TASKMATCH: LEARNING TASK-AWARE RELEVANCE BEYOND SEMANTIC SIMILARITY FOR AGENT HARNESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern LLM agent harnesses augment the underlying model with reusable components such as external skills and procedural memories, making effective retrieval increasingly important as these repositories grow. However, existing retrieval pipelines often rank stored skills or memories by semantic similarity, which measures how similar a skill or memory appears to the current retrieval query but does not necessarily reflect how useful it is for completing the task. We introduce TaskMatch, a lightweight plug-and-play model that learns task-aware relevance from offline examples labeled as required, supportive, or irrelevant. Using the fixed query and skill and memory representations already available in the host retrieval pipeline, TaskMatch combines direct feature-wise matching with nonlinear interactions while leaving candidate generation and downstream agent execution unchanged. We instantiate the same architecture separately for two representative harness components—skill retrieval and procedural-memory retrieval—with fewer than 0.30M trainable parameters per model and no additional online LLM calls. Across both components, TaskMatch consistently improves retrieval ranking quality and downstream task execution. It raises SKILLDAG success on ALFWorld from 67.10% to 81.43%, improves MEMP Proceduralization success from 81.34% to 90.30%, and increases TravelPlanner hard-constraint satisfaction from 64.06 to 76.19. Our code is available at `https://anonymous.4open.science/r/learned-relevance-scorer-agent-retrieval-4683/`.

1 INTRODUCTION

Large language model (LLM) agent harnesses increasingly augment base models with retrievable tools (Schick et al., 2023; Qin et al., 2024; Hao et al., 2023; Yuan et al., 2024), reusable skills (Wang et al., 2023; Liu et al., 2026), workflows (Wang et al., 2024), and procedural memories (Fang et al., 2026; Sarch et al., 2024). These components allow agents to reuse capabilities and prior experience rather than repeatedly planning and exploring from scratch. As these repositories grow, retrieval becomes a critical bottleneck: large skill libraries shift the challenge toward selecting the right skills, while procedural-memory systems depend on retrieving experience that is actually useful for the current task (Bai et al., 2026; Fang et al., 2026; Sarch et al., 2024). This makes effective selection increasingly important as agent harnesses scale (Chen et al., 2024b; Shi et al., 2025; Du et al., 2024; Liu et al., 2026). At each retrieval event, we refer to the skill or memory entries considered for ranking as *candidates*.

A natural and widely adopted approach is to represent the current retrieval query and each candidate skill or memory as embedding vectors and rank the candidates according to their cosine similarity (Reimers & Gurevych, 2019; Gao & Callan, 2022; Fang et al., 2026; Bai et al., 2026). Let q denote the text query used for a retrieval event, and let c denote one candidate memory/skill entry in the external repository. Let $E(\cdot)$ denote the fixed text encoder used by the original retrieval system, which maps a text input into a d-dimensional embedding vector. The cosine-based relevance score is then

$$s_{\cos}(q, c) = \cos(E(q), E(c)). \tag{1}$$

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.

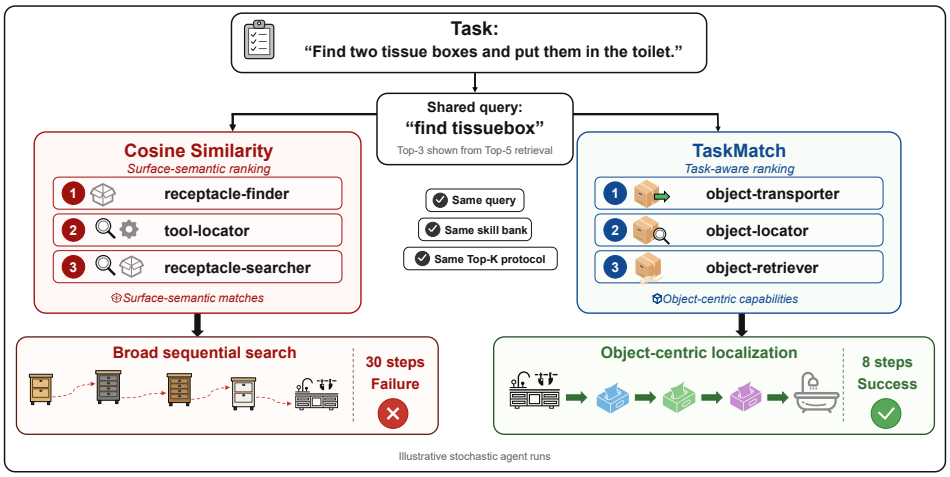

Figure 1: A representative ALFWorld case illustrating the mismatch between semantic similarity and task relevance. For the shared query "find tissuebox," cosine similarity ranks receptacle-oriented skills highly, whereas TaskMatch promotes object transportation, localization, and retrieval capabilities that better match the required operations. The figure shows the Top-3 entries from the Top-5 ranking returned for this query. The corresponding stochastic agent runs end in a 30-step failure and an 8-step success.

Here, $\cos(\cdot,\cdot)$ denotes cosine similarity, and a larger $s_{\cos}(q,c)$ indicates closer query and candidate embeddings. Because both $E(\cdot)$ and the scoring rule are fixed, this ranking cannot learn cases in which a less similar candidate is nevertheless more useful for the task. This fixed-score limitation appears in both cosine-based memory retrieval and structure-aware skill systems whose final ranking still contains similarity-based components (Fang et al., 2026; Bai et al., 2026; Liu et al., 2026; Wang et al., 2026a; Goulart et al., 2026).

This mismatch becomes especially important when semantic similarity differs from what execution actually requires. A prerequisite skill may be less textually similar than a redundant skill, while memories with similar wording may encode different operations, object counts, or preconditions (Liu et al., 2026; Bai et al., 2026; Fang et al., 2026; Fu et al., 2024; Cao et al., 2026). In the held-out case shown in Figure 1, the query "find tissuebox" ranks receptacle-oriented skills highly under cosine similarity, even though the tissue box is itself the object that needs to be located, retrieved, and transported. TaskMatch instead promotes object-centric capabilities. This example motivates learning task-aware relevance rather than relying on semantic similarity alone.

We refer to the original agent and its retrieval pipeline as the host system. At each retrieval event, the host system provides a query q, which may be the full task or a shorter operation-specific request; for example, Figure 1 uses "find tissuebox" for the full task "Find two tissue boxes and put them in the toilet." TaskMatch neither generates nor rewrites these queries: given the same query, candidate set, and fixed representations, it changes only candidate scoring and ranking.

TaskMatch learns from offline query–candidate pairs labeled as required, supportive, or irrelevant, including hard cases where semantic overlap conflicts with what the task requires. We distill these labels into a lightweight content-based scoring function

$$s_\theta(q,c) = f_\theta(E(q), E(c)), \tag{2}$$

where f_θ is a trainable interaction model with parameters θ, and $s_\theta(q,c)$ is its predicted task-aware relevance score. To move beyond fixed cosine similarity, TaskMatch combines direct feature-wise matching with nonlinear query–candidate interactions. Its interaction design draws on Neural Collaborative Filtering (NCF) (He et al., 2017), but operates on fixed content representations rather than learned user–item identifiers, allowing it to score unseen queries and entries without identifier memorization. In this sense, **retrieval shifts from passive similarity matching to learned, task-aware recommendation.**

At inference time, TaskMatch reranks candidates returned by the host retriever, optionally combining its score with the original retrieval score. All supervision is constructed offline, so online scoring requires no judge, learned identifier lookup, or additional LLM call. This makes TaskMatch a lightweight plug-in for vector-, graph-, and memory-based retrieval pipelines (Goulart et al., 2026; Zheng et al., 2026; 2024b; Nogueira & Cho, 2019).

We evaluate TaskMatch on two representative retrieval components of agent harnesses—skills and procedural memories—covering both structure-aware and similarity-based pipelines. Across both settings, learned task-aware relevance consistently improves the original retrieval mechanism.

Our main contributions are summarized as follows:

- **Beyond fixed similarity for agent retrieval.** We show that semantic similarity does not necessarily reflect the operational relevance of skills or procedural memories, and formulate retrieval as content-based task–candidate relevance learning.
- **TaskMatch.** We distill graded task–candidate supervision into an NCF-inspired scorer with only 0.15–0.30M trainable parameters. It operates on the host system's existing queries and candidate representations, requiring no additional online LLM call or planning module.
- **Improved retrieval and task execution.** The scorer improves both skill and procedural-memory retrieval, raising SKILLDAG ALFWorld success from 67.10% to 81.43% and MEMP Proceduralization success from 81.34% to 90.30%.

2 RELATED WORK

2.1 SKILL AND TOOL RETRIEVAL FOR LLM AGENTS

Structural approaches such as GOS and SKILLDAG further model dependencies, prerequisites, conflicts, and specializations that semantic matching alone may miss (Liu et al., 2026; Bai et al., 2026). Recent concurrent work improves other stages of the skill-use pipeline: SKILLZIP compresses skill libraries into reusable procedural graph units, Capability Pages enrich skill documents with contrastive capability boundaries, SKILLTRACE retrieves executable skill compositions through query–skill graph traversal, and SKILLALIGNER adapts already retrieved skills before execution (Tan et al., 2026; Wang et al., 2026b; Yao et al., 2026; Li et al., 2026a). These approaches modify the skill representation, retrieval structure, or downstream use of retrieved skills. TaskMatch instead leaves these components unchanged and learns only the task–candidate scoring function, making it complementary to such advances while also extending the same formulation to procedural-memory retrieval.

2.2 AGENT MEMORY AND PROCEDURAL MEMORY RETRIEVAL

External memory enables LLM agents to reuse past experience rather than repeatedly solving similar tasks from scratch. Existing methods distill trajectories into reusable workflows, guidelines, exemplars, manuals, or structured memories, and retrieve them to support subsequent execution (Wang et al., 2024; Fu et al., 2024; Zheng et al., 2024a; Chen et al., 2024a; Tan et al., 2025; Tang et al., 2025; Xu et al., 2025). Among them, MEMP systematically studies procedural-memory construction, retrieval, and update, using task-based cosine matching and feature-based similarity to retrieve relevant experience (Fang et al., 2026), while REMEMBER ME, REFINE ME further emphasizes dynamic refinement of procedural memory from execution experience (Cao et al., 2026). These methods demonstrate the value of retrieving prior experience, but memory selection still commonly depends on semantic similarity or predefined retrieval heuristics. TaskMatch focuses specifically on this stage: without changing how memories are constructed, stored, or updated, it learns task–candidate relevance from offline supervision and uses it in place of a fixed matching rule.

2.3 FROM FIXED SIMILARITY TO LEARNABLE INTERACTION

A common retrieval paradigm encodes queries and candidates into a shared space and ranks them with fixed similarity functions such as inner product or cosine similarity (Reimers & Gurevych, 2019; Gao & Callan, 2022). While simple and efficient, such scoring largely reduces task–candidate compatibility to geometric similarity. Neural information retrieval instead learns richer query–candidate interactions, as in DRMM, MatchPyramid, K-NRM, CEDR, and ColBERT (Guo et al.,

2016; Pang et al., 2016; Xiong et al., 2017; MacAvaney et al., 2019; Khattab & Zaharia, 2020), while neural reranking further learns pairwise relevance functions directly (Nogueira & Cho, 2019).

Recent agent-retrieval methods similarly use learned rerankers or task-aware scoring to improve tool and skill selection beyond fixed embedding similarity (Zheng et al., 2024b; 2026; Goulart et al., 2026; Chen et al., 2026; Wang et al., 2026a). Some rely on substantially larger models; for example, SKILLROUTER uses a 0.60B reranker after a 0.60B retrieval encoder (Zheng et al., 2026), whose cost can accumulate across repeated retrieval events in long-horizon agents. TaskMatch instead learns task–candidate relevance with a 0.15–0.30M interaction model over frozen host representations. Inspired by Neural Collaborative Filtering (He et al., 2017), it combines explicit feature matching with nonlinear task–candidate interactions without candidate IDs, and applies the same formulation to both skill and procedural-memory retrieval.

3 METHODOLOGY

3.1 PROBLEM FORMULATION

We study task-aware relevance for two representative retrieval components of agent harnesses: skills and procedural memories. Let q_t denote the query issued by the host system at retrieval step t, which may be the original task or a shorter operation-specific request. Given an external repository $\mathcal{C} = \{c_1, \ldots, c_N\}$, where each c_i is a skill or procedural memory, the host retriever returns

$$\mathcal{C}_t = \mathcal{R}(q_t, \mathcal{C}), \qquad |\mathcal{C}_t| = M_t. \tag{3}$$

Here, M_t is the number of candidates returned at retrieval step t and $\mathcal{R}$ can be any existing retrieval mechanism, including vector-based memory retrieval or structure-aware skill retrieval (Fang et al., 2026; Liu et al., 2026; Bai et al., 2026). It may also provide an original retrieval score $s_{\text{base}}(q_t, c)$, such as cosine similarity or a structure-derived score.

Let $E(\cdot)$ denote the fixed text encoder used by the host retrieval pipeline. For the cosine baseline, candidate relevance is scored by

$$s_{\cos}(q_t, c) = \cos(E(q_t), E(c)). \tag{4}$$

TaskMatch instead learns a task-aware relevance score $s_\theta(q_t, c)$ from the same fixed representations, as detailed in Section 3.2.

From training tasks and candidates, we construct the graded relevance target:

$$g(q, c) = \begin{cases} 2, & c \text{ is required for solving } q, \\ 1, & c \text{ is supportive but not indispensable}, \\ 0, & c \text{ is irrelevant or potentially harmful.} \end{cases} \tag{5}$$

We learn $s_\theta(q, c) \in \mathbb{R}$ from this supervision. Let $K \leq M_t$ denote the number of candidates selected for the downstream agent. At retrieval step t,

$$\widehat{\mathcal{C}}_t = \mathrm{TopK}_{c \in \mathcal{C}_t} \, s_\theta(q_t, c), \qquad |\widehat{\mathcal{C}}_t| = K. \tag{6}$$

Here, TopK returns the K candidates with the largest TaskMatch scores.

Shared-query comparison. At every retrieval event, the baseline and TaskMatch receive the same host-issued query, candidate set, and fixed representations; only candidate scoring and ranking are changed. Figure 2 summarizes the pipeline.

3.2 TASKMATCH: LIGHTWEIGHT TASK-AWARE RELEVANCE MODELING

Given $\mathcal{C}_t$, TaskMatch learns task-aware relevance beyond fixed cosine similarity while remaining lightweight enough for repeated retrieval. It reuses the fixed representations already available in the host pipeline:

$$\mathbf{x}_{q_t} = E(q_t), \qquad \mathbf{x}_c = E(c), \qquad \mathbf{x}_{q_t}, \mathbf{x}_c \in \mathbb{R}^d, \tag{7}$$

The encoder $E(\cdot)$ remains fixed, allowing candidate representations to be precomputed and cached.

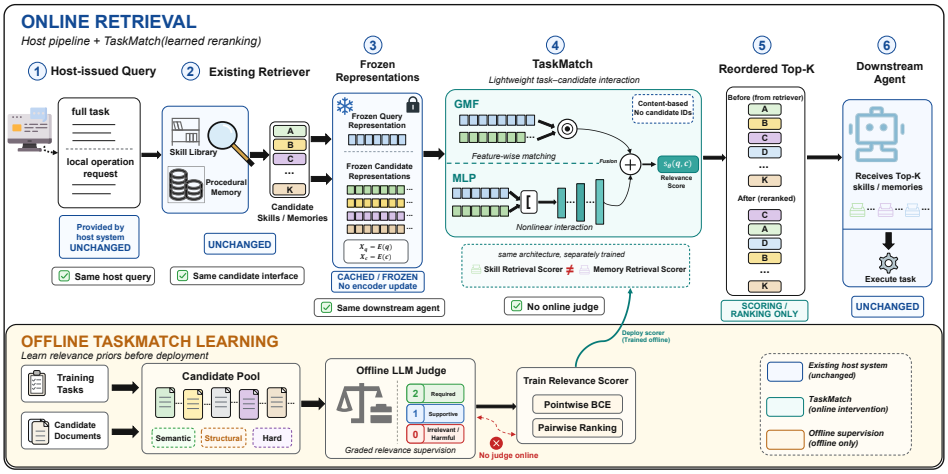

Figure 2: Overview of TaskMatch. Offline graded relevance supervision trains TaskMatch. At inference, TaskMatch reranks candidates using fixed representations while leaving the host retriever and downstream agent unchanged.

To model task-aware relevance, we use two complementary interaction paths inspired by Neural Collaborative Filtering (NCF) (He et al., 2017): a Generalized Matrix Factorization (GMF) branch for element-wise matching and a Multilayer Perceptron (MLP) branch for nonlinear interaction. Unlike ID-based recommendation, both branches operate on fixed query and candidate representations.

$$\mathbf{z}^G_{q_t,c} = (\mathbf{W}^G_q \mathbf{x}_{q_t}) \odot (\mathbf{W}^G_c \mathbf{x}_c), \tag{8}$$

$$\mathbf{z}^M_{q_t,c} = \mathrm{MLP}\big([\mathbf{W}^M_q \mathbf{x}_{q_t}; \mathbf{W}^M_c \mathbf{x}_c]\big). \tag{9}$$

Here, $\mathbf{W}^G_q$ and $\mathbf{W}^G_c$ are the query and candidate projection matrices of the GMF branch, while $\mathbf{W}^M_q$ and $\mathbf{W}^M_c$ are the corresponding projection matrices of the MLP branch. The operator $\odot$ denotes element-wise multiplication, and $[\cdot;\cdot]$ denotes vector concatenation, where $d_G = 16$ and $d_M = 32$ are the projection dimensions of the two branches. The final relevance score combines both interaction signals:

$$s_\theta(q_t, c) = \mathbf{w}^\top [\mathbf{z}^G_{q_t,c}; \mathbf{z}^M_{q_t,c}] + b, \qquad \hat{r}_{q_t,c} = \sigma(s_\theta(q_t, c)). \tag{10}$$

Here, $\mathbf{w}$ and b are the learnable fusion weight and bias, respectively, and $\sigma(\cdot)$ denotes the sigmoid function. The scalar $s_\theta(q_t, c)$ is the ranking score, while $\hat{r}_{q_t,c} \in (0, 1)$ is its sigmoid-normalized relevance prediction. All trainable parameters of TaskMatch are collectively denoted by θ. TaskMatch combines direct feature-wise and nonlinear interactions without learned query or candidate identifiers. Skill and procedural-memory retrieval use the same architecture but separately trained TaskMatch models.

3.3 Distilling Offline Relevance Priors

For each training query q, let $\mathcal{P}(q)$ denote the candidate pool used to construct offline supervision:

$$\mathcal{P}(q) = \mathcal{P}_{\mathrm{sem}}(q) \cup \mathcal{P}_{\mathrm{struct}}(q) \cup \mathcal{P}_{\mathrm{hard}}(q), \tag{11}$$

Here, $\mathcal{P}_{\mathrm{sem}}(q)$ contains semantically retrieved candidates, $\mathcal{P}_{\mathrm{struct}}(q)$ contains structurally related candidates when available, and $\mathcal{P}_{\mathrm{hard}}(q)$ contains hard negatives that are semantically similar but execution-incompatible. An LLM judge assigns grades $g(q, c) \in \{2, 1, 0\}$ for *required*, *supportive*, and *irrelevant/harmful*, respectively; ambiguous cases are discarded. Hard negatives expose cases where semantic similarity conflicts with task requirements. Full prompts and construction details are provided in Appendix B.

Composite training tasks are also rendered as operation-specific task views that approximate the granularity of the local retrieval queries issued by the host system during execution; the labels further

constitute relevance supervision rather than causal effect estimates. Both points are detailed in Appendix B.

Let $\mathcal{D}$ denote the set of labeled training query–candidate pairs. For each $(q, c) \in \mathcal{D}$, we normalize the graded target as $t_{q,c} = g(q, c)/2 \in \{0, \frac{1}{2}, 1\}$ and optimize the pointwise binary cross-entropy objective

$$\mathcal{L}_{\text{point}} = -\frac{1}{|\mathcal{D}|} \sum_{(q,c)\in\mathcal{D}} \left[t_{q,c} \log \hat{r}_{q,c} + (1 - t_{q,c}) \log(1 - \hat{r}_{q,c}) \right]. \tag{12}$$

Since retrieval ultimately depends on relative ordering, let $\mathcal{Q}$ denote the set of preference triples (q, c_i, c_j) satisfying $g(q, c_i) > g(q, c_j)$. We optimize

$$\mathcal{L}_{\text{pair}} = \frac{1}{|\mathcal{Q}|} \sum_{(q,c_i,c_j)\in\mathcal{Q}} \log\left(1 + \exp[-\left(s_\theta(q, c_i) - s_\theta(q, c_j)\right)]\right), \tag{13}$$

which explicitly encourages $\text{required} > \text{supportive} > \text{irrelevant}/\text{harmful}$. The final objective is

$$\mathcal{L} = \mathcal{L}_{\text{point}} + \lambda \mathcal{L}_{\text{pair}}. \tag{14}$$

Here, $\lambda \geq 0$ controls the relative weight of the pairwise ranking objective. Following NeuMF (He et al., 2017), we pretrain the GMF and MLP branches independently before jointly optimizing the full TaskMatch model. Exact initialization, optimization hyperparameters, and loss coefficients are reported in the Appendix A for reproducibility.

3.4 Integration with Existing Retrievers

TaskMatch uses the task query already produced by the host system and is inserted at the candidate-scoring stage. Given the candidate set $\mathcal{C}_t$ returned by the base retriever, it reorders the surfaced candidates using $s_\theta(q_t, c)$ as defined in Section 3.2, with final Top-K selection following Eq. 6. The base retriever remains responsible for candidate recall and any retriever-specific structure. In deployments that retain the base score, the TaskMatch and base scores are normalized and combined with a fixed coefficient selected on the development split. This calibration changes neither the TaskMatch input nor the online information budget.

Since TaskMatch operates on frozen task and candidate representations, its parameter count scales linearly with the representation dimension; the resulting TaskMatch model sizes and the effect of the inherited embedding dimension are reported in Appendix C.5. Both settings use the same interaction architecture.

4 Experiments

We evaluate TaskMatch by answering three questions:

RQ1 Does TaskMatch improve intrinsic retrieval quality for both skills and procedural memories?

RQ2 Do better retrieval results translate into higher end-to-end task performance?

RQ3 Which components drive the gains, and what training and inference overhead does TaskMatch introduce?

4.1 Experimental Setup

Benchmarks. We evaluate two retrieval settings. For **skill retrieval**, we use SkillsBench-1000 (Li et al., 2026b) and ALFWorld with the 37-skill library used by SkillDAG (Shridhar et al., 2020; Bai et al., 2026). SkillsBench evaluates skill selection from a large repository, while ALFWorld evaluates long-horizon embodied task execution. For **procedural-memory retrieval**, we follow the MemP evaluation framework on ALFWorld and TravelPlanner (Fang et al., 2026; Shridhar et al., 2020; Xie et al., 2024), covering Script, Trajectory, and Proceduralization memories.

Table 1: Intrinsic skill-retrieval results. TaskMatch is applied to skills produced by SkillDAG. Higher is better for all metrics.

		SkillsBench-1000			ALFWorld-37		
Model	**Method**	Ret@5↑	Ret@1↑	MRR↑	Ret@5↑	Ret@1↑	MRR↑
MiniMax-M2.7	Vector Skills	10.80	3.60	5.80	68.60	37.90	49.20
	GoS	65.50	50.60	57.30	86.40	56.40	67.90
	SkillDAG	78.20	66.70	71.30	92.10	57.90	71.10
	SkillDAG + TaskMatch	**84.13**	**76.98**	**79.70**	**99.40**	**86.03**	**92.50**
GPT-5.2-Codex	Vector Skills	10.80	3.60	5.80	68.60	37.90	49.20
	GoS	65.50	50.60	57.30	86.40	56.40	67.90
	SkillDAG	75.90	70.10	73.00	85.10	60.40	69.60
	SkillDAG + TaskMatch	**80.49**	**70.73**	**74.23**	**100.00**	**74.52**	**86.45**

Baselines and Models. For skill retrieval, we compare against Vanilla Skills, Vector Skills, Graph-of-Skills (GoS) (Liu et al., 2026), and SkillDAG (Bai et al., 2026), using MiniMax-M2.7 and GPT-5.2-Codex. TaskMatch is added at SkillDAG's scoring interface: it receives the same task queries and skill representations while leaving graph construction and candidate generation unchanged. For procedural memory, we use GPT-4o and retain the memory representations and execution pipeline of MemP (Fang et al., 2026), changing only the score used to rank the same memories. We focus on controlled integrations with two representative host retrievers—structure-aware SkillDAG and similarity-based MemP—to isolate the effect of TaskMatch scoring across substantially different retrieval mechanisms; see Appendix C.4 for a discussion of concurrent methods. The skill and memory settings use **separately trained TaskMatch models**; they share the same architecture and training procedure but no parameters.

Retrieval Protocol. For intrinsic memory retrieval, we consider two settings. In the **full-pool** setting, Query cosine and TaskMatch rank the same memory pool, enabling a controlled comparison between fixed similarity and learned scoring. In the **cascade** setting, cosine retrieval first reduces the pool to 50 memories and TaskMatch selects the final Top-10. We use the latter as a practical efficiency–recall configuration.

Metrics and Implementation. For intrinsic retrieval, we report Ret@K, which measures whether the Top-K results contain a ground-truth relevant candidate, and mean reciprocal rank (MRR), which measures how early the first ground-truth candidate appears in the ranking. For procedural-memory retrieval, we additionally report NDCG@10 for overall Top-10 ranking quality, Precision@10 for the proportion of relevant memories among the Top-10 results, and Harm@10 for the proportion of incompatible or unhelpful memories retrieved. For end-to-end evaluation, SkillsBench reports benchmark reward, ALFWorld reports task success rate and average execution steps, and TravelPlanner reports Common Sense (#CS), Hard Constraint (#HC), and step metrics. Higher is better except for Harm@10 and Steps. Full metric definitions are provided in Appendix C.3, and TaskMatch implementation details are provided in Appendix A.

4.2 Intrinsic Retrieval Quality

We first isolate retrieval quality from downstream agent execution.

RQ1 Summary. Table 1 and Table 2 show that TaskMatch consistently improves intrinsic ranking over the corresponding host retrievers. With MiniMax-M2.7, skill Ret@1 increases from 66.70 to 76.98 on SkillsBench and from 57.90 to 86.03 on ALFWorld, and the improvements persist with GPT-5.2-Codex. Under the matched ALFWorld memory setting, Ret@1 rises from 73.84 to 91.04, and on TravelPlanner Top-K coverage improves substantially while Harm@10 drops from 88.77 to 66.59. The gains recur across all four evaluation settings, spanning both structure-aware skill retrieval and similarity-based procedural-memory retrieval. Per-metric details, including the optional cascade variant, are reported in Appendix D.4.

Table 2: Intrinsic procedural-memory retrieval. Full-pool rows score the complete memory pool, while cascade rows first retrieve 50 memories with cosine similarity and then apply TaskMatch only to this subset. The final row of each dataset reports the optional cascade variant.

Dataset	Retriever	Pool Size	Ret@1↑	Ret@5↑	Ret@10↑	MRR↑	NDCG@10↑	Prec.@10↑	Harm@10↓
ALFWorld	Random	300	7.36	26.84	39.76	15.70	7.33	7.44	–
	Query cosine	300	73.84	95.70	98.92	82.98	71.81	58.69	–
	TaskMatch	300	**91.04**	**98.57**	**99.28**	**93.82**	**84.45**	**71.73**	–
	Query cosine + TaskMatch	50	96.06	100.00	100.00	97.79	94.68	80.54	–
TravelPlanner	Random	519	0.44	2.56	4.78	1.41	1.61	1.66	98.34
	Query cosine	519	7.78	16.22	18.78	12.23	11.00	11.23	88.77
	TaskMatch	519	**8.78**	**94.67**	**98.67**	**40.36**	**49.96**	**33.41**	**66.59**
	Query cosine + TaskMatch	50	34.11	64.67	64.67	48.60	33.99	16.62	83.38

Table 3: End-to-end skill-selection performance. SkillsBench reports reward and ALFWorld reports task success rate.

Model	Method	SkillsBench Reward↑	ALFWorld Success (%)↑
MiniMax-M2.7	Vanilla Skills	17.20	47.10
	Vector Skills	10.40	50.70
	GoS	18.70	54.30
	SkillDAG	27.30	67.10
	SkillDAG + TaskMatch	**29.21**	**81.43**
GPT-5.2-Codex	Vanilla Skills	27.40	89.30
	Vector Skills	21.50	92.90
	GoS	34.40	93.60
	SkillDAG	36.80	93.60
	SkillDAG + TaskMatch	**38.33**	**96.43**

4.3 End-to-End Agent Performance

We next test whether the retrieval gains translate into better task execution.

RQ2 Summary. The intrinsic retrieval gains generally translate into stronger downstream execution. With MiniMax-M2.7, adding TaskMatch to SkillDAG raises ALFWorld success from 67.10% to **81.43%**, and the improvement persists with GPT-5.2-Codex (93.60% to **96.43%**); SkillsBench reward increases from 27.3 to 29.21. For procedural memory, Proceduralization raises ALFWorld Test SR from 81.34% to **90.30%** while reducing average steps from 13.30 to **12.22**, and on TravelPlanner it improves #HC from 64.06 to **76.19**. The same scoring formulation therefore complements both structure-aware skill retrieval and cosine-based procedural-memory retrieval without redesigning the underlying retrieval mechanisms. Per-setting details are reported in Appendix D.5.

Script memory provides a counterexample to a strictly linear relationship between ranking quality and agent success. Table 5 provides a second example: the TaskMatch (MLP) variant obtains the highest Ret@1 and MRR under the deployed protocol, while the full TaskMatch model achieves the highest task success.

4.4 Ablation and Efficiency

We finally examine which components contribute to the gains and how much additional computation TaskMatch introduces. We first evaluate component variants using intrinsic retrieval metrics and report the complete architectural ablation in Appendix D.2, together with the efficiency measurements in Appendix C.6. We then conduct end-to-end agent evaluation for the principal architectural variants to verify whether retrieval-level improvements translate into better task execution.

End-to-End Ablation. Table 5 complements the analysis in Appendix D.2 with matched local evaluations of the TaskMatch variants, while the SkillDAG row is retained as the published reference. Unlike the pure full-pool analysis, the intrinsic columns here report each branch under its deployed calibrated ranking protocol. Both single-branch TaskMatch variants improve task success over SkillDAG, reaching 72.14% for TaskMatch (GMF) and 75.71% for TaskMatch (MLP), while

Table 4: End-to-end procedural-memory results. Query-cosine baselines are our reruns under the same MemP evaluation pipeline. The memory representation is unchanged; only memory scoring differs.

		ALFWorld			TravelPlanner		
Memory	**Retriever**	Dev Success↑	Test Success↑	Avg. Steps↓	#CS↑	#HC↑	Avg. Steps↓
NoMemory	–	39.28	42.14	23.76	71.93	12.88	17.84
Script	Query cosine	71.43	84.33	**13.43**	76.32	12.14	**16.58**
	Query cosine + TaskMatch	**78.57**	**85.07**	13.85	**76.81**	**13.81**	16.66
Trajectory	Query cosine	79.29	80.60	13.72	81.39	66.25	15.83
	Query cosine + TaskMatch	**80.71**	**84.33**	**13.63**	**81.53**	**70.13**	**15.44**
Proceduralization	Query cosine	85.71	81.34	13.30	80.69	64.06	15.73
	Query cosine + TaskMatch	**89.29**	**90.30**	**12.22**	**80.76**	**76.19**	**14.66**

Table 5: Focused architectural ablation on ALFWorld with MiniMax-M2.7. The intrinsic columns use each branch's deployed calibrated ranking protocol and are not directly comparable to the pure full-pool scores reported in Appendix D.2.

Variant	Ret@1↑	MRR↑	Success (%)↑
SkillDAG	57.90	71.10	67.10
SkillDAG + TaskMatch (GMF)	76.85	87.01	72.14
SkillDAG + TaskMatch (MLP)	89.22	93.91	75.71
SkillDAG + TaskMatch (Full)	**86.03**	**92.50**	**81.43**

the full TaskMatch model achieves the highest success rate of 81.43%. Notably, TaskMatch (MLP) obtains slightly higher Ret@1 and MRR than the full TaskMatch model under this protocol, yet its task success is 5.72 points lower. This discrepancy highlights a retrieval–execution gap: intrinsic ranking metrics evaluate whether highly relevant candidates are placed near the top of the ranking, but they do not fully characterize how useful the resulting retrieved set is for downstream decision making. In particular, end-to-end success depends not only on the top-ranked candidate, but also on the composition of the retrieved candidates and on how the agent uses them during multi-step execution. The full TaskMatch model may therefore produce rankings that are more helpful for execution even when its Ret@1 is slightly lower, by balancing direct feature-wise matching with nonlinear query–candidate interactions. More broadly, these results show that improvements in intrinsic retrieval metrics should be validated with end-to-end agent evaluation rather than treated as a direct proxy for task success.

5 Conclusion

During agent execution, the host system naturally issues retrieval queries that reflect the current task state. We leave this query-generation process unchanged and study how candidates should be scored for the same query and candidate interface. The TaskMatch training pipeline converts existing tasks and candidates into graded weak supervision and distills their relevance patterns into a content-based TaskMatch with at most 0.30M trainable parameters. It requires no online judge, additional LLM call, or planning module. Across skill and procedural-memory retrieval, TaskMatch improves intrinsic ranking and downstream execution, including ALFWorld gains from 67.10% to 81.43% on top of SkillDAG and from 81.34% to 90.30% under MemP Proceduralization. The architectural ablation also shows that the highest Ret@1 and MRR do not necessarily yield the highest agent success. TaskMatch improves the selection of candidates, while the downstream agent remains responsible for using the retrieved candidates effectively.

6 Limitations

TaskMatch has three main limitations. Its relevance supervision is LLM-annotated rather than causally derived, its reranking quality can be bounded by upstream recall in restricted candidate sets, and skill and memory retrieval currently require separately trained TaskMatch models. Future work could address these issues through execution-derived supervision, adaptive candidate expansion, and shared cross-repository training.

AI USE STATEMENT

We used generative AI tools solely to assist with language polishing and improving the clarity and organization of the manuscript. All scientific content, analyses, experiments, and conclusions were produced and verified by the authors.

REFERENCES

Tong Bai, Zhenglin Wan, Pengfei Zhou, Xingrui Yu, Yang You, and Ivor W Tsang. Skilldag: Self-evolving typed skill graphs for llm skill selection at scale. *arXiv preprint arXiv:2606.03056*, 2026.

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.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. Automanual: Constructing instruction manuals by llm agents via interactive environmental learning. *Advances in Neural Information Processing Systems*, 37:589–631, 2024a.

Yanfei Chen, Jinsung Yoon, Devendra Singh Sachan, Qingze Wang, Vincent Cohen-Addad, Mohammadhossein Bateni, Chen-Yu Lee, and Tomas Pfister. Re-invoke: tool invocation rewriting for zero-shot tool retrieval. *arXiv preprint arXiv:2408.01875*, 2024b.

Yanping Chen, Weijie Shi, Wen Yang, and Jiajie Xu. Task decomposition-guided reranking for adaptive agent skill retrieval. *arXiv preprint arXiv:2607.06283*, 2026.

Yu Du, Fangyun Wei, and Hongyang Zhang. Anytool: Self-reflective, hierarchical agents for large-scale api calls. *arXiv preprint arXiv:2402.04253*, 2024.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502, 2026.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. *Advances in Neural Information Processing Systems*, 37: 119919–119948, 2024.

Luyu Gao and Jamie Callan. Unsupervised corpus aware language model pre-training for dense passage retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2843–2853, 2022.

Paimon Goulart, Liang Wu, Kelly Wan, Evangelos E Papalexakis, and Liangjie Hong. Field aware agent skill retrieval. *arXiv preprint arXiv:2608.02880*, 2026.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pp. 55–64, 2016.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *Advances in neural information processing systems*, 36:45870–45894, 2023.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pp. 173–182, 2017.

Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pp. 39–48, 2020.

Qinfeng Li, Dalin He, Yuntai Bao, Ying Yang, Ruoxi Chen, Xinyan Yu, Lizhou Liang, Ge Su, Wenqi Zhang, and Xuhong Zhang. Skillaligner: Treating retrieved skills as adaptable drafts at execution time. *arXiv preprint arXiv:2608.06880*, 2026a.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, et al. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026b.

Dawei Liu, Zongxia Li, Hongyang Du, Xiyang Wu, Shihang Gui, Yongbei Kuang, and Lichao Sun. Graph-of-skills: Dependency-aware structural retrieval for massive agent skills. *arXiv preprint arXiv:2604.05333*, 2026.

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. Cedr: Contextualized embeddings for document ranking. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, pp. 1101–1104, 2019.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. Text matching as image recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Gabriel Sarch, Lawrence Jang, Michael J Tarr, William W Cohen, Kenneth Marino, and Katerina Fragkiadaki. Vlm agents generate their own memories: Distilling experience into embodied programs of thought. *Advances in Neural Information Processing Systems*, 37:75942–75985, 2024.

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.

Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Shuaiqiang Wang, Dawei Yin, and Zhaochun Ren. Retrieval models aren't tool-savvy: Benchmarking tool retrieval for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24497–24524, 2025.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Xiaoyu Tan, Bin Li, Xihe Qiu, Chao Qu, Wei Chu, Yinghui Xu, and Yuan Qi. Meta-agent-workflow: Streamlining tool usage in llms through workflow construction, retrieval, and refinement. In *Companion Proceedings of the ACM on Web Conference 2025*, pp. 458–467, 2025.

Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, Liming Zhu, and Wenjie Zhang. Skillzip: Contract-preserving graph compression for scalable agent skill libraries. *arXiv preprint arXiv:2608.05604*, 2026.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, et al. Agent kb: Leveraging cross-domain experience for agentic problem solving. *arXiv preprint arXiv:2507.06229*, 2025.

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. *arXiv preprint arXiv:2305.16291*, 2023.

Zifei Wang, Wei Wen, Qiang Ji, Keyu Chen, Ruizhi Qiao, and Xing Sun. Skill is not document: Query-conditioned compatibility for llm agent skill routing. *arXiv preprint arXiv:2606.03565*, 2026a.

Zifei Wang, Wei Wen, Qiang Ji, and Ruizhi Qiao. Skills know their neighbors: Cluster-contrastive capability pages for skill retrieval. *arXiv preprint arXiv:2608.04482*, 2026b.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arXiv:2402.01622*, 2024.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval*, pp. 55–64, 2017.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 17577–17604, 2025.

Yue Yao, Shengyuan Wang, Xin Chen, Minke Zhang, Jia He, Bingjun Luo, and Tom Gedeon. Skilltrace: Traversing a query-skill graph for composable llm agents. *arXiv preprint arXiv:2608.02356*, 2026.

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi Fung, Hao Peng, and Heng Ji. Craft: Customizing llms by creating and retrieving from specialized toolsets. In *International Conference on Learning Representations*, volume 2024, pp. 40097–40125, 2024.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *International Conference on Learning Representations*, volume 2024, pp. 19036–19066, 2024a.

YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, and Gang Yu. Skillrouter: Skill routing for llm agents at scale. *arXiv preprint arXiv:2603.22455*, 2026.

Yuanhang Zheng, Peng Li, Wei Liu, Yang Liu, Jian Luan, and Bin Wang. Toolrerank: Adaptive and hierarchy-aware reranking for tool retrieval. In *Proceedings of the 2024 joint international conference on computational linguistics, language resources and evaluation (LREC-COLING 2024)*, pp. 16263–16273, 2024b.

A Implementation Details

A.1 TaskMatch Architecture

Both retrieval settings use the same NeuMF interaction architecture, with GMF and MLP projection dimensions of $d_G = 16$ and $d_M = 32$, respectively. Skill retrieval uses frozen 3072-dimensional `text-embedding-3-large` representations, while procedural-memory retrieval uses frozen 1536-dimensional `text-embedding-3-small` representations. For an input dimension d, TaskMatch contains

$$P(d) = 96d + 2641 \tag{15}$$

trainable parameters. This gives 297,553 parameters for skill retrieval and 150,097 parameters for procedural-memory retrieval.

A.2 Optimization Details

Grades $g \in \{0, 1, 2\}$ are mapped to pointwise targets $y = g/2$. The pointwise term uses binary cross-entropy with logits. For each task context, a candidate with a strictly higher relevance grade is paired with a lower-grade candidate, and the pairwise objective is

$$\mathcal{L}_{\text{pair}} = \text{softplus}\left(-\left[s_\theta(q, c^+) - s_\theta(q, c^-)\right]\right). \tag{16}$$

The total objective uses a pairwise coefficient of $\lambda = 0.2$. GMF and MLP are pretrained independently for at most 20 epochs with Adam at a learning rate of 10^{-3}. Their parameters initialize the full TaskMatch model, which is fine-tuned for at most 20 epochs using SGD at a learning rate of 10^{-2}. We use pointwise and pairwise batch sizes of 512, early-stopping patience of 5, no weight decay, and random seed 2026. Checkpoint selection is performed only on the corresponding internal development split.

B Graded Relevance Supervision Details

B.1 Candidate Pool Construction

Task–skill supervision. For composite ALFWorld tasks, operation-specific task views are derived from the training task schema (locate, transform, place, and verify when applicable). Candidate skills are collected from task-view cosine retrieval, full-task cosine retrieval, and hard candidates surfaced by the previous TaskMatch model. Route indicators are retained for auditing, whereas the network input contains only the task-view and skill-document embeddings. The implementation refers to these views internally as "phases." Throughout the paper, we use *task view* or *retrieval-query view* to distinguish this offline supervision representation from an inference-time planning component. The views mirror the short operation-specific queries naturally issued by the host agent during skill search; they are not generated by an additional planner. The 37-skill bank is small enough that intrinsic evaluation can rank the complete bank.

Task–memory supervision. For ALFWorld memories, query views are constructed from the training tasks and the source queries already associated with the 300 Script memories. Candidate routes include query cosine, memory-content cosine, structured exact matches, structured complementary matches, and structured hard negatives. A query is retained only when the fixed memory bank contains at least one calibrated Grade 1 or Grade 2 candidate. During training, every Grade 1/2 pair is retained and four Grade 0 candidates per positive are selected deterministically by hardest cosine-route rank. The internal development split retains its complete candidate set.

More precisely, the construction starts from 945 public ALFWorld training queries and deterministically renders each query from four relation-focused views: the complete task, operation, object, and destination. This gives 3,780 canonical views. We first split these views by parent-task family, obtaining 3,440 internal-training and 340 model-development views. Only the internal-training views are then expanded with three slot-preserving surface forms each, adding 10,320 views and producing 14,100 train-derived views in total. The positive-evidence gate excludes 152 training and four development views for which the fixed bank contains no defensible positive, leaving 13,608 training and 336 model-development views. All variants of a parent query remain in the same split.

These are alternative supervision contexts derived from training tasks, not independent benchmark instances or evaluation-derived examples.

B.2 LLM JUDGE PROMPTS

The skill judge is used only during offline supervision construction. It receives the full task workflow, an operation-specific task view and its objective, together with the candidate skill identifiers, descriptions, and instruction previews. The exact prompt used in our experiments is reproduced below. In the prompt, "phase" refers only to the operation-specific task view used for constructing supervision and does not denote an inference-time task decomposition or planning module.

```
You are an offline annotator for phase-level skill retrieval in an
    embodied agent.
Judge each candidate skill for the CURRENT PHASE, while using the full
    workflow only as context.

Grades:
- 2: directly provides a distinct capability normally required to
    complete the current phase.
- 1: useful complementary support, prerequisite, verification, or
    recovery for the current phase, but not directly required.
- 0: irrelevant to the current phase, belongs mainly to another phase, is
     redundant without adding a distinct capability, or may distract the
    agent.
- uncertain: the supplied skill text is insufficient or genuinely
    ambiguous.

Be conservative with grade 2. Similar wording alone is not enough. Do not
     mark several interchangeable generic skills as required unless each
    supplies a distinct necessary capability.

Confidence calibration:
- 0.90-1.00 only for an explicit, unambiguous capability match or
    mismatch.
- 0.70-0.89 for a strong inference.
- below 0.70 when context or scope is ambiguous; use uncertain if needed.

Return exactly one JSON object with every skill exactly once:
{"judgments":[{"skill_id":"exact-id","grade":2,"confidence":0.91,"reason
    ":"short reason"}]}
```

The memory judge is likewise used only during offline supervision construction. To match the information available to the query-cosine baseline, it receives only train-derived task queries and memory source queries, without access to trajectory bodies or evaluation outcomes. The exact prompt used in our experiments is reproduced below.

```
You are an offline annotator for query-based procedural-memory retrieval
    in an embodied ALFWorld agent.

Grades:
- 2: the memory source query directly matches the current retrieval focus
    , including every stated constraint.
- 1: it supplies a distinct complementary relation needed by the full
    task, such as the correct operation/cardinality, object-handling
    pattern, or destination/tool pattern, but is insufficient for the
    current focus by itself.
- 0: it conflicts with the required operation or cardinality, only shares
     incidental words, duplicates another stronger candidate's role, or
    does not help the full task.
- uncertain: the two queries genuinely do not provide enough evidence.

Rules:
- clean, heat, cool, examine, one-object placement, and two-object
    placement are different operations.
```

```
- operation agreement dominates an operation-focused query.
- object identity and cardinality dominate an object-focused query.
- destination/tool identity and cardinality dominate a destination-
    focused query.
- for the unmodified full task, grade 2 requires operation, object,
    cardinality, and destination/tool to match.
- keep at most one interchangeable grade-1 representative per support
    role.
- do not invent workflow steps absent from the two queries.

Return JSON with grade, confidence, operation, object, cardinality,
destination, procedural_validity, distinct_contribution, and reason.
```

B.3 Label Definitions and Worked Examples

Grade 2 denotes a directly required capability or a memory query that covers the current retrieval focus. Grade 1 denotes a distinct prerequisite, complement, verification, recovery step, or transferable subprocedure. Grade 0 denotes irrelevance, an execution conflict, a redundant alternative, or a potential distraction. The `uncertain` label is excluded until it can be resolved without evaluation data.

For the operation-specific task view "clean a fork," `clean-object` is Grade 2 because it directly supplies the state-changing capability. A valid object-handling or verification skill can be Grade 1 when it contributes a separate prerequisite or postcondition check. A cooling skill is Grade 0 because it conflicts with the required operation despite both skills modifying object state. For a memory query "clean a spoon and place it on the countertop," an exact operation/object/cardinality/destination match is Grade 2; a clean-and-place procedure for another utensil may be Grade 1 when it supplies a transferable operation-specific subprocedure; a heat or cool procedure is Grade 0.

B.4 Hard-Negative Construction

Hard negatives are deliberately plausible under semantic similarity but incompatible with the task requirement. In skill retrieval, a verification view such as "verify the tomato is heated and on the dining table" can rank heater and temperature-management skills highly by cosine, although a verifier is the required candidate. In memory retrieval, a source query may share the same object and destination while specifying the wrong operation or cardinality. These candidates are more informative than uniformly sampled negatives because they teach TaskMatch which apparently related candidate should not displace an execution-compatible one.

B.5 Data Statistics and Leakage Control

Table 6 reports the frozen supervision used in each of the four retrieval settings. Pair counts are setting-specific rather than a fixed budget shared across datasets: they depend on the number of train-derived query views, the candidate-pool size and granularity, the positive-evidence gate, and the number of hard negatives retained per query.

The ALFWorld skill corpus contains 700 train-derived tasks split by task identity into 560/70/70 internal train/dev/test subsets, yielding 2,791 task views and 49,773 total pairs. SkillsBench uses a larger 1,000-skill catalog but caps the judged candidate set for each document-derived query; its frozen skill-only corpus contains 6,147 queries and 69,379 total pairs. Its official benchmark tasks

Table 6: Frozen task–candidate supervision statistics for the four target settings. "Pairs/query" normalizes training pairs by training query views; "Preferences" counts strict within-query graded comparisons used by the pairwise objective.

Candidate Type	Benchmark	Train tasks/queries	Candidates	Train pairs	Pairs/query	Dev pairs	Train G0	Train G1	Train G2	Preferences
Skill	ALFWorld	560 tasks / 2,230 views	37	39,769	17.83	4,969	22,585	9,193	7,991	180,054
Skill	SkillsBench-1000	4,963 queries	1,000	54,472	10.98	14,907	44,341	4,505	5,626	106,087
Memory	ALFWorld	13,608 queries	300	198,696	14.60	8,064	151,156	36,164	11,376	584,136
Memory	TravelPlanner	1,270 retrieval views	519	30,840	24.28	11,017	10,591	8,762	11,487	246,673

are used only for evaluation. After the construction detailed above, the ALFWorld memory corpus contains 14,100 train-derived query views; 13,944 satisfy the positive-evidence gate, of which 13,608 are used for training and 336 for model selection. TravelPlanner instead represents procedures as 519 operation-level candidates and uses 1,980 train-derived retrieval views, split by task family into 1,270 training and 710 development queries, yielding 41,857 total pairs.

These differences are intentional and do not indicate that one setting receives more evaluation supervision. We construct only the informative task–candidate relations supported by the training data rather than materializing every query–candidate Cartesian product. The amount of supervision consequently tracks the available train-derived views and the density of defensible hard comparisons. In particular, Memory–ALFWorld materializes 198,696 training pairs but only 14.60 pairs per training query; its larger absolute corpus is primarily explained by its larger set of train-derived query views rather than unusually dense supervision. All method comparisons within a benchmark use the same frozen queries, candidate pool, labels, and splits; pair counts are never tuned using the corresponding evaluation outcomes. The ALFWorld skill-template audit contains 33 generic templates and 1,221 MiniMax-M2.7 judgments (812/325/84 for Grades 0/1/2); all responses parsed successfully.

No official evaluation task, reward, expert plan, or evaluation trajectory is used to construct labels, model inputs, weights, or checkpoints. Task identity is separated across internal splits. For memory data, canonical cross-split overlap is zero. Judge reasons, confidence, calibration actions, and all privileged execution fields are retained only in audit artifacts and excluded from model inputs.

Composite training tasks are additionally rendered as operation-specific task views. These views approximate the granularity of the local retrieval queries issued by the host system during execution and are used only to construct offline supervision; they do not add an inference-time task decomposition or planning module. Finally, the labels are relevance supervision rather than causal effect estimates: the objective is to rank candidates for the host-issued query, not to estimate the marginal contribution of an individual candidate to task success. We therefore evaluate TaskMatch using both intrinsic ranking metrics and end-to-end agent outcomes.

C Full Experimental Settings

C.1 Skill Retrieval

ALFWorld uses the 37-skill library and graph released with SkillDAG. Offline analysis ranks the complete 37-skill pool and reports Top-5 retrieval. The end-to-end experiment follows the 140-task `valid_seen` setting with a 30-environment-step limit and a single run, matching the budget used in the reported SkillDAG comparison. The graph remains editable. During execution, SkillDAG can invoke skill retrieval repeatedly as the agent progresses through a composite task. These events naturally use local operation-specific queries rather than always repeating the initial task description. TaskMatch does not create or decompose these queries. At every retrieval event, cosine and TaskMatch scoring receive the same host-issued query, candidate interface, and frozen representations. Any deterministic query transformation in the host pipeline is shared by all TaskMatch variants. SkillDAG's graph interface and execution loop remain unchanged. Values attributed to prior methods in the main tables are quoted from the corresponding papers; rows containing TaskMatch use our frozen checkpoints and local runs.

SkillsBench uses the 1,000-skill collection and final Top-5 retrieval. The skill documents and cold-start graph supply training evidence; the official 87 evaluation tasks and their outcomes are excluded from training and model selection. The two agent backbones are MiniMax-M2.7 and GPT-5.2-Codex, as specified in the main tables.

C.2 Procedural-Memory Retrieval

The ALFWorld memory bank contains 300 Script memories. Full-pool retrieval ranks all 300 memories; the cascade first keeps 50 cosine memories and returns the Top-10 after TaskMatch scoring. End-to-end comparisons reuse the same GPT-4o agent, prompts, action budget, memory format, and MemP execution code, changing only memory ordering. Script, Trajectory, and Proceduralization results are reported separately because their retrieved content and downstream use differ.

TravelPlanner uses 519 candidate operations derived from the same memory bank available to both compared retrievers. A fixed parser constructs five operation-specific retrieval views, and the selected operations are assembled back into memory context before the unchanged GPT-4o planner is called. Query cosine and TaskMatch use the same five views; the parser is part of their shared retrieval protocol. Full-pool rows score all 519 candidates; cascade rows use $519 \rightarrow 50 \rightarrow 10$. The official test labels are hidden; we therefore follow the released validation protocol for model development.

C.3 Evaluation Metrics

For intrinsic skill retrieval, we follow the retrieval evaluation protocol of SkillDAG and report Ret@K, Ret@1, and mean reciprocal rank (MRR). Ret@K measures whether the Top-K retrieved results cover the ground-truth skill required by the retrieval query, while Ret@1 measures whether a ground-truth skill is ranked first. MRR further accounts for ranking position by assigning a larger score when the first ground-truth skill appears earlier in the retrieved list. Higher values indicate better retrieval quality.

For intrinsic procedural-memory retrieval, we additionally report NDCG@10, Precision@10, and Harm@10. NDCG@10 measures the overall quality of the Top-10 ranking by rewarding relevant memories that appear closer to the top of the list. Precision@10 measures the proportion of relevant memories among the Top-10 retrieved results, while Harm@10 measures the proportion of retrieved memories that are incompatible with or unhelpful for the current retrieval query. Higher NDCG@10 and Precision@10 are better, whereas lower Harm@10 is better.

For end-to-end evaluation, we follow the metrics of the corresponding host benchmarks. SkillsBench reports task reward. On ALFWorld, task completion is determined by the environment, and success rate measures the proportion of episodes completed successfully; average steps measures the number of environment interactions required for execution. On TravelPlanner, #CS and #HC denote the Common Sense and Hard Constraint scores, respectively, obtained by evaluating the generated travel plan against the benchmark requirements, while Steps measures execution efficiency. Higher reward, success rate, #CS, and #HC are better, whereas fewer execution steps are better.

C.4 Choice of Host Retrievers

Recent concurrent methods improve complementary stages of the skill-use pipeline, including skill representation, library organization, structural retrieval, and post-retrieval adaptation. Rather than reimplementing each method under a new evaluation stack, we focus on controlled integrations with two representative host retrievers: structure-aware SkillDAG and similarity-based MemP. These settings allow us to isolate the effect of TaskMatch scoring while covering substantially different retrieval mechanisms.

C.5 TaskMatch Implementation and Evaluation Protocol

The TaskMatch models for skill and memory retrieval inherit the frozen representations used by their respective host retrieval systems. Skill retrieval uses 3072-dimensional `text-embedding-3-large` representations and yields a 297,553-parameter TaskMatch model (0.298M), while procedural-memory retrieval uses 1536-dimensional `text-embedding-3-small` representations and yields a 150,097-parameter TaskMatch model (0.150M). Both settings use the same interaction architecture, so the difference in model size comes solely from the embedding dimensionality inherited from the host retrieval system. Candidate representations are precomputed and cached in both settings. Training and model selection use only non-test supervision; test task–candidate labels and execution outcomes are reserved for final evaluation. Detailed optimization hyperparameters are provided in Appendix A.

C.6 Hardware and Efficiency Measurement

Routine TaskMatch training was performed on a dual-socket Intel Xeon Gold 6530 machine with 64 physical cores and 128 hardware threads. For the efficiency measurements reported in Table 7, we used a single NVIDIA GeForce RTX 4090 with 48 GiB of memory. Inference latency was mea-

sured by batch-scoring the complete 300-memory ALFWorld bank using one CPU thread, with 200 warm-up queries followed by 10,000 timed queries. Each method was evaluated in a fresh process, and we report the arithmetic mean latency. Embedding computation is excluded because both methods use the same precomputed and cached representations. Peak GPU memory was measured using the PyTorch allocator after resetting peak-memory statistics. We additionally reran the complete TaskMatch training pipeline from scratch on the same RTX 4090: GMF pretraining, MLP pretraining, and NeuMF fine-tuning together required 295.02 s, with a peak allocated training memory of 128.25 MiB.

Table 7: Additional cost of TaskMatch in the ALFWorld procedural-memory setting with a 300-memory pool. Latency excludes the shared text-encoding stage and measures candidate scoring after frozen representations are available. Training time covers TaskMatch optimization after offline supervision construction and excludes LLM annotation.

Method	**Added Params**	**#Scored**	**TaskMatch Train**	**Latency↓**	**Mem.↓**
Query cosine	0	300	–	0.057 ms	13.83 MiB
TaskMatch (Full)	0.15M	300	295.02 s	0.248 ms	17.21 MiB

Efficiency. Table 7 reports the additional training and inference cost of TaskMatch once offline supervision and frozen representations are available. Candidate representations are cached, and task–candidate interactions are batched across candidates. On the 300-memory ALFWorld pool, scoring and ranking the complete bank takes 0.248 ms per query, compared with 0.057 ms for cosine retrieval, corresponding to an absolute increase of 0.191 ms. The full training recipe (20 GMF epochs, 20 MLP epochs, and early-stopped NeuMF fine-tuning) completes in 295.02 s, excluding LLM annotation. Peak inference memory increases from 13.83 MiB to 17.21 MiB. Because TaskMatch reuses the frozen embeddings required by the host retriever, it introduces no additional online embedding or LLM calls.

D Additional Results and Analysis

D.1 Ablation Protocol

All offline architectural variants in Table 8 use the same arrays, splits, embeddings, candidate pool, seed, and evaluation code. Only the named component changes. In particular, TaskMatch (GMF), TaskMatch (MLP), and TaskMatch (Full) are pure full-pool variants in this table; no variant receives a different graph or cosine-fusion advantage. Binary-label training maps both supportive and required candidates to one positive class. The no-pairwise variant sets $\lambda = 0$, and the random-negative variant replaces deterministic hard negative selection while keeping the pair budget fixed.

The offline columns of Table 5 use the deployed online calibration for each branch and are therefore intentionally distinct from the pure full-pool architecture table. Success is measured only by a full agent run; it is never inferred from Ret@1 or MRR.

D.2 Full Architectural Ablation

Component Ablation. Table 8 isolates the contributions of the interaction architecture and training strategy. The TaskMatch (Linear), TaskMatch (GMF), and TaskMatch (MLP) variants compare increasingly expressive task–candidate interaction functions, while the remaining variants ablate graded labels, pairwise ranking, and hard-negative construction with the other components fixed. TaskMatch (Linear) performs substantially below the base retriever in both settings, indicating that a simple linear mapping is insufficient for modeling task–candidate relevance from the frozen representations. GMF and MLP substantially improve over TaskMatch (Linear), while combining both interaction branches yields the strongest overall retrieval performance. On skill retrieval, the full TaskMatch model improves Ret@1 by 1.80 points over TaskMatch (MLP) and by 8.19 points over TaskMatch (GMF). The supervision design also matters, particularly for memory retrieval: binarizing the graded labels reduces Ret@1 from 91.04 to 48.75, while removing the pairwise objective reduces it to 56.63. Random negatives remain competitive for memory retrieval but reduce skill

Table 8: Component ablation on ALFWorld. Skill retrieval uses SkillDAG candidates; memory retrieval uses Script memory with the matched full pool.

	Skill Retrieval		Memory Retrieval		
Variant	Ret@1↑	MRR↑	Ret@1↑	MRR↑	NDCG@10↑
Base retriever	59.88	71.08	73.84	82.98	71.81
TaskMatch (Linear)	30.14	46.16	5.73	12.42	6.39
TaskMatch (GMF)	63.47	79.64	65.59	74.68	61.01
TaskMatch (MLP)	69.86	83.03	85.30	90.07	79.10
TaskMatch (Full + Binary labels)	68.86	81.95	48.75	58.49	48.72
TaskMatch (Full w/o Pairwise loss)	62.67	79.49	56.63	66.68	51.08
TaskMatch (Full w/ Random negatives)	59.08	78.53	87.46	92.48	82.57
TaskMatch (Full)	**71.66**	**83.85**	**91.04**	**93.82**	**84.45**

Ret@1 from 71.66 to 59.08, suggesting that targeted hard-negative construction is particularly useful in the skill setting.

D.3 Retrieval–Execution Gap

The experiments show that retrieval is important but not sufficient. For ALFWorld skills, the large Ret@1 improvement coincides with a 14.33-point success gain. For Script memory, however, a 17.20-point Ret@1 gain yields only 0.74 points of Test SR. TravelPlanner exhibits the same pattern: much stronger Top-K coverage produces modest Script-level gains because planning quality, constraint grounding, and model stochasticity remain downstream bottlenecks. We therefore report intrinsic and end-to-end metrics separately.

D.4 RQ1: Detailed Intrinsic Retrieval Results

Skill Retrieval. As shown in Table 1, TaskMatch consistently improves SkillDAG's intrinsic retrieval quality. With MiniMax-M2.7, Ret@1 increases from 66.70 to 76.98 on SkillsBench and from 57.90 to 86.03 on ALFWorld, while MRR increases from 71.30 to 79.70 and from 71.10 to 92.50, respectively. Improvements remain with GPT-5.2-Codex, including an ALFWorld MRR increase from 69.60 to 86.45. Thus, TaskMatch improves candidate ordering even when candidates are produced by a structure-aware retriever.

Procedural-Memory Retrieval. Table 2 shows that TaskMatch also improves procedural-memory retrieval. Under the matched ALFWorld full-pool setting, Ret@1 increases from 73.84 to 91.04, MRR from 82.98 to 93.82, and NDCG@10 from 71.81 to 84.45. On TravelPlanner, the largest improvements appear in Top-K coverage: Ret@5 rises from 16.22 to 94.67 and Ret@10 from 18.78 to 98.67, while Harm@10 decreases from 88.77 to 66.59.

Cascade Retrieval. We additionally evaluate a two-stage cascade in which cosine similarity first reduces the memory pool to 50 candidates and TaskMatch reranks only this subset. This setting tests whether TaskMatch can be used as a lightweight second-stage reranker when scoring the full repository is unnecessary or undesirable. On ALFWorld, cosine retrieval already provides strong candidate recall, so reranking the reduced pool further improves Ret@1 from 73.84 to 96.06. On TravelPlanner, however, the cosine first stage has much lower recall and removes relevant memories before TaskMatch is applied, reducing Ret@5 and Ret@10 relative to full-pool scoring. We therefore use the cascade as an efficiency-oriented deployment variant rather than the primary evaluation of TaskMatch.

D.5 RQ2: Detailed End-to-End Results

Skill Selection. Table 3 shows that the retrieval improvements translate into stronger agent execution. With MiniMax-M2.7, adding TaskMatch to SkillDAG increases SkillsBench reward from 27.30 to 29.21 and ALFWorld success from 67.10% to **81.43%**. The improvement persists with

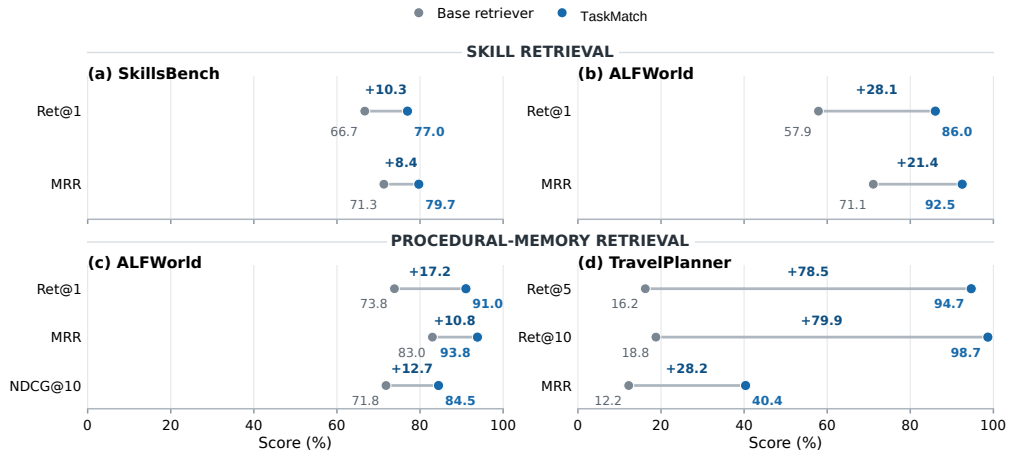

Figure 3: Intrinsic retrieval improvements across skill and procedural-memory settings. Each panel compares the corresponding host retriever with TaskMatch, while annotated differences show the absolute improvement in percentage points. The gains recur across all four evaluation settings, spanning both structure-aware skill retrieval and similarity-based procedural-memory retrieval. Detailed results are reported in Tables 1 and 2.

GPT-5.2-Codex, where ALFWorld success increases from 93.60% to **96.43%**. These gains show that TaskMatch can further complement a strong structure-aware retriever across different agent backbones.

Procedural Memory. As reported in Table 4, TaskMatch improves downstream performance without changing the memory representation. On ALFWorld, Test SR increases from 80.60% to 84.33% for Trajectory memory and from 81.34% to **90.30%** for Proceduralization, while the latter reduces average steps from 13.30 to **12.22**. On TravelPlanner, Proceduralization improves #HC from 64.06 to **76.19** and reduces average steps from 15.73 to **14.66**. Script memory exhibits only modest downstream gains despite substantial intrinsic retrieval improvements, indicating that retrieval quality contributes to, but does not fully determine, final agent performance.

E Case Studies and Failure Analysis

E.1 Detailed Skill Retrieval Case

Figure 1 provides a compact visualization of this case; here we report the complete retrieval rankings and execution details. We examine a held-out ALFWorld task that requires finding two tissue boxes and placing them in the toilet. Both runs issue the same batched retrieval call with the queries "find tissuebox" and "put tissuebox in toilet," requesting five skills for each query over the same skill library. For the shared initial query "find tissuebox," cosine similarity ranks `receptacle-finder`, `tool-locator`, `receptacle-searcher`, `storage-explorer`, and `search-pattern-executor` in its Top-5. This ranking is consistent with the surface association between "box" and receptacle-oriented or generic-search capabilities, although the tissue box is the object that must be located, retrieved, and transported. In contrast, TaskMatch ranks `object-transporter`, `object-locator`, `object-retriever`, `inventory-management`, and `tool-locator`, providing a more complementary set of capabilities for locating, retrieving, transporting, and tracking multiple target objects.

In the corresponding execution traces, the cosine-based run follows a broad sequential search through eight drawers before reaching the countertop and exhausts the 30-step budget. The run using the TaskMatch ranking prioritizes the countertop, immediately locates multiple tissue boxes, and completes the two-object transport task in 8 environment steps. Because the two trajectories are independent stochastic agent runs, this example is intended only to illustrate how different retrieval

rankings can align with different execution behaviors; our quantitative conclusions are based on the complete benchmark.

E.2 Cardinality-Sensitive Procedural-Memory Reranking Case

We next examine a real retrieval case from the frozen, train-derived model-development split. The query requires placing *two* spray bottles *in* a toilet, jointly specifying the target object, cardinality, and placement relation. Query cosine ranks a single-object procedure, `put a spraybottle in toilet`, first and places the exact two-object procedure only fourth. Its remaining Top-3 results also match the target object lexically but fail to preserve the required cardinality and placement relation.

TaskMatch instead promotes `find two spraybottle and put them in toilet` from cosine rank four to rank one. This memory is the only displayed candidate that satisfies all three requirements: the target object, the two-object workflow, and the intended placement relation. The case illustrates the distinction between surface similarity and fine-grained task–memory relevance. It is an intrinsic retrieval example and does not use downstream agent outcomes or official evaluation trajectories.

E.3 Failure Modes Across Retrieval and Execution

End-to-end agent performance depends on more than candidate ranking. We distinguish three failure modes according to where they arise in the retrieval–execution pipeline: candidate-recall failure, ranking failure, and downstream execution failure. This distinction is important because TaskMatch directly targets only the middle stage.

Candidate-recall and ranking failures. In restricted candidate-generation settings, a relevant memory may be filtered out before reranking, leaving a post-retrieval scorer no opportunity to recover it. This limitation does not apply to the full-pool setting, where the complete memory bank is directly available for scoring. In contrast, when a relevant memory is present but underranked, TaskMatch can intervene directly. The cardinality-sensitive example in Section E.2 illustrates this case: cosine similarity favors lexically close but constraint-incomplete memories, whereas TaskMatch promotes the memory that better satisfies the task's procedural requirements.

Downstream execution failures. Improved retrieval quality does not uniquely determine the final agent outcome. In our paired ALFWorld analysis, we observe cases in which two runs receive the same retrieved memory contents in the same order but nevertheless follow different execution paths and reach different outcomes. Such divergence can arise from downstream behaviors including inefficient object search, invalid actions, state-transition mistakes, repeated actions, or exhaustion of the step budget. These observations indicate that retrieval quality and execution quality are related but distinct: TaskMatch reranking can improve the information exposed to the agent, while the downstream policy remains responsible for converting that information into successful actions.