pages. https://doi.org/10.1145/3709719

[13] Hanjun Dai, Bethany Wang, Xingchen Wan, Bo Dai, Sherry Yang, Azade Nova, Pengcheng Yin, Phitchaya Mangpo Phothilimthana, Charles Sutton, and Dale Schuurmans. 2024. UQE: A Query Engine for Unstructured Databases. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*. Curran Associates, Inc., Vancouver, BC, Canada, 29807–29838. http://papers.nips.cc/paper_files/paper/2024/hash/34b3a40ec9752c1ae48fe85fef8fe8dc-Abstract-Conference.html

[14] Google. 2025. Gemini Deep Research. Retrieved 2026-04-16 from https://gemini.google/overview/deep-research

[15] Yuxiang Guo, Zhuoran Du, Nan Tang, Kezheng Tang, Congcong Ge, and Yunjun Gao. 2026. DTBench: A Synthetic Benchmark for Document-to-Table Extraction. *CoRR* abs/2602.13812 (2026). https://doi.org/10.48550/arXiv.2602.13812

[16] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language?. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 3651–3657. https://doi.org/10.18653/v1/P19-1356

[17] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, et al. 2023. Mistral 7B. *CoRR* abs/2310.06825 (2023). https://doi.org/10.48550/arXiv.2310.06825

[18] Yizhu Jiao, Sha Li, Sizhe Zhou, Heng Ji, and Jiawei Han. 2024. Text2DB: Integration-Aware Information Extraction with Large Language Model Agents. In *Findings of the Association for Computational Linguistics: ACL 2024*. Association for Computational Linguistics, Bangkok, Thailand, 185–205. https://doi.org/10.18653/v1/2024.findings-acl.12

[19] Nicola Jones. 2025. OpenAI's 'deep research' tool: is it useful for scientists? Nature News. https://doi.org/10.1038/d41586-025-00377-9

[20] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language Models (Mostly) Know What They Know. *CoRR* abs/2207.05221 (2022). https://doi.org/10.48550/arXiv.2207.05221

[21] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. DSPy: Compiling Declarative Language Model Calls into State-of-the-Art Pipelines. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*. OpenReview.net, Vienna, Austria. https://openreview.net/forum?id=sY5N0zY5Od

[22] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*. Curran Associates, Inc., Virtual. https://papers.nips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[23] Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023. RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. AAAI Press, Washington, DC, USA, 13067–13075. https://doi.org/10.1609/AAAI.V37I11.26535

[24] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chen-Chuan Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can LLMs Already Serve as a Database Interface? A Big Bench for Large-Scale Database Grounded Text-to-SQLs. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*. Curran Associates, Inc., New Orleans, LA, USA, 42330–42357. http://papers.nips.cc/paper_files/paper/2023/hash/83fc8fab1710363050bbd1d4b8cc0021-Abstract-Datasets_and_Benchmarks.html

[25] Teng Lin, Yuyu Luo, and Nan Tang. 2026. AnnoRetrieve: Efficient Structured Retrieval for Unstructured Document Analysis. *CoRR* abs/2604.02690 (2026). https://doi.org/10.48550/arXiv.2604.02690

[26] Teng Lin, Yizhang Zhu, Yuyu Luo, and Nan Tang. 2025. SRAG: Structured Retrieval-Augmented Generation for Multi-Entity Question Answering over Wikipedia Graph. *CoRR* abs/2503.01346 (2025). https://doi.org/10.48550/arXiv.2503.01346

[27] Teng Lin, Yizhang Zhu, Zhengxuan Zhang, Yuyu Luo, and Nan Tang. 2026. DocSage: An Information Structuring Agent for Multi-Doc Multi-Entity Question Answering. *CoRR* abs/2603.11798 (2026). https://doi.org/10.48550/arXiv.2603.11798

[28] Yiming Lin, Mawil Hasan, Rohan Kosalge, Alvin Cheung, and Aditya G. Parameswaran. 2025. TWIX: Automatically Reconstructing Structured Data from Templatized Documents. *CoRR* abs/2501.06659 (2025). https://doi.org/10.48550/arXiv.2501.06659

[29] Yiming Lin, Madelon Hulsebos, Ruiying Ma, Shreya Shankar, Sepanta Zeighami, Aditya G. Parameswaran, and Eugene Wu. 2025. Querying Templatized Document Collections with Large Language Models. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE Computer Society, Hong Kong, China, 2422–2435. https://doi.org/10.1109/ICDE65448.2025.00183

[30] Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baile Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, Rana Shahout, et al. 2025. Palimpzest: Optimizing AI-Powered Analytics with Declarative Query Processing. In *15th Conference on Innovative Data Systems Research (CIDR 2025)*. CIDR, Amsterdam, The Netherlands, 1–7. https://vldb.org/cidrdb/2025/palimpzest-optimizing-ai-powered-analytics-with-declarative-query-processing.html

[31] Chunwei Liu, Gerardo Vitagliano, Brandon Rose, Matthew Printz, David Andrew Samson, and Michael Cafarella. 2025. PalimpChat: Declarative and Interactive AI Analytics. In *Companion of the 2025 International Conference on Management of Data, SIGMOD/PODS 2025, Berlin, Germany, June 22-27, 2025*. ACM, Berlin, Germany, 183–186. https://doi.org/10.1145/3722212.3725122

[32] Seiji Maekawa, Hayate Iso, and Nikita Bhutani. 2025. Holistic Reasoning with Long-Context LMs: A Benchmark for Database Operations on Massive Textual Data. In *The Thirteenth International Conference on Learning Representations (ICLR 2025)*. OpenReview.net, Singapore. https://openreview.net/forum?id=5LXcoDtNyq

[33] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK.

[34] Vasilios Mavroudis. 2024. LangChain v0.3. Preprints.org. https://doi.org/10.20944/preprints202411.0566.v1

[35] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and Editing Factual Associations in GPT. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*. Curran Associates, Inc., New Orleans, LA, USA, 17359–17372.

[36] Jerzy Neyman and Egon Sharpe Pearson. 1933. IX. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 231 (1933), 289–337.

[37] OpenAI. 2025. Introducing Deep Research. Retrieved 2026-04-16 from https://openai.com/index/introducing-deep-research

[38] OpenAI. 2025. Introducing GPT-5. Retrieved 2026-04-16 from https://openai.com/index/introducing-gpt-5

[39] OpenAI. 2025. Introducing OpenAI o3 and o4-mini. Retrieved 2026-04-16 from https://openai.com/index/introducing-o3-and-o4-mini

[40] Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. 2025. LLMs Know More Than They Show: On the Intrinsic Representation of LLM Hallucinations. In *The Thirteenth International Conference on Learning Representations (ICLR 2025)*. OpenReview.net, Singapore. https://openreview.net/forum?id=KRnsX5Em3W

[41] Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. 2025. Semantic Operators and Their Optimization: Towards AI-Based Data Analytics with Accuracy Guarantees. *Proc. VLDB Endow.* 18 (2025), 4171–4182. https://doi.org/10.14778/3749646.3749685

[42] Jonathan Roberts, Kai Han, and Samuel Albanie. 2025. Needle Threading: Can LLMs Follow Threads Through Near-Million-Scale Haystacks?. In *The Thirteenth International Conference on Learning Representations (ICLR 2025)*. OpenReview.net, Singapore. https://openreview.net/forum?id=wHLMsM1SrP

[43] Dario Satriani, Enzo Veltri, Donatello Santoro, Sara Rosato, Simone Varriale, and Paolo Papotti. 2025. Logical and Physical Optimizations for SQL Query Execution over Large Language Models. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 181:1–181:28. https://doi.org/10.1145/3725411

[44] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 9895–9901. https://doi.org/10.18653/v1/2021.emnlp-main.779

[45] Glenn Shafer and Vladimir Vovk. 2008. A Tutorial on Conformal Prediction. *Journal of Machine Learning Research* 9, 12 (2008), 371–421.

[46] Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. 2025. DocETL: Agentic Query Rewriting and Evaluation for Complex Document Processing. *Proc. VLDB Endow.* 18, 9 (2025), 3035–3048. https://doi.org/10.14778/3746405.3746426

[47] Qwen Team. 2025. Qwen3 Technical Report. *CoRR* abs/2505.09388 (2025). https://doi.org/10.48550/arXiv.2505.09388

[48] Ryan J. Tibshirani, Rina Foygel Barber, Emmanuel J. Candès, and Aaditya Ramdas. 2019. Conformal Prediction Under Covariate Shift. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, Vol. 32. Curran Associates, Inc., Vancouver, BC, Canada, 2526–2536. https://proceedings.neurips.cc/paper/2019/hash/8fb21ee7a2207526da55a679f0332de2-Abstract.html

[49] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Siddharth Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR* abs/2307.09288 (2023). https://doi.org/10.48550/arXiv.2307.09288

[50] Matthias Urban and Carsten Binnig. 2024. Demonstrating CAESURA: Language Models as Multi-Modal Query Planners. In *Companion of the 2024 International Conference on Management of Data*. Association for Computing Machinery, New York, NY, USA, 472–475. https://doi.org/10.1145/3626246.3654732

[51] Matthias Urban and Carsten Binnig. 2024. ELEET: Efficient Learned Query Execution over Text and Tables. *Proc. VLDB Endow.* 17, 13 (2024), 4867–4880. https://doi.org/10.14778/3704965.3704989

[52] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. 2005. *Algorithmic learning in a random world*. Vol. 29. Springer, New York, NY, USA.

[53] Ante Wang, Linfeng Song, Ye Tian, Baolin Peng, Lifeng Jin, Haitao Mi, Jinsong Su, and Dong Yu. 2024. Self-Consistency Boosts Calibration for Math Reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. Association for Computational Linguistics, Miami, Florida, USA, 6023–6029. https://doi.org/10.18653/v1/2024.findings-emnlp.349

[54] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 7567–7578. https://doi.org/10.18653/v1/2020.acl-main.677

[55] Jiayi Wang and Jianhua Feng. 2025. Unify: An Unstructured Data Analytics System. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE Computer Society, Hong Kong, China, 4662–4674. https://doi.org/10.1109/ICDE65448.2025.00374

[56] Jiayi Wang, Guoliang Li, and Jianhua Feng. 2025. iDataLake: An LLM-Powered Analytics System on Data Lakes. *IEEE Data Eng. Bull.* 49, 1 (2025), 57–69. Retrieved 2026-04-16 from http://sites.computer.org/debull/A25mar/p57.pdf

[57] Eric Wu, Kevin Wu, Roxana Daneshjou, David Ouyang, Daniel E Ho, and James Zou. 2021. How Medical AI Devices Are Evaluated: Limitations and Recommendations from an Analysis of FDA Approvals. *Nature Medicine* 27, 4 (2021), 582–584. https://doi.org/10.1038/s41591-021-01312-x

[58] xAI. 2025. Grok 3 Beta — The Age of Reasoning Agents. Retrieved 2026-04-16 from https://x.ai/blog/grok-3

[59] Fatih Furkan Yilmaz and Reinhard Heckel. 2023. Test-time Recalibration of Conformal Predictors Under Distribution Shift Based on Unlabeled Examples. *CoRR* abs/2210.04166 (2023). https://doi.org/10.48550/arXiv.2210.04166

[60] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 3911–3921. https://doi.org/10.18653/v1/D18-1425

[61] Mozhi Zhang, Mianqiu Huang, Rundong Shi, Linsen Guo, Chong Peng, Peng Yan, Yaqian Zhou, and Xipeng Qiu. 2024. Calibrating the Confidence of Large Language Models by Eliciting Fidelity. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Miami, Florida, USA, 2959–2979. https://doi.org/10.18653/v1/2024.emnlp-main.173