UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

UNIVERSITY OF MODENA AND REGGIO EMILIA

"Enzo Ferrari" Department of Engineering

Master's Degree in Artificial Intelligence Engineering (LM-32)

Document Understanding for Automated ESG Reporting: System Design and Experimental Validation

Supervisor:
Prof. Nicola Bicocchi
Dr. Marcello Pietri

Candidate:
Giorgia Bertacchini
Student ID 193871

ACADEMIC YEAR 2024/2025

Abstract

Document Understanding for Automated ESG Reporting:
System Design and Experimental Validation

This thesis addresses the increasing demand from financial institutions and ESG-oriented (Environmental, Social, and Governance) organizations for automated solutions capable of improving the retrieval, interpretation and analysis of information contained in corporate reports, with particular emphasis on structured financial data embedded in complex tabular structures. While modern Artificial Intelligence (AI) techniques have significantly advanced document analysis, extracting reliable and structured information from complex and heterogeneous corporate documents remains a challenging task, particularly in ESG reporting contexts where numerical accuracy and semantic consistency are critical for supporting transparent sustainability reporting and data-driven decision-making.

Particular attention is given to Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) architectures, which provide a promising framework for enhancing information access and analytical capabilities. However, the effectiveness of these systems is strongly dependent on the accuracy and reliability of document data extraction processes, especially when dealing with heterogeneous information such as numerical data, textual content and graphical indicators including embedded symbols and icons.

To address this challenge, the thesis proposes a structured pipeline for generating and processing PDF-based datasets specifically designed to evaluate document extraction tasks involving complex tabular structures. The pipeline enables controlled generation of evaluation documents and the systematic benchmarking of document understanding tools and AI models.

An experimental evaluation is conducted to assess the performance of different extraction approaches in terms of accuracy, robustness, and suitability for ESG-related analytical workflows. The results contribute to identifying the strengths and limitations of current AI-based document extraction techniques and provide insights into their integration within intelligent ESG reporting systems.

Keywords: ESG, Document Extraction, Tabular structures, AI, RAG

Contents

List of Figures

Introduction

In recent years, environmental and ethical issues have assumed growing importance. Public demand for transparency and accountability has accelerated this shift. As a result, the integration of environmental, social, and governance (ESG) factors has radically transformed corporate reporting practices. Regulatory frameworks such as the European Union's Corporate Sustainability Reporting Directive (CSRD) now require organizations to collect, validate, and disclose increasingly detailed ESG data. In practice, much of this work still relies on manual, labour-intensive workflows that are vulnerable to inconsistencies and human error, and that are becoming unsustainable as regulatory requirements expand [25]. ESG reports are multimodal artefacts in which narrative text, numerical tables, and graphical indicators jointly communicate evidence. Tables, in particular, exhibit multi-level headers, merged cells, implicit hierarchies, footnotes, and formula-based aggregation rows. Unlike linear corpora assumed by standard Natural Language Processing (NLP) pipelines, ESG disclosures are typically distributed as PDF documents in which spatial arrangement carries semantic meaning [22, 30].

Large Language Models (LLMs) have demonstrated strong capabilities in document understanding and information extraction, making them relevant for ESG workflows. However, their direct application to regulated reporting introduces the risk of hallucination, which in a compliance context can lead to regulatory penalties and financial liability. Retrieval-Augmented Generation (RAG) [12] mitigates this limitation; however, the reliability of such systems depends critically on upstream document parsing quality [27, 32]. At the same time, existing benchmarks, including PubTables-1M [22], PubTabNet [30], and more recent finance-oriented datasets, still do not reflect the full structural complexity of sustainability documents, while commonly used metrics such as TEDS treat all cells uniformly and ignore computational dependencies.

The document parsing landscape itself is increasingly heterogeneous, ranging from rule-based PDF extractors, through modular pipelines that combine dedicated table structure recognizers with OCR, to end-to-end frameworks and frontier multimodal LLMs. Recent comparative studies suggest that no single paradigm consistently outperforms the others across all document categories, since performance depends strongly on the structural characteristics of the input, the balance between

textual and tabular content, and the specific extraction objectives of the downstream task [23, 16].

This thesis investigates the problem of automated document understanding for ESG reporting. The work encompasses the design of a synthetic data generation strategy tailored to the structural complexity of sustainability documents, the definition of a domain-aware evaluation methodology that accounts for the heterogeneous nature of ESG content and weights errors by criticality annotations, and a comparative assessment of representative systems spanning different methodological families. Beyond extraction itself, this work is motivated by the need for document representations that preserve tabular structure as a meaningful form of knowledge, enabling more reliable retrieval and reasoning over the complex relationships encoded in ESG reports [27].

Figure 1 provides an overview of the thesis structure and its three main contributions.

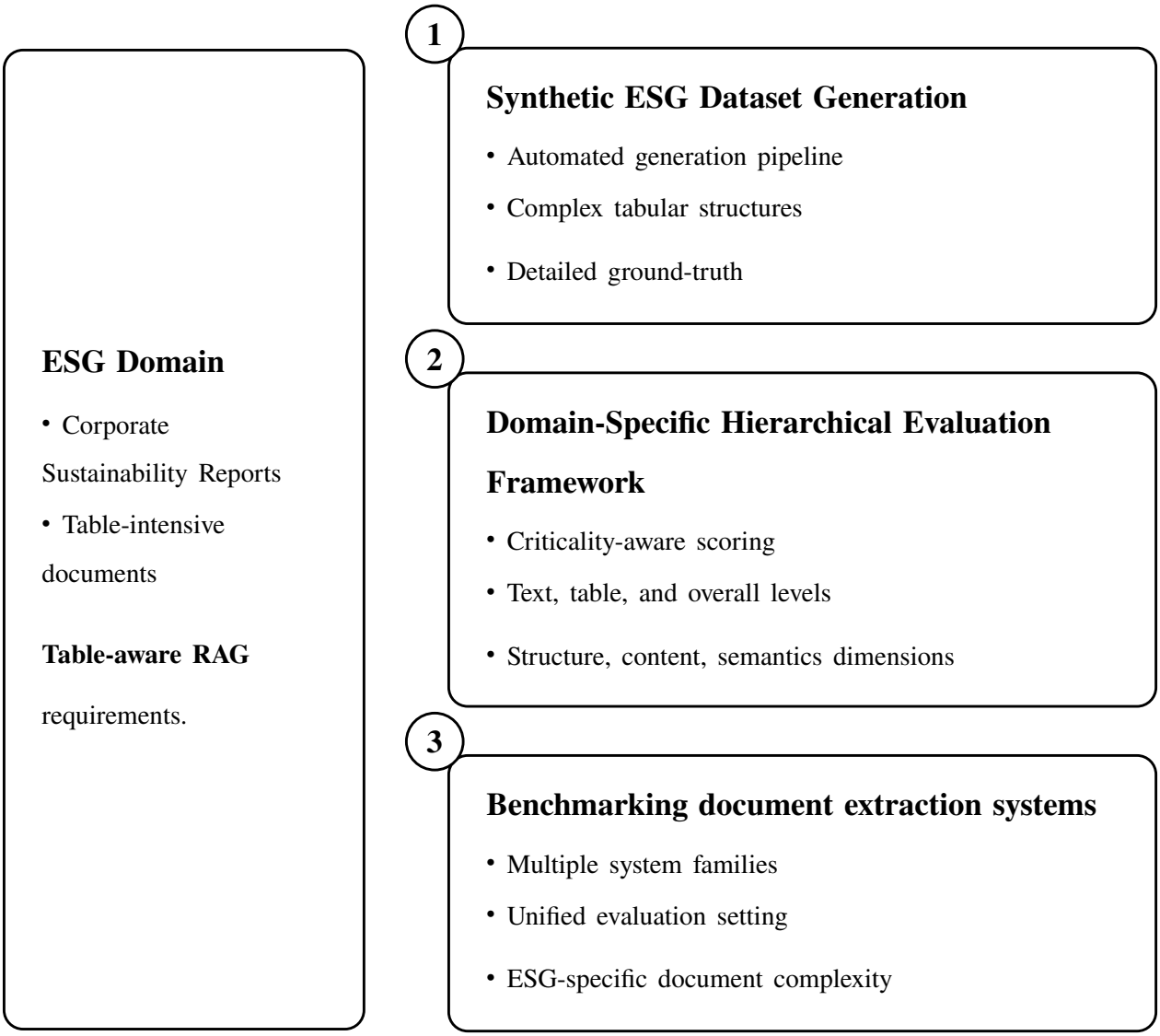

Figure 1: Overview of the thesis structure, illustrating the three main contributions and their grounding in the ESG document understanding domain.

1. ESG Document Reporting

1.1 Regulatory Pressure in ESG Disclosure

As introduced in Introduction, the expansion of ESG regulations has placed growing demands on corporate disclosure. This chapter examines the domain-specific characteristics that make ESG reporting particularly challenging from a document understanding perspective and motivates the technical requirements that any automated system must satisfy.

In the financial sector, ESG indicators extend the assessment of corporate performance beyond traditional economic metrics by incorporating environmental impact, social responsibility, and governance practices. Today, financial institutions are not only responsible for reporting their own ESG performance but must also collect and validate sustainability data from counterparties and suppliers, creating complex, multi-layered reporting structures in which inconsistencies can easily propagate across documents.

The challenge is amplified by the diversity of ESG data sources. Relevant data may come from sustainability reports, regulatory filings, compliance questionnaires, internal documentation, and third-party providers, all of which differ in structure, terminology, and reporting conventions. As a result, professionals spend significant time reviewing documents, extracting relevant data, and verifying numerical indicators from different sources.

From Manual Processing to Intelligent Automation

Once data have been accurately extracted, additional AI models can support downstream tasks such as risk assessment, financial assessment of potential new partnerships, or predictive analysis of ESG indicators. AI-based systems can also support questionnaires by pre-filling ESG forms, identifying missing information, and flagging inconsistencies with data extracted from other documents. However, these tasks remain challenging because ESG questionnaires and reports often differ substantially in format and structure.

1.2 ESG Reporting Challenges

Unlike traditional financial reporting, ESG information is not yet fully standardized across jurisdictions, industry sectors, or reporting bodies. Although harmonization efforts are underway, differences remain in terminology, units of measurement, reporting templates, and key performance indicators.

This heterogeneity introduces semantic ambiguity and complicates automated processing. ESG reports typically combine qualitative descriptions with quantitative indicators embedded in structured tables. Automated systems must address both linguistic variability and numerical precision, while maintaining consistency and traceability across different document sources.

Nature of ESG Reporting Data

From the perspective of document intelligence, ESG reports are challenging not only due to regulatory pressure but also because of the way information is physically represented within documents. They are multimodal artifacts in which narrative text, numerical tables, charts, graphical indicators, and explanatory notes jointly convey evidence. Moreover, they are only partially structured: key values are often embedded in tables with multi-level headers, merged cells, implicit hierarchies, and references distributed across captions and footnotes.

A further source of difficulty is the temporal and documentary variability. The same indicator may be reported for different years, entities, or document versions using different layouts, renamed metrics, and inconsistent units of measurement. The position of a value on the page, its alignment with row and column headers, and its connection to surrounding notes often determine its correct interpretation. Automated ESG reporting therefore requires an integrated document AI pipeline that combines OCR, layout-aware parsing, table reconstruction, retrieval, and grounded generation.

Visually-Rich Document Structures

Beyond the heterogeneity of the reported content, ESG and financial documents also pose substantial technical challenges due to their visually complex layout. Reports often include multi-column pages, irregular reading order, nested tables with hierarchical headers, merged cells, footnotes, figures, dashboards, and form-like elements such as checkboxes or conditional fields. In these settings,

relevant information is not defined only by textual content, but also by its spatial organization on the page.

Traditional Information Extraction (IE) methods are often insufficient, especially for tables and semi-structured regions. The interpretation of a numerical value may depend on its alignment with the row and column headers, its position within a hierarchical table, or its connection to the surrounding notes. Robust ESG document understanding requires layout-aware models and parsing pipelines capable of table recognition, structural reconstruction, and value validation.

Grounded Reasoning and Hallucination Risk

However, the naive implementation of LLMs for ESG reporting activities introduces risks that are difficult to tolerate in a regulated and high-risk environment. LLMs are prone to generating plausible but factually incorrect results, a phenomenon commonly referred to as hallucination. When applied to financial data, they can misinterpret numerical indicators, fabricate metric values, or confuse entities between documents. In the context of ESG compliance, such errors have real-world consequences such as regulatory penalties, reputational damage, and financial responsibility.

These challenges are not new to regulatory authorities. For example, the European Securities and Markets Authority (ESMA) recently published the report "Leveraging Large Language Models in Finance: Pathways to Responsible Adoption."[1], in which numerous experts explicitly addressed the growing integration of large language models in financial services. It emphasized that systems implemented in regulated contexts must ensure explainability, traceability, robust governance, and verifiable results. Like many other corporate and institutional initiatives, this report reinforces the need for architectures capable of combining advanced generative reasoning with rigorous factual control.

In this context, retrieval-augmented generation (RAG) has emerged as a promising paradigm for reconciling the expressive capabilities of LLMs with the evidentiary standards required in ESG reporting. Within such a setting, ensuring that the retrieved evidence is relevant to the user query is essential for maintaining factual consistency, especially in ESG reporting, where answers must remain strictly grounded in source documents.

[1]ESMA, *Leveraging Large Language Models in Finance: Pathways to Responsible Adoption*, 2025. https://www.esma.europa.eu/sites/default/files/2025-06/LLMs_in_finance_-_ILB_ESMA_Turing_Report.pdf.

2. Background and Related Work

This chapter examines the main technical foundations underlying AI-based ESG document understanding. While Chapter 1 introduced the regulatory context and the main operational challenges of ESG reporting, the focus now shifts to the computational pipeline required to transform heterogeneous reports into searchable, auditable, and machine-usable evidence.

A significant share of ESG indicators is reported in tabular form. In the sustainability domain, Weichel et al. [25] highlight the importance of robustness in determining the variability of layout, since sustainability reports often depart from standardized tabular structures. This challenge is also reflected in benchmark datasets such as PubTables-1M [22], introduced to support fine-grained evaluation of table detection and table structure recognition. Yu et al. [27] similarly show that conventional retrieval-augmented generation approaches remain limited when applied to heterogeneous documents that combine text and tables, as converting tables into linear text tends to destroy their relational structure and weaken the model's reasoning capabilities.

The chapter therefore reviews the literature on retrieval-augmented generation, document and table understanding benchmarks, evaluation metrics, and parsing methods for visually structured business documents.

2.1 Retrieval-Augmented Generation (RAG) in ESG

Large language models (LLMs) are becoming increasingly important in business settings, but when they operate without sufficient contextual grounding they may produce outputs that appear credible while lacking factual support, a phenomenon commonly referred to as hallucination. This limitation emerged in our earlier investigations in other domains, including AI-driven agents in urban environments [3] and Industrial IoT scenarios from a worker-centric safety perspective [17]. In both cases, the broader lesson was that reliable AI behavior depends on access to structured, context-aware information. A similar need for contextual grounding has also been highlighted in ESG data extraction, where access to relevant document context is essential for accurate interpretation and reliable information retrieval [32]. Therefore, Retrieval-Augmented Generation (RAG) [12]

mitigates this issue by relying on an external knowledge corpus.

Why RAG Fits ESG and Financial Reporting

RAG is particularly suitable for ESG and financial reporting because it improves *auditability* by grounding outputs in retrieved evidence, supports *knowledge* currency by allowing the indexed corpus to be updated without retraining, and adapts well to the *heterogeneous* nature of ESG information, which is often distributed across narrative text, tables, appendices, questionnaires, and internal documents.

At the same time, errors introduced during parsing, chunking, or retrieval may propagate to the final answer. In regulated settings, RAG is usually combined with structured outputs, numerical checks, and human validation, making document parsing a critical prerequisite for reliable retrieval.

2.1.1 RAG Architecture

RAG combines two forms of memory: the *parametric memory* encoded in model weights and a *non-parametric memory* represented by an external document collection. During inference, the model first retrieves relevant evidence and then generates an answer conditioned on that evidence.

Core Intuition and Formal View

Given a query q, a RAG system retrieves a set of passages $D = \{d_1, \ldots, d_k\}$ and generates an answer y conditioned on the retrieved evidence:

$$p(y \mid q) \approx \sum_{i=1}^{k} p(y \mid q, d_i)\, p(d_i \mid q)$$

where $p(d_i \mid q)$ denotes the *retrieval relevance* assigned to passage d_i, and $p(y \mid q, d_i)$ is the *probability* of generating y conditioned on that passage. In practice, most systems *concatenate the top-k retrieved passages* into a single context window and generate a final response from that context. This approximation preserves the grounding in the retrieved evidence while remaining computationally efficient.

RAG Pipeline Overview

The RAG pipeline considered in this thesis can be described as consisting of two main phases:

1. **Offline indexing (knowledge preparation).** Documents are first acquired and normalized, then segmented into retrievable units, embedded, and stored together with their metadata.
2. **Online inference (question answering).** Given a user query, the system retrieves the most relevant chunks, optionally reranks them, and passes them to the language model as context to produce a grounded answer.

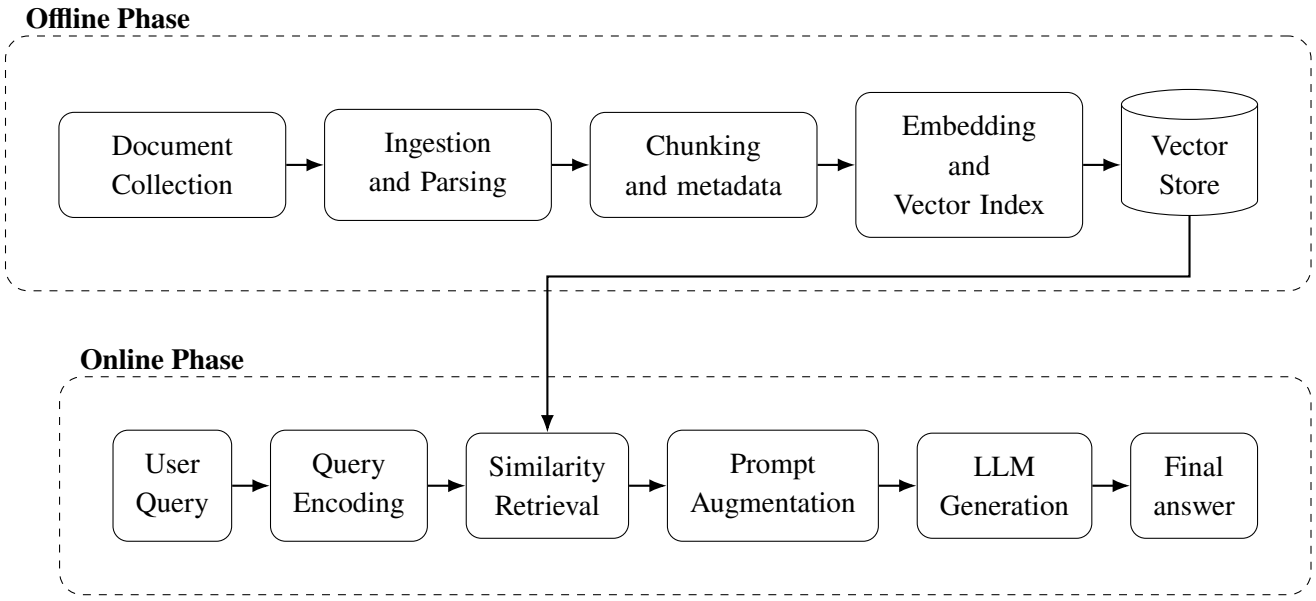

Figure 2.1: Overview of a basic Retrieval-Augmented Generation (RAG) pipeline, illustrating the offline ingestion phase and the online query-answering phase.

Offline Phase: Knowledge Ingestion and Indexing

Document ingestion and normalization. ESG reporting draws on heterogeneous sources, including sustainability reports, supplier questionnaires, financial documents, and Excel-based metric files. These documents must be converted into a machine-readable form and normalized while preserving key metadata, such as provenance, section hierarchy, page references, and table or figure boundaries.

Chunking and metadata. After extraction, documents are divided into chunks that balance context-window limits with semantic coherence. Each chunk is linked to metadata supporting filtering and traceability, such as company, year, document type, language, taxonomy, and page number.

Embedding and vector indexing. Each chunk d is projected into a dense vector space through an embedding model. These vectors are stored in a vector database to support efficient similarity search. The resulting index forms the knowledge base queried during the online phase.

Online Phase: Query-to-Answer Generation

Query understanding and query transformation. The effectiveness of retrieval largely depends on how well the query captures the underlying information need. For this reason, many systems apply query transformation before retrieval. In ESG settings, this may include *query rewriting*, such as acronym expansion, terminology disambiguation, and unit normalization, as well as *multi-query generation* to improve recall.

Retrieval strategies. Retrieval can be implemented through several complementary strategies. *Sparse retrieval* based on term-frequency statistics (e.g., BM25) is particularly effective when exact lexical matches matter. *Dense retrieval* [10] captures semantic similarity beyond exact wording. *Hybrid retrieval* combines both signals and is often preferable in ESG scenarios, where precise terminology and semantic coverage are both required.

Reranking and evidence selection. The initial top-k results may include duplicates, outdated passages, or only weakly relevant evidence. A reranking stage, often based on a cross-encoder, can refine candidate ordering by scoring each passage jointly with the query. In ESG applications, evidence selection may also consider source priority and numerical consistency.

Context construction and prompting. Selected passages are inserted into the LLM input together with the query and explicit task instructions. A typical prompt structure follows the pattern:

Instruction + **Question** + **Retrieved Evidence** (with identifiers) + **Output constraints**

In ESG settings, prompts often require source attributions, structured outputs such as JSON, and an explicit `not found` response when evidence is missing. This makes the generated answer more firmly grounded in retrieved evidence and, at the same time, easier to audit and trace back to its sources.

2.1.2 LLM Orchestration Framework

RAG is a conceptual paradigm and must be integrated into a broader software framework to become a usable system. This requires orchestration of document parsers, embedding models, vector stores, retrievers, language models, and output validators. In ESG reporting, such frameworks are particularly important because they must also support traceability and human oversight.

In recent years, the ecosystem for LLM applications has matured considerably. Instead of building RAG systems entirely from scratch, developers can rely on frameworks that provide reusable abstractions for retrieval, memory, tool use, and output validation.

LangChain. LangChain is one of the most widely adopted open-source orchestration frameworks for LLM applications. It is based on a modular design, allowing components such as chains, agents, retrievers, tools, memory, and output parsers to be combined into pipelines. It is also provider-neutral, which makes it easier to switch between model providers and infrastructure backends with limited changes. One of LangChain's main strengths is its support for agent-based workflows, in which the model can iteratively invoke tools, issue retrieval steps, and refine intermediate results before producing a final answer.

LlamaIndex. LlamaIndex has a more document-centric focus. It is designed primarily for building and querying indexes, offering multiple indexing strategies such as vector, keyword, tree, and knowledge-graph structures. It also provides advanced capabilities for document segmentation, embedding, and structured content handling.

2.1.3 Table-Aware RAG

Traditional RAG pipelines are primarily designed for unstructured text, whereas ESG and financial reports contain a large amount of tabular information. Tables encode meaning through row–

column relations, header hierarchies, alignment patterns, and sometimes arithmetic dependencies. Flattening them into plain text simplifies indexing, but weakens the recovery of their original semantics. For this reason, recent work on tabular RAG explores representations that better preserve table structure. The challenge lies not only in answer generation, but also in retrieving the correct structured evidence from heterogeneous documents.

Structure-Preserving Linearization and Element-Aware Indexing

One line of work remains compatible with standard vector retrieval while improving how tables are represented during indexing. Instead of splitting financial reports into generic paragraph chunks, these methods organize documents according to structural elements such as headings, narrative sections, lists, and tables, preserving more of the original layout.

A closely related strategy consists in serializing tables in HTML or Markdown rather than in plain text. This keeps row labels, column headers, and part of the structure visible to the embedding model and retrieval system, while remaining compatible with common embedding pipelines and vector databases.

However, once a table is converted into a sequence, operations such as filtering, aggregation, comparison across periods, or exact value matching become less reliable. This is especially problematic in ESG reporting, where numerical evidence must often be interpreted together with units, temporal references, and reporting boundaries.

SQL-Oriented Retrieval over Tables

Another line of work treats tables as relational objects rather than as text. When queries require row selection, header-based filtering, value comparison, or aggregation, purely semantic retrieval is often insufficient. In these cases, retrieval identifies the relevant table or schema, while part of the reasoning is delegated to SQL or SQL-like executable programs.

This logic underlies approaches such as TableRAG [27], which separate narrative and tabular content and process them through complementary retrieval mechanisms. Prose is retrieved semantically, while tables are stored in structured form and queried through executable operations. Related work on table-augmented generation (TAG) and text-to-SQL systems similarly supports the use of structured backends for precise evidence extraction [7].

At the architectural level, extracted tables are not only summarized textually, but also stored in a relational database together with provenance and structural metadata, such as document identifier, page number, caption, reporting year, unit of measurement, row labels, and column headers. A common two-phase process first uses semantic search over summaries and metadata to identify the relevant table, and then delegates exact extraction, filtering, aggregation, or comparison to the SQL engine.

This strategy is especially useful for ESG reporting, where many questions are semi-structured and require combining textual definitions with filtering over KPI tables by year, scope, or emission category. A relational representation also improves traceability, since answers can be linked back to the source table, page, and document [25, 32].

Graph-Based and Hybrid Retrieval

Research has also explored methods for retrieving information distributed across multiple tables, sections, or documents. In these cases, indexing each table independently may be insufficient, because the answer depends on explicit relations between entities, metrics, time periods, or reporting units. Graph-based RAG addresses this limitation by representing knowledge as nodes and edges rather than isolated chunks.

A recent example is GTR (*Graph-Table-RAG*) [31], which organizes a table corpus as a heterogeneous graph and applies a hierarchical coarse-to-fine retrieval strategy. Its main advantage is that retrieval can exploit structural and semantic links among tables, rather than relying only on similarity with a single serialized table.

More broadly, graph-based RAG improves generation when retrieved evidence must preserve explicit relationships among multiple knowledge elements. GRAG (*Graph retrieval-augmented generation*) [8] retrieves textual subgraphs rather than isolated fragments and integrates both textual and topological signals into generation. In the financial domain, HybridRAG [19] combines vector retrieval with knowledge graph retrieval and reports improved performance.

For ESG reporting, these approaches are particularly useful when queries require linking information across disclosures, years, or document sections, such as comparing an emissions indicator over time, linking a KPI in a summary table to its methodological explanation, or reconciling related values across tables.

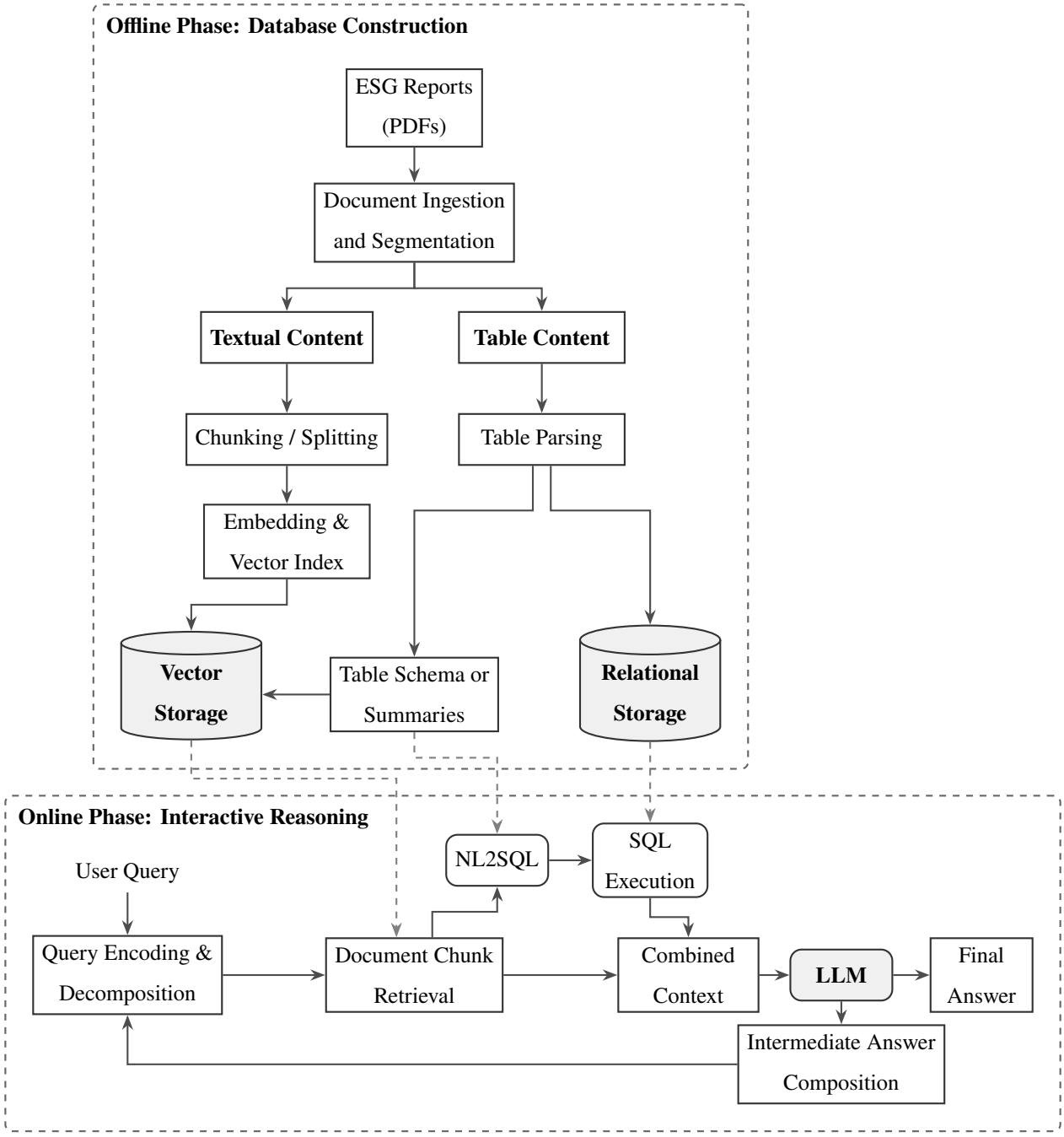

Figure 2.2: Table-aware retrieval-augmented generation pipeline for ESG reporting, combining unstructured text retrieval with structured querying over extracted tables. Solid lines denote the primary data and processing flow, while dashed lines indicate auxiliary connections and supporting interactions.

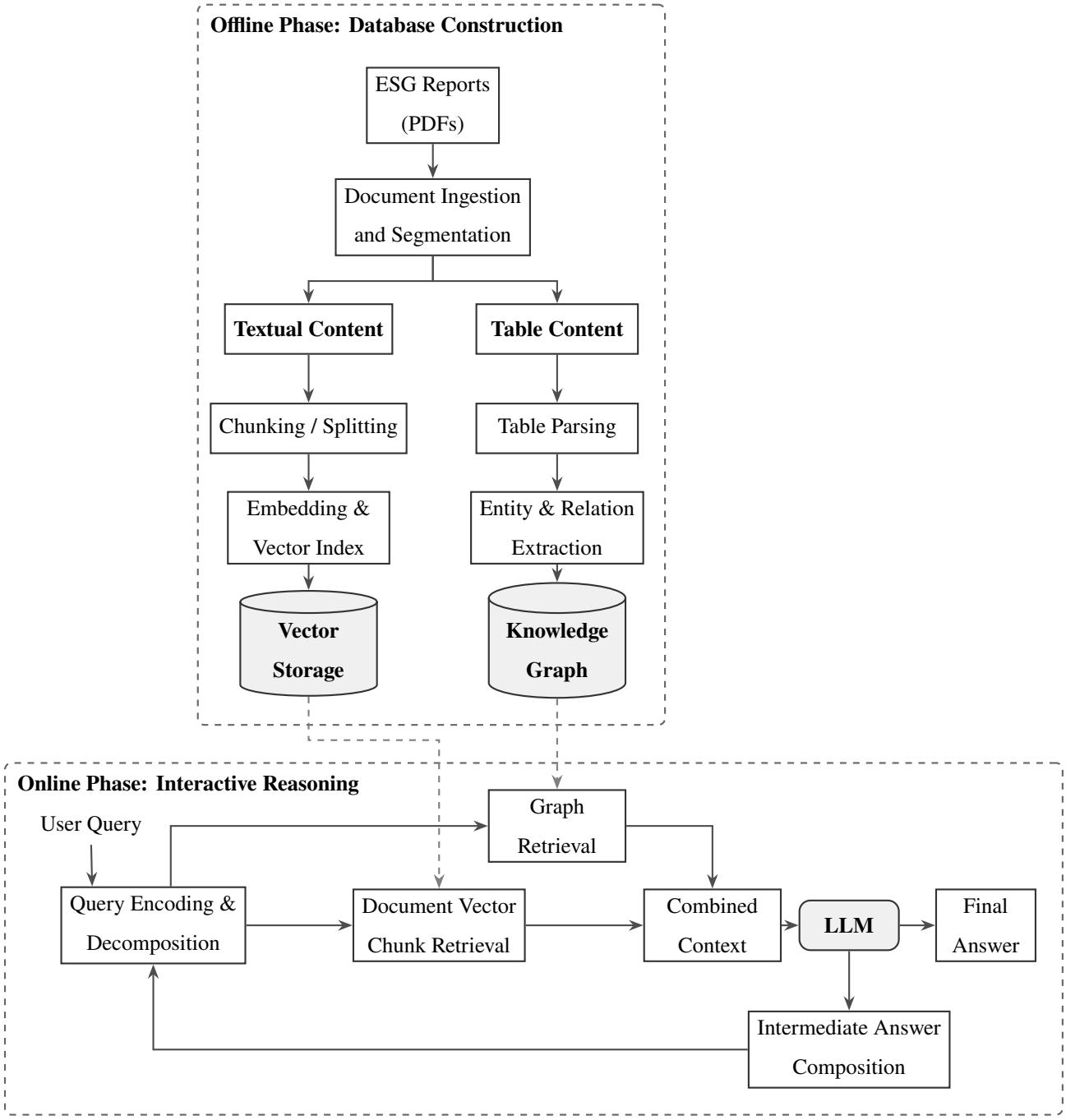

Figure 2.3: Overview of a hybrid RAG pipeline integrating vector retrieval for text and knowledge-graph retrieval for table-derived structured information. Solid lines denote the primary data and processing flow, while dashed lines indicate auxiliary connections and supporting interactions.

A more recent direction is represented by relational foundation models such as KumoRFM [6], which perform in-context learning directly over structured relational data without task-specific fine-tuning. In the ESG context, such models could reason directly over extracted relational structures by learning representations that capture dependencies, as illustrated in Figure 2.4. This paradigm is still in its early stages, but its potential to simplify the pipeline while preserving relational semantics makes it a compelling avenue for future investigation.

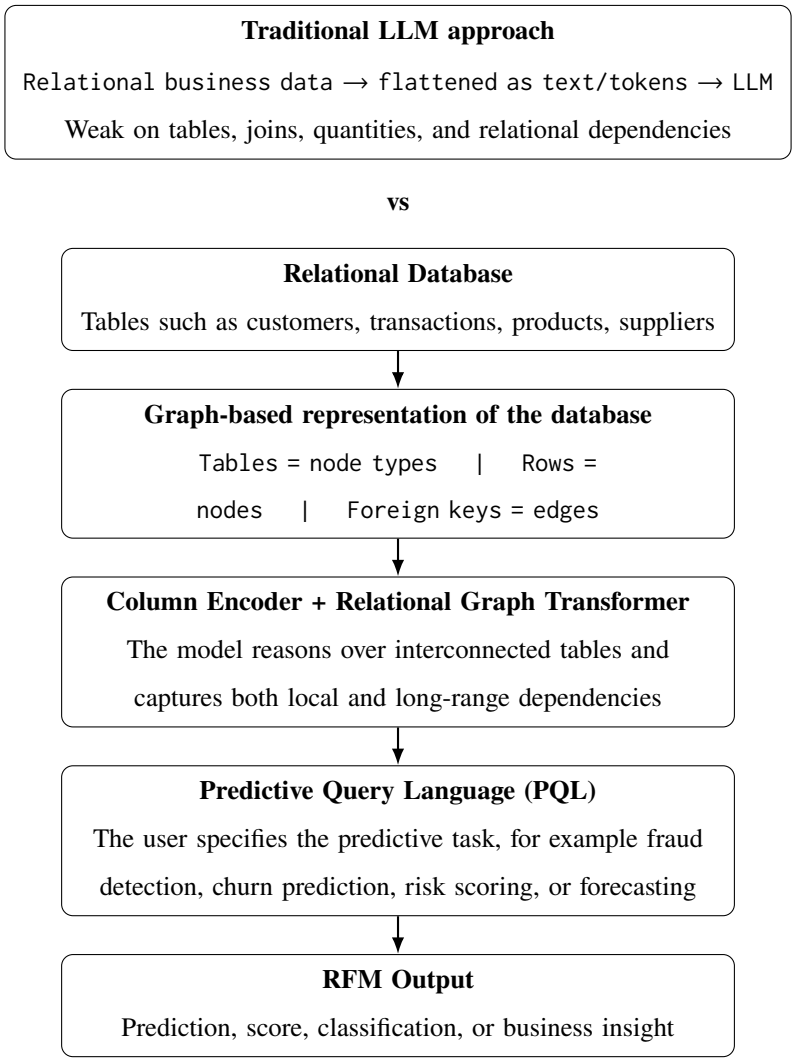

Figure 2.4: Conceptual schema of a relational foundation model performing in-context learning on structured relational data.

2.2 Document Parsing for Visually Structured Reports

Document parsing refers to the transformation of a document into a machine-readable representation that preserves not only textual content, but also the spatial and logical relations between its components. In the case of ESG and financial reports, this task is particularly challenging because relevant information is distributed across heterogeneous elements such as narrative paragraphs, section hierarchies, captions, footnotes, tables, and multi-column layouts. Moreover, these documents may be born-digital PDFs, in which the textual information is directly accessible, or image-based files, in which the content must first be recovered from the page image.

From an information extraction perspective, processing structured report pages involves two distinct but interdependent activities: the reconstruction of the document or table structure and the semantic interpretation of each element within that structure.

2.2.1 From OCR to Document Understanding

Optical Character Recognition (OCR) has historically represented the first step in document understanding pipelines. Its purpose is to convert page images into textual sequences, making scanned documents searchable and processable by downstream systems. Classic OCR engines such as Tesseract have played a major role in this evolution and remain relevant in practical pipelines, particularly when image-based reports or rasterized pages are encountered [20]. In its modern versions, Tesseract relies on a neural architecture based on Long Short-Term Memory (LSTM), which is specifically designed to recognize sequences of data and is therefore well suited to line-based text recognition.

Even in domains where most documents are digitally generated, OCR cannot be entirely discarded, because reports may still be distributed as scanned copies, embedded images, or visually protected PDFs. However, OCR alone is not sufficient for robust document parsing. Its output is primarily textual, whereas many downstream tasks require structural interpretation. In a complex table, for example, recognizing the words contained in a cell is only part of the problem. The system must also infer whether the cell belongs to a header or to the body, whether it spans multiple columns, and which row or column labels define its meaning.

For this reason, OCR should be seen as an enabling component rather than as a complete

document understanding solution. More recent research in document AI has progressively moved beyond plain text recognition by treating documents as multimodal objects, in which textual, visual, and spatial signals must be jointly interpreted. This transition is particularly important in ESG reports, where headings, nearby prose, table captions, indentation, and spatial alignment jointly contribute to the meaning of the extracted content. A parser that ignores layout may still recover fragments of text, but it will struggle to reconstruct the semantic relationships necessary for reliable structured extraction.

2.2.2 Table detection and structure recognition

Tables introduce a specific layer of complexity within document parsing, organizing information through explicit and implicit row-column relations. As a result, table understanding is usually decomposed into multiple subtasks: *table detection* (TD), which localizes the table region on the page; *table structure recognition* (TSR), which reconstructs rows, columns, and cell boundaries; and, in some cases, *functional analysis* (FA), which distinguishes headers, body cells, and projected row headers.

Early deep learning approaches adapted generic object detectors to the document domain. Architectures derived from Faster R-CNN and related detectors were among the first to obtain strong results in table localization, while later systems such as CascadeTabNet extended this paradigm toward end-to-end detection and structure recognition. A major step forward was then introduced by transformer-based detection models. In particular, Table Transformer (TaTR), developed together with the PubTables-1M benchmark, showed that DETR-style architectures can perform table detection and structure recognition effectively when trained on large-scale, carefully canonicalized annotations [4, 22]. These approaches are especially relevant because the parser recovers the table as a structured object, rather than merely as a collection of text.

Another challenge concerns semantic interpretation. A numerical value inside a table is not self-contained, its meaning depends on the relational context established within the table. Table extraction remains difficult because real-world documents contain merged cells, blank separator rows, footnotes, missing ruling lines, and visually aligned layouts that do not correspond to a simple regular grid. ESG reports often intensify these problems by combining narrative commentary with quantitative disclosures in compact layouts. Therefore, accurate table parsing requires not only

strong detection performance, but also the ability to preserve logical cell relations that will later support numerical validation and semantic interpretation.

2.2.3 Parsing frameworks and extraction paradigms

Practical document parsing frameworks have also received increasing attention in recent years. Libraries such as PyMuPDF, Unstructured, pdfplumber, Camelot, and Tabula remain widely used in applied pipelines because they offer relatively efficient access to text spans, page coordinates, and, in some cases, rule-based table extraction. Recent comparative studies have shown that no parser dominates across all document categories, because performance varies significantly depending on document type and on the specific subtask under consideration [1].

Recent end-to-end parsing frameworks such as Docling have attracted growing attention. Docling is an open-source document conversion framework that combines layout analysis and table structure recognition with a unified structured output suitable for downstream AI pipelines [2]. Its practical relevance is also supported by its ecosystem, which emphasizes PDF understanding, reading order, table preservation, OCR support, and integration with tools such as LangChain and LlamaIndex. This interest is consistent with recent benchmarking results on heterogeneous scientific PDFs, which show that end-to-end table extraction remains challenging when evaluated on diverse layouts, especially in terms of robustness and generalizability [23]. In that comparison, Docling performs well overall, confirming that parser selection depends on document variability and application needs rather than on a single universally superior solution.

In recent years, table extraction has increasingly converged toward a relatively standard paradigm: the combination of a table structure recognizer, often Table Transformer (TaTR), with an OCR component for recovering cell text.

Beyond the academic literature, industrial open-source projects are also advancing document parsing. A notable example is OpenDataLoader [1], which explicitly presents itself as a PDF parser designed to extract structured data for RAG pipelines, with emphasis on reading order, table extraction, bounding boxes. Its public materials also report benchmark-style comparisons against other practical parsers, including Docling and PyMuPDF4LLM, and emphasize hybrid OCR and

[1]OpenDataLoader is an open-source project for structured PDF parsing and RAG-oriented document extraction, available at `https://opendataloader.org/` .

AI-based processing for difficult layouts. Although this project does not yet appear to be supported by a dedicated peer-reviewed scientific paper.

Recent comparative studies have also examined multimodal large language models for table extraction, including GPT-4o, Phi-3 Vision, and Granite Vision 3.2 [16]. The evidence suggests a nuanced trade-off: conventional vision-based pipelines still tend to be more reliable for structural reconstruction, whereas multimodal models can be stronger in recovering textual cell content. This flexibility is promising for highly variable layouts, but in regulated domains it also raises concerns regarding reproducibility, hallucination, and limited control over structural fidelity.

2.3 Datasets for Document Understanding and Table Analysis

Before introducing the synthetic dataset developed in this work, it is useful to review the main benchmarks established for document understanding and table extraction.

The datasets introduced below can be organized into three groups: general-purpose document understanding benchmarks, domain-specific tabular structure datasets, and financial document reasoning benchmarks. Each group reflects a distinct stage in the document intelligence pipeline, ranging from the physical detection of tabular regions to the semantic interpretation of their content.

2.3.1 General-Purpose Document Understanding Benchmarks

The most established benchmarks for table parsing were designed to address one or more of three sub-tasks: *Table Detection* (TD), which locates the bounding box of a table within a page; *Table Structure Recognition* (TSR), which reconstructs the logical grid of rows, columns and extended cells; and *Functional Analysis* (FA), which identifies the semantic role of each cell, distinguishing column headers, row headers, and row header cells from data cells.

TableBank. TableBank [13] is one of the first large-scale benchmarks and was automatically created from Word and LaTeX documents sourced from academic archives. By taking advantage of the native markup of these source formats, the dataset was assembled without the need for manual annotation, enabling its large-scale implementation. However, all documents are represented as JPEG images rather than PDF files, and the tables they contain tend to be visually clean and struc-

turally simple, with clearly defined borders and regular grid layouts. TableBank therefore provides a useful baseline for detection and recognition models, but this benchmark cannot consistently predict performance on the irregular and dense tables that characterize financial and ESG reporting.

PubTabNet. PubTabNet [30], developed by IBM Research, enhances the TSR subtask by providing both the cropped image of each table and its structural annotation in HTML format. This representation captures the internal organization of the table: cells, rows, columns, merged cells and headers. However, PubTabNet tables are also drawn exclusively from scientific publications and feature the visual clarity and structural regularity typical of academic typesetting.

PubTables-1M. PubTables-1M [22], published by Microsoft alongside the Table Transformer (TATR) model, which is considered in this thesis, is currently one of the most authoritative references for TD, TSR and FA. Each table in the dataset is annotated in HTML, XML or LaTeX format, and includes the bounding box of each individual cell within the page, the multi-level heading structure, logical extension relationships and projected spatial hierarchies. However, the dataset remains focused on scientific documents.

2.3.2 Financial Table Benchmarks

Realizing that scientific publication datasets do not reflect the structural complexity of financial documents, the research community has developed dedicated benchmarks that replicate the properties of real corporate financial reports.

FinTabNet. FinTabNet [29], developed by IBM Research, is the most widely used financial table benchmark, consisting of annotated tables from real corporate annual reports. It is often regarded as a reference benchmark for evaluating table detection and structure recognition in the financial sector, exposing models to multi-level column headers, merged cells, numerical values with heterogeneous units and footnotes. Each table is annotated with its HTML structural representation and bounding boxes for both the table region and individual cells within the page. FinTabNet provides only structural annotations, but it does not provide the underlying truth for the semantic interpretation of cell values, making it insufficient for evaluating end-to-end extraction systems that must also

validate numerical correctness.

SynthTabNet. SynthTabNet [15], also developed by IBM Research, generates synthetic financial tables instead of annotating real documents. By synthesizing tables programmatically, it is possible to produce a large-scale dataset with perfectly accurate ground truth, without the inconsistencies that occur with human annotations. Synthetic generation also allows for controlled distribution of structural characteristics (table size, rowspan and colspan frequency, cell density and visual style), enabling researchers to construct datasets that are deliberately balanced across levels of complexity, rather than oriented toward the distribution found in real documents used as annotation sources.

Financial Document Reasoning Benchmarks

Also relevant are datasets that investigate the semantic reasoning required to answer questions based on financial documents. These benchmarks are not intended to assess table detection components; rather, they target the final reasoning capabilities of a document intelligence pipeline, particularly the RAG systems discussed earlier.

FinQA. FinQA [5] was built based on the annual reports of listed companies, and each example matches a document containing both narrative text and tabular data with a natural language question and its correct answer. The answer is accompanied by an explicit structured sequence of arithmetic operations, such as additions, subtractions, percentage changes, and year-on-year comparisons, thereby formalizing the problem of multi-step numerical reasoning.

DocFinQA. DocFinQA [18] extends the FinQA paradigm by operating on annual reports consisting of tens or hundreds of pages, rather than on pre-extracted document fragments. This makes DocFinQA especially suitable for evaluating end-to-end RAG systems, directly measuring the ability to identify the correct table or passage within a long document before performing any numerical computation.

The Gap Addressed by This Work

Existing benchmarks leave an important gap. Existing benchmarks usually focus either on table structure analysis or on question answering over financial documents. None of them evaluates the full pipeline, from PDF table extraction to numerical value verification against a detailed ground truth. This limitation is also consistent with the literature on document understanding. According to the VRDU, effective benchmarks should capture the complexity of real-world business documents, including visually rich layouts, diverse templates, and richer annotation schemes [24].

Fragmented scope of evaluation. More specifically, the *scope of evaluation* is fragmented. Benchmarks such as TableBank, PubTabNet, and PubTables-1M address table detection and structure recognition in isolation, typically on pre-cropped table images or single-page academic documents. FinTabNet extends this to financial tables but still limits annotation to bounding boxes and HTML structure, without verifying the semantic correctness of extracted cell values. FinQA and DocFinQA evaluate numerical reasoning through question answering, but they presuppose that the relevant evidence has already been correctly located and extracted; they do not assess the parsing stage. No existing benchmark evaluates the full pipeline in a unified way, from raw PDF input to layout reconstruction, table structure recovery, and cell-level value extraction.

Limited semantic richness of annotations. In addition, current annotations contain limited *semantic information*. In structural benchmarks, a cell is typically represented only by its position and textual content. Information such as the value type of a cell (whether it contains a raw number, a computed formula, or a graphical symbol), its unit of measure, or its importance for regulatory compliance is absent. These attributes are exactly the ones that determine how serious an extraction error is in ESG reporting.

Absence of computational dependencies. Furthermore, the ground truth does not represent the *computational dependencies between cells*. In ESG and financial tables, subtotals, percentage changes, and aggregate rows are derived from other cells through arithmetic formulas. Reasoning benchmarks such as FinQA encode these operations as external programs attached to question–answer pairs, but the dependency is not embedded in the table representation itself. This makes it

impossible to evaluate whether a parser preserves the internal consistency of derived values or to understand how an error in one cell affects the others that depend on it.

Constraints of real-report benchmarks. Also, most financial benchmarks are based on *actual company reports*, which introduces practical and methodological constraints. Copyright restrictions limit redistribution and reproducibility, while the visual conventions imposed by corporate templates reduce the diversity of table layouts represented in the dataset. SynthTabNet [15] partially addresses the layout diversity issue through programmatic generation, but its schema remains limited to structural attributes and does not include semantic cell annotations. Benchmarks derived from real reports are susceptible to annotation noise, since manual labeling of complex multi-level headers, merged cells, and footnotes is inherently error-prone and difficult to scale.

Limited attention to document-level context. Moreover, the *document-level context* receives limited attention. Real ESG reports interleave narrative text, section headings, tables, and visual elements within a coherent layout. Benchmarks that evaluate tables in isolation cannot assess whether a system correctly reconstructs the full document structure, including the relative ordering and spatial arrangement of heterogeneous content blocks. This is particularly relevant for downstream RAG applications, where the retrieval unit may span both a table and its surrounding explanatory text.

These limitations jointly motivate the experimental work presented in the following chapter, which introduces a synthetic ESG document generation pipeline producing PDF reports paired with a complete JSON ground truth that encodes hierarchical row groups, multi-level headers, formula-based aggregation rows, merged cells, heterogeneous value types, footnotes, criticality annotations, and narrative context within a unified schema. This synthetic setup ensures consistency between each rendered document and its annotation, avoids copyright-related restrictions, and makes it possible to control the structural complexity of the benchmark.

2.4 Evaluation Metrics for Document Understanding and Table Extraction

Because document extraction quality is critical, the choice of method or specialized software should be guided by performance. This requires evaluation metrics that are appropriate to the target task. This section reviews the main evaluation metrics used in document intelligence, highlighting their strengths and limitations for real-world applications such as ESG document analysis.

2.4.1 Character-Level and Word-Level Recognition Metrics

One of the first tasks is converting visual content into machine-readable text. Optical character recognition (OCR) systems are typically evaluated with complementary metrics that measure how accurately document text is reproduced.

Character Error Rate (CER)

The Character Error Rate quantifies the proportion of character-level edits required to transform the predicted text into the ground truth. To formally define the CER, given a ground truth string g and a predicted string p:

$$\text{CER} = \frac{S_c + D_c + I_c}{N_c}$$

where S_c, D_c, and I_c denote the number of character-level substitutions, deletions, and insertions, respectively, and $N_c = |g|$ is the total number of characters in the ground truth. The theoretical basis behind this is dictated by the Levenshtein distance [11]. CER is particularly informative for evaluating OCR on documents with any textual and character content. However, CER treats all characters equally and does not distinguish between an error in a numeric cell and an error in a white space character or punctuation mark.

Word Error Rate (WER)

Word Error Rate operates at the word level and follows the same formulation as CER. It is widely used in speech recognition and has also been applied to document understanding tasks involving

narrative text extraction. However, it has a clear limitation: a minor OCR artifact affecting one character is penalized in the same way as a completely wrong word.

Other text-oriented overlap metrics sometimes used in document understanding are **BLEU** (Bilingual Evaluation Understudy), which measures n-gram overlap between a candidate and one or more references, and **ROUGE** (Recall-Oriented Understudy for Gisting Evaluation), widely used in summarization to compare generated and reference text spans. In document intelligence, they can be useful for generated answers or extracted narrative passages, for example in financial question answering such as FinQA [5]. However, they play only a secondary role in this context, since token-overlap metrics may reward lexical similarity even when exact numerical values, layout dependencies, or table structure are not preserved.

For ESG and financial reporting, CER and WER are useful but incomplete because they ignore layout and structure. A table may achieve low error rates yet remain unusable if row, column, and cell relations are lost. They also fail to distinguish minor textual variations from critical numerical errors unless custom weighting is introduced.

2.4.2 Object Detection Metrics for Table Detection

Table localization is usually framed as an object detection task and evaluated through the spatial overlap between predicted and reference bounding boxes.

Intersection over Union (IoU)

Given a predicted bounding box B_p and a ground truth bounding box B_g, the Intersection over Union quantifies the degree of spatial overlap between the two bounding boxes:

$$\text{IoU}(B_p, B_g) = \frac{|B_p \cap B_g|}{|B_p \cup B_g|}$$

IoU ranges from 0 (no overlap) to 1 (perfect alignment) and a prediction is typically considered a true positive if $\text{IoU}(B_p, B_g) \geq \tau$, where τ is a threshold commonly set to 0.5 (the PASCAL VOC convention) or evaluated across multiple thresholds as in the COCO protocol.

Precision, Recall, and F1-Score

Among the most common metrics in document understanding are Precision, Recall, and F1-score. They are widely used in layout analysis, table detection, key information extraction, and cell matching tasks because they are simple, interpretable, and applicable across many subtasks. In document understanding, they are commonly used when predictions can be matched to reference items in a reasonably discrete way, for instance document objects, entities, or table cells. In particular, once predictions have been classified through an IoU threshold, TP, FP, and FN can be computed; these represent the numbers of true positives, false positives, and false negatives, respectively.

Precision measures the proportion of predicted tables that are correct, *Recall* the proportion of true tables detected, and the *F1-Score* their harmonic mean, providing a balanced summary measure:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

These metrics constitute the standard evaluation protocol in the table detection literature. The ICDAR Table Competition series, by the 2019 Competition on Table Detection and Recognition (cTDaR), established IoU-based precision, recall, and F1 as the primary measures for comparing table detection systems.

Mean Average Precision (mAP)

In multi-class detection benchmarks (e.g., tables, figures, formulas), mean Average Precision (mAP) provides an overall measure, with Average Precision (AP) computed as the area under the precision-recall curve for each class. However, AP is less interpretable than F1, and mAP is mainly suited to detection or retrieval tasks rather than aligned comparison of structured document representations. More importantly, mAP is designed for prediction settings such as object detection and information retrieval, whereas the task of interest in document understanding often requires an aligned comparison between structured document representations.

2.4.3 Table Structure Recognition Metrics

In table structure recognition, the goal is to reconstruct the internal organization of the table by identifying rows, columns, cells, and spanning relationships.

Tree-Edit-Distance-based Similarity (TEDS)

Tree-Edit-Distance-based Similarity (TEDS), introduced with the PubTabNet benchmark by Zhong et al. [30], is one of the most widely used metrics for table structure recognition. It was designed to evaluate image-based systems that generate tables in HTML-like structural form.

The tree edit distance $\Delta(T_g, T_p)$ between the ground truth tree T_g and the predicted tree T_p is the minimum cost sequence of node-level edit operations — insertion, deletion, and substitution — required to transform one tree into the other.

TEDS normalizes this distance by the maximum tree size, producing a similarity score in $[0, 1]$:

$$\text{TEDS}(T_g, T_p) = 1 - \frac{\Delta(T_g, T_p)}{\max\big(|T_g|, |T_p|\big)}$$

where $|T|$ is the number of nodes in tree T. A score of 1 indicates a perfect match, while 0 indicates complete dissimilarity.

Strengths and Limitations. TEDS has several strengths: it handles tables of different sizes, captures hierarchical structure through the tree representation, and accounts for spanning cells via node attributes. However, it is sensitive to the chosen HTML serialization: structurally identical tables may receive different scores if represented with different markup conventions (e.g., `<thead>` versus bare `<tr>`). In addition, TEDS treats all cells equally, so errors in header cells are penalized in the same way as errors in data cells, and numerical errors are not distinguished from textual ones. It also does not natively support cell importance or domain-specific weighting.

Grid Table Similarity (GriTS)

Grid Table Similarity (GriTS), introduced by Smock et al. [21] for the PubTables-1M benchmark, was proposed to address some limitations of tree-based table evaluation. Instead of representing a table as a serialized tree, GriTS evaluates it in its natural grid form. The table is modeled as a

matrix of rows and columns, where each position contains the content of the corresponding cell. When a cell spans multiple rows or columns, its content is replicated across all covered positions so that the grid remains complete.

GriTS evaluates similarity through three complementary aspects: topology, which checks whether cells occupy the correct grid positions independently of content; content, which verifies whether the correct text appears in the cells; and location, which assesses whether the correct content is placed in the correct grid position. For each aspect, precision- and recall-like scores are computed from the overlap between predicted and reference grids and then combined through their harmonic mean.

Strengths and Limitations. The main advantage of GriTS is that it provides a direct and spatially meaningful evaluation of tables, avoiding the dependence on HTML or tree serialization that affects other metrics. Its separation into topology, content, and location also improves interpretability, since it helps distinguish structural errors from content extraction errors. However, like TEDS, GriTS assigns the same importance to all grid positions and therefore does not reflect that some cells, such as key financial values, may be more critical than others.

2.4.4 Cell-Level Extraction Metrics

When the evaluation focus shifts from structural reconstruction to the accuracy of individual cell values, simpler but more targeted metrics are used.

Exact Match (EM)

Exact Match is the strictest cell-level metric, assigning a score of 1 if and only if the predicted cell value is identical to the ground truth value, and 0 otherwise:

$$\mathrm{EM}(g, p) = \mathbb{1}[g = p]$$

While straightforward and unambiguous, Exact Match is sensitive to trivial formatting differences. For example, the values “1,234.56” and “1234.56” would be considered a mismatch despite representing the same number. This rigidity makes EM problematic for financial documents, where number formatting conventions vary across locales and reporting standards.

Normalized Exact Match

To reduce the formatting sensitivity of raw Exact Match, a normalization step is often applied before comparison. Common operations include removing whitespace and punctuation, converting to lowercase, stripping currency symbols and unit suffixes, and parsing numerical strings into canonical floating-point form. This variant is widely used in table extraction, but the absence of a standard normalization protocol leads to inconsistencies across benchmarks and evaluation settings.

Cell-Level F1

An alternative to binary Exact Match is a token-level or character-level F1 score computed for each cell. The predicted and ground truth cell values are tokenized (by whitespace or characters), and precision, recall, and F1 are computed based on the overlap of token sets:

$$P_{\text{cell}} = \frac{|T_g \cap T_p|}{|T_p|}, \quad R_{\text{cell}} = \frac{|T_g \cap T_p|}{|T_g|}, \quad F_{1,\text{cell}} = \frac{2 \cdot P_{\text{cell}} \cdot R_{\text{cell}}}{P_{\text{cell}} + R_{\text{cell}}}$$

where T_g and T_p are the token sets of the ground truth and predicted values. This provides partial credit for approximately correct extractions, but like all token-overlap metrics, it does not capture the semantic significance of different types of errors.

Limitations of Existing Metrics for ESG Document Understanding

However, in ESG and financial document understanding, several limitations remain. The metrics reviewed above are useful for specific document intelligence subtasks, they are less suitable for evaluating the understanding of complete ESG documents. In this setting, the goal is not only to reconstruct isolated text fragments or table structures, but to assess whether a system correctly captures narrative sections, heterogeneous tables, and the relationships between their elements.

Uniform weighting of all elements. Most existing metrics assign the same importance to every error. In ESG, extraction errors have very different consequences depending on the semantic and regulatory importance of the affected element. Misreading a key indicator such as Scope 1 emissions is far more serious than misreading a secondary label or a formatting element. Likewise, an incorrect header can compromise the interpretation of all the values below it. Yet TEDS assigns the same

cost to a node substitution regardless of whether the affected cell contains a critical KPI or a minor annotation, and GriTS evaluates all grid positions with equal importance. For regulated settings, where the cost of misreporting specific indicators is disproportionately high, for this reason, existing metrics do not reflect the real risk associated with reporting errors.

Absence of computational dependency modeling. Another limitation is that current metrics ignore computational dependencies between cells. In ESG and financial tables, many values are derived from other cells through formulas. An error in one source cell can therefore propagate through its dependents, potentially corrupting an entire column of derived values. Standard metrics cannot capture this: they count downstream errors separately, but they do not represent the underlying dependency that connects them. This makes them less suitable for scenarios in which internal consistency is essential. In ESG reporting, where auditors routinely verify that totals match their constituents, the ability to detect and penalize broken computational relationships is essential for any evaluation that aims to predict real-world usability.

Separation between structure and content evaluation. Another limitation is the separation between structure and content. Metrics such as TEDS and GriTS mainly assess table layout, while CER, WER, and Exact Match focus on content accuracy. This fragmentation is particularly problematic for ESG documents, where a table may be structurally perfect yet contain incorrect cell values, or conversely, may have all values correct but placed in the wrong grid positions due to a misaligned header. The evaluator ideally needs a single framework that decomposes quality into interpretable sub-dimensions, such as structural fidelity, content accuracy, and semantic correctness, while still producing an aggregated score that reflects overall extraction quality.

Lack of precision–recall decomposition at the table level. The metrics reviewed above generally produce a single similarity or accuracy score per table or per document. They do not distinguish between precision and recall, even though these two measures reflect different types of failure. Recall reflects the system's ability to recover all ground-truth elements, whereas precision reflects its tendency to introduce hallucinated or spurious content. These two dimensions are especially important in regulated ESG workflows: a system with high recall but low precision may fabricate values that appear plausible but have no basis in the source document, while a system with high

precision but low recall may silently omit critical indicators. Neither TEDS nor GriTS natively supports this distinction, and Exact Match reduces the comparison to a binary outcome per cell without any accumulation into separate coverage and correctness measures.

Neglect of document-level composition. Finally, existing metrics pay limited attention to document-level structure. ESG reports combine headings, paragraphs, tables, and other visual blocks within a coherent layout. The order and function of these elements influence both interpretation and downstream use. Metrics applied only to isolated tables or text passages cannot assess whether the full document structure is correctly reconstructed. A system may extract individual tables correctly while still producing a poor document representation if it misorders sections, misclassifies headings, or fails to preserve the relation between tables and surrounding text.

Taken together, these limitations define the requirements for a more comprehensive evaluation methodology. For this reason, the following chapters introduce a domain-specific hierarchical evaluation framework. The framework combines criticality-weighted scoring, verification of formula dependencies, a unified multi-level comparison of structure and content, explicit precision and recall at each level of granularity, and document-wide block alignment for the joint evaluation of textual and tabular elements.

3. Experimental Setup

3.1 Agent RAG Orchestration

The experimental work began with the implementation of an agent orchestration layer using LangChain. LangChain was selected because its modular design made it easy to extend the system step by step, from a basic chat interface to a retrieval-augmented and agent-based pipeline. Starting from a minimal chat-model interface, the prototype was progressively extended with retrieval, memory, and tool-calling capabilities as the experimental setup evolved.

Using LangChain's extensive integration library, a RAG agent was quickly assembled to interact with an LLM hosted on our local Ollama server. This was especially useful in the ESG domain, where outputs need stronger validation. In particular, LangChain supports schema-constrained generation through Pydantic and HITL steps that allow manual approval during execution.

For document ingestion, three categories of loaders were especially relevant: PDF loaders (with the Unstructured integration for layout-aware parsing), online loaders for regulatory sources such as ESMA publications and CSRD guidelines, and spreadsheet loaders for ESG data in Excel or CSV format. The chunking phase relied on `recursive character splitting` and `token-based splitting`, producing `Document` objects with `page_content` and `metadata` fields important for ESG traceability. For embedding, the open-source `bge-small-en` model from the BGE family was selected for its balance between retrieval quality and computational efficiency, and Chroma Database was adopted as vector store for its minimal configuration requirements.

Identified Limitations and Shift Toward Document Analysis

Early experiments with the prototype revealed several limitations that progressively shifted the research focus from retrieval orchestration toward document analysis quality.

The first limitation concerned PDF segmentation. Many document loaders in the LangChain ecosystem adopt a page-based strategy by default, treating each page as a self-contained unit. This approach worked reasonably well for presentation-style PDFs, where `PyPDFLoader` and texttttRecursiveCharacterTextSplitter gave acceptable results. It was much less effective for continuous-text

documents such as sustainability reports, because relevant information often extends across multiple pages and gets split into disconnected chunks.

A related issue involved chunking calibration. Fixed-boundary splitting can create artificial segment boundaries, and even moderate changes to `chunk_size` or `chunk_overlap` significantly affect retrieval. A better option is semantic chunking. Tools such as LlamaIndex's `Semantic Splitter` try to split the text where semantic transitions actually occur, instead of relying only on fixed size boundaries. However, even with improved text segmentation, tables remained problematic: linear chunking does not preserve the spatial and relational structure of rows, columns, and headers. To address this, LlamaIndex proposes a `RecursiveRetriever`[1], in which a parent node contains a compact table summary while a child node points to a structured query engine that accesses the table directly through pandas or SQL, delegating exact value extraction to a structured backend.

These observations motivated the consideration of table-aware retrieval extensions such as Table-RAG and Graph-RAG, both discussed in Chapter 2.1.3. A Table-RAG configuration would allow the agent to delegate filtering and aggregation to a relational backend, while a Graph-RAG configuration would better handle evidence distributed across multiple tables and reporting dimensions. LangChain already offers integrations for SQL and graph databases to support such extensions.

However, both strategies depend critically on the quality of the upstream document analysis phase. In SQL-oriented settings, the extracted table must preserve the correct schema—including headers, column boundaries, merged cells, units, and cell values—for executable queries to be reliable. In graph-based settings, the same requirement applies: entities and relationships must be correctly reconstructed from the source document. This dependency confirmed that robust document parsing and table reconstruction are prerequisites for any advanced retrieval strategy, and motivated the experimental focus on document analysis described in the following sections.

[1]The LlamaIndex documentation provides an example in which a table is represented through a concise summary node connected to an SQL/Pandas engine for targeted queries. `https://github.com/run-llama/llama_index/blob/main/docs/examples/query_engine/pdf_tables/recursive_retriever.ipynb`.

3.2 Synthetic Dataset Generation

3.2.1 Generator System Pipeline

As discussed in Chapter 1, ESG document understanding requires systems able to extract structured information from visually complex PDF documents. Evaluating such systems requires datasets that reflect the structural complexity of real ESG reports while preserving an unambiguous ground truth for every value. Manual annotation is costly and error-prone, especially in the presence of multi-level headings, formulas, and footnotes.

For this reason, this work implemented a fully automated end-to-end pipeline for generating synthetic ESG documents in PDF format, each paired with a complete JSON ground truth. The pipeline produces documents with realistic narrative text, complex tables, and variable visual styles, while preserving exact knowledge of each cell value, formula, and structural relation. As shown in Figure 3.1, the process consists of successive stages that transform abstract templates into fully rendered ESG-like reports.

For each generated sample, the pipeline outputs two paired artifacts: a PDF document, used as input for document understanding systems, and the corresponding JSON ground truth. The JSON preserves the complete structural semantics of the document, including the value, type, format, unit, formula expression, hierarchical position, and criticality annotation of each cell.

Step 1: Table Template Design and Randomized Generation

The process starts from a set of table templates used to create and randomize documents. This makes the dataset configurable, since different table types and structural patterns can be selected for specific use cases. These templates are not simple row–column grids, but encode hierarchical row groups, multi-level headers, formula-based aggregation rows, merged cells with `rowspan` and `colspan` attributes, and heterogeneous column types. In the ESG setting considered here, a limited set of diverse and structurally complex table types was derived from ESG and financial reports published by the Credem Banking Group in order to provide the dataset with a reasonable degree of realism.

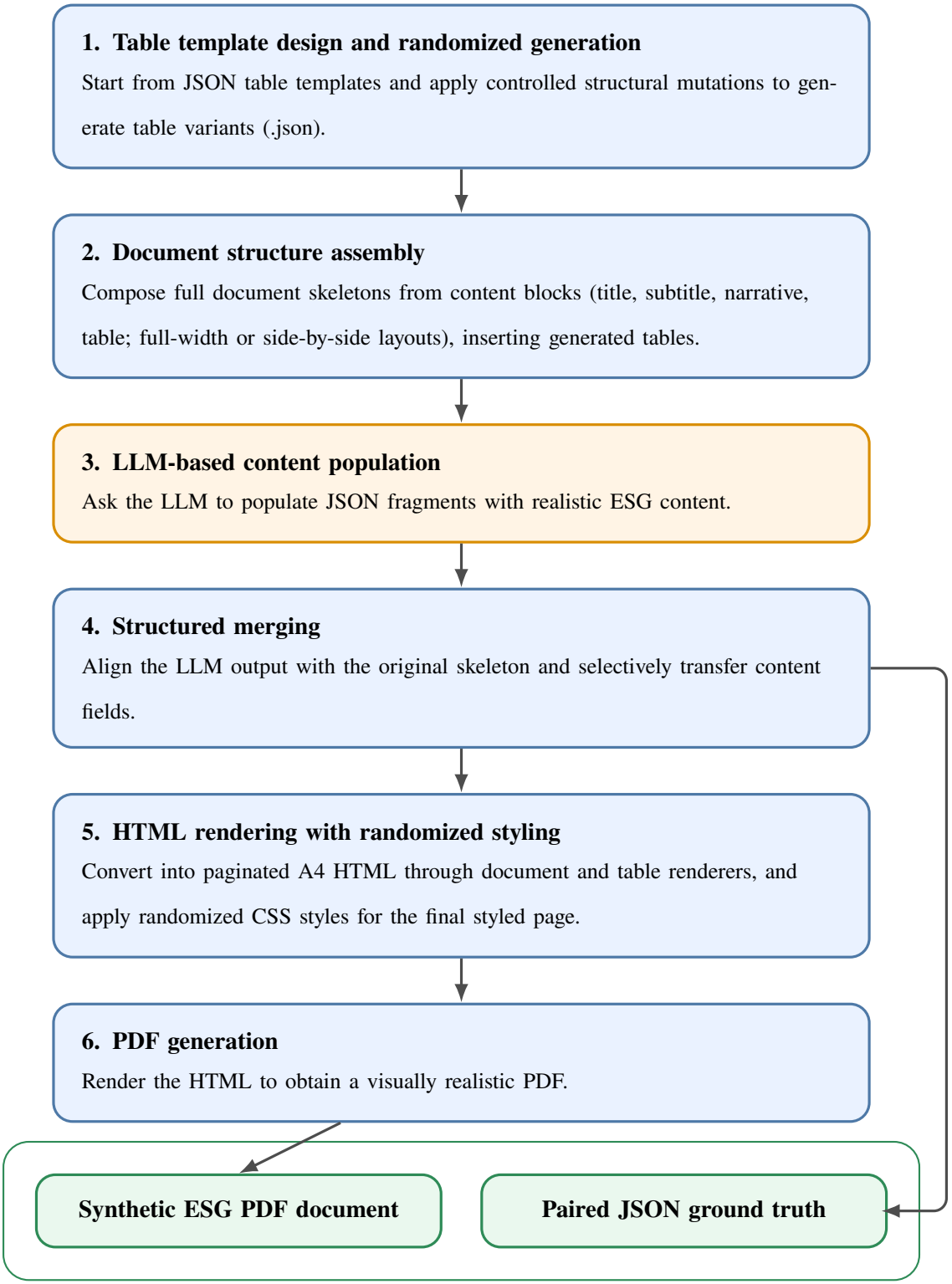

Figure 3.1: Synthetic dataset generation pipeline. Automated generation of ESG-like PDF documents paired with complete JSON ground truth for evaluation.

The JSON Table Schema. Each table is represented as a JSON object that captures both logical content and rendering semantics. Root-level fields include `title`, `footnotes`, `columns`, `headerRows`, and `rows`. Column definitions contain an `id`, a `label`, and a `criticality` annotation; header rows support `colspan` and `rowspan` for multi-level headers and may also carry `criticality`.

Each cell within a row specifies a `valueType` chosen from a controlled vocabulary of five types:

- **`text`**: string labels such as category names, row descriptors, or qualitative indicators.
- **`numerical`**: quantitative values stored as a structured object `{raw, format, unit}`, where `raw` contains the number, `format` specifies the locale-dependent rendering convention (e.g., `en-US` for comma-separated thousands or `it-IT` for period-separated thousands), and `unit` indicates the measurement symbol ($, €, %).
- **`boolean`**: binary indicators, for example to indicate presence or absence.
- **`indicator`**: graphical indicators stored as a structured object `{category, label, symbol}`, mirroring the structured design adopted for numerical values. The `category` field identifies the semantic family to which the indicator belongs (e.g., `trend`, `status`, `financial`, `priority`, `approval`); the `label` field specifies the meaning within that family (e.g., `growth`, `check`, `profit`); and the `symbol` field records the actual Unicode glyph rendered in the PDF (e.g., ↗, ✓). This three-level decomposition separates the visual recognition target (symbol) from the semantic interpretation target (label) and the categorical constraint (category), enabling more granular evaluation diagnostics.
- **`formula`**: calculated cells whose value is derived from other cells via an expression such as `= {a1_f1} + {a2_f1} + {a3_f1}`, where the tokens in braces reference other cell identifiers.

Every semantically relevant table element also carries a `criticality` annotation with a severity level (`critical`, `high`, `medium`, or `low`) and a human-readable reason. In the current schema, this metadata is attached not only to body cells but also to column descriptors and header or row-header cells whenever they contribute to table interpretation. This supports downstream weighting of extraction errors, distinguishing, for example, between a misread aggregation total, an omitted KPI label, and a secondary descriptive field.

Structural Randomization. To avoid overfitting evaluation to a fixed set of table structures, a dedicated generation script applies a controlled sequence of structural mutations to the base template after assigning globally unique cell identifiers. Mutations are drawn from a weighted pool and include adding or removing data rows within a group, adding or removing columns with corresponding updates to formula expressions, changing the `valueType` of an entire column, modifying units or number formats across numerical cells, adding or removing aggregation rows, and, where supported, adding or removing dimension groups or hierarchical categories.

Each mutation preserves strict invariants: formula expressions must continue to reference exactly the relevant data cells, and the table must remain rectangular, with every row containing exactly as many cells as there are columns. The result is a family of structurally valid and internally consistent tables exhibiting meaningful variation in row and column counts, value-type combinations, formula topologies, and hierarchical depth. Each generated variant is stored as JSON, preserving the full schema for ground-truth purposes.

Step 2: Document Structure Assembly

Real ESG reports combine narrative text, tables, and hierarchical headings, often through mixed layouts such as side-by-side text and tables or paired tables. To reproduce this organization, the second stage assembles complete JSON documents from content blocks. The schema supports six block types within the `content` array: `title`, `subtitle`, `narrative`, `table`, `table_pair`, and `text_table_sidebyside`. Each document also includes top-level metadata, namely `doc_id`, `file_name`, and `doc_category`.

The dataset includes both concise documents composed of 4–5 blocks, suitable for focused extraction tests, and longer documents with 25–60 blocks, designed to simulate multi-section sustainability reports with interleaved tables and extended narrative passages. For each document, the generator selects previously created table variants and inserts them into the appropriate content slots. A tracking mechanism ensures balanced table usage and records the provenance and placement of each inserted variant.

At this stage, all textual fields, including titles, subtitles, narrative paragraphs, table headers, and cell values, remain empty or set to `null`. The result is a fully specified structural skeleton without semantic content. This separation between structure and content fixes the structural ground truth

deterministically before any stochastic content generation is applied.

Step 3: LLM-Based Content Population

The third stage fills the structural skeleton with realistic ESG content through queries to a large language model, transforming the template into a plausible document simulation. Rather than simply completing empty fields, this stage uses generative AI to produce realistic reporting instances.

This is the only non-deterministic phase of the pipeline, and its design therefore required careful prompt engineering to balance output quality, structural fidelity, and robustness to model failures.

The Population Prompt. The core of this phase is a system prompt instructing the LLM to act as an ESG analyst and populate a JSON template with coherent dummy data, following the generative information extraction paradigm [26]. The prompt defines field-specific requirements: metadata fields must receive plausible document identifiers and categories; title and subtitle blocks must contain concise ESG-related headings; and narrative blocks must contain analytical prose of about 200–250 words on topics such as emissions, energy consumption, diversity metrics, governance practices, and performance targets.

Table-cell instructions are type-specific. Text cells receive realistic labels; numerical cells must populate the `raw` field while respecting existing `format` and `unit` constraints; Boolean cells receive `true` or `false`; indicator cells are filled with a controlled `{category, label, symbol}` object, with the additional constraint that all indicator cells in the same column share the same `category`; and footnote text is generated only for pre-existing footnote symbols. Strict output constraints require raw JSON only, without Markdown formatting, explanatory text, or modifications to field names, nesting structure, or existing identifiers.

Prompt Construction and Processing. Before submission to the model, the JSON document is pre-processed to remove fields irrelevant to content generation, such as `criticality` annotations, `width` directives, and `table_source` references. This reduces prompt length and avoids distracting the model with unnecessary fields. The cleaned JSON is serialized in compact form and appended to the system prompt.

For long documents, the input may exceed the model context window; a chunking strategy

therefore divides each document into smaller segments processed through independent requests, whose outputs are later merged through a dedicated deduplication phase. Because LLM outputs may contain malformed JSON, truncation, or omitted fields, a retry mechanism attempts generation up to five times per document, with a configurable delay under high-load conditions. Each response is validated through JSON parsing, and only successfully parsed outputs proceed to the next stage.

In the implemented setup, the model is accessed through the Ollama API, enabling the use of locally hosted open-source models. Model selection, temperature, and endpoint URL are configurable through command-line arguments or environment variables, allowing adaptation to different hardware settings and model choices.

Step 4: Structured Merging of Template and LLM Output

Once the LLM has generated the document content, the next stage merges it with the original structural template, producing a *compiled* document that combines semantic content with structural precision. Some fields were deliberately hidden from the LLM during content generation, including formula expressions, layout widths, structural metadata, and criticality annotations. These fields must nevertheless be preserved in the final ground truth. For this reason, the merging script performs a selective replacement strategy, transferring only model-generated content into the corresponding positions of the original template. Criticality annotations attached to body cells, column descriptors, and header cells are therefore always restored from the template rather than generated by the model.

Matching Strategy. A significant challenge at this stage concerns the alignment between the original template and the LLM output. The model does not always preserve cell identifiers exactly and may introduce suffixes or minor variations. The merging procedure therefore relies not only on exact identifier matching but also on positional alignment strategies, reducing errors caused by identifier inconsistencies or deduplication artifacts. This alignment is applied consistently across all document levels, including top-level blocks, column definitions, header cells, and data cells.

Selective Value Transfer. The merging process intentionally restricts the fields that can be overwritten. Narrative text, titles, and subtitles are transferred from the LLM output to fill previously empty fields. For table cells, only the `value` field is transferred: specifically, the `raw` component

of numerical values, the direct value for Boolean and text cells, and the full `{category, label, symbol}` object for indicator cells. Formula cells are explicitly excluded, since their values are computed deterministically from the formula expression and the referenced cells, preventing the LLM from altering the computational integrity of the table. Header labels and footnote text are transferred unconditionally.

To preserve document-level consistency, an additional normalization step is applied to numerical formatting. Different tables within the same document may contain mixed locale conventions, such as `en-US` and `it-IT`. The normalization algorithm collects all non-null format strings in the document, identifies the most frequent convention, and standardizes numeric cells accordingly. Null formats, which denote unformatted integers, are left unchanged.

Step 5: HTML Rendering with Randomized Styling

Once post-processing is completed, a clean and consistent JSON ground truth becomes available. To generate the PDF files while preserving layout consistency and formatting, each populated JSON document is first converted into a visually structured HTML page through two main modules. The `Document JSON To HTML` renderer processes each content block sequentially, generating headings for titles and subtitles, paragraph containers for narrative text, and table containers for the supported layouts. For `table_pair` and `text_table_sidebyside` blocks, elements are arranged in adjacent columns through CSS, and the page is sized for A4 paper with configurable margins. The `Table JSON To HTML` renderer converts each table definition into HTML with full support for merged cells, formula resolution, type-specific formatting, indicator rendering through the `symbol` field, and footnote display. Formula expressions are resolved at render time by recursively evaluating referenced cells, so that the HTML and subsequent PDF display computed totals and aggregations even though the JSON stores the expressions rather than resolved values.

Visual Style Randomization. A key requirement of the system is to avoid a uniform visual appearance. Real ESG reports vary widely in typography, color choices, table borders, title treatments, and spacing conventions. To reproduce this diversity, the pipeline includes a style-randomization module that generates a complete CSS stylesheet by sampling from a constrained configuration space. Font pairings span classic serif, sans-serif, mono-serif, and mixed combinations. Color

palettes are drawn from many corporate-style schemes specifying coordinated values for background, text, accents, headers, borders, and alternating rows. Additional variation is introduced through table styling, including full-grid, horizontal-only, minimal, boxed, striped, executive, and borderless layouts, as well as title treatments such as underline, accent bar, background band, plain, uppercase with letter-spacing, and thin-rule variants. Further parameters include base font size, line height, cell padding, page margins, border thickness, and relative scaling of headings, subtitles, narrative text, and tables. All choices remain constrained within a corporate design register, so that generated documents remain professionally formatted despite their randomized composition.

Step 6: PDF Generation

At this stage, the HTML file is converted into a PDF document through Playwright, a headless browser automation library that exposes the native rendering and PDF printing capabilities of browsers such as Chromium. This approach was preferred over Python libraries dedicated to PDF generation because it more faithfully preserves complex CSS-based layouts, including multi-column layouts, table styling, page-break behavior, and font rendering. Preliminary tests with WeasyPrint, for example, frequently truncated tables and other page elements.

The conversion is configured for A4 format and includes an optional setting controlling whether table headers are repeated at the top of each page when a table extends beyond a page boundary. Since this is not always a common convention in financial reports, table headers were not repeated in the generated dataset, increasing document complexity. Rendering fidelity is especially important for benchmark generation because an atypical PDF renderer could introduce systematic distortions that would affect the evaluation of document understanding systems.

3.2.2 Ground Truth and Dataset Characteristics

The benchmark includes not only the visual document representation, but also a fully structured reference enabling automatic and reproducible assessment. The JSON ground truth encodes the full semantic and structural content of each document. As discussed in the previous subsection, this representation supports evaluation at multiple levels of granularity, including cell-level value extraction accuracy, table-level structural reconstruction fidelity, and document-level content completeness.

Criticality-Weighted Evaluation. A distinctive feature of the schema is the `criticality` annotation attached to semantically relevant table elements, not only to body cells. Body cells, column descriptors, and header or row-header cells may all carry a severity level (`critical`, `high`, `medium`, `low`) reflecting their importance for downstream ESG analysis. In general, key numerical data are annotated as `critical` or `high`, aggregated formula results or visual indicators as `medium`, and descriptive labels or secondary headers as `low` or `medium`, unless they identify a particularly relevant KPI, reporting period, or business dimension.

This makes it possible to define a weighted scoring methodology in which errors on high-criticality elements are penalized more heavily than those on low-criticality ones. The effect extends beyond numerical body cells: misreading a crucial column label or row-header that determines the meaning of an entire table region can be penalized more strongly than an error on a secondary descriptive element. Such weighted metrics are more informative than unweighted accuracy for assessing real-world readiness, since in a regulatory context the cost of misreporting a KPI is much higher than the cost of misreading a description.

Scalability, Provenance, and Error Traceability. The tracking system associated with dataset generation records the provenance of each document, including the structural model from which it was derived, the instantiated table variants, and their placement within the final content. This metadata supports more detailed benchmark analyses, for example by identifying whether specific layouts, structural templates, or table families systematically lead to lower extraction quality.

The evaluation framework was designed not only to produce final scalar scores, but also to generate structured reports that make results fully inspectable. For each evaluated document, the pipeline exports the overall score, the text and table sub-scores, document-level table precision, recall, and F1, and the complete block alignment between ground truth and prediction, including matched, missing, extra, and merged blocks, type mismatches, text similarity, and layout agreement. For table blocks, reports also include differences in columns, header rows, and footnotes, row-level alignment, canonical row–column localization of each discrepancy, cell criticality, partial cell score, exact differing fields, and the mapping between ground-truth and predicted cell identifiers. A further diagnostic stage produces analogous reports for the structural, content, and semantic components of the metric, enabling both quantitative comparison and detailed qualitative analysis of failure modes.

3.2.3 Comparative Advantages over Existing Benchmarks

The dataset presented in this work is not intended as just another collection of annotated tables, but as a benchmark addressing several limitations of the datasets discussed in Chapter 2. Existing benchmarks often focus on a single aspect of the problem, such as table structure recognition or document-level financial question answering. By contrast, the proposed dataset combines these perspectives within a single evaluation setting that is more closely aligned with ESG and sustainability report understanding. The schema was designed specifically for ESG-related content; in particular, the inclusion of merged cells, graphical indicators, Boolean policy-related values, and linked explanatory notes makes the benchmark more representative of sustainability reporting practice.

Formal Consistency Through Procedural Generation. The synthetic nature of the dataset provides an important advantage in terms of formal consistency. Because both the PDF and the JSON annotation are generated by the same process, the benchmark avoids many of the ambiguities and annotation errors typical of manually labeled datasets. The result is not only scalable and customizable, but also formally validated and suitable for detailed error analysis.

Joint Structural and Semantic Representation. A first distinctive feature is the joint representation of table structure and cell semantics. Datasets such as TableBank, PubTabNet, PubTables-1M, and, in a more domain-specific setting, FinTabNet, mainly focus on the physical organization of tables. By contrast, the proposed dataset assigns a semantic type to each cell, distinguishing among text cells, raw numerical values, calculated formulas, Boolean indicators, and graphical indicators. This makes it possible to evaluate not only whether a model reconstructs the visual table structure, but also whether it correctly interprets the informational role of each element.

Formula Dependencies, Layout, and Narrative Context. Another important contribution is the explicit modeling of *computational dependencies*. Formula cells are encoded as expressions that reference other cells through unique identifiers, supporting the identification of subtotals and aggregate rows. This is especially relevant in ESG and financial reporting, where derived indicators must be distinguished from atomic observations. Reasoning-oriented benchmarks such as FinQA and DocFinQA include arithmetic programs, but these remain external to the table structure; here,

the computational relation is embedded directly in the ground-truth representation.

The benchmark preserves the *narrative context* of the document. Headings, subheadings, and paragraphs are stored alongside tables within a unified representation, enabling evaluation of systems that must jointly process structured and unstructured information.

Footnotes, Header Hierarchies, and Criticality-based Scoring. The schema explicitly models *footnotes* and *multi-level headers*, which are often essential for interpreting ESG tables. Footnotes are represented as relational objects, and the schema distinguishes between physical columns and logical header rows. Unlike PubTabNet, where part of this structure remains implicit in HTML markup, the proposed schema makes the header hierarchy explicit and requires full structural coverage without gaps, so that each header cell correctly covers its declared column range through `colspan` and `rowspan`. In addition, each semantically relevant table element carries a criticality level reflecting its importance for downstream ESG analysis. This makes it possible to penalize errors on critical values and contextual structural labels more heavily than errors on secondary elements, which is more realistic for regulatory scenarios.

End-to-end benchmarking perspective. Finally, the dataset and evaluation methodology form a coherent benchmarking system. The same schema used to generate the ground truth is compatible with a document-level evaluation framework able to compare heterogeneous blocks, weight errors by semantic criticality, and verify formula consistency. This goes beyond benchmarks that evaluate only table structure or only downstream QA responses and supports end-to-end evaluation of ESG document understanding systems, from layout reconstruction to extraction of semantically meaningful values.

Containerized Deployment and Reproducibility

The synthetic document generation pipeline was packaged and executed inside a Docker container to ensure a reproducible and portable runtime environment, reducing inconsistencies across machines and deployment settings. The pipeline is implemented as a single parameterized entry-point script supporting both full end-to-end execution and selective execution of individual stages. This is useful for debugging, testing, and incremental validation, since the artifacts produced at each step can be

inspected separately rather than remaining hidden within the execution flow.

The container also defines a clean execution boundary through explicit inputs, outputs, and runtime parameters, making the pipeline more compatible with microservice-oriented deployment, where components are expected to be self-contained and easy to orchestrate. The same principle was adopted for the evaluation framework described in Section 3.3, which is also packaged as a container, accepts JSON files as input, and produces both overall and component-specific scores together with detailed evaluation reports. Docker is therefore used as an enabling mechanism for reproducibility, modular execution, artifact-level observability, and robust integration of both generation and evaluation within a scalable document-processing architecture.

3.3 Domain-Specific Hierarchical Evaluation Framework

Given the generation pipeline and the ground-truth schema, the remaining component of the benchmark is the evaluation methodology. The proposed metric quantifies how accurately a document understanding system reconstructs the structured content of an ESG document. It operates at multiple levels of granularity, from cell fields to complete documents, and incorporates domain-specific aspects such as criticality weighting and formula dependency checking.

The evaluation takes as input two JSON representations of the same document: the ground truth G and the prediction P. It produces:

1. An overall score $S \in [0, 1]$ that measures document-level accuracy
2. sub-scores for text blocks (S_{text}) and table blocks (S_{table}), with the latter decomposed into document-level precision (P_{doc}), recall (R_{doc}), and F1 (F_1^{doc}),
3. A detailed difference report that identifies every mismatch at the block- and cell-level

The framework distinguishes between textual blocks and tabular structures. Text blocks are mainly scored with similarity-based measures, whereas tables are evaluated with precision and recall, which better capture omissions, incorrect values, and hallucinated content.

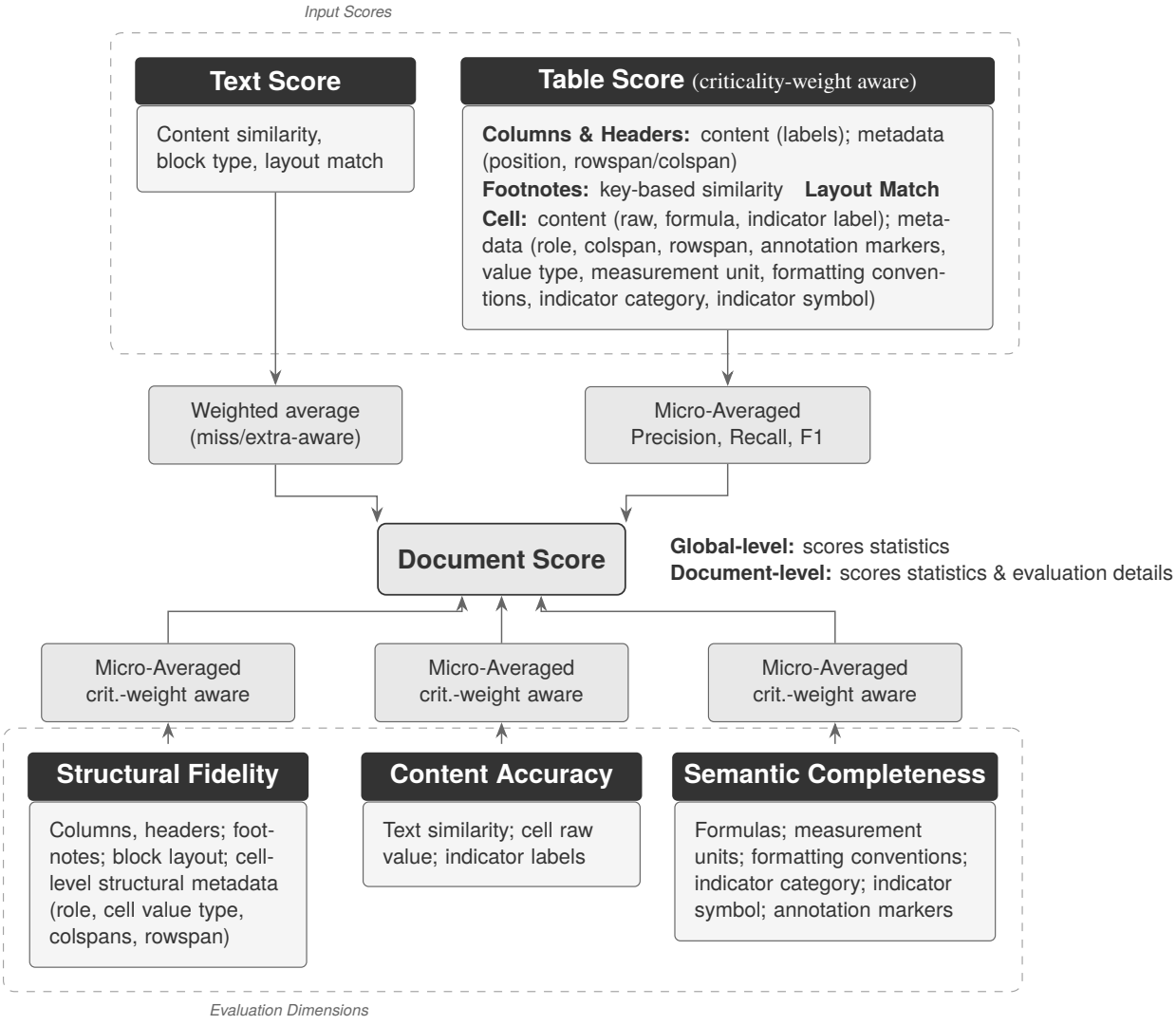

Figure 3.2: Domain-specific hierarchical scoring schema of the evaluation framework, showing the metrics aggregation for the overall document score.

Document-Level Structure: Flattening and Block Alignment

Both the ground truth and the prediction represent a document as an *ordered sequence* of content blocks: $D = (B_1, B_2, \ldots, B_n)$ where each block B_i is associated with a type

$$t_i \in \{\texttt{title}, \texttt{subtitle}, \texttt{narrative}, \texttt{table}, \texttt{table\_pair}, \texttt{text\_table\_sidebyside}\}.$$

Composite structures such as `table_pair` and `text_table_sidebyside` are flattened during preprocessing into atomic blocks. The evaluator therefore always operates on a linearized document whose elements are either textual blocks or table blocks.

Sequential Alignment with Lookahead. Because a system may omit blocks, introduce extra ones, misclassify their type, or split a single block into several fragments, the evaluator cannot assume exact positional correspondence. It therefore uses a *sequential alignment* procedure with lookahead and block merging.

Content Similarity Function. Alignment relies on a *content similarity function* rather than on block type alone. For textual blocks, similarity is computed on normalized text; for tables, it is computed on a lightweight structural signature derived from column labels and row patterns through `SequenceMatcher`. This content-aware design makes the alignment robust, because it can still associate blocks whose semantic content is the same even when their predicted type is different.

Block Merging for Split Detection. A common failure mode is *block splitting*, where a single ground-truth block is fragmented into consecutive predicted blocks. To handle this, the evaluator can merge consecutive predicted blocks and compare their concatenated content with a single ground-truth block. Merging is only attempted when all the predicted blocks carry textual content.

$$\text{csim}_{\text{merge}}\big(g_i,\ (p_j, \ldots, p_{j+k-1})\big) = \text{sim}_L\Big(\text{text}(g_i),\ \text{concat}\big(\text{text}(p_j), \ldots, \text{text}(p_{j+k-1})\big)\Big),$$

Alignment Procedure. Let $G = (g_1, \ldots, g_n)$ and $P = (p_1, \ldots, p_m)$ be the flattened ground-truth and predicted sequences. The algorithm maintains two pointers i and j, a bounded lookahead window of size $L = 4$, and two pending sets: $\mathcal{G}_{\text{pend}}$ for unmatched ground-truth blocks and $\mathcal{P}_{\text{pend}}$ for unmatched predicted blocks. At each step it selects the best of the following actions:

1. **Direct match.** If $\text{csim}(g_i, p_j) \geq \tau_{\text{align}}$, align (g_i, p_j) and advance both pointers. A type mismatch $\text{type}(g_i) \neq \text{type}(p_j)$ is recorded but does not prevent alignment.

2. **Merge match.** For each $k = 2, \ldots, L$, if $\text{csim}_{\text{merge}}(g_i, (p_j, \ldots, p_{j+k-1})) \geq \tau_{\text{align}}$, align $(g_i, (p_j, \ldots, p_{j+k-1}))$ and advance i by 1 and j by k. The alignment records the merged group as a single match with g_i.

3. **Lookahead in prediction.** Search positions $p_{j+1}, \ldots, p_{j+L}$ (and their merge groups) for the best content match with g_i. If a match is found at offset δ, the intervening predicted blocks $p_j, \ldots, p_{j+\delta-1}$ are placed in $\mathcal{P}_{\text{pend}}$. The matched pair is recorded, and the pointers advance past the match.

4. **Lookahead in ground truth.** Symmetrically, search positions $g_{i+1}, \ldots, g_{i+L}$ for the best content match with p_j. If a match is found at offset δ, the intervening ground truth blocks $g_i, \ldots, g_{i+\delta-1}$ are placed in $\mathcal{G}_{\text{pend}}$.

5. **Skip.** If no match is found within the lookahead window, place g_i in $\mathcal{G}_{\text{pend}}$ and advance i.

When multiple candidates exceed τ_{align}, the evaluator selects the one with the smallest total offset, breaking ties by highest content similarity. Blocks remaining in $\mathcal{G}_{\text{pend}}$ are declared missing, and those remaining in $\mathcal{P}_{\text{pend}}$ are declared extra and following considered for the next alignment step. In the default configuration, $\tau_{\text{align}} = 0.5$ and $L = 4$.

The procedure produces an alignment $\mathcal{A} = \big((u_1, v_1), (u_2, v_2), \ldots, (u_K, v_K)\big)$, where each pair may be $(i, \{j, \ldots, j+k-1\})$ for a merge match, (i, j) for a one-to-one match (possibly reordered), $(i, \varnothing)$ for a missing ground-truth block, or $(\varnothing, j)$ for an extra predicted block.

3.3.1 Text Block Scoring

For aligned pairs that belong to the textual types {`title`,`subtitle`,`narrative`}, the evaluator compares their textual content through normalization followed by normalized Levenshtein similarity.

Text Normalization

Before comparison, both texts are normalized to remove superficial differences caused by PDF extraction, OCR artifacts, or encoding inconsistencies. The procedure removes invisible formatting

characters, applies Unicode normalization, repairs line-break hyphenation —e.g. "envi-\nronment" becomes "environment"—, standardizes quotes and dashes (replaced with their ASCII equivalents), normalizes spacing, and collapses repeated whitespace. This ensures that similarity focuses on substantive extraction errors rather than formatting noise. This normalization ensures that the similarity computation focuses on substantive textual differences rather than formatting artifacts.

Levenshtein Similarity

Given two normalized strings $\text{text}(g_i)$ and $\text{text}(p_j)$, let $d_L(\text{text}(g_i), \text{text}(p_j))$ denote their Levenshtein distance [11], namely the minimum number of single-character insertions, deletions, and substitutions required to transform one string into the other. This normalization makes the score comparable across text blocks of different lengths.

The normalized similarity is defined as:

$$\text{sim}_L(\text{text}(g_i), \text{text}(p_j)) = 1 - \frac{d_L(\text{text}(g_i), \text{text}(p_j))}{\max(1, |\text{text}(g_i)|, |\text{text}(p_j)|)}.$$

The textual score is defined directly as: $\text{TextScore}(\text{text}(g_i), \text{text}(p_j)) = \text{sim}_L(\text{text}(g_i), \text{text}(p_j))$. No threshold-based or sigmoid transformation is applied. After normalization, the residual mismatch reflects genuine extraction errors, so the raw similarity remains directly interpretable and consistent with the weighted-ratio formulation used for tables.

Layout Verification for Text Blocks

In addition to textual content, each block contains a `width` attribute that takes one of two values: `full` for blocks that occupy the entire width of the page, and `half` for blocks that occupy approximately half the page. Blocks from composite structures inherit their width during flattening: both the narrative and the table produced by flattening a `text_table_sidebyside` block are marked as `half`. Standalone blocks (`title`, `subtitle`, standalone `narrative`) are implicitly `full`. The layout match for an aligned textual block pair (g_i, p_j) is defined as a binary indicator:

$$s_{ij}^{\text{layout}} = \mathbb{1}\big[\text{width}(g_i) = \text{width}(p_j)\big].$$

Type Verification for Text Blocks

Although the alignment algorithm matches blocks by content similarity regardless of their declared type (Section 3.3), the correct classification of a block as `title`, `subtitle`, or `narrative` carries important semantic information for downstream ESG reporting workflows. A document whose main heading is misclassified as a narrative paragraph, or whose section titles are confused with subtitles, may produce a structurally misleading output even if the textual content is perfectly extracted. The evaluator therefore verifies the block type of each aligned textual pair through a binary indicator:

$$s_{ij}^{\text{type}} = \mathbb{1}\big[\text{type}(g_i) = \text{type}(p_j)\big].$$

This check captures whether the system has correctly recognized the functional role of the block within the document hierarchy.

Combined Text Block Score

Both the layout attribute and the type classification are treated as *metadata* of the text block, while the textual similarity represents its *content*. Using the same content-to-metadata ratio $\lambda = w_C/w_M$ already defined at the cell level (Section 3.3.2), the final score for an aligned textual block pair (g_i, p_j) becomes:

$$\text{TextScore}(g_i, p_j) = \frac{\lambda\ \text{sim}_L(a, b) + s_{ij}^{\text{layout}} + s_{ij}^{\text{type}}}{\lambda + 2},$$

where each metadata component contributes with unit weight and the content component is scaled by λ. With $\lambda = 3$, the text similarity accounts for $3/5 = 60\%$ of the block score, the layout check for $1/5 = 20\%$, and the type check for $1/5 = 20\%$. This formulation introduces no additional free parameter: it reuses the same λ that governs the content-versus-metadata balance at the cell level, ensuring a consistent treatment of metadata across all granularity levels of the metric.

3.3.2 Criticality-Weighted Table Score

For `table` blocks, the evaluator adopts a hierarchical comparison procedure inspired by TEDS [30], but extended to support the richer cell schema used here and to decompose evaluation into *recall* and *precision* not only for cells, but also for columns, headers, and footnotes.

Throughout the comparison, the evaluator maintains four accumulators: (E_R, W_R) are the earned and possible recall contributions and (E_P, W_P) the corresponding precision contributions. Recall is grounded in the ground truth, whereas precision is grounded in the prediction; while missing elements reduce recall and extra elements reduce precision.

Column Comparison

Let $C^G = (c_1^G, \ldots, c_{n_c}^G)$ and $C^P = (c_1^P, \ldots, c_{m_c}^P)$ denote the ordered column descriptors of the ground truth and predicted table. Columns are compared positionally: the k-th ground truth column is paired with the k-th predicted column, up to the length of the shorter sequence. Column descriptors are criticality-aware, so each ground truth column c_k^G carries a criticality level, mapped to a weight $\omega_k^{\text{col}} = \omega\big(\text{crit}(c_k^G)\big)$. This allows the evaluator to penalize more strongly the omission or corruption of semantically important columns, such as those identifying reporting years, KPI families, or regulatory categories.

For the aligned column positions $k = 1, \ldots, \min(n_c, m_c)$, define

$$s_k^{\text{col}} = \text{sim}_L(c_k^G, c_k^P)$$

with $k \in \{1, \ldots, n_c\}$ for recall $s_k^{\text{col},R}$ and $k \in \{1, \ldots, m_c\}$ for precision $s_k^{\text{col},P}$.

For precision, extra predicted columns have no ground-truth criticality annotation, therefore they receive the same default structural hallucination penalty, namely $\omega(\texttt{high})$.

In accumulator form:

$$E_R^{\text{col}} = \sum_{k=1}^{n_c} \omega_k^{\text{col}}\, s_k^{\text{col},R}, \qquad W_R^{\text{col}} = \sum_{k=1}^{n_c} \omega_k^{\text{col}}, \qquad E_P^{\text{col}} = \sum_{k=1}^{m_c} \bar{\omega}_k^{\text{col}}\, s_k^{\text{col},P}, \qquad W_P^{\text{col}} = \sum_{k=1}^{m_c} \bar{\omega}_k^{\text{col}}.$$

Then:

$$R_{\text{col}} = \frac{E_R^{\text{col}}}{W_R^{\text{col}}} \qquad P_{\text{col}} = \frac{E_P^{\text{col}}}{W_P^{\text{col}}}.$$

This asymmetric and criticality-aware formulation ensures that missing important columns penalize recall proportionally to their semantic importance, while extra predicted columns penalize precision through a fixed structural hallucination weight.

Row Header Comparison

Multi-level headers are first flattened into a single ordered sequence of header cells. Let $H^G = (h_1^G, \ldots, h_{n_h}^G)$ and $H^P = (h_1^P, \ldots, h_{m_h}^P)$ denote the resulting ground truth and predicted header cell sequences.

Header cells are also criticality-aware in the updated schema. Each ground truth header cell h_k^G carries a weight $\omega_k^{\text{hdr}} = \omega\big(\text{crit}(h_k^G)\big)$. Each header cell carries a textual label, a `colspan`, and a `rowspan`. For an aligned pair at position k, the score combines all three through a weighted composite:

$$s_k^{\text{hdr}} = \frac{\lambda\ \text{sim}_L\big(\text{label}(h_k^G),\ \text{label}(h_k^P)\big) + \mathbb{1}\big[\text{cspan}(h_k^G) = \text{cspan}(h_k^P)\big] + \mathbb{1}\big[\text{rspan}(h_k^G) = \text{rspan}(h_k^P)\big]}{\lambda + 2}.$$

The label component receives the largest share because it carries the most informational content, while the structural span attributes receive smaller but non-negligible shares, since an incorrect colspan or rowspan can misalign the entire header hierarchy. All three sub-scores are in $[0, 1]$, so $s_k^{\text{hdr}} \in [0, 1]$.

For the aligned header indices $k = 1, \ldots, \min(n_h, m_h)$, define s_k^{hdr}. This quantity is then used in the recall view $s_k^{\text{hdr},R}$ for $k \in \{1, \ldots, n_h\}$ and in the precision view $s_k^{\text{hdr},P}$ for $k \in \{1, \ldots, m_h\}$.

As for columns, extra predicted header cells receive a default structural hallucination weight.

Similar to the comparison between columns, the E^{hdr} and W^{hdr} values are calculated in an accumulator for relative precision and recall.

This makes missing important header cells penalize recall more than secondary ones, while extra predicted headers penalize precision through the same structural hallucination mechanism used elsewhere.

Footnote Comparison

Let F^G and F^P denote the sets of footnote keys in the ground truth and prediction, respectively, and let w_{fn} be the weight associated with each footnote. For each shared footnote key $k \in \text{keys}(F^G) \cap \text{keys}(F^P)$, define $\tilde{s}_k^{\text{fn}} = \text{sim}_L(F^G[k], F^P[k])$.

This score is then used in the recall view $s_k^{\text{fn},R}$ for $k \in \text{keys}(F^G)$ and in the precision view $s_k^{\text{fn},P}$ for $k \in \text{keys}(F^P)$.

In accumulator form:

$$E_R^{\text{fn}} = \sum_{k \in \text{keys}(F^G)} s_k^{\text{fn},R}, \qquad W_R^{\text{fn}} = |\,\text{keys}(F^G)|\,, \qquad E_P^{\text{fn}} = \sum_{k \in \text{keys}(F^P)} s_k^{\text{fn},P}, \qquad W_P^{\text{fn}} = |\,\text{keys}(F^P)|\,.$$

This asymmetric formulation is coherent with the intended interpretation of recall and precision applied uniformly across all structural components: omitted footnotes reduce recall, while hallucinated footnotes reduce precision.

Row Alignment

It follows the procedure described for block-level alignment, because even here the order of the rows is expected to be largely preserved, but without support for merging, since row splitting is not a typical error mode within tables.

Let $R^G = (r_1^G, \ldots, r_u^G), R^P = (r_1^P, \ldots, r_v^P)$ be the row sequences of the ground truth and predicted table. Each row is first converted into a lightweight textual signature as concatenation of the normalized textual values of the row cells.

The similarity between two row signatures is computed with `SequenceMatcher`, which returns a ratio in $[0, 1]$ based on the total extent of matched subsequences. A row pair is considered a plausible match only if: $\sigma_{\text{row}}(r_i^G, r_j^P) > \tau_{\text{align}}$

Cell Comparison

Once rows have been aligned, cells within each aligned row pair are matched positionally. Let $\mathcal{F}$ denote the set of fields defined by the cell schema, each field $f \in \mathcal{F}$ has an associated weight w_f, and each comparison produces a score $s_f \in [0, 1]$. The field-level comparison rules are:

- **Simple fields** {`type`, `valueType`, `note`, `colspan`, `rowspan`}: compared by Normalized Exact Match, $s_f = \mathbb{1}[\text{str}(g.f) = \text{str}(p.f)]$.
- **Value raw**: if the value is numeric, the `raw` field is compared by Normalized Exact Match; if it is textual, the comparison uses sim_L.
- **Unit and format**: compared by normalized Exact Match if present on at least one side.

- **Graphical indicator value**: when `valueType` is `indicator`the `label` carries the main semantic meaning and is treated as content (e.g., `growth`, `check`, `profit`)); `category` and `symbol` are treated as semantic metadata, compared by Normalized Exact Match.
- **Formula**: if at least one cell contains a formula, the comparison requires a preliminary *cell ID resolution* step. This is described in the dedicated paragraph below.

Formula Comparison with Cell ID Resolution. Formula strings encode computational relationships among cells and have the general form

$$=\texttt{\{id\_1\}}\, op_1\, \texttt{\{id\_2\}}\, op_2 \cdots$$

where each $\texttt{id}_k$ is a cell identifier and each op_k an arithmetic operator. Because the ground truth and prediction assign different internal identifiers, direct string comparison is not meaningful. In addition, formulas that differ only by harmless reordering of commutative operands should not be penalized. Formula evaluation therefore performs cell-ID resolution followed by AST-based canonical normalization, but the final score remains an exact operator-aware token comparison.

Cell ID Mapping. The evaluator constructs a *cell ID mapping* μ from predicted identifiers to ground truth identifiers, derived from the row alignment (Section 3.3.2) and the positional cell matching already performed during the cell comparison phase. Specifically, for each aligned row pair (r_i^G, r_j^P), cells are matched by their column position: the k-th cell in the predicted row is paired with the k-th cell in the ground truth row. If a predicted cell c^P with identifier $\texttt{id}^P$ is positionally matched to a ground truth cell c^G with identifier $\texttt{id}^G$, then $\mu(\texttt{id}^P) = \texttt{id}^G$. Cells in unmatched rows or in positions beyond the length of the shorter row have no mapping. This step makes formula comparison meaningful across the two JSON representations.

AST-Based Canonical Normalization. After identifier resolution, each formula φ is parsed into an abstract syntax tree (AST), denoted by $A(\varphi)$. In this representation, leaf nodes correspond to cell references, while internal nodes correspond to arithmetic operators. For the predicted formula φ^P, the mapping μ is applied to the reference leaves so that matched predicted identifiers are rewritten in the ground-truth namespace. The role of the AST is limited to normalization.

This normalization is applied exclusively to the commutative operators in the set $C = \{+, *\}$. By contrast, non-commutative operators preserve their original order.

Canonical Token-Level Exact Match Score. Once the two formulas have been expressed in canonical form, the metric compares them exactly as token sequences. Let $(\widetilde{t}_1^G, \ldots, \widetilde{t}_n^G)$ be the canonical sequence of the ground-truth formula and let $(\widetilde{t}_1^P, \ldots, \widetilde{t}_m^P)$ be the canonical sequence of the predicted formula after identifier resolution. For formulas with $\max(n, m) > 0$, the score assigned to the `formula` field is

$$s_{\texttt{formula}} = \frac{\sum_{k=1}^{\min(n,m)} \mathbb{1}\left[\widetilde{t}_k^G = \widetilde{t}_k^P\right]}{\max(n,\, m)}.$$

The denominator $\max(n, m)$ ensures that both missing and superfluous tokens reduce the score. A formula receives credit only when the resolved operands and operators appear in the correct canonical positions. This preserves the rigour of exact matching while reducing sensitivity to irrelevant syntactic variation.

Two-Group Field Weighting. The evaluator partitions the field set into two groups:

- **Content fields** $\mathcal{F}_C = \{\texttt{raw}, \texttt{formula}, \texttt{label}\}$: carry the primary informational payload of the cell (the extracted value and its computational derivation).
- **Metadata fields** $\mathcal{F}_M = \{\texttt{type}, \texttt{valueType}, \texttt{unit}, \texttt{format}, \texttt{note}, \texttt{colspan}, \texttt{rowspan}, \texttt{category}, \texttt{symbol}\}$: describe structural and formatting properties.

The resulting $s_{\texttt{ind}}$ directly substitutes ρ_{cell} for indicator-typed cells, so that the same λ parameter governs the content-metadata balance for all cell types without introducing additional parameters. Set $\mathcal{F}^*$ the subset of fields that are present in at least one of the two cells, the cell ratio becomes:

$$\rho_{\text{cell}} = \frac{\sum_{f \in \mathcal{F}^* \cap \mathcal{F}_C} \lambda\, s_f \;+\; \sum_{f \in \mathcal{F}^* \cap \mathcal{F}_M} s_f}{\lambda\, |\mathcal{F}^* \cap \mathcal{F}_C| \;+\; |\mathcal{F}^* \cap \mathcal{F}_M|},$$

Criticality-Weighted Contributions of Cells, Columns, and Headers

Each ground truth data cell, column descriptor, and row-header cell carries a criticality annotation drawn from the ordered set {`low`, `medium`, `high`, `critical`}. The evaluator maps these levels to

numerical weights through a linear scale controlled by a single parameter $\kappa \geq 1$, which defines the ratio between the highest and lowest weights:

$$\omega(\texttt{low}) = 1, \qquad \omega(\texttt{medium}) = 1 + \tfrac{\kappa-1}{3}, \qquad \omega(\texttt{high}) = 1 + \tfrac{2(\kappa-1)}{3}, \qquad \omega(\texttt{critical}) = \kappa.$$

In the default configuration, $\kappa = 5$, yielding weights of approximately 1.0, 2.3, 3.7, and 5.0. Increasing κ makes the metric more sensitive to errors on high-risk fields, while $\kappa = 1$ recovers uniform weighting. For matched body-cell pairs, let $\rho_{\text{cell}}(c)$ be the field-level score and let ω_c be the criticality weight inherited from the ground truth cell. The contribution of that cell is:

$$e_c = \rho_{\text{cell}}(c)\, \omega_c,$$

Criticality is used not only to weight body-cell content errors, but also to modulate the contribution of structural labels that determine the semantic interpretation of the table.

Extra Predicted Cells. If a predicted row is unmatched, or if an aligned predicted row contains more cells than its ground truth counterpart, each additional predicted cell contributes to the precision denominator without earned score. Since hallucinated cells have no ground truth counterpart, they carry no criticality annotation, and the penalty for an extra predicted cell is: $w_{\text{extra},c} = \omega(\texttt{high})$. This design ensures that the extra-cell penalty is expressed on the same scale as the criticality weights used for matched and missing cells. In particular, the omission of a critical ground truth cell adds $\omega(\texttt{critical}) = \kappa$ to the recall denominator

Table-Level Precision, Recall, and F1

After structural comparison, row alignment, and cell comparison, the evaluator aggregates all contributions into table-level recall and precision. As with textual blocks (Section 3.3.1), each table block also carries a `width` attribute (`full` or `half`), and the layout match is verified as an additional structural field of the table itself. The layout score for an aligned table block pair (g_i, p_j) is:

$$s_{ij}^{\text{layout}} = \mathbb{1}\big[\text{width}(g_i) = \text{width}(p_j)\big],$$

contributing to both recall and precision accumulators with the same unit structural weight used for columns, headers, and footnotes ($w_{\text{lay}} = 0.1$). This introduces no additional free parameter: layout is simply another structural element on the same scale as the existing ones.

Let:

$$E_R^{\text{struct}} = E_R^{\text{col}} + E_R^{\text{hdr}} + E_R^{\text{fn}} + w_{\text{lay}}\, s_{ij}^{\text{layout}}, \qquad W_R^{\text{struct}} = W_R^{\text{col}} + W_R^{\text{hdr}} + W_R^{\text{fn}} + w_{\text{lay}},$$

$$E_P^{\text{struct}} = E_P^{\text{col}} + E_P^{\text{hdr}} + E_P^{\text{fn}} + w_{\text{lay}}\, s_{ij}^{\text{layout}}, \qquad W_P^{\text{struct}} = W_P^{\text{col}} + W_P^{\text{hdr}} + W_P^{\text{fn}} + w_{\text{lay}}.$$

Let C_{match} denote the set of matched cell pairs, C_{miss} the set of ground truth cells with no matched prediction, and C_{extra} the set of extra predicted cells.

Then:

$$R_{\text{table}} = \frac{E_R^{\text{struct}} + \sum_{c \in C_{\text{match}}} e_c}{W_R^{\text{struct}} + \sum_{c \in C_{\text{match}}} w_c + \sum_{c \in C_{\text{miss}}} w_c} \qquad P_{\text{table}} = \frac{E_P^{\text{struct}} + \sum_{c \in C_{\text{match}}} e_c}{W_P^{\text{struct}} + \sum_{c \in C_{\text{match}}} w_c + \sum_{c \in C_{\text{extra}}} w_{\text{extra},c}}.$$

The table-level F1 is the harmonic mean of precision and recall:

$$F_{1,\text{table}} = \frac{2\, P_{\text{table}}\, R_{\text{table}}}{P_{\text{table}} + R_{\text{table}}}, \qquad \text{for } P_{\text{table}} + R_{\text{table}} > 0.$$

When both precision and recall are zero, the table-level F1 is taken by convention to be its minimum value.

3.3.3 Document-Level Score Aggregation

Document-level evaluation combines aligned text and table block contributions into two category scores — a text score and a table score — which are then merged into an overall document score through the weights w_{text} and w_{table}. In the default configuration, textual blocks are assigned weight $w_{\text{text}} = 1.0$, whereas table blocks are assigned weight $w_{\text{table}} = 3.0$.

Extra Predicted Blocks. At the document level, a predicted block with no aligned ground truth counterpart contributes to the denominator of the corresponding document-level score without earned score. For textual blocks, the penalty is w_{text}; for table blocks, the extra table's full precision accumulator $W_P^{(t)}$ is added to the document-level precision denominator (Section 3.3.3).

Text Score

Text blocks are evaluated through a single similarity-based score, without an explicit precision–recall decomposition. This choice is motivated by the fact that the normalized text similarity already provides a direct scalar measure of the quality of textual recovery.

Each aligned pair of textual blocks contributes according to its text similarity, weighted by the importance assigned to textual content. Missing ground truth text blocks contribute zero, thereby penalizing omissions. Unmatched predicted text blocks are treated as spurious content and are penalized by contributing only to the denominator through the block-level penalty factor.

The final document-level text score is therefore defined as

$$S_{\text{text}} = \frac{\sum_{(g_i,p_j)\in\mathcal{A}_{\text{text}}} w_{\text{text}}\ \text{TextScore}(g_i, p_j)}{\sum_{(g_i,p_j)\in\mathcal{A}_{\text{text}}} w_{\text{text}}\ +\ \sum_{g_i\in\mathcal{M}_{\text{text}}} w_{\text{text}}\ +\ \sum_{p_j\in\mathcal{X}_{\text{text}}} w_{\text{text}}}.$$

where $\mathcal{A}_{\text{text}}$ denotes the set of aligned textual block pairs, $\mathcal{M}_{\text{text}}$ the set of missing ground truth textual blocks, and $\mathcal{X}_{\text{text}}$ the set of unmatched predicted textual blocks.

Table Score: Micro-Averaged Precision, Recall, and F1

Table blocks, unlike text blocks, are evaluated through a full precision-recall decomposition, as defined in Section 3.3.2. This is motivated by the richer structure of tables and by the greater severity of hallucinated tabular content in ESG reporting. While each individual table already produces its own precision, recall, and F1 (Section 3.3.2), these per-table scores must be aggregated into a single document-level table score. Rather than averaging the per-table F1 values —which would treat a small 3-row table and a large 50-row table as equally important— the evaluator adopts a *micro-averaging* strategy: the raw earned and possible accumulators from every table are pooled before computing a single document-level precision and recall. This ensures that larger tables and tables containing more critical cell, naturally contribute proportionally more to the document-level result.

Accumulator Pooling. Let $\mathcal{T}_{\text{align}}$ denote the set of aligned table block pairs, $\mathcal{T}_{\text{miss}}$ the set of ground truth table blocks with no corresponding prediction, and $\mathcal{T}_{\text{extra}}$ the set of predicted table blocks with no corresponding ground truth block. For each aligned table pair $t \in \mathcal{T}_{\text{align}}$, let $(E_R^{(t)}, W_R^{(t)})$ and $(E_P^{(t)}, W_P^{(t)})$ be the recall and precision accumulators produced by the table-level evaluation. Recall that these accumulators already include the structural components (columns, headers, footnotes), with criticality-aware weighting for columns and header cells, together with the criticality-weighted body-cell contributions.

The document-level recall accumulators are obtained by pooling over all table blocks:

$$E_R^{\text{doc}} = \sum_{t \in \mathcal{T}_{\text{align}}} E_R^{(t)}, \qquad W_R^{\text{doc}} = \sum_{t \in \mathcal{T}_{\text{align}}} W_R^{(t)} + \sum_{t \in \mathcal{T}_{\text{miss}}} W_R^{(t)}.$$

Missing ground truth tables contribute only to the recall denominator: each missing table t adds $W_R^{(t)}$, computed as the total possible recall weight of that table, with zero earned score, thereby penalizing the omission in proportion to the table's size and criticality.

Similarly, the document-level precision accumulators are:

$$E_P^{\text{doc}} = \sum_{t \in \mathcal{T}_{\text{align}}} E_P^{(t)}, \qquad W_P^{\text{doc}} = \sum_{t \in \mathcal{T}_{\text{align}}} W_P^{(t)} + \sum_{t \in \mathcal{T}_{\text{extra}}} W_P^{(t)}.$$

Extra predicted tables contribute only to the precision denominator: each spurious table t adds $W_P^{(t)}$, computed as the total possible precision weight of that table's predicted content, with zero earned score, thereby penalizing hallucinated tabular content in proportion to its extent.

Document-Level Table Precision, Recall, and F1. The document-level table recall, precision and F1 as the harmonic mean are:

$$R_{\text{doc}} = \frac{E_R^{\text{doc}}}{W_R^{\text{doc}}}, \qquad P_{\text{doc}} = \frac{E_P^{\text{doc}}}{W_P^{\text{doc}}},$$

$$F_1^{\text{doc}} = \frac{2\, P_{\text{doc}}\, R_{\text{doc}}}{P_{\text{doc}} + R_{\text{doc}}}, \qquad \text{for } P_{\text{doc}} + R_{\text{doc}} > 0.$$

When $P_{\text{doc}} = R_{\text{doc}} = 0$, the score is taken by convention to be its minimum value. The document-level table score used in the overall aggregation is defined as: $S_{\text{table}} = F_1^{\text{doc}}$.

This micro-averaging approach has the advantage to naturally weights each table by its size and criticality content. It, also, preserves the full precision–recall decomposition at the document level: R_{doc} directly quantifies how much of the ground truth tabular content was recovered, while P_{doc} quantifies how much of the predicted tabular content is grounded in the ground truth. All three quantities R_{doc}, P_{doc}, and F_1^{doc} are reported alongside the per-table scores to support diagnostic analysis at multiple levels of granularity.

Overall Document Score

The overall document score combines S_{text} and S_{table} as a weighted average:

$$S = \frac{w_{\text{text}}\, S_{\text{text}} + w_{\text{table}}\, S_{\text{table}}}{w_{\text{text}} + w_{\text{table}}}.$$

With the default weights ($w_{\text{text}} = 1$, $w_{\text{table}} = 3$), the table score contributes three-quarters of the overall result, reflecting the greater importance of structured content extraction in ESG document understanding.

Micro-Averaged Document Evaluation. To obtain a unified score over heterogeneous document elements, the framework adopts micro-averaging over the full set of evaluation units U, including textual blocks, table cells, formula cells, and semantic indicators such as graphical-indicator sub-fields or boolean values. Each unit $u \in U$ receives a similarity score $s(u) \in [0, 1]$ according to its value type. The overall document score is defined as:

$$S_{\text{doc}} = \frac{\sum_{u \in U} w_u \cdot s(u)}{\sum_{u \in U} w_u}$$

where w_u is the criticality weight associated with unit u, derived from the severity annotations in the ground truth schema. This formulation assigns greater penalty to errors on business-critical elements than to errors on secondary descriptive content.

3.3.4 Diagnostic Score Decomposition

Although the overall score S and the text/table sub-scores summarize extraction quality, they do not show which aspect of document understanding is responsible for the errors. The framework therefore decomposes the score into three diagnostic sub-scores: *structural fidelity*, *content accuracy*, and *semantic completeness*.

Partitioning of Evaluation Units

Let U denote the complete set of evaluation units contributing to the document score. This set encompasses every scored element across all aligned blocks, including textual block comparisons, table structural components, and individual cell fields. We define a partition of U into three disjoint categories:

$$U = U_{\text{struct}} \cup U_{\text{content}} \cup U_{\text{sem}}.$$

Structural Units (U_{struct}). Structural units capture whether the organization of the document is reconstructed correctly. This includes columns, header cells, footnotes, layout-related properties of

text and table blocks, and cell-level structural metadata such as role and spanning behaviour.

Content Units (U_{content}). Content units measure whether the main informational payload is extracted correctly. They include *textual similarity* of aligned text blocks, *raw value* associated with each table cell, and *label* sub-field that carries the semantic meaning of the graphical indicator.

Semantic Units (U_{sem}). Semantic units capture higher-level interpretation. They include formulas expressing dependencies between cells, measurement units, formatting conventions, annotation markers (notes or footnote references), and, for indicator cells, the *category* and *symbol* sub-fields.

Diagnostic Score Computation

Each evaluation unit $u \in U$ is associated with a weight w_u and a similarity score $s(u) \in [0, 1]$, both inherited from the corresponding evaluation component described in the preceding sections. For body-cell fields, the weight incorporates the criticality factor as $w_u = w_f \cdot \omega_c$, where w_f is the field weight (λ for content fields, 1 for metadata fields) and ω_c is the criticality weight of the ground truth cell. For columns and header cells, the same principle is applied through the criticality-aware structural weights introduced in the corresponding subsection.

The diagnostic score for each category $C \in \{\text{struct}, \text{content}, \text{sem}\}$ is computed as a micro-averaged weighted ratio:

$$S_C = \frac{\sum_{u \in U_C} w_u \cdot s(u)}{\sum_{u \in U_C} w_u}. \tag{3.1}$$

Handling of Missing and Extra Elements

The diagnostic decomposition is aligned with the penalty logic of the main metric. Missing body cells distribute their penalty across structure, content, and semantics according to the composition of the missing fields. Missing columns and header cells are assigned fully to structure, with their criticality-aware weight. Extra predicted cells, columns, and headers are likewise treated as structural errors, since they reflect hallucinated table structure. Missing or extra text blocks are

split between content and structure, while missing or extra tables are distributed across categories according to their structural component and estimated cell composition.

Diagnostic Interpretation

The three sub-scores capture different aspects of system behaviour. High structure with low content indicates that layout is reconstructed correctly but values are wrong. Low structure with high content suggests that values are extracted correctly but placed in the wrong layout. Low semantics with otherwise accurate content shows that higher-level information, such as units, formulas, or formatting conventions, is not preserved. For indicator cells, the same decomposition helps distinguish failures in semantic interpretation from failures in glyph recognition. This distinction is especially relevant in ESG document understanding, where structural, textual, and semantic errors have different downstream effects.

Parameters

To reduce arbitrariness and improve reproducibility, most weights are fixed or derived deterministically from the criticality scale. The metric contains only a minimal number of free parameters.

Fixed Algorithmic Constants. Algorithmic values are determined by the expected characteristics of the input (mostly co-ordered sequences with local deviations) and do not require tuning across different ESG document types (see Table 3.1)

Parameter	Value	Rationale
L (lookahead)	4	Sufficient for typical block gaps
M (merge window)	3	Blocks rarely split into more than 3 fragments
τ_{align}	0.5	Stricter threshold, since rows usually share a common schema

Table 3.1: Fixed algorithmic constants used in the evaluation procedure.

Free Scoring Parameters After fixing the above, only three independent parameters are expressed in the Table 3.2

Parameter	Default	Meaning	Effect
$\lambda = w_C/w_M$	3	Content/metadata ratio	Higher λ penalizes value errors more
κ	5	Criticality spread	Higher κ amplifies critical-cell errors
$w_{\text{table}}/w_{\text{text}}$	3	Table/text ratio	Higher ratio favors table accuracy

Table 3.2: Default values of the free parameters.

Each parameter has a clear semantic interpretation: λ controls how much the metric cares about the actual extracted value relative to its metadata —both at the cell level (value and formula versus type, format, and spans) and at the text block level (textual similarity versus layout and type classification)— κ controls how much the metric amplifies errors on high-risk fields, and $w_{\text{table}}/w_{\text{text}}$ controls the relative importance of structured versus unstructured content in the overall document score. These three free parameters makes the framework practical to calibrate, easy to report, and reproducible across different evaluation settings.

Summary of Advantages over Existing Metrics

Section 2.4.4 highlighted several limitations of existing metrics for ESG document understanding, which the proposed framework addresses directly.

A first distinctive feature is *criticality weighting* (Section 3.3.2), through which errors on high-risk values and on the structural labels that contextualize them are penalized more heavily than errors on secondary items. This is particularly important in ESG reporting, where some fields have much greater downstream relevance than others. Existing metrics such as TEDS, GriTS, and cell-level Exact Match generally treat all elements uniformly.

A second feature is *formula awareness* (Section 3.3.2). Formula strings are treated as first-class cell fields, allowing the metric to distinguish systems that recover computational relationships from those that only reproduce final values. Formula comparison also accounts for the fact that ground-truth and predicted representations assign independent cell identifiers: predicted references are first translated into the ground-truth namespace through cell-ID resolution, after which token-level exact match evaluates operands and operators.

A third advantage is the *unified evaluation of structure and content* within a single precision–recall accumulation framework. Structural components, textual blocks, and cell fields are evaluated

coherently under the same weighted formulation, removing the need to combine separate structural metrics (such as TEDS or GriTS) with separate content metrics (such as exact match or cell-level F1).

In addition, the framework provides a *diagnostic score decomposition* (Section 3.3.4) into structural fidelity, content accuracy, and semantic completeness. This makes the evaluation more informative than a single aggregate score, because it helps identify whether performance degradation is mainly caused by layout reconstruction errors, incorrect extracted values, or loss of higher-level information such as units, formulas, formatting conventions, or graphical indicator semantics.

Finally, the framework offers a complete *precision–recall breakdown* from individual structural elements to the micro-averaged document-level scores P_{doc} and R_{doc}, making omissions and hallucinations explicitly observable at every level of granularity. To support error analysis further, it also produces a detailed document-level inconsistency report that records the specific mismatches contributing to the final score.

3.4 Document Parsing: Tools, Models, and Implementations

Since the quality of a retrieval-augmented generation system depends strongly on the upstream parsing stage, this section presents the tools and models evaluated in this work. All implementations receive a PDF document as input and produce a structured JSON output containing a sequence of content blocks. This canonical output format was designed to be directly compatible with the evaluation benchmark described in Section 3.2.

3.4.1 Rule-Based PDF Extractors

The first category of tested tools includes libraries that parse directly the internal PDF structure, relying on the native text layer and coordinate metadata, to make them efficient and lightweight for large-scale pipelines. However, their heuristic nature makes them less robust on complex or highly variable layouts.

pdfplumber. pdfplumber is a pure-Python library for extracting text, metadata, and tables from PDF files by analyzing internal character and line objects. In this work, it was not evaluated in

depth because preliminary tests produced weak results. Although it can identify content blocks and table bounding boxes, it may miss tables defined mainly by whitespace or background fills rather than explicit lines. Moreover, it operates only on the native PDF text layer and does not perform OCR, which limits its applicability to scanned or image-based documents.

PyMuPDF4LLM. PyMuPDF4LLM is a higher-level interface built on PyMuPDF to generate LLM-friendly Markdown from PDF documents. Unlike the base PyMuPDF library, which returns raw text without semantic annotation, PyMuPDF4LLM preserves structural elements such as headings, paragraphs, and tables, making it more suitable for downstream chunking and retrieval. In this work, the extracted Markdown was converted into the canonical JSON format by mapping headings to titles or subtitles, paragraphs to narrative blocks, and Markdown tables to structured tables. A distinctive feature of this tool is its configurable table detection strategy and, in this work, the `lines` strategy was adopted as it provided the best trade-off, capturing lightly-shaded tables that the strict mode would miss while avoiding the over-segmentation sometimes produced by the text-based mode.

The pipeline first applies OCR through PyMuPDF's Tesseract integration when the PDF lacks a native text layer. It then extracts Markdown with `pymupdf4llm.to_markdown()`, using the `pymupdf.layout` extension to improve reading order and table recognition, providing AI-based page layout analysis. The resulting Markdown is finally converted into the canonical JSON format by mapping headings, tables, and paragraphs to their corresponding content blocks.

An important limitation is that Markdown tables are inherently flat, so merged cells, row or column spans, and multi-level headers are not fully preserved. As a result, hierarchical tables common in ESG reports may lose structural information during conversion.

3.4.2 Layout-Aware Document Preprocessing Platforms

The second category includes platforms that integrate OCR, layout analysis, and table recognition into a unified pipeline. Rather than extracting plain text only, these systems first identify document regions such as titles, paragraphs, and tables, and then process each region according to its type. This produces a richer representation for downstream RAG workflows.

Unstructured. Unstructured decomposes PDF files into semantically typed elements such as titles, text blocks, tables, headers, and footers, each annotated with metadata including the page number and element coordinates.

A central concept in Unstructured is *partitioning*, that is, the segmentation of raw document content into typed elements. For PDFs, the library supports different strategies with different trade-offs between speed and accuracy. In this work, the `hi_res` strategy was adopted, combining YOLOX for layout analysis, Tesseract for OCR, and a dedicated table structure recognition module that produces HTML representations of detected tables. Its main advantage is the explicit categorization of document elements, which helps separate meaningful content from structural noise. However, the output often requires post-processing to merge fragmented text and normalize table content.

Docling. Docling is an advanced PDF processing library and is particularly relevant here because its documentation explicitly mentions support for XBRL financial reports, which is closely related to the structured corporate reporting scenario.

For layout analysis, it employs RT-DETR [28], a real-time detection transformer that identifies different layout components on each page, including tables, text blocks, titles, pictures, captions, footnotes, and formulas. RT-DETR extends the original DETR architecture with an efficient hybrid encoder for multi-scale feature processing. For table structure recognition, it uses TableFormer [15], a vision-transformer model that identifies the logical structure of a table starting from its image. TableFormer employs a CNN backbone to encode the table image, followed by a Structure Decoder that generates the logical table structure as a sequence of HTML tokens, and a Cell Bounding Box Decoder that simultaneously predicts bounding boxes for each table cell. This architecture handles complex table characteristics such as partial or absent borderlines, empty cells, row and column spans, and inconsistent alignment. For OCR, in this work, Tesseract was configured. For layout analysis, the pipeline was configured to use the Egret Large layout model, which improved the accuracy of the reported data and reduced hallucination cases compared to earlier layout configurations tested during development.

A Granite VLM pipeline variant was also tested but produced less accurate outputs compared to the standard Docling pipeline with its specialised components, confirming that dedicated models for layout analysis and table structure recognition remain more effective than general-purpose

vision-language models for this task.

OpenDataLoader. OpenDataLoader PDF is a document conversion library that produces LLM-ready JSON while preserving structural elements such as headings, lists, tables, reading order, bounding boxes, and cell spans. In this work, it was used in hybrid mode (`hybrid="docling-fast"`), which applies page-level triage: simple pages are processed through the fast local rule-based pipeline, whereas complex pages, such as those containing tables or OCR-heavy content, are routed to a local Docling backend for layout detection and table structure recognition. The outputs are then merged into a unified hierarchical JSON representation, which was finally mapped into the canonical schema used in this thesis.

This configuration combines the efficiency of deterministic PDF parsing with the stronger structural reconstruction provided by Docling on visually complex pages.

3.4.3 Transformer-Based Document Models

The third category includes models that learn document structure directly from data. These approaches are more computationally expensive than rule-based extractors, but they generally provide stronger structural understanding and better robustness on visually complex documents.

Table Transformer (TaTR). In this work, TaTR was implemented as a two-stage table extraction pipeline using separate models for table detection and structure recognition. PDF pages were first rasterised at 300 DPI, then processed by the detection model to localise table regions. Each detected table was cropped and passed to the structure model, which predicted rows, columns, headers, and spanning cells.

A grid was then reconstructed by intersecting the predicted row and column regions, with an additional row-merging step to reduce spurious subdivisions caused by multi-line text. The text content of each cell was finally extracted with Tesseract OCR, and the resulting values were normalised into the canonical JSON format.

Since TaTR is limited to tables, narrative text outside detected table regions was extracted separately through full-page OCR and simple heuristic rules for titles, subtitles, and text blocks. The overall output is therefore a complete JSON document combining table reconstruction with

OCR-based extraction of the remaining page content.

LayoutLMv3. LayoutLMv3 [9] is a multimodal transformer that jointly encodes textual, spatial, and visual document features. In this work, a token-classification checkpoint was used to assign labels such as title, section header, text, or table to OCR tokens, which were then grouped into higher-level content blocks. The model uses ViT-style patch embeddings to unify text, layout, and image information within a single representation. However, table reconstruction still required additional grouping and heuristic post-processing. For this reason, it was not included in the final evaluation, as preliminary results were weak and the required post-processing was too laborious.

3.4.4 Vision-Language Models

The final category includes vision-language models, which process document pages as images and generate textual outputs from prompts. These models do not explicitly separate layout analysis, OCR, and table reconstruction, but attempt to infer structure in a single generative step.

GLM-OCR. GLM-OCR is a multimodal document-understanding model that generates structured outputs such as Markdown, JSON, and tables. In this work, each page was rendered as an image, processed by the model, and then converted from Markdown into the canonical JSON format.

The model supports tasks such as document parsing, text recognition, table extraction, and key information extraction, with particular strength on table understanding. Because of context window limits, multi-page documents were processed in chunks and later merged through overlap-aware deduplication. Although the model showed good flexibility, the outputs were not always stable. Markdown was more reliable than JSON or HTML, but some text blocks and tables were still missed, and cross-page tables were not correctly merged.

LLaVA. LLaVA [14] is a general-purpose vision-language model that was also tested for document parsing. In practice, however, it proved poorly suited to this task. The outputs were unstable, text extraction was inaccurate, and structured formats such as JSON, HTML, or Markdown were often incomplete or inconsistent. It should be noted that subsequent work in the literature has produced

specialised variants: LLaVA-Read for improved OCR capabilities, Table-LLaVA for enhanced table understanding, and FinTab-LLaVA specifically for financial table comprehension. However, they were not evaluated in this work during the experimental phase.

Common Post-Processing and Value Classification

All implementations shared a common post-processing layer for value classification. Cell contents were normalised into graphical indicators, boolean values, numerical values, or plain text, while footnote markers were separated when present. This common step ensured that score differences across tools reflected actual extraction and structural performance rather than inconsistencies in value interpretation.

4. Results

This chapter reports the empirical results obtained on the benchmark introduced in Chapter 3.2. The analysis proceeds from the global ranking to the text–table decomposition, the precision-recall trade-off, the diagnostic breakdown into structure, content, and semantic dimensions, and a family-level summary.

Since vision-language models receive document pages as rendered images and the benchmark includes multi-page reports that can exceed context-window limits, all multimodal LLMs were evaluated using a uniform chunking strategy: each document was split into chunks of five consecutive pages with a one-page overlap between adjacent chunks. Per-chunk outputs were then merged through overlap-aware deduplication to reconcile duplicate blocks from shared boundary pages. The overlap ensures that cross-page tables and narrative blocks are seen in full by at least one chunk, while the fixed chunk size standardises the effective input length across models with different context limits.

4.1 Overall Ranking and Distributional Behaviour

Table 4.1 reports the summary statistics of the overall score for each system. The evaluation covers thirteen systems spanning four methodological families: frontier online LLMs, document parsing and OCR pipelines, an open-weight LLM, and a table-specific baseline.

Sonnet 4.6 and Gemini 3 achieve virtually identical mean scores (0.833 and 0.831) with the lowest standard deviations in the benchmark (0.134 and 0.138), placing them as the most accurate and consistent systems. Within the frontier LLM group a clear gradient emerges, descending through Sonnet 4.5 (0.766), GPT-5.2 (0.729), Gemini 2.5 (0.687), and GPT-5.1 (0.659), with variability increasing progressively.

The parsing pipelines occupy the intermediate range: OpenDataLoader and Docling are closely matched at 0.636 and 0.631, followed by Unstructured (0.514), PyMuPDF4LLM (0.482), and GLM-OCR (0.462). These tools generally exhibit higher standard deviations than the frontier LLMs, with OpenDataLoader reaching 0.230, indicating stronger sensitivity to document-level variation. The

Table 4.1: Overall score summary statistics for all evaluated systems, grouped by system category.

Model	Mean	Median	Std. Dev.	IQR
Frontier online LLMs				
Sonnet 4.6	0.833	0.841	0.134	0.144
Gemini 3	0.831	0.852	0.138	0.174
Sonnet 4.5	0.766	0.790	0.184	0.166
GPT-5.2	0.729	0.760	0.179	0.202
Gemini 2.5	0.687	0.687	0.146	0.171
GPT-5.1	0.659	0.662	0.175	0.200
Document parsing and OCR pipelines				
OpenDataLoader (hi_res)	0.636	0.651	0.230	0.377
Docling	0.631	0.621	0.195	0.237
Unstructured	0.514	0.528	0.158	0.265
PyMuPDF4LLM	0.482	0.441	0.197	0.256
GLM-OCR	0.462	0.455	0.187	0.211
Open-weight LLM				
Gemma 3	0.459	0.448	0.135	0.156
Table-specific baseline				
TATR	0.340	0.274	0.178	0.198

open-weight Gemma 3 (0.459) and the table-specific TATR (0.340) close the ranking.

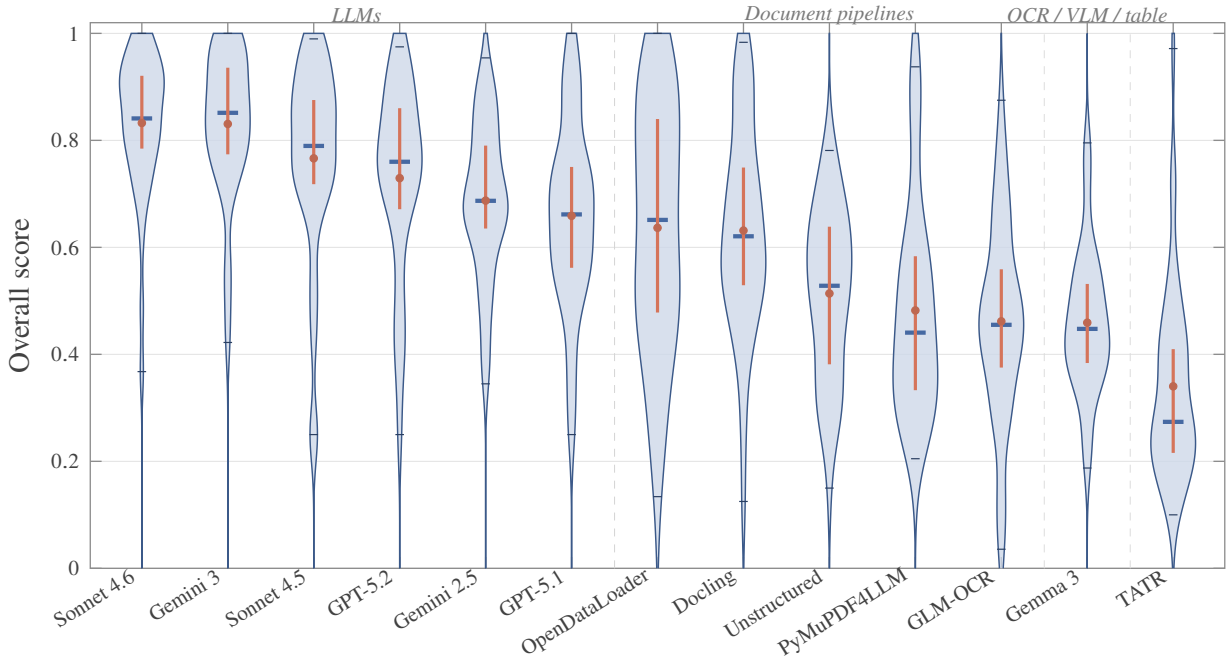

Figure 4.1: Distribution of per-document overall scores across the evaluated systems. The width of each violin reflects the kernel density of the score distribution; horizontal markers indicate the median and interquartile range.

The violin plots in Figure 4.1 confirm that the frontier LLMs show compact, top-heavy distributions with density concentrated above 0.75, whereas parsing pipelines display broader and more symmetric profiles. The mean standard-deviation scatter in Figure 4.2 further illustrates the accuracy–consistency trade-off. Sonnet 4.6 and Gemini 3 lie closest to the ideal top-left corner (high mean, low variability). Gemma 3 achieves the lowest standard deviation in the entire benchmark (0.135), but at a much lower mean, demonstrating that low variability alone does not indicate strong performance.

4.2 Modality-Specific Performance: Text and Table Extraction

A more detailed view of the benchmark is obtained by separating the overall score into text and table components. Table 4.2 reports the aggregated results, and Figure 4.3 presents the corresponding per-document distributions. The distinction is relevant because these two modalities pose different

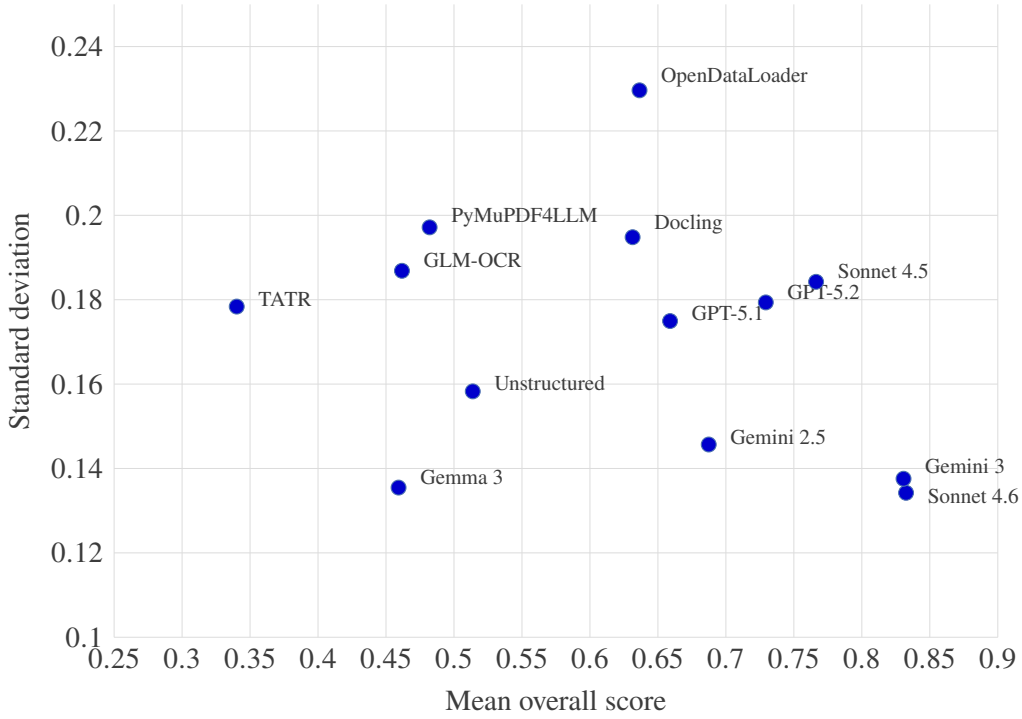

Figure 4.2: Trade-off between mean overall score and standard deviation across the evaluated systems. Proximity to the top-left corner indicates higher accuracy with lower variability.

levels of difficulty, with tables generally representing the harder extraction problem.

Text Extraction

Text extraction is largely solved for most systems. All frontier LLMs score above 0.89, with Sonnet 4.6 reaching 0.992 and an IQR of zero. Among the remaining systems, Gemma 3 achieves a text mean of 0.920, higher than several parsing pipelines, indicating that its weaknesses lie elsewhere. PyMuPDF4LLM also performs well (0.901), benefiting from direct access to the native PDF text layer. The only substantial exception is GLM-OCR (0.686), whose reliance on page-level image processing makes it more susceptible to layout ambiguity and rendering artefacts.

Table Extraction

Table extraction is the main discriminating factor in the benchmark. Gemini 3 achieves the highest mean table score (0.782), just outperforming Sonnet 4.6 (0.779), but reflects Sonnet 4.6's marginally better text score. The gap between frontier LLMs and parsing pipelines grows significantly on this

Table 4.2: Descriptive statistics for text and table scores across the evaluated systems, grouped by methodological category.

	Text score					Table score				
Model	Mean	Median	Q1	Q3	IQR	Mean	Median	Q1	Q3	IQR
Frontier online LLMs										
Sonnet 4.6	0.992	1.000	1.000	1.000	0.000	0.779	0.802	0.711	0.904	0.193
Gemini 3	0.978	1.000	0.983	1.000	0.017	0.782	0.803	0.699	0.925	0.225
Sonnet 4.5	0.965	1.000	0.967	1.000	0.033	0.700	0.719	0.630	0.840	0.210
GPT-5.2	0.942	1.000	0.917	1.000	0.083	0.658	0.680	0.545	0.815	0.270
Gemini 2.5	0.961	1.000	0.948	1.000	0.052	0.596	0.600	0.510	0.723	0.213
GPT-5.1	0.896	1.000	0.895	1.000	0.105	0.580	0.616	0.430	0.681	0.251
Document parsing and OCR pipelines										
OpenDataLoader (hi_res)	0.878	0.999	0.750	1.000	0.250	0.556	0.578	0.331	0.795	0.464
Docling	0.893	0.932	0.799	1.000	0.201	0.544	0.543	0.406	0.758	0.351
Unstructured	0.804	0.890	0.685	1.000	0.315	0.417	0.433	0.287	0.559	0.272
PyMuPDF4LLM	0.901	0.975	0.891	1.000	0.109	0.343	0.300	0.114	0.465	0.352
GLM-OCR	0.686	0.695	0.500	1.000	0.500	0.387	0.375	0.256	0.519	0.263
Open-weight LLM										
Gemma 3	0.920	1.000	0.917	1.000	0.083	0.306	0.286	0.189	0.427	0.238
Table-specific baseline										
TATR	0.483	0.444	0.306	0.590	0.284	0.293	0.241	0.141	0.402	0.261

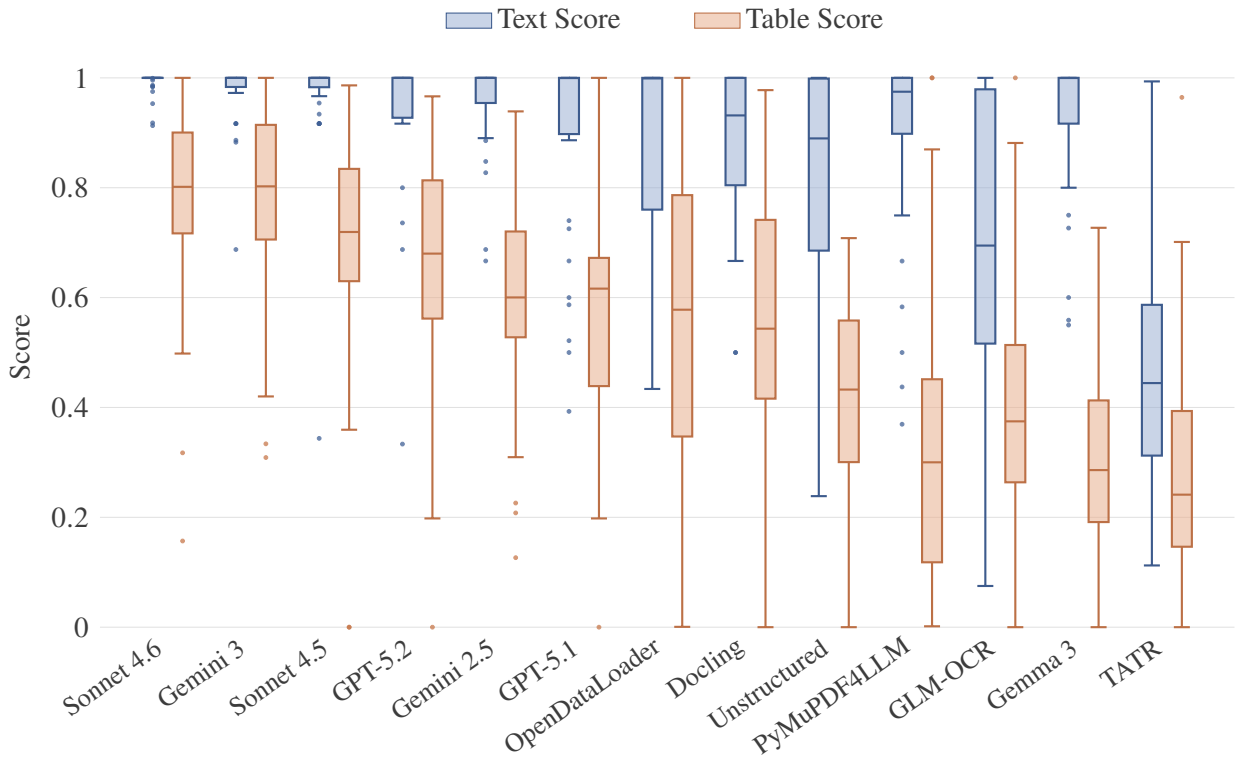

Figure 4.3: Distribution of per-document text score and table score across the evaluated systems. Boxes show the interquartile range, the central line marks the median, whiskers follow the 1.5 IQR rule, and dots indicate outliers.

dimension: OpenDataLoader and Docling reach 0.556 and 0.544, about 23 points below the leaders, while PyMuPDF4LLM drops to 0.343 and Gemma 3 to 0.306. TATR records the lowest table score at 0.293.

From Figure 4.3, the text–table asymmetry is pronounced across all families: text distributions are compressed near the top of the scale, whereas table distributions are wider and shifted downward. This confirms that table reconstruction, with its requirements for structural parsing, merged-cell handling, and value-type classification, remains the central challenge in ESG document understanding.

Table Precision and Recall

Table 4.3 and Figure 4.4 further characterize table extraction through precision and recall.

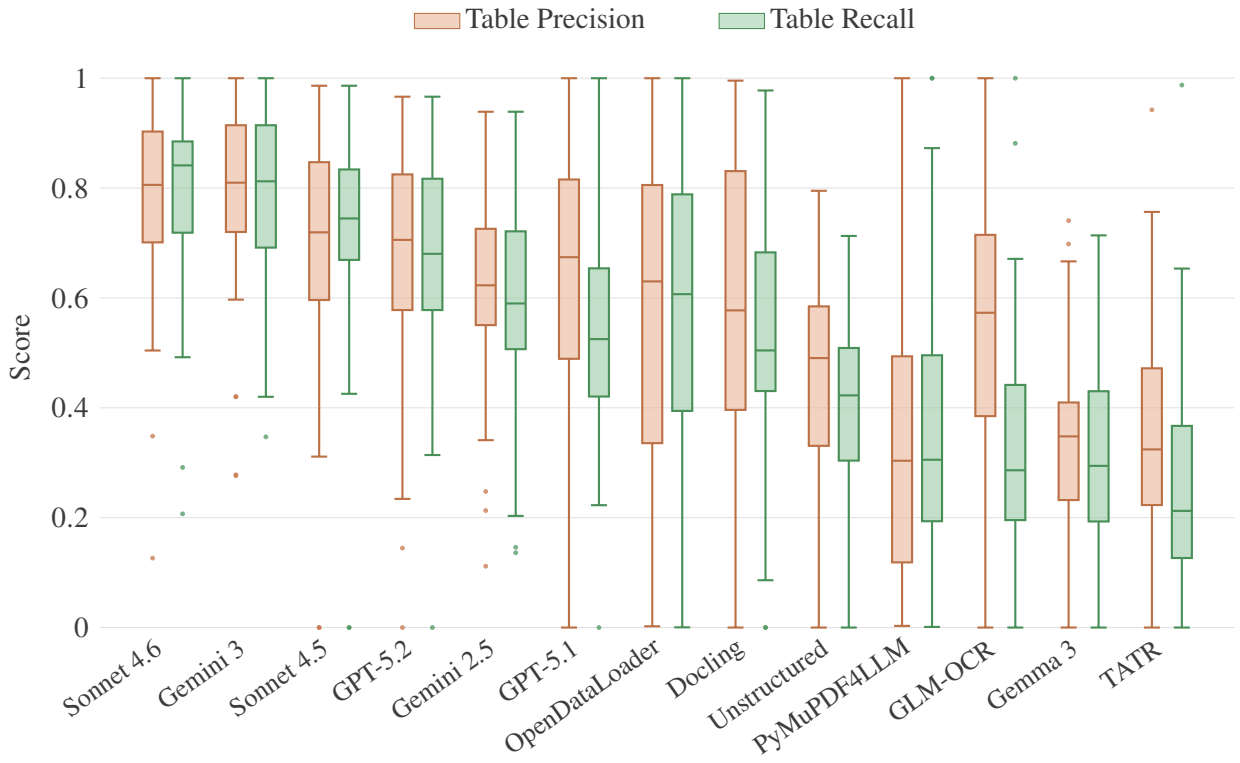

Figure 4.4: Distribution of per-document table precision and table recall across the evaluated systems. Boxes show the interquartile range, the central line marks the median, whiskers follow the 1.5 IQR rule, and dots indicate outliers.

The top frontier LLMs exhibit balanced profiles: Gemini 3 achieves 0.781 precision and 0.783

Table 4.3: Descriptive statistics for table precision and recall across the evaluated systems, grouped by methodological category.

	Table precision					Table recall				
Model	Mean	Median	Q1	Q3	IQR	Mean	Median	Q1	Q3	IQR
Frontier online LLMs										
Sonnet 4.6	0.775	0.806	0.695	0.904	0.209	0.786	0.841	0.714	0.890	0.176
Gemini 3	0.781	0.810	0.718	0.925	0.206	0.783	0.812	0.684	0.925	0.241
Sonnet 4.5	0.689	0.719	0.596	0.855	0.259	0.715	0.745	0.667	0.840	0.172
GPT-5.2	0.663	0.706	0.545	0.832	0.287	0.667	0.680	0.545	0.822	0.277
Gemini 2.5	0.611	0.623	0.547	0.729	0.182	0.599	0.590	0.505	0.724	0.219
GPT-5.1	0.639	0.674	0.476	0.823	0.346	0.547	0.525	0.403	0.660	0.257
Document parsing and OCR pipelines										
OpenDataLoader (hi_res)	0.576	0.630	0.326	0.815	0.489	0.571	0.607	0.377	0.804	0.426
Docling	0.572	0.577	0.391	0.841	0.450	0.535	0.504	0.423	0.691	0.269
Unstructured	0.440	0.491	0.318	0.588	0.270	0.408	0.423	0.301	0.516	0.216
PyMuPDF4LLM	0.352	0.304	0.115	0.513	0.399	0.353	0.305	0.187	0.499	0.312
GLM-OCR	0.556	0.573	0.381	0.730	0.349	0.334	0.286	0.187	0.454	0.268
Open-weight LLM										
Gemma 3	0.327	0.348	0.223	0.414	0.191	0.304	0.294	0.186	0.452	0.265
Table-specific baseline										
TATR	0.354	0.324	0.205	0.495	0.290	0.263	0.212	0.118	0.382	0.264

recall, and Sonnet 4.6 is similarly symmetric at 0.775 and 0.786. GPT-5.1 deviates from this pattern, with precision (0.639) exceeding recall (0.547) by nearly ten points, suggesting a more conservative extraction strategy that misses table content.

Among the pipelines, the most asymmetry appears for GLM-OCR: a precision of 0.556 paired with a recall of only 0.334, indicating that its extracted content is reasonably accurate but a large fraction of tables is missed entirely. This is consistent with a vision-language model that captures the most visually salient table regions. OpenDataLoader is relatively balanced (0.576 vs. 0.571) but with the largest IQRs in the benchmark (0.489 and 0.426), confirming highly document-dependent performance.

PyMuPDF4LLM, Gemma 3, and TATR all remain below 0.36 on both metrics, with TATR's recall of 0.263 suggesting that its detection module fails to localize a substantial fraction of the tables despite its specialized design.

Family-Level Comparison

Table 4.4 aggregates the results by system family. The frontier LLM family leads across all dimensions (overall 0.751, text 0.956, table 0.683) with the second-lowest average variability (0.159). The parsing pipelines follow at 0.566 overall, maintaining competitive text extraction (0.869) but falling behind on tables (0.465). The 18.5-point gap between these two families is predominantly driven by the 21-point difference on table extraction. The OCR/VLM (GLM-OCR, 0.462) and open LLM (Gemma 3, 0.459) categories are closely matched overall, though for different reasons: GLM-OCR is limited by weak text extraction, while Gemma 3 is limited by weak table handling despite strong text performance. TATR records the lowest family-level scores across all dimensions. These aggregated results confirm that the ability to handle tabular content is the primary differentiator among system families, since text extraction is consistently strong for any system with access to the native PDF text layer or high-quality OCR.

4.3 Diagnostic Decomposition

The evaluation metric provides three diagnostic dimensions: structure (row–column grid, headers, merged cells), content (cell value accuracy, weighted by criticality), and semantic (value types,

Table 4.4: Aggregate comparison by system family.

Family	N	Overall Mean	Text Mean	Table Mean	Average Std. Dev.
Frontier LLM	6	0.751	0.956	0.683	0.159
Parsing pipeline	4	0.566	0.869	0.465	0.195
OCR/VLM	1	0.462	0.686	0.387	0.187
Open LLM	1	0.459	0.920	0.306	0.135
Table specialist	1	0.340	0.483	0.293	0.178

formula consistency, indicator interpretation). Table 4.5 and Figure 4.5 report the corresponding statistics.

Structure. Structural reconstruction is the strongest diagnostic dimension for frontier LLMs. Gemini 3 leads at 0.844, followed by Sonnet 4.6 at 0.828, both with a median of 0.880. Among the pipelines, Docling achieves 0.607, benefiting from its dedicated TableFormer module, while PyMuPDF4LLM reaches only 0.396, reflecting the inability of Markdown-based table representations to encode merged cells or hierarchical headers.

Content. Content accuracy follows a similar ranking, but with notable reversals. Sonnet 4.6 leads at 0.808, ahead of Gemini 3 at 0.797, suggesting that Sonnet 4.6 is slightly more precise on cell values despite its marginally weaker layout reconstruction. OpenDataLoader records a content mean of 0.682, well above its structural score (0.577), indicating that it often extracts values reasonably well even from imperfectly reconstructed tables. Gemma 3 shows the same pattern in reverse: content (0.454) exceeds structure (0.329), suggesting that its language modelling capabilities help transcribe cell values even when table layout is not properly recovered.

Semantic. The semantic dimension is the most challenging and shows the widest cross-family variation. Only Sonnet 4.6 (0.723) and Gemini 3 (0.714) exceed 0.70. One remarkable result concerns the syntactic analysis pipelines: OpenDataLoader (0.720) and Docling (0.710) achieve semantic scores comparable to those of leading large language models (LLMs), likely due to the level of post-processing involving the classification of shared values described in section 3.4.4. In

Table 4.5: Diagnostic scores, reporting mean, median, and interquartile range (IQR), grouped by methodological category.

	Structure			Content			Semantic		
Model	Mean	Median	IQR	Mean	Median	IQR	Mean	Median	IQR
Frontier online LLMs									
Sonnet 4.6	0.828	0.880	0.236	0.808	0.878	0.275	0.723	0.735	0.273
Gemini 3	0.844	0.880	0.264	0.797	0.817	0.250	0.714	0.671	0.397
Sonnet 4.5	0.757	0.847	0.327	0.722	0.801	0.334	0.581	0.578	0.350
GPT-5.2	0.721	0.787	0.400	0.704	0.772	0.362	0.503	0.527	0.294
Gemini 2.5	0.674	0.652	0.324	0.686	0.708	0.244	0.471	0.479	0.349
GPT-5.1	0.706	0.742	0.285	0.668	0.700	0.387	0.432	0.424	0.404
Document parsing and OCR pipelines									
OpenDataLoader (hi_res)	0.577	0.592	0.471	0.682	0.722	0.352	0.720	0.792	0.407
Docling	0.607	0.638	0.409	0.640	0.627	0.261	0.710	0.779	0.387
Unstructured	0.474	0.475	0.286	0.535	0.556	0.279	0.568	0.661	0.389
PyMuPDF4LLM	0.396	0.324	0.369	0.455	0.423	0.351	0.416	0.397	0.592
GLM-OCR	0.517	0.553	0.309	0.414	0.386	0.268	0.515	0.620	0.526
Open-weight LLM									
Gemma 3	0.329	0.336	0.205	0.454	0.526	0.345	0.115	0.000	0.268
Table-specific baseline									
TATR	0.385	0.359	0.373	0.357	0.327	0.248	0.414	0.481	0.734

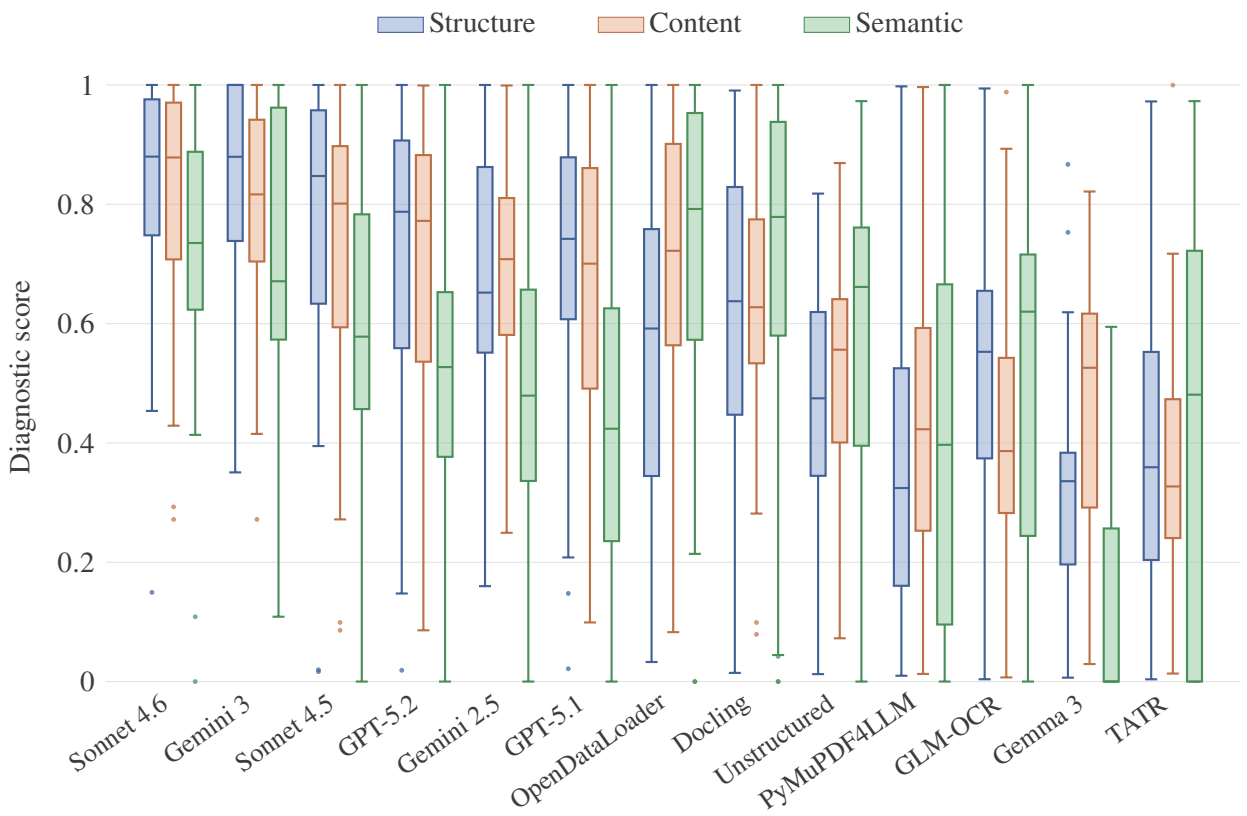

Figure 4.5: Distribution of per-document diagnostic scores for structure, content, and semantic dimensions across the evaluated systems. Boxes show the interquartile range, the central line marks the median, whiskers follow the 1.5 IQR rule, and dots indicate outliers.

contrast, Gemma 3 records a semantic mean of 0.115 with a median of 0.000, indicating that for the majority of documents it produces no semantically correct table output, a fundamental limitation that its reasonable text and content scores cannot compensate.

4.4 Per-Model Capability Analysis

The aggregate metrics of the preceding sections are complemented here by a synthesis of the recurring strengths and failure modes observed in the per-document evaluation reports, organised by system family.

Frontier LLMs

Sonnet 4.6. Sonnet 4.6 delivers the most balanced profile. Text extraction is near-perfect, with around 75% of documents scoring 1.0 and virtually no block-level misses or hallucinations. Approximately 57% of tables score above 0.90, but a remaining around 12% falls below 0.50 on documents with complex hierarchical headers or graphical indicators. The residual errors are predominantly representational rather than semantic — cell-type confusion and number-format locale mismatches. Merged-cell spans are occasionally flattened and footnote markers missed or hallucinated, but these affect a small fraction of cells. Performance does not degrade on longer documents; the hardest cases are single-page reports packing multiple complex tables with graphical indicators.

Gemini 3. Gemini 3 matches Sonnet 4.6 globally and leads on table extraction (78.2% mean), but exhibits a sharp size-dependent degradation: small tables (≤6 rows) average around 87%, whereas tables exceeding 15 rows collapse to around 37%, with the model reducing row counts and hallucinating extra columns. Number-locale omission and cell-type confusion dominate the cell-level errors. Footnote hallucination accounts for around 67% of footnote errors, while graphical indicators are often interpreted at a generic symbolic level rather than mapped to their intended document-specific meaning, suggesting limited semantic.

Sonnet 4.5. Sonnet 4.5 achieves reliable text extraction (around 71% of documents at a perfect score) but fails to produce any output for around 17% of the corpus (the largest PDFs), suggesting context-window constraints. Formula recognition is its weakest capability: 96% of expected formulas are undetected, and only a few achieve near-perfect recovery. Cell-level hallucinations appear in around 34% of documents, including fully fabricated table with plausible but incorrect ESG data. Graphical indicators trigger type confusion, swapping icon and text classifications between adjacent columns. The semantic diagnostic (0.581) lags behind structure (0.757) and content (0.722), confirming that higher-level interpretive tasks remain challenging for this model.

GPT-5.2. GPT-5.2 produces valid output for around 83% of the corpus, with strong text extraction (median 1.0) and a bimodal table distribution (around 39% above 0.90, around 15% below 0.50 driven by structurally complex tables). The dominant errors are number-format loss, raw-content

mismatches, and cell-role confusion. Formula preservation is poor (89% dropped), and boolean indicators are misclassified as free text. The semantic axis (0.503) is the weakest, with metadata and interpretive annotations as the principal bottleneck.

Gemini 2.5. Gemini 2.5 achieves strong text extraction (96% mean) and a perfect row-level alignment, even if around 75% of matched rows contain at least one cell-level error, revealing pervasive per-cell inaccuracies despite correct row detection. The error profile is dominated by metadata-layer failures: format-locale loss, cell-type omission, and footnote non-recovery (around 50% miss rate). Formula recognition is weak (around 79% miss rate), and when raw values are wrong the errors tend to be severe (77% below 0.5 similarity). The semantic diagnostic (0.471) is substantially below the structural (0.674) and content (0.686) axes, positioning the interpretive layer as the model's clear bottleneck.

GPT-5.1. GPT-5.1 adopts a conservative extraction strategy — precision exceeds recall by nine points — omitting content rather than fabricating it, but around 20% of tables are entirely absent on long documents. The semantic dimension is the weakest axis (mean 0.432, with graphical indicators routinely omitted), with 47% of documents scoring below 0.30 despite reasonable structural reconstruction above 0.70 in 60% of cases.

Parsing Pipelines

OpenDataLoader. OpenDataLoader delivers reliable text extraction (88% mean) and competitive semantic classification via the shared post-processing layer, but its table reconstruction is highly inconsistent (IQR 46.4 pp). The main structural failure is column collapse: 42% of tables exhibit column-count mismatches, with 90% of those predicting fewer columns than expected, reflecting the inability to resolve multi-level headers. Row loss affects many expected rows, and 92% of footnotes are missed or incorrectly transcribed, struggling on multi-page table layouts.

Docling. Docling benefits from its dedicated TableFormer module (structure score 0.607) and achieves a semantic score (0.710) comparable to frontier LLMs thanks to the shared post-processing layer. Its main weaknesses are footnote recovery (83% missed), merged-cell handling, and subtitle

detection with a persistent title-subtitle confusion. Long documents degrade primarily on structural scores while semantic scores remain stable.

Unstructured. Unstructured achieves strong narrative fidelity (93.3% of blocks at perfect similarity) but the weakest table structure among the pipelines: row counts disagree in 58.3% of tables, and 62.9% have at least one layout mismatch. Graphical indicators are a near-complete failure (91.7% misclassified, around 50% producing empty output), reflecting Tesseract's inability to decode vector-graphic symbols. Formula cells are missed in around 90% of cases. Footnote handling is pervasively weak (83% of tables with at least one discrepancy). Among numerical misreads, many show order-of-magnitude errors from misplaced decimal separators.

PyMuPDF4LLM. PyMuPDF4LLM leverages the native PDF text layer for approximately 90% text fidelity at negligible cost, but table extraction is fundamentally limited by Markdown's flatness: merged cells, hierarchical headers, and column spans are systematically lost (structural score below 0.40), one third of tables are completely missed, and many of detected cells lose their content during Markdown conversion.

GLM-OCR. GLM-OCR operates through page-level image processing, achieving adequate narrative extraction (84% of narrative blocks above 0.95) but the weakest text score (0.686) owing to high block miss and spurious insertion rates. All graphical indicators are re-classified as boolean or plain text, destroying icon semantics entirely. Table precision exceeds recall (0.556 vs. 0.334), confirming that the model captures visually salient regions while missing a large fraction of content. Rowspan reconstruction fails universally — a limitation inherent to the Markdown intermediate representation —, and no table in the benchmark achieves a correct footnote match.

Open-Weight LLM and Table-Specific Baseline

Gemma 3 27B-IT. Gemma 3 presents the sharpest capability asymmetry: competitive text extraction (around 79% of documents above 0.90) coexists with a near-zero semantic table score (mean 0.115, median 0.0). Table fragmentation—splitting a single complex table into multiple misaligned structures—affects 53% of documents, and around 35% of ground-truth tables are entirely

missing. Graphical indicators are either omitted or transcribed as unrelated values, and merged-cell attributes (rowspan, colspan) are almost never predicted. The model fails to produce output for 21% of the corpus, all longer reports, consistent with context-window overflow. Gemma 3 achieves the lowest standard deviation (0.135), a consequence of consistently low scores rather than robust performance.

Table Transformer (TaTR). Despite its specialised architecture, TaTR records the lowest overall score (0.340). The table miss rate reaches around 43%, row counts disagree in 48% of matched tables, with rows predominantly lost rather than hallucinated, and Tesseract-induced OCR artefacts cause around 27% of raw-value differences to exhibit severe distortion (similarity below 0.70). Graphical indicators, inter-cell formulas (90% at zero score), and footnotes (86% missing) are all near-total failure modes. These weaknesses compound multiplicatively, demonstrating that a detection-focused pipeline trained on generic datasets transfers poorly to ESG reporting requirements.

4.5 Discussion

Frontier multimodal LLMs provide the strongest end-to-end solution. Sonnet 4.6 and Gemini 3 clearly occupy the top of the ranking, with overall scores above 0.83, and table scores close to 0.78. On the most consequential dimension for ESG reporting — table extraction — they outperform the best parsing pipelines by more than 20 points. On this benchmark, this suggests that unified multimodal models are better able to integrate layout, textual content, and local visual cues than modular pipelines.

Table extraction remains the central bottleneck, even when text extraction is nearly solved. Across all system families, text extraction is consistently stronger than table extraction. For the strongest models, text performance is close to saturation, whereas table performance remains substantially lower and far more variable. The gap is not only quantitative but also qualitative: the dominant failures concern multi-level headers, merged cells, footnotes, graphical indicators, and large tables.

The diagnostic decomposition reveals different strengths for LLMs and traditional parsers. One of the most informative findings is that the ordering of the diagnostic dimensions differs by system family. For frontier LLMs, performance typically follows the pattern *structure* > *content* > *semantic*, indicating that global layout and cell alignment are often recovered before metadata-rich interpretation becomes fully reliable. For pipelines such as Docling and OpenDataLoader, the pattern tends instead toward *semantic* > *content* > *structure*. This should not be seen as an inconsistency of the metric, but as a meaningful behavioural distinction: these systems often preserve textual surface forms and, after the shared post-processing layer, can recover units, types, or simple semantic labels reasonably well, while still struggling with rowspan, colspan, header hierarchy, and table boundaries.

Precision-recall balance and dispersion are also crucial. The results show that average score alone is not sufficient for system selection. Sonnet 4.6 and Gemini 3 are not only the most accurate systems, but also among the most stable. By contrast, OpenDataLoader combines a competitive mean with the largest dispersion, while GLM-OCR exhibits a marked precision-recall asymmetry. A recall-deficient system tends to omit relevant evidence; a precision-deficient system tends to hallucinate or over-segment structure; and a high-variance system behaves unpredictably across documents. In regulated settings such as ESG reporting, this distinction is practically important, because even a model with acceptable mean performance may remain unsuitable if its failures are difficult to anticipate.

The practical implication. When the objective is end-to-end extraction from ESG reports with limited manual intervention, frontier multimodal LLMs currently appear to be the most reliable option. At the same time, the intermediate parsing pipelines, like Docling and OpenDataLoader, remain competitive on narrative text and, once coupled with the shared post-processing layer, approach frontier LLMs on the semantic dimension. However, this advantage is partly tied to the fact that the post-processing stage for these systems, as well as for the other non-frontier multimodal LLMs, was tailored to the structure of the benchmark proposed in this work. On a much more heterogeneous benchmark, requiring more complex normalization and reconstruction rules, their performance may prove less stable and less easily transferable. Modular systems may therefore

remain attractive when transparency, controllability, or cost constraints are prioritised, but their practical adoption should also consider the effort needed to adapt post-processing to new document settings. Conversely, the poor transfer of TaTR and the near-zero semantic behaviour of Gemma 3 indicate that strength in only one component of the task — detection and strong text capabilities — is not sufficient for robust ESG document understanding.

The results reinforce the need for domain-specific evaluation. Several failure modes that are decisive in practice — omitted units, broken formulas, unresolved footnotes, or incorrect interpretation of graphical indicators — would be only weakly reflected by conventional text-oriented evaluation. By decomposing quality into structure, content, and semantic dimensions, and by assigning greater weight to critical cells, the proposed framework produces conclusions that are better aligned with the actual risks of automated ESG reporting. The benchmark is not limited to ranking systems, it also indicates where future improvements are most needed, namely robust table-structure recovery, metadata-sensitive extraction, and better grounding of document-specific.

Expanding the synthetic dataset The synthetic dataset used in this work remains relatively small in scale, and do not yet capture the full diversity of real-world ESG reports. It can be expanded by integrating additional noise elements such as page headers, footers, and footnotes that act as distractors in retrieval settings More complex layouts — including highly disorganized arrangements of text and table blocks on the same page, variable text orientations within individual cells, and heterogeneous font styles carrying distinct semantic roles within a single document — would better reflect the diversity of real sustainability disclosures. and custom graphical logos requiring interpretation. The inclusion of custom graphical logos and their interpretation, as well as richer footnote structures and cross-references, would further stress-test the robustness of parsing systems. These extensions are directly supported by the modular architecture of the generator, which allows new templates and structural mutations to be added without redesigning the pipeline.

5. Conclusions and Future Work

To support the implementation of RAGs within organizations and improve the automated document understanding in ESG reporting, this work developed three key components: (i) a fully automated pipeline for generating synthetic ESG documents in PDF format, each paired with a complete JSON ground truth encoding hierarchical row groups, multi-level headers, formula-based aggregation rows, merged cells, and heterogeneous value types; (ii) a domain-specific hierarchical evaluation framework that decomposes extraction quality into structure, content, and semantic dimensions, weighting errors by cell criticality; and (iii) a unified empirical comparison of thirteen systems from four methodological families—frontier multimodal LLMs, document parsing pipelines, an open-weight vision-language model, and a table-specific detection baseline. Both the dataset generator and the evaluation framework are containerized through Docker for portability and reproducibility.

The results showed that frontier multimodal LLMs provide the most accurate and consistent performance. Sonnet 4.6 and Gemini 3 achieved mean overall scores above 0.83 with the lowest standard deviations. Text extraction proved largely solved across most families, whereas table extraction remained the central bottleneck, with dominant failures on multi-level headers, merged cells, footnotes, graphical indicators, and large tables. The analysis also highlighted that precision-recall balance and score dispersion matter for system selection in regulated settings, since acceptable mean performance may still be unsuitable if failures are unpredictable.

Future Work. Two directions for future work emerge. The first concerns expanding the synthetic dataset with additional noise elements such as page headers, more complex layouts, variable text orientations, custom graphical logos—extensions already supported by the modular architecture of the generator. The second concerns integrating extracted structured content into retrieval-augmented generation pipelines that preserve the relational semantics of tables. Table-RAG, Graph-RAG, Hybrid-RAG, and relational foundation models such as KumoRFM are all promising directions for ESG reporting. However, the findings of this thesis show that the effectiveness of these downstream approaches depends strongly on the quality of upstream parsing, because inaccurate extraction can compromise the structured representation on which retrieval and reasoning rely.

Bibliography

[1] Narayan S Adhikari and Shradha Agarwal. “A comparative study of PDF parsing tools across diverse document categories”. In: *arXiv preprint arXiv:2410.09871* (2024).

[2] Christoph Auer et al. “Docling technical report”. In: *arXiv preprint arXiv:2408.09869* (2024).

[3] **Giorgia Bertacchini, Marcello Pietri, and Marco Mamei. “AI-Driven Engineering for Urban Problem Solving: Recent Advances and Case Studies”. In: *Journal of AI-Driven Communication Engineering* 1 (2025), pp. 72–76.**

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. “End-to-end object detection with transformers”. In: *European conference on computer vision*. Springer. 2020, pp. 213–229.

[5] Zhiyu Chen et al. “Finqa: A dataset of numerical reasoning over financial data”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 3697–3711.

[6] Matthias Fey, Vid Kocijan, Federico Lopez, J Lenssen, and Jure Leskovec. *Kumorfm: A foundation model for in-context learning on relational data*. 2025.

[7] Zijin Hong et al. “Next-generation database interfaces: A survey of llm-based text-to-sql”. In: *IEEE Transactions on Knowledge and Data Engineering* (2025).

[8] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. “Grag: Graph retrieval-augmented generation”. In: *Findings of the Association for Computational Linguistics: NAACL 2025*. 2025, pp. 4145–4157.

[9] Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. “Layoutlmv3: Pre-training for document ai with unified text and image masking”. In: *Proceedings of the 30th ACM international conference on multimedia*. 2022, pp. 4083–4091.

[10] Vladimir Karpukhin et al. “Dense passage retrieval for open-domain question answering”. In: *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*. 2020, pp. 6769–6781.

[11] Vladimir I Levenshtein et al. “Binary codes capable of correcting deletions, insertions, and reversals”. In: *Soviet physics doklady*. Vol. 10. 8. Soviet Union. 1966, pp. 707–710.

[12] Patrick Lewis et al. “Retrieval-augmented generation for knowledge-intensive nlp tasks”. In: *Advances in neural information processing systems* 33 (2020), pp. 9459–9474.

[13] Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li. “TableBank: Table Benchmark for Image-based Table Detection and Recognition”. In: *CoRR* abs/1903.01949 (2019). arXiv: 1903.01949. URL: http://arxiv.org/abs/1903.01949.

[14] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. “Visual instruction tuning”. In: *Advances in neural information processing systems* 36 (2023), pp. 34892–34916.

[15] Ahmed Nassar, Nikolaos Livathinos, Maksym Lysak, and Peter Staar. *TableFormer: Table Structure Understanding with Transformers*. 2022. arXiv: 2203.01017 [cs.CV]. URL: https://arxiv.org/abs/2203.01017.

[16] Guilherme Nunes, Vitor Rolla, Duarte Pereira, Vasco Alves, André Carreiro, and Márcia Baptista. “Benchmarking Table Extraction: Multimodal LLMs vs Traditional OCR”. In: *Proceedings of the 1st Joint Workshop on Large Language Models and Structure Modeling (XLLM 2025)*. 2025, pp. 8–15.

[17] **Marcello Pietri, Matteo Martinelli, Fabio Turazza, Giorgia Bertacchini, Marco Picone, and Marco Mamei. “Digital Twins and Federated Learning in Industrial IoT: Worker-Centric Safety Perspectives”. In: *2026 IEEE 23rd Consumer Communications & Networking Conference (CCNC)*. IEEE. 2026, pp. 1–6.**

[18] Varshini Reddy, Rik Koncel-Kedziorski, Viet Dac Lai, Michael Krumdick, Charles Lovering, and Chris Tanner. “Docfinqa: A long-context financial reasoning dataset”. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2024, pp. 445–458.

[19] Bhaskarjit Sarmah, Dhagash Mehta, Benika Hall, Rohan Rao, Sunil Patel, and Stefano Pasquali. “Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction”. In: *Proceedings of the 5th ACM International Conference on AI in Finance*. 2024, pp. 608–616.

[20] Ray Smith. “An overview of the Tesseract OCR engine”. In: *Ninth international conference on document analysis and recognition (ICDAR 2007)*. Vol. 2. IEEE. 2007, pp. 629–633.

[21] Brandon Smock, Rohith Pesala, and Robin Abraham. “GriTS: Grid table similarity metric for table structure recognition”. In: *arXiv preprint arXiv:2203.12555* (2022).

[22] Brandon Smock, Rohith Pesala, and Robin Abraham. “PubTables-1M: Towards comprehensive table extraction from unstructured documents”. In: *CoRR* abs/2110.00061 (2021). arXiv: 2110.00061. URL: https://arxiv.org/abs/2110.00061.

[23] Marijan Soric, Cécile Gracianne, Ioana Manolescu, and Pierre Senellart. “Benchmarking Table Extraction from Heterogeneous Scientific Extraction Documents”. In: *arXiv preprint arXiv:2511.16134* (2025).

[24] Zilong Wang, Yichao Zhou, Wei Wei, Chen-Yu Lee, and Sandeep Tata. “Vrdu: A benchmark for visually-rich document understanding”. In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2023, pp. 5184–5193.

[25] Hendrik Weichel, Martin Simon, and Jörg Schäfer. “Robust Table Information Extraction from Sustainability Reports: A Time-Aware Hybrid Two-Step Approach”. In: *Proceedings of the 2nd Workshop on Natural Language Processing Meets Climate Change (ClimateNLP 2025)*. 2025, pp. 233–244.

[26] Derong Xu et al. “Large language models for generative information extraction: A survey”. In: *Frontiers of Computer Science* 18.6 (2024), p. 186357.

[27] Xiaohan Yu, Pu Jian, and Chong Chen. “Tablerag: A retrieval augmented generation framework for heterogeneous document reasoning”. In: *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 2025, pp. 14074–14093.

[28] Yian Zhao et al. “Detrs beat yolos on real-time object detection”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2024, pp. 16965–16974.

[29] Xinyi Zheng, Douglas Burdick, Lucian Popa, and Nancy Xin Ru Wang. “Global Table Extractor (GTE): A Framework for Joint Table Identification and Cell Structure Recognition Using Visual Context”. In: *CoRR* abs/2005.00589 (2020). arXiv: 2005.00589. URL: https://arxiv.org/abs/2005.00589.

[30] Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno-Yepes. “Image-based table recognition: data, model, and evaluation”. In: *CoRR* abs/1911.10683 (2019). arXiv: 1911.10683. URL: http://arxiv.org/abs/1911.10683.

[31] Jiaru Zou et al. “GTR: graph-table-rag for cross-table question answering”. In: *arXiv e-prints* (2025), arXiv–2504.

[32] Yi Zou et al. “ESGReveal: An LLM-based approach for extracting structured data from ESG reports”. In: *Journal of Cleaner Production* 489 (2025), p. 144572.