FROM KNOWLEDGE ACCESS TO SOURCE LEARNING: DEVELOPING SOURCE-SPECIFIC COMPETENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents increasingly rely on persistent external sources to solve sequences of knowledge-intensive tasks. Existing methods improve how source content is accessed and organized, while agent-memory systems preserve reusable knowledge from prior interactions, but repeated use of the same source is still largely treated as repeated access rather than an opportunity to progressively improve understanding of that source. We study *source learning*: developing reusable *source-specific competence* over a persistent authoritative source. We represent this competence with a persistent *source model* that captures reusable understanding of the source, including how its knowledge is structured, interpreted, and applied. To construct and progressively refine such models, we propose **`SourceLearn`**, which combines two complementary learning mechanisms. *Self-Directed Source Learning* identifies what remains incompletely understood and adaptively revisits the source, while *Task-Guided Source Learning* uses downstream experience to reveal local representational gaps and recurring needs in how source knowledge should be organized. In both cases, learning signals determine what should be reconsidered, while persistent updates are reconstructed from the authoritative source. Across five benchmarks and three LLM backends, **`SourceLearn`** achieves the best performance in 13 of 15 settings, with gains of up to 22.6 points over Hybrid RAG and substantial overall improvements over static source representations and experience-based memory baselines. The code is available at `https://anonymous.4open.science/r/SourceLearn`.

1 INTRODUCTION

Large language model (LLM) agents increasingly rely on external sources to solve knowledge-intensive tasks. Here, a *source* denotes an identifiable body of external knowledge, such as a document collection, a code repository, or API documentation. Existing work has substantially improved how such sources are accessed and organized through retrieval, long-context modeling, and structured source representations (Lewis et al., 2020; Xu et al., 2024; Sarthi et al., 2024; Edge et al., 2024; Gutiérrez et al., 2024; 2025). In parallel, agent-memory methods preserve reusable knowledge acquired from prior interactions so that experience can influence future behavior (Shinn et al., 2023; Zhao et al., 2024; Xu et al., 2025). These directions make source content easier to access and organize, and past experience easier to reuse, but they optimize different objects: source-access methods primarily support query-time use of external knowledge, while agent memory primarily preserves knowledge derived from previous interactions. When many tasks depend on the same source, an additional opportunity arises: the agent can use prior study and task experience to improve how it represents and uses that source. This motivates a complementary learning problem: can repeated interaction with the source progressively develop reusable competence about the source itself, beyond repeatedly accessing its content or simply accumulating task experience?

We call this process *source learning*. Its goal is to develop *source-specific competence*: reusable understanding of a source that can be applied to new tasks grounded in that source. Such competence goes beyond retaining or compressing source content: it captures how source knowledge is structured, interpreted, and applied, including relationships among concepts, conditions governing their applicability, procedures, and how different pieces of information combine to support future reasoning and action. For example, an agent studying API documentation may learn how prerequisites and dependencies connect multiple calls, enabling it to reason about new combinations of those calls. In

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

our setting, the source is *persistent* in that it remains available across tasks, and *authoritative* in that it serves as the reference for factual content. The agent's accumulated understanding complements rather than replaces direct access to this source. Repeated study and use can reveal gaps in this understanding and guide its refinement, turning source use into a cumulative learning process rather than a sequence of isolated access events.

Developing such competence raises two key challenges. **I)** ***What source knowledge should be learned before future tasks are known?*** A source may contain far more information than should be retained, while a one-pass representation may fail to connect information distributed across the source or capture conditions that are only implicit. Therefore, source learning must identify which knowledge is broadly reusable and organize it into a form that can support future tasks whose exact demands are not yet known. **II)** ***How can task-specific feedback improve source-specific competence beyond the task that produced it?*** Downstream use may expose deficiencies that prior study did not anticipate, but directly retaining the observed answer or correction risks learning the task rather than the source. Effective adaptation must instead use task-level evidence to identify deficiencies in source understanding and address them with reusable, source-grounded knowledge.

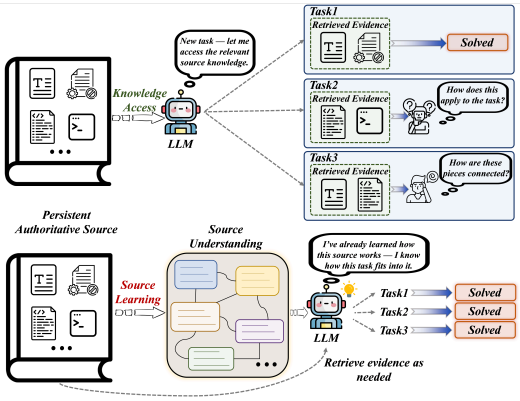

Figure 1: **Problem Illustration. I)** Knowledge access retrieves task-relevant evidence from the source, but does not necessarily accumulate reusable source understanding across tasks, requiring the model to repeatedly reconstruct how evidence should be interpreted or combined. **II)** Our goal is source learning: progressively developing reusable source understanding through continued study and use, while retaining direct access to the authoritative source.

We introduce `SourceLearn`, which maintains a persistent *source model* through two complementary learning mechanisms. *Self-Directed Source Learning* revisits incompletely understood source content and connects knowledge across entities. *Task-Guided Source Learning* uses failures to identify local representational gaps and recurring task demands to improve how knowledge is organized. Both mechanisms use observations to guide learning while reconstructing persistent updates from the authoritative source.

Our contributions are threefold:

❶ **Source Learning Formulation and Representation.** We formulate *source learning* as the development of reusable *source-specific competence* and introduce the *source model* as an explicit, persistent, and revisable representation of that competence, complementing direct source access while preserving the source as the factual authority.

❷ **Self-Directed and Task-Guided Source Learning.** We develop two complementary mechanisms for progressively building source-specific competence: Self-Directed Source Learning proactively studies the source to deepen incomplete understanding and connect fragmented knowledge, while Task-Guided Source Learning uses deficiencies and recurring demands revealed through downstream tasks to further refine that understanding.

❸ **Strong Empirical Performance.** We evaluate `SourceLearn` across document QA, code QA, tool use, and interactive environments. It substantially improves overall same-source task performance over task-local retrieval, static source representations, and experience-based memory baselines, with gains from both self-directed and task-guided source learning.

2 Related Work

2.1 External Knowledge Access and Source Representations

Retrieval-augmented language models incorporate external knowledge through task-time retrieval or retrieval-augmented pretraining (Guu et al., 2020; Karpukhin et al., 2020; Lewis et al., 2020; Izacard & Grave, 2021; Borgeaud et al., 2022; Izacard et al., 2023; Shi et al., 2024). Subsequent work improves when and how evidence is accessed through iterative, active, adaptive, or self-reflective

retrieval (Trivedi et al., 2023; Jiang et al., 2023; Asai et al., 2024; Jeong et al., 2024; Yan et al., 2024), while long-context methods expose larger portions of the source directly but still face challenges in reliably using distributed information (Xu et al., 2024; Liu et al., 2024). Complementary approaches structure source knowledge: RAPTOR builds hierarchical summaries (Sarthi et al., 2024); GraphRAG, LightRAG, GraphReader, and KAG use graph-based structures (Edge et al., 2024; Guo et al., 2025; Li et al., 2024; Liang et al., 2025); and HippoRAG, HippoRAG 2, and MemoRAG support associative retrieval, continual knowledge integration, and global memory, respectively (Gutiérrez et al., 2024; 2025; Qian et al., 2025). StructRAG constructs task-conditioned representations at inference time (Li et al., 2025). These approaches primarily improve how source knowledge is accessed, organized, or retained. In contrast, `SourceLearn` treats competence over a persistent authoritative source as the object of learning itself, progressively refining a reusable source model through continued study and use.

2.2 Agent Memory and Experiential Learning

Agent-memory methods enable LLM agents to retain and reuse information accumulated through prior interactions (Wang et al., 2024b; Zhang et al., 2025). Existing systems maintain conversational, episodic, or hierarchical memories (Park et al., 2023; Zhong et al., 2024; Liu et al., 2023; Packer et al., 2023), or learn from task experience by retaining feedback, abstracting reusable insights, and acquiring skills or workflows (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2024a; 2025). More recent work further organizes such experience into structured and revisable representations, including knowledge-graph world models in AriGraph (Anokhin et al., 2025), interconnected memories in A-MEM (Xu et al., 2025), and procedural memory in ReMe (Cao et al., 2026). `SourceLearn` differs in the object and authority of learning: task experience is used to reveal deficiencies or recurring representational needs in source understanding, while persistent updates are reconstructed from the authoritative source rather than stored directly from interaction experience.

3 Problem Formulation

3.1 Knowledge Access over a Persistent Source

We consider a persistent authoritative source D and a sequence of tasks $(q, y) \sim \mathcal{P}_D$ grounded in the same source. Let $R(D, q)$ denote the source context made available for task q, and let F denote the task-solving LLM agent. Existing source-grounded systems can be broadly written as

$$\hat{y} = F\big(q, R(D, q)\big), \tag{1}$$

where R may represent retrieval, long-context access, or a structured source representation. These methods differ in how source knowledge is organized and accessed, but primarily aim to provide useful information for the current task.

Access Does Not Imply Source Learning. Repeated access to the same source does not necessarily improve the system's understanding of that source. Even with persistent indexes or structured representations, reusable knowledge may remain fragmented or need to be reconstructed across tasks. Source learning instead asks whether repeated study and use can progressively develop reusable understanding that carries across future tasks.

3.2 Source-Specific Competence

We define *source-specific competence* as reusable understanding of a source that can support new tasks grounded in that source, including how its knowledge is structured, interpreted, and applied. We represent this competence with a persistent and revisable *source model* M. Given the source D and accumulated study and task experience H_t, source learning develops

$$M_t = \mathrm{Learn}(D, H_t), \tag{2}$$

so that understanding acquired through earlier study and use can be carried forward to future tasks.

For a future task, this accumulated understanding complements rather than replaces direct source access:

$$\hat{y} = F\big(q, R(D, q), M_t\big), \tag{3}$$

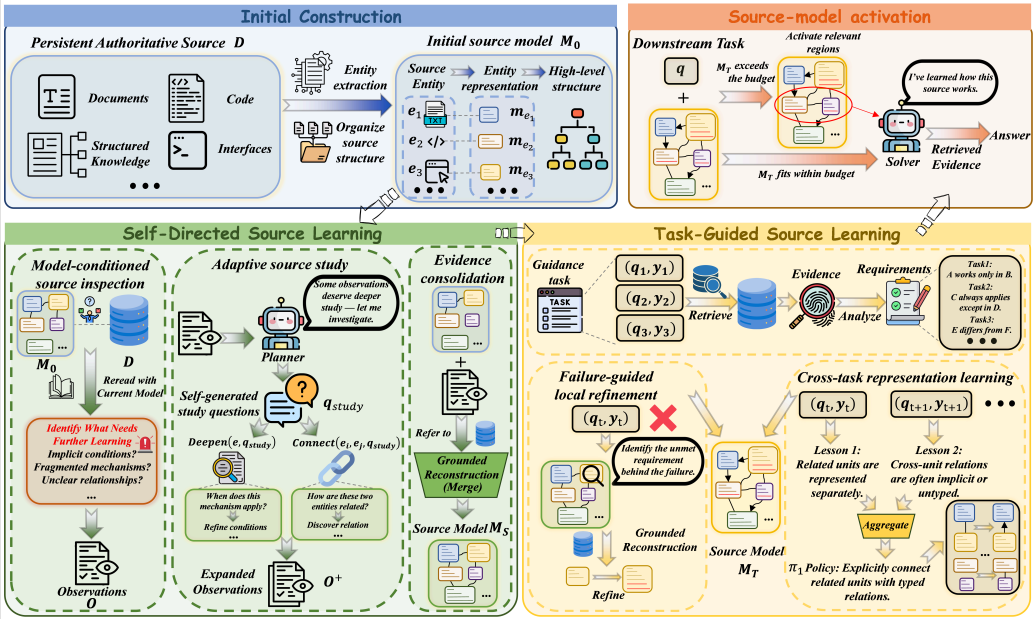

Figure 2: **Overview of `SourceLearn`. Initial Construction** builds a provisional source model M_0 from the authoritative source D. ***Self-Directed Source Learning*** deepens and connects source understanding through adaptive study, producing M_S. ***Task-Guided Source Learning*** uses guidance-task failures and recurring representation needs to further refine the model into M_T. At inference time, **Source-Model Activation** combines relevant source-model context with directly retrieved evidence for downstream tasks.

where $R(D, q)$ provides task-specific source information, while M_t provides reusable understanding accumulated through prior study and use. The original source remains authoritative for factual content. Source learning therefore seeks to improve future same-source task performance by progressively developing stronger and more reusable understanding of the source.

4 Method

4.1 Overview

`SourceLearn` develops reusable competence over a persistent authoritative source D by maintaining an explicit and revisable source model M. Rather than treating each task as an independent access event, the system carries forward an organized understanding of the same source and progressively develops that understanding through continued study and use.

The source model evolves through two complementary learning mechanisms. *Self-Directed Source Learning* uses the current model to guide renewed study of the source, with the goal of deepening incomplete understanding and connecting knowledge that remains fragmented. *Task-Guided Source Learning* uses downstream task experience to reveal both concrete representational deficiencies and recurring preferences about how source knowledge should be represented. The overall process is

$$M_0 = \text{Construct}(D), \quad M_S = \text{SelfLearn}(D, M_0), \quad M_T = \text{TaskLearn}(D, M_S; \mathcal{G}). \tag{4}$$

Here, M_0 is the initial source model, M_S is the model after self-directed learning, and M_T is the final model after task-guided learning. $\mathcal{G} = \{(q_t, y_t)\}_{t=1}^{T}$ denotes a set of guidance tasks grounded in the same source.

4.2 Source Model Construction and Activation

Source representation. We organize the source model around entities e, each associated with a reusable representation m_e. Entity representations capture source-grounded understanding that may be useful across tasks, such as important properties, mechanisms, governing conditions, procedures,

exceptions, distinctions, and relations to other entities. Beyond individual entities, M may also contain higher-level structure that organizes recurring patterns and dependencies across multiple entities.

The purpose of M is not to reproduce all of D. Instead, it maintains a compact and reusable understanding of the source while leaving easily retrievable low-level details in the original source. This separation is important: the source model captures what prior study has learned about the source, while D remains directly available as the authoritative factual reference.

Initial construction. We construct a provisional source model M_0 through source-aware, entity-centered reading. Large sources may first be partitioned into coherent regions, while native organization such as code modules, symbols, or document sections is used when available. Within each region, the reader constructs representations of the major entities and organizes reusable structure that spans them.

Construction follows a generic modeling instruction Π_0 that favors major entities, important relations, procedures, governing conditions, distinctions, and representative details while avoiding redundant low-level content. The resulting M_0 is deliberately provisional: it provides a structured starting point for subsequent learning rather than an exhaustive representation of the source.

Grounded source reconstruction. All later learning stages follow a common principle: learning signals determine *what should be reconsidered*, while persistent knowledge must be reconstructed from the authoritative source. For an affected source region R,

$$O_R = \mathrm{Inspect}(D_R, M_R; \xi), \quad M' = \mathrm{GroundApply}\big(M, \mathrm{Reconstruct}(M_R, O_R); D_R\big). \tag{5}$$

Here, ξ is the learning signal and O_R contains temporary observations obtained by rereading the relevant source content against the current model.

A reconstruction is committed only when the proposed content is supported by source evidence and previously supported meaning is preserved unless it is subsumed by a stronger representation. Reconstruction may merge fragmented knowledge, make important conditions or relations explicit, or remove redundant organization, but observations themselves are never copied directly into persistent memory. This shared reconstruction mechanism is used by both learning stages below.

Source-model activation. The source model may eventually exceed the context budget available to a task. We therefore expose at most B tokens of accumulated source understanding:

$$A_B(M, q) = \begin{cases} M, & \mathrm{tok}(M) \le B, \\ \mathrm{ActivateRegions}(M, q; B), & \mathrm{tok}(M) > B. \end{cases} \tag{6}$$

When activation is necessary, it operates over coherent source-model regions established during construction, selecting task-relevant regions while preserving their internal structure.

For a task q, the solver receives both activated source understanding and task-specific evidence directly retrieved from the authoritative source:

$$\hat{y} = F\big(q, R(D, q), A_B(M, q)\big). \tag{7}$$

The two contexts therefore serve complementary roles: $A_B(M, q)$ provides reusable understanding accumulated through prior study and use, while $R(D, q)$ preserves direct access to precise source evidence.

4.3 Self-Directed Source Learning

An initial source model cannot anticipate everything that may later prove useful to understand. Some mechanisms may remain only partially represented, governing conditions may remain implicit, and knowledge distributed across multiple entities may remain fragmented. Self-Directed Source Learning therefore uses the current model to guide renewed study of the source, allowing the system to decide what deserves deeper investigation before future task requirements are known.

Each self-study cycle follows an *Inspect–Study–Consolidate* process:

$$O = \mathrm{Inspect}(D, M), \quad O^+ = \mathrm{Study}(D, M, O), \quad M' = \mathrm{Consolidate}(D, M, O^+). \tag{8}$$

Model-conditioned source inspection. The cycle begins with a broad inspection of the source conditioned on the current model. For each entity e, the system rereads its associated source content S_e together with its current representation m_e:

$$O_e = \mathrm{Observe}(S_e, m_e). \tag{9}$$

Rather than summarizing S_e again, inspection asks what the current representation still fails to explain well. For example, a mechanism may be represented only partially, the governing condition of a rule may remain implicit, or a reference to another entity may not yet be integrated into the model.

Each observation records the relevant source evidence and its implication for the current representation. These observations are provisional: they do not directly modify M, but instead identify promising directions for further study and provide material for later consolidation.

Adaptive source study. Inspection may reveal many possible gaps, not all of which are equally important. At planning round b, an adaptive planner selects a small set of study actions using the current model, accumulated observations, and the history of previous study:

$$\Gamma_b = \mathrm{PlanStudy}(M, \overline{O}_b, H_{<b}), \quad \Gamma_b \subseteq \left\{\textsc{Deepen}(e, q^{\mathrm{study}}), \textsc{Connect}(e_i, e_j, q^{\mathrm{study}})\right\}. \tag{10}$$

DEEPEN investigates an unresolved aspect of one entity, such as a mechanism, condition, or procedure. CONNECT jointly studies two entities when inspection suggests that an important dependency or shared structure remains insufficiently understood.

For each action, the planner formulates a focused temporary study question q^{study}, which guides retrieval and renewed source reading. The resulting evidence produces additional observations that can inform subsequent study rounds. The questions themselves are learning instruments and are not stored in the source model.

Evidence consolidation. Inspection and study accumulate observations while leaving the persistent model unchanged. Once study terminates, the accumulated observations are consolidated through the grounded reconstruction mechanism in Eq. (5). Entity-specific observations revise the corresponding representations, while cross-entity observations can update higher-level relational structure.

This produces a *read-many, write-once* pattern: the system can broadly inspect and selectively study the source before committing a persistent update. Applying this process to M_0 yields the self-directed source model M_S.

4.4 TASK-GUIDED SOURCE LEARNING

Self-directed learning asks what the current source model still needs to understand without assuming a particular future task. Actual downstream use provides a complementary signal: it reveals which parts of source knowledge matter in practice and whether the current representation makes them available in a useful form.

Task experience can reveal deficiencies at two levels. A particular failure may show that some required source knowledge is missing from the model, while repeated tasks may reveal a broader preference about *how* source knowledge should be represented. We therefore perform Task-Guided Source Learning through two complementary pathways: *failure-guided local refinement* and *cross-task representation learning*.

Diagnosing source requirements from task use. For each guidance task $(q_t, y_t) \in \mathcal{G}$, we first return to the authoritative source to identify what source evidence supports the reference outcome and what source understanding the task requires:

$$\widetilde{E}_t = \mathrm{RetrieveGuide}(D, q_t, y_t), \quad (E_t^*, C_t) = \mathrm{Diagnose}(q_t, y_t, \widetilde{E}_t). \tag{11}$$

Here, E_t^* contains the source evidence that directly supports the reference outcome, while C_t contains the corresponding source requirements. Thus, E_t^* specifies where the required support comes from, whereas C_t describes what understanding the source model would need to make available.

This is the only stage of source learning that directly observes y_t. The reference outcome is used only to locate authoritative support and derive source requirements; neither the answer nor its supporting evidence is stored directly as persistent knowledge. The guidance task itself is solved using the same task-time interface in Eq. (7).

Failure-guided local refinement. If a guidance task is answered incorrectly, we determine which of its requirements are absent from the current source model:

$$C_t^- = \{c \in C_t : c \text{ is not represented in } M_t\}, \quad M_{t+1} = \mathrm{Refine}(D, M_t; C_t^-). \tag{12}$$

When $C_t^- \neq \varnothing$, the missing requirements identify the part of the source model whose current representation is insufficient. They act only as learning targets: the implicated source region is reread from D and revised through grounded reconstruction rather than by storing the task correction itself.

Requirements that are already represented do not trigger additional writing, which separates a representational deficiency from an error that occurs despite the required source understanding already being available. Processing the guidance tasks in this way produces a locally refined model M_{local}.

Cross-task representation learning. Task experience also provides a more transferable signal. Across multiple tasks, the system may repeatedly encounter cases where certain distinctions should remain explicit, conditions should not be collapsed, related knowledge should be represented together, or information should be maintained at a different level of granularity.

We summarize the signal from each guidance task as a representation lesson and aggregate recurring lessons into a source-level representation policy:

$$\ell_t = \mathrm{Lesson}(q_t, C_t, M_t, \hat{y}_t), \quad \Pi_1 = \mathrm{Aggregate}\left(\Pi_0, \{\ell_t\}_{t=1}^{T}\right). \tag{13}$$

Each lesson describes a preference about *how* source knowledge should be represented rather than a particular source fact or task answer. Lessons are constrained to avoid task-specific identifiers, and Π_1 retains recurring preferences that can generalize beyond the task that produced them.

Importantly, Π_1 does not itself introduce new source knowledge. Instead, it indicates what kinds of source structure deserve clearer or more explicit representation. We therefore return to the authoritative source and recalibrate the locally refined model under the learned policy:

$$M_T = \mathrm{Recalibrate}(D, M_{\mathrm{local}}; \Pi_1). \tag{14}$$

Recalibration revisits the source under Π_1, identifies source-supported structure that is currently missing or represented at an inappropriate granularity, and incorporates it through the same grounded reconstruction mechanism.

Task-Guided Source Learning therefore transfers experience at two levels: concrete failures identify local deficiencies that can be repaired from the source, while recurring task demands induce broader preferences for how source knowledge should be represented. Both pathways modify the same persistent source model, and both preserve the same invariant: task experience determines what should be reconsidered, while the authoritative source determines what can become persistent knowledge.

5 Experiments

We evaluate `SourceLearn` by addressing three questions: **Q1**: does source learning improve same-source task performance across source types and LLM backends (Section 5.2)? **Q2**: how do Self-Directed and Task-Guided Source Learning contribute to the final source model (Section 5.3)? **Q3**: does source learning develop task-relevant source understanding beyond static source compression (Section 5.4)?

5.1 Experimental Setup

Tasks and Datasets. We evaluate `SourceLearn` on five benchmarks spanning document, code, and API sources: **MultiDoc2Dial** (Feng et al., 2021), **NarrativeQA** (Kočiskỳ et al., 2018), and **SWE-QA** (Peng et al., 2026) for source-grounded QA; **APIBench** (Patil et al., 2024) for tool use; and **AppWorld** (Trivedi et al., 2024) for interactive tasks. Each document collection, story, repository,

Table 1: **Main results across document QA, code QA, tool use, and interactive environments.** We report task performance (%) with changes relative to Hybrid RAG shown as subscripts. The best and second-best results are highlighted with **bold** and underline, respectively.

Model	Method	Document QA MultiDoc2Dial	Document QA NarrativeQA	Code QA SWE-QA	Tool Use APIBench	Interactive Env. AppWorld
GPT-5.6 -Luna	Hybrid RAG	64.9	66.0	56.9	47.8	72.0
	RAPTOR	71.2$_{\uparrow 6.3}$	68.1$_{\uparrow 2.1}$	50.8$_{\downarrow 6.1}$	61.9$_{\uparrow 14.1}$	75.6$_{\uparrow 3.6}$
	HippoRAG 2	66.3$_{\uparrow 1.4}$	66.6$_{\uparrow 0.6}$	62.7$_{\uparrow 5.8}$	55.2$_{\uparrow 7.4}$	77.4$_{\uparrow 5.4}$
	AWM	65.1$_{\uparrow 0.2}$	62.1$_{\downarrow 3.9}$	56.7$_{\downarrow 0.2}$	62.8$_{\uparrow 15.0}$	66.7$_{\downarrow 5.3}$
	`SourceLearn`	**77.6**$_{\uparrow 12.7}$	**80.2**$_{\uparrow 14.2}$	**69.5**$_{\uparrow 12.6}$	**70.4**$_{\uparrow 22.6}$	**81.5**$_{\uparrow 9.5}$
gpt-oss -120b	Hybrid RAG	64.5	45.6	52.5	60.3	30.4
	RAPTOR	61.5$_{\downarrow 3.0}$	40.4$_{\downarrow 5.2}$	32.8$_{\downarrow 19.7}$	54.7$_{\downarrow 5.6}$	26.8$_{\downarrow 3.6}$
	HippoRAG 2	66.7$_{\uparrow 2.2}$	45.7$_{\uparrow 0.1}$	**56.7**$_{\uparrow 4.2}$	56.0$_{\downarrow 4.3}$	**33.9**$_{\uparrow 3.5}$
	AWM	64.1$_{\downarrow 0.4}$	43.2$_{\downarrow 2.4}$	51.5$_{\downarrow 1.0}$	63.2$_{\uparrow 2.9}$	28.6$_{\downarrow 1.8}$
	`SourceLearn`	**68.7**$_{\uparrow 4.2}$	**56.7**$_{\uparrow 11.1}$	53.4$_{\uparrow 0.9}$	**71.8**$_{\uparrow 11.5}$	27.4$_{\downarrow 3.0}$
DeepSeek-V4.1 -Flash	Hybrid RAG	63.3	64.2	45.9	67.1	83.9
	RAPTOR	73.2$_{\uparrow 9.9}$	60.4$_{\downarrow 3.8}$	36.8$_{\downarrow 9.1}$	69.3$_{\uparrow 2.2}$	88.7$_{\uparrow 4.8}$
	HippoRAG 2	67.3$_{\uparrow 4.0}$	62.0$_{\downarrow 2.2}$	55.7$_{\uparrow 9.8}$	69.6$_{\uparrow 2.5}$	86.9$_{\uparrow 3.0}$
	AWM	63.7$_{\uparrow 0.4}$	61.4$_{\downarrow 2.8}$	47.4$_{\uparrow 1.5}$	70.8$_{\uparrow 3.7}$	83.9$_{\uparrow 0.0}$
	`SourceLearn`	**81.3**$_{\uparrow 18.0}$	**76.2**$_{\uparrow 12.0}$	**61.6**$_{\uparrow 15.7}$	**82.2**$_{\uparrow 15.1}$	**89.9**$_{\uparrow 6.0}$

or API library defines one persistent source. Guidance and test tasks are disjoint and shared across methods; we use three fixed 30%/70% splits except for AppWorld, which uses its official training and test-normal splits. QA answers are judged by GPT-5.6-Luna (OpenAI, 2026), while APIBench and AppWorld use their official evaluators. Full dataset and evaluation details are provided in Appendix A.

LLM Backends. We use GPT-5.6-Luna (OpenAI, 2026), gpt-oss-120b (Agarwal et al., 2025), and DeepSeek-V4.1-Flash (Xu et al., 2026). Within each configuration, the same backend performs all method-specific LLM operations, and all methods share the same embedding model (OpenAI, 2024) and, on AppWorld, the same agent scaffold. At task time, `SourceLearn` receives the same retrieved evidence as Hybrid RAG together with the activated source model (Eq. (7)).

Counterparts. We compare against task-local retrieval with Hybrid RAG, persistent source representations with RAPTOR (Sarthi et al., 2024) and HippoRAG 2 (Gutiérrez et al., 2025), and experience-based memory with AWM (Wang et al., 2025), which learns from the same guidance tasks as `SourceLearn`. Implementation details for all counterparts are given in Appendix A.2.

5.2 Main Results

To answer **Q1**, Table 1 reports results across all five benchmarks and three backends. `SourceLearn` achieves the best result in 13 of 15 settings and the second-best in one more, improving over Hybrid RAG by **+14.3**, **+4.9**, and **+13.4** points on average with GPT-5.6-Luna, gpt-oss-120b, and DeepSeek-V4.1-Flash, respectively. The gains span document, code, and API sources and exceed 10 points on four of five benchmarks with both GPT-5.6-Luna and DeepSeek-V4.1-Flash. In contrast, RAPTOR, HippoRAG 2, and AWM improve more unevenly across benchmarks, suggesting that progressively learning the source transfers more consistently than static source structuring or retaining task experience alone.

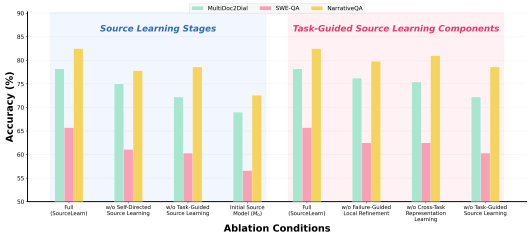

Figure 3: Ablation of `SourceLearn` with GPT-5.6-Luna: the source learning stages (left) and the two pathways of Task-Guided Source Learning (right). Each bar reports answer accuracy (%) on two sources of a QA benchmark.

5.3 Ablation Study

To address **Q2**, we remove each learning mechanism of `SourceLearn` in turn and re-evaluate on two sources of each QA benchmark with GPT-5.6-Luna, keeping the guidance tasks, test splits, and task-time source access of the main results (Figure 3). The left panel ablates the source learning stages. The full model M_T is the most accurate on all three benchmarks. Omitting either Self-Directed or Task-Guided Source Learning lowers accuracy, and the initial source model M_0 is the

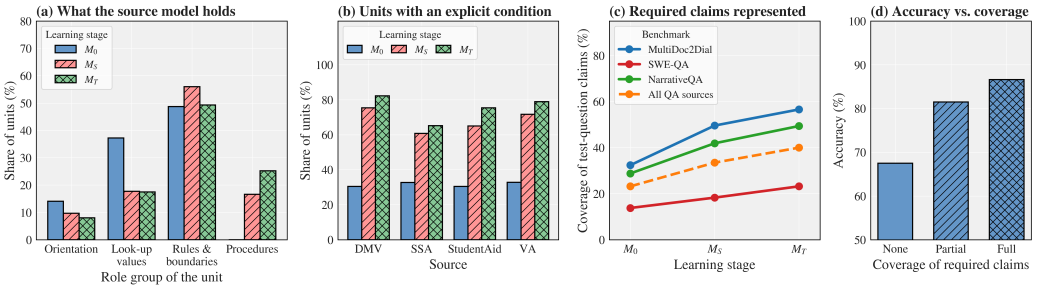

Figure 4: **Source learning beyond compression.** From M_0 to M_S to M_T, source learning (a) changes what the model represents, (b) makes conditions more explicit, (c) increases coverage of source knowledge required by future tasks, and (d) higher coverage is associated with higher answer accuracy.

lowest, showing that the two learning mechanisms are complementary. The right panel ablates the two pathways of Task-Guided Source Learning. Removing either failure-guided local refinement or cross-task representation learning lowers accuracy on every benchmark, and removing both, which recovers M_S, lowers it further. The two pathways thus carry different information from the same guidance tasks: concrete failures repair local gaps in the source model, while recurring representation lessons change how the whole model represents the source, and neither alone recovers the performance of the full model.

5.4 Source Learning Beyond Compression

A natural concern is that the source model is merely a compressed summary of the source. To address **Q3**, we examine how its representation changes through learning and whether these changes matter for future tasks (Figure 4).

Learning changes what is represented. We group source-model units into source overview, facts and attributes, rules, relations and constraints, and procedures. From M_0 to M_S and M_T, the representation shifts away from isolated look-up content toward structured rules and procedures (Figure 4a), while units with explicit applicability conditions rise from roughly 30% in M_0 to 65–82% in M_T across sources (Figure 4b). Thus, source learning reorganizes what is represented rather than merely retaining more content.

The learned representation is more useful for future tasks. The share of source claims required by test questions that are represented in the model rises from 23.2% in M_0 to 40.0% in M_T, with consistent gains across QA benchmarks (Figure 4c). This coverage is associated with downstream success: accuracy rises from 67.5% when none of the required claims are represented to 86.6% when all are represented (Figure 4d). Together, these results suggest that source learning develops a more structured and task-relevant understanding of the source, rather than merely compressing its content.

6 Conclusion

We formulate *source learning* as turning repeated interaction with a persistent authoritative source into the cumulative development of reusable *source-specific competence*, rather than repeated source access or accumulation of task experience. We represent this competence with a persistent and revisable source model and propose `SourceLearn`, which develops it through complementary Self-Directed and Task-Guided Source Learning while grounding persistent updates in the original source. Across five benchmarks and three LLM backends, `SourceLearn` substantially improves overall same-source task performance over task-local retrieval, static source representations, and experience-based memory. Our analyses further show that source learning progressively makes conditions, procedures, exceptions, and relations explicit and increases coverage of the source knowledge required by future tasks, providing evidence that the learned source model goes beyond static compression. A current limitation is scope: we focus on persistent, authoritative, and relatively stable sources with repeated same-source tasks; extending source learning to evolving, noisy, or conflicting sources remains an important direction for future work.

AI USE STATEMENT

We used generative AI tools to assist with language refinement, organization, limited feedback on the research methodology and experiments, code assistance, and the execution of experimental workflows. Separately, GPT-5.6-Luna, gpt-oss-120b, and DeepSeek-V4.1-Flash serve as the LLM backends of `SourceLearn` and its counterparts. GPT-5.6-Luna additionally serves as the judge for QA answers. In these roles, LLMs are components of the experimental methodology rather than research or writing assistance. The conceptual framework, hypotheses, experimental design, methodological decisions, and interpretation of results were developed and finalized by the authors. All AI-assisted manuscript content and code were reviewed and verified by the authors, and AI-assisted implementations were tested for correctness by at least two authors. The authors take full responsibility for the final content, claims, and artifacts of this work.

ETHICS STATEMENT

We strictly adhere to the ICLR Code of Ethics. This study involves no human subjects or private user data. All sources and tasks are from publicly available benchmarks: MultiDoc2Dial, NarrativeQA, SWE-QA, APIBench, and AppWorld, with AppWorld simulating apps and users in a synthetic environment. Source models are generated by LLMs and may therefore contain errors. `SourceLearn` mitigates this risk by grounding every persistent update in the source and by keeping the source available for task-time retrieval. The learned source model should therefore be treated as an auxiliary representation rather than a replacement for the authoritative source. Our findings concern controlled comparisons across matched experimental conditions.

REPRODUCIBILITY STATEMENT

An anonymous code release containing the implementation and experiment scripts is currently available at `https://anonymous.4open.science/r/SourceLearn`. The benchmarks, splits, metrics, and judging protocol are described in Appendix A.1, the counterparts in Appendix A.2, and the backends, decoding settings, hyperparameters, and hardware in Appendix A.3. GPT-5.6-Luna and DeepSeek-V4.1-Flash are accessed through hosted APIs, while gpt-oss-120b is served locally with vLLM on a server with NVIDIA A100 80GB GPUs. The QA benchmarks and APIBench use three fixed guidance/test splits, while AppWorld follows its official training and test-normal splits. We will make the repository public upon acceptance of this work.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Petr Anokhin, Nikita Semenov, Artyom Sorokin, Dmitry Evseev, Andrey Kravchenko, Mikhail Burtsev, and Evgeny Burnaev. Arigraph: Learning knowledge graph world models with episodic memory for llm agents. In *IJCAI*, 2025.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pp. 9112–9141, 2024.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pp. 2206–2240. PMLR, 2022.

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 16803–16822, 2026.

Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 758–759, 2009.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Song Feng, Siva Sankalp Patel, Hui Wan, and Sachindra Joshi. Multidoc2dial: Modeling dialogues grounded in multiple documents. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6162–6176, 2021.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and fast retrieval-augmented generation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 10746–10761, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.568. URL https://aclanthology.org/2025.findings-emnlp.568/.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569, 2024.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From rag to memory: Non-parametric continual learning for large language models. In *ICML*, volume 267, pp. 21497–21515, 2025.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pp. 3929–3938. PMLR, 2020.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: main volume*, pp. 874–880, 2021.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251): 1–43, 2023.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 conference of the north american chapter of the association for computational linguistics: Human language technologies (volume 1: Long papers)*, pp. 7036–7050, 2024.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7969–7992, 2023.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6769–6781, 2020.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018.

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.

Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, et al. Graphreader: Building graph-based agent to enhance long-context abilities of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12758–12786, 2024.

Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. Structrag: Boosting knowledge intensive reasoning of llms via inference-time hybrid information structurization. In *International Conference on Learning Representations*, volume 2025, pp. 36107–36124, 2025.

Lei Liang, Zhongpu Bo, Zhengke Gui, Zhongshu Zhu, Ling Zhong, Peilong Zhao, Mengshu Sun, Zhiqiang Zhang, Jun Zhou, Wenguang Chen, et al. Kag: Boosting llms in professional domains via knowledge augmented generation. In *Companion Proceedings of the ACM on Web Conference 2025*, pp. 334–343, 2025.

Lei Liu, Xiaoyan Yang, Yue Shen, Binbin Hu, Zhiqiang Zhang, Jinjie Gu, and Guannan Zhang. Think-in-memory: Recalling and post-thinking enable llms with long-term memory. *arXiv preprint arXiv:2311.08719*, 2023.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

OpenAI. text-embedding-3-large. OpenAI API documentation, 2024. URL `https://developers.openai.com/api/docs/models/text-embedding-3-large`. Accessed: 2026-09-20.

OpenAI. GPT-5.6 Luna. OpenAI API documentation, 2026. URL `https://developers.openai.com/api/docs/models/gpt-5.6-luna`. Accessed: 2026-09-20.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565, 2024.

Weihan Peng, Yuling Shi, Yuhang Wang, Xinyun Zhang, Beijun Shen, and Xiaodong Gu. Swe-qa: Can language models answer repository-level code questions? In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 8230–8245, 2026.

Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Defu Lian, Zhicheng Dou, and Tiejun Huang. Memorag: Boosting long context processing with global memory-enhanced retrieval augmentation. In *Proceedings of the ACM on Web Conference 2025*, pp. 2366–2377, 2025.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and trends® in information retrieval*, 4(1-2):1–174, 2009.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher Manning. Raptor: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, volume 2024, pp. 32628–32649, 2024.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 conference of the north american chapter of the association for computational linguistics: Human language technologies (volume 1: Long papers)*, pp. 8371–8384, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

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)*, pp. 10014–10037, 2023.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *TMLR*, 2024a. ISSN 2835-8856.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of computer science*, 18(6):186345, 2024b.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *ICML*, volume 267, pp. 63897–63911, 2025.

Anyi Xu, B Li, Bangcai Lin, Bing Xue, BingCheng Xian, Bingzheng Xu, Bochao Wu, Bowei Zhang, Boyi Deng, CC Yu, et al. Deepseek-v4. 1-flash: Pushing the limits of kv cache compression. *arXiv preprint arXiv:2609.19969*, 2026.

Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. Retrieval meets long context large language models. In *International Conference on Learning Representations*, volume 2024, pp. 49569–49584, 2024.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 17577–17604. Curran Associates, Inc., 2025. doi: 10.52202/085713-0593. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/19909c36f51abc4856b4560aff3d36d6-Paper-Conference.pdf.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*, 2024.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 19724–19731, 2024.

A Experimental Details

A.1 Dataset Details

Every benchmark is cast into the same form: a persistent source, a set of guidance tasks $\mathcal{G}$ used for learning, and a disjoint set of test tasks grounded in the same source. On the three QA benchmarks and APIBench, the tasks of each source are split into 30% guidance and 70% test tasks under three fixed seeds, keeping related tasks (questions from the same dialogue, queries for the same API) on the same side; the same splits are used by every method and backend.

- **MultiDoc2Dial** (Feng et al., 2021). Goal-oriented dialogues grounded in the documents of four government domains (DMV, SSA, StudentAid, and VA). Each domain is one source (92–149 documents) with 60 questions drawn from dialogue turns.
- **NarrativeQA** (Kočiskỳ et al., 2018). Reading comprehension over complete stories. Each of 12 stories from the official test split is one source and is read in full rather than through the provided summaries: six movie scripts (*The Breakfast Club*, *Taxi Driver*, *All About Eve*, *Broadcast News*, *Demolition Man*, and *The Lord of the Rings: The Fellowship of the Ring*) and six books (*Press Cuttings*, *Out of Time's Abyss*, *The Call of the Wild*, *The Castle of Otranto*, *Persuasion*, and *The Old Wives' Tale*), spanning a wide range of genres and lengths, with 27–30 questions each.
- **SWE-QA** (Peng et al., 2026). Repository-level questions about the design and behavior of Python projects. We use six of the fifteen official repositories, chosen to span project kinds and sizes: streamlink (media streaming CLI), sphinx (documentation generator), xarray (labelled N-D arrays), pytest (testing framework), flask (web framework), and conan (C/C++ package manager). Each repository at its pinned commit is one source, consisting of its source code, documentation, and configuration files, with 48 questions.
- **APIBench** (Patil et al., 2024). Instructions that must be fulfilled by a call to the correct API of a model hub. The documentation of a hub is one source: TorchHub (186 instructions), HuggingFace (100), and TensorFlow Hub (100). Correctness is decided by the official AST-based functionality evaluator; a query whose prompt exceeds the context window of a backend is counted as a failure. The three hubs are reported as one benchmark by averaging their accuracies.
- **AppWorld** (Trivedi et al., 2024). Everyday tasks that an agent completes by writing code against the APIs of nine simulated apps. The environment's API documentation (457 API specifications) is a single source. We use the official splits: the 90 training tasks are the guidance tasks and the 168 tasks of test-normal are the test tasks. The built-in documentation app is removed for every method, so that API documentation is reachable only through each method's own access to the source. We report the official task goal completion (TGC).

Judging. On the QA benchmarks, an answer is judged correct if it makes the same key claims as the reference answer without contradicting it. The judge is GPT-5.6-Luna (OpenAI, 2026) at medium reasoning effort for every method, backend, and split. Accuracy is macro-averaged over the sources of a benchmark and then averaged over the three splits. APIBench and AppWorld use their official evaluators.

A.2 Counterpart Details

All counterparts share the backend, embedding model, and answer or agent prompt with `SourceLearn`, and otherwise follow their official or reproduction settings unless noted below.

- **Hybrid RAG**. Documents are split into sections and code into files and symbol-level units. The top-30 elements by BM25 (Robertson & Zaragoza, 2009) and by dense retrieval (Karpukhin et al., 2020) with text-embedding-3-large (OpenAI, 2024) are fused by reciprocal rank fusion (Cormack et al., 2009), and the top 8 are passed to the solver.
- **RAPTOR** (Sarthi et al., 2024). The official implementation with its default settings (100-token chunks, recursive clustering, cluster summaries written by the backend, collapsed-tree retrieval of the top 10 nodes within 3,500 tokens); the tree is built once per source and backend.
- **HippoRAG 2** (Gutiérrez et al., 2025). The official package in its reproduction setting, with open information extraction performed by the backend; the top 8 passages are passed to the solver so that the evidence budget matches Hybrid RAG.

- **AWM** (Wang et al., 2025). The authors' offline workflow-induction pipeline. Workflows are induced by the backend from the successful guidance-task experiences of a source and split, and the induced set is placed in the prompt of every test task of that split. Source evidence at test time comes from the same Hybrid RAG retriever, so both methods receive the same task-time source evidence while differing in what reusable information they retain from the guidance tasks.

A.3 Implementation Details

Backends. GPT-5.6-Luna and gpt-oss-120b are used at medium reasoning effort, and DeepSeek-V4.1-Flash at high effort on the QA benchmarks and medium effort on APIBench and AppWorld. Within a configuration, the same backend performs source-model construction and learning, the index construction of RAPTOR and HippoRAG 2, workflow induction for AWM, and task solving; text-embedding-3-large is the embedding model of every method.

`SourceLearn`. The initial source model M_0 is constructed once per source and backend. Self-Directed Source Learning runs one Inspect–Study–Consolidate cycle with $K = 20$ adaptive study actions, and the resulting M_S is shared by the three splits. Task-Guided Source Learning then processes the guidance tasks of a split in one pass, followed by recalibration under the learned representation policy. At task time, the solver receives the Hybrid RAG evidence for the task together with the activated source model under a budget of $B = 24$k tokens (Eq. (6)).

Ablation. The ablation of Section 5.3 uses two sources per QA benchmark, with the same guidance/test splits as the main results: SSA and VA from MultiDoc2Dial, pytest and conan from SWE-QA, and the scripts of *All About Eve* and *The Breakfast Club* from NarrativeQA. The *w/o Self-Directed Source Learning* variant applies Task-Guided Source Learning directly to M_0; *w/o Task-Guided Source Learning* corresponds to M_S; *w/o Cross-Task Representation Learning* corresponds to M_{local}; and *w/o Failure-Guided Local Refinement* recalibrates M_S under Π_1 without local refinement.

Task solving. On the QA benchmarks and APIBench, every method uses the same answer prompt with its own source context. On AppWorld, every method drives the official ReAct-style code agent with a limit of 50 steps; the documentation calls of the original prompt are replaced by a documentation channel that returns the method's own source context, so that methods differ only in how the API documentation is represented and accessed.

Hardware. GPT-5.6-Luna and DeepSeek-V4.1-Flash are accessed through their hosted APIs; gpt-oss-120b is served locally with vLLM on a server with six NVIDIA A100 80GB GPUs.

B Additional Results

All experiments in this section use GPT-5.6-Luna with the sources, guidance/test splits, and judge of the main results.

B.1 Robustness to Incomplete Retrieval

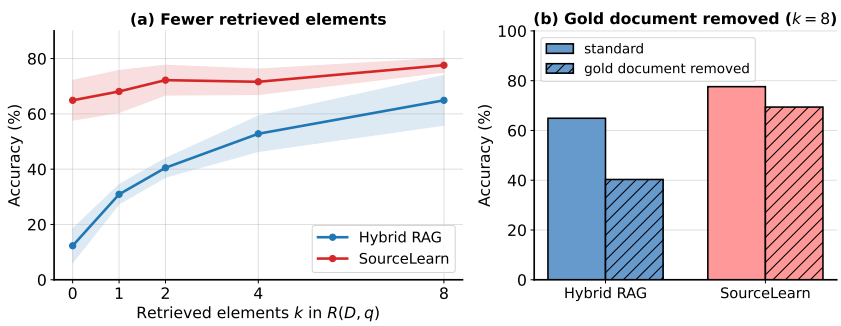

Figure 5: **Robustness to missing retrieved evidence** on MultiDoc2Dial with GPT-5.6-Luna. (a) Accuracy as the number of retrieved elements decreases; at $k = 0$, **`SourceLearn`** uses only the activated source model. (b) Accuracy when the gold document is removed from retrieval at $k = 8$.

At task time, `SourceLearn` combines retrieved evidence $R(D, q)$ with the activated source model (Eq. (7)). We isolate their roles by degrading retrieval on MultiDoc2Dial, whose questions provide gold-document labels (Figure 5). With no retrieved evidence, the source model alone performs comparably to Hybrid RAG with all eight retrieved elements, and `SourceLearn` degrades substantially more slowly as retrieval is reduced. Removing the gold document lowers Hybrid RAG by about 25 points but `SourceLearn` by only about 8. These results show that the learned source model provides reusable source understanding when task-local retrieval is incomplete, while remaining complementary to direct access to the source.

B.2 Comparison at Matched Context Budgets

The source model also provides more task-time context than the native retrieval counterparts. To control for this, we give each retrieval method the same per-question context budget used by `SourceLearn`, without changing its ranking or index. As shown in Tab. 2, `SourceLearn` remains best on three of four sources and second on Sphinx under matched context budgets. Thus, the amount of task-time context alone does not fully account for its performance; the representation analyses in Section 5.4 further examine what is learned beyond additional source content.

Table 2: **Retrieval counterparts at matched context budgets** (GPT-5.6-Luna; accuracy %). Each counterpart receives the same per-question context budget as `SourceLearn` by reading further from its own ranking. Best and second-best results are in **bold** and underlined.

Method	MultiDoc2Dial DMV	MultiDoc2Dial VA	SWE-QA Sphinx	APIBench HuggingFace
Hybrid RAG	<u>76.2</u>	70.2	62.7	48.6
RAPTOR	70.2	<u>73.8</u>	61.8	<u>52.9</u>
HippoRAG 2	71.4	71.5	**73.5**	51.4
`SourceLearn`	**77.0**	**77.8**	<u>72.5</u>	**56.7**
Evidence tokens	16.5k	18.2k	23.4k	17.1k

B.3 Grounding and Source Authority

A central principle of `SourceLearn` is that learning signals determine what should be reconsidered, while persistent updates are reconstructed from the authoritative source (Eq. (5)). We audit this property on one document, code, and API source by judging sampled units from M_0, M_S, and M_T against their cited source evidence, and by checking whether content removed during rewriting remains represented afterward (Tabs. 3 and 4). Direct contradictions remain rare ($\leq 1\%$), and 93–100% of dropped claims are preserved after rewriting. However, some Task-Guided units on the document and code sources are only partially supported or unsupported, showing that the learned representation is not itself a replacement for the source. This supports our design of keeping the original source authoritative and directly accessible while using the source model as accumulated, reusable understanding.

Table 3: **Grounding audit of source-model units** (%). Each unit is judged against its cited source elements (120 units per model): *supported* if all claims are entailed, *partial* if the main claim is supported but some detail is not, *unsupported* if the cited evidence does not establish the claim, and *contradicted* if it states otherwise.

Source	Model	Supported	Partial	Unsupported	Contradicted
MultiDoc2Dial (StudentAid)	M_0	79.2	19.2	1.7	0.0
	M_S	75.0	20.8	4.2	0.0
	M_T	63.3	22.5	13.3	0.8
SWE-QA (pytest)	M_0	89.2	10.8	0.0	0.0
	M_S	86.7	13.3	0.0	0.0
	M_T	65.0	31.7	3.3	0.0
APIBench (HuggingFace)	M_0	65.0	32.5	1.7	0.8
	M_S	76.7	18.3	5.0	0.0
	M_T	84.2	15.8	0.0	0.0

Table 4: **Preservation of dropped units** (%). Units removed by a learning stage are compared with the new model on the same entity (60 units per step). A claim is *preserved* if it remains represented after rewording, merging, or condensation.

Source	Dropped by	Preserved	Lost	Contradicted
MultiDoc2Dial (StudentAid)	Self-Directed ($M_0 \to M_S$)	98.3	1.7	0.0
	Task-Guided ($M_S \to M_T$)	98.3	0.0	1.7
SWE-QA (pytest)	Self-Directed ($M_0 \to M_S$)	93.3	5.0	1.7
	Task-Guided ($M_S \to M_T$)	100.0	0.0	0.0
APIBench (HuggingFace)	Self-Directed ($M_0 \to M_S$)	96.7	3.3	0.0
	Task-Guided ($M_S \to M_T$)	100.0	0.0	0.0

B.4 Transfer Beyond Guidance-Task Regions

Task-Guided Source Learning is intended to turn task experience into reusable source understanding rather than task-specific corrections. We therefore partition test questions according to whether their source region overlaps with any guidance task of the same split. As shown in Tab. 5, `SourceLearn` improves over Hybrid RAG by 14.9 points on seen regions and 12.1 points on unseen regions. The comparable gain on unseen regions shows that the benefit extends beyond source regions directly exposed by the guidance tasks, supporting the transfer of learned source understanding to new parts of the same source.

Table 5: **Accuracy (%) by overlap with guidance-task regions** (GPT-5.6-Luna). *Seen* regions overlap with at least one guidance task; *unseen* regions do not. Subscripts show changes relative to Hybrid RAG within each stratum.

Benchmark	Hybrid RAG		M_0		M_S		M_T (SourceLearn)	
	Seen	Unseen	Seen	Unseen	Seen	Unseen	Seen	Unseen
MultiDoc2Dial	64.2	65.3	$69.0_{\uparrow 4.8}$	$72.3_{\uparrow 7.0}$	$77.4_{\uparrow 13.2}$	$72.0_{\uparrow 6.7}$	$\mathbf{78.9}_{\uparrow 14.7}$	$\mathbf{76.7}_{\uparrow 11.4}$
SWE-QA	57.3	56.4	$70.3_{\uparrow 13.0}$	$61.1_{\uparrow 4.7}$	$66.2_{\uparrow 8.9}$	$64.6_{\uparrow 8.2}$	$\mathbf{72.9}_{\uparrow 15.6}$	$\mathbf{66.3}_{\uparrow 9.9}$
NarrativeQA	65.8	66.2	$74.2_{\uparrow 8.4}$	$74.0_{\uparrow 7.8}$	$77.2_{\uparrow 11.4}$	$78.4_{\uparrow 12.2}$	$\mathbf{80.1}_{\uparrow 14.3}$	$\mathbf{79.9}_{\uparrow 13.7}$
All QA	62.4	62.9	$71.6_{\uparrow 9.2}$	$69.6_{\uparrow 6.7}$	$73.3_{\uparrow 10.9}$	$72.2_{\uparrow 9.3}$	$\mathbf{77.3}_{\uparrow 14.9}$	$\mathbf{75.0}_{\uparrow 12.1}$

B.5 Number of Self-Directed Learning Cycles

`SourceLearn` uses one Inspect–Study–Consolidate cycle by default. We extend Self-Directed Source Learning to four cycles on two MultiDoc2Dial sources to test whether repeated study provides further gains (Tab. 6). Additional cycles continue to generate observations and expand the source model, but accuracy shows no consistent improvement beyond the first cycle. This suggests diminishing returns from repeated self-directed study, motivating the single-cycle setting used throughout the main experiments.

Table 6: **Self-Directed Source Learning over one to four cycles** on two MultiDoc2Dial sources (GPT-5.6-Luna). Cycle 0 is M_0 and cycle 1 is M_S. Observations are inspection notes produced in each cycle; accuracy is measured over the 60 questions of each source.

Source	Cycle	Units	Observations	Accuracy (%)
SSA	0 (M_0)	571	–	73.3
	1 (M_S)	615	95	75.0
	2	660	53	80.0
	3	683	54	75.0
	4	688	45	73.3
StudentAid	0 (M_0)	478	–	73.3
	1 (M_S)	741	204	75.0
	2	891	149	73.3
	3	989	120	71.7
	4	1,018	110	76.7

C Prompt Details

This section presents the prompts behind the main operators of `SourceLearn`: initial construction (Construct), model-conditioned inspection (Observe), adaptive study planning (PlanStudy), evidence consolidation with the grounding gate (Consolidate / GroundApply), and cross-task representation learning (Lesson / Aggregate). Prompts are quoted from the released code; every call

returns a fixed JSON schema, omitted here. The remaining prompts (source-requirement diagnosis, local refinement, and the preservation check) follow the same conventions and are included in the code.

C.1 Initial Source Model Construction (Construct)

Construction reads everything the source says about one entity and fills the slots of its representation m_e; the generic modeling instruction Π_0 states what deserves representation.

Entity-Centered Reading Prompt

You are reading everything the source says about ONE entity (product / module / topic) side by side, to build its profile — not to summarize it. Fill the slots:

- `attributes`: decision-relevant properties as name/value pairs (short `snake_case` names another entity of the same kind would also have); one element id each. Skip properties that decide nothing.
- `mechanisms`: how something works or a procedure's logic, when worth understanding; cite ≥ 1 element ids.
- `distinctions`: things a reader must not confuse.
- `decisive_details`: the few remaining facts that change a decision or behavior — one element id each. Rarity alone is NOT a reason; it must plausibly change an answer, boundary, exception, or decision. Skip anything merely retrievable.

Prefer fewer, structural entries. `element_id` / `element_ids` must be copied VERBATIM from the [bracketed] ids in the material; never invent labels. The raw source stays available at use time: do NOT store facts that a reader can simply look up. Keep constraints, mechanisms, shared rules, exceptions and distinctions.

Generic modeling instruction Π_0 Preserve the major entities of the source, their roles, the important relations between them, procedures, governing conditions and representative details, while avoiding redundant low-level content.

C.2 Model-Conditioned Source Inspection (Observe)

The same prompt drives every rereading of the source against the current model: inspection in Self-Directed Source Learning, the study actions, and recalibration under Π_1. A short note appended to it names the mode; the one for full-entity inspection is shown.

Source Inspection Prompt

You are studying a knowledge source to improve a compact SOURCE MODEL of it. The relevant part of the current model and the raw evidence for a learning target are shown. Record what the evidence teaches that the model does not yet explain well, as STUDY NOTES (temporary; they will be consolidated later into a rewrite of this region, so do not write model units now): each note = one observation the evidence establishes (a rule, condition, dependency, procedure step order, exception, distinction, or how two things connect), the evidence ids it rests on, its implication for the model (e.g. 'these three facts are stages of one procedure', 'unit X is missing the governing condition'), and the model units it concerns. Do not note look-up values the source answers on its own. Return an empty observations list (`learned=false`) when the model already explains the target.

Appended for full-entity inspection This is a REVIEW: the entity's complete source text is shown, so judge the model's portion about it as a whole — is every rule, condition, procedure step, exception and cross-reference the text establishes represented, correctly scoped and not fragmented? Reconstruct only what a reader would otherwise get wrong; look-up values stay in the source.

C.3 Adaptive Source Study Planning (PlanStudy)

Given the inspection observations, the planner selects the next batch Γ_b of Deepen and Connect actions, each with one focused study question (Eq. (10)).

Study Planning Prompt

You direct the self-study of an agent that maintains a compact SOURCE MODEL of a knowledge source. A full inspection of the source against the model has produced the OBSERVATIONS below: temporary notes on what each entity's source establishes that the model does not yet explain well. Before the model is consolidated, choose the {n} most consequential observations to resolve, ONE observation per action: DEEPEN one entity when its observation leaves a mechanism, governing condition, step order or exception unresolved; CONNECT two entities when an observation names a relation between them the model does not express (one constraining, triggering or feeding the other; a shared rule with different instances; a procedure or dependency spanning both). For each action copy the observation it resolves and write ONE focused question — the single thing that has to be understood to resolve that observation (for example: under what condition does Y switch to Z, and what follows?). Never ask for everything about an entity and never enumerate several aspects in one question; a question is an instrument for resolving one identified gap. Prefer entities and pairs not yet studied; never repeat an action already taken. Copy entity keys exactly as listed.

C.4 Evidence Consolidation and Grounding (Consolidate, GroundApply)

Consolidation rewrites one region from its current units, the accumulated notes, and their evidence; every proposed unit then passes the grounding gate, and the rewrite is committed only if the preservation check finds no lost meaning (Eq. (5)).

Region Rewrite and Grounding Prompts

Region rewrite Revise the SOURCE MODEL of one region of a knowledge source. You are given the region's current units, the study notes gathered by inspecting and questioning this region against the raw source, and the source evidence those notes rest on. The objective: the same compact model, revised at the SAME level of abstraction so that it also carries what the notes established — not a fuller account of the source. The source evidence is there to verify and ground what you write; it is NOT material to summarise: never restate the source passage by passage, never copy or closely paraphrase its sentences, never take over its voice (no 'you', 'your', 'we'). Write statements ABOUT the source in the model's own register, each unit ONE reusable semantic commitment (a rule with its governing condition, a mechanism with its consequence, a procedure with its step order, a distinction, an exception to a shared rule, a dependency between entities) — not one source excerpt, and not a replay of implementation steps unless the sequence itself is what matters. Preserve every source-supported meaning of the old units unless a stronger statement subsumes it; merge fragments that are stages of one procedure or instances of one rule; split a unit that mixes unrelated things; drop nothing the evidence supports. Never write a negative claim (something does not exist, is not supported, cannot be done) unless the source states it explicitly. Do not add look-up values the source answers on its own and do not keep isolated look-up details (a phone number, an office hour, a form number, a single value) solely for completeness. Entity cards (attribute grids) are kept as they are and are shown only as context: do not rewrite them. For each unit give its role, its scope (an entity scope copied from the old units, or the region), the old unit ids it absorbs, and the note numbers it draws on. Names, identifiers, numbers and conditions exactly as in the source; statements at most {chars} characters; declarative.

Grounding gate Decide whether the candidate statement is SUPPORTED by the source excerpts shown (entailed by them, not merely consistent). If supported, return the ids of the excerpts that establish it; if any part is not established, `supported=false` and list the unsupported parts. Source-bound rule for NEGATIVE claims: a statement saying that something does NOT exist, is NOT supported, or CANNOT be done is supported only if an excerpt explicitly states that denial; a feature merely absent from the excerpts is NOT evidence that it is absent from the source.

Preservation check A region of a source model was rewritten. Below are the OLD units' meanings (numbered) and the NEW region. Return the numbers of old meanings that the new region no longer represents — a meaning counts as represented when the new region states it, entails it, or expresses it through a broader rule, procedure, or abstraction that covers it (merging, factoring and abstracting are fine; a lost condition, exception, value, entity or step is not).

C.5 Cross-Task Representation Learning (Lesson, Aggregate)

Each guidance task yields at most one representation lesson ℓ_t, and the lessons are aggregated into the representation policy Π_1 (Eq. (13)), whose rendered text is appended to every writer prompt during recalibration (Eq. (14)).

Representation Lesson and Policy Aggregation Prompts

Representation lesson A real task was answered with a SOURCE MODEL (a compact, persistent representation of a knowledge source) next to raw retrieval. You are shown what the task required from the source (the excerpts the reference rests on and what each establishes), the model units the system had about it, and the outcome. State the ONE general representational preference this experience suggests for building a reusable model of this source: what kind of information, at what granularity, which distinction, relation or condition deserves explicit representation — and when. A solved task shows what was used and deserves to stay represented; a failed task additionally shows what the current representation lacked or compressed away. Kinds: preserve (a type of information to keep), granularity (level of detail), distinguish (things not to conflate), relate (cross-entity relations), condition (conditions, defaults, exceptions). The lesson must contain NO source-specific identifier (class, function, field, option, file, product or value names) and must NOT restate this task's topic: name the CLASS of source content the needed knowledge belongs to (e.g. 'the options of a configuration file', 'public fields of configurable classes', 'the steps of a lifecycle procedure', 'exceptions to a shared rule') and what about it to represent and when, so that the same preference applies to every other instance of that class elsewhere in this source. Return lesson null when the task required nothing beyond generic reading of the source.

Policy aggregation You maintain the REPRESENTATION POLICY of a source model: a short numbered list of preferences saying what kinds of information, at what granularity, and which distinctions, relations and conditions deserve explicit representation when this knowledge source is modelled for use. You are given the current task-derived items (if any) and new lessons, one per task, each marked solved or not. Consolidate the lessons into general preferences: merge lessons that express the same preference, keep each preference actionable for a writer (what to preserve / distinguish / relate, at what level of detail, under which condition), and cite for each the lesson numbers it consolidates. A preference may restate or refine an existing item (give `keeps_item`). No source-specific identifiers, and no single feature of the source: when several lessons concern one feature, state the preference for the CLASS of content that feature belongs to (e.g. 'the syntax rules of any addressing or naming scheme', 'the options of any configuration surface'), so that it applies to other instances in this source. At most {`n`} preferences, the most broadly supported first. Do not invent preferences no lesson supports.

Policy note appended to writer prompts Representation policy for this source model (what kinds of information, at what granularity and which relations deserve explicit representation; apply it when deciding what to write, keep, merge or leave to the raw source): {`policy`}