SKILLTRACE: TRAVERSING A QUERY–SKILL GRAPH FOR COMPOSABLE LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model agents increasingly solve complex tasks by composing reusable skills from a library. To address this, the key challenge is not merely to retrieve individually relevant skills, but to identify a complete and executable skill composition. In this paper, we argue that this problem can be solved in a graph with three levels: compositional relations among skill queries, similarity between queries and candidates in the skill library, and the dependencies among the selected candidates. We introduce SkillTrace, which organizes the user query into a semantic hierarchy, matches skill queries and candidates, and propagates over the skill dependencies. Experiments on SkillsBench and ALFWorld demonstrate that SkillTrace achieves state-of-the-art performance, reaching a success rate of 53.17% on SkillsBench and 91.43% on ALFWorld. SkillTrace also delivers consistent improvements across different backbone language models, demonstrating the generality and robustness of graph-based skill retrieval.

1 INTRODUCTION

Large language model agents have shown increasing promise in solving complex tasks through reasoning, planning, and interaction with external environments. A growing line of work equips these agents with reusable skills, where each skill encapsulates a task-specific capability, executable procedure, or tool-use routine that can be invoked when needed (Wang et al., 2023). For example, SkillsBench represents skills as structured packages of procedural knowledge, including instructions, scripts, and resources for completing expertise-intensive workflows (Li et al., 2026b). ToolBench evaluates API-based capabilities, where an agent must select appropriate tools and compose them into a valid sequence of calls to fulfill single- or multi-tool instructions (Qin et al., 2023). In embodied environments such as ALFWorld, reusable capabilities can take the form of high-level policies (Shridhar et al., 2020). Despite their different forms, these benchmarks share a common requirement: an agent is required to identify and coordinate the capabilities needed to complete a complex task.

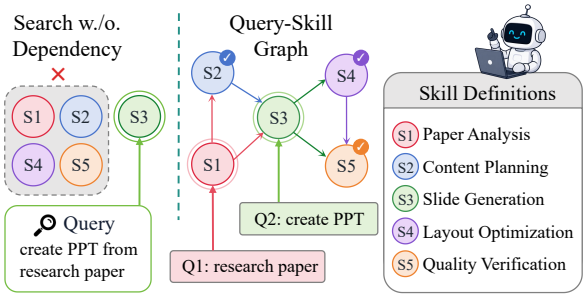


Figure 1: **Motivation.** We argue that independent skill retrieval may overlook skill dependencies, while SkillTrace connects atomic queries and skills through a query–skill graph, enabling dependency-aware traversal to retrieve a composable skill set.

Unlike conventional information retrieval (Karpukhin et al., 2020), retrieval-augmented generation (Lewis et al., 2020), and tool selection (Patil et al., 2024; Qin et al., 2023), composable skill retrieval requires identifying a coherent set of skills that jointly enable task execution. This setting introduces three key challenges. First, a complex user query should be decomposed into atomic skill queries while preserving their compositional relations. Second, matching skill queries to candidate skills requires both semantic alignment in task functionality and compatibility between their input–output specifications. Third, the selected skills should be completed according to their dependencies, since prerequisite or complementary skills may not be explicitly specified in the original

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.

query. These challenges motivate a unified graph formulation that jointly models relations among skill queries, connections between skill queries and candidate skills, and dependencies among skills.

In this paper, we argue that this challenge can be naturally formulated as graph search. As shown in Fig. 1, specifically, according to the challenge mentioned above, skill retrieval also involves three levels of relations. First, a complex query can be decomposed into finer-grained skill queries that are compositionally related to one another. These relations preserve how atomic requirements jointly constitute the original task. Second, each skill query is semantically connected to candidate skills in the library, reflecting which skill can best fulfill a particular requirement. Third, candidate skills are themselves connected through dependency relations, since executing one skill may require prerequisite or complementary skills that are not explicitly mentioned in the user query. Together, these relations form a Query–Skill Graph that connects the internal structure of the query with the organization of the skill library. Searching this graph provides a principled way to move beyond isolated semantic retrieval and identify a coherent skill composition.

We evaluate SkillTrace on two well-established benchmarks, *i.e.*, SkillsBench (Li et al., 2026b) and ALFWorld (Shridhar et al., 2020), covering cross-domain procedural task solving and embodied multi-step interaction. SkillTrace achieves state-of-the-art performance on both benchmarks, reaching 53.17% success rate on SkillsBench and 91.43% success rate on ALFWorld. It also consistently improves different backbone language models, demonstrating the generality of the proposed Query–Skill Graph formulation. These results further validate the benefit of explicitly structuring skill retrieval as a graph, which enables the agent to jointly exploit query composition, query–skill correspondence, and inter-skill dependencies when identifying executable skill compositions.

2 Related Work

Composable skills for LLM agents. LLM agents increasingly rely on reusable skills to solve complex tasks. A skill packages procedural knowledge for a specific capability. It may include instructions, executable scripts, reference materials, and task-solving routines. Skills can be reused across tasks rather than generated from scratch for each request. More importantly, complex tasks often require several skills to work together. These skills may cover different subtasks, depend on one another, or need to be applied in a particular order. Effective skill augmentation therefore requires more than retrieving a single relevant skill. The agent needs to identify a suitable skill set and use its members coherently during execution. SkillsBench shows that skill utility varies across tasks and that poorly matched skills can reduce agent performance (Li et al., 2026b). SRA-Bench further divides skill augmentation into retrieval, incorporation, and execution, showing that successful retrieval does not guarantee successful use (Su et al., 2026). Skills in the Wild studies large and noisy skill repositories, where distractors and overly general skills make suitable capabilities harder to identify and combine (Liu et al., 2026b).

Skill retrieval methods. As skill libraries grow, placing all available skills in the prompt becomes increasingly ineffective. Recent methods therefore retrieve a compact set of task-relevant skills before execution. SkillRet improves query–skill matching through retrieval-specific training, while SkillRouter adopts a retrieve-and-rerank pipeline that considers more complete skill content (Cho et al., 2026; Zheng et al., 2026). However, these methods largely score skills as independent candidates and return a ranked top-k list. Such a list can identify individually relevant skills, but it cannot determine whether they depend on one another, conflict with each other, provide redundant functionality, or require a specific execution order. To move beyond independent skill ranking, graph-based methods explicitly model relations among skills. SkillNet organizes skills with an ontology and a relation graph, while AgentSkillOS retrieves skills through a capability tree and composes them into task-specific directed acyclic graphs (Liang et al., 2026; Li et al., 2026a). Graph-of-Skills further uses graph propagation to expand initially relevant skills with their prerequisites (Liu et al., 2026a). SkillGraph and SkillDAG incorporate execution experience to update skill nodes and relations (Li et al., 2026c; Bai et al., 2026). Nevertheless, constructing a skill graph does not directly determine which connected skills should be selected together for a specific task. Existing methods often rely on fixed propagation, predefined update rules, or LLM-based orchestration, rather than jointly learning a task-specific executable subgraph under dependency and context-budget constraints.

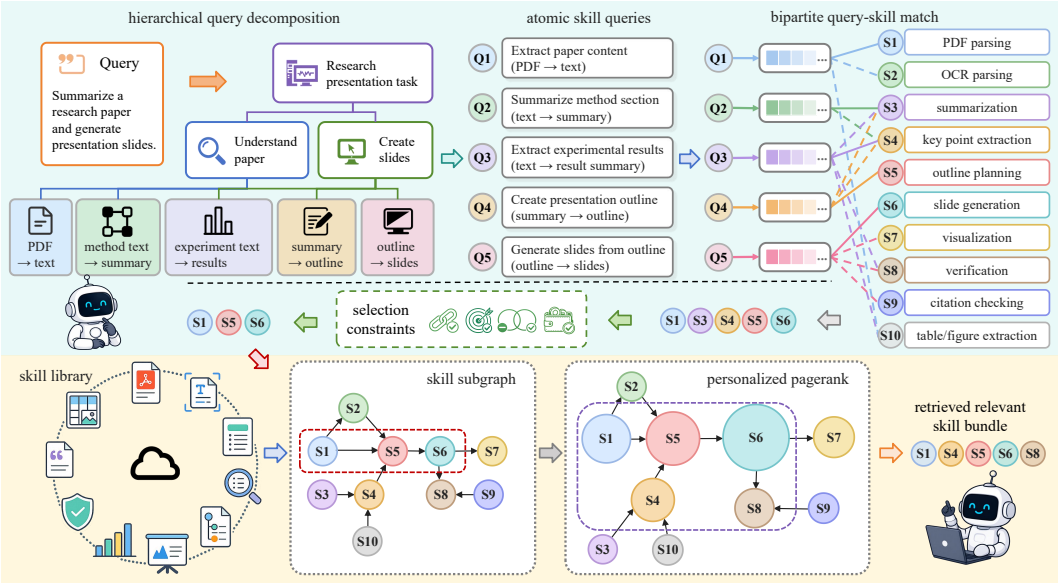


Figure 2: **Overview of SkillTrace with an intuitive example.** SkillTrace decomposes a query into atomic skill queries, identifies seed skills through bipartite matching, and uses personalized PageRank with selection constraints to retrieve a composable skill bundle.

3 METHOD

3.1 PROBLEM DEFINITION

Our goal is to retrieve a set of skills from the skill library for a given complex query, ensuring that the selected skills cover the query requirements and include the dependencies needed for successful execution. This approach enables an LLM agent to complete the target task with a complete and executable skill composition. In order to retrieve the desired skill set effectively, we propose the SkillTrace framework.

Given a user query x and a skill library $\mathcal{S} = \{s_1, \ldots, s_M\}$, our goal is to retrieve a subset of skills that can be composed to accomplish the task specified by x. Let selected skill subset $\mathcal{S}' \subseteq \mathcal{S}$, and let $\mathcal{R}(x, \mathcal{S}')$ measure the resulting task success rate. The objective of composable skill retrieval is formulated as

$$\mathcal{S}_x^* = \arg\max_{\mathcal{S}' \subseteq \mathcal{S}} \mathcal{R}\left(x, \mathcal{S}'\right). \tag{1}$$

Unlike independent skill retrieval, the utility of a skill subset is determined jointly by its constituent skills. The selected skills should collectively cover the requirements of the query, match the corresponding task content and input-output specifications, and include the dependencies required for execution. Retrieving more skills does not necessarily improve task performance, since irrelevant, redundant, or incompatible skills may interfere with skill composition and agent execution.

Directly optimizing Eq. 1 is usually not practical as evaluating a candidate skill composition requires executing the complete downstream task, which is often time-consuming and computationally expensive. Moreover, task performance can be observed only after execution and generally provides no tractable differentiable objective for directly optimizing the discrete skill subset. Exhaustively evaluating possible skill combinations is also infeasible as the search space (*i.e.*, skill library size) grows exponentially with the size of the skill library. We therefore formulate composable skill retrieval as a structured search problem and develop SkillTrace to efficiently identify promising skill compositions without repeatedly executing the downstream task.

3.2 QUERY-SKILL GRAPH CONSTRUCTION.

As shown in Fig. 2, SkillTrace instead addresses this objective through a structured graph traversal process. Given an input query x and a skill library $\mathcal{S}$, SkillTrace first decomposes x into atomic skill

queries, then identifies a primary skill for each atomic requirement through query–skill assignment, and finally traverses the dependencies among skills to retrieve the supporting skills required for execution. We organize these relations into a Query-Skill Graph:

$$\mathcal{G}_{x,\mathcal{S}} = \mathcal{T}_x \cup \mathcal{B}_{x,\mathcal{S}} \cup \mathcal{D}_{\mathcal{S}}, \tag{2}$$

with $\mathcal{T}_x$, $\mathcal{B}_{x,\mathcal{S}}$, and $\mathcal{D}_{\mathcal{S}}$ being the hierarchical query tree, the query-skill bipartite graph, and the skill dependency graph, respectively. Here, $\mathcal{T}_x$ is determined by the input query x, $\mathcal{B}_{x,\mathcal{S}}$ jointly depends on the query x and the skill library $\mathcal{S}$, and $\mathcal{D}_{\mathcal{S}}$ represents the global dependency structure defined over $\mathcal{S}$.

Hierarchical query tree. Given an input query x, we prompt a language model p_θ to organize its requirements into a hierarchical semantic tree. Formally, the tree is generated by

$$\mathcal{T}_x \sim p_\theta\left(\mathcal{T} \mid x, p_{\text{atom}}\right), \tag{3}$$

where p_{atom} is the atomic decomposition prompt that specifies the output schema and the atomicity constraints. Internal nodes represent higher-level actions that group related requirements, whereas each leaf represents a self-contained atomic skill query. The atomic skill query set is obtained as $\mathcal{A}_x = \text{Leaf}\left(\mathcal{T}_x\right)$. Each atomic skill query specifies one task requirement, its input and output types, and a rationale for why it can be primarily fulfilled by one skill. The prompt requires each leaf to describe exactly one input–output transformation, prohibits decomposition below the skill level, and prevents the introduction of requirements absent from the original query.

Query-Skill bipartite graph. Given the atomic skill queries $\mathcal{A}_x$ and the skill library $\mathcal{S}$, SkillTrace constructs a weighted query-skill bipartite graph

$$\mathcal{B}_{x,\mathcal{S}} = \left(\mathcal{A}_x, \mathcal{S}, \mathcal{E}^B_{x,\mathcal{S}}, \mathcal{W}^B_{x,\mathcal{S}}\right), \tag{4}$$

where $\mathcal{A}_x$ and $\mathcal{S}$ are two disjoint node sets, $\mathcal{E}^B_{x,\mathcal{S}} \subseteq \mathcal{A}_x \times \mathcal{S}$ denotes the query-skill edges, and graph weights $\mathcal{W}^B_{x,\mathcal{S}}$ denotes their semantic compatibility scores computed by embedding similarity in feature space. Embeddings are extracted from OpenAI embeddings (OpenAI, 2024). Specifically, the edge weight between an atomic skill query $q_i \in \mathcal{A}_x$ and a skill $s_j \in \mathcal{S}$ is defined as $w_{ij} = \text{sim}\left(f_Q(q_i), f_S(s_j)\right)$, where $f_Q(\cdot)$ and $f_S(\cdot)$ encode atomic skill queries and skills into a shared representation space, respectively, and $\text{sim}(\cdot,\cdot)$ denotes cosine similarity.

Since each atomic skill query is designed to be fulfilled by one primary skill, SkillTrace selects the highest-scoring skill for each query:

$$s_i^* = \underset{s_j \in \mathcal{S}}{\text{argmax}}\ w_{ij}, \qquad q_i \in \mathcal{A}_x. \tag{5}$$

The resulting primary skill set is $\mathcal{P}_{x,\mathcal{S}} = \{s_i^* \mid q_i \in \mathcal{A}_x\}$. These primary skills directly address the atomic requirements represented by the leaf nodes of $\mathcal{T}_x$ and serve as seed nodes for subsequent dependency-aware traversal over the skill graph.

SkillTrace then performs maximum-weight bipartite matching over $\mathcal{B}_{x,\mathcal{S}}$ to identify the primary skills. The matching maximizes the total semantic compatibility of the selected query–skill edges, subject to the constraint that each atomic skill query is matched with exactly one skill and each skill is matched with at most one atomic skill query. This one-to-one constraint prevents multiple atomic queries from selecting the same skill and encourages the matched skills to jointly cover the distinct atomic requirements of the input query.

The skills incident to the matched edges constitute the primary skill set $\mathcal{P}_{x,\mathcal{S}}$. These primary skills directly address the atomic requirements represented by the leaf nodes of $\mathcal{T}_x$ and serve as seed nodes for subsequent dependency-aware traversal over the skill graph.

Skill dependency subgraph. The skills selected through primary-skill assignment described in the last section may not be directly executable together, as one skill may rely on the output of another skill. We therefore organize the skill library as a directed skill dependency graph:

$$\mathcal{D}_{\mathcal{S}} = \left(\mathcal{S}, \mathcal{E}^D_{\mathcal{S}}, \mathcal{W}^D_{\mathcal{S}}\right), \tag{6}$$

where $\mathcal{S}$ denotes the skill-node set, $\mathcal{E}^D_{\mathcal{S}} \subseteq \mathcal{S} \times \mathcal{S}$ denotes the directed dependency-edge set, and $\mathcal{W}^D_{\mathcal{S}}$ contains the corresponding dependency weights. A directed edge $(s_i, s_j) \in \mathcal{E}^D_{\mathcal{S}}$ indicates that

Algorithm 1 SkillTrace for Composable Skill Retrieval

Input: query x, skill library $\mathcal{S}$, dependency threshold δ_D, number of supporting skills K, and restart probability α.
Begin:
$\mathcal{D}_{\mathcal{S}} \leftarrow (\mathcal{S}, \emptyset, \emptyset)$ ▷ dependency graph init.
for each ordered pair $(s_i, s_j) \in \mathcal{S} \times \mathcal{S}$ **do**
  $w_{ij}^D \leftarrow \mathrm{overlap}(\mathcal{O}_i, \mathcal{I}_j)$
  if $w_{ij}^D \geq \delta_D$ **then**
    Add (s_i, s_j, w_{ij}^D) to $\mathcal{D}_{\mathcal{S}}$
  end if
end for ▷ dependency construction
$\mathcal{T}_x \sim p_\theta(\mathcal{T} \mid x, p_{\mathrm{atom}})$, $\mathcal{A}_x \leftarrow \mathrm{Leaf}(\mathcal{T}_x)$ ▷ query decomposition
$\mathcal{B}_{x,\mathcal{S}} \leftarrow \{w_{ij} = \mathrm{sim}(f_Q(q_i), f_S(s_j))\}$ ▷ query-skill graph
$\sigma^* \leftarrow \mathrm{Hungarian}(\mathcal{B}_{x,\mathcal{S}})$ ▷ primary matching
$\mathcal{P}_{x,\mathcal{S}} \leftarrow \{s_{\sigma^*(i)} \mid q_i \in \mathcal{A}_x\}$
$\mathbf{p} \leftarrow \mathrm{Normalize}(\{w_{i,\sigma^*(i)}\})$ ▷ seed initialization
$\mathbf{r}^* \leftarrow \mathrm{ReversePPR}(\mathcal{D}_{\mathcal{S}}, \mathbf{p}, \alpha)$ ▷ dependency propagation
$\mathcal{S}_x^* \leftarrow \mathcal{P}_{x,\mathcal{S}} \cup \mathrm{TopK}_{\mathcal{S} \setminus \mathcal{P}_{x,\mathcal{S}}}(\mathbf{r}^*)$ ▷ skill augmentation
return $\mathcal{S}_x^*$

the output of skill s_i can be used as an input by skill s_j. The dependency edges and their weights are defined as

$$w_{ij}^D = \mathrm{match}(\mathcal{O}_i, \mathcal{I}_j), \qquad \mathcal{E}_{\mathcal{S}}^D = \{(s_i, s_j) \mid w_{ij}^D \geq \delta_D\}, \tag{7}$$

where $\mathcal{O}_i$ and $\mathcal{I}_j$ denote the output and input schema of skills s_i and s_j, respectively. Match function measures whether the $\mathcal{O}_i$ and the input $\mathcal{I}_j$ are matched, using the similarity score measured in the embedding feature space. If the resulting similarity score exceeds a predefined threshold δ_D, we consider the two skills connected and add the directed edge (s_i, s_j) to the dependency graph.

Skill augmentation. SkillTrace follows the Reverse Personalized PageRank (ReversePPR) paradigm (Yang et al., 2024) to augment the primary skill set with structurally relevant supporting skills. Specifically, we treat $\mathcal{P}_{x,\mathcal{S}}$ as personalized seed nodes and propagate their relevance through the skill dependency graph. The query–skill matching scores of the primary skills are normalized to form the personalization vector $\mathbf{p}$, whose nonzero entries correspond to $\mathcal{P}_{x,\mathcal{S}}$.

Let $\mathbf{A}^D$ denote the weighted adjacency matrix of $\mathcal{D}_{\mathcal{S}}$, where $A_{ij}^D = w_{ij}^D$ if $(s_i, s_j) \in \mathcal{E}_{\mathcal{S}}^D$, and $A_{ij}^D = 0$ otherwise. Since a dependency edge is directed from a supporting skill to the skill that consumes its output, we additionally include reverse transitions to propagate relevance from a primary skill toward its prerequisites. The resulting reverse-aware PPR process is

$$\mathbf{T}^D = \mathrm{RowNorm}\big(\mathbf{A}^D + (\mathbf{A}^D)^\top\big), \quad \mathbf{r}^{(t+1)} = \alpha\mathbf{p} + (1-\alpha)(\mathbf{T}^D)^\top\mathbf{r}^{(t)}. \tag{8}$$

Here, $\mathrm{RowNorm}(\cdot)$ denotes row normalization, $\mathbf{T}^D$ is the reverse-aware dependency transition matrix, and $\mathbf{r}^{(t)}$ is the skill relevance vector at iteration t, initialized as $\mathbf{r}^{(0)} = \mathbf{p}$. The restart probability $\alpha \in (0, 1)$ controls the balance between retaining relevance on the primary skills and propagating it through their dependency neighborhoods. The iteration continues until convergence, yielding the stationary relevance vector $\mathbf{r}^*$.

Finally, SkillTrace retains the primary skills and augments them with the K highest-scoring supporting skills:

$$\mathcal{S}_x^* = \mathcal{P}_{x,\mathcal{S}} \cup \mathrm{TopK}_{s_i \in \mathcal{S} \setminus \mathcal{P}_{x,\mathcal{S}}}(r_i^*), \tag{9}$$

where K specifies the maximum number of supporting skills added through dependency-aware propagation.

Algorithm 1 summarizes the complete traversal of the Query-Skill Graph. Overall, SkillTrace first traverses query composition through $\mathcal{T}_x$, then resolves query–skill correspondence through global bipartite matching on $\mathcal{B}_x$, and finally recovers execution dependencies through reverse-aware propagation on $\mathcal{G}_S$. This staged traversal converts a complex query into a query-aligned and dependency-aware skill composition.

4 EXPERIMENT

We evaluate whether traversing the Query-Skill Graph improves task performance, skill retrieval quality, and execution efficiency over full-library access and existing skill retrieval methods.

4.1 EXPERIMENT SETUP

Benchmarks. We evaluate SkillTrace on two benchmarks, SkillsBench (Li et al., 2026b) and ALFWorld (Shridhar et al., 2020). SkillsBench contains expertise-intensive technical tasks spanning eight domains, with skills represented as structured packages of instructions, scripts, templates, and reference materials (Li et al., 2026b). Many tasks require multiple skills to be identified and combined into a complete workflow. We conduct experiments on its Core, Extended, and Extreme subsets. ALFWorld is a text-based embodied benchmark, and we evaluate 140 tasks from its development in-distribution split. These tasks involve multi-step household interactions, including navigation, object search, and object manipulation (Shridhar et al., 2020). For analysis, we stratify the 140 ALFWorld episodes by the number of atomic skill queries identified through hierarchical query decomposition. We use this number as a proxy for compositional difficulty: more atomic skill queries indicate that a task contains more distinct requirements and therefore requires the agent to retrieve, coordinate, and execute a larger set of skills. Accordingly, episodes with one atomic skill query are categorized as Easy (43 episodes), those with two as Medium (73 episodes), and those with three or more as Hard (24 episodes). Together, the two benchmarks evaluate SkillTrace on both cross-domain technical problem solving and sequential embodied interaction.

Baselines. We compare SkillTrace with four baselines. *Vanilla Skills* provides the entire skill library to the agent without retrieval, allowing full access to all skills but introducing a large and potentially noisy context. *Vector Skills* retrieves a fixed number of skills according to the semantic similarity between the original query and each skill description. It treats skills independently and does not consider their dependencies. *SkillDAG* organizes skills and their relations as a directed acyclic graph, enabling graph-based skill selection while retaining the original query as a single retrieval unit (Bai et al., 2026). *GoS* first retrieves query-relevant seed skills and then propagates their relevance over a skill graph to identify additional skills connected through structural relations (Liu et al., 2026a).

Models and evaluation metrics. For the main comparison, we use GPT-5.6 as the backbone language model for all methods, so that the performance differences primarily reflect the effectiveness of different skill retrieval strategies. On SkillsBench, we report the number of successfully completed tasks on the Core, Extended, and Extreme subsets, together with the overall success rate SR (%). The subset counts include only fully completed tasks, whereas SR averages task-level scores that assign 1 to full completion and may award partial credit otherwise, as in prior graded task-success evaluation (Karimi et al., 2022). On ALFWorld, we report the number of successfully completed tasks in the Easy, Medium, and Hard groups, together with the overall success rate SR (%), computed from binary task outcomes. Higher values indicate better task performance.

Since the effectiveness of a skill retrieval method may vary across backbone models, we further evaluate SkillTrace with DeepSeek-V3.2 (Liu et al., 2025), Kimi-K2.5 (Kimi Team et al., 2026), Qwen3.5-397B-A17B (Qwen Team, 2026), and Gemini 3.1 Flash-Lite (Google DeepMind, 2026). In this cross model evaluation, we compare SkillTrace with GoS under the same skill library, task instructions, retrieval budget, and execution environment.

4.2 MAIN RESULTS

SkillTrace achieves the strongest overall performance on both benchmarks. As shown in Table 1, SkillTrace achieves the highest overall success rate of 53.17% on SkillsBench and the highest success rate of 91.43% on ALFWorld. On SkillsBench, it outperforms GoS, the strongest baseline, by 6.68 percentage points, and exceeds SkillDAG, Vector Skills, and Vanilla Skills by 11.79, 11.05, and 16.48 points, respectively. These substantial gains over both graph-based and similarity-based baselines indicate that semantic relevance or dependency modeling alone is insufficient; effective skill retrieval requires jointly covering the query requirements and completing the dependencies needed for execution.

Table 1: **Main results on SkillsBench and ALFWorld.** We show the superiority of SkillTrace over existing methods. Core, Extended, and Extreme are the three subsets divided in SkillsBench, while Easy, Medium, and Hard group ALFWorld tasks according to the number of required atomic skills. Category columns report the number of successful tasks, and SR reports the overall success rate (%) on SkillsBench or ALFWorld. The best and second-best results in each column are highlighted in dark green and light green, respectively.

Method	SkillsBench				ALFWorld			
	Core	Extended	Extreme	SR (%)	Easy	Medium	Hard	SR (%)
Vanilla Skills	2	17	10	36.69	38	**68**	20	90.00
Vector Skills	4	22	10	42.12	37	65	**21**	87.86
SkillDAG (Bai et al., 2026)	3	20	12	41.38	37	**68**	**21**	90.00
GoS (Liu et al., 2026a)	3	24	12	46.49	36	66	19	86.43
SkillTrace	**5**	**26**	**13**	**53.17**	**39**	**68**	**21**	**91.43**

On ALFWorld, SkillTrace surpasses SkillDAG, the strongest baseline, by 1.43 percentage points and improves over GoS and Vector Skills by 5.00 and 3.57 points, respectively. The consistent improvements across SkillsBench and ALFWorld demonstrate that SkillTrace generalizes well to substantially different settings, ranging from expertise-intensive technical tasks to long-horizon embodied interactions. This suggests that its query decomposition, query–skill matching, and dependency-aware traversal together produce skill compositions that are not only relevant but also complete and executable.

The overall improvements are consistent across task subsets. The advantage of SkillTrace is not driven by a single task category. On SkillsBench, it achieves the best results on Core, Extended, and Extreme, successfully completing 5, 26, and 13 tasks, respectively. Compared with the strongest baseline in each subset, SkillTrace solves one more Core task, two more Extended tasks, and one more Extreme task. On ALFWorld, SkillTrace successfully completes 39 Easy, 68 Medium, and 21 Hard tasks. It achieves the best result on Easy and matches the strongest results on Medium and Hard. The consistent category-wise performance shows that SkillTrace remains effective across tasks requiring different numbers and combinations of skills.

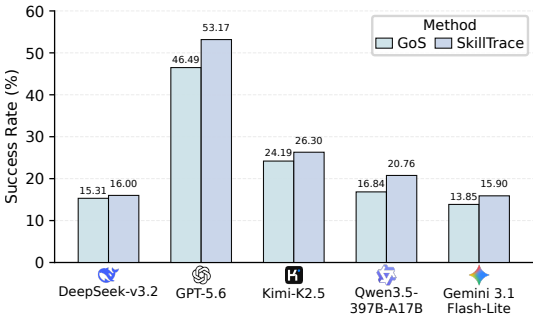


Figure 3: Success rate comparison between GoS and SkillTrace across five backbone language models on SkillsBench.

Task-specific graph traversal is more effective than flat retrieval and graph organization alone. On SkillsBench, Vanilla Skills provides the entire skill library to the agent and obtains an overall success rate of 36.69%, while Vector Skills independently retrieves skills according to semantic similarity and reaches 42.12%. SkillDAG obtains 41.38%, indicating that organizing skills into a graph does not by itself guarantee improved task performance. GoS increases the success rate to 46.49% by propagating relevance over skill relations, while SkillTrace further improves it to 53.17% by aligning atomic task requirements with primary skills and recovering their dependencies through graph traversal. A similar advantage is observed on ALFWorld, where SkillTrace achieves 91.43%, outperforming all flat and graph-based baselines. These results show that effective skill retrieval requires not only modeling skill relations, but also traversing them according to the specific requirements of each query.

4.3 Cross Model Generalization

In the Table 1, we use GPT-5.6 as the common backbone model to control the influence of model capability and focus the comparison on different skill retrieval methods. However, results obtained with a single backbone cannot determine whether the improvement of SkillTrace comes from its retrieval mechanism or from the particular reasoning and execution capabilities of GPT-5.6. We

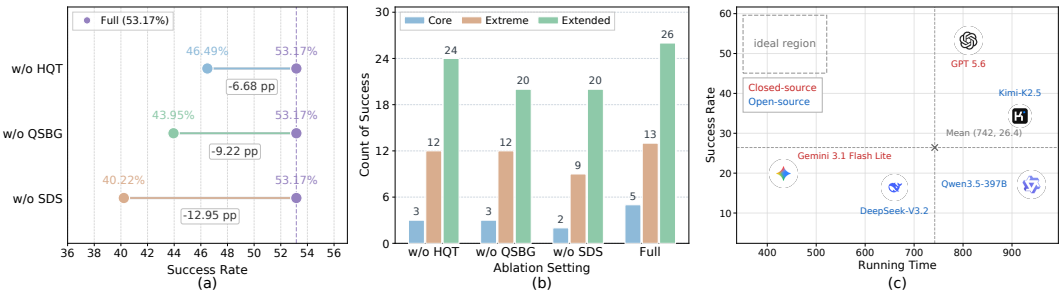


Figure 5: **Additional analysis.** (a) Overall success rates of the full SkillTrace and its variant ablations without the Hierarchical Query Tree (*w/o HQT*), Query-Skill Bipartite Graph (*w/o QSBG*), or Skill Dependency Subgraph (*w/o SDS*). (b) Numbers of successfully completed tasks on the Core, Extreme, and Extended subsets under each ablation setting. (c) Execution time and success rate of SkillTrace across five backbone models, where we have the success rate (%) and the end-to-end execution time in seconds.

therefore conduct an additional cross model experiment to examine whether SkillTrace remains effective when paired with backbone models of different capability levels. Specifically, we compare SkillTrace with GoS using five language models under the same skill library, retrieval budget, task instructions, and execution environment.

As shown in Fig. 3, SkillTrace consistently outperforms GoS across five backbone models. It improves the success rate by 0.69 points on DeepSeek-V3.2, 6.68 points on GPT-5.6, 2.11 points on Kimi-K2.5, 3.92 points on Qwen3.5-397B-A17B, and 2.05 points on Gemini 3.1 Flash-Lite. Although the magnitude of improvement varies across models, the gain remains positive in every case. This result indicates that the advantage of SkillTrace is not specific to GPT-5.6 and can generalize to backbones with different reasoning, composition, and execution capabilities.

The overall success rate nevertheless varies substantially across backbone models. With SkillTrace, GPT-5.6 achieves a success rate of 53.17%, whereas DeepSeek-V3.2, Kimi-K2.5, Qwen3.5-397B-A17B, and Gemini 3.1 Flash-Lite obtain 16.00%, 26.30%, 20.76%, and 15.90%, respectively. This variation suggests that skill retrieval determines whether relevant and composable skills are provided, while the backbone model still affects how accurately these skills are understood and executed.

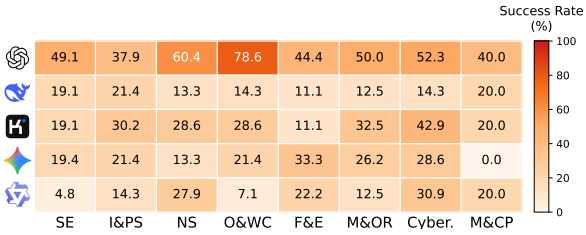


Figure 4: Category-wise performance of SkillTrace with five backbone language models across the eight SkillsBench task domains. The full names of the category abbreviations are provided in the appendix.

To further examine whether these differences are uniformly distributed across task domains, we report the category-wise performance of SkillTrace in Fig. 4. GPT-5.6 achieves the highest performance across all eight SkillsBench categories, with particularly strong results in Office & White Collar and Natural Science. The remaining models exhibit more pronounced domain-specific variation. Kimi-K2.5 performs relatively well in Industrial & Physical Systems, Mathematics & OR, and Cybersecurity, while Gemini 3.1 Flash-Lite obtains its strongest relative result in Finance & Economics. These results show that different backbones retain distinct domain strengths and weaknesses, while the consistent improvements over GoS demonstrate that SkillTrace provides a generally effective retrieval mechanism across models.

4.4 Ablation Study

We conduct ablation experiments on SkillsBench to examine the contributions of the Hierarchical Query Tree (HQT), Query-Skill Bipartite Graph (QSBG), and Skill Dependency Subgraph (SDS). As shown in Fig. 5 (a) and (b), the full SkillTrace achieves an overall success rate of 53.17%, completing 5, 26, and 13 tasks in the Core, Extended, and Extreme subsets, respectively. Without HQT, the original query is used as a single retrieval unit for primary-skill retrieval before dependency-

aware augmentation. This variant achieves 46.49%, corresponding to a decrease of 6.68 percentage points, showing that atomic query decomposition improves the coverage of distinct task requirements.

Retaining the HQT but removing the QSBG further reduces the success rate to 43.95%, a decrease of 9.22 percentage points. Successful Extended tasks drop from 26 to 20, suggesting that structured query–skill assignment prevents redundant skill selections from becoming propagation seeds. Finally, removing the SDS and directly returning the primary skills identified by the HQT and QSBG produces the largest performance degradation. The success rate decreases to 40.22%, while the numbers of successful Core, Extended, and Extreme tasks fall to 2, 20, and 9, respectively. This 12.95-point decrease highlights the importance of dependency-aware propagation for recovering prerequisite and supporting skills not explicitly expressed in the input query. Overall, the three components are complementary, *i.e.*, the HQT represents atomic requirements, the QSBG identifies primary skills, and the SDS completes the executable skill composition.

4.5 Efficiency Analysis

SkillTrace models richer graph structures while maintaining manageable computational complexity. Let N and E denote the numbers of skills and dependency edges, respectively, Q the number of atomic skill queries, and K the number of propagation iterations. The skill embeddings and dependency graph are constructed offline and reused across queries. Although dependency-graph construction may require pairwise comparisons among skills, this cost is incurred only once and is amortized over subsequent tasks. At inference time, constructing the query-skill similarity matrix requires $O(QN)$ computations. Maximum-weight bipartite matching requires $O(Q^2N)$ time when $Q \leq N$, while personalized PageRank requires $O(K(N+E))$ time. Therefore, the online graph-retrieval complexity is

$$O\big(QN + Q^2N + K(N+E)\big). \tag{10}$$

In practice, Q and K are small, and the skill dependency graph is sparse. SkillTrace thus avoids exhaustive search over skill combinations and remains approximately linear in the size of the skill library. The end-to-end runtime results are consistent with the practical efficiency of SkillTrace. On ALFWorld, SkillTrace takes 73.66 seconds, which is 6.57% less than GoS, while improving the success rate by 5.00 percentage points.

To examine whether stronger task performance is simply associated with longer execution time, Fig. 5 (c) compares the runtime and success rate of SkillTrace across five backbone language models. The Pearson correlation coefficient is only $r = 0.08$, indicating no clear linear relationship between runtime and task success rate on this yet. For example, GPT-5.6 achieves the highest success rate without having the longest runtime, whereas Qwen3.5-397B-A17B has the longest runtime but one of the lowest success rates. Given the limited number of evaluated backbones, this result should be interpreted as descriptive evidence rather than a statistically conclusive correlation test. Overall, these results show that the performance gains of SkillTrace cannot be explained solely by increased execution time.

5 Conclusion

In this paper, we introduced SkillTrace, a graph-based framework for retrieving composable skills for large language model agents. SkillTrace formulates skill retrieval through three levels of relations: compositional relations among skill queries, semantic correspondences between queries and candidate skills, and dependency relations among skills. It first organizes a complex user query into a semantic hierarchy, then identifies seed skills through query–skill matching, and finally traverses the skill graph to complete the required skill composition. Experiments on SkillsBench and ALFWorld show that SkillTrace achieves state-of-the-art performance, reaching 53.17% success rate on SkillsBench and 91.43% success rate on ALFWorld. Its consistent improvements across different backbone language models further demonstrate the generality and robustness of the proposed formulation. These results suggest that explicitly modeling query composition and skill dependencies is important for retrieving complete and executable skill sets, providing a promising direction for building more capable skill-based language model agents.

AI USE STATEMENT

We used AI tools during the preparation of this paper to revise English prose, adjust LaTeX formatting, and find potentially relevant literature. They were used to assist with writing. The authors checked suggested references against the original papers, reviewed and tested AI-assisted code, and edited the manuscript to ensure that it accurately describes the work carried out. The research questions, experimental design, analysis, and conclusions were determined by the authors, who take responsibility for the final paper.

ETHICS STATEMENT

This work studies composable skill retrieval for LLM agents and is evaluated on existing publicly available benchmarks, including SkillsBench and ALFWorld. It does not involve human participants, the collection of personal data, or the construction of datasets containing sensitive information. The experiments are conducted solely for research evaluation in benchmark and simulated environments rather than for real-world decision making.

REPRODUCIBILITY STATEMENT

The source code used in our experiments is included in the supplementary material. In this paper, we provide detailed information to support the reproducibility of our results. The SkillTrace framework and retrieval procedure are described in Section 3 and Algorithm 1. We use publicly available benchmarks, baselines, and evaluation metrics, which are presented in Section 4.1. Dataset statistics, hardware and software environments, and hyperparameters and experimental configurations are provided in Appendix A.1–A.3.

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.

Hongcheol Cho, Ryangkyung Kang, and Youngeun Kim. Skillret: A large-scale benchmark for skill retrieval in llm agents. *arXiv preprint arXiv:2605.05726*, 2026.

Google DeepMind. Gemini 3.1 flash-lite model card. `https://deepmind.google/models/model-cards/gemini-3-1-flash-lite/`, March 2026. Accessed: 2026-07-29.

Pegah Karimi, Emanuele Plebani, Aqueasha Martin-Hammond, and Davide Bolchini. Textflow: Toward supporting screen-free manipulation of situation-relevant smart messages. *ACM Transactions on Interactive Intelligent Systems*, 12(4):31:1–31:29, 2022. doi: 10.1145/3519263.

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

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Hao Li, Chunjiang Mu, Jianhao Chen, Siyue Ren, Zhiyao Cui, Yiqun Zhang, Lei Bai, and Shuyue Hu. Organizing, orchestrating, and benchmarking agent skills at ecosystem scale. *arXiv preprint arXiv:2603.02176*, 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.

Xiaoyuan Li, Moxin Li, Keqin Bao, Yubo Ma, Wenjie Wang, Dayiheng Liu, and Fuli Feng. Skillgraph: Skill-augmented reinforcement learning for agents via evolving skill graphs. *arXiv preprint arXiv:2605.12039*, 2026c.

Yuan Liang, Ruobin Zhong, Haoming Xu, Chen Jiang, Yi Zhong, Runnan Fang, Jia-Chen Gu, Shumin Deng, Yunzhi Yao, Mengru Wang, et al. Skillnet: Create, evaluate, and connect ai skills. *arXiv preprint arXiv:2603.04448*, 2026.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

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*, 2026a.

Yujian Liu, Jiabao Ji, Li An, Tommi Jaakkola, Yang Zhang, and Shiyu Chang. How well do agentic skills work in the wild: Benchmarking llm skill usage in realistic settings. *arXiv preprint arXiv:2604.04323*, 2026b.

OpenAI. New embedding models and api updates. `https://openai.com/index/new-embedding-models-and-api-updates/`, 2024.

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

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 *The twelfth international conference on learning representations*, 2023.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

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.

Weihang Su, Jianming Long, Qingyao Ai, Qiaozhi He, Yichen Tang, Changyue Wang, Yiteng Tu, Yingbo Wang, and Yiqun Liu. Skill retrieval augmentation for agentic ai. *arXiv preprint arXiv:2604.24594*, 2026.

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.

Mingji Yang, Hanzhi Wang, Zhewei Wei, Sibo Wang, and Ji-Rong Wen. Efficient algorithms for personalized pagerank computation: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 36(9):4582–4602, 2024. doi: 10.1109/TKDE.2024.3376000.

YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuan Zhu, Baohua Dong, and Hangcheng Zhu. Skillrouter: Retrieve-and-rerank skill selection for llm agents at scale. *arXiv preprint arXiv:2603.22455*, 4, 2026.

A Appendix

A.1 Dataset Details and Subset Statistics

We use two public benchmarks with complementary forms of compositionality. SkillsBench (Li et al., 2026b) evaluates whether an agent can identify and apply reusable skill packages to expertise-intensive technical tasks, whereas ALFWorld (Shridhar et al., 2020) evaluates sequential planning and execution in text-based household environments. This combination lets us measure skill retrieval in both artifact-producing technical workflows and long-horizon embodied interaction.

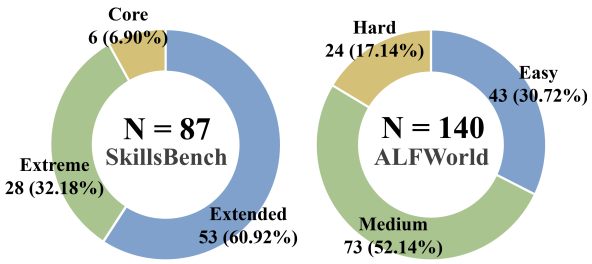


Figure 6: Benchmark subset statistics used in the main experiments. The charts show task counts and percentages for the SkillsBench subsets and ALFWorld analysis strata.

SkillsBench. We evaluate all 87 tasks in the public benchmark split. The tasks span eight domains and are grouped into the Core, Extended, and Extreme subsets used in the main paper. The split contains 6 Core, 53 Extended, and 28 Extreme tasks. Figure 6 reports their exact proportions, and Table 2 reports the domain composition.

ALFWorld. We evaluate all 140 episodes in the public development in-distribution (valid_seen) split. The official task types describe the household objective, rather than retrieval difficulty, and comprise Pick & Place, Examine in Light, Clean & Place, Heat & Place, Cool & Place, and Pick Two & Place. For the compositional analysis in the main paper, we create a fixed task-to-stratum mapping from the number of atomic skill queries in the hierarchical decomposition and reuse this mapping for every method. Episodes containing one atomic query are labeled Easy, those containing two are labeled Medium, and those containing three or more are labeled Hard. These labels are an analysis stratification introduced in this work, not official ALFWorld difficulty annotations. They measure the number of distinct requirements that must be coordinated, rather than merely the length of an observed action trajectory.

A.2 Hardware and Software Environment

Experiments were orchestrated on a Linux server with two AMD EPYC 9454 48-core processors (96 physical cores and 192 logical CPUs) and 125 GiB of system memory. The host ran Ubuntu 22.04.5 LTS with Linux kernel 6.8.0-124-generic. SkillsBench tasks were isolated in Docker containers and were allocated the CPU, memory, storage, network, and timeout limits declared by each task's public `task.toml`. ALFWorld was executed through its TextWorld interface. The backbone LLM and text-embedding model were accessed through OpenAI-compatible hosted APIs. The server was additionally equipped with seven NVIDIA GeForce RTX 4090 D GPUs, each providing 49,140 MiB of memory.

The Python environment was managed from a locked dependency file and used Python 3.12.13. The principal package versions were ALFWorld 0.4.2, TextWorld 1.7.0, NumPy 2.4.3, SciPy 1.17.1, NetworkX 3.6.1, Pydantic 2.12.5, OpenAI Python SDK 1.109.1, LiteLLM 1.80.0, and FastGraphRAG 0.0.5.

A.3 Hyperparameters and Experimental Configuration

All methods in a benchmark use the same task instances, skill library, backbone model, and task verifier. The main comparison uses GPT-5.6 as the agent backbone. SkillsBench uses a 200-skill library, while ALFWorld uses the 37 ALFWorld-specific skills contained in the same skill collection. Skill and query embeddings are produced by `text-embedding-3-large` with 3,072 dimensions. Embeddings and the skill graph are constructed once and cached. The cache is shared across queries but not across different skill libraries.

Table 2: Task-category composition of the evaluated SkillsBench and ALFWorld splits. Counts and within-benchmark percentages cover 87 SkillsBench tasks across eight domains and 140 ALFWorld episodes across six task types.

Benchmark	Category	n	Share (%)
SkillsBench	Software Engineering	16	18.39
	Industrial & Physical Systems	14	16.09
	Natural Science	14	16.09
	Office & White Collar	14	16.09
	Finance & Economics	9	10.34
	Mathematics & OR	8	9.20
	Cybersecurity	7	8.05
	Media & Content Production	5	5.75
ALFWorld	Pick & Place	35	25.00
	Examine in Light	13	9.29
	Clean & Place	27	19.29
	Heat & Place	16	11.43
	Cool & Place	25	17.86
	Pick Two & Place	24	17.14

For graph construction, each skill considers eight linking candidates and a dependency edge is retained at a matching threshold of 0.6. For GoS on SkillsBench, four seed skills are selected and at most five skills are returned. On ALFWorld, GoS uses five seeds and returns at most four skills in the initial bundle. Personalized PageRank uses restart probability $\alpha = 0.2$, at most 50 iterations, and an ℓ_1 convergence tolerance of 10^{-6}. In the GoS baseline implementation, reverse transitions are weighted by relation type: 1.0 for dependency, 0.5 for workflow, 0.2 for semantic, and 0.1 for alternative relations. SkillTrace instead uses the reverse-aware dependency transition defined in the Method section.

The SkillTrace retrieval procedure follows the Method section. Here we report only implementation and evaluation settings omitted from the main text. In particular, SkillTrace uses the same PPR convergence settings reported above, and its ALFWorld retrieval bundle is capped at six skills.

On ALFWorld, every method receives the same textual observation and action grammar, domain randomization is disabled, the environment seed is 42, and an episode is capped at 30 agent steps. On SkillsBench, one attempt is scheduled for each task in the main comparison and the task-specific verifier determines success. Infrastructure retries are limited to one and do not change the task success rate. Invalid or incomplete trials are recorded as errors rather than silently counted as successful completions.