Relational Deep Dive: Error-Aware Queries Over Unstructured Data

Daren Chao
University of Toronto

Kaiwen Chen
University of Toronto

Naiqing Guan
University of Toronto

Nick Koudas
University of Toronto

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.

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) [15], 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 [21, 30]. State-of-the-art approaches for text-to-SQL [10] or traditional information extraction techniques [22], rely on predefined schemas or explicit semantic information, which is generally unavailable or undefined in unstructured corpora.

In extracting value out of unstructured data, frameworks such as *DeepResearch* [12, 14, 25] and *DeepSearch* [41] are gaining popularity. These are agentic frameworks that scan web pages (or documents) in a query-driven manner, producing detailed summaries and analysis in response to a *specific* user query. Motivated by such frameworks, we seek to analyze semantic information from unstructured documents and extract *query-specific structured data*. Moreover, given that our focus is on running analytical queries on extracted structured data, we are interested in providing *error-aware query processing* with controllable accuracy—such as the ability to specify query-time error bounds or adjust accuracy-cost trade-offs.

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*.

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

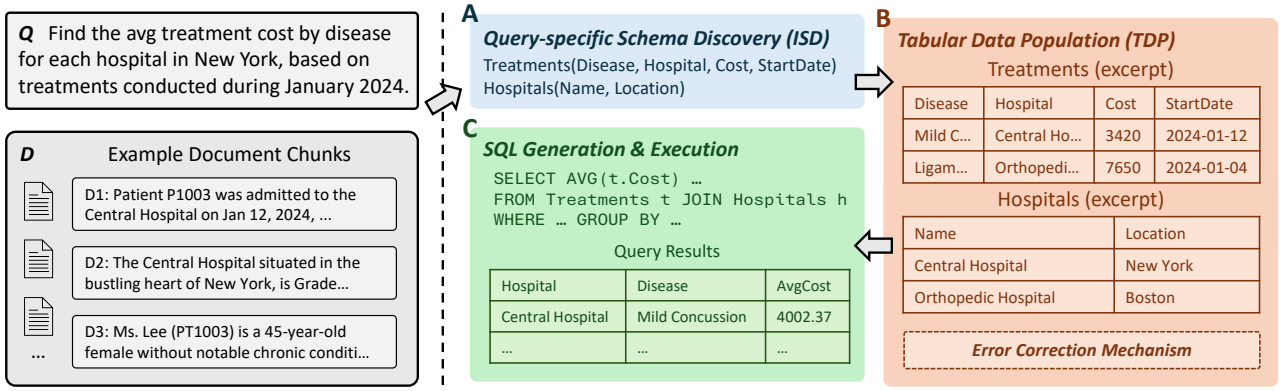

Figure 1: Overview of the query processing pipeline in REDD. The left side of the dashed line shows the raw input, consisting of a natural language query and a collection of unstructured document chunks. The right side illustrates the core system workflow of REDD, comprising: (A) schema discovery; (B) data population; and (C) SQL query generation and execution (not the focus of this work). Within the data population component (B), an error correction mechanism is integrated to automatically detect and rectify low-confidence extractions, enabling controllable accuracy.

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 §4.

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 framework 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 structured query execution over unstructured text. The framework bridges the gap between raw documents and structured query processing by dynamically discovering schemas, populating tables, and correcting extraction errors. REDD achieves this through a cohesive pipeline comprising Iterative Schema Discovery (ISD), Tabular Data Population (TDP), and error detection and control, enabling accurate query answering with structured representations even in the absence of predefined schemas or annotations.
- We develop a two-stage schema discovery pipeline that constructs minimal, joinable tables tailored to each query. Empirical results demonstrate that this two-phase approach yields more accurate and query-complete schemas compared to single-phase alternatives.
- We introduce SCAPE (Spatial Conformal Activation Partitioning for Errors) a statistically calibrated method that guarantees error coverage (Theorem 4.1) while asymptotically being optimal in minimizing human correction costs (Theorem 4.2) by partitioning high-dimensional non-conformity scores (Algorithm 1). Moreover, we introduce SCAPE-HYB, a hybrid approach that integrates conflict-aware signals with SCAPE, maintaining its well calibrated properties (Theorem 4.1) while enabling flexible trade-offs between error detection recall and correction costs with provable guarantees (Theorem 4.3).
- We validate REDD on several real-world datasets, showing that it scales to large unstructured document collections and reduces query error rates from up to 30% to below 1%. The results underscore REDD's effectiveness in transforming unstructured text into query-ready structured data with provable guarantees.

In §2 we introduce the core components of the REDD framework. §3-4 present our error management module for data extraction, followed by §5 presenting our schema discovery methodology. §6 reports experimental results across multiple datasets[1]. §7 reviews related work and we present our closing remarks in §8.

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

[1]Our code and datasets are available at: https://github.com/daren996/REDD.

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[2] 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 [31] and Palimpzest [19]), the schema may be known or provided (see §7 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 §3-4). The entire pipeline is driven by the input query q, while also mining latent semantics across chunks to construct accurate, structured outputs. Given a query specific schema S_D^q at hand (a process described in §5) we next detail the TDP phase (§3-4) followed by ISD.

3 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[3].

- Table Resolver $\mathcal{X}_T$. For every chunk d_k, the resolver selects the most semantically compatible table $T_k \in S_D^q$:
$$T_k = \mathcal{X}_T\left(d_k, S_D^q\right), \quad \text{for } k = 1..n \tag{1}$$
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 $a_i \in T_k$:
$$v_{k,i} = \mathcal{X}_A\left(d_k, a_i, T_k\right), \quad \text{for } k = 1..n,\ a_i \in T_k \tag{2}$$
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 §6.2). 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.

3.1 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. 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 (1)-(2)) 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 [27] and Claude-4 [3]) independently extracts and evaluates the same tuples. The correctness of a

[2]For brevity and to ease presentation, in the remainder of this section we assume that each chunk yields one row for illustration; extending to other scenarios such as one-to-many and many-to-one mappings is straightforward (§3) and imposes no changes to the downstream modules. We empirically validate these along with their trade-offs, in §6.4.1 (one-to-many) and §6.2.1 (many-to-one).
[3]Concrete prompt templates and implementation details are provided in the accompanying technical report [9].

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.

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:

$$y_t = \begin{cases} 1 & \text{if } t \text{ matches majority of } \mathcal{M} \text{ or human confirms,} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

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 §6.2 by varying its size and measuring its impact on extraction accuracy. These classifiers (§3.2) 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 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 §4.

3.2 Error Detection via Latent Representations

A cornerstone of our approach is a lightweight binary classifier that determines whether each extracted attribute value is correct, using the LLM's own hidden representations. Recent studies have demonstrated that LLMs' internal states encode rich information about the truthfulness of their outputs [28], which we exploit to identify potential extraction errors in ReDD.

Leveraging LLM Hidden States. As described in §3, for each document chunk d_k, the TDP stage first assigns a target table T_k via the table resolver $\mathcal{X}_T$ (Equation (1)), and then extracts attribute values $v_{k,i}$ using the attribute extractor $\mathcal{X}_A$ (Equation (2)). 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:

$$\begin{aligned} h^{(l)}_{k,\text{table}} &= \texttt{concat}\left(\operatorname*{mean}_j h^{(l)}(w^j_{k,\text{table}}), \max_j h^{(l)}(w^j_{k,\text{table}})\right), \\ h^{(l)}_{k,\text{attr-}i} &= \texttt{concat}\left(\operatorname*{mean}_j h^{(l)}(w^j_{k,\text{attr-}i}), \max_j h^{(l)}(w^j_{k,\text{attr-}i})\right), \end{aligned} \tag{4}$$

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 §3.3), 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[4], as recent studies have shown that certain layers encode richer and more informative signals than others [13, 23]. The aggregated hidden representations obtained using Equation (4) 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. The classifiers are trained using $\mathcal{D}_{\text{cls}}$.

Example. Consider the value "Central Hospital" extracted for the attribute *Name* in the table *Hospitals*. The LLM may tokenize this value into multiple subword tokens as the output, such as "Central" $w^1_{k,\text{attr-}i}$ and "Hospital" $w^2_{k,\text{attr-}i}$. For each LLM layer l, we obtain the corresponding hidden states $h^{(l)}(w^1_{k,\text{attr-}i})$ and $h^{(l)}(w^2_{k,\text{attr-}i})$ for all tokens in the value. We then compute both the mean and max over these token representations and concatenate the results to obtain a fixed-length vector $h^{(l)}_{k,\text{attr-}i}$ following Equation (4). The resulting vectors $h^{(l)}_{k,\text{attr-}i} \mid l\in\mathcal{L}$ are used as input to the per-layer binary classifiers that estimate whether the extracted value is erroneous.

3.3 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[5] $f^{(l)}:\mathbb{R}^{2d}\to[0,1]$ to predict if the

[4]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.
[5]Each classifier is a multilayer perceptron (MLP) with a sigmoid output.

extraction associated with hidden state $h^{(l)}_{k,\circ}$ (as in §3.2) is erroneous:

$$\pi^{(l)}_{k,\circ} = f^{(l)}\left(h^{(l)}_{k,\circ}\right), \tag{5}$$

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:

$$\hat{y}^{(l)}_{k,\circ} = \mathbf{1}\left[\pi^{(l)}_{k,\circ} > \theta\right]. \tag{6}$$

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:

$$\hat{y}^{\text{MV}}_{k,\circ} = \mathbf{1}\left[\sum_{l\in\mathcal{L}} \hat{y}^{(l)}_{k,\circ} > \frac{|\mathcal{L}|}{2}\right]. \tag{7}$$

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. 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:

$$\kappa = \sum_{l\in\mathcal{L}} \mathbf{1}\left\{\hat{y}^{(l)}_{k,\circ} \neq \hat{y}^{\text{MV}}_{k,\circ}\right\}, \tag{8}$$

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

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

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.

4 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 proposals, SCAPE and a hybrid approach SCAPE-Hyb. Instead of relying on binary classifier predictions and aggregations thereof to quantify the accuracy of a prediction, our proposals leverage the continuous activation probabilities of classifiers jointly to quantify uncertainty more precisely.

- SCAPE (§4.1): 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 (§4.2): 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.

4.1 SCAPE: Spatial Conformal Activation Partitioning for Errors

The SCAPE framework introduces a novel approach to uncertainty quantification by leveraging a high-dimensional non-conformity score space [6, 37], contrasting with the previously introduced methods that rely on independent thresholds for binary classifiers [2, 7, 32]. Instead of treating each classifier's decision boundary in isolation, this technique constructs a multi-dimensional (spatial) score by combining outputs from classifiers trained on different layers. The key innovation lies in partitioning this high-dimensional space into adaptive cells centered around calibration data, which are then ranked by the empirical ratio of correct to incorrect labels observed among the calibration samples contained in each cell.

By selecting regions with low concentrations of incorrect labels (where the classifiers historically perform well), the method generates more efficient and precise prediction sets while maintaining guaranteed coverage. This approach avoids rigid thresholding or weighted aggregation, instead exploiting the richer geometric structure of the multi-dimensional space to better separate true from false predictions, yielding smaller and more informative uncertainty sets. The framework's flexibility allows it to outperform traditional conformal prediction, particularly in scenarios where classifiers provide complementary information across input or output regions. SCAPE enables coverage (or recall) control of error detection under mild distributional assumptions, ensuring a specified proportion of true errors is identified with high probability.

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 §3.2), 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 (5)). 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:

$$\mathbf{s}(c) = \left[s_l(c) \right]_{l=1}^{\mathcal{L}}, \quad \text{where } s_l(c) = \begin{cases} 1 - \pi^{(l)}_{k,\circ} & \text{if } c = 1, \\ \pi^{(l)}_{k,\circ} & \text{if } c = 0. \end{cases} \tag{9}$$

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 labeled dataset $\mathcal{D}_{\text{cal-base}} = \{(x_i, y_i)\}_{i=1}^{N_{\text{cal-base}}}$, constructed using the same procedure as in §3.1, where each x_i denotes an extracted item and $y_i \in \{0,1\}$ indicates whether the extraction is erroneous (1) or correct (0).[6] We randomly split this dataset into two disjoint subsets, $\mathcal{D}_{\text{cell}}$ and $\mathcal{D}_{\text{re-cal}}$, which serve distinct purposes in the calibration procedure.

- $\mathcal{D}_{\text{cell}} = \{(x_i, y_i)\}_{i=1}^{N_{\text{cell}}}$ used to construct cells in the non-conformity score space by applying k-means clustering to the score vectors $\mathbf{s}(\hat{y}_i)_{i=1}^{N_{\text{cell}}}$, producing K clusters. Each cluster defines a cell, yielding a partition of the score space into non-overlapping regions $C_1, \ldots, C_K \subset \mathbb{R}^{|\mathcal{L}|}$, where any new score vector can be assigned to one of the cells by finding its nearest cluster centroid in Euclidean space.
- $\mathcal{D}_{\text{re-cal}} = \{(x_j, y_j)\}_{j=1}^{N_{\text{re-cal}}}$ used to select cells for coverage.

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):

$$\rho_m = \frac{\sum_{c \in \{0,1\}} |(x_j, c) \in \mathcal{D}_{\text{re-cal}} : \mathbf{s}(1-c) \in C_m|}{\sum_{c \in \{0,1\}} |(x_j, c) \in \mathcal{D}_{\text{re-cal}} : \mathbf{s}(c) \in C_m|}. \tag{10}$$

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 $C_\alpha \subset \mathbb{R}^{|\mathcal{L}|}$:

$$C_\alpha = \bigcup_{j=1}^{\eta^*} C_{(j)}, \quad \text{where } \eta^* = \min\left\{ \eta \in 1..K \mid \left| \{(x_j, y_j) \in \mathcal{D}_{\text{re-cal}} : \mathbf{s}(y_j) \in C_\alpha\} \right| \geq \lceil (1-\alpha)(N_{\text{re-cal}} + 1) \rceil \right\} \tag{11}$$

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

Test-Time Inference. At test time, for each new extraction with hidden representations $\{h^{(l)}_{k,\circ}\}_{l \in \mathcal{L}}$ (as per §3.2), we compute the non-conformity vectors $\mathbf{s}(y)$ for both possible labels $y \in \{0,1\}$ (via Equation (9)), and define the conformal prediction set as:

$$\hat{y}^{\text{SCAPE}}_{k,\circ} = \{ y \in \{0,1\} \mid \mathbf{s}(y) \in C_\alpha \}. \tag{12}$$

[6] $\mathcal{D}_{\text{cal-base}}$ is a small, user-curated labeled dataset, generated by applying the LLM prompts used in TDP stage (as Equations (1)-(2)) to a small number of document chunks. The dataset size is varied in the experiments to study its impact on overall accuracy in §6.2.

By evaluating both candidate labels, we construct a prediction set that includes multiple labels only when necessary to satisfy the desired α-coverage for error detection, while keeping the set as small as possible to reduce correction cost. If $\hat{y}^{\text{SCAPE}}_{k,\circ} = \{0\}$, we accept the extraction as correct. Otherwise, if the prediction set contains 1 (i.e., $\{1\}$ or $\{0,1\}$), the extraction is flagged for potential error and triggers a correction step, typically by routing the value for human verification and correction.

SCAPE is presented as Algorithm 1, where line 1 corresponds to the computation of non-conformity vectors as defined in Equation (9), lines 2-3 implement the clustering and ranking procedure described in Equation (10), line 4 selects cells according to the coverage constraint in Equation (11), and line 5 defines the prediction set $\hat{y}^{\text{SCAPE}}_{k,\circ}$ for each new extraction as Equation (12).

Algorithm 1: Spatial Conformal Activation Partitioning for Errors (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 $\hat{y}^{\text{SCAPE}}_{k,\circ} \subseteq \{0,1\}$;

1: Compute non-conformity vectors using Eq. (9), for all $(x_i, y_i) \in \mathcal{D}_{\text{cal-base}}$.
2: Partition non-conformity space using k-means clustering on $\mathcal{D}_{\text{cell}}$ to obtain cells $C_1, \ldots, C_K$.
3: Rank cells in ascending order of false-to-true ratio ρ_m computed using Eq. (10): $C_{(1)}, C_{(2)}, \ldots, C_{(K)}$.
4: Select top-ranked cells C_α through Eq. (11).
5: For new extraction: $\hat{y}^{\text{SCAPE}}_{k,\circ} = \{ y \in \{0,1\} : \mathbf{s}(y) \in C_\alpha \}$.

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

Theorem 4.1 (Coverage Guarantee under Exchangeability). *Under the assumption that calibration and test examples are exchangeable, the conformal prediction set determined by SCAPE satisfies:*

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

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

The proof is available in the accompanying technical report [9].

Set Size Optimality. Theorem 4.2 below establishes that SCAPE achieves *asymptotic optimality* in prediction set size under a mixture model assumption. This guarantees that, as the calibration data grows ($|\mathcal{D}_{\text{cal-base}}| \to \infty$), the method:

- Minimizes the expected number of extractions flagged for human review ($\mathbb{E}[|\hat{y}^{\text{SCAPE}}_{k,\circ}|]$),
- While maintaining the desired error coverage ($1 - \alpha$).

The proof, which leverages the Neyman-Pearson lemma [24, 32] to show that ranking cells by false-to-true ratio ρ_m (as Equation (10)) is equivalent to optimizing the likelihood ratio $\Lambda(\mathbf{s})$, is provided in the technical report [9].

Theorem 4.2 (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* $\mathbb{E}[|\hat{y}^{SCAPE}_{k,\circ}|]$ *subject to coverage* $\geq 1 - \alpha$.

Limitations. Theorem 4.2 establishes optimality under the assumption of a monotonic likelihood ratio between $\Lambda(\mathbf{s})$ and ρ. In practice, this requires the classifier outputs to be well-calibrated, which is

challenging as we wish to keep $|\mathcal{D}_{\text{cal-base}}|$ very small. Moreover, finite-sample effects may lead to marginal under-coverage when $|\mathcal{D}_{\text{cells}}|$ is small. Our empirical results demonstrate that extractions with high disagreement among layer-wise classifiers are more likely to be erroneous. The raw non-conformity scores in SCAPE (based on classifier probabilities) may not fully capture this disagreement. Metrics like κ—the number of layers disagreeing with the majority vote—provide an orthogonal signal that improves error detection. We thus extend SCAPE below to capture such conflict.

4.2 SCAPE-Hyb: Hybrid Method

We now introduce SCAPE-Hyb, a unified method that incorporates the inter-layer conflict signal (§3.3) into the conformal prediction framework (§4.1). 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 used in CF (Equation (8)) captures binary disagreement among classifiers, but tends to be, after probability thresholding per classifier, small or even zero; this happens even when the probability outputs of the classifiers vary significantly as they are "smoothed" by thresholding. Thus the value of κ itself is not a very informative signal. To address this, we replace κ with a more granular, 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:

$$\mathbf{s}_\lambda(c) = \Big[(1-\lambda)\cdot s_1(c), \ldots, (1-\lambda)\cdot s_{|\mathcal{L}|}(c), \lambda\cdot\Delta\Big], \tag{13}$$

where each $s_l(c)$ is the layer-wise non-conformity score defined in §4.1, and $\lambda\in[0,1]$ is a tunable parameter that adjusts the relative weight of the disagreement signal inside the conformal prediction pipeline. When λ=0, the method falls back to SCAPE (§4.1); on the other hand, when λ=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 (§3.3).

Calibration and Prediction. Following the calibration protocol described in §4.1, we compute the conflict-augmented non-conformity vectors at the given λ-weighting for all examples in the calibration dataset and partition the non-conformity space into cells. We rank the cells through the conflict-augmented false-to-true ratio ρ_m^λ and retain only the cells C_α^λ that satisfy the conformal coverage condition at level $1-\alpha$ (see Equation (11)). At test time, the conformal prediction set is defined as:

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

As before, we treat the extraction as correct if $\hat{y}_{k,\circ}^{\text{Hyb}}$=0, and trigger correction if the set includes 1 or both labels. It is easy to see that the conformal guarantee (Theorem 4.1) still holds. This is because the augmented score $\mathbf{s}_\lambda(c)$ retains exchangeability, and the cell selection process is still thresholding a well-defined statistic.

Conflict-Aware Optimality. SCAPE-Hyb extends the optimality guarantee of SCAPE (Theorem 4.2) to conflict-augmented non-conformity scores $\mathbf{s}_\lambda$. Following the same principle as SCAPE, it ranks cells in the score space by their empirical false-to-true ratio, which remains proportional to the inverse of the likelihood ratio. By selecting the smallest set of cells C_α^λ that satisfies the user-specified coverage constraint $(1-\alpha)$, SCAPE-Hyb asymptotically minimizes the expected prediction set size. Proof details are provided in [9].

Advantage of SCAPE-Hyb. As long as the conflict score provides an additional discriminatory signal—specifically, if erroneous outputs tend to have higher conflict than correct ones—the hybrid method SCAPE-Hyb can yield smaller expected prediction sets (thus reducing human correction cost in expectation), while preserving the same coverage.

Theorem 4.3 (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:*

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

A detailed formal proof of Theorem 4.3 is provided in the accompanying technical report [9]. Our experimental results (§6) empirically corroborate this theoretical advantage.

Besides, SCAPE-Hyb can be viewed as a *soft* compatible generalization of conflict filtering: instead of enforcing hard thresholds, the conflict score is smoothly embedded into the non-conformity space and calibrated within the conformal framework. In this way, the hybrid method preserves a key advantage of CF—its ability to improve recall at relatively low cost—while avoiding brittle thresholds and retaining the formal coverage guarantees of SCAPE. The continuous weighting of conflict also gives practitioners finer control over the accuracy-cost trade-off.

Summary. The parameter λ controls the relative influence of conflict: higher values give more weight to conflict, approaching CF behavior as $\lambda \to 1$; lower values emphasize the original conformal score. Low values in λ prioritize probabilistic uncertainty and are better for well-calibrated classifiers (when $|\mathcal{D}_{\text{cls}}|$ is large). High values of λ prioritize conflict, which is better when layer disagreements correlate with errors and as we empirically demonstrate in §6.2 when $|\mathcal{D}_{\text{cls}}|$ is small. The optimal setting can be selected via grid search on a validation set. In practice, intermediate values (e.g., λ=0.5) often yield a good trade-off between error detection recall and extra correction cost, as demonstrated in §6.2.

5 ITERATIVE SCHEMA DISCOVERY

The goal of this stage is to derive a relational schema S_D^q over a collection of document chunks D that contains exactly the entities (attributes, such as names and locations) and relationships required to answer the query q. Schema discovery occurs dynamically in two phases. Phase I induces a general (query-agnostic) schema S_D^{gen}, that captures all salient attributes and relationships present in the documents, independently of any specific query. Phase II then adapts this into a query-specific schema S_D^q tailored explicitly to the requirements of query q.

Phase I: General Schema Discovery. We treat schema discovery as an iterative process of reading and abstraction. It begins with an empty schema and processes the document chunks in a sequential, one-pass manner. At each step, it incrementally revises the current schema state. As new information becomes available, each step may refine earlier decisions by revising the previously constructed schema state. The final schema is the result of this sequence of incremental updates across all chunks. We next describe the structure of the schema state as maintained during this process, i.e., how the algorithm represents and updates it at each step.

The schema state is organized as a collection of relational tables. Each table corresponds to either an entity type (e.g., `Person`, `Hospital`) or a relationship (e.g., `Admission`, `Treatment`). To help the algorithm maintain and use the schema state effectively, each table is annotated with the following information.

- A canonical table name and a concise natural language description, both generated by the LLM based on the semantics of relevant document chunks and prior schema context;
- A small set of example document chunks for each table, selected by the LLM to motivate the creation of the table and provide grounding for its semantics;
- A list of attributes for each table, each annotated with a name and a usage-based explanation derived by the LLM from the context in which it appeared.

Schema updates are performed by a prompt-based function $\mathcal{S}_G$, which uses a fixed prompt template to invoke an LLM[7]. Given the current schema state S_{k-1}^{gen} and the k-th document chunk d_k, the algorithm computes the next state as:

$$S_k^{\text{gen}} = \mathcal{S}_G\left(S_{k-1}^{\text{gen}}, d_k\right), \quad \text{for } k = 1..n \tag{14}$$

Updates may involve introducing new tables, adding attributes to existing tables, or refining existing descriptions. Crucially, the current schema state remains in memory at every step, enabling the function $\mathcal{S}_G$ to leverage prior schema structure and annotations to extract additional structure information from each new chunk.

As the process continues, earlier schema elements that were ambiguous or incomplete may be clarified by later chunks, supporting limited self-correction without retroactive reconstruction. We empirically validate the effectiveness of our method in §6.3. After all chunks are processed, the final schema S_n^{gen} constitutes S_D^{gen}. It serves as a comprehensive, query-agnostic abstraction over the document collection and provides the structural basis for the next stage of query-specific schema adaptation.

Phase II: Query-Specific Schema Discovery. In the second phase, the goal is to transform the general schema S_D^{gen} into a query-specific schema S_D^q that contains only the tables and attributes necessary to answer the input query q. The process is again iterative, following the same schema structure and update pattern as in Phase I. Schema updates are now performed by a different prompt-based function $\mathcal{S}_Q$, which also uses a fixed LLM prompt[8], but incorporates the query q as an additional input to guide the refinement.

$$S_k^q = \mathcal{S}_Q\left(S_{k-1}^q, d_k, q, S_D^{\text{gen}}\right), \quad \text{for } k = 1..n \tag{15}$$

[7]Details of the prompt design and its implementation are available in the accompanying technical report [9].

[8]Details of the prompt design and its implementation are available in the accompanying technical report [9].

In this phase, $\mathcal{S}_Q$ selectively removes irrelevant schema elements, adds previously overlooked attributes, if any, and restructures attributes to precisely match the intent of query q. Such refinement includes accommodating explicit query constraints (e.g., filters, group-by keys) and implicit query requirements (e.g., join paths, derived attributes). Similar to Phase I, schema decisions can be iteratively revised if errors are introduced in earlier steps.

The result is a minimal and query-complete schema S_D^q, essential for accurate data population and effective query execution. Without this targeted refinement, critical attributes might be omitted or extraneous attributes retained, impairing query effectiveness. We empirically demonstrate in §6.3 that omitting this step leads to lower schema completeness and reduced query accuracy.

Repair Step. As an optional enhancement, a *fallback repair* step can be applied after Phase II, using a powerful LLM (e.g., GPT-5 [26]) to verify whether the extracted schema S_D^q suffices to answer the query q; if not, the system re-invokes schema discovery with additional iterations to repair and complete the schema. We empirically show in §6.3 that this repair mechanism discovers any attributes, achieving perfect attribute-level recall across all datasets.

Performance Considerations. Our schema discovery pipeline performs two sequential passes over the document collection: Phase I builds a general schema from scratch, and Phase II refines it to match the specific query intent. While techniques such as sampling, chunk clustering, or selective analysis could significantly reduce computational cost, they are orthogonal to the focus of this work. We aim to understand whether one can indeed design an *accurate* schema discovery and data extraction strategy for a specific query without hard computational considerations (e.g., token usage [19, 31]). 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.

6 EXPERIMENTAL EVALUATION

6.1 Experimental Setup

6.1.1 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 [42] and Bird [16] benchmarks. Using their original schemas and tabular data, following [21] we convert tabular rows into natural language document chunks using a state-of-the-art LLM (GPT-5 [26]). This ensures that the model used in ReDD (Qwen3-30B-A3B [34]) 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

Table 1: Evaluation Datasets Used for Assessing ReDD.

Dataset	Dataset Source	# Queries	Avg. Result Rows
Spider	Spider [42]	86	1733
Bird	Bird [16]	36	2435
Galois	Galois [31]	10	497
FDA	FDA 510(k) [40]	6	100
CUAD	CUAD [33]	15	501

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 §6.4. The prompts used to generate the documents, due to space constraints, are provided in the accompanying technical report [9]. The dataset Galois is sourced from the Fortune and Premier datasets as described in [31]. The dataset FDA is sourced from FDA 510(k) regulatory filings [4, 40], which are long-form and heterogeneous, containing narrative summaries, tabular sections, and metadata. The dataset CUAD is a legal-contract benchmark [33], whose lengthy documents make single-pass LLM processing infeasible due to context limitations [33]. 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[9].

6.1.2 Measurements. We evaluate ReDD in two stages: schema discovery (ISD) and data population (TDP). 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 [33]:

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

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:

$$FPR_{pop} = \frac{FP}{FP + TN}, \quad (17)$$

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.

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.

[9]We plan to make available all artifacts associated with this paper.

Table 2: Summary of Data Population Accuracy (ACC_{pop}). All values are averages over queries in Tb. 1. ACC_{pop} is defined in Eq. (16).

Method	Spider	Bird	Galois	FDA	CUAD
EVAPORATE [4]	-	-	0.475	0.516	0.209
Palimpzest [19]	-	-	0.867	0.924	0.613
ReDD (No Correction)	0.938	0.949	0.873	0.965	0.661
ReDD (SCAPE)	0.991	0.992	0.989	0.988	0.724
ReDD (SCAPE-Hyb)	**0.993**	**0.994**	**0.989**	**0.990**	**0.983**

Table 3: Correction Overhead in Data Population. All values are averages over queries in Tb. 1. FPR_{pop} quantifies the cost of unnecessary corrections, defined in Eq. (17).

Method	Spider	Bird	Galois	FDA	CUAD
ReDD (SCAPE)	0.063	0.054	0.044	0.051	0.114
ReDD (SCAPE-Hyb)	**0.038**	**0.039**	**0.027**	**0.032**	**0.072**

6.1.3 Experimental Settings. ReDD is implemented in Python. For schema discovery, it employs GPT-5 [26], a state-of-the-art LLM. For data population, it employs Qwen3-30B-A3B [34], a leading open-source model at the 30B scale, deployed on an A100 (80GB) GPU. Unless otherwise specified, all experiments use the following default settings: the conflict weight parameter is set to λ=0.5 (see Equation (13)); the classifiers are trained with 50 entries and calibrated on 150 entries randomly sampled per query.

6.2 Results of Data Population and Correction

6.2.1 Accuracy and Correction Cost. Table 2 reports the data population accuracy (ACC_{pop}, defined in Equation 16) under different configurations proposed in this paper: (i) ReDD without Correction, which executes without attempting to detect errors (outputs the extraction directly from the LLM), (ii) SCAPE, and (iii) SCAPE-Hyb, which integrates the proposed algorithms with default parameters (α=0.15), as well as prior baselines, EVAPORATE [4] and Palimpzest [19, 20][10]. All metrics are averaged over all queries in Table 1.

Without correction, ReDD achieves reasonably high accuracy (e.g., 0.938 on Spider and 0.949 on Bird), but still produces a substantial number of error cells (e.g., 516 remaining error cells on Spider and 491 on Bird). *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 [34]), EVAPORATE performs considerably worse than other approaches, while Palimpzest achieves accuracy comparable (though slightly lower) than ReDD *without correction* (e.g., 0.867 vs. 0.873 on Galois).

By contrast, applying our correction algorithms yields substantial improvements. With α=0.15, both SCAPE and SCAPE-Hyb

[10][31] 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 [31] demonstrated inferior to Palimpzest. We do this as ccuracy is the focus of our work. We run experiments with the same configuration parameters as in [31].

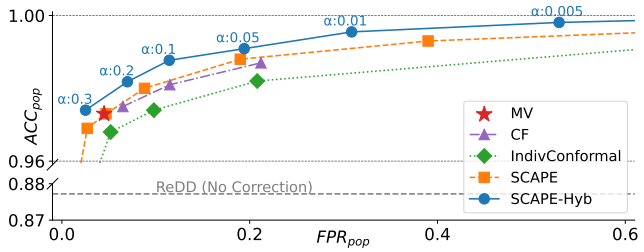

Figure 2: Trade-off between data population accuracy (ACC_{pop}) and correction cost measured by the false positive rate (FPR_{pop}). For the SCAPE-Hyb curve, labels above each point indicate the corresponding α value used to produce that accuracy–cost trade-off.

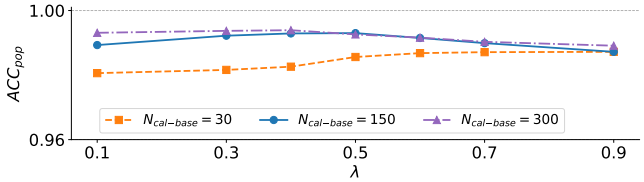

Figure 3: Data population accuracy ACC_{pop} for SCAPE-Hyb with different calibration dataset size $N_{\text{cal-base}}$ on dataset Spider, varying conflict weight λ, under FPR_{pop}=0.2.

raise accuracy to **above or near 0.99** on Spider, Bird, Galois, and FDA, with SCAPE-Hyb consistently achieving the highest 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 [34]. To address this, we adapt a map-reduce style document chunking strategy inspired by DocETL [33]. Each long contract is divided into smaller chunks (e.g., sections or pages) that can fit within the LLM's context window. Instead of assuming that one document chunk corresponds to one complete table row (as in the default setting), we allow multiple chunks (from the same legal contract) to collectively provide the attribute values required for a single row. Attribute values are first extracted independently from each chunk, and then consolidated into a complete row. Combined with our correction algorithm, this chunk-merge (map-reduce) pipeline enables SCAPE-Hyb to reach 0.983 accuracy on CUAD—substantially higher than SCAPE (0.724, without chunk merge) and Palimpzest (0.583).

Table 3 reports on the corresponding correction overhead in terms of false positive rate (FPR_{pop}). Here, SCAPE-Hyb consistently maintains lower false positive rates than SCAPE across all datasets (e.g., 0.038 vs. 0.063 on Spider and 0.039 vs. 0.054 on Bird), indicating fewer correctly extracted cells are unnecessarily flagged for review. The overhead remains modest overall, particularly on large-scale benchmarks where SCAPE-Hyb achieves both higher accuracy and lower correction cost. These results demonstrate the effectiveness of our correction module in improving extraction quality while incurring minimal verification overhead.

6.2.2 Ablation: Accuracy-Cost Tradeoff. We conduct ablation experiments to compare SCAPE-Hyb against several error detection methods proposed in this paper: MV (§3.3), CF (§3.3), and SCAPE. We also evaluate an alternative method that calibrates each base classifier independently using conformal prediction [2], followed by majority voting over the calibrated outputs (denoted as *IndivConformal*). The results are presented in Figure 2, which plots data population accuracy (ACC_{pop}) against unnecessary correction cost measured by the false positive rate (FPR_{pop}) averaged over all datasets. The gray dashed line marks the accuracy of ReDD without correction. All methods yield notable improvements over the baseline. The MV approach, lacks tunable parameters and provides only a fixed trade-off point. The curve for IndivConformal consistently lies below others, indicating weaker performance. CF supports limited tuning but remains inferior in accuracy—its curve consistently lies below SCAPE, indicating higher correction cost for the same level of accuracy. In contrast, SCAPE and SCAPE-Hyb offer significantly better accuracy-correction cost trade-offs. SCAPE-Hyb consistently outperforms all baselines. It achieves the highest data population accuracy while incurring the lowest rate of unnecessary corrections, demonstrating the effectiveness of combining SCAPE with conflict information.

6.2.3 Effect of Coverage Threshold α. The results in Figure 2, averaged across all datasets, reveal a key trend: SCAPE-Hyb achieves high initial accuracy (≥0.974 across all datasets) even at large α, with more accuracy gains as α decreases. For small α, accuracy can reach 1, but at the cost of a higher false positive rate (FPR_{pop}).

Using the Spider dataset as an example (avg. 1,733 rows per query), setting α=0.5 yields 0.947 accuracy with 4.9% of the data reviewed (including 8 false positives). Reducing the threshold to α=0.3 improves accuracy to 0.975, with 9.2% of rows reviewed (37 false positives). At α=0.01, accuracy further increases to 0.998, but 31% of the data must be reviewed (608 false positives). As α decreases, the proposed framework becomes more conservative—fewer predictions are accepted as confident, and more data are flagged for review. This generally improves accuracy, as more uncertain predictions are reviewed and corrected. However, it also increases false positives, reflected in higher FPR_{pop}. Conversely, larger α values yield cheaper but relatively less accurate results.

The results suggest that minimal user effort (relative to output size) suffices for high accuracy (>0.97), while perfect accuracy (1.0) demands significantly more verification effort. This finding opens several research directions, as discussed in §8.

6.2.4 Effect of Conflict Weight λ. Figure 3 illustrates the impact of the conflict weight (λ) on data population accuracy (ACC_{pop}) for SCAPE-Hyb on dataset Spider (under FPR_{pop}=0.2). According to the blue curve under default setting calibration dataset size $N_{\text{cal-base}}$=150 the results reveal a clear trend: accuracy peaks at $\lambda \approx 0.5$ and declines when λ is either too low or too high. This indicates that an optimal balance in weighting conflicts is critical—under weighting ($\lambda \ll 0.5$) fails to leverage disagreement signals effectively, while over weighting ($\lambda \gg 0.5$) can suppress correct but less frequent outputs. These findings validate our choice of λ=0.5 as the default in SCAPE-Hyb. This trend also holds well across other datasets.

However, when the size of the calibration dataset $N_{\text{cal-base}}$ varies, the peak of the curve shifts accordingly. For instance, when $N_{\text{cal-base}}$ is relatively small (=30), SCAPE is significantly weaker than the conflict signal, resulting in the curve peaking at a larger λ, around 0.9. In contrast, when $N_{\text{cal-base}}$ is large (=300), the peak shifts leftward to a λ value around 0.3-0.4.

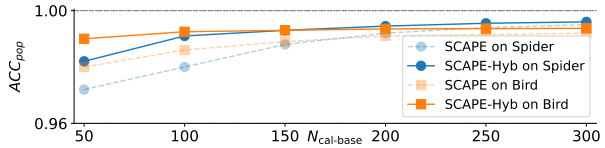

Figure 4: Data population accuracy ACC_{pop} of SCAPE and SCAPE-Hyb varying calibration dataset size $N_{\text{cal-base}}$, under FPR_{pop}=0.2.

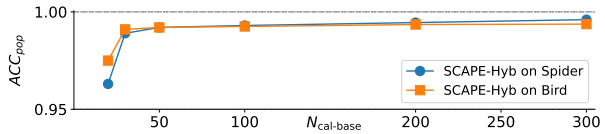

Figure 5: Data population accuracy ACC_{pop} varying training dataset size N_{cls}, under FPR_{pop}=0.2.

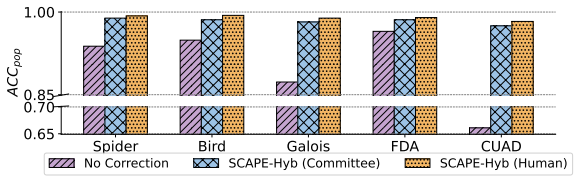

Figure 6: Data population accuracy ACC_{pop} using human-annotated vs. LLM committee-generated training data.

Table 4: Evaluation of Schema Discovery over All Datasets. *# Invalids* refers to the number of schema discoveries missing essential attributes, such that the query cannot be answered. This is across all queries in Table 1.

Method	# Invalids	Rec_{sch}^{attr}	Pre_{sch}^{attr}
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**

6.2.5 Effect of Calibration Dataset Size $N_{cal\text{-}base}$. Figure 4 illustrates the impact of calibration dataset size ($N_{\text{cal-base}}$, defined in §4.1) on data population accuracy (ACC_{pop}) for both SCAPE and SCAPE-Hyb (measured at FPR_{pop}=0.2). We report results only on Spider and Bird for brevity. Across the reported datasets, we observe a steady improvement in accuracy as $N_{\text{cal-base}}$ increases, with performance plateauing once the calibration set exceeds roughly 100-150 examples.

This finding demonstrates that near-optimal accuracy can be attained with only a modest calibration set. For example, SCAPE-Hyb achieves accuracy above 0.99 on Spider with just 150 calibration examples. These results demonstrate the data efficiency of our method, as high accuracy is attainable even with limited calibration data, an advantage in low-resource or high-cost settings.

Furthermore, SCAPE-Hyb consistently surpasses SCAPE when calibration data is scarce. By leveraging conflict information, SCAPE-Hyb compensates for the reduced supervision in small calibration sets, reinforcing its robustness in such scenarios.

6.2.6 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.

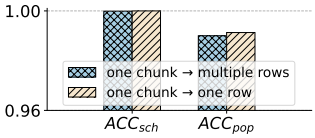

Figure 7: Schema discovery accuracy ACC_{sch} and data population accuracy ACC_{pop} under one-to-many chunk-to-table setting.

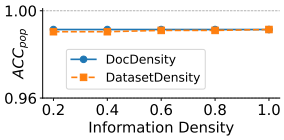

Figure 8: Data population accuracy (ACC_{pop}) of SCAPE-Hyb under diff information density levels, under FPR_{pop}=0.2.

6.2.7 Impact of Label Source: Human vs. LLM-Synthetic. 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. 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.

6.3 Results of Schema Discovery

We evaluate the schema discovery stage (ISD) over all datasets. As shown in Table 4, the final pipeline (Phase I & II + Repair, see §5) 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_{sch}^{attr}=1.000. In addition, the extracted schemas remain concise, with an average precision of Pre_{sch}^{attr}=0.956, meaning only a few redundant attributes are occasionally included—improving the efficiency of downstream data population.

6.3.1 Ablation Study of ISD Components. We conduct an ablation study to assess the contribution of each ISD component. Table 4 compares four configurations derived from Section 5:

- 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.

6.4 Additional Analyses

6.4.1 Results under One-to-Many Chunk-to-Table Mapping. We created a modified Spider dataset, merging two or three document chunks with different schema to simulate one-to-many mappings, where each chunk contributes to multiple schemas, contrasting with the default one-to-one setup. As shown in Figure 7, schema

discovery accuracy (ACC_{sch}) remains unchanged, while data population accuracy (ACC_{pop}) drops slightly by 0.13% due to increased ambiguity when mapping chunks to multiple rows. This trend holds across datasets. Since our system processes attributes iteratively, performance stays robust if schema descriptions are clear, emphasizing the need for precise schema discovery. The results confirm our method's ability to handle one-to-many mappings when schemas are well-defined.

6.4.2 Impact of Information Density. Figure 8 tests ReDD's performance under different information density levels, examining two aspects: DocDensity, which quantifies how much of a document chunk's text is relevant to the extracted data, and DatasetDensity which asseses how many document chunks in the dataset are query-relevant (both presented as fractions with 1 being the most relevant and lower values signifying increased presence of irrelevant information aiming to deter correct inference). ReDD maintains high accuracy across all scenarios. Our key findings are that DocDensity has minimal effect because ReDD processes attributes independently during TDP, avoiding confusion from irrelevant content. In addition DatasetDensity has little impact as ReDD's ISD stage performs global schema discovery, easily filtering irrelevant chunks. These results show ReDD works well with real-world data where relevant information may be sparse or unevenly distributed.

6.4.3 Token and Performance Considerations. This study demonstrates that near-perfect accuracy for unstructured data query processing is achievable. While monetary cost modeling and query performance remain critical considerations, they are beyond the scope of this work. They do represent key directions for future research however.

In all cases in our experiments total per query execution for SCAPE-Hyb (including schema discovery, data extraction, automated training and calibration data acquisition, classifier training, calibration and query execution) is under 4 hours. This time is roughly broken down as follows: approximately 40 minutes for schema discovery on average, which is not optimized currently and sequentially processes all documents two times (via GPT-5 API [26]); on average 3 hours per query to extract the data using a multi-threaded setting to parallelize the process without any inference optimizations (on a 8 A100 GPU cluster) and around 20 minutes for data correction, which is a manual process in our implementation currently. Token consumption is not currently optimized. For the Galois dataset, average token consumption per query execution is approximately 18M.

There is ample scope for optimizations. This could involve deploying techniques from the literature, such as sampling to reduce the number of documents processed for schema discovery, leveraging LLMs to automatically generate code for efficient data extraction, thereby minimizing token usage as well as using advanced LLMs for correction. In all cases, however, formally quantifying the trade-offs between these optimizations and accuracy remains a primary focus.

7 RELATED WORK

Document-to-Table Extraction. Large Language Models (LLMs) have enabled a new generation of systems that transform heterogeneous documents into structured, queryable tables. TWIX [17] extracts structured data from templatized documents by first inferring their underlying visual template, while ZENDB [18] extends this line of work by supporting ad-hoc SQL queries over collections of such documents. DocETL [33] introduces abstractions for corpus splitting and declarative pipeline specification. It's scope is limited to queries not involving aggregation or cross-document reasoning however, without the ability of structured data extraction. As such the focus is orthogonal to the problem setting of our work. Pneuma [5] focuses specifically on table selection from large document collections, a task orthogonal to the end-to-end extraction and correction problem studied in this work.

Declarative Pipelines and Semantic Operators. A complementary line of work has investigated declarative abstractions for semantic data processing. Patel et al. [29] introduce semantic operators, extending relational algebra with natural-language specifications for AI-driven transformations. Dai et al. [11] present UQE, a universal query engine that leverages a dialect of SQL (UQL) for flexible analytics on unstructured data. Palimpzest [19, 20] generalizes these ideas to optimize semantic analytics applications (SAPPs), which interleave AI reasoning with traditional relational processing. These efforts highlight the growing role of declarative models in unstructured data management, though they do not directly address accurate table population from free-form text.

Query Execution and Optimization. Another set of systems focuses on query execution over unstructured or multimodal data. ELEET [36] introduces learned multi-modal operators that treat text as a first-class data type alongside tables. TAG [8] unifies Text2SQL and retrieval-augmented generation (RAG) through table-augmented generation. [1] introduces infrastructure for unstructured data analytics. Unify [38] leverages LLMs to automatically generate and optimize query plans over unstructured data, while CAESURA [35] develops a multi-modal query planner capable of producing complex execution pipelines. [39] is a novel LLM-powered analytics system designed to handle data analytics queries over multi-modal data lakes by taking natural language queries as input, orchestrating a pipeline, and outputting results. These systems primarily focus on query planning and optimization, while our work is the first to propose a formal framework that quantifies query execution accuracy over unstructured data with guarantees.

8 CONCLUSION AND FUTURE WORK

This paper introduces ReDD, a novel framework for executing error-aware analytical queries over unstructured data. ReDD features a two-stage pipeline that first discovers a query-specific schema and then populates relational tables. A key contribution is the introduction of SCAPE and SCAPE-Hyb, statistically calibrated methods for error detection that provide coverage guarantees. Experimental results demonstrate that ReDD significantly reduces data extraction errors from as high as 30% to less than 1%, while ensuring high schema completeness with 100% recall. The core reliability framework presented raises numerous promising avenues for future work, ranging from incorporating ReDD to other unstructured data query processing frameworks [35], to designing novel query processing techniques to enhance performance, optimizing for monetary cost, and mitigating bounded errors in extraction results.

REFERENCES

[1] Eric Anderson, Jonathan Fritz, Austin Lee, Bohou Li, Mark Lindblad, Henry Lindeman, Alex Meyer, Parth Parmar, Tanvi Ranade, Mehul A. Shah, Benjamin Sowell, Dan Tecuci, Vinayak Thapliyal, and Matt Welsh. 2024. The Design of an LLM-powered Unstructured Analytics System. arXiv:2409.00847 [cs.DB] https://arxiv.org/abs/2409.00847

[2] Anastasios N. Angelopoulos and Stephen Bates. 2022. A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification. arXiv:2107.07511 [cs.LG] https://arxiv.org/abs/2107.07511

[3] Anthropic. 2025. Introducing Claude 4. 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. *arXiv preprint arXiv:2304.09433* (2023). https://arxiv.org/abs/2304.09433

[5] 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), 1–28.

[6] 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.

[7] Rina Foygel Barber, Emmanuel J. Candes, Aaditya Ramdas, and Ryan J. Tibshirani. 2023. Conformal prediction beyond exchangeability. arXiv:2202.13415 [stat.ME] https://arxiv.org/abs/2202.13415

[8] Asim Biswal, Liana Patel, Siddarth Jha, Amog Kamsetty, Shu Liu, Joseph E. Gonzalez, Carlos Guestrin, and Matei Zaharia. 2024. Text2SQL is Not Enough: Unifying AI and Databases with TAG. arXiv:2408.14717 [cs.DB] https://arxiv.org/abs/2408.14717

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

[10] 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 (Feb. 2025), 30 pages. doi:10.1145/3709719

[11] Hanjun Dai, Bethany Yixin 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. arXiv:2407.09522 [cs.DB] https://arxiv.org/abs/2407.09522

[12] Google. 2025. Gemini Deep Research. https://gemini.google/overview/deep-research.

[13] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language?. In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

[14] Nicola Jones. 2025. OpenAI's 'deep research' tool: is it useful for scientists? *Nature* (2025). doi:10.1038/d41586-025-00377-9

[15] 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. [n. d.]. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. https://arxiv.org/abs/2005.11401

[16] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems* 36 (2023), 42330–42357.

[17] Yiming Lin and et al. 2025. TWIX: Automatically Reconstructing Structured Data from Templatized Documents. *arXiv preprint arXiv:2501.06659* (2025). arXiv:2501.06659 [cs.CL]

[18] Yiming Lin, Madelon Hulsebos, Ruiying Ma, Shreya Shankar, Sepanta Zeigham, Aditya G. Parameswaran, and Eugene Wu. 2024. Towards Accurate and Efficient Document Analytics with Large Language Models. arXiv:2405.04674 [cs.DB] https://arxiv.org/abs/2405.04674

[19] 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 *Proceedings of the Conference on Innovative Database Research (CIDR)*. 2.

[20] 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*. 183–186.

[21] 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*. https://openreview.net/forum?id=5LXcoDtNyq

[22] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press. Web publication at informationretrieval.org..

[23] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in neural information processing systems* 35 (2022), 17359–17372.

[24] 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, 694-706 (1933), 289–337.

[25] OpenAI. 2025. Introducing Deep Research. https://openai.com/index/introducing-deep-research.

[26] OpenAI. 2025. Introducing GPT-5. https://openai.com/index/introducing-gpt-5/

[27] OpenAI. 2025. Introducing OpenAI o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/

[28] Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. 2024. Llms know more than they show: On the intrinsic representation of llm hallucinations. *arXiv preprint arXiv:2410.02707* (2024).

[29] Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. 2025. Semantic Operators: A Declarative Model for Rich, AI-based Data Processing. arXiv:2407.11418 [cs.DB] https://arxiv.org/abs/2407.11418

[30] 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*. https://openreview.net/forum?id=wHLMsM1SrP

[31] 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), 1–28.

[32] Glenn Shafer and Vladimir Vovk. 2008. A tutorial on conformal prediction. *Journal of Machine Learning Research* 9, 3 (2008).

[33] Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G Parameswaran, and Eugene Wu. 2024. Docetl: Agentic query rewriting and evaluation for complex document processing. *arXiv preprint arXiv:2410.12189* (2024).

[34] Qwen Team. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL] https://arxiv.org/abs/2505.09388

[35] 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* (Santiago AA, Chile) *(SIGMOD '24)*. Association for Computing Machinery, New York, NY, USA, 472–475. doi:10.1145/3626246.3654732

[36] Matthias Urban and Carsten Binnig. 2024. Efficient Learned Query Execution over Text and Tables [Technical Report]. arXiv:2410.22522 [cs.DB] https://arxiv.org/abs/2410.22522

[37] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. 2005. *Algorithmic learning in a random world*. Vol. 29. Springer.

[38] 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, Los Alamitos, CA, USA, 4662–4674. doi:10.1109/ICDE65448.2025.00374

[39] 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. http://sites.computer.org/debull/A25mar/p57.pdf

[40] 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.

[41] xAI. 2024. Introducing Grok-3: xAI's Next-Generation Language Model. https://x.ai/news/grok-3

[42] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887* (2018).