Hemolix.TabGen: Optimized Table Generation from Documents

Gyanendra Shrestha[1], Todor Ivanov[1], Karthik Vemireddy[1], Anna Pyayt[2], Michael Gubanov[1]
[1]Department of Computer Science, Florida State University
[2]Department of Chemical, Biological, and Materials Engineering, University of South Florida

Abstract

Modern Data Lakes contain vast and heterogeneous document collections, making table generation from documents a persistent and nontrivial challenge. Traditional approaches are often rigid — i.e. domain-specific, require extensive supervision, or are limited to set of pre-defined schemas; LLM-based approaches are more flexible, but typically suffer from hallucinations, non-determinism, and high computational costs. To overcome these limitations, we introduce Hemolix.TabGen, a novel scalable LLM-based table generation system that comprehends documents and generates Bi-dimensional tables based on the entire document content. We evaluated TabGen on 4 publicly available datasets spanning multiple domains and observed an Average Precision delta up to 30% compared to vanilla LLMs.

1 Introduction

Large enterprise and medical Data Lakes contain vast collections of different documents (e.g., medical case reports, legal contracts, financial disclosures, insurance summaries, research articles, policy forms, etc.) (Romero and Ventura, 2013; Bolyen et al., 2019). These documents represent information in a variety of different modalities, such as free text, images, tables, graphs, figures, etc.

Information Extraction (IE) (Sarawagi, 2008; Nargesian et al., 2019; Yafooz et al., 2013; Cafarella et al., 2007) is an area that helps solve this problem by developing approaches that automatically extract structured knowledge from raw documents (e.g., DOCX, PDF, plain text, etc.). Classical IE approaches (Lample et al., 2016; Zhong and Chen, 2021; Lin et al., 2020; Li et al., 2021; Wu et al., 2022) rely on manually designed templates or supervised learning models (Lample et al., 2016; Lewis et al., 2020; Devlin et al., 2019; Lee et al., 2020). While being reasonably effective within specific domains, these systems require large, domain-dependent annotated datasets (Gui et al., 2024) for training as well as IE accuracy is sensitive to the domain and starts degrading when it changes. For example, Text-to-Table (Wu et al., 2022), relies on labeled data and sequence-to-sequence formulations to extract relatively simple relational tables from short texts, with several extensions (Li et al., 2023; Deng et al., 2024; Pietruszka et al., 2024; Jiang et al., 2024). However, such approaches do not support longer multi-page documents, require domain-alignment and are limited in handling complex structured data.

Recent LLM-based table generation systems, such as Evaporate (Arora et al., 2023b), ZenDB (Lin et al., 2024), Palimpzest (Liu et al., 2024), LOTUS (Patel et al., 2024), Doctopus (Chai et al., 2025), and TabAgent (Wu et al., 2025), leverage LLMs to extract tables from semi-structured documents, offering promising zero-shot capabilities and domain independence. However, they do not support complex Bi-dimensional tables with hierarchical metadata, which are essential in many domains, such as medicine and finance (Wang et al., 2020; Shrestha et al., 2025; Kandibedala et al., 2025).

Here, we describe `Hemolix.TabGen`, a novel LLM-based domain-independent system designed specifically for scalable, unsupervised complex table generation from heterogeneous document collections. We observe the following contributions:
(1) Domain-agnostic Extraction: We introduce a novel *Table Generation* operator that does not require any domain-specific pre-training. It generalizes across domains, enabling extraction of both relational (Codd, 1983) and Bi-dimensional tables (Shrestha et al., 2025; Kandibedala et al., 2025) through simple task descriptions, eliminating expensive annotation and pre-training overhead.
(2) Document-wide Comprehension: For multi-page documents, `Hemolix.TabGen` employs advanced LLM-based comprehension. This signif-

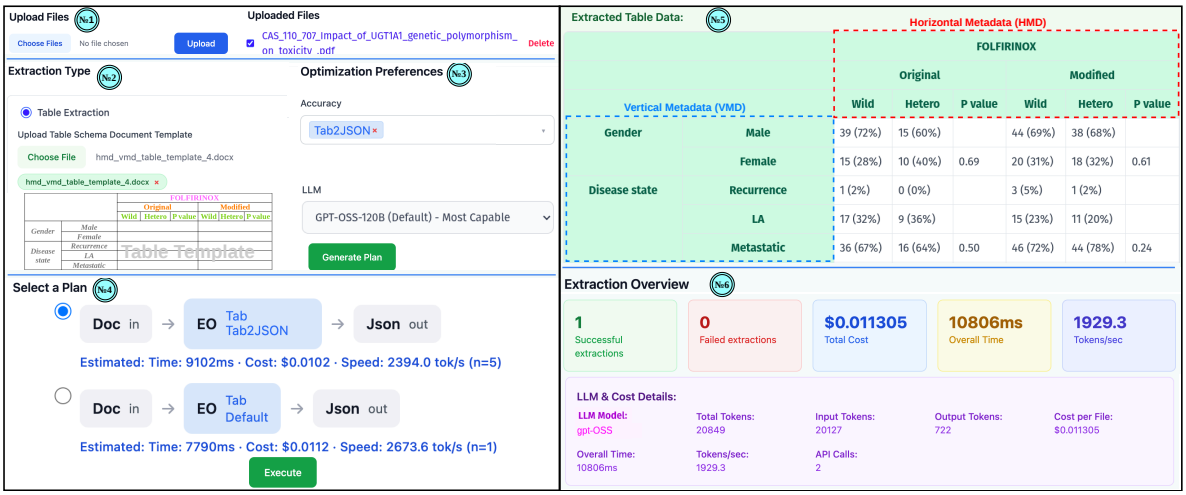

Figure 1: Table Generation in `Hemolix.TabGen`.

icantly increases the amount of information considered for table generation compared to classic solutions such as Text-to-Table, allowing efficient and accurate extraction from large multi-page documents.

(3) Adaptive Optimization: `Hemolix.TabGen` uses *Document Profiles* that capture document characteristics, such as *language, lexical diversity*, etc., and *LLM Profiles* that log the IE runtime metrics, such as *accuracy, cost, latency, throughput*, etc (Ivanov et al., 2026). Both profiles enable learning and subsequent selection and configuration of a specific LLM that would be the best fit for the task as well as selection and configuration of a best fitting set of `Hemolix.TabGen` custom optimizations.

(4) Table Generation Optimizer: `Hemolix.TabGen`'s supports an Optimizer that supports a variety of optimization strategies specifically designed and optimized for table generation.

We evaluate `Hemolix.TabGen` on several key IE tasks across four publicly available corpora spanning multiple domains. The system is evaluated to perform research and information retrieval at Moffit Cancer Center.

2 Usage Scenario

This section presents the `Hemolix.TabGen` interface through a *Table Generation* use case in the medical domain. Figure 1 shows a screenshot of the system (opened in the Web browser), where typical users—such as clinical data scientists, data engineers, health informatics specialists, and medical data analysts—extract key information (e.g., patient characteristics, treatments, adverse events, and outcomes) from multiple documents and output it in tabular format for further analysis by downstream analytical systems.

The process starts in Step №1, where the user uploads a document (the system can also process multiple documents) from the MCR and SCBC oncological data lakes (Shrestha et al., 2025). In Step №2, the user selects *"Table Extraction"* as action and uploads a Bi-dimensional table schema to be used as a *Table Template* (e.g. in Microsoft Word format in this example) to format and structure the extraction results. Next, in Step №3, the user selects the optimization and chooses GPT-OSS-120B LLM. The Execution Plans are then generated by the system and presented in Step №4. The user may select between the optimized plan or the default plan (without optimization). Estimated metrics for *execution time*, *cost*, and *speed*, based on previous executions, are provided for both plans. Here the user decided to select and execute the optimized plan. The extraction results are shown on the right in Figure 1. They are comprised of the *Extracted Table Data* (Step №5) and *Extraction Overview* (Step №6). Step №5 presents the generated table, which matches the schema of the provided *Table Template* as well as contains now the extracted values for the Vertical Metadata variables (VMDs, marked in blue) and Horizontal Metadata variables (HMDs, marked in red) (Shrestha et al., 2025; Kandibedala et al., 2025). This means the extraction was successful - i.e. all desired template attributes were located and extracted from the source document. SOTA LLM-based solutions and vanilla LLMs typically fail to preserve hierarchical relationships in HMD and VMD of Bi-dimensional tables that are critical for medical domain and might

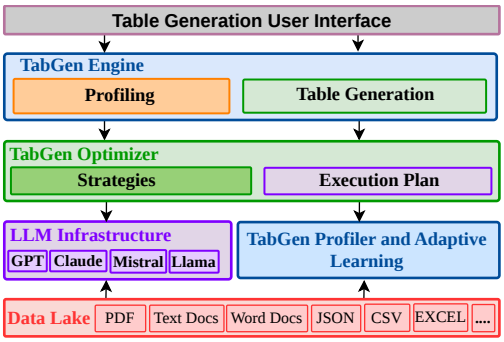

Figure 2: Architecture.

resort to flattening and extracting it as a relational table, which would be inaccurate. Finally, Step №6 reports on the successful execution, showing profiling metrics, such as total cost, overall execution time, and tokens per second.

3 System Overview

Here, we describe `Hemolix.TabGen` architecture.

TabGen Engine (TE) takes raw documents, extraction specifications (e.g., attribute lists or schema templates), and user-defined optimization preferences (*performance*, *accuracy*, or *cost*). TE consists of two main components: *Profiling* module and *Table Generation Operator* (EO^{TAB}). The *Profiling* module computes per-document properties/statistics (e.g., format, size, word/sentence counts, domain, etc.) for each input document. These properties are stored as a *Document Profile* in a centralized Profile Repository (Ivanov et al., 2026). EO^{TAB} generates structured tables with the specified *attributes* and their values coming from the document content. The actual location, where the variables were found in the document can widely vary depending on the document. They can be on different pages, in tables or other document structures. The generated output can be either a relational table (Codd, 1983) or a table with Bi-dimensional hierarchical metadata (Shrestha et al., 2025; Kandibedala et al., 2025). $\mathbf{EO^{TAB}}$ supports both *schema-free* and *template-driven* table generation. In the schema-free setting, the LLM autonomously identifies the most relevant attributes from the document, whereas template-driven generation follows a user-provided schema template.

TabGen Optimizer improves extraction *accuracy*, *efficiency*, and reduces *cost* by generating an optimized *Execution Plan*. To achieve this, it applies a range of optimization strategies (Table 1) that also help mitigate LLM hallucinations (Huang et al., 2023; Zhang et al., 2025). The Optimizer maintains custom parameterized prompt templates that define operator logic and available optimizations, and uses the *Document* and *LLM Profiles* to fetch the information needed for the selected optimization. The prompt template varies by LLM type and optimization setting, and includes placeholders for both user-specified table entities and the corresponding document content. The number of templates expands as new LLM types and optimization strategies are added.

Execution Plan (EP) is generated by the Optimizer according to the user preferences (*performance*, *accuracy*, or *cost*) and the current *LLM Profiles*. If no preferences are provided, the system defaults to a non-optimized execution. Each *EP* specifies an ordered sequence of operators and their selected optimizations, from which the *Optimizer* generates executable code, as illustrated in Figure 1.

TabGen Profiler supports adaptive model selection for table extraction tasks by maintaining the LLM Profile repository that stores *LLM Profiles*. Each *LLM Profile* accumulates runtime metrics, capturing variations across IE tasks, LLMs, and optimization settings. The *Profiler* provides task- and document-aware performance monitoring, enabling the system to automatically route table extraction tasks according to user-defined priorities. After each execution, runtime metrics are recorded in the *Execution Statistics* table and propagated to the corresponding *LLM Profile*, enabling informed LLM selection for similar future tasks. To support optimal EP selection, the *Profiler* employs three XGBoost models (Chen and Guestrin, 2016), trained on historical *Document Profile* and *LLM Profile* data, each predicting one optimization objective (*accuracy*, *cost*, or *execution time*) (Ivanov et al., 2026). The system thus predicts the optimal EP for a given workload, forming an adaptive feedback loop that continuously adapts to changes in LLM performance, model updates, and document domains.

4 Evaluation

We evaluated our system on four datasets using multiple SOTA LLMs.

Datasets: The Medical Case Reports (MCR) and Small Cell Bladder Cancer (SCBC) datasets (Shrestha et al., 2025) comprise 1,500 and 320 open-access medical publications, respectively, including text, figures, and both relational (20%) and Bi-dimensional (80%) tables,

Feature	Strategy	Use Case	Observed Gain
Scalability	Document-wide Comprehension	Comprehension of large multi-page documents, where using the entire document content improves table generation quality	BERT Score (BE) improvement up to 8.9%
	Table Validation	Automated quality assurance for generated tables without manual inspection (Algorithm 1)	Avg. Extraction Precision (AP) 85% (manual) vs. 82% (automated)
Metadata	Tab2JSON	Bi-dimensional tables with hierarchical metadata where vanilla LLMs fail to preserve schema	Metadata Precision (MP) improved from 0% to 100%
Hallucination Mitigation	Evidence Grounding	Extraction with supporting source spans for verification	AP improvement up to 21.3%
	Multi-step	Ambiguous schemas, long documents, or noisy source text where single-pass extraction is incomplete	AP improvement up to 8.4%; hallucinations reduced up to 3.4% (Claude)
	KG Augmentation & Inference	Domain-specific tasks where semantic knowledge can resolve ambiguous or missing attributes	AP improvement up to 10%

Table 1: Specific Use Cases and Gain Observed for Several Optimization Strategies.

Algorithm 1 Table Validation with LLM-based RAG

Input: Table **T** with **m** rows and **k** columns, source document **D**, iterations **n** (default 10)
Output: Average Precision score (P_{avg})
Initialize Array **scores**$[n]$ $=$ $\{\}$ $\rightarrow$to store precision scores
Number of iterations i = 1
while $i < n$ **do**
  Initialize correct_count $= 0$
  for each row **r** in table **T** **do**
    for each value **v** in row **r** **do**
      if Verify_with_LLM(**v**, **D**) **then**
        correct_count $++$
      end if
    end for
  end for
  Precision $P_i = \frac{\text{correct\_count}}{m*k}$; Store P_i in scores$[i]$; $i++$
end while
Average Precision P_{avg} = Average(scores)
return P_{avg}

with documents. We also use two Text-to-Table (Wu et al., 2022) datasets, E2E (Novikova et al., 2017) and RotoWire (Wiseman et al., 2017) (see (Wu et al., 2022) for detailed statistics).

Evaluation Metrics: We use task- and dataset-appropriate metrics to ensure fair comparability with prior work. BERTScore (BE) is used for E2E and RotoWire, following Text-to-Table (Wu et al., 2022). Average Precision (AP) measures cell-level correctness on MCR and SCBC (where ground-truth cell values are available). Metadata Precision (MP) measures structural fidelity i.e., the fraction of correctly preserved metadata attributes and hierarchical relationships relative to the schema template. Finally, the Presence metric (Pr) quantifies extraction completeness across iterative multi-step extraction steps.

System	E2E	Rotowire	SCBS
Text-to-Table	**98.56**	**92.97**	76.60
LOTUS	56.44	61.04	77.00
Doctopus	58.00	60.02	78.00
TabAgent	83.75	81.34	–
TabGen(Gemini)	92.34	86.00	82.50
TabGen(GPT-4)	94.21	83.72	83.30
TabGen(Llama 3)	89.27	85.76	83.30
TabGen(Claude 3.7)	93.26	75.82	**85.50**

Table 2: Comparison of `Hemolix.TabGen` Table Generation with SOTA. BERT Score (BE) in %.

4.1 Comparison with SOTA Methods

We compare `Hemolix.TabGen` against four recent SOTA LLM-based IE systems: Text-to-Table (Wu et al., 2022), LOTUS (Patel et al., 2024), Doctopus (Chai et al., 2025), and TabAgent (Wu et al., 2025). Table 2 summarizes the results.

On the E2E and RotoWire benchmarks, Text-to-Table achieves higher BERTScores (BE) due to task-specific training and fine-tuning, whereas `Hemolix.TabGen` generates schemas directly from input without predefined attributes or domain-specific training, and supports multi-page documents. On the E2E benchmark, `Hemolix.TabGen` (GPT-4) achieves a higher BE improvement over LOTUS (37.77%), Doctopus (36.21%), and TabAgent(83.75%); on the RotoWire benchmark, `Hemolix.TabGen` (Gemini) similarly outperforms LOTUS (24.96%), Doctopus (25.98%), and TabAgent (4.66%). On the SCBC dataset, `Hemolix.TabGen` (Claude 3.7) achieves BE of 85.5%, surpassing Text-to-Table (76.6%), LOTUS (77%) and Doctopus (78%), without task-specific fine-tuning. As TabAgent is not publicly available, we could not evaluate it on our datasets.

`Hemolix.TabGen` outperforms vanilla LLM baselines (i.e., LLMs queried directly without

Hemolix.TabGen's pre-processing, profiling, and optimization infrastructure) by up to 30% in AP, demonstrating the concrete benefit of the system's optimizer-driven architecture.

4.2 Ablation Study

Here, first we evaluate Hemolix.TabGen's ability to extract information from complex documents (MCR and SCBC datasets) using the specified table schemas in the template files. We tested both *relational* and *Bi-dimensional* tables across MS Word, HTML, JSON, and XML formats, (see three schema-templates in Figure 3): two *Bi-dimensional* tables with *hierarchical* metadata (one with *Bi-dimensional hierarchical* metadata (Shrestha et al., 2025), the other with HMD only), and a standard *relational* table with 28 attributes.

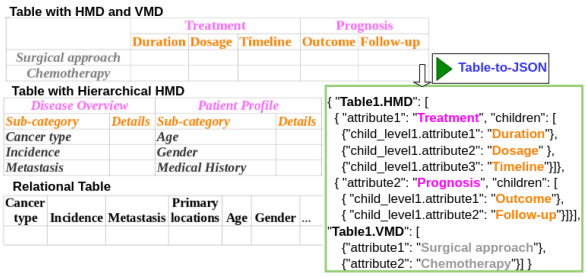

Figure 3: Sample Schema-Templates.

Templates	TabGen (GPT-4)		TabGen (Claude 3.7)	
	Data	Metadata	Data	Metadata
MS Word-Relational	43.03	26	61.57	24
MS Word-HMD	43.82	0	66.39	0
MS Word-HMD_VMD	44.98	48	60.11	26
HTML-Relational	79.90	28	73.30	26
HTML-HMD	61.19	0	65.70	0
HTML-HMD_VMD	67.52	34	63.39	54
JSON (All Types)	**82.14**	**100**	**84.08**	**100**
XML (All Types)	76.52	**100**	71.05	**100**

Table 3: Average Precision (in %) for Different Table Types and Templates. The best results are in **bold**.

Templates	TabGen (GPT-4)		TabGen (Claude 3.7)	
	STemp.	JSON	STemp.	JSON
Relational Table	**74.62**	**94.94**	**80.45**	94.15
HMD Table	58.20	88.48	74.82	88.33
HMD_VMD Table	64.79	93.50	68.60	**95.50**

Table 4: Average Precision (in %) for Schema-Template (STemp.) converted to JSON.

For each experiment, the user provides a document, chooses a schema-template file, then Hemolix.TabGen generates a prompt to extract information while preserving the template's structure. The initial experiments were conducted without any optimizations using GPT-4 and Claude 3.7 due to a higher cost incurred with using more LLMs. We evaluate the extraction based on two criteria: *Precision* of the extracted data, measured by the correctness of cell values (*Data* columns in Table 3) and the *Metadata Precision* (MP), which measures preservation of the correct schema-template structure i.e. schema/metadata and hierarchical metadata (*Metadata* columns in Table 3). MP is measured by the number of correctly generated metadata and hierarchical relationships compared to the template. Table 3 reports Average Precision (AP) across table types and formats. Results for MS Word and HTML are reported separately for each table schema, whereas JSON and XML results are averaged across different table types, as we observed minimal variation in *Precision*. MS Word templates yield the lowest MP (24.67% for GPT-4 and 16.67% for Claude 3.7). HTML templates also performed poorly, with averages of 20.67% (GPT-4) and 26.67% (Claude 3.7). Notably, both models failed entirely to preserve HMD templates in MS Word and HTML. Errors that lowered MP included missing attributes, omission of parent-child relationships, and hierarchy distortions (i.e., the structure of the extracted metadata hierarchy differed from the original template). In contrast, JSON and XML formats preserved the metadata hierarchy accurately. For data extraction, MS Word templates performed the worst, with AP of 43.94% (GPT-4) and 62.69% (Claude 3.7), followed by HTML templates 69.54% and 67.46%, respectively). Both models achieved the highest *Precision* with the JSON template. Similar results were observed in (Singha et al., 2023), where the authors found that the JSON outperformed HTML format in structural table understanding tasks, aligning with our findings. We hypothesize that JSON and XML perform better due to their explicit structural encoding (e.g., attribute-value pairs, nested hierarchies), which aligns closely with schema-templates, whereas MS Word and HTML rely on inconsistent visual cues (e.g., merged cells, indentation, font styling) or tag-based layouts, making extraction more difficult.

Next we applied the *Tab2JSON* optimization (see Table 1), which converts templates to JSON while preserving structure. Table 4 compares AP from the previous experiment under the *STemp.* column with the new *Precision* after applying the *Tab2JSON* optimization under the *JSON* column. This improved AP by 26.43% (GPT-4) and 18.04% (Claude 3.7). The results suggest that Claude outperforms GPT for Schema-Template Extraction.

We define the *Presence* metric to measure gener-

ation completeness:

$$P_r(d_i) = \frac{M_{T_i}}{N_{T_1 \cup T_2 ... \cup T_k}} \quad (1)$$

where $P_r(d_i)$ (in %) is the *Presence* for output table T_i of an individual document d_i, M_{T_i} is the number M attribute values X stored in the output table T_i, and $N_{T_1 \cup T_2 ... \cup T_k}$ is the total number N of attribute values $X_1 \cup X_2 ... \cup X_k$ stored across all K output generated tables $T_1 \cup T_2, .., \cup T_k$.

Multi-step Table Generation: Given an input document and a task description specifying the target attributes and table schema (see PT1 in Appendix C for a sample prompt template), `Hemolix.TabGen` generates tables (Figure 3 shows an example of the extracted Bi-dimensional schema - *Table with HMD and VMD*.) performing a single-pass extraction. `Hemolix.TabGen` supports *Multi-step Extraction* optimization (Table 1), which improves accuracy and reduces hallucination errors (Huang et al., 2023). We applied *Multi-step Extraction* over k iterations (user defined parameter), logically divided into $k+1$ steps. First, a document and task description are submitted to `Hemolix.TabGen`, producing "generated Table 1" T_1. The next $k-1$ steps repeat this process, yielding T_2 to T_k. In the final step, these are merged to produce the final table T_f. Moreover, since too many iterations might lead to longer processing time and higher monetary costs, we use k = 3 as suggested (Arora et al., 2023a).

We conducted this experiment with three LLM configurations: GPT-4, Claude 3.7, and an ensemble of both. On the MCR dataset, the merged table achieved the highest *Presence* of 76.1%, while individually generated tables showed a slight lower scores, ranging from 67% to 68.5%. This indicates that extracting attribute values using LLMs is more accurate when the merged table T_f from *Multi-step Extraction* is used, rather than relying on individually generated tables ($T_1, T_2, \ldots, T_k$). In terms of cost and runtime, GPT-4 results in an average cost of $0.83 with a runtime of 137.43 sec per multi-step table generation, while Claude 3.7 is notably cheaper at $0.093, but slower with a runtime of 332.28 sec. More detailed error analysis is summarized in Appendix A.1.

Ablation Summary: We systematically evaluate each optimization by comparing the system with and without the optimization. Tab2JSON fully eliminates metadata hallucinations on HMD tables, improving MP from 0% to 100%, since the hierarchical JSON explicitly encodes complex table structure. Multi-step extraction reduces hallucination rates from 5.7% to 3.1% for GPT-4 and from 8.2% to 4.8% for Claude (see Table 6 in Appendix), by exploiting LLM stochasticity across independent runs and fusing outputs via majority-vote-style selection. The Evidence Grounding yields up to 21.3% higher AP by enforcing explicit grounding through supporting text spans. KG (Knowledge Graph)-augmented extraction additionally improves accuracy up to 10% as shown in Figure 4. Together, these optimizations substantially reduce hallucinations and improve extraction quality.

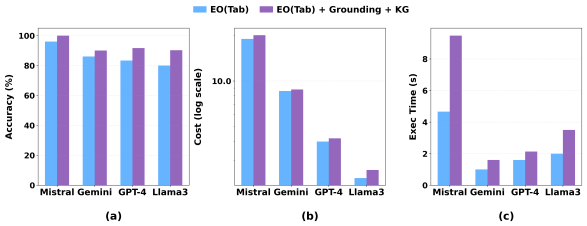

Figure 4: LLM-Profile-guided Table Generation. (a) Accuracy (%); (b) Cost ($); (c) Execution Time (sec.). 'EO(Tab)' denotes Table Extraction Operator; '+' denotes optimized extraction (see Table 1).

4.3 TabGen Profiler Analysis

The TabGen Profiler selects the most suitable model for each table extraction task using previous execution history and user-specified preferences. As illustrated in Figure 4(a), Mistral achieves the highest accuracy. In contrast, when the preference is to optimize cost (Figure 4(b)), Llama 3 provides the lowest execution cost, though with a small reduction in accuracy. The three XGBoost models underlying the Profiler, trained on 1,500 historical execution records combining Document Profile and LLM Profile features, achieve F1 scores of 0.98 (accuracy objective), 0.91 (cost objective), and 0.90 (latency objective), confirming sufficient signal for reliable execution plan selection. In our experiment Profiler improved accuracy by 8–12% at comparable cost and reduced cost by 10–15% at comparable accuracy versus static model selection. To summarize, these results show that our TabGen Profiler enables effective model selection based on task-specific priorities such as *accuracy*, *cost*, or *execution time*.

5 Related Work

Table Extraction (TE) IE task aims to extract structured knowledge from documents. Converting text to tables has been approached through super-

vised learning (Smock et al., 2022) and sequence-to-sequence generation (Wu et al., 2022). Systems such as (Li et al., 2023; Deng et al., 2024; Pietruszka et al., 2024; Chen and Koudas, 2024) primarily handle simple relational tables from short texts using seq2seq frameworks from Text-to-Table (Wu et al., 2022), with extensions for parallel row generation (Li et al., 2023), sports-related table summarization (Deng et al., 2024), T5-based approaches (Pietruszka et al., 2024), and retrieval-augmented enrichment (Chen and Koudas, 2024). Unlike these, our approach generalizes to both relational and Bi-dimensional tables and supports documents of varying lengths and structural complexity (Kandibedala et al., 2023; Gubanov et al., 2009, 2008; Gubanov, 2017; Gubanov et al., 2014).

Prior work, such as (Circi et al., 2024; Pavia et al., 2022) focuses on domain-specific extraction from tables in materials science articles, whereas our system is domain-agnostic. SciDaSynth (Wang et al., 2024) integrates LLMs (GPT) with retrieval-augmented generation (RAG) for domain-specific relational tables extraction. ArxivDIGESTables (Newman et al., 2024) uses LLMs to automatically generate literature review tables in the scientific domain, but is limited to relational tables (Gubanov et al., 2017; Khan and Gubanov, 2018b; Gubanov and Shapiro, 2012; Khan and Gubanov, 2020a, 2018a). In contrast, our method is evaluated across multiple domains and supports complex Bi-dimensional table extraction, with optimizations targeting time, cost, and accuracy (Gubanov et al., 2024; Villasenor et al., 2017; Chauhan et al., 2023).

Recent systems, such as Evaporate (Arora et al., 2023b), LOTUS (Patel et al., 2024), ZenDB (Lin et al., 2024), Doctopus (Chai et al., 2025), Palimpzest (Liu et al., 2024) and TabAgent (Wu et al., 2025) leverage LLMs for table extraction from semi-structured documents, but do not support complex Bi-dimensional table generation. TableCoder (Dong et al., 2025) extracts relational and complex tables by prompting LLMs to generate executable code (e.g., Python or SQL), but relies on a prompt-based pipeline rather than an optimizer-driven, operator-based design and is not publicly available for comparison. Systems such as Aryn (Anderson et al., 2024) and DocETL (Shankar et al., 2024) primarily optimize output quality, our approach jointly optimize cost, accuracy, and performance. Unlike prompt- or supervision-based methods such as GPTuner (Lao et al., 2024) and FieldSwap (Xie et al., 2024), our solution is unsupervised, does not require domain-specific pre-training and supports both relational and Bi-dimensional table generation.

6 Conclusion

Here we described `Hemolix.TabGen`, a novel, domain-independent LLM-based system for unsupervised extraction of structured information from complex documents (Gubanov and Pyayt, 2014; Simmons et al., 2017; Podkorytov et al., 2017; Pavia et al., 2021). The system supports the generation of relational and complex Bi-dimensional tables, with/without predefined schemas (Gubanov and Pyayt, 2013; Khan and Gubanov, 2020b). It performs document-level reasoning to locate and extract information that may appear across multiple pages or heterogeneous layouts, and incorporates several optimization strategies to alleviate typical SOTA LLMs shortcomings, such as hallucinations.

The current system relies on commercial SOTA LLMs which exhibit shortcomings standard for all LLMs including hallucinations. According to our evaluation, we demonstrated significant progress in this direction by developing multiple advanced optimizations for table generation and continue working to even further increase accuracy and scalability.

Acknowledgments

This work was supported by NSF Grant 2345794 and Florida Department of Health Grant 25C06.

References

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.

Simran Arora, Avanika Narayan, Mayee F Chen, Laurel J Orr, Neel Guha, Kush Bhatia, Ines Chami, and Christopher Ré. 2023a. Ask me anything: A simple strategy for prompting language models. In *ICLR*.

Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. 2023b. Language models enable simple systems for generating structured views of heterogeneous data lakes. *Proc. VLDB Endow.*

Evan Bolyen, Jai Ram Rideout, Matthew R Dillon, Nicholas A Bokulich, Christian C Abnet, Gabriel A

Al-Ghalith, Harriet Alexander, Eric J Alm, Manimozhiyan Arumugam, Francesco Asnicar, and 1 others. 2019. Reproducible, interactive, scalable and extensible microbiome data science using qiime 2. *Nature biotechnology*, 37(8):852–857.

Michael J Cafarella, Christopher Re, Dan Suciu, Oren Etzioni, and Michele Banko. 2007. Structured querying of web text. In *3rd Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, California, USA*.

Chengliang Chai, Jiajun Li, Yuhao Deng, Yuanhao Zhong, Ye Yuan, Guoren Wang, and Lei Cao. 2025. Doctopus: Budget-aware structural table extraction from unstructured documents. *VLDB Endowment*.

Maitry Chauhan, Anna Pyayt, and Michael N. Gubanov. 2023. Learning topical structured interfaces from medical research literature. In *ACM Web Conference 2023, WWW*. ACM.

Kaiwen Chen and Nick Koudas. 2024. Unstructured data fusion for schema and data extraction. *Proc. ACM Manag. Data*, 2(3):181.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794.

Defne Circi, Ghazal Khalighinejad, Anlan Chen, Bhuwan Dhingra, and L Brinson. 2024. Extracting materials science data from scientific tables. In *ACL Workshop Language+ Molecules*.

E. F. Codd. 1983. A relational model of data for large shared data banks. *CACM*, 26(1):64–69.

Zheye Deng, Chunkit Chan, Weiqi Wang, Yuxi Sun, Wei Fan, Tianshi Zheng, Yauwai Yim, and Yangqiu Song. 2024. Text-tuple-table: Towards information integration in text-to-table generation via global tuple extraction. In *EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Haoyu Dong, Yue Hu, Huailiang Peng, and Yanan Cao. 2025. Tablecoder: Table extraction from text via reliable code generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 1399–1412. Association for Computational Linguistics.

M. Gubanov and A. Pyayt. 2013. Readfast: High-relevance search-engine for big text. In *ACM CIKM*.

M. Gubanov and A. Pyayt. 2014. Type-aware web search. In *EDBT*.

Michael Gubanov. 2017. Polyfuse: A large-scale hybrid data fusion system. In *ICDE*.

Michael Gubanov, Manju Priya, and Maksim Podkorytov. 2017. Cognitivedb: An intelligent navigator for large-scale dark structured data. In *WWW*.

Michael Gubanov, Anna Pyayt, and Aleksandra Karolak. 2024. Cancerkg.org - A web-scale, interactive, verifiable knowledge graph-llm hybrid for assisting with optimal cancer treatment and care. In *CIKM*. ACM.

Michael Gubanov and Linda Shapiro. 2012. Using unified famous objects (ufo) to automate alzheimer's disease diagnostics. In *BIBM*.

Michael N Gubanov, Philip A Bernstein, and Alexander Moshchuk. 2008. Model management engine for data integration with reverse-engineering support. In *2008 IEEE 24th International Conference on Data Engineering*, pages 1319–1321. IEEE.

Michael N Gubanov, Lucian Popa, Howard Ho, Hamid Pirahesh, Jeng-Yih Chang, and Shr-Chang Chen. 2009. Ibm ufo repository: Object-oriented data integration. *VLDB*.

Michael N. Gubanov, Michael Stonebraker, and Daniel Bruckner. 2014. Text and structured data fusion in data tamer at scale. In *ICDE*.

Honghao Gui, Shuofei Qiao, Jintian Zhang, Hongbin Ye, Mengshu Sun, Lei Liang, Jeff Z Pan, Huajun Chen, and Ningyu Zhang. 2024. Instructie: A bilingual instruction-based information extraction dataset. In *ISWC*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM TOIS*.

Todor Ivanov, Gyanendra Shrestha, Karthik Vemireddy, Anna Pyayt, and Michael Gubanov. 2026. Hemolix.extract.v: Llm-based information extraction for documents with ai-based plan selection. In *SIGMOD*, New York, NY, USA. ACM.

Peiwen Jiang, Xinbo Lin, Zibo Zhao, Ruhui Ma, Yvonne Jie Chen, and Jinhua Cheng. 2024. Tkgt: Redefinition and a new way of text-to-table tasks based on real world demands and knowledge graphs augmented llms. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16112–16126.

Bhimesh Kandibedala, Anna Pyayt, Nickolas Piraino, Chris Caballero, and Michael N. Gubanov. 2023. COVIDKG.ORG - a web-scale COVID-19 interactive, trustworthy knowledge graph, constructed and interrogated for bias using deep-learning. In *EDBT*. OpenProceedings.org.

Bhimesh Kandibedala, Gyanendra Shrestha, Anna Pyayt, and Michael Gubanov. 2025. Scalable tabular hierarchical metadata classification in heterogeneous structured large-scale datasets using contrastive learning. *ICDE*.

Rituparna Khan and Michael Gubanov. 2018a. Nested dolls: Towards unsupervised clustering of web tables. In *IEEE Big Data*.

Rituparna Khan and Michael Gubanov. 2018b. Towards unsupervised web tables clustering. In *IEEE BigData*.

Rituparna Khan and Michael Gubanov. 2020a. Towards tabular embeddings, training the relational models. In *IEEE Big Data*.

Rituparna Khan and Michael Gubanov. 2020b. Weblens: Towards interactive large-scale structured data profiling. In *CIKM*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL-HLT*.

Jiale Lao, Yibo Wang, Yufei Li, Jianping Wang, Yunjia Zhang, Zhiyuan Cheng, Wanghu Chen, Mingjie Tang, and Jianguo Wang. 2024. Gptuner: A manual-reading database tuning system via gpt-guided bayesian optimization. *Proceedings of the VLDB Endowment*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinform.*

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*.

Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. In *NAACL-HLT*. ACL.

Tong Li, Zhihao Wang, Liangying Shao, Xuling Zheng, Xiaoli Wang, and Jinsong Su. 2023. A sequence-to-sequence&set model for text-to-table generation. In *ACL*.

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.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *ACL*.

Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baille Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, and Gerardo Vitagliano. 2024. A declarative system for optimizing ai workloads.

Fatemeh Nargesian, Erkang Zhu, Renée J Miller, Ken Q Pu, and Patricia C Arocena. 2019. Data lake management: challenges and opportunities. *Proceedings of the VLDB Endowment*, 12(12):1986–1989.

Benjamin Newman, Yoonjoo Lee, Aakanksha Naik, Pao Siangliulue, Raymond Fok, Juho Kim, Daniel S Weld, Joseph Chee Chang, and Kyle Lo. 2024. Arxivdigestables: Synthesizing scientific literature into tables using language models. In *EMNLP*.

Jekaterina Novikova, Ondrej Dusek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *SIGDIAL*. ACL.

Liana Patel, Siddharth Jha, Parth Asawa, Melissa Pan, Carlos Guestrin, and Matei Zaharia. 2024. Semantic operators: A declarative model for rich, ai-based analytics over text data.

Sophie Pavia, Rituparna Khan, Anna Pyayt, and Michael Gubanov. 2021. Towards unveiling dark web structured data. In *BigData*. IEEE.

Sophie Pavia, Nickolas Piraino, Kazi Islam, Anna Pyayt, and Michael Gubanov. 2022. Hybrid metadata classification in large-scale structured datasets. *J. Data Intell.*, 3(4).

Michal Pietruszka, Michal Turski, Lukasz Borchmann, Tomasz Dwojak, Gabriela Nowakowska, Karolina Szyndler, Dawid Jurkiewicz, and Lukasz Garncarek. 2024. Stable: Table generation framework for encoder-decoder models. In *EACL*.

Maksim Podkorytov, Dylan Soderman, and Michael N. Gubanov. 2017. Hybrid.poly: An interactive large-scale in-memory analytical polystore. In *ICDM Workshops*, pages 43–50. IEEE Computer Society.

Cristobal Romero and Sebastian Ventura. 2013. Data mining in education. *Wiley Interdisciplinary Reviews: Data mining and knowledge discovery*, 3(1):12–27.

Sunita Sarawagi. 2008. Information extraction. *Foundations and Trends® in Databases*.

Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. 2024. Docetl: Agentic query rewriting and evaluation for complex document processing.

Gyanendra Shrestha, Chutian Jiang, Sai Akula, Vivek Yannam, Anna Pyayt, and Michael Gubanov. 2025. Tabular embeddings for tables with bi-dimensional hierarchical metadata and nesting. In *EDBT*.

Mark Simmons, Daniel Armstrong, Dylan Soderman, and Michael Gubanov. 2017. Hybrid.media: High velocity video ingestion in an in-memory scalable analytical polystore. In *IEEE Bigdata*.

Ananya Singha, José Cambronero, Sumit Gulwani, Vu Le, and Chris Parnin. 2023. Tabular representation, noisy operators, and impacts on table structure understanding tasks in llms. *arXiv preprint arXiv:2310.10358*.

Brandon Smock, Rohith Pesala, and Robin Abraham. 2022. Pubtables-1m: Towards comprehensive table extraction from unstructured documents. In *CVPR*.

Santiago Villasenor, Tom Nguyen, Anusha Kola, Sean Soderman, and Michael Gubanov. 2017. Scalable spam classifier for web tables. In *IEEE Big Data*.

Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Michael Kinney, and 1 others. 2020. Cord-19: The covid-19 open research dataset. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*.

Xingbo Wang, Samantha L Huey, Rui Sheng, Saurabh Mehta, and Fei Wang. 2024. Scidasynth: Interactive structured knowledge extraction and synthesis from scientific literature with large language model. *arXiv preprint arXiv:2404.13765*.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2253–2263. Association for Computational Linguistics.

Jingfei Wu, Chaoyuan Shen, Qiyan Deng, Yuping Wang, Jiajun Li, Yuhao Deng, and Minghe Yu. 2025. Tabagent: A multi-agent table extraction framework for unstructured documents. *VLDB Endowment. ISSN*.

Xueqing Wu, Jiacheng Zhang, and Hang Li. 2022. Text-to-table: A new way of information extraction. In *ACL*.

Jing Xie, James B Wendt, Yichao Zhou, Seth Ebner, and Sandeep Tata. 2024. Fieldswap: Data augmentation for effective form-like document extraction. In *ICDE*.

Wael MS Yafooz, Siti ZZ Abidin, Nasiroh Omar, and Zanariah Idrus. 2013. Managing unstructured data in relational databases. In *2013 IEEE Conference on Systems, Process & Control (ICSPC)*, pages 198–203. IEEE.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, and 1 others. 2025. Siren's song in the ai ocean: A survey on hallucination in large language models: Siren's song in the ai ocean: A survey on hallucination in large language models". *Computational Linguistics*, 51(4).

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *NeurIPS*.

Zexuan Zhong and Danqi Chen. 2021. A frustratingly easy approach for entity and relation extraction. In *NAACL-HLT*. ACL.

A Experiments

Setup. The experiments were performed on a server equipped with a 112-core Intel Xeon Platinum 8180 2.50 GHz CPU, 3 NVIDIA V100 GPUs, 1.5 TB RAM, and 16 TB of storage.

A.1 Error Evaluation

In our experimental evaluation of the LLM-based approaches, we encountered multiple errors and inconsistencies in the multi-step table generation involving data extraction and fusion steps. After thorough analysis, we identified two main types of errors, which are explained below with specific examples for each type.

Extraction Errors - These errors occur when the LLM omits information that was present in the source document. We distinguish four types of extraction errors:

- *Missed/skipped attributes/variables*: For example, Physical Exam results *(Lungs were clear to auscultation)* were mentioned in the original case report, but LLM failed to extract and capture them in the response table.
- *Incorrectly extracted attributes/variables*: For example, the LLM recorded in its response 1.5 months instead of capturing *1 month 3 weeks (Duration)*.
- *Incorrect table structure/categorization*: For example, *Case report details* category mixed patient demographics with medical details.
- *Hallucinations*: For example, patient's family and social history was invented, without any mention in the original document.

Fusion Errors - These errors occurred when the LLM attempted to merge the three output response tables into a consolidated merged table. We identified five distinct fusion errors:

- *Mismatches in merging attributes/variables*: For example, *Incidence* sub-category mismatched with *Diagnostic Tests* data.
- *Missed/skipped attributes/variables in source tables*: For example, several relevant attributes from the three response tables, such as *Initial Symptoms* and *Key Diagnostic Findings*, were omitted in the merged table.
- *Duplicate attributes/variables in the merged table*: For example, redundant volume measurements and repeated age range information.
- *Hallucinations in the merged table*: For example, merged table introduces a combined chemotherapy timeline not present in the original document.
- *Incorrect data transformation/merging*: For example, the final "Follow-up outcomes" section transformed the timeline incorrectly. Instead of clearly stating the 12-month follow-up with no tumor recurrence, it generalized the time frame and created confusion about the exact period.

We summarized the different types of errors in Table 6. The table presents both the absolute number of errors and their corresponding percentages of the total errors for GPT and Claude. Analysis reveals that extraction errors significantly outnumber fusion errors across both LLMs. Specifically, the first three types of extraction errors account for 70.5% of GPT's errors and 55.3% of Claude's errors. In terms of fusion errors, Claude exhibits a higher proportion, at 36.6% of total errors, compared to GPT's 23.8%.

In summary, GPT demonstrates a greater tendency for extraction errors, while Claude shows a higher susceptibility to fusion errors.

B TabGen Optimizer

B.1 Preserving Metadata Integrity

While out-of-the-box LLMs can manage simple relational tables (Codd, 1983), they fail to correctly infer headers (metadata per (Kandibedala et al., 2025)) in complex, non-relational Bi-dimensional tables (Shrestha et al., 2025). When such schemas contain the attributes to extract, `Hemolix.TabGen` applies the **Tab2JSON** optimization to convert the table into a hierarchical JSON format that preserves both horizontal (HMD) and vertical metadata (VMD) correctly. After the schema is converted to JSON, the LLM extracts attribute values from the document using this structured representation as guidance. This conversion helps greatly improve accuracy, as LLMs tend to extract and align values more accurately when operating over well-defined hierarchical structures rather than raw,

Category	Sub-Category	Details
Disease Characteristics	Nature of Disease	Renal Medullary Carcinoma (RMC) is a rare, aggressive malignancy predominantly found in young patients of African descent with sickle cell trait.
	Incidence and Prognosis	Very poor prognosis, with survival typically less than 1-year post-diagnosis. Most cases present metastasis or local invasion at diagnosis.
Patient Demographics	Demographics	29-year-old African female with sickle cell trait.
	Medical History	No significant medical history prior to current diagnosis.
Clinical Presentation and Symptoms	Symptoms	Presented with chronic cough, fever, and abnormal chest X-ray showing mediastinal widening.
	Diagnostic Tests	CT scans revealed multiple masses in the mediastinum, lungs, liver, and a large kidney mass. Diagnosis confirmed by pathology with loss of INI1 expression.

Table 5: Sample Ground Truth Table Structure.

Operation	Type of Error	GPT	GPT %	Claude	Claude %
LLM Extraction	Missed/skipped attributes/variables	**123**	**33.6**	**91**	**27.5**
	Incorrectly extracted attributes/variables	**65**	**17.8**	**47**	**14.2**
	Incorrect table structure/categorization	**70**	**19.1**	**45**	**13.6**
	Hallucinations	21	5.7	27	8.2
LLM Fusion	Mismatches in merging attributes/variables	22	6.0	29	8.8
	Missed/skipped attributes/variables in source tables	26	7.1	40	12.1
	Duplicate attributes/variables in the merged table	18	4.9	22	6.6
	Hallucinations in the merged table	11	3.0	17	5.1
	Incorrect data transformation/merging	10	2.7	13	3.9
	Total	**366**		**331**	

Table 6: LLM Error Types.

complex tables (Singha et al., 2023). Prompt Template (PT3) provides a step-by-step description of the optimization strategy.

B.2 Table Validation

For table validation processes each row by applying an LLM-based RAG verification (i.e. an LLM-as-a-Judge (Zheng et al., 2023) approach) to compare attribute values against the source document. Implementation details to calculate *Precision* are provided in Algorithm 1, which takes as input the table T, the source document D, and an optional number-of-iterations n parameter, and returns the average precision (AP) P_{avg} for the table. The number of iterations is configurable, allowing users to balance processing time and cost, although reducing the number of iterations may impact accuracy. Importantly, this validation is implemented as an optional optimization: users can either enable it for automated checking or rely on manual inspection when the higher cost is not a concern. Across 500 experiments on relational and Bi-dimensional table generation (e.g., JSON, HTML), manual evaluation yields an AP of 85%, while automated validation achieves 82%, closely approximating the manual score and enabling the scale up of our evaluation process.

C Prompts

PT1. Example Table Generation Prompt Template (EO^{TAB})

<Organize the information from the provided document into a table with specific categories and sub-categories. For each main category - *Disease Characteristics, Patient Demographics, ...* — please include relevant sub-categories based on the document's content. Each section should summarize key details from the document in a structured manner.>

PT2. Example Merge Tables Prompt Template)

Merge the provided tables into a single table, ensuring consistency and avoiding duplication.

PT3. Tab2JSON Optimization

<*instructions*>, *Extract and convert the schema of each provided table into a hierarchical JSON format while preserving its structural relationships.*
Analyze the the table schema (i.e., header and sub-header rows/columns) and identify hierarchical relationships between attributes. Treat each header or sub-header as a **schema node** in the JSON structure.