computers

MDPI

*Article*

# An Adaptive Multi-Agent Framework for Semantic-Aware Process Mining

**Xiaohan Su [1], Bin Liang [1,*], Zhidong Li [1], Yifei Dong [1], Justin Wang [2] and Fang Chen [1]**

[1] Faculty of Engineering and IT, University of Technology Sydney, 15 Broadway, Ultimo, NSW 2007, Australia; xiaohan.su@student.uts.edu.au (X.S.); zhidong.li@uts.edu.au (Z.L.); yifei.dong@uts.edu.au (Y.D.); fang.chen@uts.edu.au (F.C.)
[2] The Property Investors Alliance Pty Ltd., 2 Australia Ave, Sydney, NSW 2127, Australia; justin@pia.com.au
* Correspondence: bin.liang@uts.edu.au

**Abstract**

With rapid advancements in large language models for natural language processing, their role in semantic-aware process mining is growing. We study semantics-aware process mining, where decisions must reflect both event logs and textual rules. We propose an online, adaptive multi-agent framework that operates over a single knowledge base shared across three tasks—semantic next-activity prediction (S_NAP), trace-level semantic anomaly detection (T_SAD), and activity-level semantic anomaly detection (A_SAD). The approach has three key elements: (i) cross-task corroboration at retrieval time, formed by pooling in-domain and out-of-domain candidates to strengthen coverage; (ii) feedback-to-index calibration that converts user correctness/usefulness into propensity-debiased, smoothed priors that immediately bias recall and first-stage ordering for the next query; and (iii) stability controls—consistency-aware scoring, confidence gating with failure-driven query rewriting, task-level trust regions, and a sequential rule to select the relevance–quality interpolation. We instantiate the framework with Mistral-7B-Instruct, Llama-3-8B, GPT-3.5, and GPT-4o and evaluate it using macro-F1. Compared to in-context learning, our framework improves S_NAP, T_SAD, and A_SAD by 44.0%, 15.6%, and 7.1%, respectively, and attains the best overall profile against retrieval-only and correction-centric baselines. Ablations show that removing index priors causes the steepest degradation, cross-task corroboration yields consistent gains—most visibly on S_NAP—and confidence gating preserves robustness to difficult inputs. The result is immediate serve-time adaptivity without heavy fine-tuning, making semantic process analysis practical under drift.

**Keywords:** process mining; LLM; RAG; multi-agent



Academic Editor: Paolo Bellavista

Received: 2 October 2025
Revised: 30 October 2025
Accepted: 31 October 2025
Published: 5 November 2025

**Citation:** Su, X.; Liang, B.; Li, Z.; Dong, Y.; Wang, J.; Chen, F. An Adaptive Multi-Agent Framework for Semantic-Aware Process Mining. *Computers* **2025**, *14*, 481. https://doi.org/10.3390/computers14110481



## 1. Introduction

Process mining studies how to discover, check, and improve real processes from event data, traditionally grounding itself in control-flow structure and standardized artifacts such as XES event logs and Petri-net models [1–3]. These standards have enabled interoperable tooling and large-scale benchmarking, yet they presuppose that the decisive information for discovery and conformance resides in the ordering and frequency of events [1,2,4,5]. In many operational settings, however, admissibility depends on textual policies, case notes, and domain manuals that are not fully captured by precedence relations—shifting the analytical burden from purely structural patterns to the semantics that link activities, resources, and constraints. This gap motivates semantics-aware problem formulations that

require reasoning over both logged behavior and unstructured knowledge. Recent position pieces and handbooks echo this shift: structure remains necessary, but accuracy increasingly depends on how well frameworks surface and use task-relevant textual evidence alongside models of allowed behavior [2,4,5].

Within this direction, tasks naturally divide by granularity. Activity-level semantic anomaly detection (A_SAD) asks whether a local relation between two activities is meaning-preserving given domain rules; trace-level semantic anomaly detection (T_SAD) asks whether an entire execution violates textual expectations even if it is frequent; semantic next-activity prediction (S_NAP) requires forecasting the next admissible step based on both history and rules expressed in text. These formulations articulate a complementary view to frequency-based discovery and conformance: the same sequence may be structurally common yet semantically inadmissible, while a rare variant may be semantically valid under documented exceptions. Benchmarks that instantiate these tasks show that the relevant signals are often buried in heterogeneous documents and case metadata rather than in the event log alone, making retrieval and evidence selection first-order concerns rather than ancillary steps [6,7].

Large language models (LLMs) offer a means to unify textual knowledge with sequence context: they can represent domain rules, map heterogeneous artifacts into a shared space, and reason over activity–constraint relations. Yet three challenges persist. First, evidence localization: generic retrieval pipelines can return passages that are topically close but decision-irrelevant, leading to semantically plausible yet wrong judgments. Second, context construction: even with good passages, assembling a context window that preserves dependencies (e.g., order constraints, exception clauses, or cross-referenced sections) is brittle, and small changes in ranking can flip decisions at activity or trace level. Third, adaptation under drift: initial prompts or few-shot exemplars rarely generalize across processes, and most deployments consume user feedback only through offline fine-tuning, so next-query behavior often fails to reflect freshly observed correctness signals. Empirical studies corroborate these bottlenecks: out-of-the-box prompting underperforms on semantics-aware tasks, while task-adapted models improve—but their gains remain sensitive to retrieval quality and how textual expectations are represented during inference [6,7].

These observations crystallize a problem setting rather than a specific solution blueprint. The framework must operate over a unified, standards-compatible substrate (e.g., XES/PNML) to remain interoperable with discovery and conformance tooling; it must expose task interfaces for A_SAD, T_SAD, and S_NAP while treating textual knowledge as first-class evidence; and it must reason under non-stationarity, where targets, policies, and documentation evolve over time. Concretely, an effective semantics-aware process-mining stack should (i) retrieve domain-grounded evidence that is causally tied to decisions, not merely topically similar; (ii) build contexts that respect activity dependencies and exceptions; (iii) reflect user-provided correctness/usefulness signals with minimal delay to avoid repeated failure modes; and (iv) preserve stability so that local updates do not cascade into erratic behavior on adjacent tasks or variants. Recent investigations into instruction-tuning across multiple PM tasks reinforce the need for such serve-time capabilities: tuning helps, but its impact varies by task—particularly on anomaly detection—suggesting that generalization hinges on how evidence is selected and weighted at inference time rather than on model size alone [8].

In summary, the field is converging on a semantics-aware view where textual knowledge, structural conformance, and sequence prediction are tightly coupled. The core research problem is to formulate serve-time mechanisms that reliably surface decision-relevant evidence from heterogeneous sources, assemble it into constraint-aware contexts,

and evolve with feedback—while remaining compatible with the standards and evaluation protocols that anchor process mining in practice [1,2,4,6].

We build on these directions but target online, semantics-aware process mining in a multi-task setting [9]. Concretely, we (i) operate a shared knowledge base across three process-mining tasks and allow cross-task corroboration at query time; (ii) calibrate retrieval online from user feedback by pushing correctness/usefulness signals into index priors that bias recall and initial ranking for the next query; and (iii) stabilize updates via an online schedule that includes corrective gating and periodic dual instruction tuning. This design aims to couple the adaptability of feedback-driven RAG with the domain regularities and evaluation protocols of process mining [6].

The main contributions of this paper are summarized as follows:

1. We present a RAG-based multi-agent framework with one shared knowledge base for A_SAD, T_SAD, and S_NAP; in semantics-aware process mining this enables cross-task corroboration and improves coverage and consistency.
2. We convert human correctness and passage usefulness into index priors updated online; in feedback-driven RAG this yields next-query gains in recall and initial ranking.
3. We design an online correction loop with failure-driven query rewriting and confidence gating; in dynamic settings this suppresses noisy retrieval and stabilizes output quality.

## 2. Related Work

### *2.1. Semantics-Aware Process Mining with LLMs*

Semantics-aware PM elevates textual expectations (policies, manuals, case notes) to decision-critical evidence that complements control-flow structure. Early explorations reported promise but also a sharp gap between out-of-the-box prompting and task-robust behavior once anomaly detection and semantic prediction were evaluated frameworkatically [10]. Benchmark studies made this gap explicit: LLMs underperform on semantics-heavy tasks when only prompted or given few-shot ICL, whereas supervised adaptation delivers stable gains and reveals that retrieval quality and context assembly dominate parameter count for downstream accuracy [6,7,11]. Instruction-tuning across multiple PM tasks improves discovery and prediction but remains mixed for anomaly detection, underscoring that serve-time evidence control is pivotal even with stronger priors [8]. Beyond PM-specific suites, broader evaluations echo similar sensitivities—ranking noise and brittle context construction often flip local activity decisions or whole-trace judgments—placing a premium on evidence localization and constraint-aware context building at inference time [6,11].

These observations motivate architectures that (i) stay compatible with standards and log-based conformance tooling, (ii) treat textual knowledge as first-class evidence, and (iii) adapt under policy/document drift. Our framework follows this arc but targets two fragile points repeatedly flagged by the literature: evidence localization and next-query adaptivity. First, instead of a single in-domain retriever, we pool in-domain and cross-task candidates and reweight them by consistency with log-derived constraints, addressing the finding that small ranking perturbations can flip A_SAD/T_SAD decisions [6]. Second, we propagate human correctness/usefulness into propensity-debiased index priors that immediately affect recall and first-stage ordering—aligning serve-time behavior with feedback signals that prior work typically consumes only offline [7,8,11].

### *2.2. Retrieval-Augmented Generation: Ranking, Feedback, and Multi-Agent Control*

RAG has progressed from static retrieve-then-generate to pipelines that reason about when and what to retrieve and how to value evidence. Self-reflective methods learn triggers

and critique contexts, improving factuality but leaving the retrieval surface unchanged between queries [12,13]. Corrective controllers explicitly score list quality and branch to re-search or decompose–recompose under low confidence, yielding robustness without persistent write-back [14]. Ranking-centric approaches instruction-tune a single LLM to both rank and generate, showing that modest ranking supervision can surpass expert rankers while preserving generation quality [15]. Production-oriented work aligns ordering with listwise human feedback signals [16] and transfers fine-grained span/passage preferences from a teacher LLM into the retriever to narrow the relevance–helpfulness gap [17]. Dual instruction tuning co-adapts generator and retriever with lightweight stages and serves as an effective consolidation mechanism between online phases [18]. Surveys synthesize these trends and emphasize the need for knowledge-oriented selection, latency-aware design, and alignment between retrieved facts and generative objectives [19,20].

Orthogonally, multi-agent RAG improves reliability via role specialization and consensus filtering without heavy fine-tuning [21]. Active RAG lines decide when/what to retrieve during generation [22] and further propose knowledge-assimilation/ accommodation agents that reconcile external evidence with parametric memory [23]. Where long-tail or cross-referenced evidence is critical, GraphRAG encodes structural relations to support multi-hop retrieval; recent surveys consolidate graph indexing and graph-guided retrieval practice [24–27]. Across these strands, two limitations recur for semantics-aware PM: many frameworks either gate low-confidence contexts or learn better rankers, yet seldom write correctness/usefulness back into the index to alter next-query recall/ordering; and they rarely exploit cross-task corroboration. Our design addresses both by (i) converting debiased feedback into smoothed priors that immediately bias recall and first-stage ranks, and (ii) querying a shared store across S_NAP/T_SAD/A_SAD with consistency-aware scoring—improving coverage and stability in dense, near-neighbor decision regimes [28].

## 3. Decomposed Task

This paper proposes a method that combines RAG, a multi-agent framework, and multiple LLMs to address the three core tasks in process mining: Activity-Level Semantic Anomaly Detection (A_SAD), Trace-Level Semantic Anomaly Detection (T_SAD), and Semantic Next Activity Prediction (S_NAP). Below, we define each task and describe how RAG generates context, assigns tasks to different agents, and the specific logic each agent follows during execution.

### *3.1. Preliminaries*

Let $\mathcal{A}$ denote the set of activities, and let $\mathcal{T}$ represent the set of traces, where each trace $\sigma \in \mathcal{T}$ is an activity sequence $\sigma = \langle a_1, a_2, \ldots, a_n \rangle$, with $a_i \in \mathcal{A}$. The event log $L$ contains multiple traces, representing real process instances. The set of tasks $\mathcal{Q} = \{q_1, q_2, q_3, \ldots, q_m\}$ denotes multiple task requests received by the framework, each of which will be processed by a specific agent based on its characteristics.

### *3.2. Multi-Agent Task Assignment and Context Generation*

When a task $q \in \mathcal{Q}$ enters the framework, RAG first determines the task type (A_SAD, T_SAD, or S_NAP) and generates the appropriate context $\mathcal{C}_q$ for the task. The context $\mathcal{C}_q$ generated by RAG is a set of relevant information, including process structure, activity dependencies, and event sequences, which supports the semantic analysis and execution of each agent. We define the context generation function $g$ as follows:

$$\mathcal{C}_q = g(q, L)$$

where the function $g$ generates context based on the type of task $q$ and information in the event log $L$. RAG then assigns the task $q$ to the corresponding agent, either $\alpha_{\text{A\_SAD}}$, $\alpha_{\text{T\_SAD}}$, or $\alpha_{\text{S\_NAP}}$, based on the generated context $\mathcal{C}_q$.

### *3.3. Activity-Level Semantic Anomaly Detection (A_SAD)*

The A_SAD task is executed by the agent $\alpha_{\text{A\_SAD}}$, which aims to detect semantic anomalies in the order of activities within a trace. Given a trace $\sigma = \langle a_1, \ldots, a_n \rangle$, we define the semantic dependency between activities $(a_i, a_j)$ as $a_i \prec_\sigma a_j$, if and only if in $\sigma$, $a_i$ occurs before $a_j$ $(1 \leq i < j \leq n)$. The agent $\alpha_{\text{A\_SAD}}$ uses the context $\mathcal{C}_q$ to verify this dependency:

$$f_{\alpha_{\text{A\_SAD}}}(a_i, a_j \mid \mathcal{C}_q) = \begin{cases} 1, & \text{if the order of } (a_i, a_j) \text{ is consistent} \\ & \text{with the semantic logic in } \mathcal{C}_q, \\ 0, & \text{otherwise.} \end{cases}$$

where $f_{\alpha_{\text{A\_SAD}}}$ represents the decision function of agent $\alpha_{\text{A\_SAD}}$ based on context $\mathcal{C}_q$.

### *3.4. Trace-Level Semantic Anomaly Detection (T_SAD)*

The T_SAD task, executed by agent $\alpha_{\text{T\_SAD}}$, aims to detect semantic anomalies across the entire trace. We define the decision function $f_{\alpha_{\text{T\_SAD}}} : \mathcal{T} \times \mathcal{C}_q \to \{0, 1\}$, which assesses any trace $\sigma \in \mathcal{T}$ as follows:

$$f_{\alpha_{\text{T\_SAD}}}(\sigma \mid \mathcal{C}_q) = \begin{cases} 1, & \text{if } \sigma \text{ conforms to the semantic rules in} \\ & \mathcal{C}_q, \\ 0, & \text{otherwise.} \end{cases}$$

Agent $\alpha_{\text{T\_SAD}}$ utilizes information on trace logic and dependencies from $\mathcal{C}_q$ to perform semantic analysis of activity order and dependencies within each trace.

### *3.5. Semantic Next Activity Prediction (S_NAP)*

The S_NAP task, executed by agent $\alpha_{\text{S\_NAP}}$, predicts the next logical activity based on a given sequence of completed activities. Given a partial trace $\sigma_k = \langle a_1, \ldots, a_k \rangle$ and context $\mathcal{C}_q$, agent $\alpha_{\text{S\_NAP}}$ uses the dependency information in $\mathcal{C}_q$ to predict the next activity $a_{k+1}$:

$$f_{\alpha_{\text{S\_NAP}}}(\sigma_k \mid \mathcal{C}_q) = a_{k+1}, \quad a_{k+1} \in \mathcal{A}$$

where $f_{\alpha_{\text{S\_NAP}}}$ is the prediction function of agent $\alpha_{\text{S\_NAP}}$ based on context $\mathcal{C}_q$, and $a_{k+1}$ is the next logical activity following $\sigma_k$.

### *3.6. Task Execution Flow*

1. Task Ingestion: Upon receiving a task $q \in \mathcal{Q}$, RAG generates the appropriate context $\mathcal{C}_q = g(q, L)$ based on the task type and event log data.
2. Task Assignment: RAG automatically assigns the task $q$ to the corresponding agent: $\alpha_{\text{A\_SAD}}$, $\alpha_{\text{T\_SAD}}$, or $\alpha_{\text{S\_NAP}}$.
3. Agent Execution: Each agent performs semantic analysis and decision-making based on its designated context $\mathcal{C}_q$ and the task-specific decision or prediction function $f_\alpha$.

This approach effectively integrates RAG, multi-agent frameworks, and multiple LLMs to address the A_SAD, T_SAD, and S_NAP tasks within process mining.

## 4. Method

The LLM-based process mining framework integrates RAG, a multi-agent framework, and LLMs, as illustrated in Figure 1. The workflow is divided into several key steps:

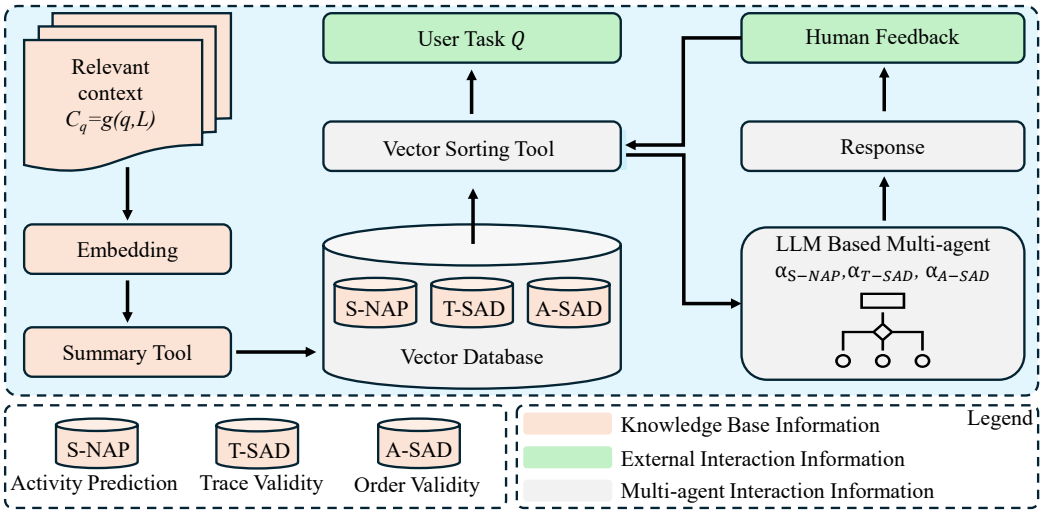


**Figure 1.** The framework of multi-agent in process mining.

To facilitate faithful implementation and reproducibility, we provide a compact pseudocode of the online cross-task retrieval and feedback-calibrated execution loop. As summarized in Algorithm 1, the following pseudocode outlines the online cross-task retrieval and feedback-calibrated execution loop. The notation and update rules are fully consistent with (1)–(8) and serve as a minimal, implementation-ready workflow.

**Algorithm 1.** Online Cross-Task Retrieval and Feedback-Calibrated Execution Loop.

**Input:** Event log $L$; KB $\{\mathcal{D}_{\text{S\_NAP}}, \mathcal{D}_{\text{T\_SAD}}, \mathcal{D}_{\text{A\_SAD}}\}$; scorers $\text{sim}_t, \text{sim}_{\neg t}$; consistency $\Delta_{\text{cons}}$; table $Q(\cdot)$; $K, \lambda_1, \lambda_2, \gamma, \beta, (\alpha_{\min}, \alpha_{\max}), w_t, \rho, \eta_t$; propensity $\pi(\cdot)$; priors $(\alpha_0, \beta_0)$; optional shrinkage $\xi$.

**Output:** For each task $q$ of type $t \in \{\text{S\_NAP}, \text{T\_SAD}, \text{A\_SAD}\}$, return the decision and update retrieval state.

1: **Candidates** $\mathcal{D}_{\neg t} \leftarrow \bigcup_{t' \neq t} \mathcal{D}_{t'}$; $\mathcal{E}_q \leftarrow \text{TopK}(\mathcal{D}_t; q) \cup \text{TopK}(\mathcal{D}_{\neg t}; q)$

2: **Base score** For $e \in \mathcal{E}_q$: $s_{\text{base}}(e \mid q) \leftarrow \lambda_1 \text{sim}_t(q, e) + \lambda_2 \text{sim}_{\neg t}(q, e)$ ▷ Equation (1)

3: **Consistency** $\tilde{s}_{\text{base}}(e \mid q) \leftarrow s_{\text{base}}(e \mid q) + \gamma \Delta_{\text{cons}}(e \mid q, L)$ ▷ Equation (2)

4: **Context** $\mathcal{C}_q \leftarrow$ concatenate top-$K$ under $\tilde{s}_{\text{base}}$

5: **Execute** If $t = \text{A\_SAD}$ use $f_{\alpha_{\text{A\_SAD}}}$; else if $t = \text{T\_SAD}$ use $f_{\alpha_{\text{T\_SAD}}}$; else predict via $f_{\alpha_{\text{S\_NAP}}}$; record decisive set $\mathcal{E}_q^\star$

6: **Feedback** Obtain $r_u \in [0, 1]$; for displayed $e$ at position $p$: weight $w \leftarrow 1/\pi(p)$

7: **Quality update** $\hat{r}_u \leftarrow \dfrac{\alpha_0 + \sum_i w_i r_{u,i}}{\alpha_0 + \beta_0 + \sum_i w_i}$; $Q_{\text{cand}}(e) \leftarrow \text{clip}((1-\beta)Q(e) + \beta \hat{r}_u, Q_{\min}, Q_{\max})$ ▷ Equations (3) and (4)

8: **Stability** $\Delta Q_t \leftarrow \sum_{e \in \mathcal{E}_q^\star} (Q_{\text{cand}}(e) - Q(e))$; enforce $\|\Delta Q_t\|_1 \leq \eta_t$ (Equation (7)); compute $\Psi$ by Equation (8); if over threshold, apply penalty $\rho$ and freeze conflicting chunks

9: **Next scoring** $s(e \mid q') = (1 - \alpha_t)\tilde{s}_{\text{base}}(e \mid q') + \alpha_t (Q(e) \cdot w_t)$; normalize by list width when applicable ▷ Equation (5)

10: **Select** $\alpha_t$ Choose $\alpha_t \in [\alpha_{\min}, \alpha_{\max}]$ that maximizes $\mu_t(\alpha) + c\,\sigma_t(\alpha)$ under a sliding window; if high noise, set $Q(e) \leftarrow (1 - \xi)Q(e) + \xi\,\bar{Q}_t$

### *4.1. Online Cross-Task Retrieval with a Unified Knowledge Base*

We maintain a unified knowledge base with three task-specific sub-databases $\mathcal{D}_{\text{S\_NAP}}, \mathcal{D}_{\text{T\_SAD}}, \mathcal{D}_{\text{A\_SAD}}$. For an incoming task $q \in \mathcal{Q}$ of type $t \in \{\text{S\_NAP}, \text{T\_SAD}, \text{A\_SAD}\}$, the base context is generated as $\mathcal{C}_q = g(q, L)$ (definitions unchanged). To strengthen coverage at run time, we augment retrieval with cross-task evidence. Let $\mathcal{D}_{\neg t} = \bigcup_{t' \neq t} \mathcal{D}_{t'}$ and define the candidate pool

$$\mathcal{E}_q \;=\; \text{TopK}(\mathcal{D}_t; q) \;\cup\; \text{TopK}(\mathcal{D}_{\neg t}; q).$$

Each chunk $e \in \mathcal{E}_q$ receives a base score that combines in-domain and cross-task similarities:

$$s_{\text{base}}(e \mid q) = \lambda_1 \,\text{sim}_t(q, e) + \lambda_2 \,\text{sim}_{\neg t}(q, e), \qquad \lambda_1, \lambda_2 \geq 0,\ \lambda_1 + \lambda_2 = 1, \tag{1}$$

where $\text{sim}_t$ and $\text{sim}_{\neg t}$ can be computed by a dual-encoder with an optional cross-encoder re-ranking stage. To enforce consistency with activity precedence and trace conformance in $L$, we add

$$\tilde{s}_{\text{base}}(e \mid q) \;=\; s_{\text{base}}(e \mid q) \;+\; \gamma\, \Delta_{\text{cons}}(e \mid q, L), \qquad \gamma \geq 0, \tag{2}$$

where $\Delta_{\text{cons}}$ aggregates pairwise entailment/contradiction signals among candidates relative to dependencies encoded by $L$. The pre-feedback context is the top-$K$ set under $\tilde{s}_{\text{base}}$ and is concatenated to yield $\mathcal{C}_q$ for the designated module $\alpha_t$.

### *4.2. Feedback-Calibrated Reranking with Debiasing and Sequential Selection*

After $\alpha_t$ produces an answer from $\mathcal{C}_q$, the user provides a rating $r_u \in [0, 1]$. For each chunk $e$, we maintain a quality statistic $Q(e)$ updated online and used to calibrate future retrieval orders. To correct presentation bias in implicit signals, we employ inverse-propensity weighting with Beta-prior smoothing. Let $\pi_i$ be the exposure propensity of $e$ at its displayed position in interaction $i$, $w_i = 1/\pi_i$ the correction weight, and define

$$\hat{r}_u \;=\; \frac{\alpha_0 + \sum_{i=1}^{n} w_i\, r_{u,i}}{\alpha_0 + \beta_0 + \sum_{i=1}^{n} w_i}, \tag{3}$$

which is then integrated by an exponential moving average with clipping,

$$Q(e) \;\leftarrow\; \text{clip}\Big((1-\beta)\, Q(e) + \beta\, \hat{r}_u,\ Q_{\min},\ Q_{\max}\Big), \qquad \beta \in (0, 1], \tag{4}$$

to limit extreme updates under noisy or sparse feedback.

At the next query $q'$, a calibrated score combines relevance and learned quality:

$$s(e \mid q') \;=\; (1 - \alpha_t)\, \tilde{s}_{\text{base}}(e \mid q') \;+\; \alpha_t \left(Q(e) \cdot w_t\right), \qquad w_t > 0,\ \alpha_t \in [\alpha_{\min}, \alpha_{\max}], \tag{5}$$

where $w_t$ emphasizes task relevance and $\alpha_t$ controls the interpolation between relevance and quality. The value of $\alpha_t$ is chosen by a sequential online selection rule that balances exploration and exploitation with finite-time guarantees. We instantiate two standard choices. First, an upper-confidence rule on a sliding window:

$$\alpha_t \in \arg \max_{\alpha \in [\alpha_{\min}, \alpha_{\max}]} \big\{ \mu_t(\alpha) + c\, \sigma_t(\alpha) \big\}, \tag{6}$$

where $\mu_t(\alpha)$ and $\sigma_t(\alpha)$ are, respectively, the empirical mean and uncertainty proxy of the utility induced by (5) for task $t$, and $c > 0$ scales the confidence term. Second, a posterior-sampling rule that maintains a posterior over utilities for each candidate $\alpha$; draw $\tilde{\mu}_t(\alpha)$ from the posterior and select $\alpha_t \in \arg\max_\alpha \tilde{\mu}_t(\alpha)$. Delayed ratings are handled by updating the window statistics or posterior when feedback arrives while keeping $\alpha_t$ fixed within a decision epoch.

When multiple chunks form $\mathcal{C}_{q'}$, we normalize the effective contribution of $Q(e)$ by the number of displayed items at comparable ranks to reduce list-truncation effects; the same propensity model for $\pi_i$ is reused for residual bias correction. For high-noise regimes, we optionally shrink $Q(e)$ toward the task mean $\bar{Q}_t$ via $Q(e) \leftarrow (1-\xi)Q(e) + \xi\,\bar{Q}_t$ with small $\xi$, which stabilizes short-horizon variance while preserving long-run adaptivity.

### *4.3. Multi-Module Execution and Stability Under Non-Stationarity*

Given $\mathcal{C}_q$ and the task type $t$, the corresponding module $\alpha_t$ executes the task-specific decision rule defined earlier (A_SAD, T_SAD, or S_NAP) and records the decisive subset $\mathcal{E}_q^\star \subseteq \mathcal{E}_q$ that influenced the output. To avoid oscillations, we constrain the aggregate update magnitude on decisive evidence by a task-level trust region:

$$\|\Delta Q_t\|_1 \;\le\; \eta_t, \qquad \Delta Q_t \;=\; \sum_{e \in \mathcal{E}_q^\star} \big(Q_{\text{new}}(e) - Q_{\text{old}}(e)\big), \tag{7}$$

where $\eta_t > 0$ may decay over time. We monitor cross-task disagreement via

$$\Psi \;=\; \sum_t \sum_{e \in \mathcal{E}_q^\star} \Big|\operatorname{sign}\big(\Delta_{\text{cons}}(e \mid q, L)\big) - \operatorname{sign}\big(\Delta Q_t(e)\big)\Big|, \tag{8}$$

and, if $\Psi$ exceeds a threshold, we temporarily downweight conflicting chunks in (5) by multiplying a penalty factor $\rho \in (0,1)$ and freeze their $Q(e)$ updates until consistency resumes. To avoid premature reliance on cross-task sources, $\lambda_2$ in (1) is annealed from a small initial value to a validated plateau, so that early decisions prefer in-domain evidence before gradually admitting cross-task corroboration. The complete online loop is: construct $\mathcal{E}_q$, compute $\tilde{s}_{\text{base}}$ by (1) and (2), form $\mathcal{C}_q$, execute $\alpha_t$, collect $r_u$, update $Q(e)$ via (3) and (4) under constraints (7) and (8), and choose $\alpha_t$ for the next round via (6) or posterior sampling. This realizes an online cross-task retrieval–generation pipeline in which calibrated ranking, sequential selection, and stability control jointly improve context quality and downstream decisions. As illustrated in Figure 2, this procedure is exemplified by a T_SAD prompt and its retrieved evidence context.

**Question (T_SAD).** Given the trace $\sigma$=Register_Case, Assess_Risk, Approve_Loan, Disburse_Funds, decide whether the execution is semantically admissible under the domain rules that govern risk assessment, approvals, and disbursement.

**Prompt to LLM (served by our framework).**

```
You are a process-mining assistant deciding if a full trace violates
textual rules (T\_SAD). Use the evidence snippets to justify your decision.
Task: T\_SAD (trace-level semantic anomaly detection)
Trace: <Register_Case, Assess_Risk, Approve_Loan, Disburse_Funds>
Output format (JSON):
{"label": "ADMISSIBLE" | "ANOMALOUS",
  "rationale": "... one paragraph using cited snippets [E#] ..."}
Decision rules:
- Respect constraints and exception clauses stated in the evidence.
- Prefer snippets that explicitly bind activities to preconditions.
- If snippets conflict, follow the most specific, later-dated rule.
Evidence snippets [E#] (do not fabricate beyond these):
[will be filled by RAG Context below]
```

---

**RAG Context (provided by our retriever; in_task evidence + cross_task corroboration).**

*In_task evidence (T_SAD):*

- **[E1] Policy Manual §4.2 (v2023-10).** "Approve_Loan shall not occur before Verify_Documents and Risk_Score_Finalized. If Assess_Risk returns `High`, a second-level review Review_Senior is mandatory prior to approval."
- **[E2] Operating Guide §5.1 (Exceptions).** "For `Low` risk cases, Approve_Loan may directly follow Assess_Risk provided Verify_Documents is already archived in the case file; otherwise approval is non-admissible."
- **[E3] Compliance Bulletin (2024-02).** "Disburse_Funds requires a recorded Approval_ID and completed KYC; disbursement before Verify_Documents is considered anomalous even if approval is present."

Cross_task corroboration (A_SAD & S_NAP pulled into the same context window):

- **[X1] A_SAD pair rule mined from audit notes.** Local order constraint: (Assess_Risk, Approve_Loan) is admissible only if flag `Docs_Verified=true`. Violations frequently arise when approval precedes verification for `Medium/High` risk.
- **[X2] S_NAP suggestion for the same partial prefix** ⟨Register_Case, Assess_Risk⟩: next admissible steps ranked by probability and rule coverage: Verify_Documents (top), Request_More_Info, then Approve_Loan only under `Low` risk with pre-verified docs.
- **[X3] Log-derived consistency signal** (from recent traces): disbursement events without prior Verify_Documents produce downstream KYC reversals; these cases were labeled anomalous by human reviewers.

---

**What the LLM will return under our pipeline.**

```
{"label": "ANOMALOUS",
  "rationale": "E1 and E3 require Verify_Documents and KYC before approval/
  disbursement. The given trace lacks Verify_Documents. X1 flags the local
  pair (Assess_Risk, Approve_Loan) as admissible only when docs are verified.
  X2 corroborates that Verify_Documents should precede approval for typical
  prefixes. Therefore the full trace violates textual rules."
}
```

**Figure 2.** T_SAD Prompt and RAG-Provided Context.

## 5. Experimental Setup

This experiment focuses on three distinct tasks in process mining: T_SAD, A_SAD, and S_NAP, utilizing the multi-agent framework based on LLM and RAG as described above. Through constructing relevant datasets, the RAG module first builds a knowledge base from documents and generates vector embeddings. We then pose questions derived from the test set to the RAG module, and responses from the multi-agent framework are compared with ground truth data. Each multi-agent setup, assigned to a different task, is tested with all LLM models, with results reported as success rates.

### 5.1. Dataset Construction

To ensure representative model performance across different tasks, we adopted a data partitioning approach similar to that used by Adrian Rebmann et al. [6]. The original event data is derived from the SAP-SAM (SAP Signavio Academic Models) collection, which consists primarily of English BPMN process models. From this collection, we generated allowed execution sequences based on sound workflow nets. In total, the corpus contains approximately 16,000 models, 49,000 unique activities, and 163,000 valid execution sequences.

Only BPMN models that could be converted into sound workflow nets were retained, and duplicated activity sets were removed during preprocessing.

Specifically, we extracted 10% of the original dataset for each task as the test set. However, unlike previous methods, the remaining data was converted into PDF documents to serve as the knowledge base for RAG. Each trace is stored as an individual document, while traces belonging to the same case are grouped under a shared case header to provide contextual continuity without aggregation. This structure enables retrieval at the trace level while preserving the semantic link between traces of the same case. To prevent data leakage, the test set and the RAG knowledge base are strictly separated: no execution trace, case, or process model appears in both sets. The split is conducted at the process-model level rather than the sequence level, ensuring that different executions of the same model cannot leak from the knowledge base into the evaluation stage. Table 1 shows the data distribution for each task, including the total dataset size, the portion stored as documents for the RAG knowledge base, and the test set size.

**Table 1.** Data distribution for each task, showing the total dataset size, the portion used for the RAG document base, and the test set size.

| Task | Total | Document | Test |
|---|---|---|---|
| T_SAD | 291,251 | 271,501 | 19,750 |
| A_SAD | 316,308 | 285,556 | 30,752 |
| S_NAP | 1,289,081 | 120,011 | 50,741 |

### 5.2. RAG Knowledge Base Construction

The RAG module plays a critical role in providing context-specific support for each task. It is important to emphasize that the knowledge base only contains documents converted from the training portion of the data and excludes all models belonging to the test set. Since our split is performed by process model, the knowledge base never contains an alternative execution of any process that is used for testing, thereby guaranteeing clean evaluation and eliminating model-level leakage. As shown in the "Document" column of Table 1, the remaining data for each task was stored in the RAG knowledge base. Embeddings for this knowledge base were generated using the "text-embedding-ada-002" model, ensuring robust and contextually relevant data representation.

To improve vector indexing accuracy, each unique case was segmented into distinct trunks. This approach preserves data integrity within each case, allowing RAG to retrieve more precise and relevant information during data retrieval, while reducing redundancy. These document embeddings were then indexed using a vector store tool, enabling RAG to efficiently retrieve the most pertinent context for each task. By dynamically extracting key information from the documents (such as process structures and event sequences), RAG provides each agent with the necessary semantic background to enhance task performance.

### 5.3. Model Selection and Agent Testing

To evaluate the framework's performance across different tasks, we selected four models to serve as agents and tested their accuracy. The models chosen for this experiment were Mistral-7B-Instruct, Meta-Llama-3-8B, GPT-3.5, and GPT-4o. Each model was tested on all tasks to provide a comparative analysis of their performance in semantic anomaly detection and prediction. During each test, all LLMs were required to answer the same set of questions, ensuring an accurate assessment of the relative differences among models. Each agent independently performed semantic analysis, anomaly detection, or next-activity prediction based on the task requirements, utilizing context provided by the RAG module.

To ensure a controlled and reproducible comparison, all LLMs were executed under the same inference configuration. We set the temperature to 0.1 and top-p to 0.95 with an input window of 8192 tokens and a maximum output length of 512 tokens. Greedy decoding was additionally tested as a controlled ablation, and the performance differences did not affect the overall trend. No task-specific prompting adjustments or sampling variations were applied between models of different sizes, preventing uncontrolled generation bias during evaluation.

### 5.4. Performance Evaluation

The overall effectiveness of the framework was assessed by comparing each model's performance on the test set, with the macro-F1 score of predictions in the T_SAD, A_SAD, and S_NAP tasks serving as the primary evaluation metric. For S_NAP, a prediction is considered correct only when the next activity generated by the model exactly matches the ground truth. To support reasoning without leaking evaluation data, the RAG module retrieves the Top-5 most relevant evidences from each task's independent knowledge base by combining semantic similarity with feedback-based relevance scores. Since the three tasks maintain separate knowledge bases, this process yields 15 retrieved evidences in total for each test query, ensuring that the model reasons over relevant knowledge while maintaining strict separation from the test set. During this process, feedback is collected after each model response as binary positive or negative signals, which are used to adjust the preference weight of the corresponding evidence vectors. Specifically, a positive rating increases the vector's weight by $+0.03$, while a negative rating decreases it by $-0.03$. Exposure propensities are estimated from the display positions of retrieved evidences, following the same bias-correction model used in our reranking pipeline. The number of interactions per query matches the retrieval setting, and no additional refinement loops are executed during evaluation. Beyond accuracy, we also recorded both the retrieved evidences and the explanations generated by each LLM during testing. This information allows for a deeper examination of each model's interpretability and provides insights into their decision-making behavior in anomaly detection and next-activity prediction.

Through this experimental setup, we aim to comprehensively assess the adaptability, reasoning capability, and semantic robustness of the framework across all three process-mining tasks.

## 6. Result and Discussion

### *6.1. Model Comparison and the Added Value of RAG over ICL*

Tables 2–4 present a performance comparison across different models on the S_NAP, T_SAD, and A_SAD tasks. We analyze the results from two perspectives: (1) the performance differences across models for each task, and (2) the comparative impact of in-context learning (ICL) versus our full adaptive framework.

Comparing model sizes, we observe that the GPT-series models (GPT-3.5 and GPT-4o) have significantly larger parameter counts than Mistral and Llama. However, this advantage does not translate uniformly across all tasks. While our framework used GPT-4o achieves the highest F1 score (0.81) on the S_NAP task, the improvements on T_SAD and A_SAD are more modest. Specifically, the gains on T_SAD and A_SAD for our framework used GPT-4o are 15.6% and 7.0%, respectively, only slightly higher than those of smaller models such as Llama.

**Table 2.** S_NAP Performance Comparison.

| Foundation Model | ICL | Ours | Improved |
|---|---|---|---|
| Mistral | 0.18 | 0.25 | +38.9% |
| Llama | 0.32 | 0.44 | +37.5% |
| GPT 3.5 | 0.43 | 0.76 | +76.7% |
| GPT 4o | 0.66 | 0.81 | +22.7% |
| **Overall** | 0.40 | 0.57 | **+44.0%** |

**Table 3.** T_SAD Performance Comparison.

| Foundation Model | ICL | Ours | Improved |
|---|---|---|---|
| Mistral | 0.49 | 0.53 | +8.2% |
| Llama | 0.51 | 0.60 | +17.6% |
| GPT 3.5 | 0.50 | 0.55 | +10.0% |
| GPT 4o | 0.53 | 0.67 | +26.4% |
| **Overall** | 0.51 | 0.59 | **+15.6%** |

**Table 4.** A_SAD Performance Comparison.

| Foundation Model | ICL | Ours | Improved |
|---|---|---|---|
| Mistral | 0.44 | 0.56 | +27.3% |
| Llama | 0.53 | 0.52 | −1.9% |
| GPT 3.5 | 0.45 | 0.47 | +4.4% |
| GPT 4o | 0.68 | 0.67 | −1.5% |
| **Overall** | 0.53 | 0.56 | **+7.1%** |

This discrepancy reflects differences in task complexity. S_NAP requires deeper semantic reasoning over activity dependencies, and large models like GPT-4o are better equipped to exploit this structure, resulting in a substantial improvement on S_NAP. In contrast, T_SAD and A_SAD primarily involve detecting semantic inconsistencies at the trace and activity level, where contextual dependencies are shallower and smaller models can perform competitively.

When comparing ICL with our full adaptive framework, we see consistent improvements across all three tasks, most notably on S_NAP. For instance, our framework used GPT-4o yields a 22.7% improvement in F1 over the ICL-only version (0.66 to 0.81). Although gains on T_SAD and A_SAD are smaller, this aligns with their lower semantic complexity.

These results confirm that our framework significantly strengthens tasks requiring complex semantic inference, while still providing measurable benefits for anomaly detection.

Overall, the findings demonstrate that the full adaptive framework consistently enhances predictive capability and foresight. On semantically demanding tasks such as S_NAP, the combination of multi-agent reasoning, retrieval, and online adaptivity yields large gains. On T_SAD and A_SAD, smaller models with our full framework already achieve strong performance, reinforcing the link between task complexity, model size, and the impact of adaptive retrieval. This indicates that integrating adaptive mechanisms with LLMs offers both robustness and flexibility in practical process mining applications.

To evaluate online adaptivity, we further conduct a streaming experiment with controlled concept drift. Test instances are sequentially fed in temporal order, and two semantic drifts are injected to emulate process-rule shifts that frequently occur in real-world operations. Figure 3 reports the macro-F1 over streaming queries, contrasting two system variants: the red dashed line represents the pre-feedback baseline without any index updates, while the blue solid line denotes our feedback-adaptive system that performs online index calibration after each drift. Following the first and second drift events, the red curve collapses and fails to recover, indicating an inability to adapt to evolving semantics. In contrast, the blue curve rapidly regains its performance within the adaptation windows, demonstrating that our feedback-calibrated retrieval mechanism effectively restores accuracy and maintains robustness in dynamic, non-stationary environments.

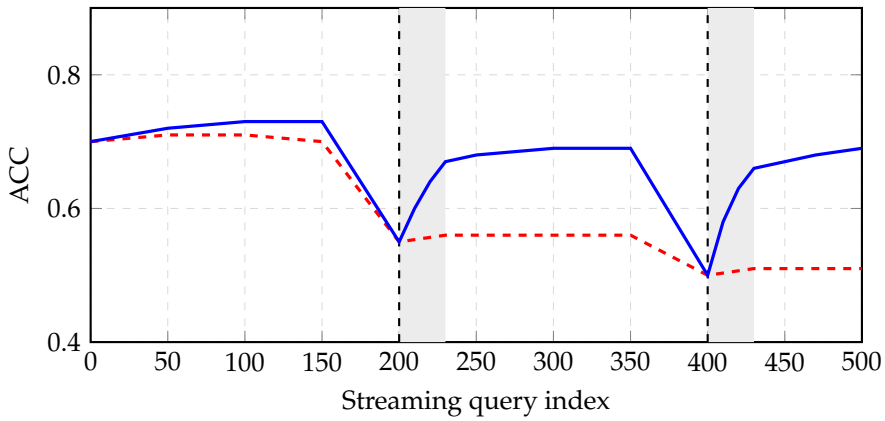


**Figure 3.** Streaming evaluation with two controlled semantic drifts. Dashed vertical lines mark drift points, and shaded regions indicate adaptation windows. The red dashed curve corresponds to the baseline (pre-feedback) system; the blue solid curve corresponds to the online feedback-adapted system.

### *6.2. Baseline Comparison*

Across the three tasks, our method achieves the strongest overall accuracy, as shown in Tables 5–7. The performance trends observed under varying parameters are consistent with the conclusions drawn in the main baseline comparison: we close the gap to correction-based frameworks while ultimately surpassing them once user feedback is incorporated into the retriever via index priors and cross-task corroboration. The largest gains remain on activity-level anomaly detection (A_SAD), where feedback-calibrated retrieval provides the most leverage when distinguishing among highly similar contexts. On trace-level detection (T_SAD), improvements are steadier rather than dramatic, consistent with CRAG's known strength on low-confidence cases; using a confidence-aware gate together with online priors retains that robustness while avoiding the coverage loss typical of correction-triggered backoff strategies. For ActiveRAG, we again observe that role specialization improves stability, but without shared priors it fails to reuse validated evidence across tasks and falls behind our approach.

Beyond the static baseline comparison, the three parameter studies provide finer-grained insights into where feedback helps most. Varying the top-$K$ (Table 5) shows that $K = 5$ yields the best balance between contextual breadth and retrieval precision: smaller $K$ under-exposes relevant corroborating events, while larger $K$ admits distractors that erode anomaly discrimination. Window size analysis (Table 6) indicates that enlarging the evidence horizon helps up to a point, with 4096 and 8192 performing comparably—suggesting diminishing returns once task-relevant evidence saturates the context window. Finally, the temperature sweep (Table 7) shows that a moderate value (0.1) produces the highest overall accuracy by encouraging lightweight exploration; extremely deterministic sampling (0.0) weakens cross-task corroboration, while higher entropy (0.2) injects noise that degrades precision. Across all three studies, our method remains the top performer under every parameter setting, whereas CRAG consistently ranks second: this reinforces that feedback-aligned priors complement, rather than replace, confidence-aware correction—recovering robustness while extending coverage.

**Table 5.** Performance vs. Top-$K$. Other parameters fixed: Window = 8192, Temperature = 0.1.

| Method | S_NAP (↑) | | | T_SAD (↑) | | | A_SAD (↑) | | |
|---|---|---|---|---|---|---|---|---|---|
| | **3** | **5** | **10** | **3** | **5** | **10** | **3** | **5** | **10** |
| Vanilla RAG [29] | 0.64 | 0.68 | 0.67 | 0.49 | 0.52 | 0.51 | 0.43 | 0.45 | 0.44 |
| RankRAG [15] | 0.65 | 0.69 | 0.68 | 0.56 | 0.59 | 0.58 | 0.44 | 0.46 | 0.45 |
| ActiveRAG [23] | 0.68 | 0.72 | 0.71 | 0.60 | 0.63 | 0.62 | 0.55 | 0.58 | 0.57 |
| CRAG [14] | 0.69 | 0.73 | 0.72 | 0.62 | 0.66 | 0.65 | 0.58 | 0.62 | 0.61 |
| **Ours** | **0.76** | **0.81** | **0.80** | **0.63** | **0.67** | **0.66** | **0.62** | **0.67** | **0.66** |

**Table 6.** Performance vs. Window size. Other parameters fixed: Top-$K$ = 5, Temperature = 0.1.

| Method | S_NAP (↑) | | | T_SAD (↑) | | | A_SAD (↑) | | |
|---|---|---|---|---|---|---|---|---|---|
| | **2048** | **4096** | **8192** | **2048** | **4096** | **8192** | **2048** | **4096** | **8192** |
| Vanilla RAG [29] | 0.65 | 0.68 | 0.68 | 0.50 | 0.52 | 0.52 | 0.43 | 0.45 | 0.45 |
| RankRAG [15] | 0.66 | 0.69 | 0.69 | 0.56 | 0.59 | 0.59 | 0.44 | 0.46 | 0.46 |
| ActiveRAG [23] | 0.69 | 0.72 | 0.72 | 0.60 | 0.63 | 0.63 | 0.56 | 0.58 | 0.58 |
| CRAG [14] | 0.70 | 0.73 | 0.73 | 0.63 | 0.66 | 0.66 | 0.59 | 0.62 | 0.62 |
| **Ours** | **0.77** | **0.81** | **0.81** | **0.64** | **0.67** | **0.67** | **0.63** | **0.67** | **0.67** |

**Table 7.** Performance vs. Temperature. Other parameters fixed: Top-$K$ = 5, Window = 8192.

| Method | S_NAP (↑) | | | T_SAD (↑) | | | A_SAD (↑) | | |
|---|---|---|---|---|---|---|---|---|---|
| | **0.0** | **0.1** | **0.2** | **0.0** | **0.1** | **0.2** | **0.0** | **0.1** | **0.2** |
| Vanilla RAG [29] | 0.67 | 0.68 | 0.66 | 0.51 | 0.52 | 0.50 | 0.44 | 0.45 | 0.43 |
| RankRAG [15] | 0.68 | 0.69 | 0.67 | 0.58 | 0.59 | 0.57 | 0.45 | 0.46 | 0.44 |
| ActiveRAG [23] | 0.71 | 0.72 | 0.70 | 0.62 | 0.63 | 0.61 | 0.57 | 0.58 | 0.56 |
| CRAG [14] | 0.72 | 0.73 | 0.71 | 0.65 | 0.66 | 0.64 | 0.61 | 0.62 | 0.60 |
| **Ours** | **0.80** | **0.81** | **0.79** | **0.66** | **0.67** | **0.65** | **0.66** | **0.67** | **0.65** |

### *6.3. Ablation Analysis*

In Table 8, removing index priors causes the sharpest degradation and largely reverts behavior toward vanilla retrieval, showing that "feedback→prior" is the main driver of next-query improvements and better initial rankings. Keeping only priors (and disabling reranker/retriever training) preserves immediate gains but plateaus below the full framework, confirming that periodic consolidation remains necessary. Disabling cross-task corroboration lowers all tasks—most visibly on next-activity prediction—highlighting the value of a single shared store when semantics overlap. Turning off the confidence gate reduces robustness to difficult or noisy inputs: accuracy drops on anomaly tasks even though

base retrieval remains unchanged, indicating that corrective triggers and failure-driven rewriting are complementary to the priors rather than redundant.

**Table 8.** Ablation study of our method (ADAM).

| Method | S_NAP (↑) | T_SAD (↑) | A_SAD (↑) |
|---|---|---|---|
| Full | **0.81** | **0.67** | **0.67** |
| w/o Index Priors | 0.67 | 0.51 | 0.45 |
| Priors-Only (no rerank/retriever) | 0.69 | 0.59 | 0.51 |
| w/o Cross-Task Corroboration | 0.72 | 0.61 | 0.57 |
| w/o Confidence Gate | 0.74 | 0.66 | 0.61 |

## 7. Limitations and Future Work

This study introduces a novel framework, though some limitations remain. Effective deployment requires more advanced LLMs with larger parameter counts to achieve optimal performance, which may challenge implementation in resource-limited environments. Additionally, the framework's reliance on accurate context generation by the RAG module introduces a dependency on context quality; imprecise retrieval can hinder decision accuracy across agents. Future work could focus on optimizing resource use through model distillation or lighter alternatives, and improving context quality with advanced retrieval techniques or knowledge graphs. Large-scale real-world validations will further assess the framework's scalability and applicability, supporting its refinement for practical deployment.

## 8. Conclusions

We introduced an online, feedback-adaptive multi-agent framework for semantics-aware process mining that operates over a single shared knowledge base and couples cross-task corroboration with retrieval calibrated by user feedback. The framework converts correctness/usefulness signals into index priors that take effect immediately at recall and initial ranking, and it applies a lightweight online correction loop with confidence gating and failure-driven query rewriting.

Across S_NAP, T_SAD, and A_SAD, the framework consistently outperforms retrieval-only and agent-only baselines while matching or exceeding correction-centric frameworks, with the most pronounced gains where fine-grained evidence discrimination is required. Ablations show that feedback-to-index priors are the primary driver of next-query improvements, cross-task corroboration contributes broadly across tasks, and confidence gating enhances robustness to difficult or noisy inputs. These results indicate that coupling shared evidence, online feedback, and targeted correction yields a stable and effective path to improving semantic process-mining tasks without heavy fine-tuning.

**Author Contributions:** Conceptualization, B.L. and X.S.; Methodology, X.S., B.L., Z.L., Y.D. and F.C.; formal analysis and investigation, Y.D., Z.L., F.C. and X.S.; Domain expert, J.W.; Writing—original draft, X.S.; Writing—review & editing, Z.L. and Y.D.; Supervision, B.L.; Funding acquisition, F.C. All authors have read and agree to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

**Conflicts of Interest:** Author Justin Wang is employed by the company The Property Investors Alliance Pty Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

## References


1. van der Aalst, W.M.P. *Process Mining: Data Science in Action*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2016. [CrossRef]
2. *IEEE Std 1849-2016*; IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams. IEEE: Piscataway, NJ, USA, 2016. [CrossRef]
3. Murata, T. Petri nets: Properties, analysis and applications. *Proc. IEEE* **1989**, *77*, 541–580. [CrossRef]
4. van der Aalst, W.M.P.; Carmona, J. (Eds.) *Process Mining Handbook*; Lecture Notes in Business Information Processing; Springer: Cham, Switzerland, 2022; Volume 448. [CrossRef]
5. *ISO/IEC 15909-1:2019*; Systems and Software Engineering—High-Level Petri Nets. ISO: Geneva, Switzerland, 2019.
6. Rebmann, A.; Schmidt, F.D.; Glavaš, G.; van der Aa, H. Evaluating the Ability of LLMs to Solve Semantics-Aware Process Mining Tasks. *arXiv* **2024**, arXiv:2407.02310. [CrossRef]
7. Rebmann, A.; Schmidt, F.D.; Glavaš, G.; van der Aa, H. On the Potential of Large Language Models to Solve Semantics-Aware Process Mining Tasks. *Process Sci.* **2025**, *2*, 10. [CrossRef]
8. Pyrih, V.; Rebmann, A.; van der Aa, H. LLMs that Understand Processes: Instruction-tuning for Semantics-Aware Process Mining. *arXiv* **2025**, arXiv:2508.16270.
9. Li, K.; Tang, Z.; Liang, S.; Li, Z.; Liang, B. MLAD: A Multi-Task Learning Framework for Anomaly Detection. *Sensors* **2025**, *25*, 4115. [CrossRef]
10. Berti, A.; Qafari, M.S. Leveraging Large Language Models (LLMs) for Process Mining. *arXiv* **2023**, arXiv:2307.12701. [CrossRef]
11. Berti, A.; Kourani, H.; Hafke, H.; Li, C.Y.; Schuster, D. PM-LLM-Benchmark: Evaluating Large Language Models on Process Mining Tasks. In Proceedings of the International Conference on Process Mining Workshops (GENAI4PM 2024), Lyngby, Denmark, 14–18 October 2024; Springer: Berlin/Heidelberg, Germany, 2024.
12. Asai, A.; Wu, Z.; Wang, Y.; Sil, A.; Hajishirzi, H. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. *arXiv* **2023**, arXiv:2310.11511. [CrossRef]
13. Guțu, B.M.; Popescu, N. Exploring Data Analysis Methods in Generative Models: From Fine-Tuning to RAG Implementation. *Computers* **2024**, *13*, 327. [CrossRef]
14. Yan, S.; Gu, J.; Zhu, Y.; Ling, Z. Corrective Retrieval Augmented Generation. *arXiv* **2024**, arXiv:2401.15884. [CrossRef]
15. Ma, Y.; Su, Y.; Vulić, I.; Korhonen, A. RankRAG: Unifying Context Ranking with Retrieval-Augmented Generation in LLMs. *arXiv* **2024**, arXiv:2407.0248.
16. Bai, J.; Miao, Y.; Chen, L.; Wang, D.; Li, D.; Ren, Y.; Xie, H.; Yang, C.; Cai, X. Pistis-RAG: Enhancing Retrieval-Augmented Generation with Human Feedback. *arXiv* **2024**, arXiv:2407.00072.
17. Liu, Y.; Hu, X.; Zhang, S.; Chen, J.; Wu, F.; Wu, F. Fine-Grained Guidance for Retrievers: Leveraging LLMs' Feedback in Retrieval-Augmented Generation. *arXiv* **2024**, arXiv:2411.03957.
18. Lin, X.V.; Chen, X.; Chen, M.; Shi, W.; Lomeli, M.; James, R.; Rodriguez, P.; Kahn, J.; Szilvasy, G.; Lewis, M.; et al. RA-DIT: Retrieval-Augmented Dual Instruction Tuning. In Proceedings of the ICLR, Vienna, Austria, 7–11 May 2024.
19. Quinn, D.; Nouri, M.; Patel, N.; Salihu, J.; Salemi, A.; Lee, S.; Zamani, H.; Alian, M. Accelerating Retrieval-Augmented Generation. *arXiv* **2024**, arXiv:2412.15246.
20. Cheng, M.; Luo, Y.; Ouyang, J.; Liu, Q.; Liu, H.; Li, L.; Yu, S.; Zhang, B.; Cao, J.; Ma, J.; et al. A survey on knowledge-oriented retrieval-augmented generation. *arXiv* **2025**, arXiv:2503.10677.
21. Chang, C.Y.; Jiang, Z.; Rakesh, V.; Pan, M.; Yeh, C.C.M.; Wang, G.; Hu, M.; Xu, Z.; Zheng, Y.; Das, M.; et al. MAIN-RAG: Multi-Agent Filtering Retrieval-Augmented Generation. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Long Papers), Vienna, Austria, 27 July–1 August 2025. [CrossRef]
22. Jiang, Z.; Xu, F.F.; Gao, L.; Sun, Z.; Liu, Q.; Dwivedi-Yu, J.; Yang, Y.; Callan, J.; Neubig, G. Active Retrieval Augmented Generation. *arXiv* **2023**, arXiv:2305.06983. [CrossRef]
23. Xu, Z.; Liu, Z.; Yan, Y.; Wang, S.; Yu, S.; Zeng, Z.; Xiao, C.; Liu, Z.; Yu, G.; Xiong, C. ActiveRAG: Autonomously Knowledge Assimilation and Accommodation through Retrieval-Augmented Agents. *arXiv* **2024**, arXiv:2402.13547.
24. Peng, B.; Zhu, Y.; Liu, Y.; Bo, X.; Shi, H.; Hong, C.; Zhang, Y.; Tang, S. Graph Retrieval-Augmented Generation: A Survey. *arXiv* **2024**, arXiv:2408.08921. [CrossRef]
25. Zhu, X.; Xie, Y.; Liu, Y.; Li, Y.; Hu, W. Knowledge Graph-Guided Retrieval Augmented Generation. *arXiv* **2025**, arXiv:2502.06864. [CrossRef]
26. Osipjan, A.; Khorashadizadeh, H.; Kessel, A.L.; Groppe, S.; Groppe, J. GraphTrace: A Modular Retrieval Framework Combining Knowledge Graphs and Large Language Models for Multi-Hop Question Answering. *Computers* **2025**, *14*, 382. [CrossRef]
27. Tsampos, I.; Marakakis, E. DietQA: A Comprehensive Framework for Personalized Multi-Diet Recipe Retrieval Using Knowledge Graphs, Retrieval-Augmented Generation, and Large Language Models. *Computers* **2025**, *14*, 412. [CrossRef]

28. Singh, A.; Ehtesham, A.; Kumar, S.; Khoei, T.T. Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG. *arXiv* **2025**, arXiv:2501.09136. [CrossRef]
29. Wang, Y.; Hernandez, A.G.; Kyslyi, R.; Kersting, N. Evaluating quality of answers for retrieval-augmented generation: A strong llm is all you need. *arXiv* **2024**, arXiv:2406.18064. [CrossRef]