
MEMENTO: Leveraging Web as a Learning Signal for Low-Data Domains

Ashutosh Ojha Vinay Aggarwal Ashutosh Srivastava
Siddharth Yedlapati Yaman K Singla Jitendra Ajmera



Adobe, Media & Data Science Research

behavior-in-the-wild@googlegroups.com

Abstract

Real-world tasks often lack large labeled datasets, motivating extensive work on learning in low-data regimes. However, existing approaches such as few-shot prompting, instruction tuning, and synthetic data generation, continue to treat labeled or pseudo-labeled data as the primary learning signal. In contrast, human practitioners acquire expertise through repeated, self-directed interaction with the open web, progressively refining both domain knowledge and search strategies. We propose **MEMENTO**, a framework that treats the web as a learning signal rather than a stateless retrieval interface. MEMENTO operates at two levels: within each session, it conducts iterative web exploration via an *Adaptive Exploration Tree* (AET) that decomposes tasks into evolving questions and reflects on intermediate findings; across sessions, it accumulates experience through dual-channel memory, separating *declarative knowledge* (facts) from *procedural knowledge* (search strategies). This design enables agents to learn reusable research strategies and domain expertise from trajectories of web interaction without additional model training. We evaluate MEMENTO on two low-data professional domains: sales automation and legal research. Our empirical results show consistent improvements in performance over ReAct based baselines (+25.6% on sales automation and 36.5% on legal research), demonstrating that the web can serve as a scalable learning source for acquiring task-specific expertise in data-scarce settings.

1 Introduction

Large labeled datasets are available for only a handful of well-studied tasks, whereas the majority of real-world tasks suffer from severe data scarcity. Domains such as legal research, sales strategy, and medical diagnosis are highly context-specific, and expert annotation at scale is infeasible Brown et al. [2020], Hedderich et al. [2021], Chalkidis et al. [2022], Rajpurkar et al. [2022]. This data scarcity has attracted sustained research attention on learning in low-data regimes.

Prior work has approached this problem primarily through better use of labeled data or model pretraining. T5 Raffel et al. [2020] showed that multi-task pretraining improves generalization under limited supervision. GPT-3 Brown et al. [2020] demonstrated few-shot adaptation from in-context examples. Instruction finetuning Wei et al. [2021] improved zero-shot generalization across diverse task collections, and synthetic data generation Wang et al. [2023], Gunasekar et al. [2023] extended adaptation to settings with minimal annotations. Despite these advances, all of these approaches treat labeled or pseudo-labeled data as the primary learning signal.

We propose the *open web* as an alternative learning signal for low-data settings. In domains where labeled supervision is scarce, human practitioners do not wait for thousands of annotated examples. They learn through repeated, self-directed engagement with web resources. A junior lawyer assigned a novel case consults legal databases, follows citation trails, and reads practitioner forums, gradually developing a sense of reliable sources and strategies. A sales representative entering a new market reads industry publications, studies competitor analyses, and internalizes which signals indicate

genuine buyer intent. The web is thus uniquely suited to low-data regimes: it is openly accessible, requires no annotation, and carries both factual content and implicit domain expertise.

Current LLM agents do not leverage this ability of the web. They treat web search as a stateless retrieval tool Nakano et al. [2021], OpenAI [2025a]: issuing queries, extracting facts, and discarding the experience accumulated along the way. Beyond direct factual content, the web contains expert-authored material such as practitioner forums, domain blogs, and methodology articles that encodes how experienced practitioners approach problem classes. Current agents have no mechanism to learn from such content across sessions, leaving the web’s value as a source of transferable domain expertise entirely untapped.

We introduce **MEMENTO**, a framework that uses the web as a learning signal for craft acquisition in low-data settings. MEMENTO operates at two levels. Within each session, MEMENTO conducts research via an **Adaptive Exploration Tree** (AET). Much like a lawyer who begins a novel case with a set of initial questions, collects relevant facts and evidence from multiple web sources, and progressively refines their queries based on what they find, the AET decomposes the research task into an initial set of questions and uses web search to answer them. The agent then reflects on all responses, identifies remaining gaps, and generates a new set of questions, repeating this cycle under a fixed budget. Information discovered at any point is stored in a local session memory, ensuring that each reflective cycle builds on what has already been found.

Across sessions, MEMENTO maintains two persistent and independent memory systems grounded in the declarative/procedural distinction from the integrated theory of mind Anderson et al. [2004]. A **Declarative Memory** stores verified facts and contextual knowledge from prior sessions, allowing future tasks to build on established foundations rather than starting from scratch. A **Procedural Memory** distills high-utility query strategies, source credibility heuristics, and domain-specific research playbooks from prior trajectories. Together, these components form an iterative learning pipeline: cross-session memory loads at the start of each session to initialize the AET with accumulated strategy and knowledge; the AET generates a rich trajectory through directed, reflective search; and that trajectory updates cross-session memory at the session’s end. Just as a lawyer accumulates case-specific precedents and generalizable research strategies that inform how to approach similar cases in the future, MEMENTO separately tracks what has been learned and how to approach unseen problems of the same kind. MEMENTO mirrors how human practitioners leverage the web in low-data settings: consulting authoritative sources to gather relevant facts, learning from expert-authored content to develop research strategies, and refining both knowledge and approach across repeated engagements.

We evaluate MEMENTO on two low-data professional domains: **Sales Automation**, which requires inferring buyer intent from indirect behavioral signals, and **Legal Research**, which requires identifying and citing authoritative legal precedent. MEMENTO yields consistent improvements in answer quality and search efficiency across both domains. The gains across two domains with distinct learning demands suggest that the improvements stem from the cross-session learning architecture rather than domain-specific adaptation. We make the following contributions:

- **Web as a Learning Signal in Low-Data Regimes:** To the best of our knowledge, we are the first to propose the open web as a learning signal for craft acquisition in domains where labeled supervision is scarce. MEMENTO uses research trajectories and accumulated factual knowledge to perform cross-session optimization, enabling agents to develop domain expertise without model fine-tuning.
- **Adaptive Exploration Tree:** We introduce a within-session search architecture that decomposes research tasks into dynamic question trees, iteratively reflecting on discovered information to identify gaps and restructure search paths under a fixed budget, producing trajectories that are directed, self-correcting, and informative as cross-session learning signals.
- **Dual-Channel Cross-Session Memory:** Grounded in the declarative/procedural distinction from the integrated theory of mind Anderson et al. [2004], MEMENTO maintains two independent persistent memory stores. Procedural Memory distills query strategies, source credibility heuristics, and domain-specific search playbooks from prior trajectories; Declarative Memory persists verified facts and contextual knowledge across sessions, allowing research strategy and domain knowledge to develop independently.
- **Empirical Validation Across Two Low-Data Domains:** We evaluate MEMENTO on Sales Automation and Legal Research, two low-data professional domains. MEMENTO demonstrates

consistent improvements in performance over ReAct-based baselines (+25.6% on sales automation and 36.5% on legal research), providing evidence that the web can serve as a learning signal for craft acquisition in low data settings.

2 Related Work

Learning in Low-Data Regimes. Few-shot learning Brown et al. [2020], instruction finetuning Wei et al. [2021], in-context learning Dong et al. [2024], and synthetic data generation Wang et al. [2023], Gunasekar et al. [2023] have each pushed the boundary of what is possible with scarce supervision. Despite their diversity, all share a common assumption: learning is bounded by a fixed training signal, whether labeled examples, instructions, or synthetic data. These approaches have no mechanism to actively seek out new information when supervision is scarce. MEMENTO addresses this directly by treating the open web as a cross-session learning signal, using research trajectories to accumulate both factual knowledge and search strategies across sessions, without requiring any labeled supervision or model fine-tuning.

Web as a Tool for LLMs. A growing body of work equips LLMs with web access at varying levels of sophistication. RAG Lewis et al. [2020] and its extensions Trivedi et al. [2023], Schick et al. [2023], Fan et al. [2024] ground LLM outputs in externally retrieved context. WebGPT Nakano et al. [2021] goes further by training a model to actively browse and cite web sources. Deep Research agents OpenAI [2025a], Google [2024], Perplexity AI [2025], xAI [2025] extend this further still with dynamic task planning and adaptive multi-step web interaction. Despite their sophistication, all of these systems are episodic: each session begins from the same baseline, with no mechanism to retain knowledge or strategy across sessions. Consequently, agents must rediscover relevant domain facts from scratch, cannot improve search strategy from prior experience, and do not become more efficient over repeated engagement with the same domain. MEMENTO addresses all three through its dual-channel memory architecture: Declarative Memory accumulates domain facts across sessions, and Procedural Memory distills effective search strategies from prior trajectories.

Within-Session Search Architectures. ReAct Yao et al. [2022] interleaves reasoning and action in linear sequences, while Tree of Thoughts Yao et al. [2023] and related approaches introduce branching over candidate reasoning paths. These methods improve over single-pass retrieval but remain limited in their ability to dynamically restructure search based on what is discovered mid-session. MEMENTO’s Adaptive Exploration Tree (AET) addresses this by decomposing tasks into dynamic question trees that continuously reflect on accumulated findings, identify remaining gaps, and restructure search paths under a fixed budget. Unlike static branching approaches, the AET treats mid-session discoveries as explicit signals for restructuring subsequent search rather than as passive context, producing trajectories that are directed and self-correcting rather than exhaustive.

RL and Memory-Augmented Agents. A growing body of work applies reinforcement learning to train LLM agents for retrieval tasks Wei et al. [2025], Chen et al. [2025], Song et al. [2025], Jin et al. [2025], with systems like MemoryR1 Yan et al. [2025] and SE-Search Li et al. [2026] augmenting these agents with explicit memory modules to improve persistence across interactions. These methods demonstrate that learned retrieval policies and memory persistence both improve performance. However, they face three limitations that make them poorly suited to low-data settings. First, RL training requires tens of thousands of rollouts, which is infeasible when labeled data is scarce. Second, weight updates mean every new domain requires retraining, which is impractical at deployment. And third, memory is treated as a monolithic store, collapsing the distinction between **what** was found and **how** it was found. Without this separation, research strategy remains frozen at initialization even as factual memory grows. MEMENTO addresses all three limitations: it requires no weight updates, operates effectively from as few as 60 training examples, and bifurcates memory into independent procedural and declarative channels, allowing craft and knowledge to accumulate and evolve independently.

Learning from Data at the Prompt Level. Recent works including GEPA Agrawal et al. [2025] and MIPROv2 Opsahl-Ong et al. [2024] optimize LLM behavior at the prompt level without modifying weights. Agrawal et al. [2025] reflects on execution trajectories in natural language to iteratively evolve system prompts, while Opsahl-Ong et al. [2024] jointly tunes instructions and few-shot demonstrations via Bayesian optimization. However, both are fundamentally dataset-driven: the learning signal comes entirely from labeled examples, meaning the quality of the learned artifact is

bounded by the coverage and size of the training set. MEMENTO takes a different approach entirely; rather than learning from a fixed labeled dataset, it uses the open web as its primary training signal, allowing craft knowledge and domain expertise to be acquired primarily from web interaction rather than from labeled examples.

3 Methodology

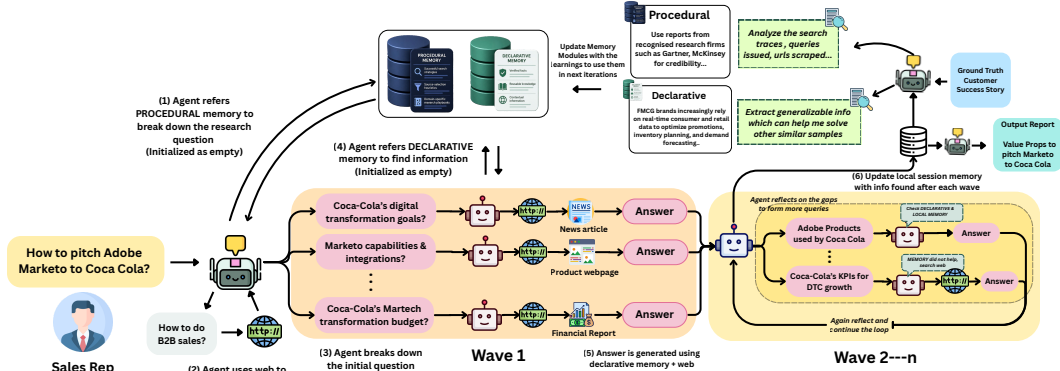


Figure 1: Overview of MEMENTO for a single training sample. Given a research question, the agent (1) consults procedural memory for question decomposition or (2) optionally explores the web to bootstrap craft knowledge when memory is empty. (3) The root is decomposed into Wave 1 sub-questions, each question is solved by a tool-augmented agent that (4) checks declarative memory before issuing date-restricted web searches and (5) returns an answer. In subsequent waves, reflection identifies gaps and adds new sub-questions, prioritizing memory over web retrieval. (6) Local session memory is updated after each wave; at the end of the sample, the trajectory is distilled into procedural rules and declarative facts that update the persistent memory stores.

3.1 Problem Formulation

We consider tasks of the form: given an input context x (a question, case, scenario, or specification), produce an output $\hat{y}$ (a prediction, analysis, or generated artifact) that can be evaluated against a ground truth y . We assume a training corpus

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$$

of examples, and a held-out test set $\mathcal{D}_{\text{test}}$. The agent has access to a web search tool with a date cutoff enforced at both training and inference time, preventing it from retrieving information about the outcome it is trying to predict. The goal is not to fine-tune model weights but to learn a set of transferable reasoning artifacts, namely procedural strategies and declarative facts, that accumulate across training examples and improve the agent’s performance and efficiency on future unseen inputs.

3.2 System Overview

Figure 1 provides an overview of the MEMENTO framework. MEMENTO operates in two interleaved phases that repeat over batches of training examples. In the **forward pass**, each input x is processed by the Adaptive Exploration Tree (AET), a hierarchical decomposition and search architecture that breaks the task into sub-questions, answers each through web interaction, and synthesizes a final output $\hat{y}$. The AET is initialized at the start of each session with the current state of two persistent memory stores, Procedural Memory and Declarative Memory, which provide accumulated search strategies and domain knowledge from prior batches. In the **batch update**, after all examples in a batch are processed, the system reflects on execution trajectories and ground-truth outcomes to update both memory stores, which are then carried forward to the next batch. Together, these two phases form a closed learning loop: the forward pass generates rich research trajectories that the batch update distills into reusable expertise, and that expertise in turn improves future forward passes. At inference time, only the forward pass runs with the final learned memory state frozen and no further updates occur.

3.3 Within-Session Research: The Adaptive Exploration Tree

Rather than asking the LLM to directly produce $\hat{y}$ from x , MEMENTO frames x as a root question and decomposes it into a tree of sub-questions $Q = \{q_0, q_1, \dots, q_k\}$, where q_0 is the root. The Adaptive Exploration Tree (AET) governs within-session research through three tightly coupled components: wave-based decomposition, which builds the tree adaptively; tool-augmented agents, which resolve individual sub-questions; and bottom-up synthesis, which aggregates answers into a final output. Throughout this process, a local session memory accumulates findings incrementally, ensuring that every component has access to what has already been discovered.

Local session memory. The AET maintains a local session memory that grows as sub-questions are resolved. Each solved sub-question contributes its answer and supporting evidence to this store. This memory is available to all components of the AET: the reflection agent consults it when generating new waves of sub-questions, and individual tool-augmented agents query it before issuing external web searches. Once the forward pass is complete, the local session memory contains a full record of the agent’s research trajectory for that sample, which is used at the end of each training batch to update the persistent cross-session memory stores described in Section 3.4.

Wave-based decomposition. The AET decomposes x in waves. In the first wave, the root question is broken into an initial set of sub-questions up to a configurable budget B_0 . Between waves, a reflection agent is invoked: it receives the answers generated so far, any intermediate findings, and the history of prior reflection decisions, and produces a new set of questions for the next wave. These questions are not predetermined; they are conditioned on what has already been learned. If an answer reveals an unanticipated dependency, an unresolved ambiguity, or a previously overlooked dimension, the reflection agent introduces targeted sub-questions to address those gaps. This process repeats until the overall question budget is exhausted. Because each wave is informed by prior results, the tree is not statically defined at initialization but iteratively reshaped in response to accumulating evidence.

Sub-question solving: tool-augmented agents. Each sub-question q_j is assigned to an independent tool-augmented agent with four primitive actions. `SEARCH_MEMORY` retrieves relevant entries from both the filtered long-term declarative memory store and the local session memory, thereby prioritizing internally available knowledge before incurring external retrieval costs. `SEARCH_WEB` issues a web query and returns a ranked list of results. `SCRAPE_RESULTS` fetches the full textual content of selected results, serving previously retrieved pages from an in-process cache to avoid redundant network requests. `GENERATE_ANSWER` synthesizes the accumulated evidence into a final response for the sub-question.

Bottom-up synthesis. Once all waves are complete, answers are synthesized upward through the tree. Only nodes that were further decomposed during reflection, and thus have children but no direct answer, require synthesis. Each such node receives the answers of its children and produces a synthesized answer via an LLM call, proceeding level by level from deepest to shallowest. The root node’s synthesized answer is the final output $\hat{y}$.

3.4 Persistent Memory Architecture

Across batches, MEMENTO maintains two memory systems grounded in the procedural-declarative distinction from the integrated theory of mind Anderson et al. [2004]: procedural memory captures knowledge of *how* to research, and declarative memory captures knowledge of *what* is known about the domain. The two systems are independent in scope but share a batch-level update cadence, described in Section 3.5.

Procedural Memory: three complementary stores. Procedural memory is split into three stores that target different stages of the AET pipeline.

Craft Knowledge (M_1) is a free-text store of high-level research strategy, which sub-questions tend to be informative, what sources are authoritative, how to reason under uncertainty etc. M_1 is learned *entirely from the web, without supervision*: each sample’s session scratchpad (a running synthesis of web evidence) is distilled by an LLM into generalizable insights, which are aggregated into M_1 at the end of every batch. No ground truth is consulted; the supervisory signal is the web evidence itself.

M_1 is consumed at the *planning* stage (by the wave-zero decomposer and the reflection agent) and is the only store that supports *cold-start enrichment*. Before any web search, the agent first runs a relevance filter over the existing M_1 to extract the craft knowledge that applies to the current sample; only if no sufficient matching knowledge is found does it issue exploratory web searches and distill the traces into new craft entries on the fly.

Decomposition Rules (M_2) is a structured catalog of conditional rewrite rules of the form “WHEN <pattern in root or sample inputs>, ADD a sub-question that COVERS <pattern>.” Whereas M_1 encodes strategic preferences in free-form prose, M_2 encodes hard, sample-conditioned compositional patterns. M_2 is consumed at the *decomposition* stage: rules are injected at every depth of the question-breakdown prompt, both at wave zero and during reflection.

Web Action Rules (M_3) is a structured catalog of rules governing the *tool-use* stage: query formulation (e.g., quoted-phrase length, language separation), URL selection (e.g., authority filters, *site:* qualifiers), and scraping constraints. M_3 is injected into the system prompt of every tool-augmented agent, so it shapes execution *inside* a sub-question solve rather than the structure of the tree itself.

The three stores form a hierarchy by stage and by representational form: M_1 is free-text strategy for the planner; M_2 is structured rules for the decomposer; M_3 is structured rules for the executor.

Declarative Memory: Declarative memory comprises a single store, the **Domain Store** (M_4): a structured free-text repository of entity-specific factual knowledge extracted from prior samples. At inference time, a retrieval step retrieves the current sample’s declarative cache against M_4 ’s keys and injects the matching subset into the solver as always-visible context in the tool-augmented agent’s system prompt.

3.5 Cross-Session Learning: The Batch Update Loop

Phase A — Forward Pass and Unsupervised Consolidation. All b examples are processed in parallel using the current memory state $\mathcal{M} = \langle M_1, M_2, M_3, M_4 \rangle$. Post-inference, two unsupervised consolidation steps run without access to ground truth: (i) craft knowledge M_1 is rewritten by an LLM aggregating per-sample methodology learnings from all successful runs, and (ii) search rules M_3 are updated from search execution logs. Domain store M_4 is then updated by folding entity-specific facts extracted during the runs.

Phase B — Scoring. Each predicted output $\hat{y}_i$ is scored against its ground truth y_i^* using an LLM judge, producing a normalized score per sample. Failed runs are excluded from the subsequent supervised phase.

Phase C — Supervised Reflection. For each successfully scored run, the system compares predicted outputs against ground truth at a fine-grained level. For each ground truth value proposition, the system inspects the question decomposition tree to determine whether the corresponding sub-question was present and whether it led to correct coverage. Rules active in runs where a value proposition was successfully found accumulate *support*; rules present in runs where it was missed accumulate *contradiction*. An LLM then proposes rule additions, reinforcements, or removals, which are canonicalized and merged into M_2 weighted by their support-to-contradiction ratio across batches. The updated $\mathcal{M}$ is carried forward to the next batch.

At test time, the forward pass runs with frozen memory. No updates occur. All memory stores are injected as context, and the system produces $\hat{y}$ using the same pipeline as training. The learned knowledge remains interpretable as human-readable text.

4 Experimental Setup

4.1 Tasks and Datasets

Task 1: Sales Automation. Given a triple of (seller company, product list, prospective customer), the system predicts the value propositions a sales representative would use to pitch the given products to the given customer, each capturing a pain point and the mechanism by which the product addresses it. The agent receives only the input triple and must recover these propositions entirely through open-web research, mirroring the workflow of a Sales Development Representative synthesizing publicly

available information into an evidence-backed pitch. We use the SDR-Bench dataset Srivastava et al. [2026], consisting of publicly available customer success stories documenting prior deals between enterprises. The dataset comprises 180 seller-customer-product triples, each paired with a ground truth pitch comprising value propositions and supporting evidence. To prevent temporal leakage, web search is restricted to sources published more than six months before each story’s publication date. The samples are partitioned with seed 42 into 60 training and 120 test samples.

Task 2: Legal Research. Given the neutral facts of a U.S. Supreme Court case, the system predicts the judgment outcome: the winning party, the disposition of the court’s decision, and a reasoned conclusion. This mirrors the challenge facing legal researchers who must assess case outcomes from factual records alone, without access to prior rulings or judicial commentary on the case at hand. We use the JUSTICE benchmark Alali et al. [2021], a dataset of U.S. Supreme Court cases sourced from Oyez. We restrict to cases decided after 2010, yielding 722 candidates with sufficient web coverage. To prevent the agent from trivially retrieving the exact case record, we apply a leakage filtering step: for each candidate, a web agent is given the case facts with a search cutoff of two years prior to the decision date; cases where the agent successfully retrieves the exact outcome are discarded. Post this filtering, we take 180 samples partitioned with seed 42 into 60 training and 120 test samples. The two-year backdating is enforced consistently at both training and inference time.

4.2 Baselines

We evaluate MEMENTO against five baselines designed to isolate the contribution of each architectural component, including web access, the Adaptive Exploration Tree (AET), and cross-session memory. First, a **closed-book LLM**, where the language model answers directly from its parametric knowledge without any web access or memory; however, performance on some tasks may be inflated due to overlap between benchmark data and pretraining corpora. Second, we evaluate **in-context learning**, where the same base models are prompted with five labeled examples from the training set, representing a standard low-data adaptation baseline. Third, we include **ReAct with web search**, which augments the base models with a flat ReAct loop and web search access but no learned memory, establishing the benefit of web access alone. Fourth, we evaluate **ReAct with MEMENTO memory**, where the same flat ReAct loop is augmented with MEMENTO’s trained procedural and declarative memory stores injected at inference time, isolating the contribution of memory independently from the AET architecture. Finally, we consider **Adaptive Exploration Trees**, which does not use the cross session memory, isolating the contribution of the within-session search architecture from that of cross-session learning.

4.3 Evaluation Metrics

Sales Automation. Predictions are evaluated by an LLM judge prompted to act as a Senior Sales Enablement Evaluator. We extract the ground truth value propositions from the actual customer success story, (y) as well as from the report generated by MEMENTO, ($\hat{y}$). For each sample, we employ the judge to score if each ground truth value proposition is covered in the generated output or not. Each GT value prop is scored on a 0–5 sales-effectiveness scale, ranging from a complete miss (0) to full recovery of the product, pain point, mechanism, and supporting evidence (5). Full rubric details are provided in Appendix E. We report the following aggregate metric: **average coverage score**, the mean per-point score across the test set.

Legal Research. We evaluate as a binary win-prediction problem. For each case, the system outputs a predicted winning party (petitioner or respondent), which is scored against the ground-truth `winning_party` field from JUSTICE using an LLM judge returning a binary CORRECT/INCORRECT verdict. The reported metric is simple accuracy.

4.4 Implementation Details

All experiments are conducted by using two different backbone LLMs: Qwen Yang et al. [2025], an open source LLM and GPT-5-mini OpenAI [2025b], a frontier LLM, for cross-model validation. The AET operates under three configurable budgets: a question budget governing the total number of sub-questions per task, a max agent steps budget governing per-question search iterations, and an

exploration budget governing craft bootstrapping cycles. Training proceeds in batches of b examples. Full hyperparameter settings are provided in Appendix A.

	Sales Automation (Coverage Score)	Legal Research (Accuracy)
LLM (only parametric knowledge)		
Qwen	0.416	0.808
Qwen + MEMENTO	0.437 (+0.021)	0.808 (+0.000)
GPT-5-mini	0.391	0.700
GPT-5-mini + MEMENTO	0.398 (+0.007)	0.717 (+0.017)
In-context learning		
5-shot Qwen	0.375	0.842
5-shot Qwen + MEMENTO	0.382 (+0.007)	0.852 (+0.010)
5-shot GPT-5-mini	0.462	0.692
5-shot GPT-5-mini + MEMENTO	0.476 (+0.014)	0.675 (-0.017)
Web as a tool (ReAct)		
ReAct + Qwen	0.461	0.592
ReAct + Qwen + MEMENTO	0.469 (+0.008)	0.742 (+0.150)
ReAct + GPT-5-mini	0.522	0.767
ReAct + GPT-5-mini + MEMENTO	0.525 (+0.003)	0.825 (+0.058)
Learning from web		
AET + Qwen	0.552 (+0.091)	0.767 (+0.175)
MEMENTO + Qwen	0.579 (+0.119)	0.808 (+0.216)
AET + GPT-5-mini	0.532 (+0.010)	0.783 (+0.016)
MEMENTO + GPT-5-mini	0.547 (+0.025)	0.838 (+0.071)

Table 1: Comparison of framework performance across Sales Automation and Legal Research tasks. Green rows show MEMENTO memory augmentation; amber rows highlight AET-only results; blue rows mark the full MEMENTO system. Δ values for *Learning from web* rows are relative to the corresponding ReAct baseline; all others are relative to the row above.

We evaluate MEMENTO against four baselines: LLMs relying solely on parametric knowledge, in-context learning, and ReAct-style agents with web search access. We benchmark the full MEMENTO system and each architectural component in isolation across Sales Automation and Legal Research.

MEMENTO consistently outperforms all baselines within each backbone. On Sales Automation, MEMENTO + Qwen (0.579) improves over ReAct + Qwen (0.461) by 25.6%, and MEMENTO + GPT-5-mini (0.547) improves over ReAct + GPT-5-mini (0.522) by 4.8%. On Legal Research, MEMENTO + Qwen (0.808) improves over ReAct + Qwen (0.592) by 36.5% (+9.2% increase with GPT-5-mini), suggesting that the framework is well suited to leverage web as a learning signal for domains with labeled data scarcity. The gains hold across both model backbones indicating that improvements stem from the cross-session learning architecture rather than the choice of underlying model.

The AET contributes independently of cross-session memory. The largest jump in performance comes from replacing the flat ReAct loop with the Adaptive Exploration Tree. On Sales Automation, AET + Qwen (0.552) outperforms ReAct + Qwen (0.461) by 19.7%, and AET + GPT-5-mini (0.532) outperforms ReAct + GPT-5-mini (0.522) by 1.9%. On Legal Research, AET + Qwen (0.767) outperforms ReAct + Qwen (0.592) by 29.6%. This demonstrates that AET’s architecture comprising of directed, reflective within-session search drives substantial improvements over linear retrieval. Adding cross-session memory on top of the AET yields further gains: MEMENTO + Qwen improves over AET + Qwen by 4.9% on Sales Automation and 5.3% on Legal Research, demonstrating each component’s contribution to the overall performance.

Zero-shot and few-shot baselines show inflated performance due to likely data leakage. On Legal Research, Qwen with no web access (0.808) matches MEMENTO + Qwen despite having no retrieval mechanism, while ReAct + Qwen (0.592) scores substantially lower. This pattern is consistent across models and suggests that zero-shot and few-shot baselines benefit from parametric

knowledge of the evaluation datasets, which are publicly available and likely present in pretraining corpora. Our web search configurations enforce a strict temporal cutoff: the web agent is given the case facts with a search cutoff of two years prior to the year of decision for legal task and web search is restricted to sources published more than 6 months before the publication date of the customer success story for the sales task, preventing the model from retrieving evaluation data directly from the web. The models are also strictly instructed to use the information from web instead of relying on their world knowledge. Under these controlled conditions, MEMENTO’s gains over web-based baselines reflect genuine learning from web interaction rather than memorized answers (0.808 vs 0.592).

MEMENTO memory improves ReAct baselines across most configurations. Injecting MEMENTO’s trained memory stores into ReAct agents yields consistent improvements. The most substantial gain is on Legal Research, where ReAct + Qwen + Memory (0.742) improves over ReAct + Qwen (0.592) by 25.3%, and ReAct + GPT-5-mini + Memory (0.825) improves over ReAct + GPT-5-mini (0.767) by 7.6%. This isolates the contribution of cross-session memory independently of the AET, confirming that accumulated procedural and declarative knowledge transfers effectively to agents not trained with the full MEMENTO pipeline. The one exception is 5-shot GPT-5-mini + MEMENTO Memory on Legal Research (0.675 vs 0.692 without memory), which we attribute to interference between parametric knowledge of GPT-5-mini and memory-injected context, an interaction we leave for future investigation.

Efficiency Analysis. The learning behavior and component-level contribution of MEMENTO are analyzed in detail in Appendix C. We summarize the headline findings here. **Quality rises while cost falls.** On both tasks, training improves the score while simultaneously reducing compute: on Sales Automation under a 10-step budget per sub-question, the score rises from 0.552 to 0.579 (+4.8%) while search queries drop by 6.6% and repeated URL scraped by 26% (Appendix C). On Legal Research, training reduces search queries by 19.8%, LLM calls by 19.3%, and sub-questions per sample by 13.0% (Appendix H). **Procedural memory carries the gain.** An ablation isolating the four memory stores on the sales task (Appendix D) shows that the procedural stores (M_1, M_2, M_3) deliver +0.0225 of the total +0.0265 improvement (roughly 85%), while declarative memory (M_4) contributes a marginal +0.0007 in isolation and is weakly complementary when combined. We attribute this to the heterogeneous nature of the evaluation set that consists of customer success stories of over 30 different companies. The impact of procedural memory mirrors the cognitive science finding that novice-to-expert progression is driven primarily by production compilation Anderson et al. [2004]: knowing *how* to search matters more than *what* entity-level facts are pre-cached.

5 Conclusion

We propose treating the open web as a cross-session learning signal for professional task acquisition in low-data settings, a departure from prior work that is bounded by fixed labeled or pseudo-labeled data. MEMENTO enables this through directed within-session search and dual-channel persistent memory, enabling agents to accumulate both domain knowledge and research strategy across sessions without model fine-tuning. Empirical results on two structurally distinct domains show consistent improvements over strong baselines, with ablations confirming that within-session iterative research, procedural and declarative memory, each contribute independently to the gains. We hope this work opens a new direction for low-data learning: one where the open web serves not as a retrieval surface but as a source of transferable expertise.

Limitations and Future Work. Memory is frozen at inference time, limiting continued learning after deployment. The AET incurs higher computational cost during training than linear retrieval baselines, and performance is dependent on web coverage of the target domain. MEMENTO has been evaluated on two domains and two backbone models; broader generalization remains to be established. Future work should explore online memory updates, longer-horizon failure patterns in procedural memory, and multi-agent settings where procedural knowledge is shared across agents.

Broader Impact. MEMENTO enables high-quality research assistance in low-data domains without model fine-tuning, with memory stored as human-readable text rather than model weights, improving auditability and interpretability. However, the same capability could enable large-scale profiling, targeted outreach, or opposition research using public web data. In high-stakes domains such as law,

MEMENTO’s outputs should augment rather than replace human judgment. The system also inherits the demographic and linguistic biases present in web content, and responsible deployment requires further auditing.

References

- Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziemis, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.
- Mohammad Alali, Shaayan Syed, Mohammed Alsayed, Smit Patel, and Hemanth Bodala. Justice: A benchmark dataset for supreme court’s judgment prediction. *arXiv preprint arXiv:2112.03414*, 2021.
- John R Anderson, Daniel Bothell, Michael D Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. An integrated theory of the mind. *Psychological review*, 111(4):1036, 2004.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Katz, and Nikolaos Aletras. Lexglue: A benchmark dataset for legal language understanding in english. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4310–4330, 2022.
- Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, et al. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025.
- Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, et al. A survey on in-context learning. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 1107–1128, 2024.
- Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pages 6491–6501, 2024.
- Google. Try deep research and our new experimental model in gemini. <https://blog.google/products-and-platforms/products/gemini/google-gemini-deep-research/>, 2024. Accessed: 2026-04-23.
- Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.
- Michael A Hedderich, Lukas Lange, Heike Adel, Jannik Strötgen, and Dietrich Klakow. A survey on recent approaches for natural language processing in low-resource scenarios. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2545–2568, 2021.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.
- Jian Li, Yizhang Jin, Dongqi Liu, Hang Ding, Jiafu Wu, Dongsheng Chen, Yunhang Shen, Yulei Qin, Ying Tai, Chengjie Wang, et al. Se-search: Self-evolving search agent via memory and dense reward. *arXiv preprint arXiv:2603.03293*, 2026.

- Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.
- OpenAI. Introducing deep research. <https://openai.com/index/introducing-deep-research/>, 2025a. Accessed: 2026-04-23.
- OpenAI. Openai gpt-5 mini. <https://platform.openai.com/docs/models/gpt-5>, 2025b. Accessed: 2026-05-07.
- Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9340–9366, 2024.
- Perplexity AI. Introducing perplexity deep research. <https://www.perplexity.ai/hub/blog/introducing-perplexity-deep-research>, 2025. Accessed: 2026-04-23.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.
- Pranav Rajpurkar, Emma Chen, Oishi Banerjee, and Eric J Topol. Ai in health and medicine. *Nature medicine*, 28(1):31–38, 2022.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.
- Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.
- Ashutosh Srivastava, Siddharth Yedlapati, Vinay Aggarwal, Shashwat Dixit, and Yaman Kumar Singla. Sdr-bench: Benchmarking the personalization capabilities of large language models. <https://behavior-in-the-wild.github.io/SDR-Bench.html>, 2026.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 10014–10037, 2023.
- Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 13484–13508, 2023.
- Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.
- Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, et al. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7920–7939, 2025.
- xAI. Grok 3 beta — the age of reasoning agents. <https://x.ai/news/grok-3>, 2025. Accessed: 2026-04-23.
- Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z Pan, et al. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

A Hyperparameters

Both tasks share the same core configuration unless noted below. We train on 60 samples and evaluate on a held-out test set of 120 samples per task. The Adaptive Exploration Tree uses a *question budget* of 25 sub-questions per sample, of which 5 are produced in the initial wave and the remainder introduced over 2 reflection checkpoints (yielding 3 waves total). Each tool-augmented agent is capped at 10 ReAct steps. Cold-start exploration is enabled during training with a budget of 3 search-and-scrape cycles, triggered whenever the relevance-filtered slice of M_1 for the current sample is empty or sparse. The batch update loop processes each training set in 6 batches of size 10. Actor, critique, reflection, and synthesis calls run at temperature 0.7. Unless stated otherwise, all LLM calls and the LLM judge use Qwen-3.5-35B-A3B.

Web search restricts page recency to a 6-month look-back window relative to each sample’s publication date. At inference time the exploration budget is set to 0: memory is consumed but not extended.

Web search restricts page recency to a 5-year (60-month) look-back window relative to each case’s decision date, ensuring the agent cannot retrieve the ruling itself. To prevent parametric leakage of the case outcome, every input is passed through an LLM-based PII-removal step that strips party names and personal identifiers, and an explicit anti-cheat instruction is injected into the ReAct system prompt forbidding reliance on internal training knowledge of the case’s disposition. Unlike the sales task, the exploration budget at inference time is held at 3 so that under-trained legal domains can still gather missing context, though no artifacts are written back to long-term memory.

B Compute Resources

All Qwen-3.5-35B-A3B calls are served from a self-hosted SGLang deployment running across 4 nodes of 8×NVIDIA A100 80 GB GPUs (32 GPUs total), exposed through a least-inflight load-balancer. The GPT-5-mini variants are accessed through Azure OpenAI’s hosted endpoint and incur no local GPU compute. Web search and HTML scraping are handled by an in-house Crawl4AI-based service shared across all experiments; to stay within its capacity, concurrent access is gated by two bounded semaphores (8 simultaneous SEARCH_WEB queries, 4 simultaneous SCRAPE_RESULTS fetches), and a persistent global URL cache amortizes repeated retrievals across runs.

All runs use a thread pool of 20–30 concurrent workers. End-to-end wall times for the sales task are summarized in Table 2. Training one 60-sample run with Qwen takes approximately **6 hours**; inference over the 120-sample test set takes approximately **8 hours** per configuration. The GPT-5-mini variants exhibit comparable wall times despite offloading inference to Azure (training: ~6 h, test: ~7 h), as end-to-end latency is dominated by API round-trips under concurrent load rather than local compute. Because all experiments share the same web-service quota, wall-clock times depend on how many runs are active simultaneously.

Stage	Backend	Compute	Wall time
Training (60 samples)	Qwen-3.5-35B-A3B	4 × 8 × A100 80 GB	6 h
Training (60 samples)	gpt-5-mini (Azure)	API only	~7 h
Prediction (120 samples)	Qwen-3.5-35B-A3B	4 × 8 × A100 80 GB	6 h
Prediction (120 samples)	gpt-5-mini (Azure)	API only	~8 h

Table 2: Compute resources and wall-clock times for the sales-task experiments.

C Additional Sales Automation Results

Metric	max_agent_steps = 20 (a)		max_agent_steps = 10 (b)	
	After training	Without training	After training	Without training
Score	0.5864	0.5691	0.5791	0.5526
<i>Decomposition behaviour</i>				
Avg. Questions / Sample	21.18	22.78	22.13	23.52
Avg. Agent Steps / Question	9.80	10.93	7.80	8.71
Avg. Scratchpad Size (chars)	40,122	48,475	41,172	39,188
<i>Search discipline</i>				
Avg. Search Queries / Sample	174.00	201.15	139.61	149.49
Avg. Unique Google URLs	786.84	804.88	659.13	644.75
Avg. Repeated Google URLs	539.58	698.84	410.97	500.71
<i>Scraping behaviour</i>				
Avg. Unique URLs Scraped	47.76	47.04	37.38	36.55
Avg. Repeated URLs Scraped	28.96	37.08	19.02	25.69
<i>Compute</i>				
Avg. LLM Calls	398.21	441.01	310.79	323.97

Table 3: Test-set performance on 120 held-out samples, grouped by the pipeline stage each metric reflects.

C.1 Exploration Phase

To bootstrap M_1 before any supervised signal is available, the system runs a *pre-solve exploration phase* when craft knowledge is absent from memory (left panel of Figure 1). At the start of solving a sample, if M_1 is empty, an exploration agent issues a fixed number of search-and-scrape cycles to discover domain strategies and methodologies, populating M_1 with a foundational procedural prior before inference begins. If M_1 already contains knowledge, this step is skipped.

Additionally, at the end of solving each sample, if exploration budget remains and the sample produced a valid answer, the agent re-enters exploration to further enrich M_1 based on what was encountered during that run.

D Ablation: Procedural vs. Declarative Memory

To isolate the contribution of each memory channel, we evaluate four configurations on the 120 held-out sales samples with a budget of 10 agent steps per question: no memory (*neither*), declarative only (M_4), procedural only (M_1, M_2, M_3), and the full system. Table 4 reports the results.

Configuration	Score	Δ vs. neither
Neither (zero-shot)	0.5526	—
Declarative only (M_4)	0.5533	+0.0007
Procedural only (M_1, M_2, M_3)	0.5751	+0.0225
Both (full system)	0.5791	+0.0265

Table 4: Ablation scores on 120 held-out sales samples at max_agent_steps = 10.

Procedural memory accounts for the majority of the gain. Of the $+0.0265$ total improvement, procedural memory alone delivers $+0.0225$ — roughly 85% of the lift. Declarative memory contributes only $+0.0007$ in isolation, within the range of run-to-run noise. The implication is clear: for this task, *how* the agent searches — query formulation heuristics, decomposition rules, known dead ends — matters far more than *what* entity-level facts it has pre-cached.

This finding aligns with the cognitive motivation of our architecture. In ACT-R, novice-to-expert progression is driven primarily by production compilation: the conversion of slow, deliberate strategies into efficient automatic routines Anderson et al. [2004]. Declarative knowledge enables reasoning but does not, on its own, make the reasoner faster or more directed. Our ablation provides an empirical parallel: the procedural stores (M_1, M_2, M_3) reshape the agent’s search trajectory, while the declarative store (M_4) leaves it largely unchanged.

Procedural memory improves quality and reduces cost simultaneously. Beyond the score gain, the procedural variants are uniformly more compute-efficient. Compared to the no-memory baseline, procedural-only reduces search queries by 5.9%, LLM calls by 4.5%, and — most strikingly — repeated URL scraping by 24.3% ($25.69 \rightarrow 19.45$). The learned search rules and failure patterns are not merely guiding the agent toward better evidence; they are actively pruning redundant exploration.

Declarative memory alone is counterproductive. The declarative-only configuration is the most expensive of all four settings: it generates the most questions per sample (24.61 vs. 23.52 for neither), the largest scratchpad (46,917 chars), and the most LLM calls (328.34), yet achieves only a negligible score improvement. Without procedural guidance to direct search, cached facts appear to encourage the agent to pursue more leads rather than better ones, inflating the research trajectory without compounding into stronger outputs. Full per-configuration behavioural metrics are reported in Table 5.

The two channels are weakly complementary. If the effects were strictly independent, the combined lift would be approximately $0.0225 + 0.0007 = 0.0232$. The full system achieves $+0.0265$, a small super-additive effect of ≈ 0.003 . This is consistent with the interpretation that declarative facts become useful *conditional* on good procedural strategy: once the agent knows how to search and which paths to avoid, having entity-specific facts on hand allows it to short-circuit a few lookups that would otherwise be redundant. The behavioural metrics for the full system are essentially indistinguishable from procedural-only — the score moves, the trajectory does not.

Metric	Procedural	Both	Declarative	Neither
Score	0.5751	0.5791	0.5533	0.5526
Avg. questions / sample	21.95	22.13	24.61	23.52
Avg. agent steps / question	7.82	7.61	7.72	8.71
Avg. scratchpad (chars)	42,059	41,172	46,917	39,188
Avg. search queries	140.63	139.61	145.91	149.49
Avg. unique URLs scraped	38.42	37.38	41.52	36.55
Avg. repeated URLs scraped	19.45	19.02	26.72	25.69
Avg. repeated Google URLs	407.10	410.97	461.56	500.71
Avg. LLM calls	309.28	310.79	328.34	323.97

Table 5: Per-configuration behavioural metrics for the four ablation settings.

E Sales Automation Evaluation Rubric

This appendix details the LLM-judge evaluation protocol used for the Sales Automation task referenced in Section 4.3. All judging is performed with Qwen-3.5-35B-A3B using a single prompt per (ground-truth point, candidate pitch) pair.

Judge persona. The judge is instructed with the following persona:

“You are a Senior Sales Enablement Evaluator. You are grading on Sales Effectiveness and Factual Precision.”

Scoring procedure. For each ground-truth winning pitch point, the judge selects the best-matching candidate point from the system’s prediction and assigns an integer score on a 0–5 scale (Table 6). The per-sample *normalized score* is then computed as

$$\text{normalized_score} = \frac{1}{5|G|} \sum_{g \in G} s(g)$$

where G is the set of ground-truth pitch points for the sample and $s(g) \in \{0, 1, 2, 3, 4, 5\}$ is the score assigned to point g . The reported *average score* in Section 5 is the mean of *normalized_score* across the test set.

Score	Label	Description
0	Miss / Irrelevant	Candidate completely misses this concept. No mention of the feature, benefit, or metric.
1	Marketing Fluff	Vaguely mentions the topic (e.g. “improved efficiency”) but lacks any specific substance. Generic platitude.
2	Topic Match	Identifies the correct product or pain point, but misses the specific solution or outcome. E.g. GT says “reduced downtime by 40%,” candidate says “helps with downtime.”
3	Implied / Soft Match	Captures the core value proposition correctly, but misses <i>hero evidence</i> (specific numbers, names, unique mechanisms). Solid conversational point but less persuasive than GT.
4	Strong Sales Argument	Captures core value <i>and</i> the key mechanism/outcome. Persuasive and accurate. May miss a minor detail (date, city) that doesn’t affect persuasion.
5	Strategic Bullseye	Perfect extraction: product + pain + value + specific metric/evidence, essentially verbatim from GT. “This is exactly why they bought.”

Table 6: Rubric used by the LLM judge to score each ground-truth pitch point against the candidate pitch.

F Generalization to Held-Out Samples

We now evaluate whether the gains observed during training transfer to unseen samples. Table 3 reports performance on the 120-sample test set, comparing the trained system against a cold-start ablation under two per-question compute budgets: a generous budget of 20 agent steps per sub-question (run (a)) and a tighter budget of 10 (run (b)).

To make the effect of training easier to read, Table 7 recasts the same numbers as relative changes induced by training, separately for each budget.

Finding: learned discipline wins under a tight budget and generalizes beyond training. Under the more realistic per-question budget of 10 agent steps (run (b)), training raises the score from 0.5526 to 0.5791 (a +4.8% relative gain) while simultaneously reducing search queries by 6.6%, LLM calls by 4.1%, agent steps per question by 10.4%, and repeated URL scraping by 26%. The same tightening appears one layer deeper in the pipeline: repeated Google URLs returned by the search API drop by 17.9%. Importantly, these gains do not reflect overfitting: there is no overlap in triples between train and test, the learned procedural efficiency patterns transfer to unseen samples, and the improvements persist across different compute budgets.

Metric	Run (a), budget = 20	Run (b), budget = 10
Score	+3.0% (better)	+4.8% (better)
Search queries	−13.5%	−6.6%
Repeated Google URLs	−22.8%	−17.9%
Repeated URLs scraped	−21.9%	−26.0%
Unique URLs scraped	+1.5%	+2.3%
LLM calls	−9.7%	−4.1%
Agent steps / question	−10.3%	−10.4%

Table 7: Relative change from *no training* to *after training*. Positive values indicate increase; for Score, positive is better. Efficiency metrics improve under training in both runs.

G Statistical Significance of the Sales Task Scores

We report 95% confidence intervals (CI) for the mean Sales Task scores using a non-parametric bootstrap procedure. For each method, we repeatedly resample the per-sample scores with replacement, compute the mean score for each resample, and take the 2.5th and 97.5th percentiles of the resulting distribution of means as the confidence interval bounds. We use 1000 bootstrap iterations. This approach avoids assuming normally distributed scores and directly estimates the sampling variability from the observed data.

Table 8: 95% bootstrap confidence intervals for the Sales Task score.

Section	Method	95% CI
LLM no-web	Qwen	[0.386, 0.446]
	Qwen + Memory	[0.402, 0.469]
	GPT-5-mini	[0.339, 0.445]
	GPT-5-mini + Memory	[0.345, 0.455]
In-context	5-shot Qwen	[0.343, 0.408]
	5-shot Qwen + Memory	[0.348, 0.415]
	5-shot GPT	[0.417, 0.507]
	5-shot GPT + Memory	[0.429, 0.519]
ReAct	ReAct + Qwen	[0.428, 0.495]
	ReAct + Qwen + Memory	[0.435, 0.502]
	ReAct + GPT	[0.486, 0.556]
	ReAct + GPT + Memory	[0.491, 0.559]
Ours	Qwen + AET	[0.515, 0.590]
	Qwen + AET + MEMENTO	[0.541, 0.614]
	GPT + AET	[0.500, 0.565]
	GPT + AET + MEMENTO	[0.516, 0.583]

The bootstrap confidence interval is computed as follows. Given per-sample scores $s = [s_1, s_2, \dots, s_N]$, we repeatedly sample N scores with replacement, compute the resampled mean, and repeat this process 1000 times to obtain a distribution of bootstrap means. The 95% CI is then given by the 2.5th and 97.5th percentiles of this distribution.

```
def bootstrap_ci(scores, n_iter=1000):
    boot = [np.random.choice(scores,
                              size=len(scores),
                              replace=True).mean()
            for _ in range(n_iter)]
    lo = np.percentile(boot, 2.5)
    hi = np.percentile(boot, 97.5)
    return scores.mean(), lo, hi
```

H Legal Task: Training Reduces Cost and Improves Efficiency

We evaluate the trained system on a 120-sample legal-task benchmark. Table 9 contrasts the baseline agent (*no training*) with the fully trained agent (*with training*); the latter receives the procedural and declarative artifacts learned during training.

Across all efficiency metrics, training yields consistent improvements. The trained agent requires fewer questions per sample (20.61 vs. 23.68), fewer agent steps per question (6.63 vs. 7.29), and substantially fewer search queries (116.09 vs. 144.72). It also reduces total web interaction, including both unique and repeated URL fetches, as well as Google result exploration.

Overall, training significantly improves system efficiency, reducing both latency and resource usage while maintaining a similar problem-solving trajectory.

Metric	Baseline	Trained	Δ
Total samples	120	120	—
Avg. questions per sample	23.68	20.61	−13.0%
Avg. agent steps per question	7.29	6.63	−9.1%
Avg. search queries per sample	144.72	116.09	−19.8%
Avg. unique URLs scraped	16.12	15.77	−2.2%
Avg. repeated URLs scraped	13.69	11.41	−16.7%
Avg. total URLs scraped	29.81	27.18	−8.8%
Avg. unique Google URLs	420.39	327.13	−22.2%
Avg. repeated Google URLs	581.80	481.99	−17.2%
Avg. short-term scratchpad size (chars)	37684.12	32738.99	−13.1%
Avg. time per sample (s)	4769.92	3436.06	−28.0%
Avg. LLM calls	295.01	238.01	−19.3%

Table 9: Trained vs. baseline agent on the legal task. Metrics computed over 120 samples.

Interpretation. The pattern matches what we observed in the SDR ablations (Table 5): procedural artifacts – search rules, craft knowledge, and failure patterns – shape the trajectory toward shorter, less redundant exploration. The agent asks fewer questions, re-scrapes fewer pages, and writes a tighter scratchpad, while still producing slightly better final answers. In other words, training does not simply trade compute for accuracy; it improves both axes simultaneously. For high-baseline tasks like this one, the dominant value of training is *efficiency*, with accuracy gain a useful by-product.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

Answer: [\[Yes\]](#)

Justification: The abstract and introduction state four main contributions: (1) using the web as a learning signal for craft acquisition, (2) the Adaptive Exploration Tree for within-session search, (3) a dual-channel memory architecture grounded in ACT-R, and (4) empirical validation across two structurally opposed domains. Each claim is directly supported by the experimental results in Section 5: Tables 1 demonstrate consistent improvements in both Sales Automation and Legal Research, and the ablation study (Tables 4–5) validates the complementary roles of procedural and declarative memory. The paper clearly scopes its claims to prompt-and-memory-level learning without weight updates, and does not overclaim generalization beyond the two evaluated domains.

Guidelines:

- The answer [\[N/A\]](#) means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [\[No\]](#) or [\[N/A\]](#) answer to this question will not be perceived well by the reviewers.

- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: Section 5 includes a dedicated Limitations and Future Work paragraph discussing five key limitations: memory is frozen at inference time, the AET is computationally expensive during training, performance depends on web coverage of the target domain, evaluation is limited to two domains and two backbone models, and the ablation of procedural vs. declarative memory has been conducted only on the sales task.

Guidelines:

- The answer [\[N/A\]](#) means that the paper has no limitation while the answer [\[No\]](#) means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate “Limitations” section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren’t acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[N/A\]](#)

Justification: The paper does not include theoretical results or proofs. Our contribution is an empirical framework validated through experiments on two domains.

Guidelines:

- The answer [\[N/A\]](#) means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.

- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: Section 4 describes both tasks, datasets, evaluation metrics, and baselines in full. Appendix A reports all hyperparameters (question budgets, wave counts, agent step caps, exploration budgets, batch sizes, temperature, search date restrictions, and the backbone LLM). The memory architecture and batch update loop are specified in Sections 3.3–3.4. The SDR-Bench dataset is publicly available Srivastava et al. [2026] and the JUSTICE benchmark is publicly available Alali et al. [2021]; our filtering pipeline for the legal task is described in detail. All prompts and memory store formats are described at a level sufficient to reimplement the system.

Guidelines:

- The answer [\[N/A\]](#) means that the paper does not include experiments.
- If the paper includes experiments, a [\[No\]](#) answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [\[No\]](#)

Justification: The underlying data is publicly available: we use SDR-Bench Srivastava et al. [2026] and JUSTICE Alali et al. [2021] unmodified, partitioned with seed 42 into the train/test splits described in Section 4.1. We do not release the MEMENTO codebase at this time, but the methodology section, Appendix A (hyperparameters), the rubric in Appendix E, and the prompts described throughout the paper provide sufficient detail to reimplement the framework. We may release code in a future version of this work.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://neurips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://neurips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Section 4 specifies both datasets, data splits (with random seed), temporal leakage prevention, evaluation metrics, and baseline configurations. Appendix A provides all hyperparameters including question budgets, wave counts, agent step caps, batch sizes, exploration budgets, temperature settings, search date restrictions, and the backbone LLM used. No weight-level optimization is performed, so optimizer details are not applicable.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report 95% bootstrap confidence intervals for the Sales Automation task in Appendix G (Table 8), covering all baselines and our full system on both Qwen3-35B and GPT-5-mini. Intervals are computed by resampling the 120 per-sample normalized scores with replacement (1,000 iterations) and taking the 2.5th and 97.5th percentiles of the bootstrap distribution of the mean. The variability captured is sampling variation across the test set. We use bootstrap rather than $\bar{x} \pm 1.96 \cdot s/\sqrt{n}$ because per-sample scores are clipped to [0, 1] and not normally distributed.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Appendix B details the hardware setup (32× NVIDIA A100 80 GB GPUs across 4 nodes for Qwen inference, Azure OpenAI endpoints for GPT-5-mini), the web-scraping infrastructure and concurrency limits, and per-run wall-clock times for both training and inference on the sales task. The full research project involved additional preliminary and debugging runs beyond those reported.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We have reviewed the NeurIPS Code of Ethics. All datasets used are publicly available, no human subjects are involved, and the legal task applies PII removal to case inputs. The work raises no dual-use concerns beyond those inherent to general-purpose language model research.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [\[Yes\]](#)

Justification: We discuss broader impacts in Section 5. On the positive side, MEMENTO lowers the barrier to applying LLM-based assistance in domains with scarce labeled data, requires no GPU-intensive fine-tuning (its memory artifacts are human-readable text), and learns to search more efficiently over time, which reduces redundant web traffic and compute cost relative to RL-based alternatives. The fact that procedural and declarative memory are stored in natural language also makes failure modes auditable in a way that weight-level adaptation is not. On the negative side, the same craft-acquisition mechanism could be misapplied: in sales, more persuasive automated pitches based on aggregated public information about prospects raise concerns about consent and manipulative micro-targeting; in legal research, an over-confident automated analysis could mislead practitioners or be used by parties without legal counsel as a substitute for professional advice; and any system that learns to search the web more effectively can in principle be repurposed for opposition research, surveillance, or terms-of-service violations against websites. We also note that web-derived expertise inherits whatever biases are over-represented online (English-language, Western, well-indexed sources). We recommend that deployments in consequential domains such as legal practice include human review, that the date-restricted retrieval mechanism we use for evaluation be retained in production for compliance with data-currency norms, and that the interpretability of MEMENTO’s memory stores be leveraged for periodic audits of what the system has learned.

Guidelines:

- The answer [\[N/A\]](#) means that there is no societal impact of the work performed.
- If the authors answer [\[N/A\]](#) or [\[No\]](#), they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [\[N/A\]](#)

Justification: The paper does not release any high-risk artifacts of the kind this question targets. We do not release a new pre-trained language model, an image generator, or

a newly scraped dataset. MEMENTO is a framework that operates on top of existing publicly available models (Qwen3-35B and GPT-5-mini), each of which is distributed under its own provider’s safeguards, and is evaluated on existing public benchmarks (SDR-Bench Srivastava et al. [2026] and JUSTICE Alali et al. [2021]). The only novel artifacts produced by training are the procedural and declarative memory stores, which are stored as human-readable natural-language text and are therefore directly inspectable and auditable rather than opaque weight updates. Domain-specific risks of misuse, including manipulative sales targeting, legal misadvice, and surveillance-style information gathering, are discussed in the Broader Impact paragraph of Section 5 along with recommended deployment-time mitigations.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [\[Yes\]](#)

Justification: Both datasets are cited: SDR-Bench Srivastava et al. [2026] and JUSTICE Alali et al. [2021]. The Qwen-2.5-235B-Instruct model is open-weight and used under its Apache 2.0 license. GPT-5-mini is accessed through Azure OpenAI under its commercial API terms of service. Specific license versions for the datasets will be included in the final camera-ready appendix.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset’s creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[N/A\]](#)

Justification: The paper does not release new datasets, models, or code as assets. The contribution is a framework evaluated on existing public benchmarks.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve any crowdsourcing or research with human subjects. We use two pre-existing public benchmarks: SDR-Bench Srivastava et al. [2026], whose ground-truth pitches are derived from publicly available customer success stories published by enterprises, and JUSTICE Alali et al. [2021], whose ground-truth case outcomes are sourced from public Oyez records of U.S. Supreme Court decisions. No new human annotations were collected for this work. All evaluation is performed by an automated LLM-as-judge (Qwen3-35B) using the rubric described in Appendix E for the Sales Automation task and a binary correctness check for the Legal Research task; no human raters were employed.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve research with human subjects. As described in the previous item, all data comes from pre-existing public benchmarks (SDR-Bench Srivastava et al. [2026] and JUSTICE Alali et al. [2021]), no new human annotations were collected, and all evaluation is performed by an automated LLM-as-judge. IRB approval (or equivalent) is therefore not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [\[Yes\]](#)

Justification: LLMs are central to the methodology. Qwen-2.5-235B-Instruct and GPT-5-mini serve as the backbone for all agent components (decomposition, tool-augmented solving, reflection, synthesis), for memory consolidation and rule updates during training, and as the LLM judge for evaluation scoring. All model identities, roles, and temperature settings are specified in Section 4.4 and Appendix A.

Guidelines:

- The answer [\[N/A\]](#) means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.