

Do Agents Help? Agentic Document Information Extraction with Enterprise Database Grounding

Zhijie Zhang and Hongxin Shao and Mattis Lennart Geiger

Tao Bai and Daniel Dahlmeier

SAP

{zhijie.zhang, hongxin.shao, mattis.geiger}@sap.com

{tao.bai, d.dahlmeier}@sap.com

Abstract

In enterprise applications, document information extraction (IE) does not operate in isolation: extracted information must match the *systems of record* (SoR) for the process to work. While reliable IE demands entity extraction, SoR grounding, and iterative refinement, existing methods rely on single-pass inference. To address this gap, we propose a SoR-grounded agentic pipeline and evaluate four agentic paradigms (Agentic Workflow, Re-Act, Plan-Execute-Critic, and Dynamic Agent Collaboration) on real-world enterprise data. We focus on the goods receipt and invoice settlement processes as representative use cases, where delivery notes and invoices must be matched and verified against purchase order SoR. Our results show that agentic approaches generally match or exceed non-agentic baselines, with simpler paradigms matching or outperforming complex alternatives. Paired with systematic error analysis, this work provides actionable guidance for deploying agentic document AI at enterprise scale.

1 Introduction

Enterprise automation relies on extracting data from business documents and matching them accurately with *Systems of Record* (SoR), such as Enterprise Resource Planning (ERP) systems. However, while enterprise processes demand near-faultless Information Extraction (IE) to automate effectively, state-of-the-art models typically achieve only 70% to 90% accuracy on real-world documents (Shen et al., 2026). This performance bottleneck is driven by the heterogeneous nature of business documents, which frequently contain ambiguous fields, abbreviated units, or system-specific codes that are uninterpretable without context (Wang et al., 2025b).

To bridge this gap, extractions must be grounded and cross-verified against the SoR, where a data match can verify correctness and resolve ambiguities. However, executing this cross-verification

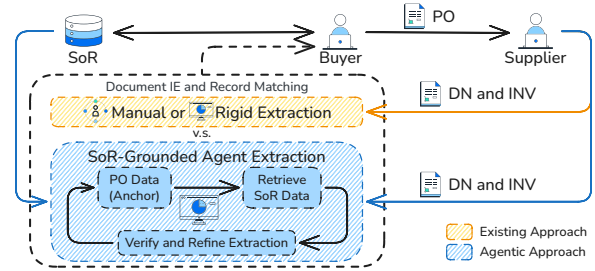


Figure 1: Existing single-pass vs. proposed SoR-grounded agentic document IE pipelines.

requires dynamic, multi-step reasoning to query databases, match fields, and resolve discrepancies. Modern LLM and VLM approaches execute single-pass IE without iterative inference (Hu et al., 2025; Cui et al., 2025). Concurrently, while agentic frameworks are designed for iterative reasoning, they remain confined to conversational or generic planning tasks (Han et al., 2025; Yu et al., 2025). To date, no SoR-grounded agentic paradigm for enterprise document IE has been systematically studied or evaluated. This paper addresses this gap by answering a critical question: *Can SoR-grounded agentic frameworks improve enterprise document IE compared to robust non-agentic baselines?*

We study the goods receipt and invoice settlement process as a representative enterprise use case: a buyer creates and records a purchase order (PO) in their ERP system, and the supplier issues a delivery note and invoice referencing that PO. Upon receipt, the buyer must parse, match, and verify values across these incoming documents against the ERP records (Fig. 1). Our central insight is that automated agentic extraction should mirror the process human experts use to parse, match, and verify document values against SoR. We therefore propose a SoR-grounded agentic document IE method instantiated for the goods receipt process.

To our knowledge, this work presents the first systematic study of grounded agentic paradigms

on real enterprise documents. We benchmark four agentic paradigms, *i.e.*, Agentic Workflow (AW), ReAct, Plan-Execute-Critic (PlanExe), and Dynamic Multi-Agent Collaboration (DAC), against robust non-agentic baselines. Each paradigm is equipped with enterprise-specific tools: a perception tool (OCR), an intelligent tool (LLMs), and execution tools (ERP fetcher and verifier). We report a comprehensive evaluation across F_1 , latency, cost, and error modes. Our key contributions are:

- We introduce a novel SoR-grounded agentic pipeline that uses enterprise context to iteratively refine document extraction via cross-verification.
- We present a systematic evaluation of four agentic paradigms on production-grade enterprise documents, demonstrating the strengths of SoR-grounded agents.
- We quantify paradigm-specific failures and translate them into deployment guidance for deploying agentic document AI at enterprise scale.

2 Related Work

Document Information Extraction. Conventional document IE follows a two-stage paradigm: OCR converts document images to text, followed by parsing text to structured data with CNN or transformer encoder-based models (Reisswig et al., 2019; Xu et al., 2020; Appalaraju et al., 2021; Liao et al., 2023). Recent LLMs have reshaped the landscape: decoder-based architectures (Wang et al., 2024b; Liao et al., 2024; Qi et al., 2024) and prompt-driven extraction (Wang et al., 2023b; Perot et al., 2024; Zhou et al., 2024) show strong performance. VLMs further advance document parsing (Poznanski et al., 2025; Nacson et al., 2025; Wei et al., 2025; Wang et al., 2024a). While existing works have shown promising results, they still face significant challenges in complex enterprise documents.

Agentic Document AI. Agent methods have shown strong performance on general tasks (Team, 2025). Paradigms like ReAct (Yao et al., 2023) and Plan-and-Execute (Wang et al., 2023a) enable interleaved reasoning and action, while Toolformer (Schick et al., 2023) and Hugging-GPT (Shen et al., 2023) coordinate multiple tools and models for adaptive extraction. Reflection (Shinn et al., 2023) mechanisms and multi-agent collaboration further improve reliability and task specialisation. Studies like MDocAgent (Han et al., 2025), ChartAgent (Kaur et al., 2025), and MACT (Yu et al., 2025) demonstrate agent effec-

tiveness on document understanding. However, no prior work studies agentic pipelines that iteratively verify and refine extraction against an enterprise SoR on real business documents.

Benchmarks and Datasets. Existing document intelligence benchmarks primarily focus on visual question answering, layout analysis, or isolated IE tasks, for instance, FUNSD (Jaume et al., 2019). While recent benchmarks have scaled up in size and diversity, spanning extensive QA pairs in DocVQA (Mathew et al., 2021) and MMDocBench (Zhu et al., 2024), dense layout and recognition annotations in OmniDocBench (Ouyang et al., 2025), and business-centric tasks in BuDDIE (Wang et al., 2025a), they fundamentally treat documents as isolated entities, without providing external grounding annotations. Consequently, the evaluation of agentic pipelines interacting with enterprise environments remains unrepresented.

3 SoR-Grounded Agentic Document IE

3.1 Task Formulation

In enterprise processes, parties exchange documents that reference business objects managed in SoR. A core IE challenge is that extracted values must be matched and verified against the SoR. We study the goods receipt and invoice settlement process as an instance of this general pattern, where a delivery note and an invoice must be matched against the buyer’s PO in the ERP system.

We formalise enterprise document IE as a structured extraction problem that maps an unstructured document image D to a structured output Y conforming to a predefined schema $\mathcal{S}$. The output $Y = \{Y^h, Y^{li}\}$ comprises header field-value pairs $Y^h = \{(f_i, v_i)\}_{i=1}^{n_h}$ and a list of line items $Y^{li} = \{Y_j^{li}\}_{j=1}^{n_{li}}$, where each item $Y_j^{li} = \{(f_{j,k}, v_{j,k})\}_{k=1}^{n_f}$. Given optional access to a tool bank $\mathcal{T}$, Y is generated via one of two pipelines:

$$Y = \begin{cases} \Theta(D; \mathcal{S}) & \text{Non-Agentic} \\ \mathcal{A}(D, \mathcal{S}, \mathcal{T}) & \text{Agentic} \end{cases} \quad (1)$$

where Θ is a single-pass LLM and $\mathcal{A}$ is an agentic system that iterates over a sequence of steps. At each step t , the system observes a state s_t (extractions, tool results, and history), executes an action a_t (planning, tool calls, updates, or inter-agent delegation), receives an observation o_t , and transitions to s_{t+1} until a termination condition is met. This formulation accommodates both single- and multi-

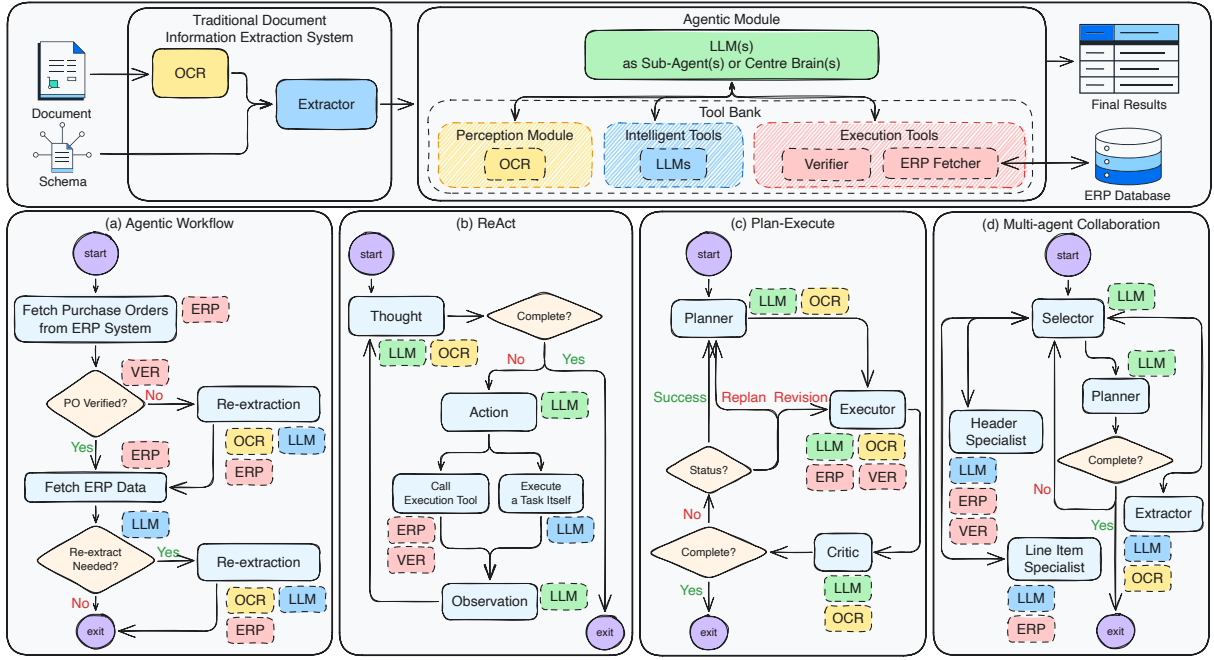


Figure 2: Overview of the agentic extraction framework. The top pipeline shows how the agentic module is plugged into the traditional system, and (a–d) illustrate the structures and data flows.

agent control flows, as well as fixed, plan-guided, or dynamic action schedules.

3.2 Agentic Approaches

Fig. 2 illustrates the paradigm shift from the traditional pipeline to agentic frameworks, where the LLM functions as a central controller orchestrating a tool bank to generate outputs by dynamically interacting with external ERP databases for iterative verification and refinement. We implement and evaluate four agentic architectures spanning a spectrum from structured to flexible:

Agentic Workflow (AW) follows a deterministic sequential pipeline. It retrieves relevant purchase order (PO) candidates from the ERP database and verifies the extracted PO against master data, initiating a targeted re-extraction loop with verified POs as exemplars when verification fails. Once the PO is confirmed, the system fetches master data and compares extractions against the ERP reference, conditionally triggering a second re-extraction if discrepancies are detected.

ReAct (Yao et al., 2023) consists of an agent following a cycle of *thought*, *action*, and *observation*. At each iteration, the agent reasons about its next step, decides to execute a task itself or call a tool in the action phase, and integrates the findings back into its context. The loop ends when the agent produces a final answer or reaches the iteration limit.

PlanExe (Wang et al., 2023a) implements a three-role orchestration loop. A *Planner* analyses the extraction state and generates a plan. An *Executor* carries out each step sequentially, invoking tools while adhering to strict scope discipline. A *Critic* evaluates the *Executor*’s output, verifying that changed values are acceptable. Detected regressions trigger the *Executor* to retry with targeted feedback; fundamentally flawed plans trigger re-planning. The loop terminates when the *Critic* confirms task completion or a maximum iteration limit is reached.

DAC coordinates multiple specialised agents via a group-chat mechanism. A *Planner* agent orchestrates the task and delegates to specialists. A *PO Specialist* verifies the PO against ERP data. A *Line-Item Specialist* compares extracted items against ERP master data. An *Extractor* is the sole agent permitted to update the extraction state, writing to a shared *Registry* that tracks the current extraction and update history. Agent selection at each turn is driven by a *Selector*.

Tool Bank. To address enterprise-specific challenges, *e.g.*, external grounding and semantic ambiguity, agentic systems are equipped with three tool classes: perception tools (a commercial-grade OCR to convert raw images into structured text), intelligence tools (LLMs for planning, disambiguation, and re-extraction), and execution tools (ERP database interface for grounding and verification).

All paradigms share a common base extraction prompt; paradigm-specific prompts govern only the control flow and role instructions. Representative prompts are provided in Appendix A.

4 Datasets and Evaluation Metrics

4.1 Datasets

Our evaluation dataset comprises 195 real-world delivery notes (DNs) from 64 suppliers and 500 invoices (INVs) from 254 suppliers, collected from enterprise transactions. DN contains 5 header and 7 line-item fields, *e.g.*, *deliveryNoteNo*, *purchaseOrderNo*, *li.customerMaterialNo*, and *li.quantity*, while INV has 33 header and 16 line-item fields, *e.g.*, *invoiceNo*, *purchaseOrderNo*, *li.MaterialNo*, and *li.TotalAmount*. The documents feature heterogeneous layouts, multi-column tables, and multi-lingual content. Ground truth follows an annotate-then-review protocol, *i.e.*, one annotator and one reviewer process each document; difficult batches use three-way consensus, followed by rule-based validation. The data were privacy-sanitised by replacing personal, company, and confidential fields while preserving document structure; all examples shown in the appendix are fully synthetic.

ERP Master Data and Augmentation. Due to privacy restrictions on production ERP access, we construct synthetic ERP master data from buyer-side annotations to mirror real-world records. ERP data are scoped to PO information, *i.e.*, they contain only fields available to the buyer (*e.g.*, material numbers) but exclude fields that exist only on incoming documents (*e.g.*, delivery note number and invoice date), giving agents a realistic but finite lookup space. The resulting master data contain 188 and 449 records for DN and INV, reflecting one-to-many mappings between POs and associated documents. Accessing the ERP data still requires the agent to identify the relevant PO from noisy OCR output, validate it by exact matching against 187 or 448 distractors, and only then retrieve the corresponding record. The correct record is available for every PO-bearing document.

To simulate production-level noise, we apply two augmentation strategies: (a) Masking: blanks field values to replicate incomplete records, and (b) Semantic Rewriting: rephrases natural language fields and varies units of measure across ISO codes and abbreviations, *e.g.*, Pieces on the invoice becomes EA in the ERP. At least one field is modified in 69% of DNs and 80% of INVs, preventing direct

copying while preserving an authoritative cross-check. Appendix B gives the full protocol and sensitivity analysis.

Experimental Design. Our evaluation follows a two-stage design. We establish non-agentic baselines across six LLMs on DN, identifying Gemini 2.5 Pro as the strongest, and use it as the non-agentic baseline, then run all four paradigms with the same six backbones (24 configs). For the INV study, we use the strongest DN baseline, Gemini 2.5 Pro, and evaluate all paradigms under thinking and non-thinking settings, resulting in eight configurations. To test whether this strong baseline compresses gains, we use GPT-4o as a weaker INV baseline and run all paradigms with GPT-4o and Gemini 2.5 Pro agentic backbones. Finally, a one-shot context-injection ablation receives the same deterministically retrieved ERP record as AW but has no refinement loop, isolating static context from iterative verification. Appendix C and E reports all configurations and implementation details.

4.2 Evaluation Metrics

Extraction performance is evaluated by the micro-averaged exact-match F_1 score. We assess gains with paired document-level bootstrap intervals ($B = 10,000$), sharing resampled document indices across the LLMs averaged within each paradigm. We call a gain significant when the 95% interval for ΔF_1 excludes zero; this quantifies document-sampling uncertainty, not run-to-run model variance. Appendix D.1 reports the full intervals, and Appendix D.4 tests fuzzy matching on free-text fields. To evaluate operational efficiency, we measure latency (seconds/doc), aggregate token consumption (input/output/thinking token counts), and total cost per document.¹ To minimise the impact of outlier API response times, we report the median for all efficiency metrics.

5 Results

Paradigm-level performance summaries and operational results are shown in Table 1 and 2. Fig. 3 (a–b) show F_1 per paradigm-model configuration. Detailed results are provided in Appendix E and F.

5.1 Delivery Notes

All paradigms significantly improve on DN. AW achieves the best average F_1 (0.864), while ReAct+Gemini 2.5 Pro+Think reaches the best F_1

¹Costs are calculated based on official provider documentation as of February 2026.

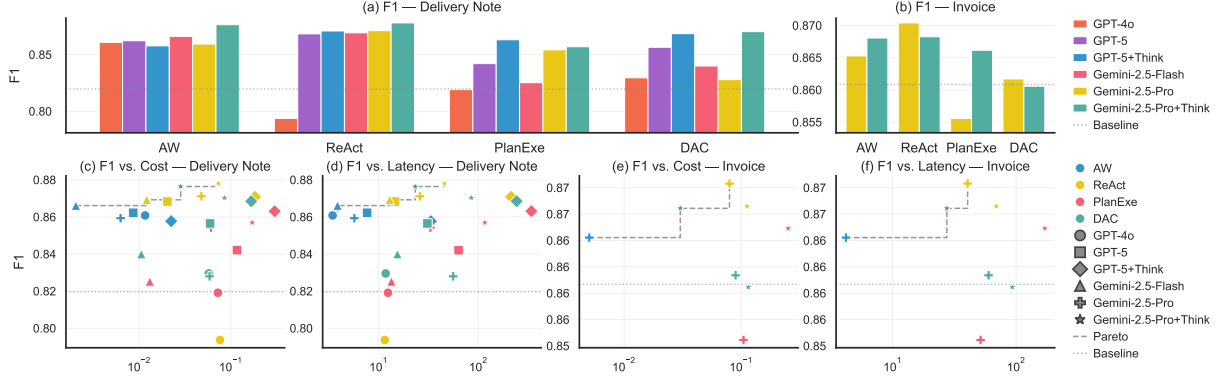


Figure 3: Performance overview and cost-latency Pareto frontier for all agentic paradigms. (a–b): F_1 for each paradigm-model on DN and INV; the dotted line marks the non-agentic baseline (BL). (c–d): F_1 vs. cost and latency on DN; each point is one paradigm-model configuration (colour = paradigm, marker = backbone model), the dashed step curve marks the Pareto frontier, and the dotted line is the BL. (e–f): Same cost and latency Pareto plots for INV.

Paradigm	Best Config Across All LLMs		Avg. Across All LLMs	
	Best Model	F_1 (Δ BL) $\uparrow$	F_1 (Δ BL) $\uparrow$	
<i>Delivery Note (baseline: 0.820)</i>				
AW	Gem 2.5 Pro+Think	0.876 (+0.056)	0.864 (+0.044)	
ReAct	Gem 2.5 Pro+Think	0.878 (+0.058)	0.859 (+0.039)	
PlanExe	GPT-5+Think	0.863 (+0.043)	0.843 (+0.024)	
DAC	Gem 2.5 Pro+Think	0.870 (+0.050)	0.849 (+0.029)	
<i>Invoice (baseline: 0.861)</i>				
AW	Gem 2.5 Pro+Think	0.868 (+0.007)	0.867 (+0.006)	
ReAct	Gem 2.5 Pro	0.870 (+0.009)	0.869 (+0.008)	
PlanExe	Gem 2.5 Pro+Think	0.866 (+0.005)	0.861 (+0.000)	
DAC	Gem 2.5 Pro	0.862 (+0.001)	0.861 (+0.000)	

Table 1: Paradigm-level results, showing the peak F_1 across all LLMs and the model achieving it, as well as the mean F_1 over all LLMs.

Models	AW			ReAct			PlanExe			DAC		
	L	C	T	L	C	T	L	C	T	L	C	T
Gem 2.5 Flash	3.9	0.002	317	13.0	0.012	2,212	13.5	0.013	1,235	15.5	0.011	795
Gem 2.5 Pro	5.7	0.006	279	26.0	0.048	2,021	33.1	0.061	1,883	56.0	0.059	272
Gem 2.5 Pro+Think	23.3	0.028	2,383	45.7	0.073	4,567	116.2	0.171	12,084	85.0	0.085	4,423
GPT-4o	3.5	0.012	208	11.5	0.076	1,161	12.4	0.072	1,126	11.8	0.058	652
GPT-5	7.7	0.009	357	14.7	0.020	740	63.4	0.117	3,844	30.9	0.059	1,367
GPT-5+Think	33.5	0.022	1,949	213.7	0.183	14,649	340.8	0.300	23,239	242.6	0.167	11,308

Table 2: Latency (L), cost per document (C), and output+thinking token counts (T) on delivery notes.

(0.878). Paired bootstrap confidence intervals for the average gains exclude zero for every paradigm, with lower bounds from +1.7 pp (PlanExe) to +3.2 pp (AW); full intervals are in Appendix D.1. PlanExe and DAC perform comparably on average, though DAC has a higher peak configuration. Notably, paradigm selection impacts performance more than model choice: switching paradigms yields greater F_1 variance than changing models within a fixed paradigm.

Simple paradigms dominate the Pareto frontier. Fig. 3(c–d) show the F_1 vs. cost and latency frontier. AW+Gemini 2.5 Flash ($F_1 = 0.866$, \$0.002/doc, 3.9s) captures 98.7% of the maximum F_1 at 2.7% of the maximum cost. Conversely, com-

	AW		ReAct		PlanExe		DAC	
	ΔF_1	Cost $\times$	ΔF_1	Cost $\times$	ΔF_1	Cost $\times$	ΔF_1	Cost $\times$
<i>Delivery Note</i>								
GPT-5	−0.005	2.6 $\times$	+0.003	9.0 $\times$	+0.021	2.6 $\times$	+0.012	2.8 $\times$
Gem 2.5 Pro	+0.017	4.5 $\times$	+0.007	1.5 $\times$	+0.003	2.8 $\times$	+0.042	1.5 $\times$
<i>Invoice</i>								
Gem 2.5 Pro	+0.003	6.0 $\times$	−0.002	1.4 $\times$	+0.011	2.4 $\times$	−0.001	1.3 $\times$

Table 3: Thinking impact. ΔF_1 : Think—No-Think.

plex models introduce high cost and latency penalties for comparable performance.

Extended thinking generally improves performance. Thinking-mode variants improve results in 7 of 8 paradigm-model pairs, with the largest gains from DAC+Gemini 2.5 Pro (Table 3). The only regression occurs for AW+GPT-5, where trace analysis suggests that extra reasoning overwrites correct values. Crucially, thinking most helps complex paradigms (PlanExe +2.1 pp w/ GPT-5, DAC +4.2 pp w/ Gemini 2.5 Pro), suggesting it compensates for reasoning gaps in those designs.

ERP-grounded fields drive agentic gains. Selected field-level results are shown in Table 4 (see Appendix E for full results). Most improvements originate from PO- and material-related fields. AW improves *purchaseOrderNo* by +15.6 pp and *li.customerMaterialNo* by +12.7 pp. Meanwhile, baseline-strong fields (*deliveryNoteNo*) show negligible change, and fields absent from the ERP schema yield marginal or negative Δ due to schema omission and over-thinking. The performance drop in *li.unitOfMeasure* is an exception, largely caused by using less capable models like GPT-4o.

5.2 Invoices

AW and ReAct show significant gains. PlanExe and DAC do not. The invoice baseline from Gem-

Field	AW	ReAct	PlanExe	DAC
<i>Delivery Note</i>				
deliveryDate [†]	0.000	-1.199	-0.476	0.335
deliveryNoteNo [†]	0.000	0.001	0.001	-0.101
documentDate [†]	0.000	-4.574	-3.260	-0.180
purchaseOrderNo	15.602	14.187	8.230	9.659
li.customerMaterialNo	12.686	11.300	4.269	7.019
li.purchaseOrderNo	3.193	20.104	22.351	12.227
li.quantity	-0.197	0.527	0.488	0.403
li.supplierMaterialNo	8.158	6.901	3.935	4.054
li.unitOfMeasure	-0.324	-1.646	-0.503	-1.561
<i>Invoice</i>				
purchaseOrderNo	2.994	5.192	3.226	1.994
vendorName	0.000	6.148	5.020	0.000
paymentTerms	0.000	-0.467	-1.033	-0.063
li.purchaseOrderNo	-7.013	11.665	-0.142	4.697
li.quantity	-0.709	-1.664	-1.702	-0.510
li.description	1.407	0.244	-2.532	-0.998
li.materialNo	3.840	2.428	1.612	1.240
li.unitOfMeasure	0.504	0.386	-1.081	-0.521

Table 4: Per-field ΔF_1 , averaged across models. [†]: fields are not presented in ERP master data.

Paradigm	Agentic Backbone			
	GPT-4o	GPT-4o ΔF_1	Gem 2.5 Pro	Gem 2.5 Pro ΔF_1
AW	0.869	+9.2	0.866	+8.9
ReAct	0.854	+7.7	0.873	+9.6
PlanExe	0.859	+8.2	0.862	+8.5
DAC	0.859	+8.2	0.862	+8.5

Table 5: Invoice with GPT-4o initial extractions (Baseline $F_1 = 0.777$).

ini 2.5 Pro is already strong ($F_1 = 0.861$), leaving less room for improvement. With the best configuration, AW gains +0.7 pp and ReAct gains +0.9 pp, with 95% intervals excluding zero; PlanExe and DAC gains are statistically indistinguishable from zero: PlanExe’s critic over-corrects during verification steps, and DAC exhausts its maximum message budget before updating results. Consistent with DN, simpler paradigms match or outperform complex multi-agent designs.

A weaker baseline reveals substantially larger gains. Replacing the initial extractor with GPT-4o lowers baseline F_1 to 0.777. Table 5 shows that all paradigms then gain +7.7 to 9.6 pp with either GPT-4o or Gemini 2.5 Pro as the agentic backbone. Thus, the invoice result with Gemini 2.5 Pro initial extractions reflects baseline saturation rather than an inherent lack of value from agentic processing. AW leads with GPT-4o, while ReAct leads with the stronger Gemini reasoning backbone.

AW dominates the Pareto frontier. As shown in Fig. 3 (e–f), AW+Gemini 2.5 Pro reaches 99.4% of the maximum F_1 at 6.4% of the maximum cost ($F_1 = 0.865$, \$0.005/doc, 4.2s). In contrast, more complex paradigms like PlanExe+Gemini 2.5 Pro+Think require \$0.251/doc and 171s of latency for $F_1 = 0.866$.

Scope	BL	Context	AW	Ctx–BL	AW–Ctx
<i>Delivery Note</i>					
All fields	0.820	0.837	0.859	+1.8	+2.2
purchaseOrderNo	0.818	0.818	0.974	+0.0	+15.6
<i>Invoice</i>					
All fields	0.861	0.864	0.865	+0.3	+0.1
purchaseOrderNo	0.851	0.851	0.880	+0.0	+2.9

Table 6: One-shot ERP context vs. iterative AW, using Gemini 2.5 Pro. Deltas are F_1 percentage points.

Extended thinking shows marginal benefit. The average gain across paradigms is only +0.25 pp at $2.8\times$ cost and $3.3\times$ latency. Thinking slightly degrades the strongest paradigm (ReAct, -0.20 pp) but shows selective benefit for complex, free-text, and calculation-oriented fields such as *li.discount*. **Hard fields are free-text fields.** The largest residual errors occur in free-text fields (*paymentTerms*, $F_1 = 0.796$), where paradigm choice has limited impact. ERP-grounded fields show the largest gains: *purchaseOrderNo* (ReAct: +5.2 pp) and *li.MaterialNo* (AW: +3.9 pp), further supporting ERP grounding as the main source of agentic value.

5.3 Ablation Study

Agentic iteration adds value beyond one-shot ERP context. We compare AW with a non-agentic extractor that receives the same deterministically retrieved ERP record in one prompt but performs no verification loop. Table 6 separates the gain from static context (Ctx–BL) from the gain added by iteration (AW–Ctx). Context injection improves aggregate DN performance, especially on line items (Appendix D.2), but does not improve the PO field itself. The loop supplies the full PO gain: +15.6 pp on DN and +2.9 on INV. On aggregate INV F_1 , context injection and AW are nearly tied, consistent with the saturated regime.

5.4 Systematic Error and Failure Analysis

We categorise extraction errors into three types: *wrong-value*: a value is extracted but incorrect, *missing-value*: a value should be extracted but is left empty, and *hallucinations*: a value is extracted but should be empty. We also identify persistent performance ceilings across all paradigms. See Appendix G for the full analysis.

Wrong-value errors are highly structured. For DN, 93% of *purchaseOrderNo* errors are due to confusing buyer and supplier codes. AW resolves most via ERP lookup ($\Delta F_1 = +0.119$). For INV, *purchaseOrderNo* errors are dominated by label ambiguity, and *vendorName* by truncation. Addi-

tionally, confusion between *deliveryDate* and *documentDate* accounts for 80% of all *deliveryDate* errors, suggesting a prompt-based mitigation.

Missing-value errors are rare but field-specific. For DN, *documentDate* has a $\sim 17\%$ baseline miss rate due to field ambiguity. No paradigm consistently resolves this because the field is excluded from ERP data by business logic, preventing external grounding. Similarly, *li.purchaseOrderItemNo* reaches a 37.5% miss rate simply because it is nearly absent from the source documents.

Hallucinations are mainly caused by cross-field leakage. For PO- and material-related fields, hallucinations are rarely fabricated values, but header values leak into line-item fields. This pattern accounts for 83% of hallucinations for DN *li.purchaseOrderNo*, 54% for DN *li.customerMaterialNo*, and 63% for INV *li.purchaseOrderNo*. Stricter instructions targeting null-output discipline would likely improve these.

Performance ceilings arise from metric mismatch and field confusion. The fraction of error documents shared across all paradigms highlights universal challenges. For INV, *paymentTerms* (89% error overlap) and *li.description* (74%) underperform because exact match penalises semantically equivalent phrasings. Deterministic fuzzy rescoring raises average *li.description* F_1 from 0.692 to 0.830–0.940 without materially changing paradigm ordering (Appendix D.4). DN *documentDate* (54%) and *deliveryDate* (43%) errors instead reflect genuine field confusion.

6 Deployment Guidance for Industry

The following rules translate Section 5 findings into actionable deployment decisions. Appendix H provides extended analysis and guidelines.

Deploy agents selectively based on SoR-field density and baseline performance. Practitioners should first measure single-pass performance and determine what fraction of residual errors can be checked against the SoR. The strong invoice baseline gains less than one pp from AW or ReAct, whereas the weaker baseline gains 7.7–9.6 pp. If a single-pass extractor is already saturated or most errors concern fields absent from the SoR, agentic latency is unlikely to pay off.

Use AW as the default; ReAct with a strong backbone as a high-accuracy alternative. AW is the robust default because its fixed control flow achieves strong Pareto trade-offs across backbones.

ReAct can exceed AW with a strong reasoning model, but GPT-4o+ReAct skips the master-data fetch after 98 of 176 successful PO validations (Appendix D.5). Our findings suggest that ReAct should be deployed after held-out validation of tool sequencing; PlanExe and DAC require a task-specific benefit to justify their overhead.

Enable extended thinking selectively, not by default. On DN, thinking generally improves complex paradigms. On INV, the average gain is marginal at significantly higher cost. This suggests enabling thinking for PlanExe or DAC on structurally complex or high-ambiguity data, but defaulting to non-thinking for AW and ReAct unless the specific field requirements demand otherwise.

7 Conclusion

We present an agentic document IE pipeline with enterprise database grounding, and systematically evaluate four agentic paradigms on real enterprise documents. Across delivery notes and invoices, the value of agentic processing is conditional: gains are large when the single-pass baseline leaves room and errors concern SoR-resolvable fields, but small near saturation, and simpler paradigms match or outperform complex alternatives at substantially lower cost. Extended thinking selectively benefits complex paradigms but is not justified by default. Our error taxonomy identifies dominant failure modes and provides actionable guidance for deploying agentic document AI at enterprise scale.

Limitations

- **Synthetic ERP master data.** ERP records are generated from annotations to comply with privacy constraints. Masking and semantic rewriting stress-test shared-provenance shortcuts but cannot reproduce all production noise. Generalisation to live ERP backends requires further validation.
- **Dataset scope and small field support.** The study covers delivery notes and invoices from enterprise processes. Generalisation to other document types and supplier ecosystems remains untested. Several low-frequency fields also exhibit high variance and should be treated as indicative rather than definitive.
- **Model coverage and ceiling effect.** The invoice evaluation with GPT-4o and Gemini 2.5 Pro initial extractions shows that a ceiling effect suppresses the gains. Most evaluated backbones are

top-tier LLMs, which sets a high performance floor and may suppress observable differences between paradigms. Testing weaker models would widen performance variance and make paradigm comparisons more discriminative.

- **Exact-match metric.** Exact-match F_1 penalises semantically equivalent free-text variants. Fuzzy rescoring raises these fields substantially without materially changing the observed free-text paradigm ordering, but its threshold remains an evaluation choice.
- **Annotation agreement.** Ground truth uses annotate-then-review and consensus for difficult cases, but formal inter-annotator agreement was not measured.
- **Tool bank scope and portability.** The evaluation focuses on ERP grounding as the primary external tool, leaving real-world challenges, *e.g.*, handwriting recognition, multi-page table splits, and cross-document reconciliation unaddressed. Extending the tool bank with VLM-based perception and domain-customised skills is a clear direction for future work. Additionally, the execution tools are designed for a specific ERP system; adapting them to a different backend requires non-trivial schema mapping and API integration, though the evaluation methodology itself is reusable.

References

- Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R. Manmatha. 2021. [Docformer: End-to-end transformer for document understanding](#). In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 973–983. IEEE.
- Cheng Cui, Ting Sun, Manhui Lin, Tingquan Gao, Yubo Zhang, Jiaxuan Liu, Xueqing Wang, Zelun Zhang, Changda Zhou, Hongen Liu, Yue Zhang, Wenyu Lv, Kui Huang, Yichao Zhang, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. 2025. [Paddleocr 3.0 technical report](#). *CoRR*, abs/2507.05595.
- Siwei Han, Peng Xia, Ruiyi Zhang, Tong Sun, Yun Li, Hongtu Zhu, and Huaxiu Yao. 2025. [Mdocagent: A multi-modal multi-agent framework for document understanding](#). *CoRR*, abs/2503.13964.
- Anwen Hu, Haiyang Xu, Liang Zhang, Jiabo Ye, Ming Yan, Ji Zhang, Qin Jin, Fei Huang, and Jingren Zhou. 2025. [mplug-docowl2: High-resolution compressing for ocr-free multi-page document understanding](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2025, Vienna, Austria, July 27 – August 1, 2025, pages 5817–5834. Association for Computational Linguistics.
- Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. 2019. [FUNSD: A dataset for form understanding in noisy scanned documents](#). In *2nd International Workshop on Open Services and Tools for Document Analysis, OST@ICDAR 2019, Sydney, Australia, September 22-25, 2019*, pages 1–6. IEEE.
- Rachneet Kaur, Nishan Srishankar, Zhen Zeng, Sumitra Ganesh, and Manuela Veloso. 2025. [Chartagent: A multimodal agent for visually grounded reasoning in complex chart question answering](#). *CoRR*, abs/2510.04514.
- Haofu Liao, Aruni RoyChowdhury, Weijian Li, Ankan Bansal, Yuting Zhang, Zhuowen Tu, Ravi Kumar Satzoda, R. Manmatha, and Vijay Mahadevan. 2023. [Doctr: Document transformer for structured information extraction in documents](#). In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 19527–19537. IEEE.
- Wenhui Liao, Jiapeng Wang, Hongliang Li, Chengyu Wang, Jun Huang, and Lianwen Jin. 2024. [Doclaylm: An efficient and effective multi-modal extension of large language models for text-rich document understanding](#). *CoRR*, abs/2408.15045.
- Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. 2021. [Docvqa: A dataset for VQA on document images](#). In *IEEE Winter Conference on Applications of Computer Vision, WACV 2021, Waikoloa, HI, USA, January 3-8, 2021*, pages 2199–2208. IEEE.
- Mor Shpigel Nacson, Aviad Aberdam, Roy Ganz, Elad Ben-Avraham, Alona Golts, Yair Kittenplon, Shai Mazor, and Ron Litman. 2025. [Docvlm: Make your VLM an efficient reader](#). In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 29005–29015. Computer Vision Foundation / IEEE.
- Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, Jin Shi, Fan Wu, Pei Chu, Minghao Liu, Zhenxiang Li, Chao Xu, Bo Zhang, Botian Shi, Zhongying Tu, and Conghui He. 2025. [Omnidocbench: Benchmarking diverse PDF document parsing with comprehensive annotations](#). In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 24838–24848. Computer Vision Foundation / IEEE.
- Vincent Perot, Kai Kang, Florian Luisier, Guolong Su, Xiaoyu Sun, Ramya Sree Boppana, Zilong Wang, Zifeng Wang, Jiaqi Mu, Hao Zhang, Chen-Yu Lee, and Nan Hua. 2024. [LMDX: language model-based document information extraction and localization](#). In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of

- Findings of ACL*, pages 15140–15168. Association for Computational Linguistics.
- Jake Poznanski, Luca Soldaini, and Kyle Lo. 2025. [olmocr 2: Unit test rewards for document OCR](#). *CoRR*, abs/2510.19817.
- Yunjia Qi, Hao Peng, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. 2024. [ADELIE: aligning large language models on information extraction](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 7371–7387. Association for Computational Linguistics.
- Christian Reisswig, Anoop R Katti, Marco Spinaci, and Johannes Höhne. 2019. Chargrid-ocr: End-to-end trainable optical character recognition for printed documents using instance segmentation. *arXiv preprint arXiv:1909.04469*.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. [Toolformer: Language models can teach themselves to use tools](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Jiyuan Shen, Yuan Peiyue, Atin Ghosh, Yifan Mai, and Daniel Dahlmeier. 2026. [OCR or not? rethinking document information extraction in the mllms era with real-world large-scale datasets](#). In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2026 - Volume 5: Industry Track, Rabat, Morocco, March 25-27, 2026*, pages 385–396. Association for Computational Linguistics.
- Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. [Hugging-gpt: Solving AI tasks with chatgpt and its friends in hugging face](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. [Reflexion: language agents with verbal reinforcement learning](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Kimi Team. 2025. [Kimi K2: open agentic intelligence](#). *CoRR*, abs/2507.20534.
- Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, Bo Zhang, Liqun Wei, Zhihao Sui, Wei Li, Botian Shi, Yu Qiao, Dahua Lin, and Conghui He. 2024a. [Mineru: An open-source solution for precise document content extraction](#). *CoRR*, abs/2409.18839.
- Dongsheng Wang, Natraj Raman, Mathieu Sibue, Zhiqiang Ma, Petr Babkin, Simerjot Kaur, Yulong Pei, Armineh Nourbakhsh, and Xiaomo Liu. 2024b. [Docllm: A layout-aware generative language model for multimodal document understanding](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 8529–8548. Association for Computational Linguistics.
- Dongsheng Wang, Ran Zmigrod, Mathieu Sibue, Yulong Pei, Petr Babkin, Ivan Brugere, Xiaomo Liu, Nacho Navarro, Antony Papadimitriou, William Watson, Zhiqiang Ma, Armineh Nourbakhsh, and Sameena Shah. 2025a. [Buddie: A business document dataset for multi-task information extraction](#). In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025 - Workshops, Abu Dhabi, UAE, January 19-24, 2025*, pages 35–47. Association for Computational Linguistics.
- Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. [Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 2609–2634. Association for Computational Linguistics.
- Weishi Wang, Hengchang Hu, Zhijie Zhang, Zhaochen Li, Hongxin Shao, and Daniel Dahlmeier. 2025b. [Document intelligence in the era of large language models: A survey](#). *CoRR*, abs/2510.13366.
- Wenjin Wang, Yunhao Li, Yixin Ou, and Yin Zhang. 2023b. [Layout and task aware instruction prompt for zero-shot document image question answering](#). *CoRR*, abs/2306.00526.
- Haoran Wei, Yaofeng Sun, and Yukun Li. 2025. [Deepseek-ocr: Contexts optical compression](#). *CoRR*, abs/2510.18234.
- Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. [LayoutLM: Pre-training of text and layout for document image understanding](#). In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 1192–1200. ACM.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. [React: Synergizing reasoning and acting in language models](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- Xinlei Yu, Zhangquan Chen, Yudong Zhang, Shilin Lu, Ruolin Shen, Jiangning Zhang, Xiaobin Hu, Yanwei Fu, and Shuicheng Yan. 2025. [Visual document understanding and question answering: A multi-](#)

agent collaboration framework with test-time scaling. *CoRR*, abs/2508.03404.

Hanzhang Zhou, Junlang Qian, Zijian Feng, Hui Lu, Zixiao Zhu, and Kezhi Mao. 2024. [Llms learn task heuristics from demonstrations: A heuristic-driven prompting strategy for document-level event argument extraction](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 11972–11990. Association for Computational Linguistics.

Fengbin Zhu, Ziyang Liu, Xiang Yao Ng, Haohui Wu, Wenjie Wang, Fuli Feng, Chao Wang, Huanbo Luan, and Tat-Seng Chua. 2024. [Mmdocbench: Benchmarking large vision-language models for fine-grained visual document understanding](#). *CoRR*, abs/2410.21311.

Description Rewrite

System: You are a procurement data specialist helping to simulate how a buyer's ERP system stores descriptions on a Purchase Order (PO).

Context:

The input text is a product description taken from a vendor <DOC TYPE>. Your task is to rewrite it as it would appear in the buyer's own PO or ERP data, from the buyer's perspective rather than the vendor's.

Rules:

1. Keep all technical specifications that the buyer would know at PO creation time: dimensions, material grade, standard part numbers, weights, tolerances.
2. Remove or neutralise vendor-internal references that the buyer would not know before delivery: vendor batch numbers, vendor serial numbers, vendor-internal production codes.
3. Use buyer-side language: procurement category terminology, buyer's own shorthand, ISO/DIN/industry-standard names rather than vendor trade names where appropriate.
4. You may rephrase, reorder, abbreviate, or expand, but do not invent specifications.
5. Preserve all numbers that represent measurable properties (size, quantity, weight, voltage, etc.).
6. Return ONLY the rewritten description, without any explanations or additional text.

User: Rewrite this vendor invoice description as it would appear in the buyer's PO/ERP: <TEXT>

Unit-of-Measure Rewrite

System: You are a procurement data specialist. Your task is to provide an alternative notation for a Unit of Measure (UoM) as it might appear in a buyer's ERP system or Purchase Order, introducing realistic variance across different notation conventions.

The same physical unit can appear in many forms depending on the system or country:

1. Has the same or similar meaning as the original
2. Could be an ISO code, ERP standard code, or commonly used abbreviation
3. Is valid for business/ERP contexts
4. Examples of equivalent representations:
 - "PC" ↔ "EA" ↔ "Each" ↔ "piece" ↔ "Pieces" (all mean the same)
 - "pcs" ↔ "pcs." ↔ "Pcs" ↔ "PCS" (variations of pieces)
 - "M" ↔ "m" ↔ "MTR" (meter variations)
 - "Stück" ↔ "ST" ↔ "piece" (German to English)
5. Return ONLY the alternative UoM code/abbreviation, without any explanations or additional text.

User: Provide one alternative notation for this unit of measure: <TEXT>

Payment-terms Rewrite (Invoice Only)

System: You are an accounts-payable specialist. Your task is to rewrite a payment terms string as it might be stored in a buyer's ERP system, simulating realistic notation variance.

Context:

Payment terms are negotiated on the Purchase Order and stored in the buyer's ERP. The same terms can be represented in several ways across different companies and systems:

- Payment key codes: e.g. NT30, Z001, Z030, 0001
- Structured short form: e.g. "2/10 Net 30", "1/15 Net 45"
- Human-readable EN: e.g. "Net 30 days", "Payable within 30 days", "Due in 30 days"
- Human-readable DE: e.g. "Zahlbar innerhalb 30 Tagen", "Netto 30 Tage"
- With early-payment discount: e.g. "2% 10 days, net 30", "2% Skonto bei Zahlung binnen 10 Tagen, netto 30 Tage"

Rules:

1. Preserve all financial values exactly: discount percentages, discount windows (days), net due dates.
2. Do not add discount terms if none exist in the original, and do not remove them if they do.
3. Switch the notation style (e.g. short code ↔ human-readable ↔ structured form).
4. The rewritten terms must be unambiguous and valid in a business context.
5. Return ONLY the rewritten payment terms string, without any explanations.

User: Rewrite these payment terms in an alternative notation: <TEXT>

Effects on Evaluation: What Augmentation Reveals? Augmented and unaugmented documents achieve similar aggregate F_1 across all paradigms, confirming that OCR text is the primary extraction signal and ERP reference data is a secondary grounding source, not a necessary one.

Table 8 reports the per-field F_1 delta (masked vs. unmasked) averaged across paradigms and models. On invoices, all Δ are within ± 0.01 except *paymentTerms* (-0.008). On delivery notes, *li.customerMaterialNo* shows the largest sensitivity (-0.022 on average across paradigms), consistent with the reliance of AW on ERP cross-validation. Positive Δ values for several fields, e.g., *li.unitOfMeasure* $+0.026$, are selection-effect artefacts: masked documents constitute a non-random subset that happens to contain simpler line items on average, rather than representing genuine accuracy gains from masking.

Rewritten ERP references have near-zero impact on extraction accuracy ($|\Delta F_1| \leq 0.02$ in all paradigms), indicating that extraction paradigms match field values against document-side OCR rather than by literal string comparison against the ERP reference. The largest positive Δ values again reflect selection effects.

Substituting equivalent unit-of-measure notations, e.g., PC and EA, has negligible impact on F_1 for all paradigms. Models successfully extract the document-side unit of measure regardless of what the ERP reference states.

Field	Doc	ΔF_1	Interpretation
<i>li.customerMaterialNo</i>	DN	-0.022	ERP cross-check absent
<i>li.supplierMaterialNo</i>	DN	$+0.003$	Negligible
<i>li.itemNo</i>	DN	$+0.007$	Selection effect
<i>li.quantity</i>	DN	$+0.001$	Negligible
<i>li.unitOfMeasure</i>	DN	$+0.026$	Selection effect
<i>paymentTerms</i>	INV	-0.008	Small ERP-hint dependency
<i>subtotalAmount</i>	INV	$+0.001$	Negligible
<i>li.description</i>	INV	$+0.100$	Selection effect
<i>li.materialNo</i>	INV	$+0.034$	Selection effect
<i>li.unitOfMeasure</i>	INV	$+0.011$	Selection effect

Table 8: F_1 delta from ERP field masking (masked vs. unmasked, avg over paradigms and models).

C Implementation Details

Iteration budgets. All paradigms use greedy decoding (temperature = 0.0). Paradigm-specific iteration budgets are:

- AW: fixed 5-step pipeline with up to 3 retries per step.

Paradigm	Avg. F_1	95% CI	ΔF_1	95% CI (ΔF_1)	Sig.
<i>Delivery Note</i>					
AW	0.8638	[0.8426, 0.8829]	+0.0441	[+0.0324, +0.0571]	Yes
ReAct	0.8585	[0.8379, 0.8771]	+0.0388	[+0.0283, +0.0502]	Yes
PlanExe	0.8434	[0.8222, 0.8630]	+0.0238	[+0.0171, +0.0316]	Yes
DAC	0.8488	[0.8273, 0.8688]	+0.0291	[+0.0219, +0.0390]	Yes
<i>Invoice</i>					
AW	0.8668	[0.8592, 0.8742]	+0.0059	[+0.0032, +0.0087]	Yes
ReAct	0.8694	[0.8588, 0.8788]	+0.0085	[+0.0013, +0.0140]	Yes
PlanExe	0.8610	[0.8504, 0.8701]	+0.0002	[−0.0066, +0.0052]	No
DAC	0.8611	[0.8535, 0.8689]	+0.0002	[−0.0016, +0.0020]	No

Table 9: Paired bootstrap confidence interval (CI) results. A gain is significant when its 95% interval excludes zero.

- ReAct: up to 20 thought-action-observation iterations.
- PlanExe: up to 15 outer iterations, 3 replans, 2 step-retries, and 5 tool rounds per step.
- DAC: up to 25 agent turns (total log messages are $2\text{--}3\times$ higher as tool calls are counted separately).

Extended thinking configuration. For Gemini 2.5 Pro, the default non-thinking setting uses a minimal thinking budget of 128 tokens, while thinking mode raises it to 8000 tokens. For GPT-5, the corresponding settings are minimal and medium. Gemini 2.5 Flash and GPT-4o have thinking disabled entirely.

Empirical termination. Iteration budgets are upper bounds rather than matched compute budgets. On DN, ReAct completed all 195 documents while averaging 4.6 messages and 2.1 tool calls, well below its 20-iteration cap. DAC completed 194 of 195 documents and hit its message limit once. The larger cost and latency therefore reflect paradigm structure on DN. This does not contradict the INV failure mode in Appendix H: without thinking, DAC exhausts its message budget more often on longer invoice traces.

D Statistical and Ablation Analyses

D.1 Paired Bootstrap Confidence Intervals

We use $B = 10,000$ paired document-level bootstrap samples. For paradigm averages, each draw shares document indices across the K agentic LLM configurations and compares them with the same Gemini 2.5 Pro baseline predictions. Table 9 reports the DN and INV experiments with Gemini 2.5 Pro baseline predictions. These intervals measure sensitivity to the evaluated document sample. Every model configuration was run once, so they do not estimate stochastic variation across repeated inference runs.

Scope	BL	CTX	AW	CTX−BL	AW−CTX
<i>Delivery Note</i>					
All fields	0.8197	0.8372	0.8593	+1.8	+2.2
Headers	0.7943	0.7943	0.8366	+0.0	+4.2
Line items	0.8356	0.8641	0.8736	+2.9	+0.9
purchaseOrderNo	0.8179	0.8179	0.9740	+0.0	+15.6
<i>Invoice</i>					
All fields	0.8608	0.8639	0.8653	+0.3	+0.1
Headers	0.8489	0.8489	0.8505	+0.0	+0.2
Line items	0.8766	0.8839	0.8848	+0.7	+0.1
purchaseOrderNo	0.8509	0.8509	0.8797	+0.0	+2.9

Table 10: Full context-injection ablation with Gemini 2.5 Pro. Deltas are F_1 percentage points.

Paradigm	F_1	ΔBL	95% CI	Significant?
<i>GPT-4o Agentic Backbone</i>				
AW	0.8693	+9.2	[+8.4, +10.1]	Yes
ReAct	0.8536	+7.7	[+6.8, +8.6]	Yes
PlanExe	0.8586	+8.2	[+7.3, +9.0]	Yes
DAC	0.8594	+8.2	[+7.4, +9.1]	Yes
<i>Gemini 2.5 Pro Agentic Backbone</i>				
AW	0.8663	+8.9	[+8.1, +9.8]	Yes
ReAct	0.8734	+9.6	[+8.8, +10.5]	Yes
PlanExe	0.8624	+8.5	[+7.7, +9.4]	Yes
DAC	0.8616	+8.5	[+7.6, +9.4]	Yes

Table 11: Full low-baseline invoice stress test. CIs are paired 95% intervals for ΔF_1 in percentage points.

D.2 One-Shot Context Injection

The context baseline receives the same deterministically retrieved ERP record as AW in a single extraction prompt, without multi-turn verification or refinement. Table 10 decomposes Context−BL as the contribution of static ERP context and AW−Context as the contribution of the loop.

D.3 Invoice Baseline-Headroom Stress Test

Table 11 uses GPT-4o as the initial invoice extractor ($F_1 = 0.7771$), then varies the LLM controlling each agentic paradigm. All intervals exclude zero. This is a separate regime from Table 9, whose invoice baseline is Gemini 2.5 Pro ($F_1 = 0.8608$).

D.4 Fuzzy Evaluation of Free-Text Fields

Free-text fields occur only on INV, where exact-match evaluation penalises predictions that are semantically correct but differ in superficial ways. We rescore the nine invoice configurations using the initial Gemini 2.5 Pro extractions, without additional inference. An exact-match error is counted as correct when the string similarity is at least 0.85. We measure similarity using two complementary metrics: the normalised Levenshtein ratio, which operates at the character level and is sensitive to character order, and the token-set ratio, which compares sets of words regardless of their order. The lenient variant takes the maximum of the two scores, counting a prediction as correct if either metric

exceeds the threshold, whereas the balanced variant takes their mean, requiring both metrics to be reasonably high. Reporting both variants provides results under permissive and stricter interpretations of semantic similarity, ensuring that our conclusions do not depend on a single similarity metric or threshold. Table 12 reports all three scores. Every field improves, especially *li.description*, while the substantive ordering is stable: ReAct remains strongest and DAC remains weakest across the free-text fields; AW and PlanExe differ by less than one pp.

D.5 GPT-4o+ReAct Failure Analysis

We inspect all 195 DN traces for GPT-4o+ReAct, whose F_1 falls below the non-agentic baseline. All traces terminate with zero malformed tool calls, and PO extraction improves when the grounding chain executes. The failure is therefore not prompt-format sensitivity. Instead, GPT-4o validates a PO but omits the subsequent master-data fetch on 98 of 176 successful validations, preferring self-reasoning over tool use by 3.2:1. Table 13 lists the detailed events.

Three downstream symptoms explain the regression: tool-sequencing skips, 152 header-to-line-item PO leakage rows vs. 11 for Gemini 2.5 Pro+Think under the same prompt, and 34 absent *deliveryDate* fields filled through over-reasoning. This model-paradigm interaction motivates validating ReAct’s tool discipline on a held-out set and using AW when the reasoning backbone is unverified.

E Quantitative Results

Non-Agentic Baselines. Table 14 reports the overall and per-field F_1 for six non-agentic single-pass extraction baselines on the delivery notes.

Full Quantitative Results. Tables 15 and 16 report F_1 , latency, cost per document, and token consumption for all configurations on DN and INV data. Tables 17 and 18 report overall and per-field F_1 scores, latency, cost per document, and token consumption for all configurations on DN and INV. The *Baseline* row is from the non-agentic Gemini 2.5 Pro extraction.

F Qualitative Results

Table 19 provides detailed single-example cases with ERP context for the three main win patterns and one representative failure.

G Error Sub-Pattern Details

Table 20 summarises the structured sub-patterns identified within each failure type, supporting the analysis in Section 5.4.

H Deployment Considerations

This appendix extends Section 6 with additional deployment guidelines and technical details.

Pre-screen the backbone before deploying ReAct. Section 6 recommends AW as the safe default when backbone capability is unverified. This guideline provides the operational detail behind that rule. ReAct’s performance collapses below the non-agentic baseline with weaker backbones: on DN it achieves $F_1 = 0.871$ with Gemini 2.5 Pro but falls to 0.794 with GPT-4o, the worst configuration in the study. We attribute this performance gap to two primary factors: a general model-quality effect, as GPT-4o often underperforms on specialised extraction tasks, and a specific failure of multi-step chain-of-thought within the ReAct reasoning loop, as evidenced by our trace analysis. AW does not exhibit this behaviour; its F_1 range is only 1.8 pp across six backbones. The decision criterion: measure task-complete rate and mean intermediate step count on a held-out sample before committing to ReAct in production. If either metric falls noticeably below Gemini 2.5 Flash levels, fall back to AW.

Audit ERP material-number coverage before deploying AW. AW’s validation loop cross-checks extracted material codes against ERP master data, which is the primary source of its gains on material-number fields (+12.7 pp on *li.customerMaterialNo*, +8.2 pp on *li.supplierMaterialNo*). The same loop can actively suppress a correct OCR extraction when the corresponding ERP entry is missing, causing systematic false negatives. Under field-masking experiments, AW degrades by −6.9 pp and −3.7 pp on those fields respectively, while ReAct and DAC remain flat ($|\Delta F_1| \leq 0.01$). The decision criterion: before deploying AW on a new supplier template, estimate ERP material-number coverage for that supplier set. If coverage is substantially incomplete, route material-number fields to ReAct instead as a more robust fallback. AW remains the preferred paradigm for PO-reference fields, where ERP coverage is typically complete.

Do not deploy DAC without extended thinking on complex documents. Section 6 recommends enabling thinking for DAC on struc-

Field	BL	AW	ReAct	PlanExe	DAC	Average
li.description	0.696 / 0.933 / 0.811	0.710 / 0.939 / 0.839	0.699 / 0.947 / 0.841	0.671 / 0.945 / 0.831	0.686 / 0.932 / 0.817	0.692 / 0.940 / 0.830
buyerAddress	0.807 / 0.979 / 0.811	0.807 / 0.979 / 0.811	0.816 / 0.982 / 0.820	0.797 / 0.976 / 0.800	0.787 / 0.959 / 0.792	0.802 / 0.974 / 0.806
shipToAddress	0.758 / 0.897 / 0.762	0.758 / 0.897 / 0.762	0.770 / 0.902 / 0.772	0.724 / 0.886 / 0.726	0.756 / 0.893 / 0.760	0.753 / 0.895 / 0.756
vendorAddress	0.798 / 0.966 / 0.814	0.798 / 0.966 / 0.814	0.808 / 0.967 / 0.824	0.797 / 0.967 / 0.811	0.785 / 0.953 / 0.804	0.797 / 0.964 / 0.813
paymentTerms	0.796 / 0.863 / 0.809	0.796 / 0.863 / 0.809	0.792 / 0.861 / 0.806	0.786 / 0.860 / 0.802	0.796 / 0.862 / 0.808	0.793 / 0.861 / 0.806

Table 12: Fuzzy re-scoring at threshold 0.85. Each cell is exact/lenient/balanced F_1 .

Event	Count	Documents
PO validation called	194	99.5%
PO accepted by ERP	176	90.3%
Master data fetched	79	40.5%
Accepted PO, fetch skipped	98	50.3%
Self-reasoning actions	899	—
Tool-call actions	284	—

Table 13: Tool behaviour in 195 GPT-4o+ReAct DN traces.

Field	G-Flash	G-Pro	G-Pro+T	GPT-4o	GPT-5	GPT-5+T	Support
All Fields	0.802	0.820	0.813	0.704	0.710	0.712	1661
All Headers	0.774	0.794	0.810	0.687	0.704	0.700	641
All Line Items	0.819	0.836	0.815	0.714	0.714	0.720	1020
deliveryDate	0.533	0.554	0.617	0.513	0.524	0.533	57
deliveryNoteNo	0.942	0.922	0.924	0.896	0.906	0.916	168
documentDate	0.807	0.822	0.821	0.633	0.697	0.693	159
purchaseOrderNo	0.733	0.818	0.822	0.568	0.578	0.532	190
supplierId	0.734	0.710	0.729	0.822	0.817	0.853	67
li.customerMaterialNo	0.746	0.798	0.776	0.590	0.598	0.600	216
li.itemNo	0.800	0.874	0.874	0.858	0.850	0.843	126
li.purchaseOrderItemNo	0.667	0.769	0.769	0.750	0.750	0.667	8
li.purchaseOrderNo	0.667	0.481	0.426	0.214	0.177	0.185	27
li.quantity	0.952	0.941	0.941	0.927	0.935	0.937	277
li.supplierMaterialNo	0.754	0.773	0.741	0.681	0.701	0.707	236
li.unitOfMeasure	0.860	0.905	0.880	0.807	0.771	0.792	130

Table 14: Overall and per-field F_1 for non-agentic single-pass baselines.

turally complex documents. This guideline provides the operational evidence. Without extended thinking, DAC exhausts the agent message budget (max_messages_reached) in 61.6% of INV runs (Gemini 2.5 Pro), with budget-exhausted runs achieving $F_1 = 0.857$ vs. 0.868 for task-complete runs ($\Delta = -1.1$ pp). The failure is architectural: DAC’s five-agent orchestration loop generates substantially more inter-agent messages per document than simpler paradigms, and invoices with many line items exhaust the fixed budget before extraction is complete, causing silent partial extractions. Adding extended thinking resolves this: DAC+Think reaches task_complete on 99.4% of runs at only $1.3\times$ additional cost on INV. The decision criterion: treat extended thinking as mandatory, not optional, for DAC on invoice-class documents with non-trivial structure; monitor task_complete rate as a production health metric, as a low rate signals budget exhaustion rather than model failure.

Validate per-field paradigm leads on a small sample before committing to field routing. The routing recommendations in Section 6

rest on consistent per-field paradigm leads that are reproducible across models: AW leads on material-number fields (+1.4 pp margin on *li.customerMaterialNo*); ReAct leads on date fields (*deliveryDate*, *documentDate*) where AW has no ERP reference to anchor against; PlanExe leads on *li.Discount* (+2.8 pp over ReAct) due to calculation reasoning; ReAct leads on *vendorName* (+1.2 pp over AW) due to legal-suffix disambiguation. However, these leads can reverse on structurally different templates: on INV, AW shows -7.0 pp on *li.purchaseOrderNo* (vs. ReAct +11.7 pp) because cross-field leakage from invoice-header PO numbers is amplified, not suppressed, by AW’s validation loop on ambiguous multi-PO layouts. The decision criterion: when onboarding a new document template, measure per-field F_1 across paradigms on a small validation sample before committing to the routing table; do not assume that material-number fields always favour AW on templates with ambiguous PO structures.

Model	All Fields↑	Header Fields↑	Line-item Fields↑	Latency (s)↓	Cost/Doc (\$)↓	Tokens (in)↓	Tokens (out)↓	Tokens (think)↓
<i>Non-Agentic Baseline</i>								
Gemini 2.5 Pro	0.820	0.794	0.836	–	–	–	–	–
<i>AW</i>								
Gemini 2.5 Flash	0.866	0.837	0.885	3.88	0.0020	3,918	317	–
Gemini 2.5 Pro	0.859	0.837	0.874	5.72	0.0063	3,124	110	169
Gemini 2.5 Pro+Think	0.876	0.837	0.901	23.31	0.0284	3,525	227	2,156
GPT-4o	0.861	0.837	0.876	3.46	0.0116	3,661	208	–
GPT-5	0.862	0.837	0.878	7.68	0.0086	3,980	357	–
GPT-5+Think	0.858	0.837	0.871	33.49	0.0224	2,074	221	1,728
<i>ReAct</i>								
Gemini 2.5 Flash	0.869	0.831	0.893	12.97	0.0121	21,262	2,212	–
Gemini 2.5 Pro	0.871	0.826	0.899	25.96	0.0477	22,098	1,655	366
Gemini 2.5 Pro+Think	0.878	0.832	0.907	45.71	0.0731	20,658	1,634	2,933
GPT-4o	0.794	0.812	0.782	11.54	0.0761	25,589	1,161	–
GPT-5	0.868	0.828	0.894	14.65	0.0203	10,366	740	–
GPT-5+Think	0.871	0.819	0.904	213.73	0.1828	27,255	1,529	13,120
<i>PlanExe</i>								
Gemini 2.5 Flash	0.825	0.801	0.841	13.46	0.0131	32,063	1,235	–
Gemini 2.5 Pro	0.854	0.819	0.877	33.13	0.0607	33,326	1,083	800
Gemini 2.5 Pro+Think	0.857	0.811	0.886	116.22	0.1705	40,137	2,287	9,797
GPT-4o	0.819	0.791	0.837	12.42	0.0721	24,342	1,126	–
GPT-5	0.842	0.798	0.870	63.41	0.1165	61,864	3,844	–
GPT-5+Think	0.863	0.827	0.886	340.81	0.3003	54,791	2,887	20,352
<i>DAC</i>								
Gemini 2.5 Flash	0.840	0.820	0.853	15.46	0.0106	28,526	795	–
Gemini 2.5 Pro	0.828	0.801	0.845	56.04	0.0586	44,467	80	192
Gemini 2.5 Pro+Think	0.870	0.826	0.898	84.95	0.0850	31,221	831	3,592
GPT-4o	0.830	0.825	0.833	11.79	0.0575	20,530	652	–
GPT-5	0.856	0.815	0.883	30.88	0.0593	35,473	1,367	–
GPT-5+Think	0.868	0.831	0.892	242.57	0.1666	40,317	1,452	9,856

Table 15: Full results on delivery notes.

Model	All Fields↑	Header Fields↑	Line-item Fields↑	Latency (s)↓	Cost/Doc (\$)↓	Tokens (in)↓	Tokens (out)↓	Tokens (think)↓
<i>Non-Agentic Baseline</i>								
Gemini 2.5 Pro	0.861	0.849	0.877	–	–	–	–	–
<i>AW</i>								
Gemini 2.5 Pro	0.865	0.851	0.885	4.19	0.0051	2,694	66	90
Gemini 2.5 Pro+Think	0.868	0.851	0.891	27.46	0.0301	3,754	180	2,296
<i>ReAct</i>								
Gemini 2.5 Pro	0.870	0.858	0.886	40.48	0.0794	36,838	3,072	383
Gemini 2.5 Pro+Think	0.868	0.862	0.877	68.95	0.1120	32,256	2,478	4,430
<i>PlanExe</i>								
Gemini 2.5 Pro	0.856	0.849	0.865	51.56	0.1046	55,934	2,546	926
Gemini 2.5 Pro+Think	0.866	0.856	0.880	170.69	0.2511	59,846	3,254	14,506
<i>DAC</i>								
Gemini 2.5 Pro	0.862	0.850	0.877	59.84	0.0895	68,613	289	246
Gemini 2.5 Pro+Think	0.861	0.847	0.878	92.89	0.1148	46,658	1,421	4,082

Table 16: Full results on invoices.

Field	G-Pro	G-Flash				G-Pro				G-Pro+T				GPT-4o				GPT-5				GPT-5+T			
	Baseline	AW	PlanExe	ReAct	DAC	AW	PlanExe	ReAct	DAC	AW	PlanExe	ReAct	DAC	AW	PlanExe	ReAct	DAC	AW	PlanExe	ReAct	DAC	AW	PlanExe	ReAct	DAC
deliveryDate†	0.554	0.554	0.548	0.556	0.554	0.554	0.524	0.550	0.554	0.554	0.512	0.570	0.562	0.554	0.498	0.485	0.557	0.554	0.613	0.605	0.557	0.554	0.600	0.486	0.560
deliveryNoteNo†	0.922	0.922	0.922	0.922	0.922	0.922	0.914	0.919	0.922	0.922	0.914	0.919	0.922	0.922	0.922	0.922	0.916	0.922	0.922	0.922	0.922	0.922	0.937	0.927	0.922
documentDate†	0.822	0.822	0.787	0.801	0.824	0.822	0.785	0.784	0.829	0.822	0.835	0.784	0.808	0.822	0.816	0.817	0.814	0.822	0.787	0.793	0.822	0.822	0.723	0.676	0.822
purchaseOrderNo	0.818	0.974	0.881	0.966	0.913	0.974	0.974	0.974	0.839	0.974	0.935	0.974	0.945	0.974	0.864	0.955	0.950	0.974	0.813	0.913	0.892	0.974	0.935	0.976	0.948
supplierId	0.710	0.710	0.698	0.720	0.703	0.710	0.698	0.709	0.706	0.710	0.673	0.720	0.702	0.710	0.698	0.702	0.692	0.710	0.717	0.721	0.710	0.710	0.737	0.754	0.710
li.customerMaterialNo	0.798	0.949	0.804	0.927	0.825	0.938	0.861	0.940	0.819	0.964	0.876	0.949	0.942	0.911	0.794	0.803	0.827	0.911	0.834	0.922	0.885	0.878	0.878	0.928	0.914
li.itemNo	0.874	0.876	0.874	0.886	0.886	0.879	0.871	0.879	0.864	0.878	0.886	0.873	0.886	0.866	0.874	0.822	0.846	0.866	0.871	0.864	0.883	0.876	0.854	0.879	0.873
li.purchaseOrderItemNo	0.769	0.625	0.714	0.571	0.769	0.706	0.667	0.640	0.769	0.538	0.778	0.609	0.800	0.706	0.769	0.444	0.714	0.588	0.769	0.667	0.769	0.769	0.769	0.667	0.769
li.purchaseOrderNo	0.481	0.448	0.495	0.619	0.553	0.419	0.722	0.693	0.495	0.591	0.788	0.812	0.581	0.525	0.486	0.254	0.515	0.571	0.842	0.754	0.742	0.525	0.897	0.963	0.735
li.quantity	0.941	0.944	0.941	0.952	0.943	0.938	0.950	0.948	0.948	0.953	0.950	0.952	0.952	0.919	0.941	0.931	0.934	0.926	0.948	0.950	0.945	0.953	0.946	0.946	0.950
li.supplierMaterialNo	0.773	0.865	0.792	0.857	0.793	0.841	0.827	0.863	0.785	0.872	0.832	0.859	0.861	0.870	0.787	0.776	0.781	0.858	0.802	0.847	0.819	0.821	0.834	0.851	0.842
li.unitOfMeasure	0.905	0.905	0.895	0.899	0.901	0.908	0.904	0.895	0.905	0.902	0.899	0.912	0.899	0.889	0.895	0.836	0.835	0.899	0.905	0.892	0.902	0.908	0.902	0.899	0.895

Table 17: Field-level results across agentic paradigms and models on delivery notes. †: fields are not presented in ERP master data.

Field	G-Pro	G-Pro				G-Pro+T			
	Baseline	AW	PlanExe	ReAct	DAC	AW	PlanExe	ReAct	DAC
buyerAddress	0.806	0.806	0.785	0.814	0.806	0.806	0.808	0.818	0.768
buyerName	0.918	0.918	0.906	0.920	0.920	0.918	0.918	0.918	0.918
country [†]	0.968	0.968	0.973	0.975	0.968	0.968	0.981	0.982	0.968
currency [†]	0.988	0.988	0.996	0.994	0.990	0.988	0.995	0.992	0.990
deliveryDate [†]	0.806	0.806	0.777	0.789	0.806	0.806	0.775	0.794	0.806
deliveryNoteNo [†]	0.814	0.814	0.807	0.804	0.814	0.814	0.803	0.816	0.814
discount [†]	0.861	0.861	0.892	0.901	0.861	0.861	0.917	0.930	0.877
dueDate [†]	0.879	0.879	0.875	0.915	0.881	0.879	0.884	0.920	0.881
employeeName [†]	0.256	0.256	0.262	0.269	0.256	0.256	0.287	0.282	0.260
invoiceDate [†]	0.966	0.966	0.966	0.960	0.966	0.966	0.964	0.960	0.966
invoiceNo [†]	0.968	0.968	0.970	0.966	0.968	0.968	0.966	0.968	0.968
paymentTerms	0.796	0.796	0.787	0.790	0.796	0.796	0.785	0.793	0.795
purchaseOrderNo	0.851	0.880	0.888	0.899	0.863	0.882	0.878	0.906	0.878
shipToAddress	0.758	0.758	0.712	0.763	0.758	0.758	0.736	0.777	0.754
shippingAmount [†]	0.680	0.680	0.675	0.691	0.680	0.680	0.675	0.711	0.689
subtotalAmount	0.928	0.928	0.934	0.941	0.928	0.928	0.932	0.937	0.930
tax1Amount [†]	0.928	0.928	0.922	0.926	0.928	0.928	0.925	0.925	0.930
tax1Description [†]	0.225	0.225	0.220	0.236	0.225	0.225	0.389	0.216	0.228
tax1Rate [†]	0.869	0.869	0.846	0.851	0.869	0.869	0.842	0.853	0.873
tax2Amount [†]	0.167	0.167	0.263	0.167	0.167	0.167	0.256	0.216	0.167
tax2Description [†]	0.061	0.061	0.059	0.059	0.061	0.061	0.111	0.118	0.061
tax2Rate [†]	0.296	0.296	0.296	0.296	0.296	0.296	0.296	0.231	0.296
tax3Amount [†]	0.167	0.167	0.240	0.167	0.167	0.167	0.240	0.167	0.167
tax3Description [†]	0.100	0.100	0.095	0.100	0.100	0.100	0.095	0.100	0.100
tax3Rate [†]	0.154	0.154	0.154	0.154	0.154	0.154	0.154	0.154	0.154
totalAmount [†]	0.996	0.996	0.993	0.996	0.996	0.996	0.996	0.996	0.996
vendorAddress	0.798	0.798	0.784	0.801	0.800	0.798	0.810	0.815	0.771
vendorBankAccountNo [†]	0.914	0.914	0.918	0.941	0.914	0.914	0.929	0.943	0.914
vendorName	0.801	0.801	0.852	0.857	0.803	0.801	0.850	0.869	0.799
vendorTaxId [†]	0.900	0.900	0.905	0.910	0.898	0.900	0.903	0.912	0.900
li.BatchNo [†]	0.242	0.250	0.242	0.242	0.267	0.235	0.242	0.235	0.242
li.Description	0.696	0.712	0.672	0.691	0.690	0.708	0.669	0.706	0.682
li.Discount [†]	0.549	0.573	0.878	0.869	0.575	0.637	0.824	0.931	0.631
li.MaterialNo	0.810	0.860	0.841	0.835	0.810	0.837	0.810	0.833	0.834
li.PurchaseOrderNo	0.550	0.447	0.529	0.672	0.573	0.513	0.568	0.662	0.621
li.Quantity	0.969	0.958	0.935	0.965	0.964	0.966	0.969	0.940	0.964
li.SerialNo [†]	0.390	0.450	0.400	0.390	0.421	0.524	0.390	0.390	0.410
li.Tax1Amount [†]	0.912	0.929	0.926	0.917	0.918	0.952	0.913	0.920	0.906
li.Tax1Rate [†]	0.925	0.940	0.939	0.927	0.926	0.938	0.926	0.928	0.927
li.Tax2Amount [†]	0.800	0.833	0.667	0.800	0.800	1.000	0.800	0.800	0.800
li.Tax2Rate [†]	0.533	0.870	0.429	0.533	0.533	0.588	0.533	0.533	0.533
li.Tax3Amount [†]	0.000	0.480	0.222	0.316	0.105	0.966	0.105	0.105	0.000
li.Tax3Rate [†]	0.000	0.480	0.222	0.316	0.105	0.966	0.105	0.105	0.000
li.TotalAmount [†]	0.968	0.971	0.931	0.960	0.970	0.977	0.970	0.933	0.972
li.UnitOfMeasure	0.947	0.950	0.923	0.950	0.945	0.954	0.949	0.951	0.939
li.UnitPrice [†]	0.973	0.974	0.934	0.966	0.973	0.976	0.972	0.944	0.971

Table 18: Field-level results across agentic paradigms and models on invoices. †: fields are not presented in ERP master data.

Document (Cropped and Sanitised)	Field / Error Description	ERP Master Data	Extraction
Delivery note Service Number48271053 Date14.03.2025 Page1 / 1 Reference numberG72K9PL Date05.03.2025 Our contact partner with youHartmann Fax Order number57839201 Date06.03.2025 Customer no.483920 Your contact partner with usMs. Sandra Hoffmann Telephone+49 6221 748310 Fax+49 6221 748399 Emailsandra.hoffmann@globalind.de Supplier no.29403715	Document Type: Delivery Note Field: <i>purchaseOrderNo</i> The document contains both a supplier PO and a customer PO. Supplier PO is followed by <i>Order number</i>, which is a common key for <i>purchaseOrderNo</i>. Baseline copies the supplier PO, and all agentic methods cross-check against ERP master data and select the correct customer PO.	"G72K9PL": { "orderDate": ["06.03.2025"], "supplierId": ["29403715"], "Items": [{ . . . }, . . .], }	Baseline: 483920 AW: G72K9PL ReAct: G72K9PL PlanExe: G72K9PL DAC: G72K9PL
Customer Order No. Customer Account Article Name Ordermaster Date 8804574 8804574 15-03-2025 PRODUCT QUANTITY UNIT PRICE 8810470 STANDARD PROTECTIVE WORKWEAR ITEM 1000 0.00 7280412	Document Type: Delivery Note Field: <i>li.customerMaterialNo</i> The document contains two material numbers from customer and supplier under the same column <i>PRODUCT</i>. Without ERP data, it is challenging to distinguish the two.	"609983721": { "supplierId": ["P982-100-30"], "Items": [{ "customerMaterialNo": ["7394012"], "supplierMaterialNo": ["581204700"], "description": ["WORKWEAR ITEM"] }], }	Baseline: 581204700 AW: 7394012 ReAct: 7394012 PlanExe: 7394012 DAC: 7394012
TAX INVOICE Name MERIDIAN HOLDINGS PTY LTD Tower 2 Level 22 Suite 8 81 King St SYDNEY NSW 2000 Australia Invoice No. 8010481 Invoice Date 15/03/2025 Due Date 04/04/2025 Customer Reference 738531567 Ship To MERIDIAN HOLDINGS PTY LTD ACCOUNTS DEPT (B2B) Tower 2 Level 22 Suite 8 81 King St # Code Product Description Comment Unit Quantity Price Discount Amount SDI 1 ITEM500183GBL 4.8M STRUCTURAL PANEL - 3 REL Each 0.00 108.19 0.00% 3,480.00 N/A Invoice Lines Additional Charges Total Invoice Before Tax 3,163.84 Before Tax 0.00 Before Tax 3,163.84 Tax 316.39 Tax 0.00 Tax 316.39 Total 3,480.00 Total 0.00 Total 3,480.00 Payment Total 0.00 Due 3,480.00	Document Type: Invoice Field: <i>li.materialNo</i> The document contains two material numbers from customer and supplier under the same column <i>CODE</i>. Without ERP data, it is challenging to distinguish the two.	"7300001547": { "buyerName": ["MERIDIAN HOLDINGS"], "paymentTerms": [], ... "Items": [{ "description": ["4.8M PANEL"], "materialNumber": ["ITEM500183GBL"], "quantity": ["32"], "unitOfMeasure": ["EA"] }], }	Baseline: 4800PANEL3RAIL: AW: ITEM500183GBL ReAct: ITEM500183GBL PlanExe: ITEM500183GBL DAC: ITEM500183GBL
DELIVERY NOTE No.: NV4817390 19. March 2025 Our ref. Your P Order Customer G7291MF Northgate Manufacturing Limited Dept. TO-42, Westside Supplier Code 803714-05 Shipment Details GLS Terms of Shipment FCA Leipzig (Incoterms) Project 310 Your Doc No. G7291MF Order No. NVS381047 Pos Quantity Unit Article/Description Tariff No. Weight [kg] 1 3 Pc. Precision bearing assembly unit 5024 5031 - 7382916 ohne spindel 8103847 Es handelt sich um eine Strukturteile Auslieferung 8 kg Net, 10 kg Brutto, 5 kg LBS 8482 1090 2.1	Document Type: Delivery Note Field: <i>li.purchaseOrderNo</i> There are two <i>purchaseOrderNo</i> instances on this document, but both are header <i>purchaseOrderNo</i>. Baseline, AW, and DAC hallucinate <i>li.purchaseOrderNo</i>, whereas ReAct and PlanExe get the correct empty output.	"G7291MF": { "supplierId": ["803714-05"], "orderDate": ["08.02.25"], "Items": [{ "customerMaterialNo": ["8103847"], "supplierMaterialNo": ["9204 5031"], "description": [], "itemNumber": ["1"], "quantity": ["3"], "unitOfMeasure": ["Pc."], }], }	Baseline: G7291MF AW: G7291MF ReAct: [''] PlanExe: [''] DAC: G7291MF

Table 19: Detailed qualitative examples with ERP master data context. **Green** = correct extraction, **Red** = incorrect extraction.

Doc	Field and Pattern	% of type	Fix
<i>Wrong-Value Errors</i>			
DN	<i>purchaseOrderNo</i> : replaced by supplier PO	93%	ERP lookup (AW: 27/28 fixed, $\Delta F_1 +0.119$)
DN	<i>deliveryDate</i> → <i>documentDate</i> swap	80%	Field-disambiguation prompt
INV	<i>purchaseOrderNo</i> : prefix retained (PO#, Order:)	14%	Output normalisation
INV	<i>vendorName</i> : truncation at comma / parenthesis	68%	Prompt instruction
INV	<i>li.quantity</i> : separator confusion (1,500 ↔ 1500)	23%	Locale-aware normalisation
INV	<i>shipToAddress</i> / <i>buyerAddress</i> : company name omitted	71–83%	Prompt instruction
<i>Missing-Value Errors</i>			
DN	<i>documentDate</i> : visual ambiguity; ReAct/PlanExe reduce misses but raise wrong-values by similar amount	17%	Field-disambiguation prompt
DN	<i>li.purchaseOrderItemNo</i> : near-absent in documents (2.8% of line items)	37.5%	Low-support field; indicative only
INV	<i>li.purchaseOrderNo</i> : field absent in most line items	14.3%	Null-output instruction
<i>Hallucinations</i>			
DN	<i>li.purchaseOrderNo</i> : header PO copied to line item	83%	Null-output instruction ($\sim 2pp$, $\Delta F_1 +0.145$)
DN	<i>li.customerMaterialNo</i> : supplier material number copied	54%	Null-output instruction ($\Delta F_1 +0.088$)
DN	<i>li.unitOfMeasure</i> : EA injected when field absent	avg 15/config	Persistent prior; prompt mitigation partial
DN	<i>li.purchaseOrderItemNo</i> : ReAct invents sequential item nos.	avg 6.2/config	ReAct-specific regression ($\Delta F_1 -0.078$)
INV	<i>li.purchaseOrderNo</i> : header PO copied to line item	63%	Null-output instruction ($\sim 2pp$)
INV	<i>vendorName</i> : name injected when field absent	100%	Prompt instruction
<i>Performance Ceilings</i>			
DN	<i>deliveryDate</i> : cross-field confusion with <i>documentDate</i>	43% concordance	Field-disambiguation prompt
INV	<i>paymentTerms</i> : exact-match penalises semantically equivalent phrasings	89% concordance	Fuzzy-match scorer
INV	<i>li.description</i> : free-text variation penalised by exact-match	74% concordance	Fuzzy-match scorer

Table 20: Systematic error sub-patterns by type, document type, and affected field. % of type: fraction of that error type on the given field at baseline. ΔF_1 : avg change vs. baseline across all paradigms and models.