WHAT DOES AN AGENT NEED? BUNDLE-AWARE AGENT RESOURCE COMPOSITION

Anonymous authors
Paper under double-blind review

ABSTRACT

As catalogs of models, tools, and reusable skills expand, determining which resources an agent should use becomes increasingly challenging. Tool-selection knowledge embedded in model parameters can lag behind catalog changes, while semantic top-k retrieval reduces context length but may retain noise and exclude complementary resources needed to satisfy implicit task requirements. Beyond retrieving relevant resources or reusing predefined agents, how to compose heterogeneous resources for a new task remains underexplored. We formulate **Task-Conditioned Agent Resource Composition (AgentComposition)**: jointly selecting a backbone model and composing a tool–skill bundle to meet task requirements. We propose **Retrieval-Grounded Bundle-Aware Agent Generation**, which retrieves resources and historical configurations as compositional evidence, recombines resources through constrained beam search with a fine-tuned generator, and reranks complete configurations using generator likelihood and a bundle-aware critic. Evaluated against reference configurations, our method achieves rank-discounted component recall and precision at 10 (RDCR@10/RDCP@10) of $60.00\%/49.18\%$ on AgentSelect and $68.11\%/58.30\%$ on SkillsBench. The results also reveal that leading LLM baselines can select relevant resources but struggle to recover complete configurations. Controlled transfer experiments show that AgentSelect-trained initialization improves adaptation to SkillsBench using only four target-domain queries, while end-to-end evaluation shows that predicted configuration rankings broadly align with execution quality. These results support resource composition as a learnable and transferable capability for task-specific agent construction. Our work shifts the focus from individual resource relevance to joint task suitability. AgentComposition thus offers a path toward more precise and personalized agents that adapt to changing resource ecosystems and continually evolve through accumulated task experience.

1 INTRODUCTION

From autoregressive text completion Radford et al. (2019); Brown et al. (2020), large language models (LLMs) have expanded to reasoning, function use, and environment interaction Yao et al. (2023); Schick et al. (2023); Qin et al. (2024b). The Model Context Protocol (MCP) Anthropic (2024) and Agent Skills Anthropic (2025b); Xu & Yan (2026) standardize access to tools, data, and reusable capabilities. Agent harnesses coordinate these capabilities around a backbone model through planning, memory, context management, and execution Microsoft (2026); OpenAI (2026); Anthropic (2025a). The growing resource catalog raises a task-specific question: **which capabilities should be assembled together?**

(1) Existing work addresses parts of this choice: tool learning enables function selection and use Qin et al. (2024a); Hao et al. (2023); Wang et al. (2025); retrieval exposes task-relevant tools in context Du et al. (2024); Qu et al. (2024); Yuan et al. (2024); model routing chooses a backbone Ong et al. (2025); and AgentSelect recommends predefined configurations Shi et al. (2026). These choices alone do not determine how to combine heterogeneous resources for a new task. *A seven-day, pet-friendly road trip under a fixed budget, for example, may require accommodation search, pet-policy checks, route planning, and cost calculation. Accommodation tools alone leave other needs unmet; a historical road-trip configuration may miss the pet constraint.* Historical configu-

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.

rations offer compositional evidence, but their components may need recombination. (2) Resource composition may support more adaptive and personalized agents Cheng et al. (2025); Hao et al. (2025), reduce irrelevant tool context Liu et al. (2026), and facilitate continual adaptation to dynamic tool ecosystems Wu et al. (2026b).

Motivated by filling existing research gap and its importance in agent area development. We therefore formulate the **Task-Conditioned Agent Resource Composition problem, abbreviated as AgentComposition**. Unlike automatic agent architecture search methods such as AgentSquare Shang et al. (2025), which optimize the mechanisms governing planning, reasoning, memory, and tool use, AgentComposition focuses on the capability resources available to an agent. Its goal is to select a backbone model and compose a tool–skill bundle that jointly meet the requirements of a target task. To address this problem, we propose **Retrieval-Grounded Bundle-Aware Agent Generation**. Retrieval first identifies task-relevant resources and historical configurations as compositional evidence. A fine-tuned generator then recombines these resources into candidate configurations through constrained beam search, after which a bundle-aware critic, together with the generator likelihood, reranks the candidates to produce the final configurations.

To evaluate whether our method can learn effective resource composition from historical task executions, we first conduct experiments on AgentSelect Shi et al. (2026). We construct model- and tool-retrieval modules, train a Llama-8B generator on real execution trajectories collected from over 40 datasets, and train a bundle-aware critic using perturbation-based augmentation and LLM supervision. Collaborative-filtering retrieval achieves the strongest resource recall among retrieval strategies, yet retrieved historical bundles remain substantially different from the gold configurations, highlighting the need for task-specific recomposition rather than direct reuse. Conventional bundle-recommendation methods also underperform in this setting, as they are primarily designed for closed-domain product recommendation rather than heterogeneous agent resources. Given contextual resource evidence, strong LLMs exhibit meaningful resource-selection ability, yet complete-configuration recovery remains limited and inconsistent across domains. Our method achieves the highest agent composition, reflected in multiple recall and precision metrics. Its advantage becomes more pronounced on SkillsBench Li et al. (2026).

Further analyses demonstrate that the learned composition capability can be efficiently transferred across resource domains. Despite limited zero-shot generalization to SkillsBench due to domain mismatch, our trained generator adapts with only four queries and 12 parameter updates, whereas a vanilla Llama-8B requires about 40 updates to reach comparable validation loss and still yields substantially weaker test performance. Structural ablations further show that constrained generation reduces invalid or hallucinated resources, beam search improves exploration, and critic reranking improves final precision and recall. Finally, end-to-end execution confirms that our predicted configuration rankings broadly align with actual execution quality, while a case study further illustrates how interactions among resources give rise to meaningful bundle effects.

To summarize, our contributions are: ❶ We formulate the **Task-Conditioned Agent Resource Composition (AgentComposition)** problem, extending agent-resource discovery beyond retrieval toward resource selection and recombination, a capability that remains underexplored in existing methods, including state-of-the-art LLMs. ❷ We propose a **Retrieval-Grounded Bundle-Aware Generation** framework that supports open-domain resource composition, constrains generation to valid resources, explores diverse configurations, and explicitly models bundle-level effects. ❸ Extensive experiments [1].demonstrate the effectiveness of our approach, including cross-domain adaptability, the contribution of its structural components, and strong alignment between predicted configuration rankings and end-to-end execution quality.

2 Related Work

Agent Construction and Resource Composition. Agent construction involves selecting capabilities and organizing their use. AgentSquare searches planning, reasoning, tool-use, and memory modules Shang et al. (2025); ADAS iteratively synthesizes and evaluates agent programs Hu et al. (2025b). SkillWeaver discovers and refines reusable skill APIs Zheng et al. (2025), while PSEC composes skill-specific LoRA modules in parameter space Liu et al. (2025). These methods modify

[1]The experiments can be reproduced at: https://anonymous.4open.science/r/AgentComposition-801C

architectures or develop capabilities. We compose existing catalog resources for a task, jointly selecting a backbone model and tool–skill bundle, complementary to model routing Ong et al. (2025).

Agent Resource Retrieval and Selection. Existing retrieval typically ranks individual resources by their relevance to a query Shi et al. (2025); Moon et al. (2024). This pointwise view is effective for identifying useful tools, but does not explicitly model how resources complement one another as a configuration. Generative approaches such as ToolkenGPT and ToolGen integrate tool selection into language-model generation Hao et al. (2023); Wang et al. (2025), but associate tools with learned identities or representations tied to a training inventory, which can require additional adaptation when new tools are introduced or the catalog changes. Our method instead treats resource selection as a composition problem, generating bundle configurations from retrieved resource without requiring a dedicated vocabulary entry for each resource.

Bundle Recommendation and Agent Discovery. Bundle recommendation ranks existing bundles or constructs new ones Sun et al. (2026). BGCN models user–item, user–bundle, and bundle–item interactions through graph convolutions Chang et al. (2020); CrossCBR aligns item- and bundle-level representations Ma et al. (2022). Construction methods include query-conditioned Text2Bundle Hu et al. (2025a), LLM-based adaptive in-context learning Sun et al. (2024), and discrete-diffusion generation of unordered bundles Tu et al. (2026). In agent discovery, AgentSearchBench and AgentSelect both formulate the problem as retrieving or ranking predefined agents or configurations from an existing catalog, rather than composing a new task-specific configuration Wu et al. (2026a); Shi et al. (2026). We instead formulate agent resources as composable bundle elements and generate configurations that are not predefined in the catalog.

3 METHOD

Problem formulation. *AgentComposition* maps a task description Q (including relevant session history) to ranked configurations. Given backbone models $\mathcal{M}$ and tools and skills $\mathcal{T}$, each configuration $A = (M, B)$ selects $M \in \mathcal{M}$ and a bundle $B \subseteq \mathcal{T}$. We return K distinct configurations, $\mathcal{G}(Q; \mathcal{M}, \mathcal{T}) = (A_Q^{(1)}, \ldots, A_Q^{(K)})$, ranked by estimated ability to satisfy the task.

Method overview. Our framework comprises three stages (Figure 1). Retrieval identifies task-relevant resources and supplies available historical configurations as compositional evidence. A fine-tuned generator then recombines these resources into candidate configurations through constrained beam search. Finally, bundle-aware reranking combines critic scores with generator likelihoods to produce the final ranked configurations. Algorithm 1 in Appendix C gives the procedure.

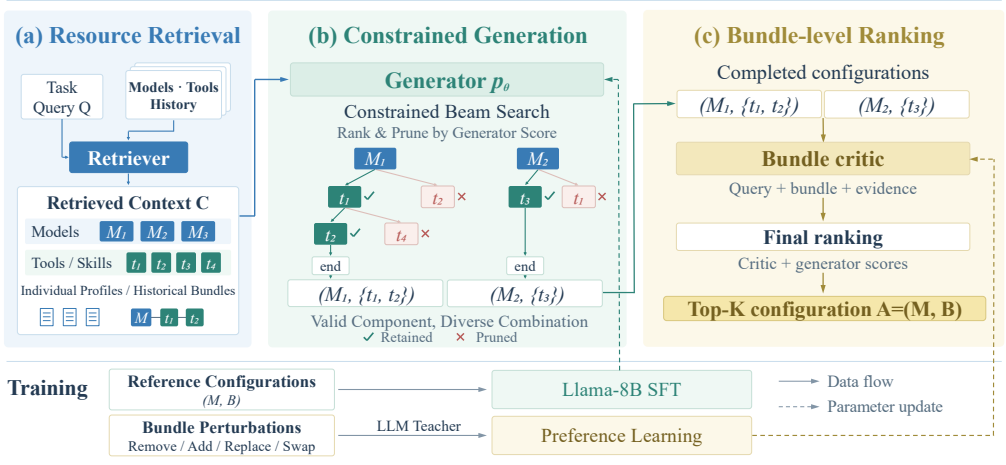


Figure 1: Retrieval-grounded, bundle-aware recommendation: (a) candidate retrieval; (b) constrained search with generator-based pruning; (c) final reranking. The lower panel shows generator and critic training; solid and dashed arrows indicate data flow and parameter updates, respectively.

3.1 RETRIEVAL CONTEXT

Retrieval provides the generator and critic with $\mathcal{C}(Q) = (\mathcal{M}_r(Q), \mathcal{T}_r(Q), \mathcal{E}_r(Q))$: retrieved LLMs, tools, and evidence comprising profiles, normalized scores, provenance, and intact historical bundles

when available. Semantic retrieval is the default; we additionally introduce a collaborative-signal retriever when query- agent interaction histories are available (Appendix B).

3.2 Structured Generator Fine-Tuning

We initialize p_θ from Meta-Llama-3-8B, apply LoRA to attention and feed-forward layers, and train the input embeddings and output projection; other pretrained parameters stay frozen. This adapts the generator to resource documentation and structured outputs. Each training instance has a query $Q^{(n)}$, context $\mathcal{C}(Q^{(n)})$, and reference $A^{(n)\star} = (M^{(n)\star}, B^{(n)\star})$. The context contains the reference resources to match constrained generation. We serialize $Y(A) = \langle M, \tau_{\text{sep}}, \text{Ser}(B), \tau_{\text{end}} \rangle$. For $B = \{t_1, \ldots, t_m\} \neq \emptyset$, $\text{Ser}(B) = \langle t_1, \ldots, t_m \rangle$ follows the supervised order; for an allowed empty bundle, it is $\langle \tau_\emptyset \rangle$. The markers τ_{sep}, $\tau_\emptyset$, and τ_{end} separate model and tools, mark an empty bundle, and end the configuration. Decoding order determines autoregressive conditioning, but the tool bundle is an unordered set. Resource identifiers are represented with existing subword tokens, without adding resource-specific tokens to the vocabulary.

Let $x^{(n)} = \text{Prompt}(I, Q^{(n)}, \mathcal{C}(Q^{(n)}))$, where I is a fixed instruction requesting only a configuration. The target $y^{(n)\star}$ is the tokenization of $Y(A^{(n)\star})$ followed by the model's end-of-sequence token. We minimize

$$\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{n=1}^{N} \sum_{i=1}^{|y^{(n)\star}|} \log p_\theta\Big(y_i^{(n)\star} \mid x^{(n)}, y_{<i}^{(n)\star}\Big). \tag{1}$$

3.3 Bundle-Level Preference Modeling

We train a retrieval-grounded critic comprising an encoder Enc_{ψ_e} and scalar head g_{ψ_s} to score complete agent compositions. The encoder consumes Q, A, and evidence $\Gamma(A, \mathcal{E}_r(Q))$ extracted for A's resources; the head produces $\widehat{U}_\psi(Q, A)$. The encoder starts from AgentEmbedding Chen et al. (2026), an EasyRec model tuned for agent-resource retrieval. We omit context dependence from $\widehat{U}_\psi(Q, A)$ for brevity.

Stage 1: Teacher-supervised preferences. Alternatives remove, add, or replace tools, or change the backbone with tools fixed. An LLM teacher assigns scores $r_i \in [0, 1]$; perturbation labels and rationales are excluded from critic input. We sample same-query pairs (i, j) with $r_i - r_j \geq 0.05$, separately capping comparisons that change the backbone, tools, or both. For minibatch $\mathcal{I}_1$, define $\widehat{U}_i = \widehat{U}_\psi(Q_i, A_i)$ and $w_{ij} = \sqrt{\max(r_i - r_j, 0.05)}$. The ranking loss is

$$\mathcal{L}_{\text{rank}} = -\frac{\sum_{(i,j)\in\mathcal{I}_1} w_{ij} \log \sigma(\widehat{U}_i - \widehat{U}_j)}{\sum_{(i,j)\in\mathcal{I}_1} w_{ij}}, \tag{2}$$

where σ is the logistic sigmoid. Let $\mathcal{L}_{\text{point}}$ be mean squared error between $\sigma(\widehat{U})$ and teacher scores over pair endpoints, counting repeated occurrences. We minimize $\mathcal{L}_{\text{critic}}^{(1)} = \mathcal{L}_{\text{rank}} + 0.1\mathcal{L}_{\text{point}}$.

Stage 2: Reference-based refinement. With the backbone fixed, we construct preference pairs from tool perturbations: removing extra tools when precision improves without reducing matches, restoring missing tools when recall improves without reducing precision, and replacing tools at fixed bundle size when precision improves without reducing recall. References are evaluated separately rather than pooling their components. Teacher-supervised comparisons are retained for backbone changes with fixed tool sets. For refinement minibatch $\mathcal{I}_2$, we minimize the following equation without the pointwise term.

$$\mathcal{L}_{\text{critic}}^{(2)} = \frac{1}{|\mathcal{I}_2|} \sum_{(i,j)\in\mathcal{I}_2} \log\Big(1 + \exp\Big[m - (\widehat{U}_i - \widehat{U}_j)\Big]\Big), \qquad m = 0.1, \tag{3}$$

3.4 Bundle-Aware Constrained Generation

Assuming unique catalog identifiers and at least one feasible configuration, search maintains at most W active hypotheses over retrieved resources. Empty bundles are allowed only when configured.

Configuration construction. Each hypothesis h has $M_h \in \mathcal{M}_r(Q)$, prefix $\rho_h = \langle M_h, \tau_{\text{sep}}, t_1, \ldots, t_\ell \rangle$, and tool set B_h. Legal actions select an unused tool from $\mathcal{T}_r(Q) \setminus B_h$ or terminate. Allowed empty bundles terminate with $\tau_\emptyset, \tau_{\text{end}}$; nonempty bundles use τ_{end}. Completion is forced at $|B_h| = L_{\max}$, the tool limit; termination is followed by the end-of-sequence token. Prefix tries enforce legal resource and control phrases, distinguishing complete identifiers from unfinished token prefixes. These constraints ensure catalog-valid decoding.

Action and path scoring. We use length-normalized generator log-likelihoods to guide action expansion and beam pruning. Given $x = \text{Prompt}(I, Q, \mathcal{C}(Q))$, each candidate action a with tokens $\pi(a) = \langle v_1^a, \ldots, v_{|\pi(a)|}^a \rangle$ is scored as

$$\bar{\ell}_\theta(a \mid x, \rho) = \frac{1}{|\pi(a)|} \sum_{j=1}^{|\pi(a)|} \log p_\theta\big(v_j^a \mid x, \rho, v_{<j}^a\big)\,, \tag{4}$$

where ρ is empty for backbone selection and ρ_h for subsequent actions. This score ranks candidate actions for expansion.

After expansion, we retain hypotheses according to their token-averaged path scores. For hypothesis tokens $\mathbf{y}(h)$, let Ω_h index the tokens of the action phrases selected along its path. The path score is

$$\bar{L}_\theta(h) = \frac{1}{|\Omega_h|} \sum_{i \in \Omega_h} \log p_\theta(y_i \mid x, y_{<i})\,. \tag{5}$$

Expansion and pruning. The beam starts with the top W retrieved LLM roots (or all available roots) and τ_{sep}. At each depth, each path expands through its top W legal actions; terminated and length-limited paths enter a completed pool. For unfinished extensions $\mathcal{V}_{\ell+1}$, Merge keeps the highest-scoring ordering per (M_h, B_h), giving $\mathcal{B}_{\ell+1} = \text{Top}_W(\text{Merge}(\mathcal{V}_{\ell+1}); \bar{L}_\theta)$. Here $\text{Top}_k(\mathcal{S}; f)$ selects up to k elements by f. Merging is approximate because tool order affects future distributions. Search ends when the active beam is empty; if $L_{\max} = 0$, roots immediately use the empty-bundle representation.

Final configuration reranking. We deduplicate completed paths by (M, B) into $\mathcal{F}(Q)$, retaining the best ordering and score $\bar{L}_\theta(Q, A)$. With the critic enabled,

$$S_{\text{final}}(Q, A) = \alpha_c z_Q(\widehat{U}_\psi(Q, A)) + \alpha_g z_Q(\bar{L}_\theta(Q, A)) - \lambda_{\text{len}}|B|, \tag{6}$$

where $\alpha_c, \alpha_g, \lambda_{\text{len}} \geq 0$ and $z_Q(f(A)) = (f(A) - \mu_f)/s_f$ uses the mean μ_f and standard deviation s_f over $\mathcal{F}(Q)$, returning zero if $s_f = 0$. The tool-count penalty favors compact bundles. Without the critic, $S_{\text{final}}(Q, A) = \bar{L}_\theta(Q, A) - \lambda_{\text{len}}|B|$. We return $\mathcal{H}_K(Q) = \text{Top}_K(\mathcal{F}(Q); S_{\text{final}}(Q, \cdot))$: up to K distinct configurations.

4 Experiments

4.1 Datasets

We derive data from **AgentSelect** Shi et al. (2026) and **SkillsBench** Li et al. (2026). AgentSelect aggregates over 40 sources: 111,179 queries, 107,721 agents, and 251,103 interactions, with a catalog of 231 LLM-only entries and 12,099 tools. From 14 selected SkillsBench with-skills submission groups, we collect 3,515 execution results and retain 1,300 successful trajectories with verifier reward 1.0. Sampling 1,000 trajectories with seed 42 yields 97 queries across 79 tasks and 615 query–agent pairs. The inventory contains 19 LLM identifiers, 155 skills, and 20 tool or action components. Table 1 summarizes the dataset splits. Due to the cost of API calls and inference, we use only randomly sampled 200 AgentSelect test queries. Appendix E provides further details on dataset construction.

Table 1: Dataset statistics.

Dataset	Split	Tasks	Queries	Pairs
AgentSelect	Train	–	5,022	12,180
	Val.	–	627	1,507
	Test	–	629	1,514
	Total	–	6,278	15,201
SkillsBench	Train	63	78	506
	Test	16	19	109
	Total	79	97	615

4.2 Implementation Details

Retrieval. On AgentSelect, we concatenate CF LLM, CF Historical Bundle, and Semantic Individual Tool Retrieval outputs as separate context sections, preserving bundle structure and shortening

component profiles to fit the input budget. Text2Bundle and subsequent generative methods, including ours, share this evidence. Component-Ret uses LLM and individual-tool outputs; Bundle-Ret uses LLM and historical-bundle outputs.

Llama Fine-Tuning. We fine-tune Llama-3-8B on AgentSelect for one epoch (425 updates) using LoRA rank 16, scaling factor 32, dropout 0.05, learning rate 10^{-4}, and effective batch size 32. We update LoRA parameters, token embeddings, and the output head, with a maximum sequence length of $8{,}192$ tokens. Loss covers only configuration output tokens; we retain the final checkpoint. SkillsBench uses task-disjoint training/test splits of 63/16 tasks and 78/19 queries. We further fine-tune the AgentSelect model for three epochs with learning rate 5×10^{-5} and effective batch size 8, updating only LoRA parameters while freezing embeddings and the output head. We retain the final checkpoint. All methods receive the same fixed inventory of 19 LLMs, 155 skills, and 20 tool or action components. No critic is trained or used. Section 5.2 follows a separate few-shot protocol.

Critic Training. We initialize the critic from AgentEmbedding Chen et al. (2026), using $1{,}000$ AgentSelect training queries and 150 held-out validation queries. Stage 1 uses four epochs of teacher-supervised pairwise ranking plus a pointwise loss weighted by 0.1; we select the checkpoint with the highest validation NDCG@10. Stage 2 refines with preference pairs for redundant additions, missing tools, and replacements, retaining teacher comparisons between LLMs. It uses a 0.1 pairwise margin without pointwise loss; we retain the second-epoch checkpoint after validation. Both stages update the encoder and scoring head with AdamW, using learning rates $(5\times10^{-6}, 10^{-4})$ and $(2 \times 10^{-6}, 2 \times 10^{-5})$ and effective batch sizes of 32 and 24 pairs, respectively.

Search Configuration. On AgentSelect, we use validation-selected adaptive beam bounds $(b_{\min}, b_{\max}) = (1, 2)$ and critic weight $\lambda = 0.5$. The critic reranks completed configurations without guiding search (Section 5.1). Appendix I reports the parameter grid: wider beams can improve component coverage and best-candidate recall without improving final ranking, while moderate critic weighting brings list quality closer to the selected candidates' oracle ordering.

4.3 Baselines

We compare three baseline families: (i) **retrieval-based composition**, pairing a retrieved LLM with independent tools (Component-Ret) or a historical bundle (Bundle-Ret); (ii) **bundle generation**, adapting sequential Text2Bundle Hu et al. (2025a) and discrete-diffusion DDBC-Adapt Tu et al. (2026); and (iii) **retrieval-augmented LLM generation**, using RAG-Llama, GPT-5.6-Luna/Terra/Sol, GPT-5.4/mini/nano, DeepSeek-V4-Pro, Qwen3.8-27B, and Kimi-K2.6. Text2Bundle and our retrieval-grounded generative variants receive the same retrieved evidence. DDBC-Adapt instead operates over its training-time component vocabulary and does not use the query-specific retrieval shortlist. Appendix F gives adaptation and output protocols.

4.4 Evaluation Metrics

Each method returns 10 configurations; metrics are query-macro-averaged on a 0–100 scale. Our primary metric, **Rank-Discounted Component Recall (RDCR@10)**, discounts component recall by rank, rewarding partial recovery near the top. **RDCP@10** similarly discounts component precision and penalizes components absent from the matched reference. **ToolR@1** measures required-tool coverage at rank 1, and **CompR@1** includes the reference LLM. **Tool-Hit@10** asks whether any candidate covers all required tools; **CR-Hit@1**, **CR-Hit@10**, and **CR-MRR@10** additionally require the reference LLM, with MRR rewarding earlier first hits. Appendix G defines all metrics and Table 2 report these results across all methods.

4.5 Overall Recommendation Performance

Historical bundles provide useful evidence, but require recomposition. On AgentSelect, Bundle-Ret achieves the highest ToolR@1 (83.08%) and CompR@1 (60.78%), indicating substantial overlap between historical bundles and the resources required by new tasks. However, directly reusing these bundles yields only 10.00% CR-Hit@10 and 29.79% RDCR@10, compared with 29.00% and 60.00% for ours. This gap suggests that historical configurations often contain useful components without matching the target configuration as a whole.

Table 2: Agent recommendation performance on two datasets. All scores are percentages; higher is better. The three highest distinct positive scores in each column and dataset are shaded.

	Tool Recovery		Agent Recovery				Ranked Quality	
Method	ToolR @1	Tool-Hit @10	CompR @1	CR-Hit @1	CR-Hit @10	CR-MRR @10	**RDCR @10**	**RDCP @10**
(a) AgentSelect								
Component-Ret	23.45	13.00	19.00	4.00	6.00	4.20	17.44	10.16
Bundle-Ret	**83.08**	**90.00**	**60.78**	9.00	10.00	9.50	29.79	29.35
Text2Bundle	40.10	35.00	31.20	3.00	5.00	3.64	30.52	18.21
DDBC-Adapt	46.10	31.00	35.87	4.00	16.00	8.13	31.35	27.92
RAG-Llama	47.00	44.00	36.20	4.00	5.00	4.20	13.42	14.59
RAG-GPT-5.6-Sol	71.05	60.00	51.90	5.00	24.00	11.46	50.47	48.25
RAG-GPT-5.6-Terra	74.52	64.00	55.05	9.00	29.00	14.44	51.85	**49.25**
RAG-GPT-5.6-Luna	68.85	60.00	49.90	6.00	23.00	10.80	47.02	45.76
RAG-GPT-5.4	71.22	62.00	52.48	8.00	26.00	12.59	49.22	47.11
RAG-GPT-5.4-mini	70.53	61.00	51.78	6.00	25.00	12.24	48.97	42.40
RAG-GPT-5.4-nano	54.12	49.00	40.17	7.00	14.00	9.01	33.65	29.59
RAG-DeepSeek-V4-Pro	66.52	54.00	48.67	6.00	20.00	9.59	47.27	39.95
RAG-Qwen3.8-27B	73.83	66.00	55.10	8.00	24.00	12.42	51.53	36.63
RAG-Kimi-K2.6	78.83	73.00	57.93	**10.00**	**33.00**	**15.85**	56.54	34.03
Ours	82.13	80.00	60.73	9.00	29.00	13.29	**60.00**	49.18
(b) SkillsBench								
RAG-GPT-5.6-Sol	64.81	21.05	52.17	5.26	5.26	5.26	53.59	50.28
RAG-GPT-5.6-Terra	63.60	10.53	51.65	0.00	0.00	0.00	52.24	44.99
RAG-GPT-5.6-Luna	28.77	0.00	25.76	0.00	0.00	0.00	26.68	43.19
RAG-GPT-5.4	62.59	36.84	50.29	0.00	21.05	4.61	53.90	38.28
RAG-GPT-5.4-mini	31.43	5.26	26.49	0.00	0.00	0.00	34.15	24.53
RAG-GPT-5.4-nano	28.70	5.26	23.75	0.00	0.00	0.00	24.54	25.05
RAG-DeepSeek-V4-Pro	40.54	5.26	34.55	0.00	5.26	0.66	36.78	36.07
RAG-Qwen3.8-27B	74.63	36.84	58.87	5.26	**26.32**	9.96	59.97	53.37
RAG-Kimi-K2.6	36.45	0.00	35.04	0.00	0.00	0.00	33.57	39.40
Ours	**76.74**	**57.89**	**64.84**	**10.53**	**26.32**	**18.42**	**68.11**	**58.30**

LLMs select relevant resources but struggle with reliable composition. Given the same input evidence within each dataset, RAG-LLMs exhibit meaningful resource-selection ability, as reflected in ToolR@1, Tool-Hit@10, and CompR@1. However, their generally low CR-Hit@1 and CR-Hit@10 indicate difficulty jointly recovering all required tools and the reference backbone. This ability is also inconsistent across domains: on SkillsBench, five of the nine RAG baselines achieve no complete-recovery hits, including Kimi-K2.6, whose CR-Hit@10 drops from 33.00% on AgentSelect to zero. Qwen3.8-27B remains competitive on both datasets, but the broader results suggest that resource selection alone does not ensure reliable task-specific composition.

Our method improves ranked recovery, especially on SkillsBench. On AgentSelect, ours achieves the highest RDCR@10 (60.00%), exceeding the strongest RAG baseline by 3.46 percentage points while maintaining near-best RDCP@10 (49.18% versus 49.25%). The advantage becomes more pronounced on SkillsBench, where ours exceeds Qwen3.8-27B by 8.14 and 4.93 points in RDCR@10 and RDCP@10, respectively. Notably, both methods achieve 26.32% CR-Hit@10, but ours increases CR-Hit@1 from 5.26% to 10.53% and CR-MRR@10 from 9.96% to 18.42%. Thus, even with the same overall complete-hit rate, our method improves ranked component recovery and places complete-recovery hits earlier in aggregate.

5 Further Analysis

5.1 Structural Ablation of Search and Critic Placement

Figure 2 compares five variants: **A (Free)**, unconstrained generation; **B (Greedy)**, constrained single-path decoding; **C (Beam)**, constrained beam search with generator scores; **D (Critic)**, critic

reranking of completed beam candidates; and **E (Beam+Critic)**, which also uses critic scores during beam expansion and pruning. D is our main configuration (Table 2); other settings are fixed, with adaptive beam bounds $(1, 2)$ and critic weight 0.5 where applicable. SUCR@10 measures union component coverage, OGR@10 the best candidate's component recall, and Oracle-RDCR@10 versus RDCR@10 oracle versus actual ranked recall. RDCP@10 is the precision counterpart. All use returned Top-10 lists; missing ranks contribute zero to discounted scores.

Constrained greedy decoding barely changes ranking but improves validity: about 7% of A outputs have formatting errors and another 3% contain non-existent components. OGR@10 closely matches SUCR@10, suggesting most recovered components lie in the best candidate. Beam search improves coverage, but its RDCR@10 remains below Oracle-RDCR@10, indicating imperfect ordering. Final critic reranking raises RDCR@10 from 49.9% (C) to 60.0% (D) and slightly improves RDCP@10; critic guidance during search (E) instead reduces ranking quality. Thus, the critic is more effective for final reranking than during beam expansion.

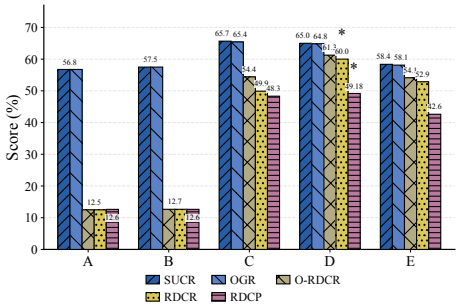


Figure 2: Structural ablation results. Asterisks mark the highest RDCR and RDCP, respectively.

5.2 Cross Domain Generalization

Controlled transfer setting. We compare our AgentSelect-trained generator with original Meta-Llama-3-8B under zero-shot transfer and few-shot adaptation. Adaptation uses 4 queries from three tasks. Both models receive the same fixed inventory context, LoRA modules, optimizer settings, and 40-update budget, without dynamic retrieval or a critic. A separate validation set of 11 queries from 10 tasks guides checkpoint selection and early stopping. We report query-averaged metrics on a task-disjoint test set of 70 queries from 54 tasks, using seed 42.

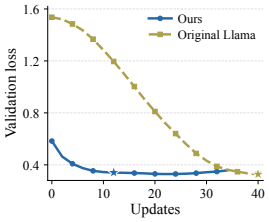


Figure 3: Validation loss.

Table 3: SkillsBench transfer.

Setting	Init.	ToolR @1	Tool-Hit @10	CompR @1	CR-Hit @1	CR-Hit @10	CR-MRR @10	RDCR @10
Zero-shot	Llama	0.24	0.00	2.72	0.00	0.00	0.00	5.04
	Ours	**12.37**	**7.14**	**16.07**	0.00	**2.86**	**0.44**	**18.54**
Few-shot	Llama	56.26	34.29	47.56	5.71	5.71	5.71	39.70
	Ours	**62.34**	**41.43**	**50.87**	**7.14**	**10.00**	**8.10**	**44.85**

Transfer quality and adaptation speed. Our pretrained generator raises zero-shot ToolR@1 from 0.24% to 12.37% and RDCR@10 from 5.04% to 18.54% relative to original Llama, although recovery remains limited under domain shift. After few-shot adaptation, it leads on all seven metrics (Table 3) and stops at update 12; original Llama reaches comparable validation loss at the full 40-update budget (Figure 3). These results suggest that source-domain recommendation training facilitates adaptation to the SkillsBench inventory.

5.3 End-to-End Execution Evaluation

Experimental setup. We investigate whether the predicted configuration rankings reflect end-to-end agent performance. The execution environment contains 14 LLMs accessed through SiliconFlow, 66 APIs, and 16 locally deployed skills from the complete resource space. Eleven APIs have executable adapters; for the remaining 55, we follow the simulation protocol of MirrorAPI Guo et al. (2025), generating responses with web-search-enabled GPT-5.6-Luna. We sample 30 evaluation queries (25 from the AgentSelect test set and 5 from the SkillsBench test set). For each, we rank configurations retained by search and those pruned during generation. Retained candidates often contain the resources needed for the task and can be difficult to distinguish by execution quality; including pruned candidates broadens the quality range. We select four representative configurations

from each ranked list: TOP, MIDDLE, P90, and LAST, yielding 120 end-to-end executions. Each configuration produces a complete interaction trajectory and final output, both assessed by the external judge GPT-5.6-Tera. Figure 4 shows the score distributions, Table 8 reports agreement between predicted and execution-quality rankings, and Appendix J provides further details.

Analysis. The end-to-end results show a substantial alignment between our predicted configuration ranking and actual execution quality. Across all 30 queries, the ranking achieves a query-mean Spearman correlation of $\rho = 0.634$ (95% CI $[0.452, 0.784]$) and 76.4% pairwise concordance ($[68.3, 83.3]$). The Top configuration also outperforms the Last by 5.17 points on the 0–10 scale ($[4.03, 6.17]$; Table 8), indicating that recommendation scores capture differences that translate into downstream agent performance. The association is stronger on AgentSelect, with $\rho = 0.714$, 80.0% concordance, and a 5.70-point Top–Last gap. SkillsBench shows weaker agreement ($\rho = 0.230$, 58.3%, and 2.50 points), although this estimate is based on only five queries and has substantially wider confidence intervals. Overall, the ranking separates configurations with large quality differences more reliably than those with similar executable capabilities.

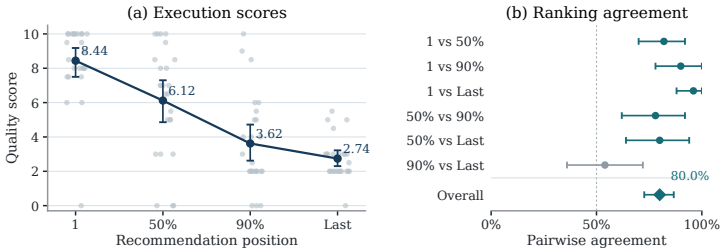


Figure 4: End-to-end execution results. (a) Mean execution quality by recommendation position. (b) Pairwise agreement between predicted ranking and execution quality. Error bars denote 95% bootstrap confidence intervals.

6 CASE STUDY

We examine a query from the end-to-end evaluation to illustrate how tool value depends on its companion resources. The user requests a domain-similarity check, NS records, and a nameserver IP. We fix DeepSeek-V4-Pro and the similarity tool S, then vary DNS Lookup D and NS Lookup N. **Tool contributions depend on the bundle.** The raw critic scores for S, $S{+}D$, $S{+}N$, and $S{+}D{+}N$ are -5.56, 0.11, 1.42, and 3.14 (Figure 5). Adding D to S yields a 5.67 gain, but only 1.72 when N is already present, indicating diminishing returns from overlapping lookup capabilities. An additive extrapolation would predict 7.09 for the full bundle rather than the observed 3.14, showing a clear non-additive interaction. **Final ranking agrees with execution quality.** Final reranking also incorporates generator likelihood. The corresponding scores are 0.27, 0.71, 1.04, and 0.89, matching the ordering of end-to-end judge scores (6.5, 9, 10, and 9.5): $S{+}N > S{+}D{+}N > S{+}D > S$. Notably, the three-tool agent does not call N, while $S{+}N$ achieves the best execution with fewer tools. This case illustrates both context-dependent tool value and alignment between final ranking and observed execution quality.

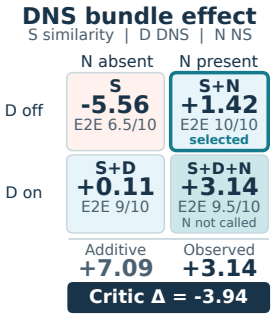


Figure 5: Bundle effects.

7 CONCLUSION AND FUTURE WORK

We formulate **AgentComposition**, addressing a missing problem in agent resource discovery: composing models, tools, and skills into task-specific configurations. We propose a retrieval-grounded, bundle-aware framework and validate its effectiveness and transferability on AgentSelect and SkillsBench. Results also expose limitations of standalone LLM generation for reliable resource composition. We believe AgentComposition can support more personalized, adaptive, and continually evolving agents. A key limitation is that critic signals are mainly used for post-hoc reranking. Future work could integrate execution feedback directly into generation, including through reinforcement learning, to improve resource composition as tasks and resource ecosystems evolve.

REFERENCES

Anthropic. Introducing the model context protocol, 2024. URL https://www.anthropic.com/news/model-context-protocol.

Anthropic. Effective harnesses for long-running agents. https://www.anthropic.com/engineering/effective-harnesses-for-long-running-agents, November 2025a.

Anthropic. Equipping agents for the real world with agent skills, 2025b. URL https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Jianxin Chang, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. Bundle recommendation with graph convolutional networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pp. 1673–1676, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401198. URL https://doi.org/10.1145/3397271.3401198.

Tingwei Chen, Yunxiao Shi, Zhengdong Chu, Qingsong Wen, and Min Xu. Adapting embedding models for agent capability retrieval. *arXiv preprint arXiv:2607.17347*, 2026. Accepted for oral presentation at the AgentSearch Workshop, SIGIR 2026.

Zihao Cheng, Hongru Wang, Zeming Liu, Yuhang Guo, Yuanfang Guo, Yunhong Wang, and Haifeng Wang. ToolSpectrum: Towards personalized tool utilization for large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20679–20699, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1063. URL https://aclanthology.org/2025.findings-acl.1063/.

Yu Du, Fangyun Wei, and Hongyang Zhang. AnyTool: Self-reflective, hierarchical agents for large-scale API calls. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11812–11829. PMLR, 2024. URL https://proceedings.mlr.press/v235/du24h.html.

Zhicheng Guo, Sijie Cheng, Yuchen Niu, Hao Wang, Sicheng Zhou, Wenbing Huang, and Yang Liu. StableToolBench-MirrorAPI: Modeling tool environments as mirrors of 7,000+ real-world APIs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5247–5270, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.273. URL https://aclanthology.org/2025.findings-acl.273/.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=BHXsb69bSx.

Yupu Hao, Pengfei Cao, Zhuoran Jin, Huanxuan Liao, Yubo Chen, Kang Liu, and Jun Zhao. Evaluating personalized tool-augmented LLMs from the perspectives of personalization and proactivity. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar

(eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21897–21935, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1064. URL https://aclanthology.org/2025.acl-long.1064/.

Juntong Hu, Shixuan Zhu, Chuan Cui, Qi Shen, Yu Ji, and Zhihua Wei. Text2bundle: Towards personalized query-based bundle generation. *ACM Trans. Recomm. Syst.*, 4(2), November 2025a. doi: 10.1145/3706125. URL https://doi.org/10.1145/3706125.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://arxiv.org/abs/2408.08435.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, Chujun Tao, Binxu Li, Xuandong Zhao, Hejia Geng, Xiaojun Wu, Junwei Zhou, Xiaokun Chen, Hanwen Xing, Yubo Li, Qunhong Zeng, Di Wang, Yuanli Wang, Roey Ben Chaim, Penghao Jiang, Haotian Shen, Luyang Kong, Xinyi Liu, Runhui Wang, Xuanqing Liu, Jiachen Li, Xin Lan, Yueqian Lin, Wengao Ye, Junwei He, Songlin Li, Yue Zhang, Yipeng Gao, Yijiang Li, Ze Ma, Liqiang Jing, Tianyu Wang, Kaixin Li, Yiqi Xue, Haoran Lyu, Yizhuo He, Yuchen Tian, Shutong Wu, Bowei Wang, Yixuan Gao, Bo Chen, Litong Liu, Sikai Cheng, Jiajun Bao, Shuaicheng Tong, Shuwen Xu, Terry Yue Zhuo, Tinghan Ye, Qi Qi, Miao Li, Longtai Liao, Zelin Tan, Chang Shi, Xilin Tang, Srinath Tankasala, Boqin Yuan, Yaoyao Qian, Jianhong Tu, Chenguang Wang, Yizhou Sun, Wei Wang, Aaron Taylor, Ziyue Yang, Changkun Guan, Zhikang Dong, Xinyu Zhang, Steven Dillmann, Han chung Lee, and Dawn Song. Skillsbench: Benchmarking how well agent skills work across diverse tasks, 2026. URL https://arxiv.org/abs/2602.12670.

Marianne Menglin Liu, Daniel Garcia, Fjona Parllaku, Vikas Upadhyay, Syed Fahad Allam Shah, and Dan Roth. ToolScope: Enhancing LLM agent tool use through tool merging and context-aware filtering. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34095–34119, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1573. URL https://aclanthology.org/2026.acl-long.1573/.

Tenglong Liu, Jianxiong Li, Yinan Zheng, Haoyi Niu, Yixing Lan, Xin Xu, and Xianyuan Zhan. Skill expansion and composition in parameter space. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=GLWf2fq0bX.

Yunshan Ma, Yingzhi He, An Zhang, Xiang Wang, and Tat-Seng Chua. Crosscbr: Cross-view contrastive learning for bundle recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pp. 1233–1241, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539229. URL https://doi.org/10.1145/3534678.3539229.

Microsoft. Agent harness. https://learn.microsoft.com/agent-framework/agents/harness, 2026. Microsoft Agent Framework.

Suhong Moon, Siddharth Jha, Lutfi Eren Erdogan, Sehoon Kim, Woosang Lim, Kurt Keutzer, and Amir Gholami. Efficient and scalable estimation of tool representations in vector space, 2024. URL https://arxiv.org/abs/2409.02141.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

OpenAI. The next evolution of the agents sdk. https://openai.com/index/the-next-evolution-of-the-agents-sdk/, April 2026.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=dHng2O0Jjr.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=dHng2O0Jjr.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. Towards completeness-oriented tool retrieval for large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 1930–1940. ACM, 2024. doi: 10.1145/3627673.3679847. URL https://doi.org/10.1145/3627673.3679847.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

Xubin Ren and Chao Huang. EasyRec: Simple yet effective language models for recommendation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 17717–17732, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.894. URL https://aclanthology.org/2025.emnlp-main.894/.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Yacmpz84TH.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic LLM agent search in modular design space. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mPdmDYIQ7f.

Yunxiao Shi, Wujiang Xu, Tingwei Chen, Haoning Shang, Ling Yang, Yunfeng Wan, Zhuo Cao, Xing Zi, Dimitris N. Metaxas, and Min Xu. Agentselect: Benchmark for narrative query-to-agent recommendation. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=4M5Kj2UqaM.

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, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.1258. URL https://aclanthology.org/2025.findings-acl.1258/.

Meng Sun, Lin Li, Ming Li, Xiaohui Tao, Dong Zhang, Qing Xie, Peipei Wang, and Jimmy Xiangji Huang. A survey on bundle recommendation: Methods, applications, and challenges. *ACM Comput. Surv.*, 58(11), May 2026. ISSN 0360-0300. doi: 10.1145/3802820. URL https://doi.org/10.1145/3802820.

Zhu Sun, Kaidong Feng, Jie Yang, Xinghua Qu, Hui Fang, Yew-Soon Ong, and Wenyuan Liu. Adaptive in-context learning with large language models for bundle generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pp. 966–976, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657808. URL https://doi.org/10.1145/3626772.3657808.

Teng Tu, Ai Li, Yunshan Ma, Shuo Xu, Xiaohao Liu, Haokai Ma, Liang Pang, and Tat-Seng Chua. Discrete diffusion for bundle construction. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dKyhgfe50H.

Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. Toolgen: Unified tool retrieval and calling via generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=XLMAMmowdY.

Bin Wu, Arastun Mammadli, Xiaoyu Zhang, and Emine Yilmaz. Agentsearchbench: A benchmark for ai agent search in the wild, 2026a. URL https://arxiv.org/abs/2604.22436.

Bin Wu, Edgar Meij, and Emine Yilmaz. Beyond static toolsets: Self-evolving LLM tool agents via continual documentation adaptation. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 21519–21539, San Diego, California, United States, July 2026b. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1082. URL https://aclanthology.org/2026.findings-acl.1082/.

Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward. In *First Workshop on Agent Skills*, 2026. URL https://openreview.net/forum?id=Er0p92BsmW.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi R. Fung, Hao Peng, and Heng Ji. CRAFT: Customizing LLMs by creating and retrieving from specialized toolsets. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=G0vdDSt9XM.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. SkillWeaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025. URL https://arxiv.org/abs/2504.07079.

A AI USAGE STATEMENT

In this work, we used generative AI tools to assist with translation, dataset cleaning and reformatting. Additionally, we used these tools to improve the readability of the manuscript, summarize and analyze existing literature, and identify relevant references. We reviewed all AI-assisted work, including checking the references suggested by these tools. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

B RETRIEVAL DESIGN

The AgentSelect configuration in Section 4.2 combines CF LLM retrieval, CF historical-bundle retrieval, and semantic individual-tool retrieval. Semantic LLM retrieval and query rewriting are optional alternatives and are disabled in this configuration. SkillsBench instead uses the fixed inventory described in Section 4.2.

CF retrieval. Let d_M and d_G be profiles of backbone $M \in \mathcal{M}$ and historical tool bundle $G \in \mathcal{G} \subseteq 2^{\mathcal{T}}$. Two trained two-tower retrievers use TF–IDF features and query-conditioned supervision from AgentSelect Shi et al. (2026):

$$s^{\mathrm{sp}}_{\mathrm{LLM}}(Q, M) = \left\langle g^{\mathrm{LLM}}_{\phi_Q}(\mathrm{TFIDF}(Q)), g^{\mathrm{LLM}}_{\phi_M}(\mathrm{TFIDF}(d_M)) \right\rangle,$$

$$s^{\mathrm{sp}}_{\mathrm{bun}}(Q, G) = \left\langle g^{\mathrm{bun}}_{\phi_Q}(\mathrm{TFIDF}(Q)), g^{\mathrm{bun}}_{\phi_G}(\mathrm{TFIDF}(d_G)) \right\rangle.$$

Historical bundles remain intact as retrieval evidence; their tools are also merged and deduplicated into the candidate tool inventory.

Semantic retrieval. Dense encoders match query text to component profiles. We use tuned EasyRec models Ren & Huang (2025) trained with AgentSelect supervision; the representations are further studied in AgentEmbedding Chen et al. (2026). For $u = M$ or t, with corresponding component encoder e_u, the score is $\langle e_Q(Q), e_u(d_u)\rangle$.

Context assembly. The selected rankings provide $\mathcal{M}_r(Q)$ and $\mathcal{T}_r(Q)$. When both LLM branches are enabled, their rankings are normalized, fused, and deduplicated. Evidence $\mathcal{E}_r(Q)$ preserves component profiles, normalized scores, provenance, and retrieved historical bundles. On AgentSelect, the three enabled retrieval outputs occupy separate context sections, with shortened profiles to fit the input budget.

C Constrained Generation and Bundle-Critic Reranking

Algorithm 1 implements the search and final reranking in Section 3.4. The critic scores completed configurations only; generator scores govern beam pruning.

Algorithm 1 Constrained Generation with Optional Bundle-Critic Reranking

Require: Query Q, context $\mathcal{C}(Q)$, generator p_θ, optional critic $\widehat{U}_\psi$, beam width $W \geq 1$, tool budget $L_{\max}$, output size $K \geq 1$
Ensure: Up to K distinct ranked configurations $\mathcal{H}_K(Q)$

Initialize up to W LLM roots by Eq. (4); append τ_{sep}
$\mathcal{F}_{\text{paths}} \leftarrow \emptyset$
If $L_{\max} = 0$, complete roots with the empty-bundle representation and move them to $\mathcal{F}_{\text{paths}}$
while the active beam is nonempty **do**
    Enumerate legal unused-tool and termination actions per path
    Retain up to W actions per path by Eq. (4), applying the optional no-tool proposal penalty
    Extend paths and update Eq. (5)
    Move terminated or budget-complete paths to $\mathcal{F}_{\text{paths}}$
    Merge unfinished paths by (M, B), keeping the best ordering; retain up to W by $\bar{L}_\theta$
end while
Deduplicate $\mathcal{F}_{\text{paths}}$ by (M, B) into $\mathcal{F}(Q)$, retaining the best ordering and score
if a domain-appropriate critic is enabled **then**
    Compute $\widehat{U}_\psi(Q, A)$ for each $A \in \mathcal{F}(Q)$
    Compute S_{final} by Eq. (6)
else
    $S_{\text{final}}(Q, A) \leftarrow \bar{L}_\theta(Q, A) - \lambda_{\text{len}}|B|$
end if
return $\text{Top}_K(\mathcal{F}(Q); S_{\text{final}}(Q, \cdot))$

D Provenance-Aware Configuration Explanation

After ranking is fixed, templates report selected component profiles, retrieval sources and scores, and overlapping historical bundles from $\mathcal{E}_r(Q)$. Stored values support statements such as "Tool t was supplied by retriever r with score s." This interface does not generate free-form rationales or alter selection.

With a critic, the interface may additionally report the bundle score and leave-one-tool-out differences:

$$\Delta_t(Q, A) = \widehat{U}_\psi(Q, A) - \widehat{U}_\psi(Q, (M, B \setminus \{t\})) .$$

A positive difference means that removing t lowers the critic score. This is a model-based diagnostic, not a causal effect or evidence of execution improvement. Critic diagnostics are separate from retrieval provenance and omitted when no critic is used. We neither train the generator to produce these statements nor evaluate explanation quality.

E CONSTRUCTION OF THE SKILLSBENCH RECOMMENDATION VIEW

Trace selection and extraction. We use archived SkillsBench Li et al. (2026) executions from `benchflow/skillsbench-leaderboard`. Eligible traces have verifier reward 1.0, a readable initial query, completed tool events, and no reported execution error or partial trajectory. Oversized logs are excluded under a fixed resource cap before sampling. We sample $1{,}000$ trajectories with seed 42 from the selected with-skills submission groups, as described in Section 4.1. Source revisions, paths, and content hashes are retained for traceability.

The result record supplies the backbone identity. Skills require explicit activation evidence or identifiable paths in completed read/execute events; mentions in reasoning, descriptions, or available-skill lists are insufficient. Native tools retain their observed names, while harness action categories are recorded separately. The unique observed skills and tools form the component set. Unresolvable skill references and opaque model identifiers are not assigned guessed names.

Queries and reference configurations. We group traces by source task and normalized initial query, merge duplicate model–component-set profiles, and retain up to ten distinct successful configurations per query without padding. Equal verifier rewards do not provide a preference ranking, so source hashes determine storage order. Evaluation matches each candidate against any individual reference (Appendix G). Task-level partitions keep all traces and query variants of a task together. The main recommendation experiment uses the train/test split in Table 1; Section 5.2 uses a separate few-shot protocol.

Inventory and label scope. The shared inventory consists of observed components and known model identifiers. Skill descriptions are linked to public documents from a pinned snapshot, which need not match the files used in each archived execution; unmatched descriptions remain absent. Methods receive query text and catalog descriptions, excluding reference configurations, rewards, trace paths, and extraction evidence. Evaluation is confined to this observed inventory. Observed component sets need not be minimal, and unobserved components may still have been available during the archived execution. Matching a successful profile does not establish that a newly assembled configuration will reproduce its execution. Generic harness actions also contribute to mixed-component overlap.

F BASELINE ADAPTATIONS AND OUTPUT PROTOCOLS

F.1 RETRIEVAL-BASED COMPOSITION

Component-Ret. We independently retrieve LLMs via CF and tools via semantic retrieval, without query rewriting. The top-10 LLMs are paired with the same six highest-ranked tools, forming ten configurations ranked by LLM CF score.

Bundle-Ret. Separate CF models retrieve one top-ranked LLM and ten historical tool bundles. Each intact bundle is paired with that LLM and ranked by bundle CF score.

F.2 TEXT2BUNDLE

We adapt Text2Bundle Hu et al. (2025a) by mapping items to tools and using the query as the request. Without persistent user identities or histories, we omit user-ID embeddings and personalization rewards. Queries and tool descriptions use AgentEmbedding Chen et al. (2026), an EasyRec Ren & Huang (2025) model tuned for agent-resource retrieval. We retain the Transformer state encoder, action-relation modeling, and learned termination action.

Pairwise bundle-completion pretraining precedes Double-DQN training with experience replay. The action pool deduplicates tools from retrieved historical bundles and semantic retrieval. Missing target tools may be added during training only; validation and inference use retrieved candidates. Before policy training, records matching the fixed evaluation set by query text or ID are excluded.

The CF-selected LLM stays fixed while beam search extends the single-output policy to Top-K tool sets. Paths are ranked by negative cumulative action regret: each increment is the chosen action's Q-value minus the highest valid-action Q-value. The greedy path scores zero and is retained first.

Duplicate tool sets retain their best path; a size cap forces termination. This ranked-output procedure is part of our adaptation.

F.3 DDBC-ADAPT

We adapt DDBC Tu et al. (2026), which uses masked discrete diffusion and residual quantization. Unlike our retrieval-grounded generative methods, DDBC-Adapt does not use the query-specific retrieval shortlist. Instead, it operates over a component vocabulary constructed from the training split. Frozen MPNet embeddings of LLM and tool descriptions, PCA, and residual k-means replace its representation learner and quantizer. An extra code distinguishes components with identical quantized codes; both the vocabulary and codebooks are constructed from training data only.

A query-conditioned denoiser jointly generates one LLM and a tool set with typed boundaries. Training masks component codes, preserves structural tokens, and randomizes tool order, without implying exact permutation invariance. Decoding retains absorbing masks, enforces code compatibility, and excludes duplicate tools. It can therefore generate unseen combinations of components, but only from components represented in its training-time vocabulary. Reference components absent from this vocabulary are unrecoverable by DDBC-Adapt, independently of query relevance.

Task-trained semantic adapters estimate query–component relevance over the training vocabulary rather than defining a retrieval shortlist. Relevance probabilities are summed over legal components sharing each code value, and the resulting log masses augment denoiser logits. Distinct valid configurations are ranked by LLM and summed tool relevance, weighted by the predicted tool-count distribution. This coverage score is not an exact configuration probability. Reported results use semantic guidance without an additional configuration scorer.

G EVALUATION METRIC DEFINITIONS

Notation. For query q, let the nonempty reference set be $\mathcal{G}_q = \{G_{q,j}\}_{j=1}^{J_q}$, where $G_{q,j} = (M^*_{q,j}, B^*_{q,j})$. SkillsBench component sets include recorded skills and tools/actions. At rank r, the prediction is $A_{q,r} = (M_{q,r}, B_{q,r})$. We evaluate at $K = 10$, retaining $n_q \le K$ candidates; missing ranks receive zero credit.

Component recall and precision. Against reference j, define

$$t_{q,r,j} = \frac{|B_{q,r} \cap B^*_{q,j}|}{|B^*_{q,j}|}, \tag{7}$$

$$c_{q,r,j} = \frac{\mathbb{I}[M_{q,r} = M^*_{q,j}] + |B_{q,r} \cap B^*_{q,j}|}{1 + |B^*_{q,j}|}, \tag{8}$$

$$p_{q,r,j} = \frac{\mathbb{I}[M_{q,r} = M^*_{q,j}] + |B_{q,r} \cap B^*_{q,j}|}{1 + |B_{q,r}|}. \tag{9}$$

Set $t_{q,r,j} = 1$ when $B^*_{q,j} = \varnothing$. Component recall and precision count the backbone as one component. For multiple references,

$$t_{q,r} = \max_j t_{q,r,j}, \qquad c_{q,r} = \max_j c_{q,r,j}, \qquad p_{q,r} = \max_j p_{q,r,j}. \tag{10}$$

Each measure matches a single reference independently, so best matches may differ; components are never pooled across references. For $r > n_q$, $t_{q,r} = c_{q,r} = p_{q,r} = 0$.

Top-rank recall and complete recovery. ToolR@1 is $t_{q,1}$ and CompR@1 is $c_{q,1}$. Tool-Hit@10 requires coverage of every tool in at least one reference, regardless of the backbone:

$$\text{Tool-Hit@10}(q) = \mathbb{I}\left[\exists\, 1 \le r \le \min(n_q, 10),\ 1 \le j \le J_q : B^*_{q,j} \subseteq B_{q,r}\right]. \tag{11}$$

A complete-recall hit additionally requires the same reference's backbone:

$$h_{q,r} = \mathbb{I}\left[\exists\, 1 \le j \le J_q : M_{q,r} = M^*_{q,j} \ \wedge\ B^*_{q,j} \subseteq B_{q,r}\right], \tag{12}$$

with $h_{q,r} = 0$ at missing ranks. For $k \in \{1, 10\}$,

$$\text{CR-Hit@}k(q) = \max_{1 \le r \le k} h_{q,r}. \tag{13}$$

Let r_q^* be the first complete-recall hit, or ∞ if none exists. Then

$$\text{CR-MRR@10}(q) = \begin{cases} 1/r_q^*, & r_q^* \le 10, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

These hit metrics allow additional tools; they measure complete recall, rather than exact set equality.

Rank-discounted component scores. With $w_r = 1/\log_2(r+1)$ and $D_K = \sum_{r=1}^{K} w_r$, define

$$\text{RDCR@}K(q) = \frac{\sum_{r=1}^{K} w_r c_{q,r}}{D_K}, \tag{15}$$

$$\text{RDCP@}K(q) = \frac{\sum_{r=1}^{K} w_r p_{q,r}}{D_K}. \tag{16}$$

Both scores lie in $[0, 1]$, favor earlier ranks, and use a fixed denominator, so missing candidates contribute zero. They do not explicitly reward diversity. Additional unmatched tools do not reduce RDCR, but they reduce RDCP through its predicted-set denominator. RDCP can favor incomplete configurations and is therefore interpreted together with RDCR.

Aggregation and scope. All scores are query-macro-averaged on a 0–100 scale:

$$\overline{s} = \frac{100}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} s(q). \tag{17}$$

Each query has equal weight regardless of its number of references. The recall and complete-recovery metrics allow additional tools without penalty, whereas RDCP penalizes components absent from the matched reference. None of these metrics measures execution success or the downstream effects of additional tools.

H Retrieval Analysis

Retrieval mechanisms and the enabled AgentSelect branches are specified in Appendix B. The following studies measure candidate coverage separately from the recommendation results in Table 2.

Evaluation protocol. Tool retrieval uses 1,000 unique AgentSelect Part II queries sampled with seed 42. Macro recall averages the fraction of reference tools retrieved per query; all-covered rate requires their complete coverage. For historical bundles, these measures use the merged tool set. Exact hit requires one retrieved bundle to equal the reference tool set, while complete hit permits additional tools. The cutoff K counts tools or bundles, respectively; Avg. Tools counts unique retrieved tools.

LLM retrieval uses all 23,073 unique Part I queries, each with multiple ranked reference LLMs. Precision, recall, MRR, and MAP use binary relevance; hit requires any relevant LLM and all-covered requires every reference LLM. For nDCG, reference rank r receives gain $1/\log_2(r+1)$.

CF retrieval coverage. At $K = 5$, historical bundles yield 5.52 unique tools on average and 88.09% macro recall (Table 4). Merging adds only 0.50 percentage points of complete coverage beyond a single bundle's complete hit. Increasing K to 20 raises merged all-covered rate to 90.00%, with 17.21 tools on average.

Table 4: CF-based tool-bundle retrieval on 1,000 Part II queries. All metrics except Avg. Tools are percentages. Avg. Tools is the number of unique tools after merging and deduplication.

K	Exact Hit	Complete Hit	Macro Recall	All Covered	Avg. Tools
5	81.20	85.10	88.09	85.60	5.52
10	85.00	87.60	89.49	88.20	9.34
15	86.70	88.50	90.10	89.20	13.19
20	88.20	89.30	90.69	90.00	17.21

At $K = 10$, CF LLM retrieval has 97.95% hit rate and 0.8565 MRR (Table 5). Increasing K to 50 mainly recovers additional relevant models: recall rises from 57.07% to 92.34%, while hit rate rises to 99.91%.

Table 5: CF-based LLM retrieval on 23,073 Part I queries. P and R are macro precision and recall; All is the all-covered rate. P, R, Hit, and All are percentages.

K	P	R	Hit	All	MRR	MAP	nDCG
5	69.43	35.43	94.75	0.00	0.8521	0.6387	0.6327
10	56.21	57.07	97.95	3.61	0.8565	0.4877	0.6244
15	45.30	68.81	98.99	13.15	0.8574	0.5519	0.6744
20	37.57	76.01	99.45	23.00	0.8576	0.5816	0.7011
50	18.29	92.34	99.91	64.63	0.8578	0.6250	0.7512

Individual-tool coverage. In the individual-tool cutoff study, increasing K from 5 to 50 raises macro recall from 39.95% to 65.46% and all-covered rate from 21.10% to 50.60% (Table 6).

Table 6: Individual-tool retrieval on 1,000 Part II queries. Both metrics are percentages.

K	Macro Recall	All Covered
5	39.95	21.10
10	48.30	30.20
15	51.93	34.00
20	55.74	38.00
50	65.46	50.60

Query rewriting is an optional alternative and is not used in the main recommendation configuration.

I Sensitivity to Beam Bounds and Critic Weight

Validation protocol. Using the checkpoint in Table 2, we test adaptive beam bounds $\{(1,2),(2,4),(3,5),(4,7),(6,10)\}$ and critic weights $\lambda \in \{0, 0.5, 1.0, 1.5, 2.0\}$, holding other settings fixed. The critic changes final Top-10 membership and ordering after search; it does not guide generation. At $\lambda = 0$, ranking uses generator scores alone. The main setting, $(1,2)$ and $\lambda = 0.5$, is validation-selected (Section 4.2). These validation scores are separate from results on the 200 AgentSelect test queries in Table 2.

Diagnostic upper bounds. For a reference component set C_G and candidate component sets C_r, union coverage is $\mathrm{SUCR@}K = |(\bigcup_{r=1}^{K} C_r) \cap C_G|/|C_G|$. With multiple references, union coverage is matched against each intact reference and maximized; reference components are not pooled. Using candidate component recalls R_r from Appendix G, define

$$\mathrm{OGR@}K = \max_{r \le K} R_r,$$

$$\text{Oracle-RDCR@}K = \frac{\sum_{r=1}^{K} \widetilde{R}_{(r)} / \log_2(r+1)}{\sum_{r=1}^{K} 1/\log_2(r+1)},$$

where $\widetilde{R}_{(r)}$ sorts recalls in descending order. SUCR measures union component coverage, OGR best-candidate recall, and Oracle-RDCR the best ordering of the selected list. All use the final Top-K, rather than the full search pool, with missing ranks scored zero and query-macro averaging.

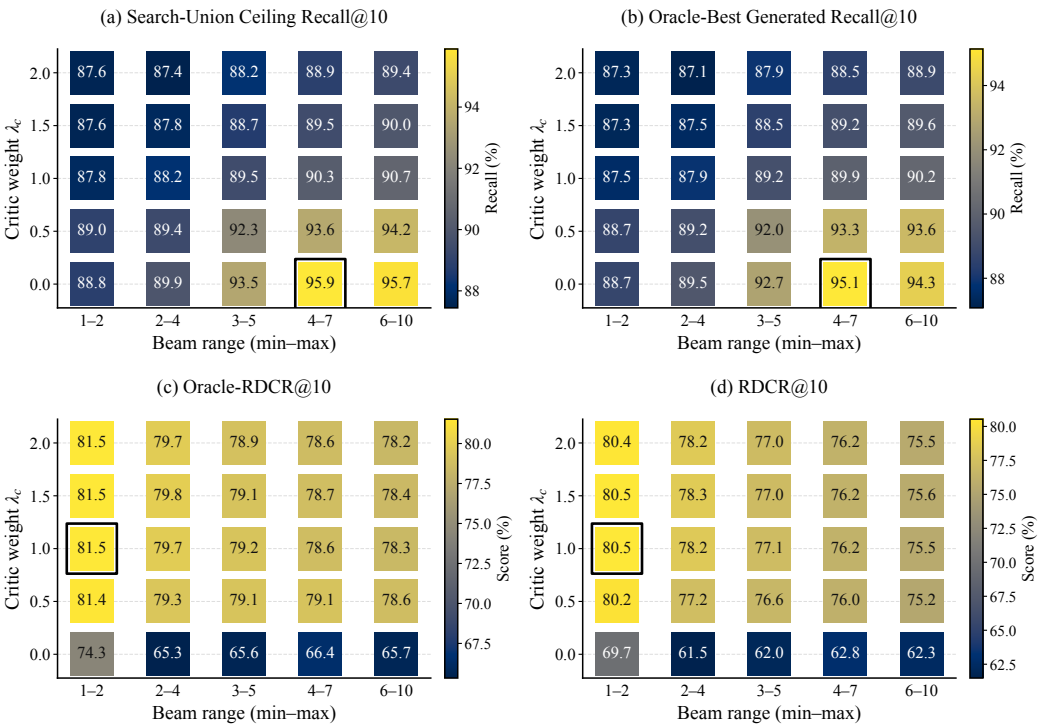


Figure 6: Validation sensitivity: (a) SUCR, (b) OGR, (c) Oracle-RDCR, and (d) actual RDCR, all on final Top-10 lists. Values are percentages; one setting attaining the largest displayed value is outlined per panel. The validation-selected main setting uses $(1, 2)$ and $\lambda = 0.5$.

Coverage and list quality. The SUCR–OGR gap is at most 1.4 percentage points across the grid. Without the critic, widening bounds from $(1, 2)$ to $(4, 7)$ increases SUCR from 88.8% to 95.9% and OGR from 88.7% to 95.1%, but reduces RDCR from 69.7% to 62.8%. The wider setting's Oracle-RDCR is 66.4%, below the narrower setting's actual RDCR, so reordering its selected list alone cannot recover the loss.

Final critic reranking. At $(1, 2)$, $\lambda = 0.5$ raises RDCR from 69.7% to 80.2% and Oracle-RDCR from 74.3% to 81.4%. The oracle-to-actual gap narrows from 4.6 to 1.2 points, reflecting improved selection and ordering. At $(4, 7)$, the same weight reduces SUCR from 95.9% to 93.6% but raises RDCR from 62.8% to 76.0%. Bounds $(1, 2)$ give the highest RDCR at every tested weight.

Across positive weights, RDCR varies by at most 1.1 points per beam setting. At $(1, 2)$ it spans 80.2–80.5%, compared with 69.7% without the critic. The selected $\lambda = 0.5$ is 0.3 points below the best displayed validation RDCR, attained at $\lambda = 1.0$ and 1.5 at reported precision. The tested positive weights therefore have similar ranked recall, while removing critic reranking produces a larger loss.

J End-to-End Execution Details

This section provides additional details for the 30-query, 120-execution evaluation reported in the main text.

J.1 Queries, resources, and candidate selection

The evaluation contains 25 queries sampled from the 629-query AgentSelect test partition and five queries from the 19-query SkillsBench test partition. The SkillsBench queries cover travel planning, dialogue parsing, invoice fraud detection, bike rebalancing, and software dependency auditing. Queries are screened only for execution feasibility before recommendation scores or execution outcomes are inspected, and reference-answer suffixes are removed from AgentSelect requests.

A normalized overlap check finds no overlap with generator-training or critic-training/validation queries.

For this execution study, all queries within a dataset use a common resource inventory containing 14 backbone models, 66 APIs, and 16 skills. No reference query–component associations are provided during candidate generation. The trained generator and ranking functions are unchanged; we only retain a larger set of legal generated candidates to evaluate configurations spanning different ranking positions.

Table 7: Execution candidate pools and runtime budgets. Candidate pools contain completed search configurations and legal pruned prefixes converted into complete configurations by termination. Candidates are deduplicated by backbone and component set.

Setting	AgentSelect	SkillsBench
Queries / executions	25 / 100	5 / 20
Backbone candidates	14	14
API endpoints / skills	66 / 16	66 / 16
Maximum components	6	10
Distinct candidates per query	311–321	366–376
Assistant turns / tool calls	12 / 24	24 / 48
Output tokens per response	4,000	6,000

Model identifiers are mapped to their corresponding SiliconFlow endpoints for execution. Candidate generation is initialized from all 14 backbone models. Each backbone root receives one component expansion before global beam pruning; thereafter, a beam width of four is used, and each active prefix considers up to ten legal component proposals without repetition. To obtain configurations covering a wider range of recommendation scores, we retain both completed search configurations and legal nonempty prefixes pruned during generation. Each retained prefix is converted into a complete candidate by appending its generator-scored termination action. This larger pool is used only for selecting configurations for execution and does not retrain or retune the generator or critic.

AgentSelect ranks the resulting candidates using

$$S(Q, A) = 0.5\, z_Q\Big(\widehat{U}_\psi(Q, A)\Big) + 0.15\, z_Q\big(\bar{L}_\theta(Q, A)\big) - 0.1|B|, \tag{18}$$

with the notation and within-pool standardization of Eq. (6). The critic is applied only after generation. SkillsBench uses its corresponding transfer-time scoring rule, ranking candidates by $\bar{L}_\theta$ without the length penalty. The final candidate pools contain 7,893 AgentSelect and 1,857 SkillsBench configurations.

For a ranked pool of size N, we select configurations at

$$(1,\ \lceil 0.5N \rceil,\ \lceil 0.9N \rceil,\ N)\,, \tag{19}$$

denoted *Top*, *Middle*, *P90*, and *Last*, respectively. These positions are fixed before execution and judging. *Last* therefore denotes the lowest-ranked configuration in the generated evaluation pool, not the lowest-ranked configuration among all combinatorially possible resource bundles.

J.2 Execution and scoring

Eleven APIs are mapped to executable public-service counterparts or a local similarity adapter. The remaining 55 follow the MirrorAPI Guo et al. (2025) simulation protocol, using web-search-enabled GPT-5.6-Luna to generate schema-constrained observations. The simulator receives the endpoint description, schema, arguments, and limited same-service interaction history, but no recommendation rank, score, configuration identifier, or backbone identity. Simulated responses are explicitly labeled as synthetic observations and are not treated as verified external facts. A fixed reference date of 23 September 2026 is used for temporal consistency. Identical simulator requests may reuse the same cached observation to reduce stochastic variation; the cache contains no rank- or configuration-specific state.

The 120 configurations are each executed once at temperature zero with provider thinking disabled, in a seed-42 randomized order independent of recommendation rank. Table 7 lists the turn, tool-call,

and output-token budgets. Wall-time guards of 3,600 seconds for AgentSelect and 5,400 seconds for SkillsBench do not terminate any observed run.

SkillsBench mounts only the selected skill packages as read-only resources in fresh network-isolated containers with two CPU cores and 8 GB of memory. All configurations share the same shell access, input files, and installed libraries; the software-auditing task additionally provides a common Trivy executable and offline vulnerability database. Because AgentSelect does not provide executable skill packages, selected skills are supplied as textual instructions. SkillsBench tasks and hidden tests use a pinned snapshot.[2] Shell calls have a 120-second limit, and exported artifacts are checked by the original hidden tests with a 150-second verifier limit.

The external judge is GPT-5.6-Tera (API identifier `gpt-5.6-terra`). Each recorded execution and final output is judged once. Judge inputs include the user request, exported artifacts, native or simulated tool observations, termination state, and available verifier results; recommendation rank, recommendation scores, backbone identity, and configuration identifier are excluded. The shared 0–10 quality rubric assigns 0–4 points each for task completion and correctness and 0–2 points for constraint satisfaction, in half-point increments. Judge inputs are limited to 55,000 characters per artifact and 7,000 characters per observation, while complete execution records are retained separately.

For each query, Spearman correlation and Kendall's τ_b compare negative recommendation rank with the four execution-quality scores s_{qi}, so positive values indicate agreement between recommendation order and execution quality. Pairwise concordance is defined as

$$C_q = \frac{1}{6} \sum_{1 \le i < j \le 4} \left[\mathbf{1}(s_{qi} > s_{qj}) + \frac{1}{2} \mathbf{1}(s_{qi} = s_{qj}) \right], \tag{20}$$

and $\Delta_q = s_{q1} - s_{q4}$ measures the Top–Last quality difference. Metrics are averaged across queries. Percentile 95% confidence intervals use 10,000 query-level bootstrap resamples with seed 42 while keeping each query's four outcomes paired. These intervals capture variation across queries rather than repeated executions or repeated judge calls.

J.3 Full-sample results and robustness check

Table 8: Query-averaged agreement between recommendation ranking and execution quality. Brackets denote 95% query-bootstrap confidence intervals; Δ is the Top–Last difference on the 0–10 quality scale.

Sample	Queries	Spearman ρ	Concordance (%)	Δ
All queries	30	0.634	76.4	5.17
		$[0.452, 0.784]$	$[68.3, 83.3]$	$[4.03, 6.17]$
AgentSelect	25	0.714	80.0	5.70
		$[0.549, 0.842]$	$[72.7, 86.7]$	$[4.68, 6.56]$
SkillsBench	5	0.230	58.3	2.50
		$[-0.343, 0.800]$	$[35.0, 81.7]$	$[-0.70, 5.70]$

All 120 configurations have recorded execution outcomes and quality scores. Table 8 reports the primary results used in the main text. Across all 30 queries, Kendall's τ_b is 0.570 ($[0.401, 0.714]$). Because the pooled sample contains 25 AgentSelect and five SkillsBench queries, the aggregate estimate is primarily driven by AgentSelect. The SkillsBench estimates have substantially wider confidence intervals; in particular, its Middle configurations obtain a higher mean score than its Top configurations (8.2 versus 6.3), so the five-query subset provides only limited evidence about fine-grained ordering within that domain.

As a robustness check, we additionally exclude one SkillsBench bike-rebalancing query in which two low-ranked runs terminate before artifact export because of a command-argument validation issue in the execution harness. The resulting 29-query estimates remain similar to the primary results: Spearman $\rho = 0.663$, concordance 77.6%, and $\Delta = 5.38$, with Kendall's $\tau_b = 0.596$. The primary analysis nevertheless retains all 30 queries and uses no score-dependent reruns.

[2] `https://github.com/benchflow-ai/skillsbench`, commit `9a1f4dd5f7659f75707435da3ce854b6e48321d1`.