

A Literature Survey of Multi-Agent Large Language Models for Startup Evaluation and Founder Decision Support

Dr. Emilin Shyni C¹, Charumathi D², Deepak Raj T³, Pradeep K⁴,
Sanju R Bannikodu⁵, Vishnu C S⁶

^{1,2,3,4,5,6}Department of CSE (Artificial Intelligence and Machine Learning), East Point College of Engineering and Technology, Karnataka, India

Abstract

Generative artificial intelligence (AI) has made it cheap to produce startup artifacts such as interface copy, mock-ups, pitch decks and working prototypes, but deciding whether those artifacts are any good has not become cheaper. This survey examines the research behind that gap. The work covered was published between 2017 and 2025 in five areas: generative models for product and code tasks, multi-agent systems built on large language models (LLMs), reasoning, reflection and memory techniques, LLM-as-a-judge evaluation and its known biases, and empirical studies of human and AI collaboration and entrepreneurial decision-making. Comparative tables bring together the main frameworks, judge biases and field studies. Across these areas, generation is studied far more than judgment, and four gaps stand out: evaluation is rarely the main job of a multi-agent system, reviewer independence is little studied, judgments cannot yet be compared across product versions, and the effect on founders' decisions has not been measured. The survey ends with open challenges and a methodological outlook.

Keywords: Literature Survey, Multi-Agent Systems, Large Language Models, LLM-as-a-Judge, Startup Evaluation, Human-AI Collaboration, Decision Support

1. Introduction

Foundation models, the core of generative artificial intelligence (AI), have changed what one person can produce. A transformer-based language model [1] trained at scale can pick up a new task from a few examples [2], and instruction tuning with human feedback made these models reliable at following plain requests [3]. A founder can now get interface copy, specifications, landing pages and working code in minutes [4, 5]. Judging whether that output is correct, and whether it is worth pursuing, still takes the same effort as before.

This matters because early ventures often fail from poor decisions rather than poor production. Post-mortem studies place the lack of market need among the leading causes of startup failure [6]. The lean-startup literature says founders should treat each product decision as a hypothesis and test it with customers [7, 8], and economists describe the same process as entrepreneurship by experimentation [9]. A randomized trial found that founders taught to form and test hypotheses did better than untrained peers

[10]. Generation speeds up the building step of this loop, but it gives the founder no help with the evaluating step.

A growing body of work asks whether large language models (LLMs) can also evaluate, and whether several models working together evaluate better than one. This paper is a literature survey and does not propose a system. It reviews research from 2017 to 2025 in five strands: (i) generative models and their use on product and code artifacts, (ii) multi-agent language model systems, (iii) reasoning, reflection, tool use and memory, (iv) LLM-as-a-judge evaluation and its documented biases, and (v) empirical studies of human-AI work and of startup decision-making. Sections 2 to 7 cover these strands. Section 8 sets out the research gaps, Section 9 lists open challenges, and Section 10 concludes.

Sources were drawn from major machine-learning venues (NeurIPS, ICLR, ICML, ACL, EMNLP), management and economics journals, and recent technical reports. Studies were retained if they either tested a language-model capability relevant to critique or measured the effect of AI assistance on human work or entrepreneurial decisions.

2 Generative Models and Product Artifacts

2.1 From Pre-training to Instruction Following

Early surveys of foundation models stressed their breadth and the uncertainty about how they fail [5]. Reinforcement learning from human feedback turned raw capability into usable assistance [3], and broad overviews now document how architectures, data mixtures and alignment recipes have diversified [11]. In 2025, reasoning models trained with large-scale reinforcement learning, such as DeepSeek-R1 [12] and Qwen3 [13], and multimodal systems such as Gemini 2.5 [14] pushed capability further. Founders can now use tools that read screenshots, long documents and whole codebases.

2.2 Code and Software Artifacts

Code was the first domain where functional correctness could be measured: models trained on code solved hand-written programming problems when sampled and tested [4]. Benchmarks have since moved to repository-level work. SWE-bench draws real GitHub issues from popular Python projects, and early models resolved only a small share of them [15]. Agent scaffolds that give the model a purpose-built interface to the editor and shell raised those numbers considerably [16]. Generation quality depends heavily on the feedback available at test time, such as unit tests or human review. A pitch deck or a pricing page has no unit test, so for founders the hard part is evaluation.

2.3 Multimodal and Design Artifacts

Founders' artifacts are rarely text alone. They include screenshots, interface mock-ups, slide decks and short demo videos. Frontier multimodal models can take such inputs [14], but the judge literature reviewed below is mostly about text [17, 18, 19]. There is little evidence on how reliable model judgments of visual design or layout are, or whether the known biases carry over to images. Until this is studied, text-based findings cannot safely be assumed to hold for visual work.

3 Multi-Agent Language Model Systems

3.1 Collaboration Frameworks

Multi-agent work began by letting several model instances talk to each other. AutoGen offers a general framework in which agents exchange messages and call tools [20]. CAMEL used role-playing with "inception prompts" so that two agents could finish a task on their own [21]. MetaGPT encoded standard operating procedures (SOPs) from software companies and passed structured documents between roles

[22]. ChatDev organized a virtual software company around chat chains [23], and AgentVerse studied dynamic group composition and emergent behaviors [24]. Table 1 summarizes these designs. In all of them the agents work at successive stages of production. None is built mainly to critique a finished artifact independently.

Table 1: Representative Multi-Agent LLM Frameworks

Work	Coordination	Agent roles	Relevance to evaluation
AutoGen [20]	Conversation	Flexible, user-defined	Enables critic agents, but supplies no rubric
CAMEL [21]	Paired role-play	Task-specific pair	Tests autonomy, not judgment
MetaGPT [22]	SOP pipeline	Company roles	Review limited to producer-checker
ChatDev [23]	Chat chain	Software team	Testing stage only
AgentVerse [24]	Dynamic groups	Recruited experts	Studies group composition
Debate [25]	Multi-round	Peer debaters	Improves factuality
MoA [26]	Layered	Proposers, aggregator	Aggregation of diverse answers
ChatEval [27]	Referee debate	Judge personas	Directly targets evaluation

3.2 Debate, Aggregation and Collective Judgment

A second line of work has several instances check one another. Multi-agent debate improved factuality and arithmetic and strategic reasoning compared with single-pass answers [25]. A related method assigned opposing stances to counter degeneration of thought, the tendency of a model to lock into its first answer [28]. Mixture-of-Agents (MoA) stacks layers of proposers whose outputs an aggregator synthesizes, and reports gains from model diversity [26]. Later analysis tempers these results. A study of debate as a framework shows that outcomes depend on how agents influence one another, and that naive protocols can amplify shared errors [29]. Taken together, these studies suggest that who takes part matters more than how many take part.

Role specialization is the other design axis. Pipelines give each agent a profession and a deliverable [22, 23]. Evaluation-oriented designs give each agent a standpoint and have all of them look at the same object [27]. This difference matters because pipeline agents are rewarded for converging on one artifact, while reviewers are useful when they disagree. Surveys note that most published systems are tested on benchmark tasks with a known answer, so it is hard to say how any of them behaves when the correct verdict is contested [30, 31].

3.3 Surveys and Failure Analyses

Several surveys organize the field by profiling, communication, capability and application [30, 31, 32]. Specialized reviews cover software engineering [33] and agent construction, collaboration and evolution [34]. Open-problem papers list coordination cost, hallucination propagation and scalability as unsolved [35]. The most systematic failure study so far annotated traces from popular frameworks and proposed fourteen failure modes in three groups: specification problems, inter-agent misalignment and weak task

verification [36]. This is a problem for evaluation, because the verification stage, which should catch errors, is one of the weakest parts.

4 Reasoning, Reflection, Tools and Memory

4.1 Reasoning and Acting

Chain-of-thought prompting showed that eliciting intermediate steps improves performance on multi-step problems [37]. Self-consistency samples several reasoning paths and takes the majority answer [38]. Tree of Thoughts extends this to deliberate search with backtracking [39]. ReAct interleaves reasoning with actions so that a model can consult external sources [40], and Toolformer showed that models can learn when to call tools with little supervision [41]. A critique that must rest on evidence, and not on impressions, needs these techniques.

4.2 Self-Critique and Reflection

Reflexion stores verbal feedback from earlier failures and conditions later attempts on it [42]. Self-Refine alternates generation, feedback and revision within a single model [43]. Constitutional AI uses written principles to drive model-generated critiques and revisions [44]. Self-critique has limits. Experiments show that models struggle to correct their own reasoning without an external signal and sometimes make correct answers worse [45]. This is a reason to use separate critics with different perspectives, and objective checks where they exist.

4.3 Retrieval and Long-Term Memory

Retrieval-augmented generation grounds outputs in external documents [46], and recent surveys catalogue naive, advanced and modular variants [47]. Long context does not replace retrieval, because accuracy often drops when the relevant information sits in the middle of a long input [48]. For continuity across sessions, generative agents combine a memory stream with reflection and planning [49]. MemoryBank adds an update rule inspired by the forgetting curve [50], MemGPT manages context as a hierarchy like operating-system virtual memory [51], and Voyager builds up an executable skill library [52]. None of these studies looks at memory as a way to keep evaluations comparable across revisions of a changing product.

Memory also bears on evaluation through context limits. Because accuracy falls when relevant evidence is buried in long inputs [48], a system that compares a current submission with many earlier ones needs selective retrieval [47] rather than concatenation. Whether such retrieval keeps enough information for scores to stay comparable over time is an empirical question that the surveyed literature leaves open.

5 LLM-as-a-Judge and Automated Evaluation

5.1 Foundations

MT-Bench and Chatbot Arena formalized the use of a strong model to grade open-ended output. GPT-4 agreed with human preferences at a rate comparable to agreement between humans, and the authors documented three biases: position, verbosity and self-enhancement [17]. G-Eval prompts a model with criteria and chain-of-thought and weights scores by token probabilities [18]. Prometheus fine-tunes an open evaluator that accepts user-defined rubrics and reference answers [53]. A broad survey classifies judge designs by input format, criteria and scoring scheme [19]. Agent-as-a-judge extends the idea so that agents inspect the intermediate steps of other agents [54].

5.2 Multi-Judge Panels

Single judges are noisy and biased, so several studies combine multiple judges. In ChatEval, judge person-

as debate before giving a verdict, and the authors report closer alignment with humans than single prompts [27]. A panel made of smaller models from different families outperformed one large judge at lower cost, with less intra-model bias [55]. Both findings agree with the debate work: judges from different families help, and a panel of similar judges shares the same blind spots.

Rubric design comes up repeatedly in practice. Rubric-conditioned evaluators rest on the premise that explicit, itemized criteria give more consistent judgments than holistic prompts [53]. Eliciting reasoning before a score is widely used [18], although a written rationale may not reflect how the verdict was actually reached [19]. Reliability is usually reported as agreement with human raters, yet humans also disagree on subjective criteria [17]. Several authors therefore recommend reporting inter-judge agreement, order-swap consistency and confidence intervals together rather than a single correlation figure [19, 56].

Standard metrics also overlook specificity. A critique can be fluent, agree with human preferences on average, and still be generic enough to fit any product. The most common remedy in the literature is to ground judgments in retrieved project facts [46] and in evidence gathered through tool calls [40]. Rubric-conditioned evaluators are designed so that explicit criteria discourage vague feedback [53]. Calibration frameworks have also proposed keeping a human in the loop for borderline cases [57], which suits settings where a wrong verdict means a misdirected product roadmap.

5.3 Documented Biases and Mitigations

Table 2 consolidates the main biases. Changing the order alone can flip pairwise verdicts, which motivated calibration by swapping positions and averaging the evidence [57]. A large study of position bias confirms that it varies with task and judge and is not simply noise [56]. Judges also favour text that resembles their own output, and this self-preference correlates with the ability to recognize their own generations [58]. A broader study quantified twelve bias types, including those tied to authority cues and formatting [59].

Table 2: Documented Judge Biases and Reported Mitigations

Bias	Evidence	Mitigation reported
Position	Verdict flips with answer order [56, 57]	Swap order; average over permutations
Verbosity	Longer answers preferred [17]	Rubrics penalizing padding; length control
Self-preference	Own outputs rated higher [58]	Judges from other model families [55]
Authority format	Citations and style sway scores [59]	Blind formatting; structured criteria [53]
Rubric drift	Scores unstable across runs [18]	Fixed rubrics, scored fields, repeated sampling

Reproducibility is a last concern. Judge outputs depend on prompt wording, sampling temperature and the model version behind an API, and all of these change over time [19, 56]. Few studies fix and publish their prompts, rubrics and model identifiers, so reported agreement figures are hard to replicate. Open-weight evaluators [53] help by letting researchers freeze the judge.

6 Human-AI Work and Evidence on Productivity

Field and laboratory studies give a mixed but informative picture of what generative AI does to human

performance (Table 3). Developers using Copilot finished a bounded programming task substantially faster [60]. Professionals writing mid-level business documents saved time and raised quality with ChatGPT [61], and customer-support agents became more productive, with the largest gains for the least experienced [62]. The "jagged frontier" experiment with consultants found large gains on tasks inside the model's competence and lower accuracy on a task outside it, and participants did not reliably know which case they were in [63]. In contrast, a 2025 randomized trial with experienced open-source developers on mature codebases measured a slowdown, even though they expected a speedup [64].

Table 3: Empirical Studies of AI-Assisted Work

Study	Population	Principal finding
Peng et al. [60]	Developers	Faster completion of a coding task
Noy & Zhang [61]	Professionals	Less time, higher rated quality
Brynjolfsson et al. [62]	Support agents	Gains largest for novices
Dell'Acqua et al. [63]	Consultants	Gains inside, errors outside frontier
Becker et al. [64]	Expert developers	Slowdown on familiar codebases

Two implications follow. First, benefits concentrate among less experienced users. Many first-time founders fit that description, so evaluation support could be most valuable where expert feedback is scarcest. Second, the jagged-frontier result means users need help knowing when not to trust the output. A critique layer could provide that help, provided the critique itself is reliable.

7. Startup Decision-Making and Experimentation

The entrepreneurship literature supplies the criteria against which an evaluator should ultimately be judged. The lean-startup method frames a venture as a sequence of build-measure-learn cycles in which a minimum viable product exists to test a hypothesis [7, 8]. Kerr et al. argue that entrepreneurship is experimentation under uncertainty, where the value of an experiment depends on its cost and on the founder's ability to read the result [9]. Generative tools sharply lower the cost of the build step. That lets a founder launch more experiments, and it also leaves more results to interpret. Camuffo et al. found that founders trained to state hypotheses and test them against evidence earned more and changed direction more readily when the evidence called for it [10]. These studies imply that automated critique is only useful if it sharpens hypotheses and brings up evidence against them. Approving artifacts does not help.

8 Synthesis and Research Gaps

8.1 Four Research Gaps

Reading the strands together reveals four gaps.

Gap 1: Evaluation is rarely the primary task of multi-agent systems. Frameworks target production pipelines [22, 23], and judge studies target model outputs rather than business artifacts [17, 19].

Gap 2: Independence of reviewers is under-studied. Debate and aggregation work shows value from diversity [25, 26, 29], but few studies compare isolated parallel reviewers with interacting ones for critique quality.

Gap 3: Temporal comparability is unexplored. Memory research targets dialogue and embodied tasks [49,

51], and no study asks whether scores for successive versions of a product can be compared meaningfully. Gap 4: Decision outcomes are not measured. Judge papers report agreement with human raters, and productivity studies report speed and quality. Whether AI critique improves founders' later choices, the quantity the entrepreneurship literature cares about [10], remains untested.

Table 4 condenses the comparison for the evaluator-oriented techniques reviewed above. No single approach combines multiple standpoints, itemized rubrics, bias controls, cross-version memory and outcome-based validation. Each covers one or two of these.

Table 4: Evaluator-Oriented Techniques against Five Desired Properties

Technique	Multiple views	Itemized rubric	Bias control	Cross-version memory	Outcome checks
G-Eval [18]	No	Yes	Partial	No	No
Prometheus [53]	No	Yes	Partial	No	No
Debate [25]	Yes	No	No	No	No
ChatEval [27]	Yes	Partial	Partial	No	No
Panel of judges [55]	Yes	Partial	Yes	No	No
MoA [26]	Yes	No	No	No	No

8.2 Cost and Latency

Every multi-sample or multi-agent technique multiplies inference cost. Self-consistency needs many sampled chains [38], debate needs several rounds [25], and layered aggregation needs several model calls per answer [26]. Panels of smaller models can match or beat a single large judge at lower cost [55], which suggests that heterogeneity and economy need not conflict. Still, the surveyed studies seldom report cost-effectiveness next to accuracy. Founders on tight budgets will care about this a great deal.

8.3 Trust and Calibration

Productivity studies show that users cannot reliably tell when a model is working outside its competence [63], and that confident expectations of AI benefit can differ from measured results [64]. Because self-correction without external signals is unreliable [45], a numeric score shown without uncertainty may attract more trust than it deserves. The judge literature offers partial remedies: reporting agreement and order-swap consistency [56], disclosing the rubric [53], and showing disagreement among reviewers instead of averaging it away [27, 55].

9 Open Challenges

Validity: Agreement with a human panel is a weak proxy when the panel is itself noisy, and ground truth for startup quality arrives only after market exposure.

Generic critique: Without retrieval of project-specific facts [46], judges drift toward platitudes that fit any startup.

Correlated reviewers: Agents drawn from one model family share blind spots [55, 58].

Overload: Many independent findings can exceed the attention of a time-poor founder, so ranking and truncation are needed.

Calibration and over-trust: Numeric scores can look more authoritative than they are, and the jagged-frontier evidence shows users misjudge when to rely on AI [63].

Evaluation methodology: Studies should combine rubric-based audits, order-swap robustness checks [57], expert comparison, and longitudinal outcome tracking in the spirit of randomized trials [9, 10].

9.1 Directions for Future Research

Four directions follow from the survey. First, benchmarks are needed for artifact critique in business settings, with expert-annotated flaws so that recall and precision of findings can be computed, extending the logic of repository-level benchmarks [15] to non-code products. Second, controlled comparisons should isolate the contribution of reviewer independence, model heterogeneity and rubric structure, borrowing the ablation style of debate studies [25, 29]. Third, longitudinal designs should test whether memory preserves comparability across product revisions [50, 51]. Fourth, randomized field trials with founders, patterned on the trial of Camuffo et al. [10], should measure decisions and venture outcomes rather than rater agreement.

9.2 Threats to Validity of This Survey

The field moves quickly, and many cited works are preprints or technical reports that have not completed peer review [12, 14, 36]. Benchmarks and model versions change faster than journals publish, so quantitative claims should be read as indicative. The survey is also selective: it emphasizes English-language work from major venues and leaves out proprietary tools whose internals are not disclosed.

10 Conclusion

The literature suggests that generation is far better studied than judgment. Multi-agent frameworks, reasoning and reflection methods, memory mechanisms and LLM-as-a-judge techniques each supply part of what founder-facing evaluation needs, but they have rarely been studied together on startup artifacts. Heterogeneous judges, fixed rubrics, bias controls and external grounding are the ingredients with the most consistent support. Whether such critique improves real decisions is still unknown. Answering that needs outcome-based studies with early-stage founders, and this is considered to be the most useful direction for future work.

References

1. A. Vaswani, et al., “Attention is all you need”, Proceedings of NeurIPS, 2017.
2. T. Brown, et al., “Language models are few-shot learners”, Proceedings of NeurIPS, 2020.
3. L. Ouyang, et al., “Training language models to follow instructions with human feedback”, Proceedings of NeurIPS, 2022.
4. M. Chen, et al., “Evaluating large language models trained on code”, arXiv, 2021. <https://arxiv.org/abs/2107.03374>
5. R. Bommasani, et al., “On the opportunities and risks of foundation models”, arXiv, 2021. <https://arxiv.org/abs/2108.07258>
6. CB Insights, “The top 12 reasons startups fail”, Research Brief, 2021.
7. E. Ries, “The Lean Startup”, Crown Business, 2011.
8. S. Blank, “Why the lean start-up changes everything”, Harvard Business Review, 2013, 91 (5), 63–72.
9. W. R. Kerr, R. Nanda, M. Rhodes-Kropf, “Entrepreneurship as experimentation”, Journal of Economic Perspectives, 2014, 28 (3), 25–48.

10. A. Camuffo, et al., “A scientific approach to entrepreneurial decision making: Evidence from a randomized control trial”, *Management Science*, 2020, 66 (2), 564–586.
11. H. Naveed, et al., “A comprehensive overview of large language models”, *arXiv*, 2024. <https://arxiv.org/abs/2307.06435>
12. DeepSeek-AI, “DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning”, *arXiv*, 2025. <https://arxiv.org/abs/2501.12948>
13. A. Yang, et al., “Qwen3 technical report”, Qwen Team, Alibaba Group, Technical Report, 2025.
14. G. Comanici, et al., “Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities”, Google DeepMind, Technical Report, 2025.
15. C. E. Jimenez, et al., “SWE-bench: Can language models resolve real-world GitHub issues?”, *Proceedings of ICLR*, 2024.
16. J. Yang, et al., “SWE-agent: Agent-computer interfaces enable automated software engineering”, *Proceedings of NeurIPS*, 2024.
17. L. Zheng, et al., “Judging LLM-as-a-judge with MT-Bench and Chatbot Arena”, *Proceedings of NeurIPS Datasets and Benchmarks*, 2023.
18. Y. Liu, et al., “G-Eval: NLG evaluation using GPT-4 with better human alignment”, *Proceedings of EMNLP*, 2023.
19. J. Gu, et al., “A survey on LLM-as-a-judge”, *arXiv*, 2024. <https://arxiv.org/abs/2411.15594>
20. Q. Wu, et al., “AutoGen: Enabling next-gen LLM applications via multi-agent conversation”, *arXiv*, 2023. <https://arxiv.org/abs/2308.08155>
21. G. Li, et al., “CAMEL: Communicative agents for 'mind' exploration of large language model society”, *Proceedings of NeurIPS*, 2023.
22. S. Hong, et al., “MetaGPT: Meta programming for a multi-agent collaborative framework”, *Proceedings of ICLR*, 2024.
23. C. Qian, et al., “ChatDev: Communicative agents for software development”, *Proceedings of ACL*, 2024.
24. W. Chen, et al., “AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors”, *Proceedings of ICLR*, 2024.
25. Y. Du, et al., “Improving factuality and reasoning in language models through multiagent debate”, *Proceedings of ICML*, 2024.
26. J. Wang, et al., “Mixture-of-Agents enhances large language model capabilities”, *arXiv*, 2024. <https://arxiv.org/abs/2406.04692>
27. C.-M. Chan, et al., “ChatEval: Towards better LLM-based evaluators through multi-agent debate”, *Proceedings of ICLR*, 2024.
28. T. Liang, et al., “Encouraging divergent thinking in large language models through multi-agent debate”, *Proceedings of EMNLP*, 2024.
29. A. Estornell, Y. Liu, “Multi-LLM debate: Framework, principals, and interventions”, *Proceedings of NeurIPS*, 2024.
30. T. Guo, et al., “Large language model based multi-agents: A survey of progress and challenges”, *arXiv*, 2024. <https://arxiv.org/abs/2402.01680>
31. L. Wang, et al., “A survey on large language model based autonomous agents”, *Frontiers of Computer Science*, 2024, 18.

32. Z. Xi, et al., “The rise and potential of large language model based agents: A survey”, arXiv, 2023.
<https://arxiv.org/abs/2309.07864>
33. J. He, et al., “LLM-based multi-agent systems for software engineering: Literature review, vision, and the road ahead”, ACM Transactions on Software Engineering and Methodology, 2025.
34. J. Luo, et al., “Large language model agent: A survey on methodology, applications and challenges”, arXiv, 2025. <https://arxiv.org/abs/2503.21460>
35. C. Han, et al., “LLM multi-agent systems: Challenges and open problems”, arXiv, 2024.
<https://arxiv.org/abs/2402.03578>
36. M. Cemri, et al., “Why do multi-agent LLM systems fail?”, arXiv, 2025.
<https://arxiv.org/abs/2503.13657>
37. J. Wei, et al., “Chain-of-thought prompting elicits reasoning in large language models”, Proceedings of NeurIPS, 2022.
38. X. Wang, et al., “Self-consistency improves chain of thought reasoning in language models”, Proceedings of ICLR, 2023.
39. S. Yao, et al., “Tree of Thoughts: Deliberate problem solving with large language models”, Proceedings of NeurIPS, 2023.
40. S. Yao, et al., “ReAct: Synergizing reasoning and acting in language models”, Proceedings of ICLR, 2023.
41. T. Schick, et al., “Toolformer: Language models can teach themselves to use tools”, Proceedings of NeurIPS, 2023.
42. N. Shinn, et al., “Reflexion: Language agents with verbal reinforcement learning”, Proceedings of NeurIPS, 2023.
43. A. Madaan, et al., “Self-Refine: Iterative refinement with self-feedback”, Proceedings of NeurIPS, 2023.
44. Y. Bai, et al., “Constitutional AI: Harmlessness from AI feedback”, arXiv, 2022.
<https://arxiv.org/abs/2212.08073>
45. J. Huang, et al., “Large language models cannot self-correct reasoning yet”, Proceedings of ICLR, 2024.
46. P. Lewis, et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks”, Proceedings of NeurIPS, 2020.
47. Y. Gao, et al., “Retrieval-augmented generation for large language models: A survey”, arXiv, 2023.
<https://arxiv.org/abs/2312.10997>
48. N. F. Liu, et al., “Lost in the middle: How language models use long contexts”, Transactions of the Association for Computational Linguistics, 2024, 12.
49. J. S. Park, et al., “Generative agents: Interactive simulacra of human behavior”, Proceedings of UIST, 2023.
50. W. Zhong, et al., “MemoryBank: Enhancing large language models with long-term memory”, Proceedings of AAAI, 2024.
51. C. Packer, et al., “MemGPT: Towards LLMs as operating systems”, arXiv, 2023.
<https://arxiv.org/abs/2310.08560>
52. G. Wang, et al., “Voyager: An open-ended embodied agent with large language models”, Transactions on Machine Learning Research, 2024.

53. S. Kim, et al., “Prometheus: Inducing fine-grained evaluation capability in language models”, Proceedings of ICLR, 2024.
54. M. Zhuge, et al., “Agent-as-a-judge: Evaluate agents with agents”, arXiv, 2024. <https://arxiv.org/abs/2410.10934>
55. P. Verga, et al., “Replacing judges with juries: Evaluating LLM generations with a panel of diverse models”, arXiv, 2024. <https://arxiv.org/abs/2404.18796>
56. L. Shi, et al., “Judging the judges: A systematic study of position bias in LLM-as-a-judge”, Proceedings of IJCNLP-AACL, 2025.
57. P. Wang, et al., “Large language models are not fair evaluators”, Proceedings of ACL, 2024.
58. A. Panickssery, S. R. Bowman, S. Feng, “LLM evaluators recognize and favor their own generations”, Proceedings of NeurIPS, 2024.
59. J. Ye, et al., “Justice or prejudice? Quantifying biases in LLM-as-a-judge”, arXiv, 2024. <https://arxiv.org/abs/2410.02736>
60. S. Peng, et al., “The impact of AI on developer productivity: Evidence from GitHub Copilot”, arXiv, 2023. <https://arxiv.org/abs/2302.06590>
61. S. Noy, W. Zhang, “Experimental evidence on the productivity effects of generative artificial intelligence”, Science, 2023, 381 (6654), 187–192.
62. E. Brynjolfsson, D. Li, L. Raymond, “Generative AI at work”, National Bureau of Economic Research (NBER) Working Paper 31161, 2023.
63. F. Dell'Acqua, et al., “Navigating the jagged technological frontier: Field experimental evidence of the effects of AI on knowledge worker productivity and quality”, Harvard Business School Working Paper 24-013, 2023.
64. J. Becker, et al., “Measuring the impact of early-2025 AI on experienced open-source developer productivity”, METR, Technical Report, 2025.