Continuous Improvement and Parallel Autonomous Exploration: An LLM-Agent Framework for Searching Large Solution Spaces

Dulmini Hettiarachchi
Mercari, Inc.
Japan
dulminih@mercari.com

Andre Rusli
Mercari, Inc.
Japan
andre.rusli@mercari.com

Julio Christian Young
Mercari, Inc.
Japan
jc.young@mercari.com

Sho Akiyama
Mercari, Inc.
Japan
s-akiyama@mercari.com

Abstract

We present a *framework* that gives LLM agents two mechanisms for searching large solution spaces autonomously. First, a leaderboard scored on held-out data acts as a reward signal that drives each agent to refine its solutions over repeated submissions, a loop that operates even with a single agent. Second, the framework enables running many agents in parallel, fully autonomously, with *no human in the loop*: agents independently analyze, survey methods, implement, self-evaluate, submit, and revise, while a moderator agent handles only logistics. Running agents in parallel under the shared reward broadens the explored region of the solution space rather than refining the single seeded paradigm. We instantiate the framework on product-to-catalog matching (a core e-commerce retrieval task with a large, category-structured solution space), posed as selective prediction with a precision-coverage operating point. A single agent refines within its seeded paradigm, whereas parallel autonomous agents surface qualitatively different solutions. On this testbed, best qualified coverage (≥95% P@1 per category) reaches 47.8–57.4% with a single agent and 62.8–69.4% with five, against a 33.3% baseline. Our contribution is the framework itself: a continuous-improvement reward loop and a substrate for fully autonomous parallel exploration, backed by case-study evidence.

CCS Concepts

• **Computing methodologies** → **Natural language processing**; **Multi-agent systems**.

Keywords

multi-agent systems, LLM agents, autonomous exploration, continuous improvement, retrieval optimization, selective prediction

1 Introduction

Large language models (LLMs) have enabled autonomous agents that write code, run experiments, and iterate on results with minimal human intervention [24]. Many practical optimization problems have a large solution space and often require task-specific optimization strategies.

We present a framework that gives LLM agents two mechanisms for searching such spaces autonomously. (i) A withheld-test *leaderboard* acts as a reward signal: each submission is scored on data the agent never sees, and the returned score drives the agent to improve its submissions over successive rounds, a loop that operates even with a single agent. (ii) The framework enables running *many agents in parallel, fully autonomously, with no human in the loop*: agents independently analyze data, survey candidate methods, hypothesize, implement, self-evaluate, submit, and revise; the only shared signal is the leaderboard, and a logistics-only moderator. Running agents in parallel under the shared reward broadens the explored region of the solution space.

We instantiate the framework on product-to-catalog matching in a large consumer-to-consumer (C2C) marketplace, a core e-commerce retrieval task. Unlike catalog-fed B2C, C2C listings are individual-seller free text with no structured stock-keeping unit (SKU) feed. Mapping each listing to its catalog SKU therefore underpins search, recommendation, deduplication, and price guidance, making the task both harder and more business-critical. Given a user listing (title, description, metadata), the system must identify the correct catalog SKU from a catalog of up to 33K SKUs. We pose this as a *selective prediction* problem [7]: systems may abstain on low-confidence listings to achieve high precision on the remainder. The practical goal, automating only high-confidence predictions, creates a precision-coverage tradeoff: maximize coverage subject to ≥95% Precision@1 on every category independently. Because different categories reward different approaches, no single method dominates; this large, category-structured solution space makes product-to-catalog matching a natural testbed for broad autonomous exploration.

Prior multi-agent LLM work has focused on collaborative or communicative architectures: MetaGPT [9] and ChatDev [21] coordinate agents through structured workflows or chains, while CAMEL [14] uses role-playing cooperation. We instead study agents that run independently and in parallel under a shared leaderboard reward, sharing only scores and high-level approach descriptions. While MLAgentBench [10] evaluates individual agent capability on ML tasks, our framework examines how a reward-driven loop plus parallel autonomous exploration searches a large space.

Our key contributions:

(1) **Framework**: we define a continuous-improvement reward loop (a withheld-test leaderboard that scores each submission and drives every agent to iteratively improve its solutions) plus a substrate for *fully autonomous parallel exploration* of large search spaces.

(2) **Autonomy**: we show the framework runs end-to-end with *no human in the loop* (no human prescribes technical approaches; the moderator is automated and handles only logistics), and that parallel agents under the shared reward broaden exploration beyond a single agent's seeded paradigm.
(3) **Case-study evidence**: on product-to-catalog matching, we report a finding that replicates across runs and is independent of agent count: a seeded single agent iterates narrowly within its paradigm while parallel autonomous agents escape it; the coverage figures are descriptive ranges, not a causal scaling law.

2 Related Work

LLM Agents for Scientific Discovery. Autonomous LLM agents drive scientific discovery (The AI Scientist [18], ChemCrow [2], Coscientist [1]) and mathematical search (FunSearch [22]), and are benchmarked individually on ML and software tasks (MLAgentBench [10], SWE-bench [11]). We complement these by studying how parallel autonomous agents collectively explore an optimization landscape under a shared reward.

Multi-Agent LLM Frameworks. AutoGen [25] provides conversational multi-agent coordination. MetaGPT [9] encodes SOPs into role-based pipelines, ChatDev [21] coordinates agents through chat chains, and AgentVerse [5] studies emergent group behaviors. Debate-based methods [6] improve factuality through argumentation. These are primarily *collaborative* designs. CompeteAI [28] studies competition among LLM agents in a social-simulation setting; we instead use a shared leaderboard as a reward signal for parallel agents' autonomous *algorithmic* exploration of a technical search space.

Iterative Agent Optimization. ReAct [27] interleaves reasoning with tool use, and Reflexion [23] introduces verbal reinforcement learning through self-reflection, directly analogous to our iterative submission-feedback loop. OPRO [26] treats LLMs as optimizers. Agent-K [8], which uses the Kaggle leaderboard as a reward signal for iterative improvement, is the closest single-agent analogue to our first mechanism; AutoKaggle [17] similarly optimizes single-pipeline ML workflows, and MLE-bench [3] benchmarks such agents. We extend this withheld-test reward loop to parallel agents, where solution diversity emerges from a shared reward rather than an explicit diversity-preservation operator. CoMind [15] is the closest multi-agent analogue; key differences are local training-proxy rewards (versus withheld-test scores), a shared community pool of code notebooks and discussions (versus scores and approach descriptions only), and a specialized-role pipeline (Coordinator, Analyzer, Idea Proposer, Evaluator) rather than independent generalist agents each running the full research loop. This contrasts with quality-diversity search [19] and FunSearch's [22] island populations.

Product Matching and E-Commerce Retrieval. DeepMatcher [20] established deep learning baselines for entity matching; Ditto [16] showed fine-tuned pre-trained LMs can outperform prior entity-matching systems. Dense passage retrieval [13] popularized the dense bi-encoder retrieval paradigm used by models like BGE-M3 [4], with FAISS [12] providing nearest-neighbor infrastructure. Selective prediction [7] underpins our precision-coverage tradeoff.

3 Method

3.1 Task Definition

Given a user listing $q = (t_q, d_q, m_q, c_q)$ with title, description, metadata, and category, retrieve the correct catalog SKU $s^* \in \mathcal{S}_{c_q}$, where $\mathcal{S}_{c_q}$ is the catalog for category c_q and each SKU $s = (t_s, d_s, m_s, c_s)$ has the same field structure. Each prediction includes a confidence score; listings below a threshold are excluded, trading coverage for precision. The evaluation metrics are:

- **Precision@1** (P@1): Fraction of *covered* predictions where the top-1 retrieved SKU is correct.
- **Coverage**: Fraction of listings for which a prediction is made.

Qualification requires ≥95% P@1 on *every* category independently. Among qualified submissions, the ranking objective is maximum average coverage. This setup reflects a practical operating point: automate high-confidence matches while routing uncertain listings for manual review.

We evaluate on three categories from a large Japanese C2C marketplace (Table 1). Agents develop methods using the training split (with ground truth for self-evaluation), then submit predictions on the test split, which is evaluated by the leaderboard against withheld ground truth.

Category	Base P@1	Base Cov.
Smartphone	100.0%	1.3%
Trading Card (Others)	94.5%	58.6%
Trading Card (Pokemon)	97.3%	39.9%
Average	**97.3%**	**33.3%**

Table 1: Dataset statistics and BGE-M3 baseline (test split). Thresholds tuned on train for 95% P@1 per category, then applied to test; the baseline is *not* qualified (Trading Card (Others) <95% P@1).

3.2 Framework Infrastructure

We built a reusable platform (Figure 1) with the following components:

Automated Leaderboard. A `LeaderboardAPI` evaluates submissions against withheld test ground truth that agents never access. Submissions must include all three categories with per-category confidence thresholds. The API computes P@1 and coverage per category, checks qualification (≥95% P@1 on every category), ranks qualified submissions by average coverage, and appends all results to a persistent JSON ledger. Agents can view the full leaderboard including scores and approach descriptions from all submissions.

Train/Test Separation and Integrity. Training data includes ground truth labels for self-evaluation with a lightweight script; test listings contain no ground truth, so agents must submit to the leaderboard for scoring. Withheld test labels live in an organizer-only location

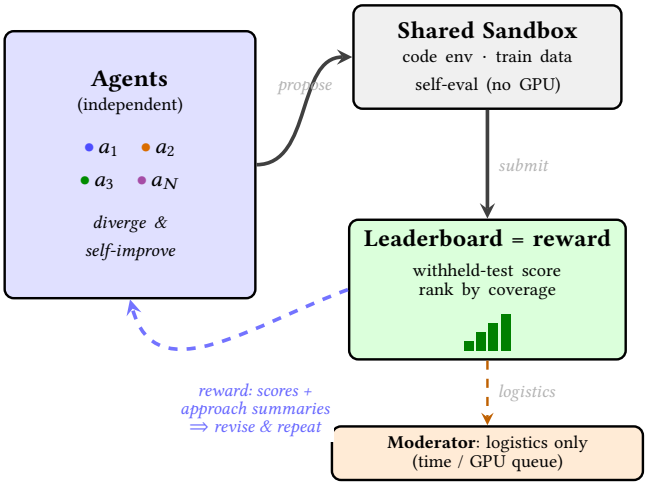

Figure 1: The framework as a continuous, autonomous loop: agents *propose* in a shared sandbox, *submit* to a leaderboard scored on withheld test data, and *revise* from the returned reward (scores and approach summaries, never code). Parallel agents broaden the explored space ($a_1 \ldots a_N$); no human is in the loop and the automated moderator handles only logistics.

with restricted file permissions. A strict organizer/shared/participant ownership boundary and an automated pre-run leak check *enforce* the separation rather than relying on convention. Each agent is further confined to its own working directory plus read-only shared resources, so attempts to reuse another agent's artifacts (which we did observe) are structurally prevented rather than merely discouraged. Each run follows a fixed protocol (setup with automated leak verification, the autonomous run scored on the withheld split by the leaderboard, and an organizer review confirming final standings), so the procedure is reproducible across sessions.

GPU Resource Management. Agents share one GPU node; naive concurrent use crashed it in early runs, motivating a two-level lock with a 30-minute `CLAIM/RELEASE GPU` protocol. GPU access is enforced programmatically by requiring agents to acquire the lock before invoking embedding or model code, and a system-level watchdog terminates unauthorized GPU processes.

Autonomy. An append-only chat log is the sole communication channel. *No human is in the loop* beyond a single launch command that specifies agent count and time budget (e.g., "Run the competition with 5 agents for 5 hours."). The moderator is itself an automated agent scoped to logistics only (time warnings, leaderboard updates, queue management); keeping it from drifting into technical guidance was an explicit design requirement. Agents devise and run the full research loop themselves. Continuous operation is enforced rather than left to agent discretion: each agent writes periodic liveness heartbeats and posts progress to the chat on a fixed cadence, and (instead of terminating after a submission) keeps analyzing, revising, and resubmitting. The moderator emits scheduled warnings (e.g., 30 and 5 minutes before deadline) and forcibly terminates remaining agent processes at the budget deadline.

3.3 Agent Setup

Each agent is powered by Claude Sonnet 3.5 (Anthropic) with access to a Python environment including embedding models (BGE-M3 [4], Qwen3-Embedding), FAISS [12] indexing, and TF-IDF libraries. Agents receive a task briefing specifying the task, data format, submission protocol, GPU coordination rules, 5-hour time budget, and the BGE-M3 baseline as a starting reference. Each follows an iterative research loop: analyze data/errors → survey candidate methods → hypothesize → implement → self-evaluate on train → submit test predictions → observe leaderboard → revise.

4 Experiments

4.1 Setup

We contrast two configurations of the framework on identical infrastructure (same tools, same fixed 5-hour per-session time budget with no submission cap), each run three times:

Single agent. One agent iterates alone.

Parallel (5 agents). Five agents run simultaneously and fully autonomously on a shared GPU. All coordination is via the append-only chat log; agents see leaderboard scores and approach descriptions but not each other's code.

Baseline. BGE-M3 embeddings with cosine similarity and per-category thresholds tuned on train for 95% P@1, applied to the test split (Table 1); not qualified.

4.2 Results

Continuous Improvement. In both configurations the leaderboard reward drove sustained self-improvement: across runs, best qualified coverage rose from the 33.3% baseline to 47.8–57.4% with a single agent and 62.8–69.4% with five (Table 2). The loop operates even with a single agent, which improved substantially over the baseline through iterative resubmission alone.

Parallel Autonomous Exploration. Across all three single-agent runs (60, 124, and 81 submissions) the agent stayed within its seeded embedding paradigm, iterating on BGE-M3 variants and never adopting a non-embedding method; in one run it prematurely judged the hardest category unsolvable and stopped early. In all three 5-agent runs, agents independently reached qualitatively different methods (TF-IDF re-ranking, string-similarity matching) within minutes. The gain concentrates in the hardest category, smartphone (single agents 0.8–1.9% coverage vs. 24.8–41.1% parallel); on the Trading Card categories a single agent with uncontested GPU is competitive with the parallel runs. Each configuration was run three times and we report ranges; these illustrate broadened exploration, not a causal effect of agent count, which stays confounded with total compute, GPU contention, and shared leaderboard visibility.

Approach Diversity (one representative 5-agent run). The five agents (α–ϵ = researcher1–5) all started from BGE-M3 embeddings but diverged: score-gap confidence calibration (α, best 50.7% qualified coverage); BGE-M3 with TF-IDF rescoring (β, 55.3%); card-number re-ranking with pure string (SequenceMatcher) matching (γ, 65.6%); string-matching plus word/char TF-IDF fusion (δ, 69.4%,

	Baseline	Single agent	Parallel (5)
Best Avg Cov.	33.3%	47.8 / 57.4 / 49.9	**62.8 / 69.4 / 68.2**
Best Avg P@1	97.3%	96.7 / 96.7 / 95.1	95.1 / 95.1 / 95.1
Qualified?	No	Yes / Yes / Yes	Yes / Yes / Yes
Submissions	—	60 / 124 / 81	274 / 550 / 301
Qualified sub.	—	22 / 44 / 46	105 / 238 / 103

Table 2: Main results: best qualified coverage/P@1 per configuration vs. the unqualified 33.3% baseline; three runs per configuration (run1 / run2 / run3). Descriptive: agent count co-varies with total compute, simultaneous workers, GPU contention, and shared leaderboard visibility; not a causal scaling law.

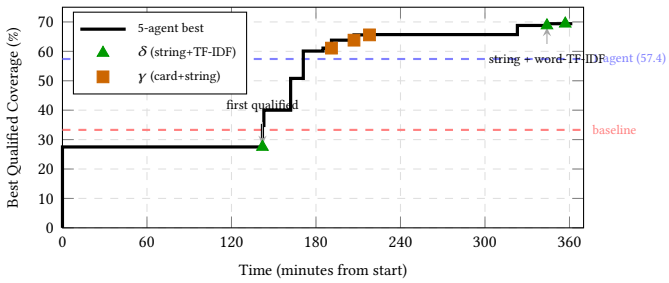

Figure 2: Best qualified coverage over time for one representative 5-agent run (run 2). Dashed lines: unqualified baseline (33.3%) and best single-agent (57.4%).

the run's best); and string-similarity re-ranking (ϵ, 63.6%). All five qualified; the non-embedding string/TF-IDF methods (γ, δ) drove the top results.

Leaderboard Dynamics. In this run the first qualified submission appeared only after extended exploration (δ, BGE-M3, 27.5%). Coverage climbed through threshold tuning and card-number re-ranking to 65.6%, then plateaued; the winning push came late, when δ added string-matching plus word-TF-IDF fusion and reached 69.4% (Figure 2).

5 Discussion

Depth vs. Breadth. The two mechanisms produce complementary behaviors. A single agent iterates deeply but narrowly: it refines BGE-M3 variants and thresholds at high precision (avg P@1 95.1–96.7%), and in one run prematurely concluded the hardest category unsolvable and stopped with budget remaining. Parallel agents instead widened the search, exploring qualitatively different (non-embedding) strategies rather than deeply refining one, at the cost of lower per-agent precision (95.1%) and heavy GPU contention. We read this as a depth-for-breadth trade and do not isolate the cause; the broader exploration nonetheless surfaced string-matching and TF-IDF methods the single agents never considered.

Narrow Exploration vs. Escape. Seeded with a BGE-M3 baseline, the single agent treated the embedding paradigm as the entire search space: across all three single-agent runs (60, 124, and 81 submissions) it iterated on BGE-M3 variants, score calibration, and thresholds and never instantiated the standard lexical/string baseline a practitioner would try first, in one run prematurely judging the hardest category unsolvable and stopping early. In all three 5-agent runs, agents reached non-embedding methods (TF-IDF, string-similarity matching) within the session. That strong lexical/string methods rival dense embeddings on near-identical-variant matching is well known and *not* our claim; the point is behavioral: a single seeded explorer iterates narrowly and prematurely converges, while parallel autonomous exploration escapes it. Because all runs use one LLM, seed, and prompt, we cannot separate this from a model-specific prior; a no-seed control is the decisive next experiment.

Information Sharing and Integrity. Agents could view others' scores and approach summaries but not their code or artifacts: the catalog index is shared infrastructure provided to all, but reusing *another agent's* pre-built indexes or trained models was prohibited. The reward channel carries both scores and brief approach descriptions (Figure 1), so parallel agents share strategies at a high level while their implementations stay isolated. This visibility is intended to support differentiation across agents' directions, broadening the explored region of the solution space.

Implications. The framework targets applied research: the reward-driven, human-free loop can speed the exploration and improvement of product-to-catalog matching beyond manual iteration and, being task-agnostic, can likewise advance applied research on other tasks in C2C marketplace systems such as search ranking, recommendation, and listing-quality checks.

Limitations. Our central qualitative finding (a single seeded agent iterating narrowly and prematurely converging while parallel agents escape) replicates across runs and is independent of agent count, so the limitations below do not affect it. We make no causal or agent-count claim: agent count co-varies with total compute, simultaneous workers, GPU contention, and shared leaderboard visibility, and the single-vs-parallel coverage figures (Table 2) are descriptive, not evidence of a scaling effect. We ran each configuration three times and report ranges. The decisive control for our central finding is a no-seed condition, separating the seeded baseline from a model-specific prior. All agents use the same LLM (Claude Sonnet 3.5), and the task is bounded (automated metrics, fast iteration); longer-horizon settings with ambiguous success criteria remain open. We instantiated the framework on a single task (product-to-catalog matching); while it is task-agnostic by design, cross-task generalization to other IR and ML problems remains to be demonstrated empirically. Agents write and submit code autonomously; individual submissions were not human-reviewed for safety. The infrastructure boundaries described in §3.2 (per-agent working directories, restricted-permission test labels, the GPU lock) prevent cross-agent damage and integrity violations, but do not audit the LLM-generated code itself.

6 Conclusion

We presented a framework that gives LLM agents two mechanisms for autonomously searching large solution spaces: a withheld-test leaderboard that acts as a reward signal driving iterative improvement of submitted solutions, and a substrate for running

many agents in parallel with *no human in the loop*. We instantiated it on product-to-catalog matching, where the central qualitative pattern held across all six runs: the seeded single agent iterated within its embedding paradigm, while parallel autonomous agents reached non-embedding methods (TF-IDF, string-similarity matching) within the session. The decisive next test is a no-seed control, separating the seeded baseline from the model's own prior; whether these dynamics generalize to other bounded optimization tasks is a further open question.

References

[1] Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. 2023. Autonomous chemical research with large language models. *Nature* 624, 7992 (2023), 570–578.

[2] Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2024. ChemCrow: Augmenting large language models with chemistry tools. *Nature Machine Intelligence* 6, 5 (2024), 525–535.

[3] 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 *The Thirteenth International Conference on Learning Representations*.

[4] Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*. Association for Computational Linguistics, 2318–2335.

[5] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. 2024. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. In *The Twelfth International Conference on Learning Representations*.

[6] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving Factuality and Reasoning in Language Models through Multiagent Debate. In *Proceedings of the 41st International Conference on Machine Learning*. PMLR, 11733–11763.

[7] Yonatan Geifman and Ran El-Yaniv. 2017. Selective Classification for Deep Neural Networks. In *Advances in Neural Information Processing Systems*, Vol. 30.

[8] Antoine Grosnit, Alexandre Maraval, James Doran, Rasul Tutunov, Wenlong Wang, Haitham Bou Ammar, et al. 2024. Large Language Models Orchestrating Structured Reasoning Achieve Kaggle Grandmaster Level. *arXiv preprint arXiv:2411.03562* (2024).

[9] Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. 2024. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In *The Twelfth International Conference on Learning Representations*.

[10] 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*. PMLR, 20271–20309.

[11] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can Language Models Resolve Real-World GitHub Issues?. In *The Twelfth International Conference on Learning Representations*.

[12] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2021), 535–547.

[13] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. 6769–6781.

[14] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. In *Advances in Neural Information Processing Systems*, Vol. 36.

[15] Sijie Li, Weiwei Sun, Shanda Li, Ameet Talwalkar, and Yiming Yang. 2025. CoMind: Towards Community-Driven Agents for Machine Learning Engineering. *arXiv preprint arXiv:2506.20640* (2025).

[16] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2021. Deep Entity Matching with Pre-Trained Language Models. *Proceedings of the VLDB Endowment* 14, 1 (2021), 50–60.

[17] Ziming Li, Qianbo Zang, David Ma, Jiawei Guo, Tuney Zheng, Minghao Liu, Xinyao Niu, Yue Wang, Jian Yang, Jiaheng Liu, Wanjun Zhong, Wangchunshu Zhou, Wenhao Huang, and Ge Zhang. 2024. AutoKaggle: A Multi-Agent Framework for Autonomous Data Science Competitions. *arXiv preprint arXiv:2410.20424* (2024).

[18] 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 preprint arXiv:2408.06292* (2024).

[19] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909* (2015).

[20] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep Learning for Entity Matching: A Design Space Exploration. In *Proceedings of the 2018 International Conference on Management of Data*. ACM, 19–34.

[21] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. 2024. ChatDev: Communicative Agents for Software Development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*. 15174–15186.

[22] 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, et al. 2024. Mathematical discoveries from program search with large language models. *Nature* 625, 7995 (2024), 468–475.

[23] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Vol. 36.

[24] Lei Wang, Chen Ma, Xueyang Feng, et al. 2024. A Survey on Large Language Model based Autonomous Agents. *Frontiers of Computer Science* 18, 6 (2024), 186345.

[25] Qingyun Wu, Gagan Bansal, Jieyu Zhang, et al. 2024. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. In *Proceedings of the First Conference on Language Modeling (COLM)*.

[26] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024. Large Language Models as Optimizers. In *The Twelfth International Conference on Learning Representations*.

[27] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations*.

[28] Qinlin Zhao, Jindong Wang, Yixuan Zhang, Yiqiao Jin, Kaijie Zhu, Hao Chen, and Xing Xie. 2024. CompeteAI: Understanding the Competition Dynamics of Large Language Model-based Agents. In *Proceedings of the 41st International Conference on Machine Learning*. PMLR, 61092–61107.