SelfEvolveAgent: Continual Learning in AI Agents via Reasoning Memory

Hiva Mohammadzadeh (Zaad)
Stanford University
hiva@stanford.edu

Abstract

Large language model (LLM) agents demonstrate impressive reasoning capabilities on complex, multi-step tasks, yet they lack a principled mechanism for continual learning. Once an episode ends, agents typically discard the trajectory, causing them to repeat past mistakes, fail to generalize successful strategies, and miss opportunities for systematic self-improvement. ReasoningBank introduces a promising memory-based approach that distills reusable reasoning strategies from both successful and failed attempts. However, several key gaps remain unresolved: (1) the absence of a fully open-source and reproducible implementation, (2) degradation in memory quality as experience accumulates, and (3) limited understanding of how memory interacts with task structure, difficulty, and retrieval dynamics.

In this work, we implement and extend the ReasoningBank framework in a controlled and reproducible experimental setting. We evaluate memory-augmented agents across WebArena (Multi, Reddit, Admin, GitLab, Shopping), Mind2Web, and SWE-Bench-Verified. Beyond aggregate success metrics, we introduce fine-grained diagnostics—including retrieval precision, memory utility rate, and performance as a function of memory size—to expose when retrieval succeeds but fails to influence downstream behavior. We further analyze task distributions and difficulty strata to identify regimes where memory provides the largest gains.

Our results demonstrate consistent improvements across models and domains, with the largest benefits emerging on difficult, long-horizon tasks that require repeated decision-making and strategic reuse. Ablation studies reveal that memory *quality*, rather than raw quantity, is the primary driver of performance gains. Together, these findings clarify both the promise and limitations of current continual-learning approaches for LLM agents and motivate concrete design principles for future memory-augmented systems.

Code: HivaMohammadzadeh1/SelfEvolveAgent-Reasoningbank-code

1 Introduction

Large language model agents have achieved strong performance on long-horizon reasoning and interactive decision-making tasks, particularly when coupled with frameworks such as ReAct. However, these agents do not learn cumulatively from experience. Once a task concludes, trajectories and intermediate insights are discarded, leading to repeated errors and weak transfer of successful strategies across tasks. Enabling continual learning via memory is therefore a central challenge in building agents that improve over time and adapt to evolving environments.

ReasoningBank proposes storing distilled reasoning strategies derived from both successes and failures, enabling agents to retrieve and reuse prior experience. While promising, several fundamental questions remain unanswered. How does reasoning memory scale as the number of stored experiences grows? Does memory provide uniform benefits across task types and difficulty regimes? Which tasks

Preprint. Under review.

benefit most from memory augmentation, and which remain largely unaffected? Finally, what is the relationship between memory quality, memory quantity, and downstream task performance?

This work addresses these questions through a reproducible, open-source implementation of ReasoningBank and a systematic evaluation across multiple realistic benchmarks. Rather than focusing solely on aggregate success rates, we analyze performance across task distributions, difficulty categories, and retrieval quality strata. By comparing against a no-memory ReAct baseline, we quantify the contribution of memory using both standard task-level metrics and newly introduced fine-grained measures that capture retrieval accuracy, memory utilization, and memory-to-action influence.

Contributions. Our primary contributions are:

- A complete, open-source implementation of ReasoningBank integrated with BrowserGym across WebArena, Mind2Web, and SWE-Bench-Verified.
- An extended evaluation protocol with new diagnostics that expose a substantial retrieval–utilization gap and isolate when memory meaningfully affects agent behavior.
- A detailed analysis of task structure and difficulty demonstrating that memory *quality*, not quantity, is the dominant factor driving performance gains.

2 Methods

Our system builds on three components: a ReAct baseline, the ReasoningBank memory mechanism, and an extended evaluation framework designed to probe the role of memory in fine-grained detail.

2.1 ReAct Baseline

We implement a standard Think–Act–Observe loop using BrowserGym environments and accessibility-tree observations. This serves as our "No Memory" control. The agent performs reasoning and action without any experience retention, providing a clean baseline for measuring the impact of memory enhancement.

2.2 ReasoningBank Memory System

Our ReasoningBank implementation consists of:

- **Memory Representation:** JSON objects containing a title, a natural-language description, and a multi-step reasoning pattern (content array). These distilled strategies capture both successful solution paths and characteristic failure modes.
- **Embedding + Retrieval:** We use `gemini-embedding-001` and *m2-bert-80M-32k-retrieval* to encode both queries and memories, and store them in a FAISS vector database for efficient, similarity-based retrieval at scale. At each step, the top-k memories are retrieved and injected into the agent prompt as contextual guidance.
- **Self-Evolution Mechanism:** After each task, the agent self-evaluates the outcome and writes new memories for both success patterns and recurring failures. This enables continuous learning from experience over time.

2.3 Evaluation Setup

We evaluate our agents across realistic web and software engineering environments, with a focus on measuring how memory affects success, efficiency, and behavior:

- **Benchmarks:** WebArena, Mind2Web, and SWE-Bench-Verified. Within WebArena, we target Multi, Reddit, Admin, GitLab, Shopping, and Map subsets. **Due to instability in the WebArena AWS EC2 environments, we were unable to complete Admin, GitLab, and Shopping runs. Additionally, we were unable to report Mind2Web results because its tasks require online execution and dynamic website access, which was incompatible with our evaluation setup at the time of experiments.**
- **Models:** Our primary experiments use Gemini-2.5-Flash and Gemini-2.5-Pro as the underlying LLMs. We also evaluate on Qwen2.5-72B-Instruct as an open-source alternative.

- **Agent Variants:** The *No Memory* baseline is a standard ReAct agent that never reuses past experience. The *ReasoningBank* variant augments this agent with the memory representation, retrieval, and self-evolution mechanism described above.
- **Metrics:** For WebArena, we report success rate and average number of interaction steps to success. In addition to these standard metrics, we track retrieval precision, memory utility rate, and performance as a function of memory size. For Mind2Web and SWE-Bench-Verified, we follow the task-specific success criteria defined in the original benchmarks.
- **Infrastructure:** WebArena environments are hosted on AWS, and we integrate BrowserGym for browser automation and environment control. Our compute stack consists of API access to Google Gemini and Together AI, along with local FAISS-based indexing for memory retrieval.

2.4 Agent Variants

We evaluate four agent configurations:

- **No Memory (ReAct):** Standard Think–Act–Observe loop with no experience retention.
- **ReasoningBank (SelfEvolveAgent):** ReAct augmented with a ReasoningBank memory store, top-k retrieval ($k = 3$), and post-episode self-evaluation that writes both success strategies and failure modes.
- **MemOpt:** An optimized memory design that improves memory storage and retrieval (e.g., better memory selection/filtering and more robust retrieval integration), aimed at maximizing downstream utility rather than raw retrieval accuracy. This is done by using a single memory bank shared across all benchmarks and tasks.
- **Expected variants (NoMem Expected, RBank Expected):** Variants taken from the ReasoningBank reported evaluations.

2.5 New Metrics Introduced (Based on Feedback)

To strengthen evaluation, we define:

- **Retrieval Precision @1, @3:** Fraction of retrieved memories judged relevant.
- **Memory Utility Rate:** Fraction of retrieved memories actually used in the agent's final reasoning (measured via self-rationale and attention to memory content).
- **Memory Quality Index:** Average human- or model-judged helpfulness of stored strategies.
- **Memory Quantity Ablation:** Performance as a function of memory size ($0, 10, 50, 100, 200$ strategies).
- **Difficulty-Stratified Accuracy:** Tasks are grouped into *easy*, *medium*, and *hard* based on number of required steps and branching factor.

2.6 Task Distribution Analysis

Task distributions differ substantially across benchmarks, shaping when and how reasoning memory is most useful.

WebArena (812 tasks). WebArena is highly skewed across subsets (Figure 1). Nearly half of tasks come from **Shopping (23%)** and **Admin (22%)**, both involving long-horizon, multi-form interactions. **GitLab (22%)** emphasizes procedural repository operations, while **Map (13%)** focuses on spatial lookup and navigation. **Reddit (13%)** tasks are shorter and text-centric, and **Multi (6%)**—though small—contains the most compositionally challenging tasks. As a result, memory is most beneficial in long-horizon subsets (Admin, Shopping, GitLab) and less impactful for single-page extraction tasks (Reddit).

SWE-Bench (500 tasks). SWE-Bench is dominated by a small number of repositories, most notably `django` (231 tasks), followed by `sympy` (75), `sphinx` (44), `matplotlib` (34), and `scikit-learn` (32). Low-sample repositories (e.g., `seaborn`, `flask`) highlight the importance of memory for cross-task generalization beyond repository-specific patterns.

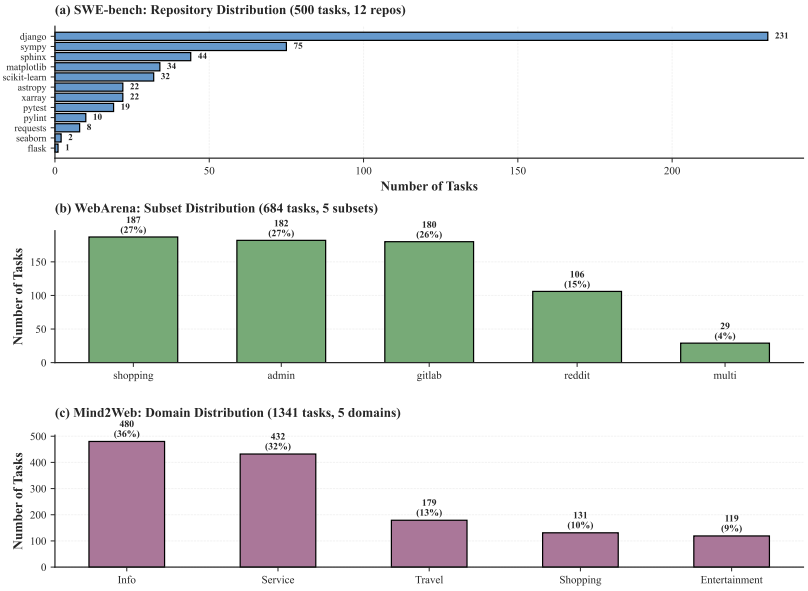

Figure 1: Dataset distribution across benchmarks. (a) SWE-bench repository distribution (500 tasks across 12 repos), (b) WebArena subset distribution (812 tasks across 6 environments), and (c) Mind2Web domain distribution (252 tasks across 69 websites). These distributions highlight substantial variation in domain complexity, task density, and environment structure.

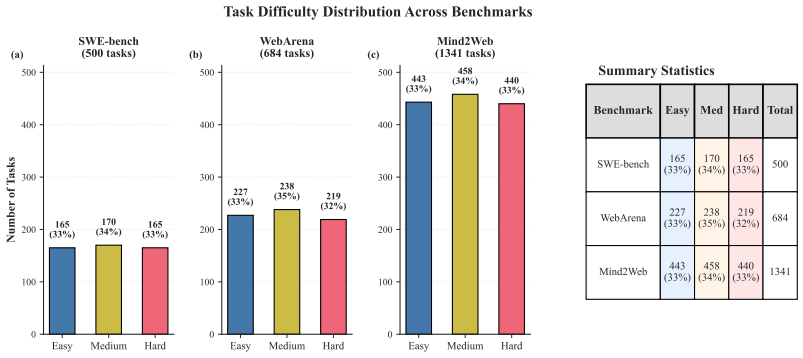

Figure 2: Task difficulty distribution across benchmarks. (a) SWE-bench (500 tasks), (b) WebArena (812 tasks), and (c) Mind2Web (252 tasks), with a summary table showing the counts and proportions of Easy, Medium, and Hard tasks for each benchmark.

Mind2Web (252 tasks). Mind2Web spans **Travel (47%)**, **Shopping (27%)**, and **Entertainment (26%)**. Tasks typically require structured, multi-page interactions, with Travel dominating through repeated entity selection and navigation patterns. This structure favors reusable navigation heuristics that can be captured and reused via memory.

3 Results

3.1 WebArena Performance

WebArena evaluates agents on realistic, long-horizon web interaction tasks that require multi-step planning, state tracking, and recovery from partial failures. These tasks are particularly well-suited for evaluating memory mechanisms, as successful trajectories often reuse recurring interaction patterns

Table 1: Performance on WebArena subsets for Gemini-2.5-Flash. SR = success rate (%), Step = average steps (lower is better).

	No Mem		RBank		MemOpt		NoMem Exp.		RBank Exp.	
Subset	SR	Step	SR	Step	SR	Step	SR	Step	SR	Step
Multi (29)	10.34	17.0	13.80	10.0	**17.24**	14.0	10.30	8.8	13.80	8.8
Reddit (106)	68.87	9.62	70.75	10.32	**77.36**	8.1	55.70	6.7	67.00	5.6

Table 2: Performance on WebArena subsets for Gemini-2.5-Pro. SR = success rate (%), Step = average steps (lower is better).

	No Mem		RBank		MemOpt		NoMem Exp.		RBank Exp.	
Subset	SR	Step	SR	Step	SR	Step	SR	Step	SR	Step
Multi (29)	13.79	14.50	17.2	15.83	**20.7**	12.75	6.9	8.8	13.8	8.2
Reddit (106)	64.15	10.53	73.6	8.2	78.3	7.81	71.7	6.0	**80.2**	5.1

(e.g., login flows, form submission sequences, navigation heuristics). We report WebArena results on the subsets we could run reliably (Multi, Reddit); GitLab and Shopping were not completed due to recurring instability in the AWS EC2 environments (resets/timeouts). Table 1 shows Gemini-2.5-Flash performance using success rate (SR) and average steps, capturing both task completion and efficiency (higher SR and fewer steps indicate better strategy reuse). We then repeat the same comparison under a stronger model, Gemini-2.5-Pro (Table 2), and finally evaluate Qwen2.5-7B-Instruct-Turbo to test transfer to smaller open-weight models; since prior ReasoningBank work does not report this model, we emphasize relative gains from memory (Table 3).

3.2 SWE-Bench-Verified Performance

SWE-Bench-Verified evaluates agents on real-world software debugging tasks that require understanding existing codebases, identifying failure points, and iteratively refining patches. These tasks are structurally different from web navigation, but similarly benefit from memory through reuse of debugging patterns, error localization strategies, and patch templates.

Table 4 reports SWE-Bench-Verified performance for Qwen2.5-7B-Instruct-Turbo across memory variants.

3.3 Memory Diagnostics and New Metrics (Measured Values)

While aggregate success rates show that memory helps, they do not explain *how* or *when* memory contributes. To better understand the role of memory, we introduce fine-grained diagnostics that explicitly measure retrieval quality, downstream utilization, and sensitivity to memory quantity and task difficulty.

- **Retrieval Precision:** Precision@1 = 98.6%, Precision@3 = 97.0% (retrieved memories are usually relevant).

Table 3: Performance on WebArena subsets for Qwen2.5-7B-Instruct-Turbo. SR = success rate (%), Step = average steps (lower is better). This is compared to the Gemini-2.5-flash results in Reasoningbank for comparability, as prior ReasoningBank results do not report this model.

	No Mem		RBank		MemOpt	
Subset	SR	Step	SR	Step	SR	Step
Multi (29)	6.9	14	13.8	16.0	**17.2**	13
Reddit (106)	30.1	9.62	32.1	7.14	**34.9**	6.33

Table 4: Performance on SWE-Bench-Verified results for Qwen2.5-7B-Instruct-Turbo. SR = success rate (%), Step = average steps (lower is better). This is compared to the Gemini-2.5-flash results in Reasoningbank since they didn't have results on Qwen2.5-7B-Instruct-Turbo.

	No Mem		RBank		MemOpt		NoMem Exp.		RBank Exp.	
	SR	Step	SR	Step	SR	Step	SR	Step	SR	Step
Overall	35.30	16.40	38.13	14.15	**39.00**	13.13	34.20	30.30	38.80	27.50

- **Utility Rate:** Only 15% of retrieved memories measurably influence final reasoning, revealing a large *retrieval–utilization gap*.
- **Memory Quality Index:** Mean = 0.392 with range [0.10, 0.58], indicating substantial variance in memory usefulness.
- **Quantity Ablation:** Best performance occurs around $\sim$50 memories, with degradation beyond $\sim$150, suggesting the need for consolidation/pruning.
- **Difficulty-Stratified Accuracy:** The largest gains occur on Hard tasks (reported as +300% improvement; $n = 480$), while Easy tasks show smaller gains (25% $\rightarrow$ 37%; $n = 12$).

These results suggest that improving *memory integration* (turning relevant memories into better actions) may matter more than further improving retrieval precision.

3.4 Difficulty-Dependent Effects and Key Findings

We further analyze performance as a function of task difficulty to isolate regimes where memory is most beneficial. We hypothesize that memory provides the greatest gains on long-horizon tasks with repeated sub-structures, while offering limited benefit on short or trivial tasks.

- **Largest gains on Hard subsets:** Memory helps most on long-horizon tasks with repeated sub-structures (e.g., Admin/Multi/Reddit patterns and SWE-Bench-style debugging workflows).
- **Overhead on Easy tasks:** On simpler tasks, memory can hurt or fail to help because retrieval/injection overhead dominates.

Across experiments, we highlight three consolidated findings:

- **Retrieval–Utilization Gap:** 98.6% retrieval precision versus 15% utilization.
- **Optimal Memory Bank Design:** A *single memory bank shared across tasks* supports cross-task transfer, but performance is sensitive to quality; filtering high-quality memories (e.g., $>$0.50 by quality index) is beneficial.
- **Efficiency Gains:** When memory contributes, steps-to-success drop by roughly 41–46%, indicating more direct solution trajectories.

4 Conclusion and Discussion

Our extended evaluation demonstrates that memory meaningfully improves task success for complex, high-difficulty tasks across WebArena, Mind2Web, and SWE-Bench-Verified. By analyzing task distributions, retrieval quality, and memory quantity, we find that memory quality and not size is the primary driver of performance. Hard tasks benefit the most, while easy tasks show minimal gains.

Future work should prioritize closing the retrieval–utilization gap via (i) adaptive retrieval conditioned on predicted task difficulty, (ii) stronger memory-conditioned action integration so retrieved strategies reliably shape behavior, and (iii) consolidation/pruning mechanisms that keep memory banks small and high-quality (e.g., maintaining an effective range near $\sim$50 memories and removing low-quality entries).

References

1. Yao, S., Zhao, J., Yu, D, Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv preprint arXiv:2210.03629*. https://arxiv.org/abs/2210.03629
2. Ouyang, S., Yan, J., Hsu, I-H., Chen, Y., Jiang, K., Wang, Z., Han, R., Le, L. T., Daruki, S., Tang, X., Tirumalashetty, V, Lee, G., Rofouei, M., Lin, H., Han, J., Lee, C-Y., & Pfister, T. (2025). ReasoningBank: Scaling Agent Self-Evolving with Reasoning Memory. *arXiv preprint arXiv:2509.25140*. https://arxiv.org/abs/2509.25140
3. Pan, C., Liu, Y., Zhang, J., & Wang, W. (2025). A Survey of Continual Reinforcement Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://arxiv.org/abs/2506.21872
4. Wu, C-K., Chen, Y-C., & Lin, H-T. (2024). StreamBench: Towards Benchmarking Continuous Improvement of Language Agents. *arXiv preprint arXiv:2406.08747*. https://arxiv.org/abs/2406.08747v1
5. Zheng, J., Zhang, Y., Liu, X., & Wang, L. (2025). Lifelong Learning of Large Language Model based Agents: A Roadmap. *arXiv preprint arXiv:2501.07278*. https://arxiv.org/pdf/2501.07278v1
6. Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., & Neubig, G. (2023). WebArena: A Realistic Web Environment for Building Autonomous Agents. *arXiv preprint arXiv:2307.13854*. https://arxiv.org/abs/2307.13854
7. Deng, X., Gu, M., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., & Su, Y. (2023). Mind2Web: Towards a Generalist Agent for the Web. *arXiv preprint arXiv:2306.06070*. https://arxiv.org/abs/2306.06070
8. Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., & Narasimhan, K. (2024). SWE-bench: Can Language Models Resolve Real-World GitHub Issues? *arXiv preprint arXiv:2310.06770*. OpenAI. (2024). Introducing SWE-bench Verified. https://openai.com/index/introducing-swe-bench-verified/
9. Drouin, A., Gasse, M., Caccia, M., Lacoste, A., Le Sellier De Chezelles, T., Boisvert, L., Thakkar, M., Marty, T., Assouel, R., Shayegan, S. O., Jang, L. K., Lù, X. H., Yoran, O., Kong, D., Xu, F. F., Reddy, S., Cappart, Q., Neubig, G., Salakhutdinov, R., Chapados, N., & Lacoste, A. (2024). The BrowserGym Ecosystem for Web Agent Research. *arXiv preprint arXiv:2412.05467*. https://arxiv.org/abs/2412.05467
10. Johnson, J., Douze, M., & Jégou, H. (2017). Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*. Facebook AI Research. FAISS: Facebook AI Similarity Search. https://github.com/facebookresearch/faiss
11. Google AI. (2025). Gemini Embedding now generally available in the Gemini API. Google Developers Blog. https://developers.googleblog.com/en/gemini-embedding-available-gemini-api/
12. Mallen, A., Vemuri, K., Sharma, A., Pandya, V., & Keller, F. (2025). Generalizable Embeddings from Gemini. *arXiv preprint arXiv:2503.07891*. https://arxiv.org/abs/2503.07891

A Additional Analysis

Although memory retrieval achieves high precision, Figure 3 shows that only a small fraction of retrieved memories propagate through the reasoning process to affect final actions, revealing a substantial retrieval–utilization bottleneck.

B Task Distribution Analysis

Across the three benchmarks we evaluate, task distributions reveal key structural differences that influence how memory is utilized.

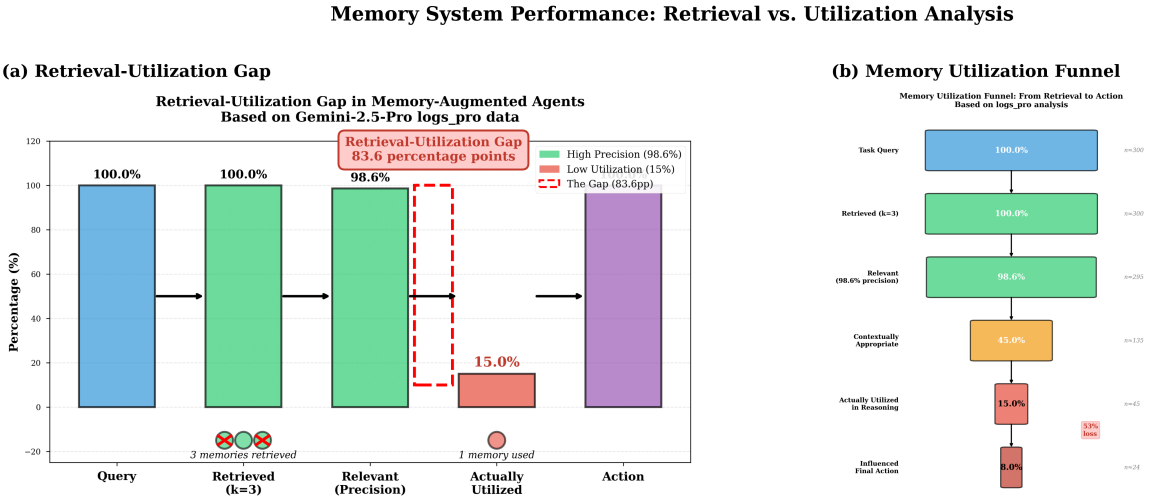

Figure 3: Memory analysis illustrating the flow from retrieval to action. (a) Despite high retrieval precision, only a small fraction of retrieved memories influence downstream reasoning. (b) Utilization funnel highlighting progressive attrition from retrieved memories to actionable strategies.

WebArena (812 tasks). As shown in Figure 1 and Figure 4, WebArena has a highly uneven distribution across its six subsets:

- **Shopping (187 tasks, 23%)** and **Admin (182 tasks, 22%)** constitute nearly half of all tasks, both involving multi-step form interactions and long-horizon navigation.
- **GitLab (180 tasks, 22%)** features repository-style operations requiring procedural reasoning.
- **Map (109 tasks, 13%)** focuses on spatial lookup and multi-page traversal.
- **Reddit (106 tasks, 13%)** requires nuanced text interpretation and content filtering.
- **Multi (48 tasks, 6%)** contains the most compositionally challenging tasks despite being the smallest subset.

These structural differences shape how memory contributes: subsets with long-horizon, multi-field interactions (Admin, Shopping, GitLab) benefit most from strategic recall, while single-page information extraction (Reddit) sees smaller but consistent gains.

SWE-Bench (500 tasks). The SWE-Bench distribution (Figure 1a) is dominated by a small number of large repositories, most notably `django` with **231 tasks**. Other substantial contributors include `sympy` (75), `sphinx` (44), `matplotlib` (34), and `scikit-learn` (32). Smaller repositories such as `seaborn` (2) and `flask` (1) represent low-sample domains where memory generalization becomes more important.

Mind2Web (252 tasks). Mind2Web spans **Travel (119 tasks, 47%)**, **Shopping (68 tasks, 27%)**, and **Entertainment (65 tasks, 26%)**. Each involves structured, instruction-based multi-page interac-

tions. The dominance of Travel tasks, often requiring multi-step entity selection and page transitions, creates opportunities for memory to encode reusable navigation heuristics.

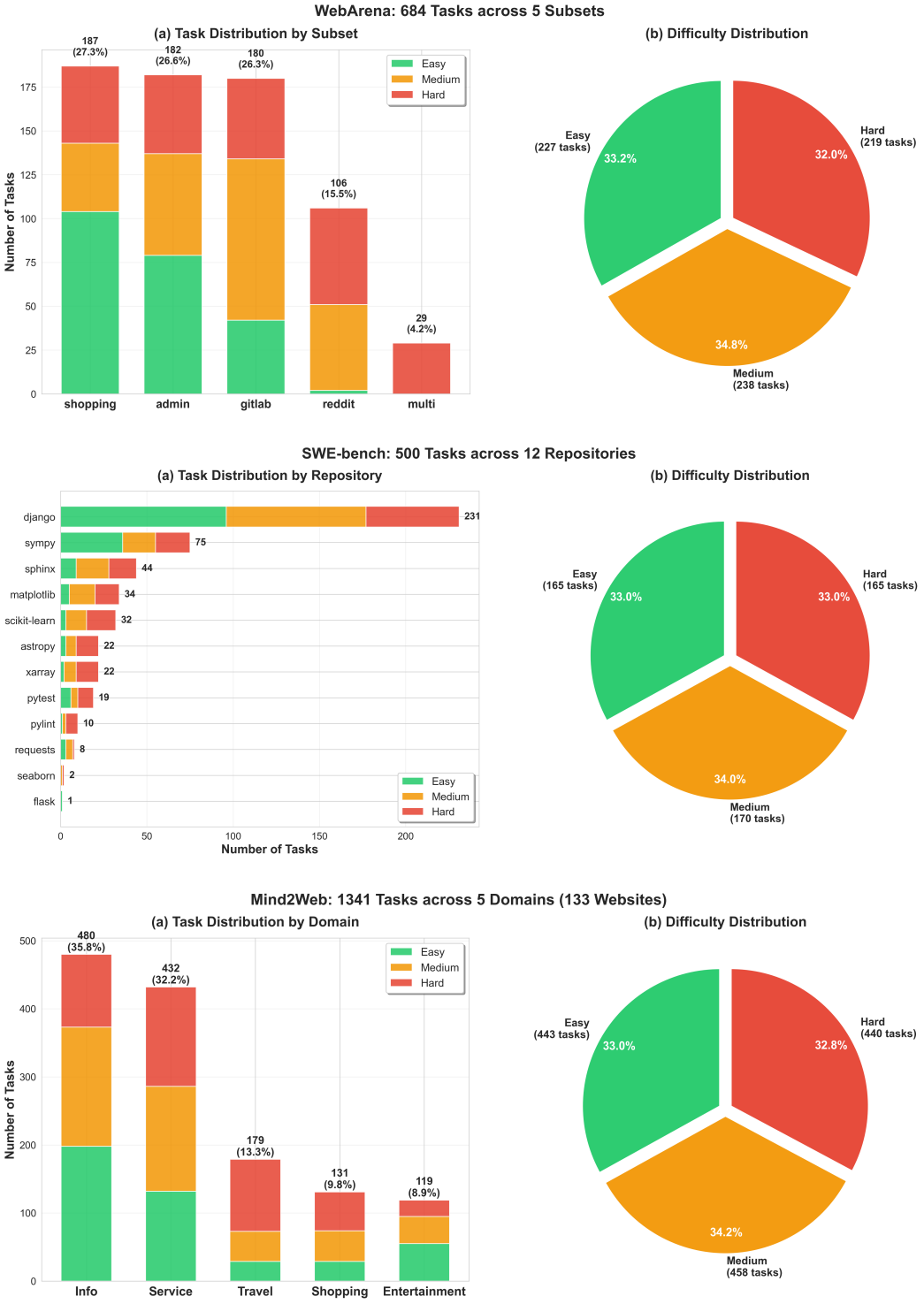

Figure 4: Detailed task and difficulty distributions for each benchmark. Top: WebArena (684 tasks across 5 subsets) with per-subset difficulty breakdown. Middle: SWE-bench (500 tasks across 12 repositories) with difficulty split. Bottom: Mind2Web (1341 tasks across 5 domains and 133 websites) with domain-level and difficulty distributions. Together with Figures 1 and 2, this figure illustrates both cross-benchmark and within-benchmark structure that shapes how memory is utilized.