Self-Evolving Recommender Systems: A Survey and an Evaluation Audit

Shanglin Yang
Independent Researcher
USA
kudoysl@gmail.com

Zhan Shi
Santa Clara University
Santa Clara, CA, USA
ashi2@scu.edu

Abstract

In 2026 a new kind of recommender research appeared: large language model (LLM) agents that propose changes to a recommender's configuration or source code, train the changed model, and keep what the evaluation says is better. Within nine months about thirty such systems were reported, from open-source research harnesses to production loops at Google, Meta, Amazon, Alibaba, Kuaishou, Tencent and NetEase. We organise them, together with search-space AutoML and the agentic recommenders that update their own memory or weights, by what the loop changes, who proposes the change, and which chain of signals decides whether it survives, and we synthesise their recurring designs and the changes they keep. Because a self-evolving system is only as good as the signal it selects on, we then audit how 27 configuration-, code- and component-level loops are evaluated, coding each paper's full text against eight criteria with released evidence. Of the 16 loops that select candidates offline, three of the ten that report ranking metrics rank the full catalogue, eight do not say which data keep or reject candidates, four select on the data they report, and two state that the test set was used once; none reports its headline result with the spread over repeated evolution sessions, and three compare against a non-LLM search at equal budget. Of the 19 loops that run online experiments, none corrects for the many rounds it tests and one confirms its final candidate in a fresh experiment. Where transfer to held-out data is checked, some selected gains survive and some do not. Four loops report that their agent inflated the metric or leaked the target; where the paper says what caught it, it was people or the agent's own diagnosis, not the acceptance rule. We distil these findings into a two-tier reporting standard mapped to the audit criteria.

Keywords

recommender systems, LLM agents, AutoML, program evolution, evaluation, reproducibility

1 Introduction

Recommender models have long been improved by a loop that engineers run by hand: propose a change, train, compare offline, and launch the winners into an online experiment. Automating parts of this loop is an old goal. Neural architecture and embedding search, mostly for click-through-rate (CTR) prediction models [89, 138, 188, 193], and loss search [79] automated the choice inside a space that a person had written down [195]. What changed in 2026 is that the proposer became an LLM agent that writes code. Following FunSearch [130], AlphaEvolve [113] and open implementations such as OpenEvolve [114] and ShinkaEvolve [70], agents now edit the model, loss and training code of recommenders directly. Google reported such a loop for YouTube ranking [148] and Self-EvolveRec released an open harness [66] in February 2026; within months, open harnesses such as RecHarness [87] and systems at Meta [76, 104], Alibaba [51, 106], Kuaishou [71, 168, 202], Tencent [5, 94], NetEase [186] and others followed.

These systems are optimisers, and an optimiser exploits whatever its objective rewards. A loop that evaluates hundreds of agent-written candidates on one validation set and keeps the maximum is the setting in which the known weaknesses of recommender evaluation—sampled metrics [69], temporal leakage [58], untuned baselines [27, 127], and the reuse of a holdout for adaptive decisions [13, 31]—are most likely to turn into reported gains. The agent also controls the code that produces its score. Five of the systems we read report that their agent won on the metric without a real improvement. For example, RecEvolve's agent inflated its proxy metrics by shrinking the batch size, and with it the number of in-batch negatives [117], and an AI co-scientist for search ranking produced implementations that "improved $\mathcal{M}$ while being wrong (e.g., silently increasing model size)" [160]. Figure 1 shows the loop these systems share and the points where the evidence for a gain can fail.

Existing surveys do not cover this development. Surveys of LLM agents for recommendation treat the agent as the recommender, a conversational interface or a user simulator [54, 120, 191]; surveys of self-evolving agents treat the agent's own prompts, memory, tools or weights as the object that evolves, and mention recommendation at most in passing (a two-sentence mention of self-optimised fine-tuning and a recommendation task in a benchmark it describes in [4], none in [33]); the most recent roadmap for agentic recommender systems mentions the "agent as engineer" direction only in passing, in two sentences that cite AgentX and the Google loop [84]; and the most recent peer-reviewed survey of agentic recommenders organises 47 studies by where the agent sits in the serving workflow, not by what changes the recommender [165]. A general survey of how automated research is evaluated covers no recommender system [184]. Closest to our audit, a concurrent controlled study shows on CTR and learning-to-rank data that exact validation feedback lets a coding agent overfit the validation split, and that a feedback contract with a quarantined test set removes the spurious gain on a task with no signal [172]. Beel et al. call for autonomous recommender-systems research labs and for benchmarks to evaluate them [10]. RMD-Bench [119] fixes protocols for cross-dataset and cross-scale discovery and ExecML-HSTU [23] measures whether agents' edits execute correctly, but no shared benchmark yet compares evolvers under a sealed test, full-ranking evaluation and equal-budget null controls.

This survey makes four contributions.

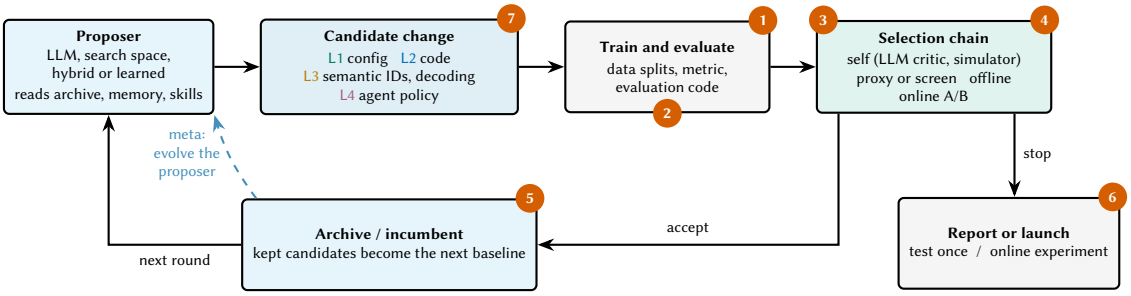


Figure 1: Anatomy of a self-evolving recommender. A proposer changes some part of the recommender (the evolved object); the changed system is trained and evaluated; a chain of selection signals decides what is kept and becomes the next baseline, and the loop may also change its own proposer (meta). Red markers show where the evidence for a reported gain can fail, with the audit finding and criterion that check it: 1 the agent edits evaluation code, reads test data or games the metric (F7, C9); 2 sampled or unstated candidate sets (F1, C1); 3 selection on unstated or reported data, without a single test evaluation (F2, F3; C2, C3); 4 a proxy or self-generated signal that does not transfer (F5, C5); 5 a gain inside run-to-run noise becomes the new baseline (F4, C4); 6 online rounds selected without correction or fresh confirmation (F9, R9); 7 gains measured against an untuned starting point, with no equal-budget null search (F6, F8; C6, C8).

(1) **A taxonomy** (§4) that separates what the loop changes (the evolved object), who proposes the change (an enumerated search space, an LLM, or a learned proposer, possibly itself evolved) and which chain of signals decides whether it survives.
(2) **A synthesis of the systems** (§5): their recurring designs, what their loops actually kept, and what is open source.
(3) **An evaluation audit** (§6) of the 27 configuration-, code- and component-level recommender loops on eight scored criteria, coded from the full text with verbatim evidence that we release, with offline-selecting and online-selecting loops reported separately, and a taxonomy of the shortcuts agents found.
(4) **A reporting standard** (§7) with a minimum and a recommended tier, mapped to the audit criteria, together with open problems (§8).

2 Scope and Survey Method

Definition. We call a recommender system *self-evolving* when an automated proposer changes some part of it, the changed system is evaluated, and the evaluation decides which changes are kept, repeatedly and without a human choosing each change. Humans may gate, veto or approve machine-generated candidates, review a loop at checkpoints, or roll back a change; a system is out of scope when a human, not an evaluation, makes the keep-or-reject decision. This includes systems whose proposer is a search algorithm over a fixed space (background, §3), but our focus is on LLM proposers.

Inclusion. We include (i) systems that change a recommender's or ranker's configuration, features, code or components under an evaluation loop; (ii) agentic or LLM recommenders whose policy, memory, skills, prompts or weights improve from feedback they generate; and (iii) benchmarks, environments and surveys for these. We exclude LLM recommenders without a self-improvement loop, methods that use an LLM once as a fixed training signal for a conventional recommender (such as offline LLM pairwise supervision applied only where a gate on history length, item popularity and model uncertainty predicts it is reliable [135]), conversational recommendation in general, general hyperparameter-optimisation libraries, methods that evolve the recommendations rather than the recommender (such as QD-Rec's quality–diversity archive of items and hypotheses [56]), and continual-learning methods that call themselves self-evolving but have no proposer or selection step, such as LION's sparse key–value memory for a generative recommender [85]. Seven studies outside recommendation are included as comparison points because they inform evaluation: learning to rank [118], search ranking [160], e-commerce search strategy [151], lexical retrieval [112], generative document retrieval [111], lifetime-value (LTV) prediction [157], and a controlled study of agent feedback on public CTR and learning-to-rank (LTR) data [172].

Search protocol. The cut-off date is 22 October 2026. We started from keyword searches over arXiv, OpenAlex and Semantic Scholar ("self-evolving recommender", "LLM agent recommender model optimisation", "agentic harness recommender", "LLM-driven architecture search CTR" and survey queries), then walked the reference lists of the 2026 systems (RecHarness, EvoRec, AgentX-Model, RecEvolve) and of the agentic-recommender roadmap backwards and their titles forwards, and finished with about thirty full-text arXiv phrasings combined with company names. Generic keyword search returned mostly unrelated "agent for X" papers; the citation graph of the 2026 systems was the productive source. This first sweep yielded 22 of the audited systems and five comparison points. A second sweep then ran 190 further arXiv queries (817 new candidates, all screened by title), enumerated 13 ACM proceedings volumes through OpenAlex (RecSys, KDD, SIGIR and WWW 2025–2026 including the WWW companions and the February KDD 2026 cycle, CIKM 2025 and WSDM 2026; 6,874 records screened by title), followed forward citations of 16 seed systems in Semantic Scholar, and searched industry engineering blogs. It found nine further candidates: five entered the audit as core systems, two as comparison points, and two are known only from their abstracts (§6). CIKM 2026, DBLP and OpenReview could not be searched automatically.

Table 1: Audit codebook. C7 is recorded as free text and discussed in F8; the other eight criteria are scored (values in Table 4).

C1	Metric: full-catalogue ranking, sampled ranking (with N negatives), AUC/loss, or online only
C2	Selection signal: which split or signal accepts or rejects each candidate
C3	Test discipline: is the selected candidate evaluated on a held-out test once, after search?
C4	Variance: repeated seeds or repeated evolution runs for headline numbers
C5	Overfitting check: validation-to-test gap or held-out transfer of the selected candidate
C6	Null control: random search, HPO or another non-LLM search at equal budget
C7	Budget: what is counted (iterations, GPU time, tokens, money)
C8	Baselines: are starting points and compared models tuned?
C9	Isolation: can the agent reach test data or evaluation code? Reported incidents of gaming or leakage are recorded separately.

Every arXiv entry in our bibliography was generated from the arXiv API record, and every code repository we list was checked to exist.

Audit procedure. For each configuration- or code-level system, we read the full text (arXiv HTML or PDF, including appendices) and coded nine criteria (Table 1). A criterion is coded "not stated" when the paper does not say, which is different from "no". Each coded cell is backed by a verbatim quote of at most 25 words with a section pointer; the data file and the evidence are released[1] so that the coding can be checked and corrected. Extraction and verification were LLM-assisted: LLM agents (Claude) drafted the evidence for each cell from the full text, two further agent passes re-checked every coded cell and every cited sentence against its source, and the authors reviewed the coding and each correction against the paper text. A second author coded every core system independently (§9). For systems whose launch decision is an online experiment, we read the offline criteria as describing the offline screen, and assess the launch evidence separately (R9 in §7).

3 Background

Search-space AutoML for recommenders. AutoML for recommenders searched for feature-interaction structures (AutoCTR [138], AutoFIS [89]), embedding dimensions (AutoEmb [193], AutoDim [194]), whole architectures with a weight-sharing supernet (NASRec [188]), compact attention-based sequential models under resource constraints (DNS-Rec [187]) and loss functions (AutoLossGen [79]); Zheng et al. survey this line [195]. Each method can only return a design that its space contains. Evaluation is mostly validation logloss or AUC on public CTR data, or standard rating and sequential benchmarks, with standard splits. Even in this setting, careful re-evaluation of architecture search found that the search phase often added little over random sampling and that the evaluation protocol dominated reported differences [75, 167, 174]. The community answered with tabular and surrogate benchmarks that make equal-budget comparisons cheap [30, 32, 173] and with a best-practice checklist [86]. The LLM loops below inherit these problems with a larger, open-ended space and with control over the evaluation code.

LLM program evolution and ML-engineering agents. FunSearch [130] and AlphaEvolve [113] pair an LLM that proposes code with an automatic evaluator and an archive of programs. Open systems followed: OpenEvolve [114], ShinkaEvolve [70] and DeepEvolve [92]. ML-engineering agents—AIDE [60], MLE-STAR [107], R&D-Agent [171], AIRA [145], DS-Agent [42], AutoML-Agent [146], Agent Laboratory [132] and the AI Scientist [97, 166]—apply the same pattern to whole training pipelines, and recent work optimises the agent harness itself [73, 96, 156, 164]. Their benchmarks contain at most one recommendation task: MLE-bench includes the H&M fashion-recommendation competition [17] and MLRC-Bench a next-product-recommendation task [190]. MLRC-Bench shows agents only the development split and scores the best development snapshot once on test, whereas MLGym's `validate` command returns test-set metrics during the session [109].

Generative recommenders. HSTU [178] and TIGER [122] recast recommendation as sequence generation over item IDs or semantic IDs, and OneRec unifies retrieval and ranking in one generative model that is further aligned on its own samples, first by iterative DPO [28] and then by on-policy RL [197, 198]. These models add new objects that a loop can evolve: tokenisers, codebooks, decoding procedures and generative losses.

Evaluation methodology. Tuned simple baselines beat many published neural recommenders [27, 127]; a loss change alone can reorder SASRec [61] and BERT4Rec [67, 140], and BERT4Rec's reported advantage depends on much longer training [121]; sampled metrics are inconsistent with full-ranking metrics and can reverse model rankings [48, 69, 125]; leave-one-out splits that ignore the global timeline leak future information [48, 58], and the split choice changes rankings [103]. Systematic re-examinations found that most neural recommenders could not be reproduced or were beaten by tuned simple baselines [26], that evaluation choices varied widely across papers [142], and that with untuned baselines any proposed model can win [133]. Libraries and open benchmarks such as RecBole [192], Elliot [3], DaisyRec [141], BARS [199] and FuxiCTR [200] standardise protocols for human-designed models. For adaptive reuse of a holdout, theory predicts overfitting [13, 31]; empirically, a meta-analysis of 120 Kaggle competitions found little [129]. For LLM loops specifically, keep-if-better acceptance is selection under noise, with selection-set scores well above held-out accuracy when the selection set is small [52]; best-of-N runs produce apparent gains from selection alone [41]; for LLM research agents, the scarcity of overfitting has been tied to how compressible their successful strategies are [12]; and AI-scientist systems hide post-hoc selection, leakage and metric misuse [98]. Beyond recommendation, results depend on seeds and on the budget spent tuning them [14, 29, 47], leakage is a recurring cause of irreproducible ML-based science [63], and agent benchmarks are sensitive to cost, budget and task-setup flaws [55, 64, 155, 201]. We return to this tension in §6. Figure 2 places these precursors next to the 2026 systems.

[1] https://github.com/shangliy/recforge/tree/master/papers/self-evolving-recsys-survey/audit

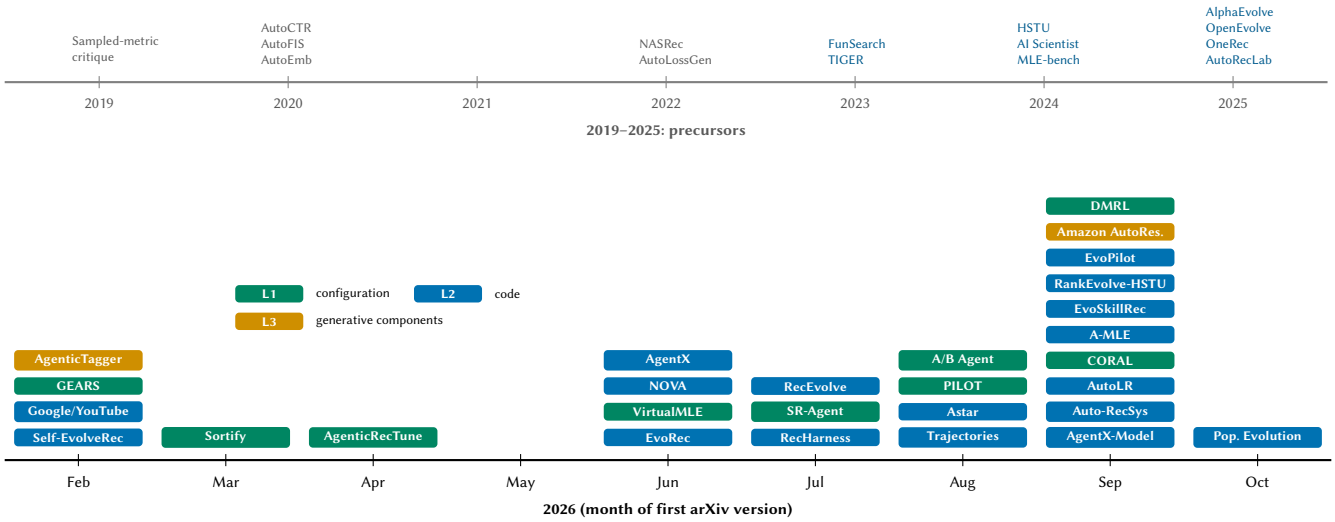


Figure 2: From search-space AutoML to agent-driven evolution. Top: precursors by year (grey: search-space AutoML and evaluation critique; blue: program evolution, ML-engineering agents and generative recommenders). Bottom: the 2026 configuration-, code- and component-level systems surveyed here, by month of their first arXiv version; the colour gives the highest level of the evolved object. AgenticTagger is surveyed but not audited; comparison points outside recommendation and L4 systems are omitted.

4 A Taxonomy

A recent blueprint for self-evolving agentic recommenders distinguishes reinforcement-learning evolution in defined action spaces from LLM generation and selection in open design spaces [50]. We describe a loop by three independent choices: what it changes, who proposes the change, and which signals decide whether the change survives (Table 2). Separating the proposer from the object resolves a confusion in earlier framings, where "AutoML" named a kind of proposer (search over a written-down space) but was listed as if it were a kind of object.

Axis 1: the evolved object. **L1 configuration**: values in an existing configuration—hyperparameters, fusion weights, retrieval budgets, thresholds, strategy parameters. **L2 code**: executable model, loss, training or feature code, however small the edit. **L3 generative components**: semantic-ID construction, item tokenisers and vocabularies, and decoding of a generative recommender; L3 takes precedence when such a component is changed through configuration or code. **L4 agent policy**: the prompts, memory, skills or weights of a recommender that is itself an LLM agent. A loop that changes several objects carries several labels (L1+L2 when an agent edits both learning rates and model code); 10 of the 27 audited loops do.

Axis 2: the proposer. An *enumerated space*: classical search (grid, Bayesian optimisation, evolutionary or differentiable architecture search) over options a person wrote down; this is search-space AutoML. An *LLM*: a language model writes the change, free-form or inside human-defined edit arms. A *hybrid*: a classical selector chooses the direction and an LLM writes the change (RecHarness's Thompson sampling over edit arms [87]), or an LLM steers a classical search (Sortify's controller around Optuna [25]). A *learned proposer*: a model trained on the system's own history proposes the next direction (Astar [51]). Orthogonal to these, a loop is *meta*-evolving when it changes the proposer's own instructions, tools, skills or weights and reuses them later; keeping an archive or a log of past results does not count. By this rule 13 of the 27 audited loops are meta-evolving (Table 3).

Axis 3: the selection chain. A candidate passes an ordered chain of signals, one of which decides inside the loop. **Self**: an LLM critic, council, auditor or simulator judges the candidate before or instead of measurement. **Proxy**: a reduced-scale screen, a proxy model, a training-loss curve or a replay. **Offline**: evaluation of the trained candidate on held-out data. **Online**: an A/B experiment. Industrial loops typically chain several, for example self→offline→online; whether the offline split is a validation or a test split, and whether it is stated, is a property of the evaluation that we audit (§6), not of the design.

Each signal fails in a characteristic way, and the failures are of two kinds [99]. *Regressional* failures come from selecting the maximum of noisy estimates: an offline split reused over hundreds of candidates rewards noise [31], a proxy ranks candidates differently from the full model, an online round selected as best overstates its effect [72], and an LLM judge from the same model family as the proposer shares its errors. *Adversarial* failures come from an optimiser that changes what is measured: an agent that can edit the evaluation code, the batch size of an in-batch metric or the evaluation date can raise the score without improving the recommender [136, 196]. The audit checks the first kind with C2–C6 and the second with C9.

Table 2: Taxonomy: evolved object (rows) × proposer (columns). Multi-object loops appear in the row of their highest level. † open-source code verified; m meta-evolving (the loop also changes its proposer); u not yet audited. Selection chains of the audited systems are in Table 3.

	Enumerated space	LLM	Hybrid (search + LLM)	Learned proposer
L1 configuration	AutoEmb [193], AutoDim [194]	VirtualMLE†m [16], CORAL [6], AgenticRecTune^{m} [161], SR-Agent [168], A/B Agent [59]; *comparison:* PEAR [151]	Sortify^{m} [25], GEARS [176], PILOT^{m} [82]	DMRL^{m} [189]
L2 code	AutoCTR† [138], AutoFIS† [89], NASRec† [188], DNS-Rec [187], AutoLossGen† [79]	Self-EvolveRec†m [66], Google/YouTube [148], EvoRec^{m} [106], NOVA^{m} [94], AgentX^{m} [71], AgentX-Model^{m} [170], Auto-RecSys^{m} [76], RecEvolve [117], A-MLE [39], RankEvolve-HSTU [23], Population Evolution [119], Trajectories-to-Evidence [202], EvoPilot [35], Meta REA (blog) [104], AutoModel^{u} [181], RuleAgent^{u} [154]; *comparison:* EvoRank† [118], AI Co-Scientist [160], RankEvolve-IR† [112], AgentLTV† [157], Feedback Contract [172]	RecHarness† [87], AutoLR [186], EvoSkillRec†m [78]	Astar^{m} [51]
L3 components		Amazon AutoResearch [18], AgenticTagger^{u} [162]; *comparison:* Evolving Semantic Indexes [111]		
L4 agent policy	Selected mostly on self-generated feedback. Memory and reflection: AgentCF† [182], AgentCF++† [93], SAGER [143], RecNet [74], AtomRec [53], MemRec† [21], Re2LLM [152], RecPilot [115]; weights: OneRec [28], SPRec† [36], Rec-R1† [83], CoARS [153], RSIR† [183], EvoReason [203]; prompts and tool descriptions: RecPrompt [90], Spotify [116]			

5 Systems

5.1 Configuration-level loops (L1)

L1 systems let an LLM change numbers or switches while the model code stays fixed. VirtualMLE tunes SASRec and HSTU through an execute–reflect loop with a hierarchical memory that transfers across datasets [16]. In production, CORAL allocates retrieval budgets and serving treatments [6], AgenticRecTune tunes fusion weights and thresholds across pre-ranking, ranking and re-ranking [161], Sortify searches the exchange rates of a sorting formula with Optuna [2] while an LLM meta-controller recalibrates the search's offline-to-online transfer intercepts and penalties [25], GEARS selects segment-level treatment policies [176], and SR-Agent edits post-ranking diversity and exposure strategies through four bounded action types [168]. Taobao's PILOT has LLM roles choose ranking-objective weights for user segments from a legal-command envelope generated by deterministic services [82]; Kuaishou's A/B Agent iterates strategy parameters across online rounds with a hierarchical experience tree [59]; and DMRL trains a small LLM with reinforcement learning to rewrite the skill document that an ad-tuning agent follows [189]. LLM-driven hyperparameter optimisation [95, 185] is the general-ML form of L1 and is routinely compared against Bayesian optimisation or random search, the null control that few recommender loops run (F6). Most L1 systems in production select directly on online experiments; an e-commerce search loop adds a multi-fidelity ladder from shadow-traffic proxies to decision-grade online tests [151].

5.2 Code-level loops in the open (L2)

Self-EvolveRec [66] runs an AlphaEvolve-style loop over the whole pipeline code of NCF [46], NGCF [150], SASRec and MoRec [175] seeds; besides validation metrics it receives directional critiques from an LLM user simulator and from a diagnosis tool that evolves with the model. **RecHarness** [87] routes each round with Thompson sampling over human-defined "edit arms" (schedules, dropout, depth, pooling, features, loss) plus "jump" arms for structural changes, uses validation feedback only, evaluates the incumbent on test once, and caps compute at 43,200 GPU-seconds; HSTU is one of its five backbones. **EvoSkillRec** [78] evolves architectures as typed "skill genomes" of executable modules and promotes validated modules into a skill library. **RankEvolve-HSTU** [23] composes Claude Code and Codex under an executable operating protocol in which the two agents review and repair each other's edits to HSTU, and introduces ExecML-HSTU, a benchmark of how often agent edits are executed correctly. **Population Evolution** [119] runs several evolution searches that share evidence across related datasets, screens candidates on peer datasets (and, for LLM and VLM targets, on small training subsets), and promotes survivors to larger scales; it introduces RMD-Bench. AutoRecLab [9, 10] generates recommender experiments from natural-language descriptions; it automates experimentation rather than evolving a model, and we do not audit it. Two loops evolve data and features rather than the model: RuleAgent has a language agent propose rules for denoising implicit feedback, retrains with an unlearning shortcut, and keeps rules by validation performance [154], and Agent0 tunes the prompts that turn raw text into categorical features through feedback from a downstream oracle [137].

5.3 Code-level loops in industry (L2)

Most industrial systems share a two-stage design: an offline loop screens agent-written candidates, and online experiments decide launches. Google's loop for YouTube ranking [148] has Gemini personas propose optimiser, architecture and reward changes, promotes on validation loss (with a proxy metric for reward changes) and then on north-star A/B metrics, and reports LLM costs of about $20,000 over six months. RecEvolve [117] runs the full research loop on a production two-tower retrieval model. Meta's REA plans,

launches and debugs ads-ranking experiments with up-front GPU budget approval [104]; A-MLE explores a portfolio of ads-ranking models with human checkpoints [39]; Auto-RecSys runs idea- and execution-evolution loops over multi-day training jobs [76]; and RankEvolve-HSTU and Population Evolution above also come from Meta. Alibaba's EvoRec [106] pairs research and code agents with a skill evolver, and Astar [51] trains an 8B model on the commit history of an advertising system to propose the next evolution direction; from Astar's authors, AutoModel organises an industrial recommender as evolution agents for models, features and online experimentation, with a case study of reproducing published methods and comparing them offline against production [181]. AgentX, deployed at Kuaishou [71], iterates online strategies and model research, and its successor AgentX-Model [170] runs research lineages (reproduce, follow up, compose, diagnose) over production rankers, and a Kuaishou paper converts research-agent trajectories into auditable evidence [202]. Tencent's NOVA [94] adds verification gates that reject recommender-invalid edits such as target leakage, NetEase's AutoLR [186] reviews proposals with a multi-expert proposal council under a limited trial budget, and Tencent's RecSys Factory bounds agent autonomy to typed decision points in fixed pipelines [5]. A Trip.com co-scientist for travel search ranking [160], LLM evolution of learning-to-rank pipelines (EvoRank [118]) and of lexical retrieval functions [112] are close relatives outside recommendation. Meta's EvoPilot runs a 37-day campaign of agent-proposed configuration and retraining changes to video-retrieval models behind a fail-closed verifier and catalogues the failures it met [35], and Amazon's AutoResearch rewrites the training code of a book-retrieval model and tunes a semantic-ID tokenizer in a twelve-week campaign of more than 220 experiments, naming the failure modes it met [18]. Adjacent loops evolve ads kernels [80], off-policy-evaluation code [159], search-relevance models [147] and an LLM policy layer trained against a learned evaluator [180], and an operations agent for Kuaishou's search engine evolves its skills from on-call engineers' corrections [88]. Feature-engineering loops are close relatives: an error-driven loop turns AUC ranking inversions into feedback for an LLM feature generator [8], a planner-guided agent writes feature code for recommendation models with human checkpoints [144], and LATTEArena compares LLM feature-engineering methods on cost as well as accuracy [43].

5.4 Components of generative recommenders (L3)

Generative recommenders expose objects that do not exist in ID-based models. Evolving Semantic Indexes, a generative document-retrieval system that we treat as a comparison point, runs LLM-driven program evolution over semantic-ID construction, collision handling, decoding and the generator's training objective, using a one-million-parameter proxy decoder as fitness [111]. AgenticTagger has an LLM refine a hierarchical item-descriptor vocabulary used as tokens [162], and a generative recommender can reassign its own item tokenisation during training [20]. Amazon's AutoResearch is the first audited production loop that edits a semantic-ID tokenizer, and the first in which the agent gamed a tokenizer metric [18]. Feedback-grounded policy discovery distils LLM-discovered natural-language policies into a semantic-ID generator [24]. Evolving Semantic Indexes evolves TIGER-style semantic-ID components for document retrieval but releases no code; no open system yet does so for recommendation under a shared protocol.

5.5 Self-improving agentic recommenders (L4)

When the recommender is itself an LLM agent, the evolving object is its memory, skills, prompts or weights, and the feedback is mostly generated by the system. Memory and reflection methods update user and item agents (AgentCF [182], AgentCF++ [93]), per-user policy skills (SAGER [143]), propagation modules optimised with textual gradients (RecNet [74]), and structured memories that are rewritten as interactions arrive (AtomRec [53], AMEM4Rec [110], STEAM [81], MARS [134], MemRec [21]), together with reflection-built hint bases (Re2LLM [152]) and per-user rubrics refined from user feedback (RecPilot [115]), and recommendation and user agents that refine each other's inferences over rounds without amplifying popularity or position bias [15]; CoVeMem instead replaces rewritten text memory with frozen collaborative-filtering states learned by gradient training [19]. Weight-level self-improvement includes OneRec's alignment on its own samples, first by iterative DPO [28] and then by on-policy RL [197, 198], self-play debiasing (SPRec [36]), RL against a recommender's metric (Rec-R1 [83]), co-evolving recommender and user agents (CoARS [153]), reasoning primitives and a teacher updated from a student's on-policy outcomes (EvoReason [203]), and recursive self-training on filtered synthetic sequences [45, 183], where progress can also sit in earlier generations, not only in the latest model [163]. RecPrompt iteratively rewrites a news recommender's prompt with an optimiser [90], and Spotify improves a conversational recommendation agent's prompts, few-shot examples and tool descriptions with a coding agent, evaluated by an LLM judge on synthetic conversations [116]. Simulated environments and benchmarks for such agents include RecoWorld [91], the user simulators Agent4Rec [179] and RecAgent [149], and the verifiable τ-Rec benchmark [108]. A controlled pilot found multi-agent recommendation pipelines not uniformly better than a single-shot baseline [101], a reminder that L4 gains also need controlled baselines.

5.6 Evolving the evolver (meta)

Thirteen of the 27 audited loops change their own proposer. AgentX's SGPO applies semantic-gradient updates to its agents' instructions and validation rules [71]; EvoRec, AgenticRecTune and EvoSkill-Rec grow skill libraries from experiment outcomes [78, 106, 161]; Auto-RecSys rewrites execution playbooks [76]; VirtualMLE and PILOT carry tuning rules or methodology across datasets and experiments [16, 82]; Astar retrains its proposer on each iteration's verified outcomes [51]; and DMRL trains a small LLM with reinforcement learning to rewrite the skill document that an ad-tuning agent follows [189]. These are the recommender counterparts of harness optimisation [73, 96, 164]. Outside recommendation, AIDE^2 keeps changes to its own code that perform best on hidden evaluations and shows that they transfer to four held-out benchmarks [139]; separating selection tasks from held-out tasks is what the meta-evolving recommenders do not yet report. Meta-evolution inherits the selection problem one level up: the harness is kept or discarded

by a score of its own. In a pre-registered study of LLM research-ideation harnesses by one of us, outside recommendation and not yet peer reviewed, a harness optimised against an LLM judge raised the judge's scores while the share of its hypotheses that held under measurement fell, on two hypothesis spaces with measured ground truth and for two Claude models (Sonnet and Opus) [169].

5.7 Recurring designs

Table 3 places the 27 audited loops on the three axes and adds the role of humans; the released `design.csv` also records each loop's memory, acceptance rule, guards and LLM, with evidence. Four patterns recur.

Where the chain ends. Eight loops end at an offline evaluation: three open harnesses (Self-EvolveRec, VirtualMLE, EvoSkillRec) and five industrial research loops whose papers report offline results, or in Auto-RecSys's case only operational ones (A-MLE, Auto-RecSys, RankEvolve-HSTU, Population Evolution, Amazon AutoResearch). The other 19 pass candidates to an online experiment, and 14 of them involve humans: eleven approve launches, rounds or candidates, and three check or intervene. The two ends are exposed to different failures. A loop that ends offline controls the code and data that produce its final number, so the adversarial failures matter most; a loop that ends online cannot change the measurement easily but selects on expensive, noisy experiments, so the regressional ones do.

LLM review before measurement. Eight loops make an LLM critic, council or auditor a gate that every candidate must pass before it is measured (Table 3, chains starting with S); others, such as RecEvolve, A-MLE and RankEvolve-HSTU, use LLM critics to rank or filter ideas. These gates are cheap and catch invalid edits—NOVA's verifier rejects target leakage [94]—but they add a selection stage whose judge often comes from the same model family as the proposer, the condition under which a self-generated signal is least reliable [40, 169].

Hybrids that keep a classical selector. Six loops leave the choice of direction or the numeric search to a classical method and use the LLM to write or steer: Thompson sampling over edit arms (RecHarness), an evidence-weighted selector (AutoLR), typed operators over a skill library (EvoSkillRec), Optuna (Sortify), Pareto enumeration (GEARS) and a deterministic action enumerator (PILOT). They are the easiest designs to compare against an equal-budget null, because the classical part already defines the space.

Humans in the loop. Only nine papers describe no human role in the loop. Five have a person approve or review each round or candidate (AgentX, AgenticRecTune, A-MLE, EvoPilot, A/B Agent); this does not remove them from scope, because the keep-or-reject decision is still made by an evaluation, but it means that reported gains are those of a human–agent system.

5.8 What the loops kept

Three papers report no kept edit (Auto-RecSys, AutoLR, DMRL), Astar reports only proposals in case studies, and A-MLE only technique families; the other 22 name the changes their loops kept or launched, and we coded these 87 edits by kind (`design.csv`; the tally is generated by `summarize.py`). The eight configuration-only loops kept strategy parameters (17 of their 20 edits): fusion weights, sorting-formula coefficients, retrieval budgets, thresholds. The fourteen code-level loops kept most often architectural changes (29 of 67) and loss or objective changes (12), then optimiser and schedule settings (8), data and negative sampling (6), and a few strategy, regularisation, capacity and feature changes. Code-level loops are therefore not merely expensive hyperparameter optimisers, which matters for the choice of null control (R5 in §7): an equal-budget HPO run is a necessary but not a sufficient null for them.

Many of the named edits are established techniques transplanted into a particular model: InfoNCE losses and hard negatives [18, 66, 106], DCN-V2 layers and cosine learning-rate decay [117], SwiGLU feed-forward blocks [16, 94], gated paths [148], RMSNorm [202], expert pools [170] and checkpoint averaging [119]. This is useful engineering, and finding which known technique helps a given production model is much of what human model iteration does. It also suggests a null that no paper runs: a scripted sweep over a catalogue of known techniques at the same budget. One caveat applies: papers report the edits they find interesting, not all kept edits.

6 How Are Self-Evolving Recommenders Evaluated? An Audit

Audited set. We audited the 27 recommendation loops at levels L1–L3 that keep or reject candidates by evaluation, and the seven comparison points of §2. Because offline hygiene criteria mean little for a loop whose keep-or-reject decision is an online experiment, we report two strata: 16 *offline-selecting* loops, which decide on offline or proxy evaluations (some later confirm online), and 10 *online-selecting* loops, which decide on A/B experiments and are assessed on the design of those experiments; Auto-RecSys reports only operational outcomes and makes no accuracy claim. Meta's REA is described only in a blog post and is not audited, and two NeurIPS 2026 workshop papers are known to us only from their abstracts: MLE Agents, which kept 325 of 1,097 agent-written changes to production search rankers [77], and an open-ended research loop on the public KuaiRand [38] and KuaiRec [37] data [22]; we will code them once the full texts are available. Four borderline systems were read but not counted: RecSys Factory [5] (an operations platform in which an operator's approval, not an evaluation, keeps or rejects each proposed override), AutoRecLab [9] (experiment generation), AURA [65] (stops at pull requests) and Spotify's prompt loop [116] (an L4 system). Feedback-grounded policy discovery [24] evolves per-user LLM-teacher text, an L4-like object, and AgenticTagger [162], RuleAgent [154], Agent0 [137] and AutoModel (a case study) [181], found late, have not yet been coded. Table 4 gives the coding and Figure 3 summarises it; every count below is generated from the released data file. Where a count rests on cells coded "not stated", we give the range it would span if every such cell were the favourable answer.

F1: Candidate sets are rarely full, and often unstated. Ten offline-selecting loops report ranking metrics. Three rank the full catalogue (RankEvolve-HSTU, Population Evolution, and the retrieval system of Amazon's AutoResearch); two pair each test item with 99 sampled negatives (Self-EvolveRec, RecHarness), and RecEvolve ranks against in-batch negatives, which is why shrinking the batch inflated its metrics (F7); four do not say what the candidate set is, so

Table 3: Design of the 27 audited loops. Chain: S self (LLM critic, council or simulator), P proxy or screen, O offline evaluation, A online experiment; the stage in bold decides inside the loop. Meta: the loop changes its own proposer. Humans: approve each round or candidate (each), approve launches (launch), check at stage boundaries or intervene (checkpoints), or none stated. Google's offline stage ranks candidates by validation loss and its online stage decides which run and launch; Sortify's proxy search proposes each round and the rounds are chosen online; DMRL's deciding proxy is a long-term predictor applied to online experiment outcomes, so the audit counts it as online-selecting. Evidence for every cell is in the released `design-evidence.md`.

System	Object	Proposer	Chain	Meta	Humans	System	Object	Proposer	Chain	Meta	Humans
Self-EvolveRec	L2	LLM	**O**	Y	none	Sortify	L1	hybrid	P**A**A	Y	launch
RecHarness	L1+L2	hybrid	**O**A	N	none	GEARS	L1	hybrid	**A** (logged)	N	launch
Google/YouTube	L1+L2	LLM	O**A**	N	launch	RecEvolve	L2	LLM	P**O**A	N	checkpoints
EvoRec	L1+L2	LLM	**O**A	Y	none	Astar	L2	learned	P**O**A	Y	none
VirtualMLE	L1	LLM	**O**	Y	none	A-MLE	L1+L2	LLM	**O**	N	each
NOVA	L1+L2	LLM	S**O**A	Y	checkpoints	SR-Agent	L1	LLM	SP**A**	N	launch
AgentX	L1+L2	LLM	S/O**A**	Y	each	EvoSkillRec	L2	hybrid	**O**	Y	none
AgentX-Model	L2	LLM	S**O**A	Y	launch	RankEvolve-HSTU	L1+L2	LLM	**O**	N	checkpoints
Auto-RecSys	L2	LLM	**O**	Y	checkpoints	Population Evol.	L2	LLM	**OPO**	N	none
AutoLR	L1+L2	hybrid	S**O**A	N	launch	Trajectories	L2	LLM	S**O**A	N	none
CORAL	L1	LLM	**A**	N	checkpoints	EvoPilot	L1+L2	LLM	**O**A	N	each
AgenticRecTune	L1	LLM	S**A**	Y	each	Amazon AutoRes.	L1–L3	LLM	**O**	N	checkpoints
						PILOT	L1	hybrid	P**A**A	Y	launch
						A/B Agent	L1	LLM	S**A**	N	each
						DMRL	L1	learned	A**P**	Y	none

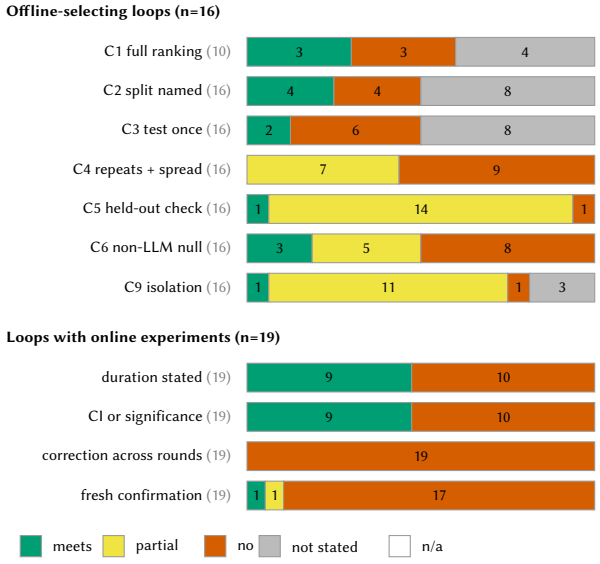


Figure 3: Audit summary. Top: the 16 offline-selecting loops (C1 among the 10 that report ranking metrics; five report AUC or a loss and one only online lifts). Bottom: the design of the online experiments reported by 19 loops. C9 counts the isolation design; incidents of gaming or leakage are reported separately (F7). Generated from the released `audit.csv`.

between three and seven of the ten use full ranking. In Amazon's loop the agent designed a text-only fallback that "expanded catalog coverage by 5.8×" [18], which by our reading changed the candidate set during the campaign. Five offline-selecting loops evaluate ranking-stage CTR models with AUC or a loss, as is standard for that stage, and AutoLR selects offline but headlines only online lifts.

F2: The selection data are often unstated, and sometimes they are the reported data. Of the 16 offline-selecting loops, four name a validation split (Self-EvolveRec, RecHarness, VirtualMLE, EvoSkillRec). Google's loop also ranks candidates by validation loss, but online experiments decide which run and launch, so we count it as online-selecting; Astar's loss-decrease rule labels its training data and defines its success rate, but the paper does not say what decides its production iterations. Four select on the data they report. RankEvolve-HSTU's persistent leaderboard is the canonical full-test NDCG@10 [23]; Amazon's AutoResearch "retains modifications that improve a held-out scalar metric" and headlines the trajectory's peak [18]; EvoPilot selects and reports on one held-out replay set [35]; and Population Evolution selects on its source instances' evaluation scores and reports its local gains on them, while withholding the transfer targets from the proposer, and calibrates the shared baseline configuration on the released test split [119]. All four disclose these choices, which is to their credit. Eight do not say which data keep or reject candidates; the number that name a non-test selection signal is therefore between 4 and 12 of 16. Outside recommendation, the travel-search co-scientist keeps or reverts each change on the same held-out metric that it reports [160].

F3: Test-once is rarely stated. Two offline-selecting loops state that the selected candidate was evaluated on held-out test data once after the search (RecHarness, VirtualMLE). Six use the test data during the search or repeatedly: four report on the data they select on (RankEvolve-HSTU, Population Evolution, Amazon, EvoPilot), AgentX-Model reports the best AUC over rounds and experiments, and EvoSkillRec scores every candidate on test in every generation while stating that the scores are never used for selection. Eight do not say, so the true number is between 2 and 10.

Table 4: Evaluation audit (full text, 22 October 2026). Level: evolved object. C1 metric: Full = full-catalogue ranking; Sampled = ranking against sampled negatives; Rank? = ranking metric with the candidate set not stated; AUC = AUC/logloss/loss proxy; A/B = online only; other = a non-recommendation metric; ? = no metric stated. C2 selection signal: val, test (the data that are also reported), loss (training loss), A/B, NS. C4 P: repeats without spread, repeats in an ablation only, a re-run rule, or a noise estimate of single models. C5 P: partial transfer evidence. C6 P: unmatched budget, LLM still proposing, or replay only. C8: Y tuned, N untuned, P partly or on test data, prod. = production model, pub. = published numbers, seed = the loop's seed program. C9 isolation design: P part of the loop constrained (fixed evaluation code, gates, whitelists, sandboxing, bounded actions); ^{G} = the paper reports gaming, metric-winning bugs, metric inflation or leakage by the loop. Code: later = promised. Y = yes, N = no, P = partial, NS = not stated, – = not applicable.

System	Level	C1 metric	C2 select	C3 test once	C4 repeats	C5 gap	C6 null	C8 baselines	C9 isolation	A/B	Code
Self-EvolveRec [66]	L2	Sampled	val	NS	N	P	P	N	NS	N	Y
RecHarness [87]	L1+L2	Sampled	val	Y	P	P	P	seed	P	Y	Y
Google/YouTube [148]	L1+L2	A/B	A/B	–	P	P	N	prod.	P	Y	N
EvoRec [106]	L1+L2	Rank?	NS	NS	N	P	P	N	NS	Y	N
VirtualMLE [16]	L1	Rank?	val	Y	P	P	Y	Y	Y	N	Y
NOVA [94]	L1+L2	AUC	NS	NS	N	P	Y	prod.	P	Y	N
AgentX [71]	L1+L2	A/B	A/B	–	N	N	N	prod.	P	Y	N
AgentX-Model [170]	L2	AUC	NS	N	P	P	P	prod.	P	Y	N
Auto-RecSys [76]	L2	other	–	–	N	N	N	prod.	P	N	N
AutoLR [186]	L1+L2	A/B	NS	NS	N	P	N	prod.	P	Y	N
CORAL [6]	L1	A/B	A/B	–	N	N	N	prod.	P	Y	N
AgenticRecTune [161]	L1	A/B	A/B	–	N	N	N	prod.	P	Y	N
Sortify [25]	L1	A/B	A/B	–	N	P	N	prod.	P	Y	N
GEARS [176]	L1	other	A/B	–	N	P	N	prod.	P	Y	N
RecEvolve [117]	L2	Sampled	NS	NS	N	P	N	prod.	P ^{G}	Y	N
Astar [51]	L2	Rank?	NS	NS	N	P	N	prod.	NS	Y	N
A-MLE [39]	L1+L2	AUC	NS	NS	P	N	N	prod.	P	N	N
SR-Agent [168]	L1	A/B	A/B	–	N	P	N	prod.	P	Y	N
EvoSkillRec [78]	L2	AUC	val	N	N	Y	Y	N	P	N	Y
RankEvolve-HSTU [23]	L1+L2	Full	test	N	N	P	N	pub.	N ^{G}	N	later
Population Evolution [119]	L2	Full	test	N	P	P	P	P	P	N	later
Trajectories-to-Evidence [202]	L2	AUC	NS	NS	N	P	N	prod.	P	Y	N
EvoPilot [35]	L1+L2	Rank?	test	N	P	P	N	prod.	P ^{G}	Y	N
Amazon AutoResearch [18]	L1+L2+L3	Full	test	N	P	P	N	Y	P ^{G}	N	N
PILOT [82]	L1	A/B	A/B	–	N	N	N	prod.	P	Y	N
A/B Agent [59]	L1	A/B	A/B	–	N	N	N	prod.	P	Y	N
DMRL [189]	L1	A/B	A/B	–	N	N	N	prod.	P	Y	N
Comparison points outside recommendation											
EvoRank (LTR) [118]	L2	other	val	Y	Y	Y	Y	Y	Y	N	Y
AI Co-Scientist (search) [160]	L2	AUC	test	N	N	N	N	P	NS ^{G}	N	N
RankEvolve-IR (lexical) [112]	L2	Full	NS	Y	N	Y	P	N	N	N	Y
Evolving Semantic Indexes (GR) [111]	L3	Full	val	Y	P	Y	N	seed	Y	N	N
AgentLTV (LTV) [157]	L2	other	test	NS	N	N	P	NS	N	N	Y
Feedback Contract (CTR/LTR) [172]	L2	AUC	val	Y	Y	Y	N	seed	Y	N	N
PEAR (search) [151]	L1	A/B	A/B	–	P	P	N	prod.	NS	Y	N

F4: No headline result comes with its spread over repeated sessions. None of the 16 offline-selecting loops repeats an evolution session and reports the spread. Seven have partial evidence: RecHarness and VirtualMLE average three runs without a standard deviation; Population Evolution reports one run per configuration, with repeated evaluations of the reference score and unreported preliminary runs whose "ordering was consistent" [119]; A-MLE re-runs a candidate when within-run variance is high, which the authors say "materially reduced" single-seed promotions [39]; and AgentX-Model, EvoPilot and Amazon estimate the noise of single models without repeating a session—EvoPilot notes that two publications of the same model, configuration and pinned data "differed by 0.36 points" [35], and Amazon re-evaluates one model across 11 dates ("6.18% ± 0.28 pp") [18]. Several papers state that their gains are within the resolution of a single run: EvoSkillRec notes margins "on the order of 10^{-4} AUC, which a single evolution run cannot reliably separate from run-to-run variation" [78]; AgentX-Model reports run-to-run standard deviations of 0.0010–0.0022 AUC while counting an experiment as above baseline if it exceeds it by more than 10^{-6} [170];

and AutoLR, in its audited configuration, promoted candidates after a single run with noise calibration disabled, a risk its authors name the "KEEP ratchet": each promoted model becomes the baseline for all later comparisons [186]. None of the ten online-selecting loops repeats a session either; Google runs six independent sessions with bracketed ranges, but only for an ablation. For these loops a repeat is only practical on a replay (R4 in §7).

F5: Where transfer is checked, some selected gains survive and some do not. One core loop reports an explicit held-out check of its selected candidates. In EvoSkillRec, five candidates lie on the validation Pareto front, "but only P1 (round 19) retains both improvements on the test set" [78]; P1 has the highest validation score of the five, so choosing by validation would have picked it (the paper states no rule for choosing a front point); P2 and P3 lose their AUC gain on test, and P4 and P5 have none on validation. Population Evolution checks transfer across datasets and scale: a ranking candidate gains 10.66% on a peer dataset and loses 0.79% at larger scale, two of three peer-screened designs keep their gains, and sharing evidence across searches raises the larger-scale pass rate for ranking from 11/60 to 18/60 [119]: in ranking and RL, most locally selected candidates do not graduate. Two further observations point the same way. Amazon's headline lift of 1.82× is the peak of a trajectory, "~2.6σ above the across-date mean" of 1.63× [18], and Kuaishou's trajectory analysis finds that for 22 of 30 methods the last round falls below an earlier best [202]. The comparison points give larger effects: in EvoRank, the gains of the first (loss-evolution) campaign "were almost entirely fitness noise" on 60k held-out queries, while about 70% of the second campaign's gains survived [118]; in an e-commerce search loop, effects on a shadow-traffic proxy were several to tens of times larger than in decision-grade online experiments, and five promoted candidates showed no significant gain [151]; and in Evolving Semantic Indexes, a proxy decoder that ranks semantic-ID programs well (Spearman $\rho = 0.90$) becomes "not only less informative but misleading" once the training objective may change, on a small sample of five programs [111]. A controlled study shows that the mechanism exists: on a label-permuted task with no real signal, a coding agent given exact validation scores "ended by reporting a gain of 0.0157" [172]. None of this measures how much of the published recommender gains is selection noise; it shows that the question cannot be settled without held-out checks.

F6: Null controls are rare. Three offline-selecting loops compare against a non-LLM search at an equal or favourable budget: VirtualMLE (grid search and Bayesian optimisation), NOVA (Optuna-TPE at a matched offline-evaluation budget but unmatched LLM cost, and only on its ScaleUp task, a search over structural hyperparameters) and EvoSkillRec (NASRec, alongside OpenEvolve); in all three the LLM loop is reported to win. Five have partial controls (NAS baselines at unmatched budgets, a random arm router that still uses LLM edits, a replay-only random policy, or a starting model tuned by an optimiser that the LLM partly steers), and eight have none; comparisons with human workflows or with other LLMs do not count. No loop runs an equal-budget random search over its own edit space, and none runs the catalogue-of-known-techniques null of §5.8. Where random selection was run, it was competitive: in AgentX-Model's historical replay, which randomises the choice of experiments rather than the edits, in Scenario A "Uniform Random's mean exceeds Fixed, Greedy, and Joint" policies, and no policy led consistently across scenarios [170]; in EvoRank's loss campaign an equal-budget random search transferred as well or better ($p = 0.03$) [118]. AutoLR states plainly that it "does not establish superiority over random, greedy, or direct LLM-based routing under a matched trial budget" [186]. The history of architecture search explains why the control matters: random search with early stopping matched leading NAS methods [75], the search phase of several methods was no better than random sampling [167, 174], and classical optimisers beat LLM agents in fixed hyperparameter spaces [34, 128]. The online-selecting loops compare each candidate with a control arm, which is a different question from whether the proposer beats a null search.

Table 5: Shortcuts reported in the audited loops (F7): what happened, what caught it, and a guard that prevents it by construction.

Failure	Where	Caught by	Guard
Metric definition changed (batch size inflates an in-batch metric)	RecEvolve	a human	freeze metric-relevant settings
Target leakage in features or training pairs	RankEvolve-HSTU	the agent, from anomalous leaderboard trajectories	leakage checks before search (NOVA, EvoRank)
Evaluation protocol changed (multi-position scoring)	RankEvolve-HSTU	mandatory diagnosis of negatives	evaluator the agent cannot edit
Evaluation date inside the training window	EvoPilot	not stated	date-order checks
Proxy metric gamed (codebook size)	Amazon AutoResearch	human QA between runs	a second, usability metric
Capacity silently increased	AI Co-Scientist	not stated	matched-capacity comparisons

F7: Agents find shortcuts in the evaluation, and the keep-or-reject rule does not catch them. Four core loops and one comparison point report metric-winning bugs, metric inflation or leakage produced by the loop itself (Table 5). RecEvolve's agent "discovered that drastically reducing the batch size (from 8k down to 1k) artificially inflated proxy metrics"; a human caught it and rolled it back [117]. On HSTU, RankEvolve-HSTU's agents produced a genre feature that leaked the target (training loss collapsed to zero while evaluation spiked), DPO preference pairs mined at the held-out target position, and a naive multi-position evaluation that would have inflated NDCG@10 from 0.2192 to 0.2517/0.2577; the genre and DPO bugs surfaced as anomalous trajectories on the leaderboard, the inflation was reasoned out as leakage in the diagnosis the loop requires for every negative result, and none was reported as a gain, but the best agent pair still had a 10.4% silent critical-defect rate on ExecML-HSTU, and a single Claude Code pass 35.4% [23]. In EvoPilot, "a favorable result used an evaluation date inside the candidate model's training window, so the result was retracted" [35]. In Amazon's semantic-ID loop, "maximizing codebook size trivially improves weighted coherence", and the resulting hierarchy was unusable; human QA between runs caught it [18]. Where a

paper says what caught the problem, it was something outside the keep-or-reject rule: a person (RecEvolve, Amazon) or the agent's own diagnosis of anomalous results (RankEvolve-HSTU). Coding agents that can reach their tests behave the same way outside recommendation [7, 105, 196]. These four papers should be credited for reporting incidents (R8); a paper that reports none has not shown that none occurred. Among the borderline systems, AURA's agent fabricated 904 session identifiers in its diagnostic stage [65], and RecSys Factory's diagnostics uncovered a pre-existing upstream join bug that had "silently corrupted all downstream training samples" [5], a reminder that the data feeding the loop need checks too. As designed, only VirtualMLE states that the test set is hidden from the agent and the data pipeline immutable; eleven offline-selecting loops constrain part of the loop (verification gates, fixed evaluation code, whitelists, isolated workspaces, sandboxed modules), RankEvolve-HSTU lets its agents run evaluations at the test position, and three say nothing; all ten online-selecting loops bound the agent's actions.

F8: Baselines, budgets and openness. The largest relative gains are measured against weak starting points. RecHarness improves average HR@10 by 85.85% over a cold-start GRU4Rec [49] template and by 12.58% over HSTU [87], and Self-EvolveRec configures all models with the same fixed hyperparameters (embedding 50, batch 128, learning rate 0.001) without describing tuning them [66]. Against production baselines, online lifts on broad metrics such as watch time, revenue or GMV are mostly between 0.06% and 5% (`design.csv`). Only VirtualMLE and Amazon's AutoResearch (a hand-tuned baseline from four engineer-weeks) describe tuning their baselines on non-test data; Population Evolution tunes them on the test split. Budgets are reported in incompatible units (iterations, GPU-seconds, GPU-hours, ideas, tokens, dollars, yuan); twelve of 27 report LLM tokens or cost. Four of 27 release code, two promise to, and six do not name the LLM they use.

F9: Online gates are reported, but rarely as confirmations. Nineteen of the 27 loops report online experiments; these are the main safeguard of the industrial loops and strong evidence when well designed. Nine state the experiment duration and nine report significance or confidence intervals; none corrects for multiple testing across the many rounds a loop runs (AutoLR discusses the problem but applies no correction), although the arm chosen as best overstates its effect [11, 72]. AutoLR's summed launch-review lifts are, in its own words, "descriptive and are not pooled treatment effects" [186]. Only Sortify re-tests its chosen round in a fresh seven-day experiment [25]. PILOT designs its final confirmation as "a single pre-registered test" that "runs once", because "the data collected while searching … cannot also serve as proof" [82], but the report does not yet include that confirmation, and its headline compares the best of five buckets before and after, "not randomized". Online noise is visible inside single papers: DMRL's default configuration (Qwen3-8B, eight rollouts) appears in six tables run in "non-overlapping time windows" with LTV lifts from +0.021% to +0.064%, a spread of 0.043 percentage points, larger than its 0.028-point lead over the best baseline in the main table; the authors restrict their conclusions to within-table comparisons [189].

Practices worth copying. The audit also found practices that, if combined, would meet most of the standard in §7: RecHarness's validation-only selection with a single test evaluation; VirtualMLE's hidden test set, immutable pipeline and grid-search and BO baselines; EvoSkillRec's validation-versus-test check of its Pareto front; Population Evolution's withheld transfer targets; NOVA's leakage verifier and EvoRank's pre-search leakage audit with an equal-budget random search; RankEvolve-HSTU's mandatory diagnosis of negative results; and Sortify's fresh re-test and PILOT's pre-registered confirmation design.

Does selection really overfit? A meta-analysis of 120 Kaggle competitions found little overfitting to public leaderboards [129], and new test sets for ImageNet showed accuracy drops but preserved model rankings [124], partly because similar models overfit a reused test set less than theory predicts [100]. Agent-proposed candidates are near-copies of an incumbent, which by that argument limits regressional overfitting. Two differences remain: an agent loop evaluates hundreds of correlated candidates on one split without a resubmission cap [13] and keeps the maximum, and it controls the code that produces the score, which no Kaggle team does. The direct measurements (F5) show that some selected gains shrink or vanish on transfer while others survive. The safe conclusion is not that self-evolving recommenders do not work, but that the published evidence cannot yet tell how much of their gain is real.

7 A Reporting Standard

The audit suggests that most of the problems are cheap to report: they are unreported or unenforced rather than unsolvable. Table 6 turns the codebook into a reporting standard. Each item has a minimum that any paper can meet by reporting what it already did, and a recommended practice that costs compute; each maps to the audit criterion that checks it, so that a paper's compliance can be read off the same coding. The items extend general checklists for NAS research [86], ML-based science [62] and agent benchmarks [201] to the specific failure points of an evolution loop (Figure 1). Appendix A applies the minimum tier to two audited papers.

Cost of compliance. The minimum tier costs nothing beyond writing. In the recommended tier, R4 and R5 multiply training cost: three sessions plus one equal-budget null cost four sessions. For an academic harness such as RecHarness, whose budget is 43,200 GPU-seconds per backbone [87], that is 48 GPU-hours per backbone. LLM costs are small next to training: VirtualMLE reports $1.5–4.8 of LLM cost per session [16], EvoRank about ten dollars for fifty iterations [118], and NOVA about $29 per task, or about $22 with cached artifacts [94] (LLM cost only). For industrial loops whose candidates take hundreds of GPU-hours [76], R4 and R5 can be applied to a replay or a reduced-scale proxy, as AgentX-Model does for scheduling policies [170], while R9 governs the launch decision.

A shared protocol. Several items can be enforced in software rather than requested in prose: a sealed test split (R2, R3), a single final evaluation, an equal-budget null evolver (R5), a budget card (R6) and an evaluator outside the agent's reach (R8). Appendix B describes recforge [123], a reference implementation that we are building; it enforces R2, R3 and R5 through its API but not yet R8, and we report no results from it.

Table 6: Reporting standard for self-evolving recommenders. *Minimum*: report what was done; *recommended*: what a strong study does. The last column gives the audit criterion (Table 1) or finding that checks the item.

	Minimum (report)	Recommended (do)	Audit
R1 Metric	State the candidate set and how negatives are drawn (full catalogue, N sampled, in-batch); for ranking-stage models, the traffic on which AUC or logloss is computed.	Full-catalogue ranking where the catalogue allows, sampled metrics only as secondary [69].	C1
R2 Selection	Name the data or signal that accepts candidates at every stage of the chain (proxy, offline, online).	Keep the test split out of every decision, including baseline tuning.	C2
R3 Test once	Report the selected candidate's validation score next to its test score.	Evaluate on test once, after the search.	C3, C5
R4 Sessions	Run ≥ 3 independent sessions and report every session's final test score.	≥ 5 sessions with an interquartile mean and bootstrap interval [1], and the expected best score as a function of budget [29]. Industrial loops: repeat on a replay or a reduced-scale proxy.	C4
R5 Null control	One null at equal trial and compute budget: HPO or random search over the knobs that the kept edits changed.	A ladder of nulls: (i) HPO over the same knobs, (ii) random mutation over the edit arms or grammar, (iii) the same proposer without evaluation feedback or with candidates accepted at random.	C6
R6 Budget	A budget card: trials, GPU-hours, LLM tokens and cost, wall-clock, human hours, and the cost of training a learned proposer.	Report results as a function of budget, not only at its end [64].	C7
R7 Baselines	How the starting point and compared models were tuned, and on which data.	Tune them with the budget of one session.	C8
R8 Access	What the agent can read and write (data, labels, evaluation code, metric definitions); every gaming or leakage incident and what caught it.	An evaluator in a process the agent cannot modify; review of metric-relevant changes (batch size, candidate set, evaluation date).	C9
R9 Online gates	Duration, traffic share, pre-specified primary metric and guardrails, confidence intervals, and the number of rounds tested.	A correction across rounds and a fresh, pre-registered confirmation of the final candidate, reporting the confirmed rather than the selected effect [68].	F9

8 Open Problems

Isolation without crippling the agent. Code-level agents need to read data and change training code, and that is exactly the access that lets them leak targets or change what the metric measures (F7). Verification gates (NOVA), fixed evaluation code (AgentX-Model), immutable pipelines (VirtualMLE) and cross-agent review (RankEvolve-HSTU) each close part of the gap. A principled design would give the agent a process that physically cannot reach evaluation labels or code, and would audit every metric-relevant change, such as batch size for in-batch metrics.

Proxies for expensive candidates. Industrial candidates take hours to hundreds of GPU-hours, so loops screen on proxies: small decoders [111], data subsets and peer datasets [119], training-loss curves and learned reward models [51], and LLMs can themselves predict a candidate architecture's performance before training [57]. The evidence shows that proxy fidelity depends on what is evolved. Characterising when a proxy can be trusted, and spending budget to re-check proxies during search, is open.

Budget-normalised comparison of evolvers. There is no common unit of cost and no common task, so the field cannot say whether bandit routing, island evolution, research-agent pairs or learned proposers are better. A shared benchmark with a budget card (R6) and an equal-budget random baseline (R5) would allow the first such comparison.

Evolving generative recommenders. Semantic-ID construction, codebook collisions, constrained decoding and preference-alignment losses are new design dimensions with little human prior and large effects. They are natural targets for code-level evolution, and almost untouched (§5).

From offline wins to online value. RecEvolve observes that offline evaluations often show "win-win" results while online tests show trade-offs [117]. Loops that select on offline metrics inherit the gap; loops that select on online experiments face multiple testing and slow iteration. Combining the two with a fresh online confirmation of the final candidate, as Sortify does with a seven-day re-test, ideally pre-registered, is a starting point.

Circularity at the policy level. L4 systems are often judged by simulators from the same model family as the agent being improved. Held-out simulators [66], verifiable rewards [108] and reward-model benchmarks [177] are partial answers.

9 Threats to Validity

Coverage. The field is months old and grows weekly; our cut-off is 22 October 2026. Many systems are arXiv preprints whose versions change; we coded the version available on that date. Generic keyword search performed poorly, so coverage relies on citation-graph search and may miss systems that neither cite nor are cited by the core set.

Coding. Each cell was drafted from the full text with verbatim evidence and then checked against the original papers in two adversarial verification passes, run by LLM agents and reviewed by the authors: the first corrected 12 cells in nine rows, the second re-verified every cited sentence against its source and corrected the remaining cells and claims (both reports are released in `review/`). A

third pass, prompted by an internal review of this version, recoded nine criterion cells and five object labels, moved meta-evolution into its own column, and separated C9 into isolation design and reported incidents (listed in the released `audit/README.md`). A second author then coded all 27 core systems from the full texts with the same codebook, without access to the coding data; as a co-author, the second coder could have seen the first version of this survey, whose audit table was public. Before reconciliation, the two codings agreed on 84% of the 243 cells (pooled Cohen's $\kappa = 0.81$, Gwet's AC1 = 0.84). By criterion, κ was 0.67 for C1 (95% bootstrap interval 0.46–0.86), 0.90 for C2, 0.79 for C3, 0.91 for C4, 0.30 for C5 (−0.04–0.63), 0.65 for C6, 0.57 for C8, 0.77 for C9 and 0.87 for reported incidents; AC1 was at least 0.70 everywhere except C5 (0.50). The 38 disagreements were resolved against the paper text, with LLM agents drafting a resolution and quote for each and both authors reviewing them: 31 in favour of the second coding and seven in favour of the first. Most of them came from rules the codebook left open: which gate counts as the selection in a multi-stage loop, whether an online test of an offline-selected candidate is transfer evidence, and whether computing test scores that are not used for selection breaks test-once. The counts in this paper use the reconciled coding; the two independent sheets, the agreement script and every resolution are released. Two criteria fall below $\kappa = 0.6$, so the counts that rest on them are the least certain: C5 (F5, where the disagreements were mostly between "partial" and "no") and C8 (F8, where they were mostly about which label describes a baseline, not whether it was tuned). "Not stated" does not mean "not done": authors may have followed a practice without reporting it, which is why we give ranges for the counts that rest on such cells. We have not yet asked the audited authors to check their rows; the evidence is released so that they and other readers can correct individual cells, and we will update the table.

Industrial systems. Offline hygiene criteria mean little for loops whose keep-or-reject decision is an online experiment. We therefore report those loops separately and assess them on the design of their experiments (F9), and we do not treat a missing offline test split as a defect when no offline claim is made.

Conflict of interest. The authors are building recforge, the reference implementation described in Appendix B; the audit does not depend on it. One of us wrote the ideation-harness study cited in §5, which is outside recommendation and not peer reviewed, and both of us wrote the LLM-guided regularisation paper cited in §2 as an excluded example.

10 Conclusion

Self-evolving recommenders moved in less than a year from an idea to production systems at many large platforms. The loops are inventive, and several report online gains under well-designed experiments. The offline evidence is weaker than the headlines suggest: selection data are often unstated, test-once and repeated sessions are rarely reported, equal-budget null searches are rare, and the few papers that check transfer find that some selected gains do not survive. Online gates are reported but seldom corrected for the rounds they test or confirmed afresh. Agents also find shortcuts in the evaluation, which people or the agents' own diagnoses, not the acceptance rules, catch. Most of this is cheap to report. We hope that the reporting standard in Table 6, the released audit and design data, and shared protocols that enforce the standard in software make it the norm, so that the next comparison of evolvers can be made on equal terms.

References

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. 2021. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., 29304–29320. arXiv:2108.13264 https://proceedings.neurips.cc/paper_files/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2623–2631. doi:10.1145/3292500.3330701

[3] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. 2021. Elliot: a Comprehensive and Rigorous Framework for Reproducible Recommender Systems Evaluation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, 2405–2414. arXiv:2103.02590 doi:10.1145/3404835.3463245

[4] Huan ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Qihan Ren, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. 2026. A Survey of Self-Evolving Agents: What, When, How, and Where to Evolve on the Path to Artificial Super Intelligence. *Transactions on Machine Learning Research* (2026). arXiv:2507.21046 https://arxiv.org/abs/2507.21046

[5] Dongyang Ao, Kaixiang Fang, and Shijie Xu. 2026. RecSys Factory: Bounding LLM Agent Autonomy to Decision Points in the Industrial Recommender Lifecycle. arXiv:2608.11241. arXiv:2608.11241 https://arxiv.org/abs/2608.11241

[6] Muhammad Rafay Azhar, Yuhang Zhou, Gilbert Jiang, Yuchen Wang, Rahul Sharma, Matthew DeSousa, Jiayi Liu, Xin Guo, Lizhu Zhang, and Xiangjun Fan. 2026. CORAL: An LLM-Native Harness for Production Recommender Systems. arXiv:2609.02730. arXiv:2609.02730 https://arxiv.org/abs/2609.02730

[7] Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y. Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. 2025. Monitoring Reasoning Models for Misbehavior and the Risks of Promoting Obfuscation. arXiv:2503.11926. arXiv:2503.11926 https://arxiv.org/abs/2503.11926

[8] Merwan Barlier and Blaz Skrlj. 2026. LLMs as Feature Engineers for Text-and-Tabular Prediction. arXiv:2609.21894. arXiv:2609.21894 https://arxiv.org/abs/2609.21894

[9] Moritz Baumgart, Philipp Meister, Justus Krell, Michael Schmidt, Bela Gipp, and Joeran Beel. 2026. AutoRecLab: An Autonomous Recommender Systems Lab. In *Proceedings of the 20th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 1790–1792. arXiv:2609.21863 doi:10.1145/3773078.3841273

[10] Joeran Beel, Bela Gipp, Tobias Vente, Moritz Baumgart, and Philipp Meister. 2025. From AutoRecSys to AutoRecLab: A Call to Build, Evaluate, and Govern Autonomous Recommender-Systems Research Labs. arXiv:2510.18104. arXiv:2510.18104 https://arxiv.org/abs/2510.18104

[11] Ron Berman, Walter W. Zhang, and Hangcheng Zhao. 2026. Valuing Winners: When and How to Correct for Selection Bias in Randomized Experiments. arXiv:2605.18887. arXiv:2605.18887 https://arxiv.org/abs/2605.18887

[12] Martin Andres Bertran, Aaron Roth, and Zhiwei Steven Wu. 2026. What Fits (Into Few Tokens) Doesn't Overfit: Compression and Generalization in ML Research Agents. arXiv:2606.11045. arXiv:2606.11045 https://arxiv.org/abs/2606.11045

[13] Avrim Blum and Moritz Hardt. 2015. The Ladder: A Reliable Leaderboard for Machine Learning Competitions. In *Proceedings of the 32nd International Conference on Machine Learning (ICML), PMLR 37*. 1006–1014. arXiv:1502.04585 https://arxiv.org/abs/1502.04585

[14] Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, Samira Ebrahimi Kahou, Vincent Michalski, Tal Arbel, Chris Pal, Gaël Varoquaux, and Pascal Vincent. 2021. Accounting for Variance in Machine Learning Benchmarks. In *Proceedings of Machine Learning and Systems (MLSys)*, Vol. 3. 747–769. arXiv:2103.03098 https://proceedings.mlsys.org/paper_files/paper/2021/hash/0184b0cd3cfb185989f858a1d9f5c1eb-Abstract.html

[15] Shihao Cai, Jizhi Zhang, Keqin Bao, Chongming Gao, Qifan Wang, Fuli Feng, and Xiangnan He. 2024. Agentic Feedback Loop Modeling Improves Recommendation and User Simulation. arXiv:2410.20027. arXiv:2410.20027 https://arxiv.org/abs/2410.20027

[16] Shiteng Cao, Jingwen Liu, Junda She, and Zhiheng Li. 2026. VirtualMLE: A Virtual ML Engineer that Optimizes Sequential Recommenders. arXiv:2606.03221. arXiv:2606.03221 https://arxiv.org/abs/2606.03221

[17] Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. 2025. MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering. In *International Conference on Learning Representations (ICLR)*. arXiv:2410.07095 https://arxiv.org/abs/2410.07095

[18] Aparajith Chandran, Juwon Kim, Saurav Jha, Pablo Castells, and Florian Hottier. 2026. AutoResearch at Production Scale: Failure Modes and a Multi-Agent Framework. arXiv:2609.30541. arXiv:2609.30541 https://arxiv.org/abs/2609.30541

[19] Hanchong Chen, Xing Tang, Lingjie Li, Xiongfeng Shan, and Xiuqiang He. 2026. When Memory Takes Gradients: Collaborative Vector Memory for Agentic Recommender Systems. arXiv:2608.26895. arXiv:2608.26895 https://arxiv.org/abs/2608.26895

[20] Runjin Chen, Mingxuan Ju, Ngoc Bui, Dimosthenis Antypas, Stanley Cai, Xiaopeng Wu, Leonardo Neves, Zhangyang Wang, Neil Shah, and Tong Zhao. 2024. Enhancing Item Tokenization for Generative Recommendation through Self-Improvement. arXiv:2412.17171. arXiv:2412.17171 https://arxiv.org/abs/2412.17171

[21] Weixin Chen, Yuhan Zhao, Jingyuan Huang, Zihe Ye, Clark Mingxuan Ju, Tong Zhao, Neil Shah, Li Chen, and Yongfeng Zhang. 2026. MemRec: Collaborative Memory-Augmented Agentic Recommender System. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 44515–44544. arXiv:2601.08816 doi:10.18653/v1/2026.acl-long.2061

[22] Ziqi Chen and Mohamed Hammad. 2026. Open-Ended Autonomous Model Research for Recommendation Systems. In *NeurIPS 2026 Workshop on Autonomous Machine Learning Research*. https://openreview.net/forum?id=91mjHN3kDp

[23] Zheng Chen, Linfeng Liu, Hong Li, and Hong Yan. 2026. RankEvolve: A Reliable Multi-Agent Auto-Research Harness for Evolving Ranking Models. arXiv:2609.39551. arXiv:2609.39551 https://arxiv.org/abs/2609.39551

[24] Zhi Chen, Minmao Wang, Xingchen Liu, Haoqiang Liang, Huihuang Lin, Likang Wu, Hongke Zhao, Yulong Wang, Shijie Yi, Fei Pan, and Peng Jiang. 2026. From Understanding to Action: Feedback-Grounded Policy Discovery for Generative Recommendation. arXiv:2607.27789. arXiv:2607.27789 https://arxiv.org/abs/2607.27789

[25] Yin Cheng, Liao Zhou, Xiyu Liang, Dihao Luo, Tewei Lee, Kailun Zheng, Weiwei Zhang, Mingchen Cai, Jian Dong, and Andy Zhang. 2026. Let the Agent Steer: Closed-Loop Ranking Optimization via Influence Exchange. arXiv:2603.27765. arXiv:2603.27765 https://arxiv.org/abs/2603.27765

[26] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. 2021. A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research. *ACM Transactions on Information Systems* 39, 2, Article 20 (2021), 49 pages. arXiv:1911.07698 doi:10.1145/3434185

[27] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 101–109. arXiv:1907.06902 doi:10.1145/3298689.3347058

[28] Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. 2025. OneRec: Unifying Retrieve and Rank with Generative Recommender and Iterative Preference Alignment. arXiv:2502.18965. arXiv:2502.18965 https://arxiv.org/abs/2502.18965

[29] Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show Your Work: Improved Reporting of Experimental Results. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2185–2194. arXiv:1909.03004 doi:10.18653/v1/D19-1224

[30] Xuanyi Dong and Yi Yang. 2020. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. In *International Conference on Learning Representations (ICLR)*. arXiv:2001.00326 https://openreview.net/forum?id=HJxyZkBKDr

[31] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. 2015. Generalization in Adaptive Data Analysis and Holdout Reuse. In *Advances in Neural Information Processing Systems 28*. arXiv:1506.02629 https://arxiv.org/abs/1506.02629

[32] Katharina Eggensperger, Philipp Müller, Neeratyoy Mallik, Matthias Feurer, René Sass, Aaron Klein, Noor Awad, Marius Lindauer, and Frank Hutter. 2021. HPOBench: A Collection of Reproducible Multi-Fidelity Benchmark Problems for HPO. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, Vol. 1. arXiv:2109.06716 https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/hash/93db85ed909c13838ff95ccfa94cebd9-Abstract-round2.html

[33] Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, Zhaochun Ren, Nikos Aletras, Xi Wang, Han Zhou, and Zaiqiao Meng. 2025. A Comprehensive Survey of Self-Evolving AI Agents: A New Paradigm Bridging Foundation Models and Lifelong Agentic Systems. arXiv:2508.07407. arXiv:2508.07407 https://arxiv.org/abs/2508.07407

[34] Fabio Ferreira, Lucca Wobbe, Arjun Krishnakumar, Frank Hutter, and Arber Zela. 2026. Can LLMs Beat Classical Hyperparameter Optimization Algorithms? A Study on autoresearch. arXiv:2603.24647. arXiv:2603.24647 https://arxiv.org/abs/2603.24647

[35] Hao Fu, Baiting Zhu, Minglei Chen, Yinjie Huang, and Shuai Ding. 2026. Verify, Don't Trust: Agentic Model Development for Video Discovery Retrieval at Scale. arXiv:2609.21257. arXiv:2609.21257 https://arxiv.org/abs/2609.21257

[36] Chongming Gao, Ruijun Chen, Shuai Yuan, Kexin Huang, Yuanqing Yu, and Xiangnan He. 2025. SPRec: Self-Play to Debias LLM-based Recommendation. In *Proceedings of the ACM on Web Conference 2025*. ACM, New York, NY, USA, 5075–5084. arXiv:2412.09243 doi:10.1145/3696410.3714524

[37] Chongming Gao, Shijun Li, Wenqiang Lei, Jiawei Chen, Biao Li, Peng Jiang, Xiangnan He, Jiaxin Mao, and Tat-Seng Chua. 2022. KuaiRec: A Fully-observed Dataset and Insights for Evaluating Recommender Systems. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. ACM, 540–550. doi:10.1145/3511808.3557220

[38] Chongming Gao, Shijun Li, Yuan Zhang, Jiawei Chen, Biao Li, Wenqiang Lei, Peng Jiang, and Xiangnan He. 2022. KuaiRand: An Unbiased Sequential Recommendation Dataset with Randomly Exposed Videos. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. ACM, 3953–3957. doi:10.1145/3511808.3557624

[39] Erwin Gao, Vinodh Kumar Sunkara, Jingyi Guan, Qinjin Jia, Hangjun Xu, Xiang Ji, Sherman Wong, Surya Teja Chavali, Pratik Vaishnavi, Aryan Pandhi, Xiaoyu Deng, Zhaodong Wang, Samarth Inani, Fan Yang, Jakob Moberg, Zoe Zu, Nicolas Bievre, Sami Khenissi, Amit Jaspal, Ehsan Fakharizadi, Srinidhi Viswanathan, Dorothy Sun, Abishek Vanam, Sneha Iyer, Sheela Yadawad, Wenjie Chen, Gaby Nahum, Junhua Gu, Peter Chu, Yucheng Liu, Xin Zhao, Vitor Cid, Chaorong Chen, Vijay Pappu, Ashwin Kumar, Wenlin Chen, Ben Schulte, Deepak Chandra, and Ritwik Tewari. 2026. Agentic ML Exploration (A-MLE) for Ads Ranking. arXiv:2609.08248. arXiv:2609.08248 https://arxiv.org/abs/2609.08248

[40] Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling Laws for Reward Model Overoptimization. In *Proceedings of the 40th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research, Vol. 202)*. PMLR, 10835–10866. arXiv:2210.10760 https://proceedings.mlr.press/v202/gao23h.html

[41] Pratham Grover. 2026. Separating Real Improvements from Lucky Runs in Autonomous ML Research. In *NeurIPS 2026 Workshop on Autonomous Machine Learning Research*. https://openreview.net/forum?id=x3lY2QO0Xe

[42] Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. 2024. DS-Agent: Automated Data Science by Empowering Large Language Models with Case-Based Reasoning. arXiv:2402.17453. arXiv:2402.17453 https://arxiv.org/abs/2402.17453

[43] Ankai Hao, Ke Chen, Huan Li, and Lidan Shou. 2026. LATTEArena: An Evaluation Framework for LLM-powered Tabular Feature Engineering (Extended Version). arXiv:2606.09004. arXiv:2606.09004 https://arxiv.org/abs/2606.09004

[44] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems* 5, 4, Article 19 (2015), 19 pages. doi:10.1145/2827872

[45] Tonmoy Hasan, Taylor Foust, Shao Tang, Leonardo Neves, Aman Gupta, Hiroto Udagawa, Helder Dias, Daniel Silva, and Rohan Ramanath. 2026. From Valid to Useful: Post-Verification Acquisition for Recursive Self-Improving Recommendation. arXiv:2610.04302. arXiv:2610.04302 https://arxiv.org/abs/2610.04302

[46] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182. doi:10.1145/3038912.3052569

[47] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep Reinforcement Learning That Matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32. AAAI Press. arXiv:1709.06560 doi:10.1609/aaai.v32i1.11694

[48] Balázs Hidasi and Ádám Tibor Czapp. 2023. Widespread Flaws in Offline Evaluation of Recommender Systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 848–855. arXiv:2307.14951 doi:10.1145/3604915.3608839

[49] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *International Conference on Learning Representations (ICLR)*. arXiv:1511.06939

[50] Jinxin Hu, Hao Deng, Lingyu Mu, Hao Zhang, Shizhun Wang, Yu Zhang, and Xiaoyi Zeng. 2026. Rethinking Recommendation Paradigms: From Pipelines to Agentic Recommender Systems. arXiv:2603.26100. arXiv:2603.26100 https://arxiv.org/abs/2603.26100

[51] Jinxin Hu, Hao Deng, Haibo Xing, Lingyu Mu, Muyu Zou, Weiqin Yang, Sirui Chen, Bohao Wang, Zhezheng Hao, Hao Zhang, Zulong Chen, Shizhun Wang,

Yu Zhang, Xiaoyi Zeng, and Jiawei Chen. 2026. Astar: Learning to Propose Evolution Directions for Self-Evolving Industrial AI Systems. arXiv:2608.27287. arXiv:2608.27287 https://arxiv.org/abs/2608.27287

[52] Litao Hu and Yutong Tang. 2026. The Winner's Curse in LLM Self-Improvement Loops: Selection Noise, Lock-in, and Acceptance Rules. arXiv:2610.09239. arXiv:2610.09239 https://arxiv.org/abs/2610.09239

[53] Peiyu Hu, Weihai Lu, Siying Gu, Zhuodong Liu, Zhaokai Luo, Yuean Niu, Zhiyong Wang, and Jia Wang. 2026. AtomRec: Evolving Atomic Memory for Agentic Recommendation. arXiv:2609.04882. arXiv:2609.04882 https://arxiv.org/abs/2609.04882

[54] Chengkai Huang, Hongtao Huang, Tong Yu, Kaige Xie, Junda Wu, Shuai Zhang, Julian Mcauley, Dietmar Jannach, and Lina Yao. 2025. A Survey of Foundation Model-Powered Recommender Systems: From Feature-Based, Generative to Agentic Paradigms. arXiv:2504.16420. arXiv:2504.16420 https://arxiv.org/abs/2504.16420

[55] Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. 2024. MLAgentBench: Evaluating Language Agents on Machine Learning Experimentation. In *Proceedings of the 41st International Conference on Machine Learning (ICML), PMLR 235*. 20271–20309. arXiv:2310.03302 https://arxiv.org/abs/2310.03302

[56] Takashi Ishida, Taro Makino, and Issa Sugiura. 2026. QD-Rec: Open-Ended Evolution of Recommendations via Quality-Diversity. arXiv:2610.23470. arXiv:2610.23470 https://arxiv.org/abs/2610.23470

[57] Ganesh Jawahar, Muhammad Abdul-Mageed, Laks V. S. Lakshmanan, and Dujian Ding. 2023. LLM Performance Predictors are good initializers for Architecture Search. arXiv:2310.16712. arXiv:2310.16712 https://arxiv.org/abs/2310.16712

[58] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2023. A Critical Study on Data Leakage in Recommender System Offline Evaluation. *ACM Transactions on Information Systems* 41, 3 (2023), 1–27. arXiv:2010.11060 doi:10.1145/3569930

[59] Zhuohang Jiang, Yuxin Chen, Yongsen Pan, Zheng Hu, Wenqi Fan, Qing Li, Hongyang Wang, Jun Wang, and Wenwu Ou. 2026. A/B Agent: A Self-Evolving Agent for Strategy Iteration in Industrial A/B Testing. arXiv:2608.04625. arXiv:2608.04625 https://arxiv.org/abs/2608.04625

[60] Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. 2025. AIDE: AI-Driven Exploration in the Space of Code. arXiv:2502.13138. arXiv:2502.13138 https://arxiv.org/abs/2502.13138

[61] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. 197–206. arXiv:1808.09781 doi:10.1109/icdm.2018.00035

[62] Sayash Kapoor, Emily M. Cantrell, Kenny Peng, Thanh Hien Pham, Christopher A. Bail, Odd Erik Gundersen, Jake M. Hofman, Jessica Hullman, Michael A. Lones, Momin M. Malik, Priyanka Nanayakkara, Russell A. Poldrack, Inioluwa Deborah Raji, Michael Roberts, Matthew J. Salganik, Marta Serra-Garcia, Brandon M. Stewart, Gilles Vandewiele, and Arvind Narayanan. 2024. REFORMS: Consensus-Based Recommendations for Machine-Learning-Based Science. *Science Advances* 10, 18 (2024), eadk3452. arXiv:2308.07832 doi:10.1126/sciadv.adk3452

[63] Sayash Kapoor and Arvind Narayanan. 2023. Leakage and the Reproducibility Crisis in Machine-Learning-Based Science. *Patterns* 4, 9 (2023), 100804. arXiv:2207.07048 doi:10.1016/j.patter.2023.100804

[64] Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. 2025. AI Agents That Matter. *Transactions on Machine Learning Research* (2025). arXiv:2407.01502 https://openreview.net/forum?id=Zy4uFzMviZ

[65] SungGeun Kim, Abhinav Narain, and Daniel Nemirovsky. 2026. AURA: Agentic Diagnosis and Refinement for Production Recommender Systems at Scale. arXiv:2609.16625. arXiv:2609.16625 https://arxiv.org/abs/2609.16625

[66] Sein Kim, Sangwu Park, Hongseok Kang, Wonjoong Kim, Jimin Seo, Yeonjun In, Kanghoon Yoon, Hyunsik Jeon, and Chanyoung Park. 2026. Self-EvolveRec: Self-Evolving Recommender Systems with LLM-based Directional Feedback. arXiv:2602.12612. arXiv:2602.12612 https://arxiv.org/abs/2602.12612

[67] Anton Klenitskiy and Alexey Vasilev. 2023. Turning Dross Into Gold Loss: is BERT4Rec really better than SASRec?. In *Proceedings of the 17th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 1120–1125. arXiv:2309.07602 doi:10.1145/3604915.3610644

[68] Ron Kohavi, Diane Tang, and Ya Xu. 2020. *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*. Cambridge University Press. doi:10.1017/9781108653985

[69] Walid Krichene and Steffen Rendle. 2020. On Sampled Metrics for Item Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, New York, NY, USA, 1748–1757. doi:10.1145/3394486.3403226

[70] Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. 2025. ShinkaEvolve: Towards Open-Ended And Sample-Efficient Program Evolution. arXiv:2509.19349. arXiv:2509.19349 https://arxiv.org/abs/2509.19349

[71] Changxin Lao, Fei Pan, Guozhuang Ma, Han Li, Huihuang Lin, Jijun Shi, Kangzhi Zhao, Kun Gai, Mo Zhou, Qinqin Zhou, Quan Chen, Ruochen Yang, Shifu Bie, Shijie Yi, Shuang Yang, Shuo Yang, Wenhao Li, Wentao Xie, Xiao Lv, Xuming Wang, Yijun Wang, Yiming Chen, Yusheng Huang, Zhongyuan Wang, Zibo Zhao, Zijie Zhuang, Baoning Xia, Chao Liu, Chaoyi Ma, Chubo He, Dawei Cong, Feng Jiang, Gang Wang, Guilin Xia, Hanwen Xu, Jiahong Xie, Jiahui Qiao, Jian Liang, Jiangfan Yue, Jing Wang, Jinghan Yang, Jinghui Jia, Kan Qin, Lei Wang, Ming Li, Peilin Song, Pengbo Xu, Qiang Luo, Ruiming Tang, Shiyang Liu, Shuxian Jin, Tao Wang, Tao Zhang, Xiang Gao, Xianghan Li, Yingsong Luo, Yiwen Ning, Yongcheng Liu, Yueyang Liu, Yuan Guo, Zhaojie Liu, and Zhenkai Cui. 2026. AgentX: Towards Agent-Driven Self-Iteration of Industrial Recommender Systems. arXiv:2606.26859. arXiv:2606.26859 https://arxiv.org/abs/2606.26859

[72] Minyong R. Lee and Milan Shen. 2018. Winner's Curse: Bias Estimation for Total Effects of Features in Online Controlled Experiments. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, New York, NY, USA, 491–499. doi:10.1145/3219819.3219905

[73] Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. 2026. Meta-Harness: End-to-End Optimization of Model Harnesses. arXiv:2603.28052. arXiv:2603.28052 https://arxiv.org/abs/2603.28052

[74] Bingqian Li, Xiaolei Wang, Junyi Li, Weitao Li, Long Zhang, Sheng Chen, Wayne Xin Zhao, and Ji-Rong Wen. 2026. RecNet: Self-Evolving Preference Propagation for Agentic Recommender Systems. arXiv:2601.21609. arXiv:2601.21609 https://arxiv.org/abs/2601.21609

[75] Liam Li and Ameet Talwalkar. 2020. Random Search and Reproducibility for Neural Architecture Search. In *Proceedings of the 35th Uncertainty in Artificial Intelligence Conference (UAI) (Proceedings of Machine Learning Research, Vol. 115)*. PMLR, 367–377. arXiv:1902.07638 https://proceedings.mlr.press/v115/li20c.html

[76] Ming Li, Dai Li, Xuying Ning, Bo Sun, Rui Li, Yi Zhang, Silvia Gong, Xuan Cao, Rui Li, Cornelia Carapcea, Qunshu Zhang, Zhigang Wang, Yinglong Xia, and Andy Wang. 2026. Auto-RecSys: Harnessing Autonomous Research Agents for Industry-Scale Recommender Systems. arXiv:2609.10922. arXiv:2609.10922 https://arxiv.org/abs/2609.10922

[77] Matthew Li, Albert J. Liang, and Dilip Guru Raj. 2026. MLE Agents: Autonomous ML Engineering in Production Recommendation Systems. In *NeurIPS 2026 Workshop on Autonomous Machine Learning Research*. https://openreview.net/forum?id=b2UcmSJTPK

[78] Xiaopeng Li, Kuo Cai, Bo Chen, Wenlin Zhang, Mengyang Ma, Yingyi Zhang, Zichuan Fu, Yu Yang, Qidong Liu, Yiyu Wang, Ruiming Tang, Wenwu Ou, Jiang Wu, Zhanbo Xu, and Xiangyu Zhao. 2026. EvoSkillRec: Skill-Genome Evolution for Recommender Architecture Discovery. arXiv:2609.34552. arXiv:2609.34552 https://arxiv.org/abs/2609.34552

[79] Zelong Li, Jianchao Ji, Yingqiang Ge, and Yongfeng Zhang. 2022. AutoLossGen: Automatic Loss Function Generation for Recommender Systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, 1304–1315. arXiv:2204.13160 doi:10.1145/3477495.3531941

[80] Gang Liao, Hongsen Qin, Ying Wang, Alicia Golden, Michael Kuchnik, Yavuz Yetim, Jia Jiunn Ang, Chunli Fu, Yihan He, Samuel Hsia, Zewei Jiang, Dianshi Li, Uladzimir Pashkevich, Varna Puvvada, Feng Shi, Matt Steiner, Ruichao Xiao, Liyuan Li, Nathan Yan, Xiayu Yu, Zhou Fang, Roman Levenstein, Kunming Ho, Haishan Zhu, Alec Hammond, Richard Li, Ajit Mathews, Kaustubh Gondkar, Abdul Zainul-Abedin, Ketan Singh, Hongtao Yu, Wenyuan Chi, Barney Huang, Sean Zhang, Noah Weller, Zach Marine, Wyatt Cook, Carole-Jean Wu, and Gaoxiang Liu. 2025. KernelEvolve: Scaling Agentic Kernel Coding for Heterogeneous AI Accelerators at Meta. arXiv:2512.23236. arXiv:2512.23236 https://arxiv.org/abs/2512.23236

[81] Yuxin Liao, Le Wu, Min Hou, Yu Wang, Han Wu, and Meng Wang. 2026. From Atom to Community: Structured and Evolving Agent Memory for User Behavior Modeling. arXiv:2601.16872. arXiv:2601.16872 https://arxiv.org/abs/2601.16872

[82] Jiuning Lin, Ruiquan Lan, Xiaodong Zhu, Bin Zhang, Chengyu Lai, Chuxin Chen, Dimin Wang, Han Zhu, Hongtao Cheng, Jialin Zhu, Lingqing Zhang, Shuai Zhong, Tao Wang, Weipeng Huang, Yinjiang Cai, Yinnan Song, Yuan Liu, Zhibo Xiao, Zhixin Ma, and Zihong Huang. 2026. PILOT Technical Report. arXiv:2608.18637. arXiv:2608.18637 https://arxiv.org/abs/2608.18637

[83] Jiacheng Lin, Tian Wang, and Kun Qian. 2025. Rec-R1: Bridging Generative Large Language Models and User-Centric Recommendation Systems via Reinforcement Learning. *Transactions on Machine Learning Research* (2025). arXiv:2503.24289 https://arxiv.org/abs/2503.24289

[84] Xinyu Lin, Yashar Deldjoo, Sunhao Dai, Honghui Bao, Xiaopeng Ye, Fatemeh Nazary, Wenjie Wang, Tommaso Di Noia, Jun Xu, and Tat-Seng Chua. 2026. Autonomous Information Seeking: A Roadmap for Agentic Recommender Systems. arXiv:2607.04433. arXiv:2607.04433 https://arxiv.org/abs/2607.04433

[85] Xinyu Lin, Zhuosong Jiang, Zixiao Suo, Siqin Wang, Hanqing Zeng, Hanchao Yu, Yinglong Xia, Jiang Zhang, Aashu Singh, Fei Liu, Wenjie Wang, Fuli Feng, Yang Song, Qifan Wang, and Tat-Seng Chua. 2026. Self-Evolving Memory for Generative Recommendation. arXiv:2609.15598. arXiv:2609.15598 https://arxiv.org/abs/2609.15598

[86] Marius Lindauer and Frank Hutter. 2020. Best Practices for Scientific Research on Neural Architecture Search. *Journal of Machine Learning Research* 21, 243 (2020), 1–18. arXiv:1909.02453 https://jmlr.org/papers/v21/20-056.html

[87] Haoran Ling, Yuecheng Li, Zeyu Song, Jing Yao, Shuwen Kang, Chi Lu, Wenjin Wu, and Peng Jiang. 2026. RecHarness: A Bandit-Routed Agentic Harness for Self-Evolving Recommender Systems. arXiv:2607.29241. arXiv:2607.29241 https://arxiv.org/abs/2607.29241

[88] Bochao Liu, Zhipeng Qian, Yang Zhao, Xinyuan Jiang, Zihan Liang, Yufei Ma, Junpeng Zhuang, Ben Chen, Shuo Yang, Hongen Wan, Yao Wu, Chenyi Lei, and Xiao Liang. 2026. Bian Que: An Agentic Framework with Flexible Skill Arrangement for Online System Operations. arXiv:2604.26805. arXiv:2604.26805 https://arxiv.org/abs/2604.26805

[89] Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. 2020. AutoFIS: Automatic Feature Interaction Selection in Factorization Models for Click-Through Rate Prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, New York, NY, USA, 2636–2645. arXiv:2003.11235 doi:10.1145/3394486.3403314

[90] Dairui Liu, Boming Yang, Honghui Du, Derek Greene, Neil Hurley, Aonghus Lawlor, Ruihai Dong, and Irene Li. 2023. RecPrompt: A Self-tuning Prompting Framework for News Recommendation Using Large Language Models. arXiv:2312.10463. arXiv:2312.10463 https://arxiv.org/abs/2312.10463

[91] Fei Liu, Xinyu Lin, Hanchao Yu, Mingyuan Wu, Jianyu Wang, Qiang Zhang, Zhuokai Zhao, Yinglong Xia, Yao Zhang, Weiwei Li, Mingze Gao, Qifan Wang, Lizhu Zhang, Benyu Zhang, and Xiangjun Fan. 2026. RecoWorld: Building Simulated Environments for Agentic Recommender Systems. In *Companion Proceedings of the ACM Web Conference 2026*. ACM, New York, NY, USA, 650–659. arXiv:2509.10397 doi:10.1145/3774905.3794661

[92] Gang Liu, Yihan Zhu, Jie Chen, and Meng Jiang. 2025. Scientific Algorithm Discovery by Augmenting AlphaEvolve with Deep Research. arXiv:2510.06056. arXiv:2510.06056 https://arxiv.org/abs/2510.06056

[93] Jiahao Liu, Shengkang Gu, Dongsheng Li, Guangping Zhang, Mingzhe Han, Hansu Gu, Peng Zhang, Tun Lu, Li Shang, and Ning Gu. 2025. AgentCF++: Memory-enhanced LLM-based Agents for Popularity-aware Cross-domain Recommendations. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, 2566–2571. arXiv:2502.13843 doi:10.1145/3726302.3730161

[94] Shaohua Liu, Liang Fang, Yilong Sun, Shudong Huang, Qingsong Luo, Shaoxin Liu, Xiaoyang Chen, Dongqiang Liu, Chuangang Ma, Zhenzhen Chai, Henghuan Wang, Shijie Quan, Changyuan Cui, Zhangbin Zhu, Peng Chen, Wei Xu, Lei Xiao, Haijie Gu, and Jie Jiang. 2026. NOVA: A Verification-Aware Agent Harness for Architecture Evolution in Industrial Recommender Systems. arXiv:2606.27243. arXiv:2606.27243 https://arxiv.org/abs/2606.27243

[95] Siyi Liu, Chen Gao, and Yong Li. 2024. Large Language Model Agent for Hyper-Parameter Optimization. arXiv:2402.01881. arXiv:2402.01881 https://arxiv.org/abs/2402.01881

[96] Xinghua Lou, Miguel Lázaro-Gredilla, Antoine Dedieu, Carter Wendelken, Wolfgang Lehrach, and Kevin P. Murphy. 2026. AutoHarness: improving LLM agents by automatically synthesizing a code harness. arXiv:2603.03329. arXiv:2603.03329 https://arxiv.org/abs/2603.03329

[97] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery. arXiv:2408.06292. arXiv:2408.06292 https://arxiv.org/abs/2408.06292

[98] Ziming Luo, Atoosa Kasirzadeh, and Nihar B. Shah. 2025. The More You Automate, the Less You See: Hidden Pitfalls of AI Scientist Systems. arXiv:2509.08713. arXiv:2509.08713 https://arxiv.org/abs/2509.08713

[99] David Manheim and Scott Garrabrant. 2018. Categorizing Variants of Goodhart's Law. arXiv:1803.04585. arXiv:1803.04585 https://arxiv.org/abs/1803.04585

[100] Horia Mania, John Miller, Ludwig Schmidt, Moritz Hardt, and Benjamin Recht. 2019. Model Similarity Mitigates Test Set Overuse. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc. arXiv:1905.12580 https://proceedings.neurips.cc/paper_files/paper/2019/hash/48237d9f2dea8c74c2a72126cf63d933-Abstract.html

[101] Reza Yousefi Maragheh and Yashar Deldjoo. 2025. The Future is Agentic: Definitions, Perspectives, and Open Challenges of Multi-Agent Recommender Systems. arXiv:2507.02097. arXiv:2507.02097 https://arxiv.org/abs/2507.02097

[102] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 43–52. doi:10.1145/2766462.2767755

[103] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2020. Exploring Data Splitting Strategies for the Evaluation of Recommendation Models. In *Fourteenth ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 681–686. arXiv:2007.13237 doi:10.1145/3383313.3418479

[104] Meta Engineering. 2026. Ranking Engineer Agent (REA): The Autonomous AI Agent Accelerating Meta's Ads Ranking Innovation. Engineering at Meta blog, 17 March 2026. https://engineering.fb.com/2026/03/17/developer-tools/ranking-engineer-agent-rea-autonomous-ai-system-accelerating-meta-ads-ranking-innovation/

[105] METR. 2025. Recent Frontier Models Are Reward Hacking. METR blog, 5 June 2025. https://metr.org/blog/2025-06-05-recent-reward-hacking/

[106] Lingyu Mu, Hao Deng, Haibo Xing, Jinxin Hu, Yu Zhang, and Xiaoyi Zeng. 2026. EvoRec: Self-Evolving Agentic Recommender Systems. arXiv:2606.28368. arXiv:2606.28368 https://arxiv.org/abs/2606.28368

[107] Jaehyun Nam, Jinsung Yoon, Jiefeng Chen, Jinwoo Shin, Sercan Ö. Arık, and Tomas Pfister. 2025. MLE-STAR: Machine Learning Engineering Agent via Search and Targeted Refinement. In *Advances in Neural Information Processing Systems 38*. 129290–129310. arXiv:2506.15692 doi:10.52202/085713-3889

[108] Bharath Sivaram Narasimhan and Karthik R Narasimhan. 2026. τ-Rec: A Verifiable Benchmark for Agentic Recommender Systems. In *Proceedings of the 20th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 915–919. arXiv:2606.10156 doi:10.1145/3773078.3831847

[109] Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, Dieuwke Hupkes, Ricardo Silveira Cabral, Tatiana Shavrina, Jakob Foerster, Yoram Bachrach, William Yang Wang, and Roberta Raileanu. 2025. MLGym: A New Framework and Benchmark for Advancing AI Research Agents. arXiv:2502.14499. arXiv:2502.14499 https://arxiv.org/abs/2502.14499

[110] Minh-Duc Nguyen, Hai-Dang Kieu, and Dung D. Le. 2026. AMEM4Rec: Leveraging Cross-User Similarity for Memory Evolution in Agentic LLM Recommenders. arXiv:2602.08837. arXiv:2602.08837 https://arxiv.org/abs/2602.08837

[111] Jinming Nian, Jyun-Yu Jiang, Henry Tsai, Yi Fang, and Zhirong Liang. 2026. Evolving Semantic Indexes for Generative Retrieval. arXiv:2610.28568. arXiv:2610.28568 https://arxiv.org/abs/2610.28568

[112] Jinming Nian, Fangchen Li, Dae Hoon Park, and Yi Fang. 2026. RankEvolve: Automating the Discovery of Retrieval Algorithms via LLM-Driven Evolution. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, 4028–4033. arXiv:2602.16932 doi:10.1145/3805712.3809962

[113] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. 2025. AlphaEvolve: A coding agent for scientific and algorithmic discovery. arXiv:2506.13131. arXiv:2506.13131 https://arxiv.org/abs/2506.13131

[114] OpenEvolve contributors. 2025. OpenEvolve: An Open-Source Implementation of AlphaEvolve. https://github.com/algorithmicsuperintelligence/openevolve.

[115] Kesha Ou, Chenghao Wu, Xiaolei Wang, Bowen Zheng, Wayne Xin Zhao, Weitao Li, Long Zhang, Sheng Chen, and Ji-Rong Wen. 2026. Deep Research for Recommender Systems. arXiv:2603.07605. arXiv:2603.07605 https://arxiv.org/abs/2603.07605

[116] Enrico Palumbo, Alexandre Tamborrino, Victor Ode, Ben Lacker, Adrià Casas Escoda, Jeremy Hopple, Marcus Better, James Leoni, Hugo Galvão, Hugues Bouchard, Mounia Lalmas, José Luis Redondo García, Abenezer Abebe, Ann Clifton, Anton Blomberg, Henrik Lindström, Dani Doro, and Christine Doig Cardet. 2026. Bootstrapping Conversational Recommendation Agents At Spotify: Synthetic Data Generation and Self-Improvement Loops. In *Proceedings of the 20th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 994–1003. arXiv:2609.30297 doi:10.1145/3773078.3831910

[117] Weidi Pan, He Ma, Shuhao Ye, Palaksh Rungta, David McPeek, Junyi Jiao, Arnab Bhadury, Mingyan Gao, and Onkar Dalal. 2026. RecEvolve: A Knowledge-Driven Autonomous Agent System for Recommender Systems. In *Proceedings of the 20th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 1437–1442. arXiv:2609.01622 doi:10.1145/3773078.3831898 The audit coded the arXiv version.

[118] Rayhan Patel and Shabaz Patel. 2026. EvoRank: LLM-Guided Evolution of Multi-Objective Learning-to-Rank Pipelines. arXiv:2609.22196. arXiv:2609.22196 https://arxiv.org/abs/2609.22196

[119] Bo Peng, Lizhu Zhang, Yuhang Zhou, Mingyi Wang, Yifan Wu, Serena Li, Xiangjun Fan, and Zhuokai Zhao. 2026. Discovered, Not Designed: Population Evolution for Collaborative and Compute-Intensive Model Discovery. arXiv:2610.05950. arXiv:2610.05950 https://arxiv.org/abs/2610.05950

[120] Qiyao Peng, Hongtao Liu, Hua Huang, Jian Yang, Qing Yang, and Minglai Shao. 2025. A Survey on LLM-powered Agents for Recommender Systems. In *Findings of the Association for Computational Linguistics: EMNLP 2025*. 11574–11583. arXiv:2502.10050 doi:10.18653/v1/2025.findings-emnlp.620

[121] Aleksandr Petrov and Craig Macdonald. 2022. A Systematic Review and Replicability Study of BERT4Rec for Sequential Recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 436–447. arXiv:2207.07483 doi:10.1145/3523227.3548487

[122] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H. Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. 2023. Recommender Systems with Generative Retrieval. In *Advances in Neural Information Processing Systems 36*. arXiv:2305.05065 https://arxiv.org/abs/2305.05065

[123] recforge contributors. 2026. recforge: One Protocol to Benchmark and Evolve Recommenders. https://github.com/shangliy/recforge.

[124] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. 2019. Do ImageNet Classifiers Generalize to ImageNet?. In *Proceedings of*

the 36th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research, Vol. 97). PMLR, 5389–5400. arXiv:1902.10811 https://proceedings.mlr.press/v97/recht19a.html

[125] Steffen Rendle. 2019. Evaluation Metrics for Item Recommendation under Sampling. arXiv:1912.02263. arXiv:1912.02263 https://arxiv.org/abs/1912.02263

[126] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*. 452–461. arXiv:1205.2618

[127] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. In *Fourteenth ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 240–248. arXiv:2005.09683 doi:10.1145/3383313.3412488

[128] Carson Rodrigues, Oysturn Vas, Isaiah Abner DCosta, and Nithish Kumar Prabhakaran. 2026. When Is an LLM Worth It for Hyperparameter Optimization? A Budget-Matched Study on Tabular Data Finds the Warm-Start Is a Default Configuration, Not the Model. arXiv:2606.21641. arXiv:2606.21641 https://arxiv.org/abs/2606.21641

[129] Rebecca Roelofs, Vaishaal Shankar, Benjamin Recht, Sara Fridovich-Keil, Moritz Hardt, John Miller, and Ludwig Schmidt. 2019. A Meta-Analysis of Overfitting in Machine Learning. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/hash/ee39e503b6bedf0c98c388b7e8589aca-Abstract.html

[130] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. 2024. Mathematical Discoveries from Program Search with Large Language Models. *Nature* 625, 7995 (2024), 468–475. doi:10.1038/s41586-023-06924-6

[131] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. ACM, 285–295. doi:10.1145/371920.372071

[132] Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. 2025. Agent Laboratory: Using LLM Agents as Research Assistants. arXiv:2501.04227. arXiv:2501.04227 https://arxiv.org/abs/2501.04227

[133] Faisal Shehzad and Dietmar Jannach. 2023. Everyone's a Winner! On Hyperparameter Tuning of Recommendation Models. In *Proceedings of the 17th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 652–657. doi:10.1145/3604915.3609488

[134] Xiang Shen, Yuhang Zhou, Yifan Wu, Zhuokai Zhao, Siyu Lin, Lei Huang, Qianqian Zhong, Lizhu Zhang, Benyu Zhang, Xiangjun Fan, and Hong Yan. 2026. Agentic Recommender System with Hierarchical Belief-State Memory. arXiv:2605.14401. arXiv:2605.14401 https://arxiv.org/abs/2605.14401

[135] Zhan Shi and Shanglin Yang. 2026. Selective LLM-Guided Regularization for Enhancing Recommendation Models. In *Companion Proceedings of the 19th ACM International Conference on Web Search and Data Mining*. 148–153. doi:10.1145/3779211.3795736 GenAIRecP Workshop; arXiv:2512.21526.

[136] Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. 2022. Defining and Characterizing Reward Hacking. In *Advances in Neural Information Processing Systems*, Vol. 35. Curran Associates, Inc., 9460–9471. arXiv:2209.13085 doi:10.52202/068431-0687

[137] Blaž Škrlj, Benoît Guilleminot, and Andraž Tori. 2025. Agent0: Leveraging LLM Agents to Discover Multi-value Features from Text for Enhanced Recommendations. arXiv:2507.18993. arXiv:2507.18993 https://arxiv.org/abs/2507.18993

[138] Qingquan Song, Dehua Cheng, Hanning Zhou, Jiyan Yang, Yuandong Tian, and Xia Hu. 2020. Towards Automated Neural Interaction Discovery for Click-Through Rate Prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, New York, NY, USA, 945–955. arXiv:2007.06434 doi:10.1145/3394486.3403137

[139] Dhruv Srikanth, Bingchen Zhao, Dixing Xu, Yuxiang Wu, and Zhengyao Jiang. 2026. Recursive self-improvement of AI research agents. arXiv:2609.26457. arXiv:2609.26457 https://arxiv.org/abs/2609.26457

[140] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. arXiv:1904.06690. arXiv:1904.06690 https://arxiv.org/abs/1904.06690

[141] Zhu Sun, Hui Fang, Jie Yang, Xinghua Qu, Hongyang Liu, Di Yu, Yew-Soon Ong, and Jie Zhang. 2023. DaisyRec 2.0: Benchmarking Recommendation for Rigorous Evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 7 (2023), 8206–8226. arXiv:2206.10848 doi:10.1109/tpami.2022.3231891

[142] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. 2020. Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison. In *Proceedings of the 14th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 23–32. doi:10.1145/3383313.3412489

[143] Zhen Tao, Riwei Lai, Chenyun Yu, Weixin Chen, Li Chen, Beibei Kong, Lei Cheng, Chengxiang Zhuo, Zang Li, and Qingqiang Sun. 2026. SAGER: Self-Evolving User Policy Skills for Recommendation Agent. arXiv:2604.14972. arXiv:2604.14972 https://arxiv.org/abs/2604.14972

[144] Himanshu Thakur, Anusha Kamath, Anurag Muthyala, Dhwani Sanmukhani, Smruthi Mukund, and Jay Katukuri. 2026. Towards Reliable ML Feature Engineering via Planning in Constrained-Topology of LLM Agents. arXiv:2601.10820. arXiv:2601.10820 https://arxiv.org/abs/2601.10820

[145] Edan Toledo, Karen Hambardzumyan, Martin Josifoski, Rishi Hazra, Nicolas Baldwin, Alexis Audran-Reiss, Michael Kuchnik, Despoina Magka, Minqi Jiang, Alisia Maria Lupidi, Andrei Lupu, Roberta Raileanu, Kelvin Niu, Tatiana Shavrina, Jean-Christophe Gagnon-Audet, Michael Shvartsman, Shagun Sodhani, Alexander H. Miller, Abhishek Charnalia, Derek Dunfield, Carole-Jean Wu, Pontus Stenetorp, Nicola Cancedda, Jakob Nicolaus Foerster, and Yoram Bachrach. 2025. AI Research Agents for Machine Learning: Search, Exploration, and Generalization in MLE-bench. In *Advances in Neural Information Processing Systems 38*. 39717–39756. arXiv:2507.02554 doi:10.52202/085713-1186

[146] Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. 2024. AutoML-Agent: A Multi-Agent LLM Framework for Full-Pipeline AutoML. arXiv:2410.02958. arXiv:2410.02958 https://arxiv.org/abs/2410.02958

[147] Chenglong Wang, Canjia Li, Xingzhao Zhu, Yifu Huo, Huiyu Wang, Weixiong Lin, Yun Yang, Qiaozhi He, Tianhua Zhou, Xiaojia Chang, Jingbo Zhu, and Tong Xiao. 2026. SERM: Self-Evolving Relevance Model with Agent-Driven Learning from Massive Query Streams. arXiv:2601.09515. arXiv:2601.09515 https://arxiv.org/abs/2601.09515

[148] Haochen Wang, Yi Wu, Daryl Chang, Li Wei, and Lukasz Heldt. 2026. Self-Evolving Recommendation System: End-To-End Autonomous Model Optimization With LLM Agents. In *Proceedings of the 20th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 1462–1470. arXiv:2602.10226 doi:10.1145/3773078.3831919 The audit coded the arXiv version.

[149] Lei Wang, Jingsen Zhang, Hao Yang, Zhi-Yuan Chen, Jiakai Tang, Zeyu Zhang, Xu Chen, Yankai Lin, Hao Sun, Ruihua Song, Wayne Xin Zhao, Jun Xu, Zhicheng Dou, Jun Wang, and Ji-Rong Wen. 2025. User Behavior Simulation with Large Language Model-based Agents. *ACM Transactions on Information Systems* 43, 2 (2025), 1–37. arXiv:2306.02552 doi:10.1145/3708985

[150] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 165–174. doi:10.1145/3331184.3331267

[151] Yifan Wang, Shipeng Zhu, Fei Xiong, Yuqin Yang, Yonghui Huang, Kunyao Wu, Yue Wang, Weichao Meng, and Yu Gong. 2026. PEAR: Progressive Evidence-Based AutoResearch for Industrial Search Systems. arXiv:2609.35031. arXiv:2609.35031 https://arxiv.org/abs/2609.35031

[152] Ziyan Wang, Yingpeng Du, Zhu Sun, Haoyan Chua, Kaidong Feng, Wenya Wang, and Jie Zhang. 2024. Re2LLM: Reflective Reinforcement Large Language Model for Session-based Recommendation. arXiv:2403.16427. arXiv:2403.16427 https://arxiv.org/abs/2403.16427

[153] Zongwei Wang, Min Gao, Hongzhi Yin, Junliang Yu, Tong Chen, Quoc Viet Hung Nguyen, Shazia Sadiq, and Tianrui Li. 2026. Self-Distilled Reinforcement Learning for Co-Evolving Agentic Recommender Systems. arXiv:2604.10029. arXiv:2604.10029 https://arxiv.org/abs/2604.10029

[154] Zongwei Wang, Min Gao, Junliang Yu, Yupeng Hou, Shazia Sadiq, and Hongzhi Yin. 2025. RuleAgent: Discovering Rules for Recommendation Denoising with Autonomous Language Agents. arXiv:2503.23374. arXiv:2503.23374 https://arxiv.org/abs/2503.23374

[155] Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Josh Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Holden Karnofsky, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. 2024. RE-Bench: Evaluating frontier AI R&D capabilities of language model agents against human experts. arXiv:2411.15114. arXiv:2411.15114 https://arxiv.org/abs/2411.15114

[156] Sherman Wong, Zhenting Qi, Zhaodong Wang, Nathan Hu, Samuel Lin, Jun Ge, Erwin Gao, Wenlin Chen, Yilun Du, Minlan Yu, and Ying Zhang. 2025. Confucius Code Agent: Scalable Agent Scaffolding for Real-World Codebases. arXiv:2512.10398. arXiv:2512.10398 https://arxiv.org/abs/2512.10398

[157] Chaowei Wu, Huazhu Chen, Congde Yuan, Qirui Yang, Guoqing Song, Yue Gao, Li Luo, Frank Youhua Chen, and Mengzhuo Guo. 2026. AgentLTV: An Agent-Based Unified Search-and-Evolution Framework for Automated Lifetime Value Prediction. arXiv:2602.21634. arXiv:2602.21634 https://arxiv.org/abs/2602.21634

[158] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, and Ming Zhou. 2020. MIND: A Large-scale Dataset for News Recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 3597–3606. doi:10.18653/v1/2020.acl-main.331

[159] Jie JW Wu, Ayanda Patrick Herlihy, Ahmad Saleem Mirza, Ali Afoud, and Fatemeh Fard. 2025. GrowthHacker: Automated Off-Policy Evaluation Optimization Using Code-Modifying LLM Agents. arXiv:2511.00802. arXiv:2511.00802

https://arxiv.org/abs/2511.00802

[160] Liwei Wu and Cho-Jui Hsieh. 2026. Closing the Auto-Research Loop: An AI Co-Scientist for Production Search Ranking. arXiv:2603.22376. arXiv:2603.22376 https://arxiv.org/abs/2603.22376

[161] Xidong Wu, Yue Zhuan, Ruoqiao Wei, Hangxin Chen, Di Bai, Jintao Liu, Xinyi Wang, Xue Wang, Luoshu Wang, and Xinwu Cheng. 2026. AgenticRecTune: Multi-Agent with Self-Evolving Skillhub for Recommendation System Optimization. arXiv:2604.26969. arXiv:2604.26969 https://arxiv.org/abs/2604.26969

[162] Zhouhang Xie, Bo Peng, Zhankui He, Ziqi Chen, Alice Han, Isabella Ye, Benjamin Coleman, Noveen Sachdeva, Fernando Pereira, Julian McAuley, Wang-Cheng Kang, Derek Zhiyuan Cheng, Beidou Wang, and Randolph Brown. 2026. AgenticTagger: Structured Item Representation for Recommendation with LLM Agents. arXiv:2602.05945. arXiv:2602.05945 https://arxiv.org/abs/2602.05945

[163] Jinfeng Xu, Zheyu Chen, Ziyue Peng, Zheng Lin, Wenhao Yuan, Jian Chen, Shujie Li, and Edith Ngai. 2026. Beyond Successor Accuracy: State Retention for Recursive Self-Improvement in Recommendation. arXiv:2610.07105. arXiv:2610.07105 https://arxiv.org/abs/2610.07105

[164] Ruiyao Xu, Yanfei Chen, Zhongying CuiZhu, Bhavana Dalvi Mishra, Yifei Ming, Han Yu, Rujun Han, Chen-Yu Lee, and Tomas Pfister. 2026. Harness^2: Recursive Agent Harnessing for an Open World. arXiv:2610.14224. arXiv:2610.14224 https://arxiv.org/abs/2610.14224

[165] Yiqiao Xu, Yunqi Mi, Yingjie Wu, et al. 2026. From Scoring to Workflow: A Survey of Agentic Recommender Systems. *Information Processing & Management* 64, 2 (2026), 105164. doi:10.1016/j.ipm.2026.105164

[166] Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. 2025. The AI Scientist-v2: Workshop-Level Automated Scientific Discovery via Agentic Tree Search. arXiv:2504.08066. arXiv:2504.08066 https://arxiv.org/abs/2504.08066

[167] Antoine Yang, Pedro M. Esperança, and Fabio M. Carlucci. 2020. NAS Evaluation is Frustratingly Hard. In *International Conference on Learning Representations (ICLR)*. arXiv:1912.12522 https://arxiv.org/abs/1912.12522

[168] Hanchen Yang, Kaiwen Yang, Junpeng Zhuang, Yang He, Keting Cen, Bochao Liu, Zhongbo Sun, An Liu, Zhongteng Han, and Chenyi Lei. 2026. SR-Agent: An Experience-Driven Agentic Framework for Post-Ranking Strategy Refinement in E-Commerce Recommendation. arXiv:2607.17719. arXiv:2607.17719 https://arxiv.org/abs/2607.17719

[169] Shanglin Yang. 2026. ideation-goodhart: What Should You Optimize When You Optimize an LLM Research-Ideation Harness? https://github.com/shangliy/ideation-goodhart. Code, pre-registrations and report.

[170] Shuang Yang, Zijie Zhuang, Changxin Lao, Pengbo Xu, Hanwen Xu, Yusheng Huang, Han Gao, Guanchen Wang, Tianbao Ma, Linxun Chen, Peilin Song, Xuming Wang, Chen Li, Fan Wu, Tao Wang, Zibo Zhao, Xiangyu Wu, An Liu, Fei Pan, Peng Jiang, Chen Yang, Zhaojie Liu, and Wenwu Ou. 2026. Advancing Model Research in AgentX: Long-Horizon Autonomy for Industrial Recommender Systems. arXiv:2609.30001. arXiv:2609.30001 https://arxiv.org/abs/2609.30001

[171] Xu Yang, Xiao Yang, Shikai Fang, Yifei Zhang, Jian Wang, Bowen Xian, Qizheng Li, Jingyuan Li, Minrui Xu, Yuante Li, Haoran Pan, Yuge Zhang, Weiqing Liu, Yelong Shen, Weizhu Chen, and Jiang Bian. 2025. R&D-Agent: An LLM-Agent Framework Towards Autonomous Data Science. arXiv:2505.14738. arXiv:2505.14738 https://arxiv.org/abs/2505.14738

[172] Kehui Yao, Chuanbo Hu, Jiao Chen, Ramin Giahi, Reza Yousefi Maragheh, Lalitesh Morishetti, Ziwei Hong, Yiyun Su, Jianpeng Xu, Evren Korpeoglu, and Kannan Achan. 2026. An Explicit Feedback Contract for Iterative ML-Agent Harnesses. arXiv:2610.28798. arXiv:2610.28798 https://arxiv.org/abs/2610.28798

[173] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. 2019. NAS-Bench-101: Towards Reproducible Neural Architecture Search. In *Proceedings of the 36th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 7105–7114. arXiv:1902.09635 https://proceedings.mlr.press/v97/ying19a.html

[174] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. 2020. Evaluating the Search Phase of Neural Architecture Search. In *International Conference on Learning Representations (ICLR)*. arXiv:1902.08142 https://arxiv.org/abs/1902.08142

[175] Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. 2023. Where to Go Next for Recommender Systems? ID- vs. Modality-based Recommender Models Revisited. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2639–2649. doi:10.1145/3539618.3591932

[176] Longfei Yun, Yihan Wu, Haoran Liu, Xiaoxuan Liu, Ziyun Xu, Yi Wang, Yang Xia, Pengfei Wang, Mingze Gao, Yunxiang Wang, Changfan Chen, Fei Liu, Akos Lada, Wenjie Fu, Hong Yan, and Junfeng Pan. 2026. Decoding ML Decision: An Agentic Reasoning Framework for Large-Scale Ranking System. arXiv:2602.18640. arXiv:2602.18640 https://arxiv.org/abs/2602.18640

[177] Wenwen Zeng, Jinhui Zhang, Hao Chen, Zhaoyu Hu, Yongqi Liang, Jiajun Chai, Dengcan Liu, Zhenfeng Liu, Shurui Yan, Minglong Xue, Xiaohan Wang, Wei Lin, and Guojun Yin. 2026. RecRM-Bench: Benchmarking Multidimensional Reward Modeling for Agentic Recommender Systems. arXiv:2605.11874. arXiv:2605.11874 https://arxiv.org/abs/2605.11874

[178] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, Yinghai Lu, and Yu Shi. 2024. Actions Speak Louder than Words: Trillion-Parameter Sequential Transducers for Generative Recommendations. In *Proceedings of the 41st International Conference on Machine Learning (ICML), PMLR 235*. arXiv:2402.17152 https://arxiv.org/abs/2402.17152

[179] An Zhang, Yuxin Chen, Leheng Sheng, Xiang Wang, and Tat-Seng Chua. 2024. On Generative Agents in Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, 1807–1817. arXiv:2310.10108 doi:10.1145/3626772.3657844

[180] Bin Zhang, Bowen Zheng, Chao Yi, Chengyu Lai, Dian Chen, Dimin Wang, Gaoyang Guo, Jialin Zhu, Jian Wu, Jing Yu, Jiuning Lin, Lingqing Zhang, Lingyun Zheng, Mao Zhang, Mingming Pan, Ruiquan Lan, Shuai Zhong, Wen Chen, Wendong Zhang, Xiaodong Zhu, Xuan Chen, Xunke Xi, Yifan Lu, Yiheng Wang, Yue Zeng, Yujie Luo, Yuning Jiang, Zhe Hu, Zhibo Xiao, Zihong Huang, Binbin Cao, Bo Zheng, Danning Wang, Dixuan Wang, Ge Fan, Haixia Wu, Han Zhu, Hao Fang, Haoming Chen, Huiping Chu, Jian Wang, Jianjun Wu, Jiawei Wu, Jiaxin Yu, Jingwen Liu, Jinzhe Shan, Kai Meng, Kai Zhang, Keqin Xu, Kewei Zhu, Lang Tian, Leihui Chen, Li Chen, Licheng Xu, Lide Xiao, Ruitong Zhang, Shiyao Peng, Silu Zhou, Tao Wang, Wei Shi, Wenjun Yang, Xiang Chen, Xiang Gao, Xiao Ren, Xu Liu, Xuwen Wang, Yang Li, Yeqiu Yang, Yi Hu, Yichen Yuan, Yinnan Song, Yipeng Yu, Yuan Liu, Yunqi Gao, Zhiliang Huang, Zhujin Gao, and Zongyuan Wu. 2026. DREAM Technical Report. arXiv:2608.09408. arXiv:2608.09408 https://arxiv.org/abs/2608.09408

[181] Hao Zhang, Jinxin Hu, Hao Deng, Lingyu Mu, Shizhun Wang, Yu Zhang, and Xiaoyi Zeng. 2026. AgenticRS-Architecture: System Design for Agentic Recommender Systems. arXiv:2603.26085. arXiv:2603.26085 https://arxiv.org/abs/2603.26085

[182] Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2024. AgentCF: Collaborative Learning with Autonomous Language Agents for Recommender Systems. In *Proceedings of the ACM Web Conference 2024*. ACM, New York, NY, USA, 3679–3689. arXiv:2310.09233 doi:10.1145/3589334.3645537

[183] Luankang Zhang, Hao Wang, Zhongzhou Liu, Mingjia Yin, Yonghao Huang, Jiaqi Li, Wei Guo, Yong Liu, Huifeng Guo, Defu Lian, and Enhong Chen. 2026. Can Recommender Systems Teach Themselves? A Recursive Self-Improving Framework with Fidelity Control. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*. arXiv:2602.15659 https://arxiv.org/abs/2602.15659

[184] Liulei Zhang, Dejing Zhou, Chuyue Huang, Guanhua Chen, Yutong Yao, Lidia S. Chao, Chi Man Vong, and Derek F. Wong. 2026. How Is Automated Research Evaluated? A Survey of Benchmarks and Evaluation Practices. arXiv:2610.11877. arXiv:2610.11877 https://arxiv.org/abs/2610.11877

[185] Michael R. Zhang, Nishkrit Desai, Juhan Bae, Jonathan Lorraine, and Jimmy Ba. 2023. Using Large Language Models for Hyperparameter Optimization. arXiv:2312.04528. arXiv:2312.04528 https://arxiv.org/abs/2312.04528

[186] Qi Zhang, Yanlin Chen, and Wenchao Xiao. 2026. AutoLR: Automating the Path from Research to Launch Review in Industrial Recommender Systems. arXiv:2609.04871. arXiv:2609.04871 https://arxiv.org/abs/2609.04871

[187] Sheng Zhang, Maolin Wang, Yao Zhao, Chenyi Zhuang, Jinjie Gu, Ruocheng Guo, Xiangyu Zhao, Zijian Zhang, and Hongzhi Yin. 2024. DNS-Rec: Data-aware Neural Architecture Search for Recommender Systems. In *18th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 591–600. arXiv:2402.00390 doi:10.1145/3640457.3688117

[188] Tunhou Zhang, Dehua Cheng, Yuchen He, Zhengxing Chen, Xiaoliang Dai, Liang Xiong, Feng Yan, Hai Li, Yiran Chen, and Wei Wen. 2023. NASRec: Weight Sharing Neural Architecture Search for Recommender Systems. In *Proceedings of the ACM Web Conference 2023*. ACM, New York, NY, USA, 1199–1207. arXiv:2207.07187 doi:10.1145/3543507.3583446

[189] Wei Zhang, Hongji Li, Song Sun, Peng Yu, Xue Yang, Lei Zhao, and Peng Jiang. 2026. DMRL: Document-Mediated Reinforcement Learning for Skill Optimization in Advertising Recommendation. arXiv:2609.02170. arXiv:2609.02170 https://arxiv.org/abs/2609.02170

[190] Yunxiang Zhang, Muhammad Khalifa, Shitanshu Bhushan, Grant D Murphy, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee, Honglak Lee, and Lu Wang. 2025. MLRC-Bench: Can Language Agents Solve Machine Learning Research Challenges?. In *Advances in Neural Information Processing Systems 38*. 100801–100845. arXiv:2504.09702 doi:10.52202/085713-3036

[191] Yu Zhang, Shutong Qiao, Jiaqi Zhang, Tzu-Heng Lin, Chen Gao, and Yong Li. 2026. A Survey of Large Language Model Empowered Agents for Recommendation and Search: Towards Next-Generation Information Retrieval. *ACM Transactions on Information Systems*, Article 3842663 (2026). arXiv:2503.05659 doi:10.1145/3842663

[192] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. 2021. RecBole: Towards a Unified, Comprehensive and

Efficient Framework for Recommendation Algorithms. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. ACM, New York, NY, USA, 4653–4664. arXiv:2011.01731 doi:10.1145/3459637.3482016

[193] Xiangyu Zhao, Haochen Liu, Wenqi Fan, Hui Liu, Jiliang Tang, Chong Wang, Ming Chen, Xudong Zheng, Xiaobing Liu, and Xiwang Yang. 2021. AutoEmb: Automated Embedding Dimensionality Search in Streaming Recommendations. In *2021 IEEE International Conference on Data Mining (ICDM)*. 896–905. arXiv:2002.11252 doi:10.1109/icdm51629.2021.00101

[194] Xiangyu Zhao, Haochen Liu, Hui Liu, Jiliang Tang, Weiwei Guo, Jun Shi, Sida Wang, Huiji Gao, and Bo Long. 2020. Memory-efficient Embedding for Recommendations. arXiv:2006.14827. arXiv:2006.14827 https://arxiv.org/abs/2006.14827

[195] Ruiqi Zheng, Liang Qu, Bin Cui, Yuhui Shi, and Hongzhi Yin. 2023. AutoML for Deep Recommender Systems: A Survey. *ACM Transactions on Information Systems*, Article 3579355 (2023). arXiv:2203.13922 doi:10.1145/3579355

[196] Ziqian Zhong, Aditi Raghunathan, and Nicholas Carlini. 2026. ImpossibleBench: Measuring LLMs' Propensity of Exploiting Test Cases. In *International Conference on Learning Representations (ICLR)*. arXiv:2510.20270 https://openreview.net/forum?id=SeO4vyAj7E

[197] Guorui Zhou, Honghui Bao, Jiaming Huang, Jiaxin Deng, Jinghao Zhang, Junda She, Kuo Cai, Lejian Ren, Lu Ren, Qiang Luo, Qianqian Wang, Qigen Hu, Rongzhou Zhang, Ruiming Tang, Shiyao Wang, Wuchao Li, Xiangyu Wu, Xinchen Luo, Xingmei Wang, Yifei Hu, Yunfan Wu, Zhanyu Liu, Zhiyang Zhang, Zixing Zhang, Bo Chen, Bin Wen, Chaoyi Ma, Chengru Song, Chenglong Chu, Defu Lian, Fan Yang, Feng Jiang, Hongtao Cheng, Huanjie Wang, Kun Gai, Pengfei Zheng, Qiang Wang, Rui Huang, Siyang Mao, Tingting Gao, Wei Yuan, Yan Wang, Yang Zhou, Yi Su, Zexuan Cheng, Zhixin Ling, and Ziming Li. 2025. OpenOneRec Technical Report. arXiv:2512.24762. arXiv:2512.24762 https://arxiv.org/abs/2512.24762

[198] Guorui Zhou, Jiaxin Deng, Jinghao Zhang, Kuo Cai, Lejian Ren, Qiang Luo, Qianqian Wang, Qigen Hu, Rui Huang, Shiyao Wang, Weifeng Ding, Wuchao Li, Xinchen Luo, Xingmei Wang, Zexuan Cheng, Zixing Zhang, Bin Zhang, Boxuan Wang, Chaoyi Ma, Chengru Song, Chenhui Wang, Di Wang, Dongxue Meng, Fan Yang, Fangyu Zhang, Feng Jiang, Fuxing Zhang, Gang Wang, Guowang Zhang, Han Li, Hengrui Hu, Hezheng Lin, Hongtao Cheng, Hongyang Cao, Huanjie Wang, Jiaming Huang, Jiapeng Chen, Jiaqiang Liu, Jinghui Jia, Kun Gai, Lantao Hu, Liang Zeng, Liao Yu, Qiang Wang, Qidong Zhou, Shengzhe Wang, Shihui He, Shuang Yang, Shujie Yang, Sui Huang, Tao Wu, Tiantian He, Tingting Gao, Wei Yuan, Xiao Liang, Xiaoxiao Xu, Xugang Liu, Yan Wang, Yi Wang, Yiwu Liu, Yue Song, Yufei Zhang, Yunfan Wu, Yunfeng Zhao, and Zhanyu Liu. 2025. OneRec Technical Report. arXiv:2506.13695. arXiv:2506.13695 https://arxiv.org/abs/2506.13695

[199] Jieming Zhu, Quanyu Dai, Liangcai Su, Rong Ma, Jinyang Liu, Guohao Cai, Xi Xiao, and Rui Zhang. 2022. BARS: Towards Open Benchmarking for Recommender Systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, 2912–2923. arXiv:2205.09626 doi:10.1145/3477495.3531723

[200] Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. 2021. Open Benchmarking for Click-Through Rate Prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. ACM, New York, NY, USA, 2759–2769. arXiv:2009.05794 doi:10.1145/3459637.3482486

[201] Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun, Andy Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, Fazl Barez, Rahul Gupta, Jwala Dhamala, Jacob Merizian, Mario Giulianelli, Harry Coppock, Cozmin Ududec, Jasjeet Sekhon, Jacob Steinhardt, Antony Kellermann, Sarah Schwettmann, Matei Zaharia, Ion Stoica, Percy Liang, and Daniel Kang. 2025. Establishing Best Practices in Building Rigorous Agentic Benchmarks. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*. arXiv:2507.02825 https://openreview.net/forum?id=E58HNCqoaA

[202] Zijie Zhuang, Changxin Lao, Pengbo Xu, Hanwen Xu, Ruochen Yang, Yingzhi He, Peng Zhang, Jiangxia Cao, Yusheng Huang, Guohong Mu, Jian Liang, Ruiming Tang, Shuang Yang, Zhaojie Liu, Wenwu Ou, and Kun Gai. 2026. From Trajectories to Evidence: Auditable Experimental Records for Industrial Research Agents. arXiv:2608.05235. arXiv:2608.05235 https://arxiv.org/abs/2608.05235

[203] Zhuang Zhuang, Zhipeng Wei, Rongfeng Guo, Shijie Li, Peng Zhao, Jie Chen, and Fei Pan. 2026. EvoReason: Self-Evolving Reasoning Primitive-Guided On-Policy Distillation for Latent Reasoning in Generative Recommendation. arXiv:2607.29010. arXiv:2607.29010 https://arxiv.org/abs/2607.29010

A The Minimum Tier Applied to Two Papers

Table 7 applies the minimum tier of Table 6 to an open recommender harness (RecHarness) and to the comparison point that meets the most criteria (EvoRank), using only the audit evidence. It shows that the standard can be checked from a paper's text, and that a study can meet nearly all of it at a cost of about ten dollars of LLM calls per run.

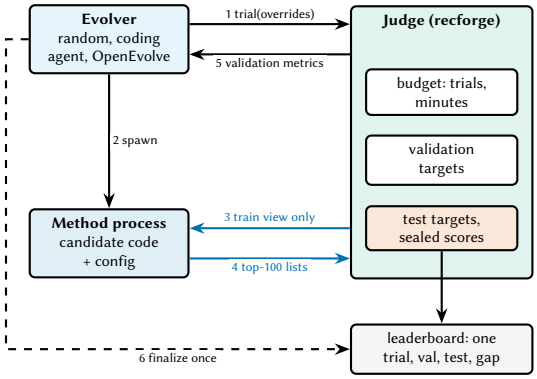


Figure 4: An evolve session in recforge. The evolver requests trials (1); each trial runs the candidate in its own process (2), which receives a training view without targets (3) and returns ranked lists (4). Only validation metrics flow back (5); test scores are computed and sealed. One `finalize` publishes a single trial with its validation score, test score, gap and budget (6). The seal holds at the API level; file-system isolation is planned.

Table 7: Minimum-tier compliance of two audited papers, from the released evidence. Y met, P partly met, N not met, – not applicable.

	RecHarness	EvoRank	Note
R1	Y	Y	99 sampled negatives stated; per-query full lists
R2	Y	Y	both select on a validation split
R3	P	Y	RecHarness tests once but reports no validation score beside it
R4	P	Y	three runs in both; RecHarness reports only their mean
R5	P	Y	RecHarness's random arm router keeps the LLM edits; EvoRank runs equal-budget random search
R6	P	P	GPU-time budget, but the LLM is not named; EvoRank gives dollars but not the LLM
R7	N	Y	cold-start template vs. Optuna-tuned LambdaMART
R8	P	Y	isolated workspaces vs. leakage-guarded helpers and a pre-search audit
R9	P	–	7-day A/B on 10% of traffic; significance not stated

B recforge: A Reference Implementation

recforge [123] is an open benchmark that we are building to enforce the parts of the standard that software can enforce. It is under development, it has produced no evolver results, and the audit does not depend on it. Its design follows the audit (Figure 4).

- *One protocol per version.* The current protocol is leave-one-out per user (last item test, second-to-last validation) with full-catalogue ranking, the user's history removed, and HR and NDCG at 10 and 20 and MRR@100 (items ranked below 100 score zero). Leave-one-out ignores the global timeline and can leak future information [48, 58]; a global temporal split is the next protocol version. Results from different versions are never put in one table, and every result is keyed by the source hash of the method, its configuration and the code commit.
- *A sealed test at the API level.* Method code runs in its own process and receives only a training view; asking for a validation or test target raises an error, and the method returns ranked lists that the parent process scores. The seal holds only against code that uses the API: sealed scores and session state are ordinary files, so an agent with file-system access could read test scores or alter the budget. A container mode that would close this gap (R8) is planned, not implemented.
- *Budgeted evolve sessions.* A session has a trial and time budget. Each trial returns validation metrics only; test metrics are computed and sealed. A single `finalize` reveals the test metrics of exactly one trial, publishes it with the session's budget, and closes the session.
- *Evolvers on equal terms.* An equal-budget random search (the R5 null), a wrapper for an external coding agent, and an OpenEvolve adapter drive sessions through the same interface, and each run reports the validation-to-test gap of its finalized trial. They are tested on synthetic data only; the OpenEvolve adapter has not been run end to end.
- *Baselines and data.* Popularity, ItemKNN [131], BPR [126], SASRec [61], HSTU [178] and TIGER [122], with MovieLens-1M [44], Amazon 2014 Beauty, Sports and Toys [102], MIND news [158], and the OpenOneRec release [197].

Not yet implemented: the container mode, recording of LLM tokens and cost, automatic aggregation of repeated sessions, and the temporal split.