Self-Improvement of Large Language Models: A Technical Overview and Future Outlook

Anonymous authors
Paper under double-blind review

Abstract

As large language models (LLMs) continue to advance, improving them solely through human supervision is becoming increasingly costly and limited in scalability. As models approach human-level capabilities in certain domains, human feedback may no longer provide sufficiently informative signals for further improvement. At the same time, the growing ability of models to make autonomous decisions and execute complex actions naturally enables abstractions in which components of the model development process can be progressively automated. Together, these challenges and opportunities have driven increasing interest in self-improvement, where models autonomously generate data, evaluate outputs, and iteratively refine their own capabilities. In this paper, we present a system-level perspective on self-improving language models and introduce a unified framework that organizes existing techniques. We conceptualize the self-improvement system as a closed-loop lifecycle, consisting of four tightly coupled processes: data acquisition, data selection, model optimization, and inference refinement, along with an autonomous evaluation layer throughout the process. Within this framework, the model itself plays a central role in driving each stage: collecting or generating data, selecting informative signals, updating its parameters, and refining outputs, while the autonomous evaluation layer continuously monitors progress and guides the improvement cycle across stages. Following this lifecycle perspective, we systematically review and analyze representative methods for each component from a technical standpoint. We further discuss current limitations and outline our vision for future research toward fully self-improving LLMs.

1 Introduction

Large language models (LLMs) have achieved rapid and consistent performance gains through scaling model size, training data, and compute (Brown et al., 2020; Ouyang et al., 2022; Hoffmann et al., 2022; OpenAI et al., 2024). A widely held assumption underlying this progress is that larger and higher-quality datasets, especially expert-annotated human supervision, lead to stronger models. In practice, methods like RLHF (Ouyang et al., 2022) rely heavily on carefully curated, high-quality supervision to align and refine pretrained models. However, as models continue to advance, the paradigm of improving them primarily through human supervision reveals several structural limitations: (1) Human data scarcity is becoming increasingly evident. High-quality, expert-annotated data is expensive and difficult to scale (Gilardi et al., 2023; Villalobos et al., 2024). The marginal cost of constructing large supervised datasets grows rapidly, while the availability of expert labor remains limited. (2) There is a deeper limitation tied to human cognitive bounds. If model supervision is permanently constrained by human intelligence, can models truly surpass human-level performance? When models approach or exceed human-level capability in certain domains, human feedback may no longer provide sufficiently informative gradients for further improvement (Bowman, 2023; Burns et al., 2023). This raises a fundamental question: how can models continue to improve once they reach parity with their supervisors? Together, these limitations motivate the exploration of model self-improvement as a promising direction. Instead of relying exclusively on external human signals, models may leverage their own capabilities to generate data, evaluate outputs, and iteratively refine their policies.

Figure 1: **Vision of self-improved language models.** Humans only bootstrap the system, after which the model autonomously performs many operations such as acquiring data, reflecting on its outputs, and iteratively refining its capabilities to improve itself, potentially enabling the system to evolve beyond human-level intelligence.

From an automation perspective, *this direction is not only desirable but natural.* As LLMs become increasingly advanced, they have demonstrated the capacity to resolve complex engineering tasks and engage in high-level decision-making. Given that the development process of LLMs, including data acquisition, data selection, and model training, is itself a highly sophisticated engineering endeavor, it is a natural progression to delegate these responsibilities to the models themselves. By utilizing LLMs as intelligent *agents* to orchestrate their own development lifecycle, a "system-side" self-improvement loop is established. As shown in Figure 1, our vision is to shift from human-driven model development to a paradigm of autonomous self-improvement system, where LLMs continuously enhance their capabilities through self-directed iteration and feedback.

We define *self-improvement* of LLMs as a learning paradigm in which a model iteratively enhances its own capabilities without continuous human-in-the-loop supervision. This paradigm is characterized by two essential properties: **Autonomy:** the improvement process operates without ongoing human annotation or manual correction. "Self" does not imply the absence of external components; auxiliary modules such as teacher models, verifiers, critics, reward models, or automated evaluators may still be used. The key requirement is that the learning loop itself is fully automated once deployed. **Continuity:** self-improvement is not a one-off refinement. It is an iterative, self-reinforcing process in which outputs or experiences from earlier stages are reused to generate stronger supervision signals for subsequent updates. Each round of improvement depends on and amplifies prior results, enabling cumulative progress over time. Under this definition, self-improvement is not merely a technique for improving task-level metrics; it is a structural capability that enables sustained, autonomous growth. From the perspective of long-term AI development, such a capability is widely considered central to building systems that continuously learn and adapt beyond their initial training regime.

Motivated by the above vision, as demonstrated in Figure 2, we propose a lifecycle self-improvement system consisting of five interconnected components. The four components: **Data Acquisition**, **Data Selection**, **Model Optimization**, and **Inference Refinement**, jointly address a central question: To build an end-to-end self-improvement system, how can the model itself be leveraged at different stages to drive continuous and autonomous contribution? Specifically:

- *Data Acquisition:* The model autonomously collects or generates its training data.

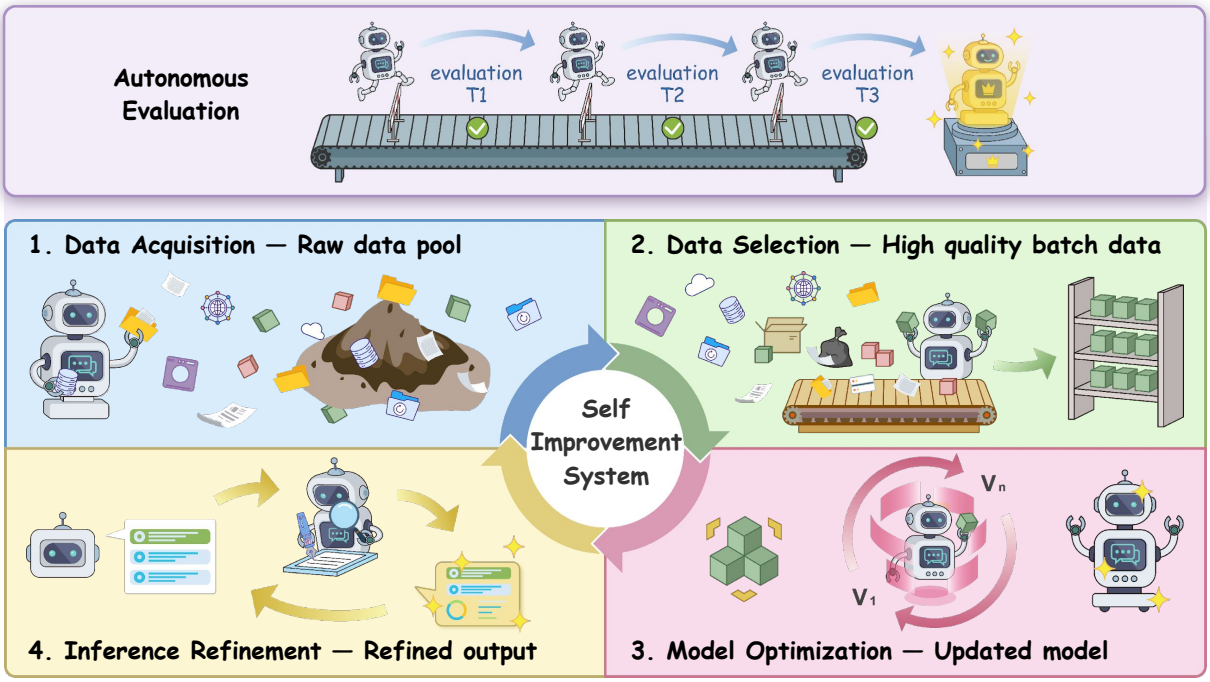

Figure 2: **Overview of the proposed self-improvement system.** The system consists of four interconnected stages: *(i) Data Acquisition*, *(ii) Data Selection*, *(iii) Model Optimization*, and *(iv) Inference Refinement*, through which the model autonomously improves its capabilities. An additional *(v) Autonomous Evaluation* module provides continuous feedback at various stages to monitor progress and guide long-term iterative improvement.

- *Data Selection:* The model independently evaluates and filters which data points are of higher quality and better suited for its own learning.
- *Model Optimization:* The model autonomously learns, effectively converting data into enhanced capabilities within its parameters.
- *Inference Refinement:* The model improves its own performance during the reasoning process without necessitating changes to its underlying parameters.

Beyond these four stages, the system further requires a mechanism for long-term measurement and guidance to ensure that self-improvement remains stable and sustainable. To this end, we introduce the fifth component, **Autonomous Evaluation**, which provides continuous feedback on the model's performance and helps steer its future development. Such a mechanism is essential, since static benchmarks quickly become outdated and human-driven evaluation does not scale with the system's growth. Through autonomous evaluation, the model can maintain timely, adaptive feedback and support sustained long-term improvement.

Together, these five components position the model as the core entity in an automated, iterative loop. This unified system ensures that improvement signals are consistently generated, filtered, applied, refined, and assessed, paving the way for broader system-level self-improvement of LLMs.

Several recent surveys have begun to examine self-improvement from different angles, reflecting the growth of this field. For example, Tao et al. (2024) focus on policy-level self-evolution through self-training and reinforcement learning, while Dong et al. (2024) review inference-time improvement techniques such as prompting and decoding refinement. Meanwhile, Fang et al. (2025a) and ang Gao et al. (2026) emphasize self-evolvement of agentic systems, highlighting memory, reflection, and tool-augmented interaction. Despite these efforts, most existing research still concentrates on localized mechanisms applied at specific stages, such

as training or inference, aiming to improve task-level performance, or focuses on peripheral components for agentic improvements. In contrast, we adopt a system-level perspective that conceptualizes self-improvement of the fundamental LLMs themselves as a unified, closed-loop lifecycle, integrating all stages of model development into a coherent end-to-end framework for scalable and autonomous evolution.

The remainder of this paper is organized into two main parts. First, from a technical perspective, we systematically study each component in the self-improvement system (from §2 to §6). For each stage, we begin with an overview to provide a high-level introduction, and then organize existing methods into structured categories, as shown in Figure 3. We further include a discussion at the end of each section to summarize key insights, as well as to analyze how each stage interacts with others and contributes to the overall self-improvement system. Second, we present a more general discussion of the overall self-improvement system (from §7 to §9), including challenges and limitations, applications, and future outlook. In these sections, we discuss the system from a broader perspective, beyond individual components. Similarly, the internal structure of each section is organized in a structured manner, as illustrated in Figure 9. In addition, although our paper is primarily centered on models, we also incorporate works and discussions on self-evolving agents. For example, we introduce agentic system-based improvement in the inference time in §5.4 and discuss self-evolving agents' applications across domains in §8. We argue that the transition from individual stages to a unified self-improvement system parallels the shift from standalone models to agentic systems, reflecting a shared trend toward more autonomous and interactive learning system-paradigms.

2 Data Acquisition for Self-Improvement

2.1 Overview

Within the self-improvement lifecycle, data acquisition is the process of leveraging the model to autonomously collect or generate the raw materials necessary for its own evolution. Two primary factors underscore the feasibility and necessity of shifting from traditional human-collected datasets toward this model-driven paradigm. First, in terms of operational efficiency, model-driven acquisition overcomes the temporal and financial constraints inherent in manual labor; unlike human curators, models can process and generate data 24/7, effectively bypassing the bottlenecks and high costs associated with human bandwidth. Second, and more critically, the intrinsic capabilities of modern LLMs have reached a threshold where the quality of model-generated signals is now highly competitive with human-curated content. In many specialized reasoning or high-complexity tasks, model-sourced data can even surpass human benchmarks in terms of logical consistency and fidelity (Gilardi et al., 2023; Bermejo et al., 2025), providing a superior foundation for continuous improvement. This dual advantage in both scale and quality empowers the model to serve as a self-sufficient engine for its own growth, dictating the scope and nature of the experiences it internalizes. To systematically analyze how models source these raw experiences, as shown in Figure 4, we categorize acquisition mechanisms into three tiers, reflecting a progressive increase in model autonomy and a corresponding decrease in reliance on existing data sources.

- **Static Curation:** The interaction between the model and the "Existing World". As an intelligent agent, the model navigates through massive internet snapshots or databases to autonomously filter out the raw corpora most valuable for its current evolution. In this stage, data is a "fixed stock," and the model's primary role is discovery.
- **Environment Interaction:** The interaction between the model and "Dynamic Tools." The model generates action trajectories by calling APIs, executing code, or operating within simulators, learning from the resulting feedback. Within this paradigm, data is no longer pre-existing; instead, it is "earned" by the model through a process of trial and error.
- **Synthetic Generation:** The interaction between the model and its "Inner Self". The model completely detaches from external environments, utilizing its intrinsic logic to produce entirely new reasoning chains or instructions. In this scenario, the model breaks free from the constraints of external data and begins to create something from nothing, generating experience through pure synthesis.

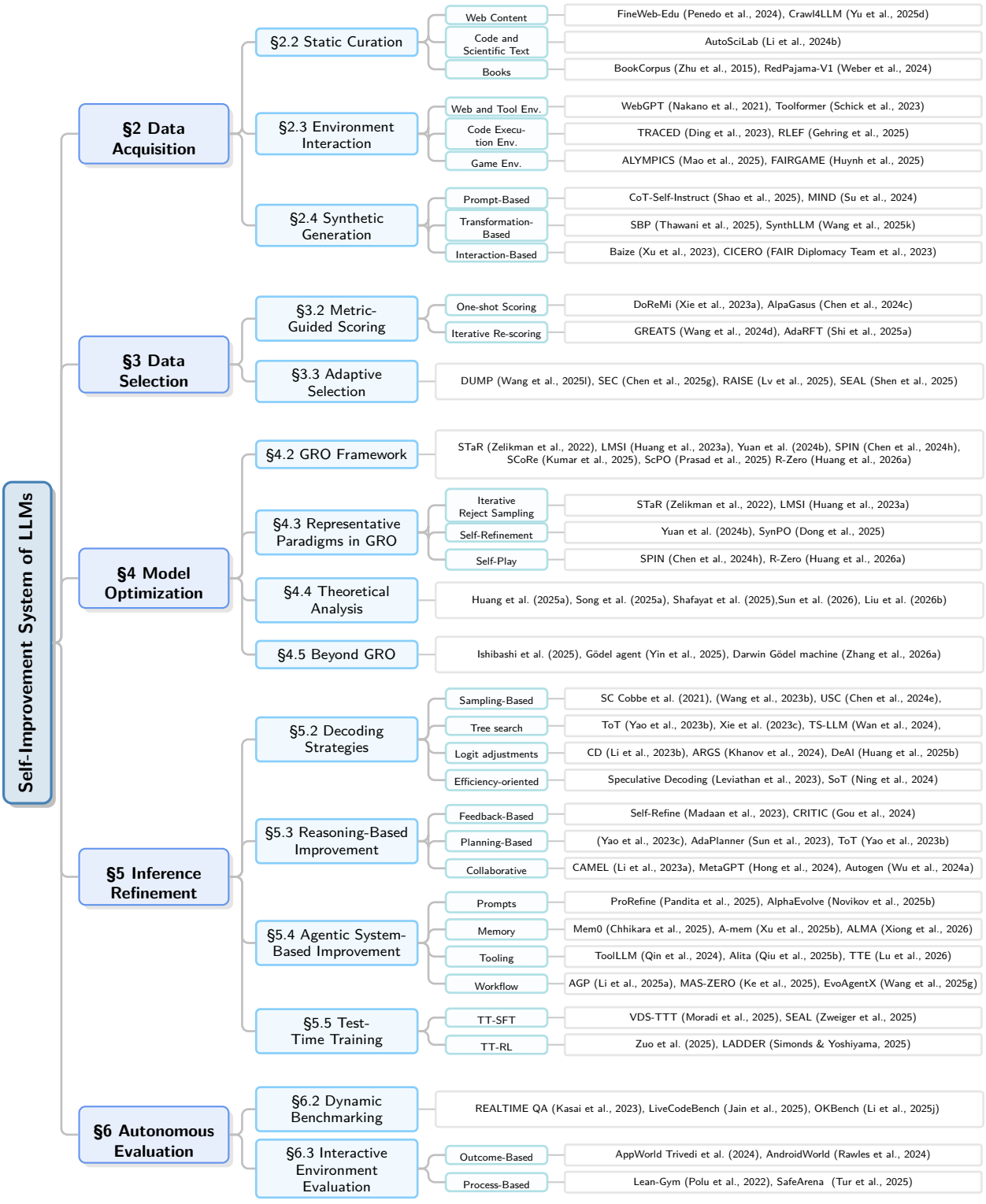

Figure 3: A taxonomy of the self-improvement system of LLMs.

This logical progression, moving from external discovery (curation) to external exploration (interaction) and finally to internal generation (synthesis), outlines the model-driven data acquisition trajectory as a comprehensive spectrum spanning both intellectual curation and exploration of external information sources

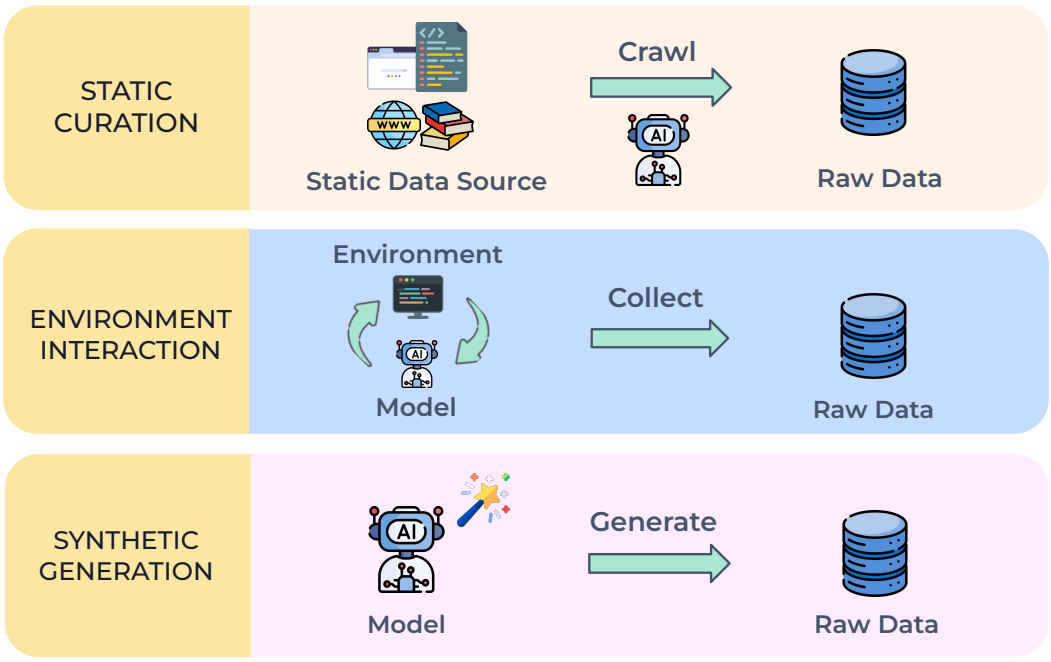

Figure 4: **Overview of data acquisition in the self-improvement system.** This stage focuses on how a model autonomously acquires raw data or experiences that can be used for self-improvement. *(i) Static Curation:* The model collects and filters information from pre-existing datasets or knowledge sources to construct curated training data. *(ii) Environment Interaction:* The model actively interacts with external environments, issuing actions and receiving observations to form interaction trajectories. *(iii): Synthetic Generation:* The model generates new training data directly from its own parameters through prompting, transformation, or multi-model interaction.

and autonomous generation of data from model internal capabilities. In the following section, we first introduce static curation in §2.2, where the model leverages existing data sources to construct training corpora. §2.3 then presents environment interaction, which enables the model to acquire data through actions from external environments. §2.4 focuses on synthetic generation, where the model produces new training data based on its own capabilities. Finally, §2.5 concludes with a discussion of their respective roles and trade-offs within the self-improvement system.

2.2 Static Curation

Static curation acquires raw data by retrieving content from fixed, externally hosted sources, where the model acts as an intelligent filter navigating massive repositories to discover the corpora most valuable for its own evolution. The core workflow begins with selecting one or more repositories (such as Common Crawl snapshots, code forges, or book collections) and then transforming the retrieved artifacts into a standardized training format. Traditionally, this pipeline has been driven entirely by heuristic rules, hand-written scripts, and tool-based filters. Foundational efforts such as C4 (Raffel et al., 2020), CCNet (Wenzek et al., 2020), RefinedWeb (Penedo et al., 2023), The Pile (Gao et al., 2020), Dolma (Soldaini et al., 2024), and RedPajama (Weber et al., 2024) established standard practices for web crawling, deduplication, language identification, and quality filtering at scale, collectively producing corpora ranging from hundreds of gigabytes to over one hundred trillion tokens. However, a growing body of work demonstrates that replacing or augmenting these heuristic pipelines with model-driven decisions, where an LLM itself selects, prioritizes, and filters data, produces substantially higher-quality corpora with less waste (Zhou et al., 2026b). We organize these emerging model-guided methods by source type and highlight how each contributes to the self-improvement paradigm.

Web Content. The majority of static curation targets the open web, including general web pages, encyclopedic sources such as Wikipedia, community platforms such as StackExchange and Reddit, and multilingual content across hundreds of languages. Traditional pipelines rely on heuristic filters (URL rules, profanity detection, sentence-count thresholds), n-gram language models, and deduplication tools such as MinHash to process raw Common Crawl snapshots into cleaned training corpora (Raffel et al., 2020; Wenzek et al., 2020; Penedo et al., 2023; Weber et al., 2024; de Gibert et al., 2024; Laurençon et al., 2022; Nguyen et al., 2024). While highly scalable, these approaches require extensive human expertise to design filtering rules and cannot adapt to downstream model needs.

A paradigm shift emerged with model-based quality filtering. FineWeb-Edu (Penedo et al., 2024) uses synthetic annotations from Llama-3-70B-Instruct to train a quality classifier that scores documents by educational value, filtering 1.3 trillion tokens down to 280 billion educationally valuable tokens and improving MMLU by 12% and ARC by 24% over unfiltered baselines. DCLM (DataComp for Language Models) (Li et al., 2024b) systematically demonstrates that model-based filtering is the single most important factor in assembling high-quality training sets, enabling a 7B parameter model trained on DCLM-Baseline to reach 64% 5-shot accuracy on MMLU with 40% less compute than the previous state of the art. Most recently, Crawl4LLM (Yu et al., 2025d) pushes model guidance even earlier in the pipeline, from filtering to crawling itself: rather than uniformly traversing the web graph and discarding most pages post-hoc, it uses a pretraining influence scorer to prioritize which URLs to crawl, achieving over 95% of oracle pretraining performance while crawling only 21% of the URLs.

These methods illustrate a clear progression in how models participate in web curation: from scoring already-crawled documents (Penedo et al., 2024), to systematically benchmarking model-driven filtering strategies (Li et al., 2024b), to guiding the crawl frontier itself (Yu et al., 2025d). For self-improvement, this trajectory suggests that stronger LLMs could autonomously compose end-to-end web curation pipelines, proposing which domains to crawl, designing quality classifiers tailored to their current capability gaps, and iteratively refining filtering rules to extend their knowledge boundary without human intervention.

Code and Scientific Text. Code corpora are typically sourced from public software forges such as GitHub and Software Heritage, while scientific text comes from repositories such as arXiv, PubMed Central, and Semantic Scholar. Traditional pipelines rely on metadata-based filtering (licenses, repository stars, citation counts) and format conversion tools (GROBID for PDFs, regex-based LaTeX extraction) rather than deep semantic analysis (Lozhkov et al., 2024; Weber et al., 2024; Lo et al., 2020; Paster et al., 2024; Gao et al., 2020; Soldaini et al., 2024). While highly scalable, these approaches are sensitive to licensing and documentation quality and cannot assess the semantic relevance of code or scientific content to a model's current training needs.

Model-guided curation of code and scientific text remains nascent but promising. Emerging AI-scientist frameworks demonstrate that LLM agents can autonomously navigate scientific literature, identify relevant papers, extract structured knowledge, and propose experimental designs (Lu et al., 2024b). In the code domain, models could parse repository metadata to identify high-quality projects, propose domain-specific subsets (for example, prioritizing security-critical code or emerging frameworks), and extract structured knowledge from papers such as theorems or experimental results. The metadata-driven nature of current pipelines makes them particularly amenable to agent-based composition and filtering, positioning code and scientific curation as a natural next frontier for model-driven static acquisition.

Books. Book corpora provide narrative structure and lexical richness. The Pile (Gao et al., 2020) includes two book subsets: Books3 (processed public-domain and freely available books) and Bibliotik (curated fiction and non-fiction), totaling approximately 100 GB. BookCorpus, used to pretrain the original GPT and BERT, comprises over 11,000 books scraped from self-publishing platforms (Zhu et al., 2015). RedPajama-V1 (Weber et al., 2024) incorporates 26 billion tokens from open book collections by statically crawling Project Gutenberg and similar archives. These pipelines typically apply minimal processing beyond format conversion (EPUB or PDF to plain text) and duplication removal.

Book curation involves copyright-sensitive decisions about which sources to include and how to balance genres and historical versus contemporary coverage. LLMs could help navigate licensing complexities, pro-

pose genre-balanced sampling strategies, and identify high-value content in emerging open-access archives. However, model-guided book curation remains largely unexplored, representing an open opportunity for self-improving systems.

Beyond curating which documents to include, a complementary line of work focuses on automating the preparation and transformation of raw data itself. Zhou et al. (2026b) survey this evolving landscape, characterizing the paradigm shift from rule-based, model-specific pipelines to prompt-driven, context-aware, and agentic preparation workflows. They organize the field into three major tasks: data cleaning (standardization, error correction, and imputation), data integration (entity matching and schema matching), and data enrichment (annotation and profiling). LLM-driven methods demonstrate improved generalization and semantic understanding compared to classical tools, yet face persistent challenges around hallucination and the cost of scaling to large corpora. For self-improving LLMs, this paradigm shift is particularly relevant: rather than relying on manually engineered cleaning rules, a model could autonomously identify data quality issues in its own pretraining corpus, propose corrections via structured prompts, and apply enrichment operations (such as generating metadata annotations or resolving entity references) to make raw collected data more useful as training signal.

Across all source types, the vast majority of static curation still relies on tool and script-based pipelines. Common tools include web crawlers, format converters such as GROBID (Lopez, 2009), language identifiers such as fastText (Joulin et al., 2017), and deduplication methods such as MinHash (Broder, 1997). Several common decision patterns emerge: (1) source selection, determining which repositories or archives to crawl; (2) filtering and quality thresholds, choosing among heuristic rules, learned classifiers, and metadata-based ranking; (3) format conversion and parsing; (4) deduplication and normalization; and (5) license and provenance tracking. Each of these decision points represents a task that could be surfaced as an action in a tool-augmented LLM agent (Yu et al., 2025d).

Despite this potential, several challenges limit LLM-guided static curation today. Trust and provenance concerns arise when models autonomously select training data, raising risks of data poisoning and unintentional bias amplification (Bender et al., 2021). System integration remains difficult because most existing pipelines are deeply embedded in institutional infrastructure. Furthermore, static curation lacks immediate feedback signals; models must rely on downstream pretraining performance to assess curation quality, which is expensive and slow. Nevertheless, emerging AI-scientist and data-engineer agent frameworks suggest this automation is increasingly feasible (Lu et al., 2024b; Zhou et al., 2026b). As LLMs become more capable at tool use, code generation, and long-horizon planning, the gap between manual and autonomous static curation will narrow, positioning static curation as the first component in a closed-loop self-improvement system where models actively expand their own pretraining data boundaries.

2.3 Environment Interaction

Environment interaction acquires raw data by letting a model *act to obtain information*, where the collected data includes not only content but also *interaction traces*: trajectories containing observations (for example, retrieved pages, tool outputs, environment states), actions (for example, search queries, clicks, API calls, code commands), and optional outcomes such as task success signals or execution feedback. This paradigm fundamentally differs from static curation in two ways: **(1) the action–observation loop**, where model actions causally determine what data is generated, creating temporal dependencies and causal structure that are absent in fixed corpora; and **(2) adaptive collection**, where exploration policies can target underrepresented domains or challenging tasks rather than passively accepting whatever pre-existing text is available. The model thus becomes an active participant in producing its own training data, turning environments into extensions of the training dataset and enabling closed-loop self-improvement.

We organize environment interaction methods by environment type, emphasizing how each domain's interaction mechanism shapes data collection and how these mechanisms can be leveraged by self-improving LLMs, as summarized in Table 1.

Web and Tool Environments. Web agents collect data through direct interaction with live websites and external APIs. While static web curation (for example, C4 (Raffel et al., 2020), FineWeb (Penedo et al.,

Table 1: **Environment interaction methods categorized by environment type.** We group representative methods based on the external environments they interact with, including web browsing, code execution, and game environments.

Environment	Methods
Web Browsing	WebGPT (Nakano et al., 2021), Toolformer (Schick et al., 2023), Go-Browse (Gandhi & Neubig, 2025), BrowserAgent (Yu et al., 2025e), InSTA (Trabucco et al., 2025), EnvScaler (Song et al., 2026)
Code Execution	TRACED (Ding et al., 2023), RLEF (Gehring et al., 2025), CWM (FAIR CodeGen team et al., 2025), CodeRL+ (Jiang et al., 2025d), AgentFounder (Su et al., 2025b) Learn-by-Interact (Su et al., 2025a)
Game Environments	Supervise Thyself (Racah & Pal, 2019), Generative Agents (Park et al., 2023), ALYMPICS (Mao et al., 2025), FAIRGAME (Huynh et al., 2025)

2024)) retrieves fixed snapshots of page content, web browsing and tool-use environments capture the full interactive process of searching, navigating, invoking functions, and synthesizing information across multiple sources. The resulting trajectories encode navigation strategies, multi-step reasoning, tool selection, and task completion patterns that are absent from static HTML dumps.

WebGPT (Nakano et al., 2021) fine-tunes GPT-3 by collecting trajectories from a text-based web browser where the model searches, navigates, and quotes passages to answer questions, using human demonstrations and preference feedback as supervision. Toolformer (Schick et al., 2023) teaches language models to autonomously invoke external APIs such as calculators, search engines, and translators through a self-supervised mechanism that inserts and evaluates candidate API calls; only those calls that reduce language modeling loss are retained, yielding an augmented corpus of tool-using text. Go-Browse (Gandhi & Neubig, 2025) applies structured exploration by framing data collection as graph search over web states, enabling efficient information reuse across exploration episodes and collecting ten thousand successful task-solving trajectories comprising forty thousand interaction steps. BrowserAgent (Yu et al., 2025e) builds an end-to-end browser-native framework that learns from real-time web interactions through fine-grained atomic operations such as scrolling, clicking, typing, and tab management, systematically generating training data from interactive search behaviors rather than relying on static snapshots or external summarization models. InSTA (Trabucco et al., 2025) introduces internet-scale data collection through a three-stage pipeline where an LLM annotates websites with candidate tasks, agents complete those tasks in live environments, and trajectories are filtered by judging task success, operating entirely without human annotation. EnvScaler (Song et al., 2026) programmatically constructs 191 synthesized tool-interactive environments with approximately seven thousand scenarios through automated synthesis, enabling multi-turn, multi-tool interactions where agents execute tools, observe state changes, and generate trajectories at scale.

For self-improvement, an LLM could iteratively search for questions it answers poorly, launch browsing episodes to collect supporting evidence, invoke APIs to verify factual claims, and then distill these trajectories into additional training data that improves its factual and procedural knowledge (Nakano et al., 2021; Trabucco et al., 2025; Schick et al., 2023).

Code Execution Environments. Code executors provide deterministic feedback through program execution, enabling models to ground learning in computational semantics. Unlike static code corpora such as The Stack v2 (Lozhkov et al., 2024) or RedPajama-V1 (Weber et al., 2024), which capture source text as written but lack any record of runtime behavior, code execution environments produce dynamic traces that encode variable states, branch coverage, and test outcomes as programs run. This distinction is critical: static code corpora provide syntactic and structural patterns, whereas execution environments reveal the causal relationship between code and its computational effects.

TRACED (Ding et al., 2023) collects execution traces by running programs in sandboxes, recording runtime variable values and branch coverage, then pretrains code models to predict these dynamic properties from static source text. RLEF (Gehring et al., 2025) trains code LLMs end to end via reinforcement learning to exploit unit test feedback over multiple turns, achieving state-of-the-art results on competitive programming with both 8B and 70B models while reducing required samples by an order of magnitude. Code World Models (CWM) (FAIR CodeGen team et al., 2025) incorporate computational trajectories into mid-training, allocating tokens specifically to interactions with code execution environments and integrating execution feedback directly into the pretraining curriculum. CodeRL+ (Jiang et al., 2025d) advances this by jointly optimizing code generation and execution semantics alignment, where failed exploration programs are repurposed to infer variable-level execution trajectories, providing dense learning signals that bridge the gap between textual fluency and execution correctness.

These approaches show how models can iteratively probe a code executor, collect rich traces, and reuse them as pretraining or continual training data. Notably, AgentFounder (Su et al., 2025b) and Learn-by-Interact (Su et al., 2025a) demonstrate that such environment interaction data can be embedded directly into continual pretraining phases rather than reserved solely for downstream fine-tuning, allocating dedicated training stages to agent trajectory data and improving sample efficiency when models are later adapted to specific tasks. In a self-improvement setting, an LLM could autonomously identify failure patterns in its own code generations, design new test cases, and schedule further execution queries to close capability gaps in targeted languages or libraries (Gehring et al., 2025; Jiang et al., 2025d).

Game Environments. Game environments provide structured settings for strategic and social interaction data. Supervise Thyself (Racah & Pal, 2019) examines self-supervised learning where agents observe the results of their actions in interactive game environments to learn representations that generalize to novel settings without explicit reward signals. Park et al. (2023) demonstrate that believable social interactions can unfold between multiple LLM-driven characters in simulated game worlds, producing realistic dialogue transcripts and action logs that can be mined as training examples for social reasoning. ALYMPICS (Mao et al., 2025) introduces a systematic framework that facilitates game-theoretic interactions, where LLMs compete or cooperate in auction-based resource allocation games, generating strategic dialogue and decision trajectories. FAIRGAME (Huynh et al., 2025) provides a modular framework for simulating repeated game-theoretic interactions such as the Prisoner's Dilemma and the Public Goods Game between LLMs, producing trajectories that encode strategic cooperation, defection, and social reasoning patterns.

These game-based systems generate interaction data that capture coordination, negotiation, and strategic planning, which are difficult to obtain from static text alone. For self-improvement, a language model could instantiate multiple copies of itself as players and critics, generate increasingly complex scenarios, and selectively add informative trajectories to its training set, thereby sharpening its social and strategic reasoning capabilities (Park et al., 2023; Mao et al., 2025).

2.4 Synthetic Generation

Synthetic generation represents a complementary paradigm to static curation and environment interaction, where models themselves produce training data at scale without environmental feedback or human supervision. Unlike environment interaction, which relies on external systems to provide observations and rewards through action–observation loops, synthetic generation produces data through generative processes guided by carefully engineered prompts, constraints, or diversity mechanisms. In this approach, an LLM (or smaller task-specific model) creates new training examples through prompting, generation templates, or learned procedures, yielding corpora that did not exist in any prior dataset. This paradigm shifts the bottleneck from data availability to *synthesis quality and diversity*: the challenge is ensuring that generated data maintains sufficient diversity, coherence, and informativeness to serve as effective training signals without inducing model collapse or distributional drift.

We organize synthetic generation methods into three paradigms based on how data is produced, as summarized in Table 2. **Prompt-based** methods generate data from scratch by prompting an LLM, either through direct prompting with topic and format specifications or through seed expansion where a small set of seed examples is iteratively amplified into a large corpus. **Transformation-based** methods take an ex-

Table 2: **Synthetic data generation methods categorized by generation mechanism.** We group representative methods based on how synthetic data is produced, including prompt-based, transformation-based, and interaction-based approaches.

Generation Mechanism	Methods
Prompt-Based	TinyStories (Eldan & Li, 2023), Phi-1 (Gunasekar et al., 2023), Phi-1.5 (Li et al., 2023c), Phi-3 (Abdin et al., 2024), Phi-4 (Microsoft Research, 2024), Self-Instruct (Wang et al., 2023c), AttrPrompt (Yu et al., 2023), WizardLM (Xu et al., 2024a), CodecLM (Wang et al., 2024i), CoT-Self-Instruct (Shao et al., 2025), Cosmopedia (Ben Allal et al., 2024), MIND (Su et al., 2024), Constraint-Based (Fedoseev et al., 2024)
Transformation-Based	WRAP (Maini et al., 2024), Instruction Pre-Training (Hsieh et al., 2024), SBP (Thawani et al., 2025), SynthLLM (Wang et al., 2025k), Gradient Matching (Wang et al., 2025a)
Interaction-Based	Shah et al. (2018), Baize (Xu et al., 2023), CICERO (FAIR Diplomacy Team et al., 2023), SPIN (Chen et al., 2024h), ALAS (Atreja, 2025), LSP (Kuba et al., 2025), R-Zero (Huang et al., 2026a), Multi-Agent Dialogues (Ueda et al., 2025), Math Gen (Wan et al., 2025)

isting corpus as input and use an LLM to rewrite, reformat, or extract new training examples (for example, converting raw web text into question-answer pairs or rephrasing documents into higher-quality formats). **Interaction-based** methods require multiple model instances (or roles) to interact with each other, generating data through self-play, debate, or multi-agent collaboration without external environmental feedback. Throughout, we emphasize both the mechanisms by which data is generated and how these methods enable self-improvement by allowing models to expand their own training distributions.

2.4.1 Prompt-Based Generation

Prompt-based methods generate synthetic data from scratch by providing an LLM with carefully designed instructions specifying the desired topic, format, audience, or task. We distinguish two sub-modes within this paradigm: *direct prompting*, where the model generates content from topic and format specifications alone, and *seed expansion*, where a small set of seed examples or instructions is iteratively amplified into a large corpus.

Direct Prompting. TinyStories (Eldan & Li, 2023) demonstrated that coherent language learning emerges in models with fewer than ten million parameters when trained exclusively on GPT-3.5 and GPT-4 generated stories constrained to child-level vocabulary, using constrained word lists and combinatorial sampling over topics and characters to ensure diversity across half a billion tokens. Phi-1 (Gunasekar et al., 2023) pioneered the "textbook quality" approach by generating less than one billion tokens of Python tutorials and coding exercises using GPT-3.5, guided by prompts that specify educational structure, target audience, and domain coverage; combined with six billion tokens of filtered web code, a 1.3 billion parameter model achieved 50.6% accuracy on HumanEval, matching or exceeding models three times larger. Phi-1.5 (Li et al., 2023c) extended this to natural language and common-sense reasoning by generating twenty billion tokens of diverse synthetic textbooks seeded from twenty thousand curated topics. Subsequent releases confirmed the scaling pattern: Phi-3 (Abdin et al., 2024) series models consistently matched much larger competitors by mixing filtered web text with synthetic content throughout pretraining, and Phi-4 (Microsoft Research, 2024), trained almost entirely on GPT-4-generated text, achieved performance comparable to or exceeding its teacher model on some reasoning benchmarks. AttrPrompt (Yu et al., 2023) explores diversity through attributed prompts that specify length, style, domain, and other properties rather than simple class-conditional prompts, achieving

high diversity with only 5% of ChatGPT's querying cost by explicitly controlling generation attributes. Cosmopedia (Ben Allal et al., 2024), the largest open synthetic dataset with twenty-five billion tokens across thirty million files, was generated using Mixtral-8x7B-Instruct by prompting the model to produce textbooks, blog posts, stories, and WikiHow-style articles across multiple domains, demonstrating that large synthetic corpora can be produced reproducibly at scale. MIND (Su et al., 2024) generates synthetic math dialogue corpora by prompting models to produce step-by-step problem-solving conversations grounded in OpenWebMath content, boosting mathematical reasoning by 13.4% on GSM8K and 4.3% on MMLU-STEM. Constraint-Based Synthetic Data Generation (Fedoseev et al., 2024) uses Satisfiability Modulo Theories solvers to generate synthetic mathematical problems and solutions, ensuring that problems satisfy formal constraints and that solutions are verifiable.

Seed Expansion. Self-Instruct (Wang et al., 2023c) generates instruction-response triples by prompting a pretrained model with 175 seed tasks, sampling new instructions, generating responses, and filtering for quality, producing over 52,000 instructions and demonstrating a 33% absolute improvement over vanilla GPT-3 on SuperNaturalInstructions. WizardLM (Xu et al., 2024a) starts from small sets of human-written instructions and iteratively uses ChatGPT to increase their complexity through "Evol-Instruct," which applies operations such as deepening, broadening, and adding constraints. CodecLM (Wang et al., 2024i) encodes seed instructions into metadata (for example, task type, domain, difficulty), then decodes them back into task-specific synthetic examples by prompting an LLM conditioned on the metadata. CoT-Self-Instruct (Shao et al., 2025) extends the paradigm with chain-of-thought reasoning by prompting the model to generate step-by-step reasoning chains before final answers, producing complex examples that improve both mathematical reasoning and general instruction-following.

These prompt-based methods, whether through direct prompting or seed expansion, illustrate how an LLM can serve as both the generator and the beneficiary of synthetic training data. For self-improvement, the model could autonomously propose new topic lists based on downstream evaluation gaps, refine generation prompts to target weak domains, and iteratively expand its training corpus without relying on external data sources.

2.4.2 Transformation-Based Generation

Transformation-based methods take an existing corpus as input and use an LLM to rewrite, reformat, or extract new training examples from it. Rather than generating content from scratch, these methods leverage the semantic content of existing data while improving its quality, structure, or task alignment.

WRAP (Web Rephrase Augmented Pretraining) (Maini et al., 2024) uses instruction-tuned LLMs to paraphrase web documents into diverse formats such as Wikipedia-style articles or question-answer pairs; jointly pretraining on original and rephrased web data accelerates convergence approximately three-fold while improving zero-shot accuracy. Instruction Pre-Training (Hsieh et al., 2024) synthesizes two hundred million instruction-response pairs from pretraining corpora using open LLMs by prompting with diverse task templates, enabling an eight billion parameter Llama model to rival or exceed a seventy billion parameter baseline on many benchmarks. SBP (Synthetic Bootstrapped Pretraining) (Thawani et al., 2025) learns inter-document relationships from a seed corpus and uses them to generate an entire new corpus of documents, enabling generation of up to one trillion tokens that consistently improve over baselines while recovering approximately 60% of the gains that would otherwise require twenty times more unique natural data. SynthLLM (Wang et al., 2025k) automatically generates large-scale synthetic question-answer datasets from pretraining corpora through concept extraction and recombination, revealing that synthetic data follows predictable scaling laws and can substitute for billions of real tokens before saturation. Gradient Matching (Wang et al., 2025a) formalizes synthesis as a learning problem by proposing a gradient-matching algorithm that optimizes synthetic examples to mimic real training gradients, yielding provably convergent synthetic examples that allow LLMs to reach the same solution as using original data.

These transformation methods show how models can reprocess existing corpora into higher-quality training data. In a self-improving LLM, this capability enables the model to revisit its own pretraining corpus, identify low-quality or redundant segments, and synthesize improved versions that better serve its learning objectives.

2.4.3 Interaction-Based Generation

Interaction-based methods require multiple model instances or roles to interact with each other, generating data through self-play, debate, or multi-model collaboration without external environmental feedback. These represent the most autonomous form of synthetic generation: models create their own training curricula by competing, cooperating, and critiquing their own outputs.

Shah et al. (2018) demonstrated that conversational agents could be bootstrapped via self-play dialogues, where two instances of the agent simulate task-oriented conversations to generate synthetic training data, yielding fully annotated dialogues without a large human-written corpus. Xu et al. (2023) use model self-chat to generate multi-turn dialogues where the model is prompted to play both user and assistant roles, generating synthetic dialogues that demonstrate strong conversational ability from entirely model-generated interactions. CICERO (FAIR Diplomacy Team et al., 2023), trained for strategic Diplomacy gameplay, was fine-tuned on millions of messages from games where model instances negotiated with each other, recording negotiation dialogues and action sequences and filtering trajectories that exhibit human-compatible negotiation tactics. SPIN (Self-Play Fine-Tuning) (Chen et al., 2024h) utilizes a self-play mechanism where the LLM generates its own training data from previous iterations, refining its policy by discerning self-generated responses from human-annotated data and progressively elevating the LLM without additional human data. ALAS (Atreja, 2025) constructs an entire learning curriculum by querying the web, synthesizing question-answer pairs from retrieved content, and continuously fine-tuning itself, combining retrieval-augmented generation with self-critique where the model evaluates its own generated questions and answers before adding them to the training set. LSP (Language Self-Play) (Kuba et al., 2025) introduces a competitive game-theoretic framework where a single LLM iteratively improves by playing Challenger and Solver roles against itself, generating increasingly difficult queries and learning to solve them through reinforcement learning without external training data. R-Zero (Huang et al., 2026a) presents a reinforcement-learning-driven self-play framework where a model iteratively generates challenges and solves them, demonstrating that pretrained LMs can improve without external data via interactive self-play curricula.

Multi-model dialogue generation further extends interaction paradigms. Research on multi-model LLM dialogues demonstrates that enlarging cohorts, deepening interaction depth, and broadening persona heterogeneity each enrich the diversity of generated ideas, with increasing critic-side diversity within ideation–critique–revision loops boosting the feasibility of final proposals (Ueda et al., 2025). Math problem generation benefits from dual mechanisms: self-play combined with multi-model cooperation generates synthetic math problems through continual learning cycles of supervised fine-tuning, data synthesis, and direct preference optimization (Wan et al., 2025).

For self-improvement, interaction-based methods enable an LLM to iteratively challenge itself, discover failure modes, and generate corrective examples without relying on external data sources or human supervision. Moreover, some of these methods (for example, R-Zero, SPIN, LSP) directly connect to policy optimization through reinforcement learning loops, illustrating how synthetic generation can serve as both a data acquisition and policy refinement mechanism.

2.5 Discussion

The three acquisition pathways described above—static curation, environment interaction, and synthetic generation—occupy complementary positions in the data landscape. Rather than competing alternatives, they form a layered system in which each pathway addresses limitations of the others. We discuss the trade-offs that govern their use, how their roles shift across the training lifecycle, and how they combine to enable autonomous data acquisition for self-improving LLMs.

Trade-offs. Each pathway presents characteristic trade-offs that pipeline designers must balance.

- **Scale versus specificity.** Static curation offers massive scale (hundreds of trillions of tokens (Weber et al., 2024)) but limited targeting; content reflects what exists on the open web. Synthetic generation offers precise control over topic, difficulty, and format, but requires careful diversity management to avoid redundancy and distributional narrowing (Zeng et al., 2024; Ma et al., 2025b). Environment

interaction offers high specificity—model actions causally determine what data is generated—but at significant execution cost per trajectory.

- **Quality versus cost.** Synthetic data from strong teacher models (for example, GPT-4) produces high-quality outputs but incurs substantial inference costs, while generation from open models (for example, Mixtral, Llama) is cheaper but may yield lower-fidelity examples (Ben Allal et al., 2024). Static curation is cheap per token after initial infrastructure investment, but filtering for quality via learned classifiers (Penedo et al., 2024; Yu et al., 2025d) adds computational overhead. Environment interaction incurs the highest per-sample cost due to runtime execution, but the resulting trajectories carry dense, verifiable learning signals.

- **Diversity versus coherence.** Synthetic corpora risk distributional narrowing if generation prompts or seed topics are insufficiently varied, potentially leading to model collapse (Dohmatob et al., 2024). Maintaining diversity requires explicit mechanisms such as entity grounding (Ma et al., 2025b), attributed prompts (Yu et al., 2023), and periodic infusions of real-world data. Static corpora naturally capture the diversity of the open web but include noise, toxicity, and duplicates that must be filtered.

- **Substitutability.** An important question is whether one pathway can replace another. BeyondWeb (Lo et al., 2025) and the Phi model family (Gunasekar et al., 2023; Microsoft Research, 2024) demonstrate that carefully synthesized data can partially substitute for large-scale web crawls, achieving comparable or superior performance with substantially fewer tokens. However, Lei et al. (2025) show that book-style synthetic data alone exhibits model-collapse patterns at small data budgets, whereas rephrased synthetic data mixed at approximately 30% with natural web text accelerates convergence five to tenfold without degradation. Environment interaction data is largely non-substitutable: the causal structure of execution traces (Ding et al., 2023; FAIR CodeGen team et al., 2025), browsing trajectories (Nakano et al., 2021; Trabucco et al., 2025), and tool-use logs (Schick et al., 2023) cannot be replicated through static text or pure generation, because the data must reflect genuine interactions with external systems. In practice, the strongest training pipelines combine all three pathways rather than relying on any single one.

Shifting Roles Across the Training Lifecycle. The three pathways do not contribute equally at every stage; rather, their relative importance shifts as the model progresses from pretraining through post-training. Understanding this progression is essential for designing effective data acquisition strategies.

During early pretraining, static curation dominates because sheer token volume and broad linguistic coverage are the primary requirements (Raffel et al., 2020; Penedo et al., 2024; Weber et al., 2024). The goal at this stage is to establish a general-purpose language foundation across diverse domains and genres. As pretraining progresses and the model acquires basic language competence, synthetic generation becomes increasingly valuable for injecting structured, high-quality supervision—such as book-style content, rephrased web data, or reasoning-intensive examples—that accelerates convergence and strengthens targeted capabilities (Gunasekar et al., 2023; Maini et al., 2024; Li et al., 2023c). Recent evidence reinforces this phase: front-loading reasoning-intensive data in pretraining yields a 19% average gain on downstream benchmarks, establishing foundational capabilities that post-training alone cannot recover (Akter et al., 2025). Moreover, pretraining benefits most from broad diversity in reasoning patterns, while supervised fine-tuning is more sensitive to data quality (Akter et al., 2025), suggesting that different data acquisition strategies are optimal at different stages.

During continual pretraining and post-training, environment interaction takes on a larger role, providing the grounded, task-specific trajectories needed for tool use, web navigation, code execution, and strategic reasoning (Su et al., 2025b;a; Gehring et al., 2025). Instruction bootstrapping and self-play methods then refine behaviors for alignment and preference optimization (Wang et al., 2023c; Chen et al., 2024h). This lifecycle perspective suggests that a self-improving LLM must not only have access to all three pathways but must also orchestrate them in a curriculum-aware manner, shifting the data mix from broad coverage to targeted synthesis to grounded interaction as its capabilities mature.

Toward Autonomous Data Acquisition. From the perspective of self-improving LLMs, data acquisition is the stage where the model expands its own knowledge boundary. The three pathways described in this section collectively provide the mechanisms for this expansion, and the emerging trend is toward systems that orchestrate all three autonomously.

In the simplest form, an LLM can direct its own static curation by autonomously discovering and prioritizing new web sources, code repositories, or scientific archives based on identified knowledge gaps, as demonstrated by LLM-guided crawling approaches (Yu et al., 2025d). Moving beyond passive retrieval, environment interaction enables the model to generate its own grounded training data through active exploration, turning web browsing (Trabucco et al., 2025; Gandhi & Neubig, 2025), code execution (Gehring et al., 2025; Jiang et al., 2025d), and tool invocation (Schick et al., 2023) into data-producing actions whose outcomes directly inform what the model learns next. Synthetic generation further enables the model to fill identified capability gaps by producing targeted training examples, with self-play mechanisms such as R-Zero (Huang et al., 2026a) and Language Self-Play (Kuba et al., 2025) demonstrating purely autonomous improvement without any external data. Recent systems such as ALAS (Atreja, 2025) illustrate how these mechanisms can be composed: the model generates a learning curriculum for a target domain, retrieves up-to-date information from the web, distills it into question-answer training data, fine-tunes itself through supervised fine-tuning and direct preference optimization, and iteratively revises the curriculum based on evaluation results.

Together, these three pathways form the input layer of a closed-loop self-improvement system. The envisioned pipeline operates as follows: the model continuously assesses its own weaknesses through evaluation, decides which acquisition pathway to invoke (crawl new sources, explore an environment, or synthesize targeted examples), collects new data accordingly, and feeds it into downstream selection and training stages. This positions autonomous data acquisition not as a one-time preprocessing step, but as a persistent, model-driven process that runs throughout the lifetime of a self-improving system, enabling the model to continuously expand the scope and quality of its own training distribution.

3 Data Selection for Self-Improvement

3.1 Overview

Within the lifecycle of a self-improvement system, data selection serves as the critical bridge between raw data acquisition and model optimization. Selection governs the screening mechanisms that decide which specific data samples are filtered and prioritized for the model to learn from. Historically, this stage relied on static filtering, where human-defined heuristics, such as language or script filtering (Wenzek et al., 2020; Gao et al., 2020), deduplication and near-duplicate removal (Brown et al., 2020), elimination of templated or non-linguistic content (Raffel et al., 2020; Gao et al., 2020), and coarse constraints like minimum-length thresholds or domain blocklists (Raffel et al., 2020), were applied. However, these methods are fundamentally model-agnostic; they cannot perceive the model's current internal state nor adapt as the model evolves, leading to a lack of both automation in signal generation and continuity across iterations.

To achieve a fully autonomous self-improvement loop, data selection must shift toward a model-driven paradigm, leveraging the model's own internal states to curate its training signals. As shown in Figure 5, this transition unfolds across two progressively sophisticated regimes:

- **Metric-Guided Scoring** transforms the model from a passive recipient into an active evaluator by replacing fixed heuristic rules with model-derived signals, such as loss, perplexity, or scores from automated evaluators. Instead of relying on handcrafted filters, the system leverages the model itself to assess data quality and utility. These scores may be computed once (one-shot) or updated alongside the model's evolving state (iterative), with the latter enabling more continuous alignment between selection and learning needs.
- **Adaptive Selection** represents the higher level evolution of this process, where a dedicated selector acts as a "strategist" by optimizing the selection logic itself. Here, data selection is no longer a fixed preprocessing step but a parameterized, learnable task. Using frameworks such as reinforcement learning, bandit optimization, or bilevel optimization, the system trains this selector to discover

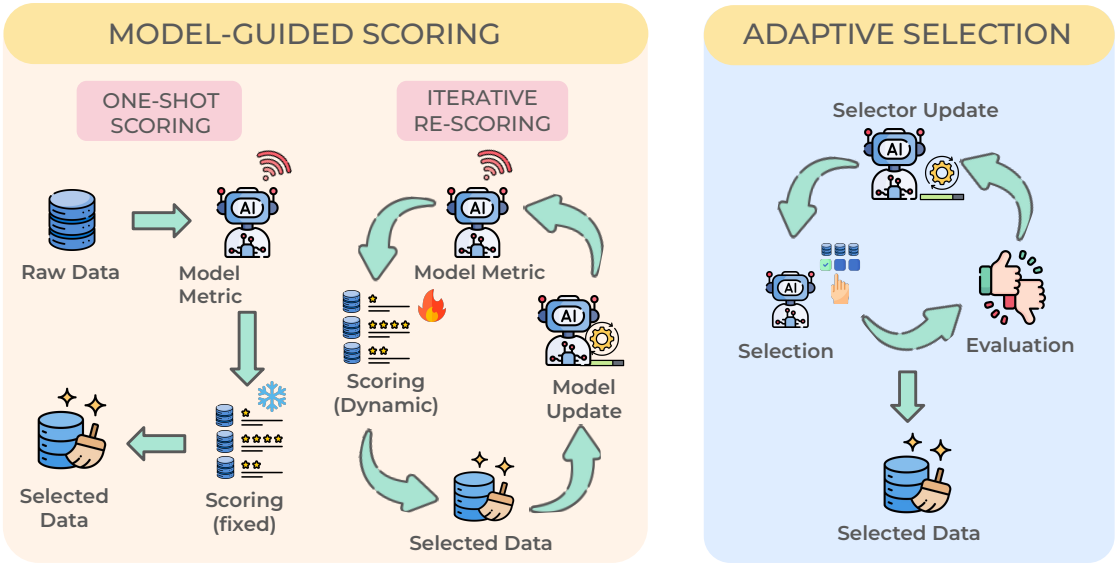

Figure 5: **Overview of data selection in the self-improvement system.** This stage focuses on how a model evaluates and selects high-quality data from raw data pools to support effective self-improvement. *(i) Metric-Guided Scoring:* The model applies predefined scoring metrics derived from model signals to rank and filter data. One-shot scoring computes scores once before training, while iterative re-scoring periodically refreshes scores as the model evolves. *(ii) Adaptive Selection:* A learnable selector dynamically chooses training data through an iterative loop of selection, evaluation, and selector update, continuously adapting the training distribution based on feedback.

> which data distributions or sample compositions maximize performance gains. By optimizing the selection policy end-to-end toward a downstream objective, the system moves beyond fixed decision rules and learns to "shape its own curriculum."

This progression from model-metric scoring to strategic adaptation closely parallels the shift from "external curation" to "self-synthesis" in data acquisition. It ensures that collected data is not merely accumulated but actively refined to match the model's evolving capabilities, moving toward the automated, continuous feedback loop envisioned for self-improving systems. In the following section, §3.2 presents metric-guided scoring, where data is evaluated using model-derived signals such as loss or uncertainty. §3.3 focuses on adaptive selection, where the selection strategy itself is learned and updated to optimize downstream performance. Finally, §3.4 concludes with a discussion of how selection influences training dynamics within the self-improvement system.

3.2 Metric-Guided Scoring

Metric-guided scoring uses model-derived signals to score candidate data while keeping the decision rule itself fixed (i.e., not learned or updated). Signals may originate from model internals (e.g., loss, perplexity, uncertainty) or external automated evaluators (e.g., verifiers, critics, reward models). Selection can be applied either **offline in a one-shot manner** or **online through iterative re-scoring**, and may produce a filtered subset, sample weights, or dynamically selected mini-batches. This regime offers strong automation, as scoring relies on model-generated signals rather than human-designed heuristics. Its degree of continuity, however, depends on when selection is applied: one-shot scoring provides limited continuity, whereas iterative re-scoring enhances continuity by repeatedly updating scores as the model evolves. Accordingly, we categorize metric-guided scoring methods into one-shot and iterative variants. Table 3 summarizes representative methods according to their scoring signal and selection setting.

3.2.1 One-Shot Scoring

One-shot scoring refers to a metric-guided scoring setting in which selection decisions are computed once prior to the target model's optimization and remain unchanged throughout training. A scoring function assigns a utility value to each candidate unit—which may be an individual instance, a pre-defined group of instances, or an entire domain/source. The final training set is then constructed by filtering, ranking, or reweighting according to these scores (Chen et al., 2024c; Li et al., 2024j; Xie et al., 2023b). Here, "offline" is defined relative to the target optimization loop: although the scoring process may use a frozen reference model or a separately trained proxy model, the resulting subset or weights are fixed and not updated during the subsequent training of the target model.

Representative one-shot scorers mainly differ in what their fixed utility signal is intended to approximate, while the resulting scores can be applied at different granularities (e.g., instances or domains) to produce a filtered subset or sampling weights. *Likelihood-based* signals use perplexity and related model-fit measures as proxies for text quality and utility, enabling offline pruning and ranking of candidate data (Marion et al., 2023; Ankner et al., 2025); this signal can also be adapted to measure how useful a candidate instruction is as an in-context example, by evaluating how much it reduces anchor-set perplexity when used as a demonstration (Li et al., 2024j), and can also be aggregated at the domain-level via perplexity–benchmark correlations to prioritize domains with higher downstream payoff (Thrush et al., 2025). *Loss- and training-dynamics-based* scorers treat utility as learnability captured by training dynamics, leveraging proxy loss-trajectory similarity to form gradient-representative subsets (Yang et al., 2024c) and learning domain mixture weights by minimizing worst-case excess loss before resampling (Xie et al., 2023a). Gradient- and influence-based approaches approximate a sample's marginal effect on target-task validation loss via first-order gradient alignment, selecting examples whose update directions align with validation gradients (Xia et al., 2024b), using influence-preserving proxies to make gradient-based selection scalable (Chen et al., 2026).

Difficulty- and hardness-based signals operationalize utility as selecting challenging yet informative instructions, using instruction-following difficulty scores (Li et al., 2024e), learning-percentage-based hardness estimated by smaller models (Mekala et al., 2024), or token-level informativeness and neighborhood consistency for robust scoring (Fu et al., 2025). *Uncertainty-based* scorers rely on intrinsic predictive uncertainty for self-filtering of instruction data (Liu et al., 2024d), while graph-based approaches further integrate uncertainty into influence maximization to model dependencies among examples (Han et al., 2025). *Similarity-based* selection treats utility as relevance to a target distribution or task, retrieving cross-task nearest neighbors (Ivison et al., 2023) or performing feature-space importance resampling for large-scale distribution matching (Xie et al., 2023b). Finally, *external-evaluator-based* scoring leverages separate quality/safety judges—including text-quality classifiers in large-scale pretraining pipelines (Du et al., 2022; Chowdhery et al., 2023) and strong LLM filters or multi-criteria alignment judgments for instruction data (Chen et al., 2024c; Liu et al., 2024g).

Compared to static filters, one-shot scoring provides stronger automation, as scoring and selection are performed using model-derived metrics. However, its continuity remains limited, since selection decisions are made only once and remain fixed throughout training. Similar to static filters, this regime is attractive at scale due to its low computational overhead and operational stability. Its primary limitation lies in adaptivity: a single utility estimate may become outdated or misaligned with the model's evolving competence and training objectives. This limitation motivates methods that adopt iterative re-scoring, which repeatedly update selection signals during training to better track the model's learning dynamics.

3.2.2 Iterative Re-Scoring

Iterative re-scoring brings metric-guided scoring selection into the target training loop: scores are recomputed repeatedly using updated model states and the same non-learned decision rule is re-applied to shape subsequent training batches (Song et al., 2020; Wang et al., 2024d). In this setting, the rule remains fixed, but the selected subset changes over time because the scoring signal changes as training progresses. Consequently, automation remains strong, while continuity is strengthened because selection is explicitly re-applied across iterations.

Table 3: **Metric-guided scoring methods based on model metrics.** This table summarizes representative methods according to the metrics they use for data selection (e.g., perplexity, loss, gradient, difficulty, uncertainty, similarity, and external evaluators), and distinguishes two settings in metric-guided scoring: one-shot scoring (offline) and iterative re-scoring (online).

Model Metric	**One-Shot Scoring (Offline)**	**Iterative Re-scoring (Online)**
Perplexity	Marion et al. (2023), NUGGETS (Li et al., 2024j), Perplexity-based Pruning (Ankner et al., 2025), Perplexity Correlations (Thrush et al., 2025)	SST (hattami et al., 2025), PREPO (Sun et al., 2025b)
Loss	DoReMi (Xie et al., 2023a), S2L (Yang et al., 2024c)	RHO-LOSS (Mindermann et al., 2022)
Gradient	LESS (Xia et al., 2024b), IProX (Chen et al., 2026)	GREATS (Wang et al., 2024d), LearnAlign (Li et al., 2025d)
Difficulty	Li et al. (2024e), Learning Percentage (Mekala et al., 2024), T-SHIRT (Fu et al., 2025)	IT2ACL (Huang & Xiong, 2024), P3 (Yang et al., 2024b), AdaRFT (Shi et al., 2025a), AdaSTaR (Koh et al., 2025), Bae et al. (2026)
Uncertainty	SelectIT (Liu et al., 2024d), UniMax (Han et al., 2025)	Recency Bias (Song et al., 2020), Active Instruction Tuning (Kung et al., 2023)
Similarity	DEFT (Ivison et al., 2023), DSIR (Xie et al., 2023b)	DiverseEvol (Wu et al., 2023), Balanced LSH (Phan et al., 2025)
External Evaluator	GLaM (Du et al., 2022), PaLM (Chowdhery et al., 2023), AlpaGasus (Chen et al., 2024c), DEITA (Liu et al., 2024g)	IterSelectTune (Song et al., 2024), LANCE (Wang et al., 2025c), Auto-CEI (Zhao et al., 2025d)

Representative iterative re-scoring methods primarily differ in how refreshed scores are injected into the optimization loop, even when the scoring rule itself remains fixed. Broadly, re-scored utilities can be applied as **soft control**—by continuously reshaping sampling probabilities or pacing—or as **hard control**—by filtering, skipping, or pruning low-utility items at each round.

For *soft control*, several works repeatedly recompute likelihood-style signals and use them to pace exposure to data as the model evolves, including perplexity-guided curricula and sampling schedules during training or RL-style post-training (hattami et al., 2025; Sun et al., 2025b). Loss-based online prioritization instead re-evaluates utility in terms of generalization improvement, preferentially training on points estimated to most reduce held-out loss (e.g., reducible holdout loss) as training progresses (Mindermann et al., 2022). Difficulty- and competence-aware schedulers update sampling scores based on the model's current proficiency, steering training toward appropriately challenging examples through easy-to-hard curricula or adaptive RL fine-tuning that maintains a learnable difficulty band (Huang & Xiong, 2024; Shi et al., 2025a; Koh et al., 2025).

For *hard control*, gradient- and alignment-based criteria derive utility from update-direction information, enabling greedy top-k selection of gradient-representative batches or filtering of reasoning data whose gradients are most aligned with desired updates (Wang et al., 2024d; Li et al., 2025d). Difficulty-based hard filtering applies threshold pruning that admits only examples within the model's current learning range—either progressively increasing the difficulty band across epochs via self-paced curricula combined with diversity-promoting subset selection (Yang et al., 2024b), or dynamically maintaining a balanced pass-rate window centered around intermediate difficulty at each training step (Bae et al., 2026). Uncertainty-based strategies similarly refresh scores from the current model state, prioritizing samples or tasks with high predictive instability (e.g., recency-based uncertainty histories or prompt-sensitivity under perturbations) to track what the model is least settled on at that moment (Song et al., 2020; Kung et al., 2023). To avoid redundancy and collapse during iterative selection, similarity- or diversity-aware methods impose explicit coverage constraints (e.g., self-evolving K-center sampling or balanced LSH-style bucketed selection), enforcing representative coverage of the data space (Wu et al., 2023; Phan et al., 2025). Finally, when external judges or reward mechanisms are available, iterative re-scoring can be instantiated as multi-round selection-and-tuning loops that repeatedly evaluate generated responses—via automated evaluators, surrogate judges, or reward functions—and refresh the training data or policy accordingly (Song et al., 2024; Wang et al., 2025c; Zhao et al., 2025d).

However, compared to the one-shot scoring, the trade-off is mainly computational and robustness-related. Re-scoring at high frequency can add overhead, especially when the scoring signal is expensive to compute.

Overall, compared to heuristic filters, they replace human-designed surface rules with model-derived signals, strengthening automation by deriving selection signals from the model itself. However, their core limitation is that adaptivity is "borrowed" from changing model states rather than "acquired" by learning a selector: the rule cannot optimize long-horizon objectives, calibrate itself against evaluator drift, or explicitly trade off exploration vs. exploitation in a principled way. This motivates the next regime, adaptive selection, where the selection policy itself is trained and updated. Notably, much of the recent literature that most strongly aligns with self-improvement (especially in post-training/alignment settings with dynamic feedback such as rewards, advantages, or verification outcomes) increasingly emphasizes learnable selection policies, making them a natural main focus for the following section.

3.3 Adaptive Selection

Adaptive selection represents the data selection regime most aligned with our vision of self-improvement. The core objective of data selection for self-improving systems is to autonomously determine what data is most beneficial given the current state of learning. A learnable selector operationalizes this objective by introducing a selection policy that decides which examples should be prioritized next.

This distinguishes it fundamentally from static filters and metric-guided scoring. In those earlier approaches, selection criteria are externally specified and remain fixed during training. Even if automated, they apply a predetermined rule to curate data. In contrast, a learnable selector makes selection itself a learnable component of the system. The decision rule is no longer handcrafted or frozen; it is optimized alongside the model. As a result, this regime achieves higher levels of both automation and continuity. Automation is strengthened because, once the pipeline is defined, selection decisions are driven by feedback signals without requiring ongoing human intervention. Continuity is strengthened because the selection policy evolves over time: as the model's capabilities, weaknesses, or objectives shift, the data acquisition strategy adapts accordingly.

We organize this section around three components: the **selection unit**, a design-level choice that fixes the granularity of selection before training begins; the **selection loop**, an iterative cycle of signal generation and policy update that drives the selector's evolution; and the **coupling** between the selector and the main model, which characterizes how tightly the two co-adapt during training. We conclude with a complementary perspective that distinguishes curriculum-driven from objective-driven selection finally.

3.3.1 Selection Unit as a Design Choice

Before the selection loop begins, the system must fix the *selection unit*: the atomic granularity at which data can be sampled, weighted, or filtered. Unlike the iterative stages of the loop described below, the unit is a design-level decision made once and held constant throughout the entire selection process. This choice directly affects how precisely the selector can shape the training distribution: a finer unit enables more targeted selection, while a coarser unit limits the selector to broader adjustments.

Existing learnable selectors operate at two levels of granularity. At the *instance* level, the smallest selectable element is an individual training sample, such as a pretraining document, an instruction–response pair, or a prompt–generation context for preference annotation. This is the most common choice and provides the finest-grained control over the training distribution (Yu et al., 2024d; Fan et al., 2026; Bai et al., 2025; Lv et al., 2025; Chen et al., 2025i; Shen et al., 2025; Das et al., 2025). At the *group* level, the selector allocates training budget across higher-level collections—such as data sources, difficulty-stratified distributions, or semantically clustered categories—rather than individual examples (Wang et al., 2025l; Chen et al., 2025g; Do et al., 2025; Yu et al., 2025f; Jha et al., 2025; Pan et al., 2025).

We note that in metric-guided scoring (§3.2), selection units are predominantly individual instances or predefined domains. While the unit choice remains relevant, the scoring signal more directly characterizes how each method estimates data utility. Moreover, most metric-guided scoring methods operate at the instance level, with a smaller subset applying scores at the domain or group level. We therefore organize metric-guided scoring approaches primarily by their scoring signal, noting the applicable granularity where appropriate. In contrast, the unit choice in adaptive selection plays a more prominent architectural role—directly shaping what signals the selector computes and how it updates its policy—making it a meaningful standalone axis of the taxonomy.

3.3.2 The Selection Loop

Given a fixed unit definition, the learnable selector operates through an iterative loop comprising two stages. In the first stage, *signal generation*, the system computes utility signals under the current model state to estimate the learning value of each unit. These signals are state-dependent: rather than being fixed properties of the data, they are computed relative to the current state of the model involved in selection, and thus shift as that model evolves during training. In the second stage, *selection update*, the selector transforms these signals into adjustments of the sampling policy—by reweighting, filtering, or reallocating emphasis across units—thereby reshaping the effective training distribution. This reshaped distribution influences subsequent model updates and, in turn, alters the signals observed in the next iteration.

To introduce the concrete instantiations of this loop, we categorize learnable selectors by how they realize each stage.

Signal Generation. Given a unit choice, selection signals are categorized by the quantity used to score each unit. *Influence-based* signals estimate the marginal training contribution of individual samples or groups. MATES (Yu et al., 2024d) trains a small data influence model that is continuously fine-tuned to approximate oracle influence scores obtained by locally probing the pretraining model's performance on a reference task, thereby selecting instances most beneficial for the current pretraining progress. Group-MATES (Yu et al., 2025f) extends this principle to the group level, learning influence models that predict domain-level utility to guide pretraining data mixture decisions.

Uncertainty-based signals prioritize data that is expected to provide high informational gain. APO (Das et al., 2025) formulates RLHF alignment as a contextual preference bandit problem and iteratively selects the most uncertain prompt contexts for preference labeling, provably achieving near-optimal sample efficiency under the Bradley-Terry-Luce model (BTL).

Advantage- or outcome-based signals rely on performance-derived quantities observed during training. DUMP (Wang et al., 2025l) uses the magnitude of policy advantages as a proxy for distribution-level learnability, dynamically adjusting sampling probabilities across data distributions via a UCB-based bandit during RL post-training. SEC (Chen et al., 2025g) similarly formulates curriculum selection as a non-stationary

multi-armed bandit, using the absolute advantage from policy gradient methods as a reward signal and updating sampling probabilities with TD(0). SPaRFT (Do et al., 2025) first applies cluster-based data reduction to partition training data by semantics and difficulty, then treats each cluster as an arm in a multi-armed bandit whose reward reflects the model's current solve-rate performance on that cluster.

Loss-based signals directly use held-out validation loss or validation performance improvement as the scoring criterion. RAISE (Lv et al., 2025) models dynamic instruction selection as a sequential decision-making process, training a sample-wise scorer (acquisition function) via reinforcement learning to maximize downstream validation performance improvement across instruction fine-tuning steps. SEAL (Shen et al., 2025) learns a data ranker through bilevel optimization, where the upper-level objective evaluates safety-alignment quality on a held-out validation set while the lower level performs standard fine-tuning, thereby up-ranking safe and high-quality data. ScaleBiO (Pan et al., 2025) introduces the first scalable first-order bilevel optimization framework for LLM data reweighting, learning per-source sampling weights that minimize validation loss across multiple data sources. RL-Guided Selection (Jha et al., 2025) trains an RL-based selector that learns to allocate training budget across data groups by optimizing for downstream task performance.

Finally, *composite* signals combine multiple measurements into a unified scalar utility. ScalingRL (Chen et al., 2025i) integrates difficulty, reasoning complexity, and reward adaptability into a dynamic effectiveness score that is recomputed at each training epoch to guide within-difficulty-level sampling during RL fine-tuning. DATAMASK (Fan et al., 2026) jointly optimizes quality and diversity via a policy-gradient-based mask learning objective over the pretraining corpus. Multi-Actor (Bai et al., 2025) aggregates heterogeneous scoring signals from multiple specialized actor models, each capturing a different aspect of data utility, to produce a unified selection decision for pretraining.

Selection Update. Selectors convert these signals into changes in the effective training distribution through several recurring update mechanisms. *Supervised* updates fit auxiliary models that directly predict selection priorities from collected oracle labels (Yu et al., 2024d; 2025f). *RL-based* updates optimize a selection policy against reward or utility feedback using policy-gradient methods (Lv et al., 2025; Jha et al., 2025; Fan et al., 2026). *Bandit-style* updates treat selection as a multi-armed bandit problem, adaptively allocating sampling probability via exploration–exploitation strategies such as UCB, Thompson Sampling, or Boltzmann exploration (Das et al., 2025; Wang et al., 2025l; Chen et al., 2025g; Do et al., 2025). *Learned reweighting* mechanisms optimize continuous per-unit weights or mixture coefficients to reshape the training distribution (Chen et al., 2025i; Bai et al., 2025). Finally, *Bilevel* approaches update selection parameters by differentiating through an inner training process with respect to an outer validation or alignment objective (Shen et al., 2025; Pan et al., 2025).

Table 4 summarizes each reviewed method under the two-stage loop, categorizing it by unit granularity, signal type, and update mechanism. The table reveals that, while the space of signal–update combinations is diverse, the interaction between the selector and the main model—specifically, whether the two co-evolve within the same loop or operate independently—remains an important yet orthogonal design dimension. We examine this dimension next.

3.3.3 Coupling between Selector and Model

Beyond the signal and update axes captured in Table 4, an important additional dimension characterizes how the selector and the main model interact during training: the degree of **coupling** between them.

In a *coupled* setting, the main model is updated on the selected data, and its changed state feeds back into the next round of signal generation, creating an interleaved loop in which selection and optimization co-adapt. The majority of surveyed methods adopt this design (Yu et al., 2024d; 2025f; Lv et al., 2025; Chen et al., 2025i; Das et al., 2025; Wang et al., 2025l; Chen et al., 2025g; Do et al., 2025).

Coupling arises naturally when the selector's signal is derived from the main model being optimized—for instance, policy advantages in RL-based curricula (Wang et al., 2025l; Chen et al., 2025g; Do et al., 2025), data influence probed from evolving pretraining checkpoints (Yu et al., 2024d; 2025f), or validation performance tracked across fine-tuning steps (Lv et al., 2025; Chen et al., 2025i).

Table 4: **Adaptive selection methods and their design dimensions.** This table summarizes representative methods by their selection unit, the signals used, the update mechanism and whether the selection process is coupled or decoupled with model training.

Method	Unit	Signal	Update	Coupling
MATES (Yu et al., 2024d)	Instance	Influence	Supervised	Coupled
RAISE (Lv et al., 2025)	Instance	Loss	RL	Coupled
ScalingRL (Chen et al., 2025i)	Instance	Composite	Learned reweighting	Coupled
APO (Das et al., 2025)	Instance	Uncertainty	Bandit	Coupled
SEAL (Shen et al., 2025)	Instance	Loss	Bilevel	Decoupled
Multi-Actor (Bai et al., 2025)	Instance	Composite	Learned reweighting	Decoupled
DATAMASK (Fan et al., 2026)	Instance	Composite	RL	Decoupled
DUMP (Wang et al., 2025l)	Group	Advantage/Outcome	Bandit	Coupled
SEC (Chen et al., 2025g)	Group	Advantage/Outcome	Bandit	Coupled
SPaRFT (Do et al., 2025)	Group	Advantage/Outcome	Bandit	Coupled
Group-MATES (Yu et al., 2025f)	Group	Influence	Supervised	Coupled
RL-Guided Selection (Jha et al., 2025)	Group	Loss	RL	Decoupled
ScaleBiO (Pan et al., 2025)	Group	Loss	Bilevel	Decoupled

In a *decoupled* setting, the selector is trained independently—often via bilevel optimization on a held-out validation objective (Shen et al., 2025; Pan et al., 2025), through a separate policy-gradient or scoring phase (Fan et al., 2026; Bai et al., 2025), or using a proxy model to provide reward signals (Jha et al., 2025)—and the resulting selection is applied before the main model begins training. Decoupled designs reduce systems complexity, but forfeit the ability to adapt selection as the model's needs shift during training.

This distinction parallels the contrast between one-shot and iterative re-scoring in the metric-guided scoring regime: just as iterative re-scoring refreshes utility estimates during training while one-shot scoring fixes them beforehand, coupled selectors continuously re-adapt their policies while decoupled selectors commit to a fixed selection before training begins.

Notably, all surveyed methods in this regime rely on a *separate* selector module—whether a small influence model (Yu et al., 2024d), a trainable MLP scorer (Lv et al., 2025), a bandit policy (Wang et al., 2025l; Chen et al., 2025g), or a bilevel-optimized ranker (Shen et al., 2025)—while a distinct main model is optimized on the selected data. No existing method in this regime has the main model curate its own training data without an external selector. This gap points to an underexplored direction for future work: *self-directed selection*, in which a single model simultaneously acts as both selector and learner, representing the strongest form of autonomy in self-improving systems.

3.3.4 Curriculum-Driven vs. Objective-Driven Selection

From another perspective, adaptive selection mechanisms can also be categorized based on what fundamentally drives the selection criterion. Although these methods generally rely on signals to update the selector over time, the underlying objectives that those signals serve are not the same. Under this view, existing methods largely fall into two paradigms: **Curriculum-Driven** and **Objective-Driven** selection.

Curriculum-driven selection aims to select data that is most appropriate for the model's current learning stage. The core idea is to adapt the training distribution to the model's evolving competence. Here, "appropriate" does not simply mean high-quality or high-reward; rather, it refers to examples whose difficulty or informational value matches the model's current capacity—neither too trivial to provide new learning signal, nor too difficult to be learnable (Chen et al., 2025i; Wang et al., 2025l; Chen et al., 2025g; Do et al., 2025).

Table 5: **Representative data selection methods across different training stages.** This table summarizes representative methods applied during pre-training and post-training, highlighting how data selection strategies are used at different stages of model development.

Training Phase	Methods
Pre-training	C4 (Raffel et al., 2020), CCNET (Wenzek et al., 2020), The Pile (Gao et al., 2020), GPT3 (Brown et al., 2020), GLaM (Du et al., 2022), RHO-LOSS (Mindermann et al., 2022), PaLM (Chowdhery et al., 2023), DSIR (Xie et al., 2023b), DoReMi (Xie et al., 2023a), GREATS (Wang et al., 2024d), DATAMASK (Fan et al., 2026)
Post-training	Recency Bias (Song et al., 2020), NUGGETS (Li et al., 2024j), S2L (Yang et al., 2024c), LESS (Xia et al., 2024b), Li et al. (2024e), SelectIT (Liu et al., 2024d) DEFT (Ivison et al., 2023), AlpaGasus (Chen et al., 2024c), DEITA (Liu et al., 2024g) SST (hattami et al., 2025), PREPO (Sun et al., 2025b), GREATS (Wang et al., 2024d), IT2ACL (Huang & Xiong, 2024), P3 (Yang et al., 2024b), AdaRFT (Shi et al., 2025a), AdaSTaR (Koh et al., 2025), LearnAlign (Li et al., 2025d), DUMP (Wang et al., 2025l), SEC (Chen et al., 2025g), RAISE (Lv et al., 2025), SPaRFT (Do et al., 2025), ScalingRL (Chen et al., 2025i), SEAL (Shen et al., 2025)

On the other hand, objective-driven selection directly prioritizes data based on its estimated contribution to a predefined optimization objective. The target objective may vary depending on the setting—for example, improving downstream task performance, enhancing reasoning accuracy, aligning the model with human preferences, or enforcing safety constraints (Lv et al., 2025; Jha et al., 2025; Shen et al., 2025; Pan et al., 2025; Fan et al., 2026).

In summary, adaptive selection advances data selection from a fixed preprocessing step to a learnable, evolving component of the training pipeline. The methods surveyed in this section differ along several design axes—unit granularity, signal type, update mechanism, and the degree of coupling with the main model—yet they share a common goal: the selection policy should be optimized to respond to the model's evolving competence, rather than fixed by hand. This responsiveness may be driven by curriculum considerations that match data difficulty to the model's current capacity, or by objective-driven criteria that maximize a downstream target. In either case, the aim is a closed feedback loop between what the model learns and what it is trained on next—one that coupled methods realize directly and decoupled methods approximate through offline optimization. Realizing this loop in practice, however, involves trade-offs in computational cost, training stability, and systems complexity that must be weighed against the gains in adaptivity—a theme we return to in the broader discussion of §3.4.

3.4 Discussion

Trade-off of three regimes. Data selection is a control layer in self-improvement pipelines: by deciding which samples enter optimization (and with what weight) at each step, it effectively decides what the model trains next. Across three regimes, increased automation and continuity comes with clear trade-offs. Static filters offer simplicity, stability, and corpus-scale throughput, but their rules remain fixed and cannot track a model's changing competence. Metric-guided scoring replace handcrafted heuristics with model- or evaluator-derived signals: one-shot scoring is cheap and stable yet may become stale as objectives shift, whereas iterative re-scoring updates the selection signal during training to better track non-stationary utility at additional computational and robustness cost. Adaptive selection goes further by optimizing the selection policy itself, enabling budget-aware curricula and longer-horizon control, but they introduce sensitivity to noisy/drifting feedback, delayed credit assignment, and extra systems overhead.

Pre-training vs. Post-training. Viewing data selection through the training pipeline (Table 5) reveals that these trade-offs manifest differently across training phases, forming a clear phase-dependent pattern. In **pre-training**, the dominant constraints are scale, throughput, and stability: datasets are massive, objectives are primarily next-token prediction, and selection must be cheap and robust. Consequently, static filtering and one-shot scoring-based mixture shaping (e.g., domain reweighting or pruning) are especially attractive, while online selection is used more selectively due to overhead. In **post-training**, which encompasses fine-tuning and alignment, the setting changes: training budgets are smaller, objectives are more targeted, and automated evaluators such as reward models, verifiers, and critics, alongside preference signals, provide richer feedback. This makes iterative re-scoring and adaptive selection substantially more practical and better aligned with self-improvement continuity, since the data stream can be adapted as the model's behavior changes under closed-loop feedback.

This phase-dependent contrast suggests that no single selection mechanism is sufficient across all stages of training. Instead, different mechanisms should play different roles depending on available feedback and computational constraints. We therefore propose self-improving data selection as a coordinated combination of strategies applied across successive training phases: stable, inexpensive mechanisms establish a broad foundation at scale, while adaptive methods operate closer to the optimization loop as richer feedback becomes available.

Concretely, a self-improvement pipeline may first enforce hard constraints (e.g., safety, licensing or provenance, toxicity filtering, deduplication), then adjust coarse mixture proportions across domains or skills, and finally apply policy-learned selection to allocate limited post-training budget toward samples that are simultaneously learnable, high-impact, and aligned with target behaviors. By separating stable, large-scale screening from fine-grained adaptive refinement, this design makes adaptivity scalable: expensive feedback and policy learning are applied only where they add the most value, while earlier stages remain robust and computationally efficient. Under this view, continuity does not come from switching to entirely new selection strategies at each training stage, but from gradually adapting the data distribution as the model itself improves.

4 Model Optimization for Self-Improvement

4.1 Overview

Model optimization is the engine that converts experiences into actual capability gains. While the acquisition and selection stages focus on preparing the curriculum, optimization is where the policy model undergoes parameter updates to internalize these signals, evolving into a more advanced version without external human intervention.

To unify the diverse approaches in self-improvement, we categorize them under the Generation–Reward–Optimization (GRO) framework as demonstrated in Figure 6. This framework treats the model optimization cycle as a continuous feedback loop consisting of three tightly coupled stages:

- **Generation:** Starting from the current policy, the model acts as its own experience generator, producing candidate outputs, reasoning chains, or action trajectories for a given set of tasks.
- **Reward:** These outputs are then evaluated automatically to produce a feedback signal. This feedback is distilled into specific formats based on the intended learning objective: binary filters to identify high-quality samples, scalar scores to quantify performance, or comparative rankings to establish preferences. Whether derived from internal model logic or external environment feedback, these signals provide the "gradient of quality" necessary for improvement.
- **Optimization:** In the final stage, the policy parameters are updated to internalize these signals, evolving into the refined model. The choice of algorithm is directly coupled with the reward format: Supervised Fine-Tuning (SFT) is applied to filtered data, Reinforcement Learning (e.g., PPO) utilizes scalar rewards to maximize expected returns, and Preference Learning (e.g., DPO) leverages

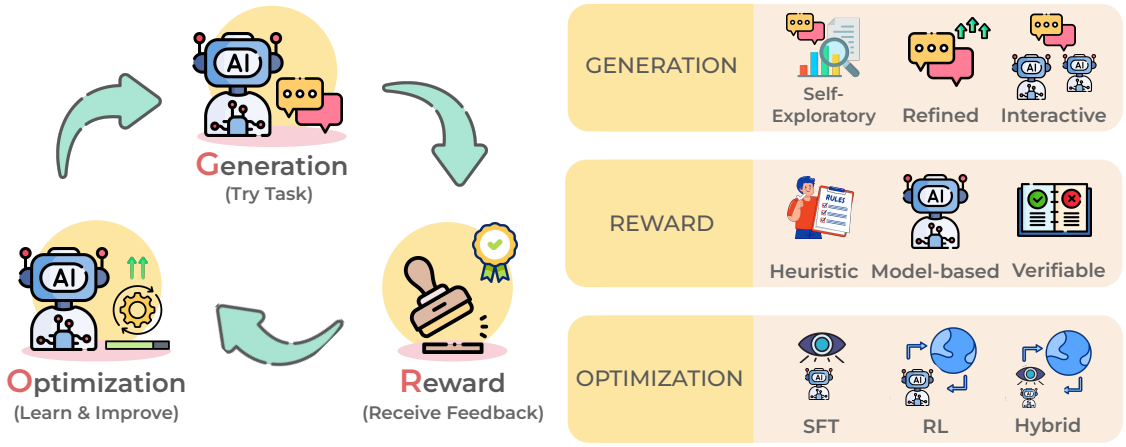

Figure 6: **Overview of model optimization in the self-improvement system.** This stage focuses on how the model improves its capabilities by iteratively generating outputs, receiving feedback, and updating its parameters. These processes are summarized as a unified *Generation–Reward–Optimization (GRO)* framework. *(i) Generation:* The model produces candidate outputs using strategies such as self-exploratory generation, refinement-based generation, or interactive generation. *(ii) Reward:* The generated outputs are evaluated using different feedback signals, including heuristic, model-based, and verifiable signals. *(iii) Optimization:* The model parameters are updated based on the evaluated feedback through methods such as supervised fine-tuning, reinforcement learning, or hybrid optimization approaches.

preference pairs to shift the model's distribution. This step executes the actual transformation of the model's capabilities, completing one cycle of the improvement.

Through repeated iterations, the model follows an evolutionary trajectory where each version serves as the foundation for its own next stage of growth. This recursive cycle enables the policy to "climb" its own performance gradient, transforming standard optimization into a continuous process of self-improvement that accumulates reasoning capabilities and insights far exceeding its initial state. In the following section, §4.2 presents the GRO framework, including generation, reward, and optimization as its core components. Table 6 summarizes the taxonomy of methods for each stage. §4.3 discusses representative paradigms built upon this framework, illustrating different optimization strategies. §4.4 provides theoretical perspectives on the GRO framework. §4.5 extends optimization approaches beyond GRO, followed by a discussion of all model optimization methods in §4.6.

4.2 GRO Framework

4.2.1 Generation

Generation refers to the process by which a model produces candidate outputs for a given task. We categorize existing generation strategies into three classes: **Self-Exploratory Generation**, **Refined Generation**, and **Interactive Generation**. Self-exploratory generation operates directly on the model's current policy, either by producing a single response that represents the policy's behavior or by sampling a batch of responses to explore the output space via search. Refined generation starts from an initial response and iteratively improves it, aiming to obtain high-quality data for training, construct preference pairs, or explicitly learn the refinement process itself. Interactive generation extends the process beyond the isolated model, relying on collaborative consensus, adversarial dynamics with an opponent, or interaction with external tools to drive generation.

Self-Exploratory Generation. The fundamental strategy in self-exploratory generation is producing outputs directly from the model's current policy. In its simplest form, the model generates a single response or reasoning path to be assessed; methods like STaR (Zelikman et al., 2022), SIRLC (Pang et al., 2024),

and RLSR (Simonds et al., 2025) utilize this approach by generating a solitary rationale or answer per input. SimRAG (Xu et al., 2025a) extends this by generating an answer followed by a retrieved question for consistency. This policy-driven generation also underpins self-play approaches, where the generated output represents a move by the current policy or an opponent to construct training signals. SPIN (Chen et al., 2024h) generates responses to contrast against ground truth in a zero-sum game, while SPPO (Wu et al., 2025g), SeRL (Fang et al., 2025c), RSPO (Tang et al., 2025b), and SOL-VER (Lin et al., 2025c) generate outputs from the current policy or a reference model to serve as players in a preference optimization or verification game.

To explore the output space more robustly, many methods sample multiple diverse candidates to identify consistent answers or construct contrastive pairs. LMSI (Huang et al., 2023a), CRESCENT (Sun et al., 2025c), and ReGenesis (PENG et al., 2025) sample diverse paths and apply majority voting to select high-quality outputs. Jiang et al. (2025a) similarly employ importance weighting on multiple samples to filter training data. Jiang et al. (2025b) and RESTRAIN (YU et al., 2026) further analyze these batches for semantic clusters or voting signals to reduce noise. Additionally, ScPO (Prasad et al., 2025), DNPO (Yang et al., 2026), Goedel-Prover (Lin et al., 2025b), and V-STaR (Hosseini et al., 2024) generate multiple candidates to be filtered by verifiers or ranked by the model itself to form valid training pairs. This strategy is also central to Yuan et al. (2024b), Wu et al. (2025c), and Liu et al. (2025d), which rely on sampling candidates for self-evaluation or verification.

In addition, to structure the exploration more effectively, some methods utilize tree search algorithms. ReST-MCTS* (Zhang et al., 2024a) and SPHERE (Singh et al., 2025) employ Monte Carlo Tree Search to actively navigate the generation space, pruning low-quality paths and selecting the most promising trajectories for learning.

Refined Generation. Refined generation employs iterative improvements on initial responses to obtain higher-quality trajectories that serve as positive training data. In this setting, the initial generation is merely a stepping stone; the model employs iterative self-correction or feedback loops to ensure the final output is correct before using it for supervised fine-tuning. SELF (Lu et al., 2024c) and STaSC (Moskvoretskii et al., 2025) implement a "generate-critique-revise" pipeline, retaining the refined trajectory only if it passes specific quality checks. RISE (Qu et al., 2024) and TRIPOST (Yu et al., 2024c) enforce a rigorous feedback loop, where the model acts on internal signals or external teacher feedback to "try again" iteratively until a correct reasoning path is achieved. ReGenesis (PENG et al., 2025) also falls into this category by explicitly refining reasoning chains based on ground truth signals to construct a high-quality dataset for fine-tuning. Similarly, Bensal et al. (2025) treat the refinement as a reflection step triggered by verification signals to produce a successful final trajectory.

Beyond simply gathering correct answers, many approaches use refinement to construct preference pairs by contrasting the initial weak response with the refined strong response. This allows the model to learn the improvement signal via algorithms like DPO. SynPO (Dong et al., 2025) explicitly constructs optimization pairs by utilizing the initial generation as the rejected sample and the refined output as the chosen sample. Ji et al. (2025) leverage external context to create this contrast, treating "closed-book" generations as weaker samples and "open-book" generations as the refined targets. SPHERE (Singh et al., 2025) and Xiong et al. (2025) further utilize the success or failure of the self-correction process to label trajectories, creating pairs of successful refinements versus failed attempts to guide preference optimization.

Finally, some methods treat the refinement step as a direct supervision signal to explicitly learn the refinement process itself. In this paradigm, the initial incorrect response serves as the input context, and the refined response serves as the target output, effectively training the model to map errors to corrections. STaPLe (Ramji et al., 2025) operationalizes this by generating a latent "principle" derived from the ground truth to bridge the gap between the initial error and the correct answer. SCoRe (Kumar et al., 2024) trains the model to maximize the quality of the second-turn response given the first-turn attempt, reinforcing the capability to self-correct without external supervision. Similarly, PAG (Jiang et al., 2025e) treats the policy as a generative verifier, learning to improve reasoning chains over multiple turns of self-correction.

Interactive Generation. Interactive generation often relies on collaborative consensus, where multiple agents cooperate to synthesize a high-quality output. In this setting, agents effectively act as a team, engaging in debate or sequential improvement. DTE (Srivastava et al., 2025) and Subramaniam et al. (2025) utilize a debate framework where distinct agents critique each other's reasoning chains, aggregating their outputs to filter for consistency and reduce hallucinations. SiriuS (Zhao et al., 2025c) adopts a sequential collaborative model, where agents hand off partial solutions or cooperate in a multi-turn dialogue to solve complex tasks that a single agent could not manage alone.

Other strategies employ adversarial generation, where the generation process is structurally dependent on an opponent. Here, the primary model's generation is often a response to a dynamic challenge created by an adversarial peer. R-Zero (Huang et al., 2026a), Dr. Zero (Yue et al., 2026), and AbsoluteZero (Zhao et al., 2025a) rely on this "Proposer-Solver" dynamic, where one agent must generate a novel problem or theorem before the solver agent can generate a solution trajectory. Similarly, Zhou et al. (2025a) involve a Challenger agent that actively probes the Executor to induce failure, forcing the Executor to generate more robust responses. SSR (Wei et al., 2025b) and SPICE (Liu et al., 2025a) also follow this paradigm, where the generation of code fixes or reasoning chains is directly conditioned on the specific bugs or difficult contexts synthesized by the adversarial role.

Lastly, generation can be tool-augmented, relying on external tools, execution environments, or retrieval systems to proceed. Unlike post-hoc verification, these methods require the tool's feedback during the generation loop to construct the final trajectory. REVEAL (Jin et al., 2026) integrates a code execution environment directly into the generation process, where agents generate code, execute it to observe the state, and use that observation to guide subsequent generation steps. CURE (Wang et al., 2025i) co-evolves a coder and a unit tester, where the generation of the solution is strictly coupled with the generation and execution of test cases, ensuring the output is grounded in verifiable execution feedback.

4.2.2 Reward

Given the generated outputs, the reward stage provides signals that assess their quality and guide subsequent training updates. We categorize reward into three major types: **Heuristic Reward**, **Model-Based Reward**, and **Verification Reward**. Heuristic reward evaluates quality without a trained reward model, relying on statistical consistency among samples, specifically designed scoring functions, or pre-defined assumptions about the relative quality of refined outputs. Model-based reward leverages the semantic capabilities of LLMs, either by having the model self-evaluate its own generations or by utilizing external peer models and judges to provide feedback. Verification reward offers the most definitive signals, derived from strict ground truth matching or execution feedback from formal verifiers and code compilers.

Heuristic Reward. Heuristic reward relies on statistical signals or rule-based metrics to approximate quality. A dominant approach is consistency-based evaluation, which operates on the premise that consensus among multiple generations indicates correctness. LMSI (Huang et al., 2023a) and IWSI (Jiang et al., 2025a) sample multiple reasoning paths and utilize majority voting to identify the most consistent answer, treating it as a positive training signal. Similarly, ReGenesis (PENG et al., 2025) and CRESCENT (Sun et al., 2025c) leverage voting consensus across diverse reasoning chains or self-generated questions to filter high-quality trajectories. In multi-agent settings, DTE (Srivastava et al., 2025) and Subramaniam et al. (2025) employ consistency scores among debating agents to reduce hallucinations. ScPO (Prasad et al., 2025) also uses consistency measures to distinguish between high- and low-quality outputs for preference optimization, while Semantic Voting (Jiang et al., 2025b) enhances this by clustering semantically similar responses to estimate correctness more robustly.

Other methods employ specifically designed heuristic functions where specific scoring rules are engineered to evaluate quality. SPIN (Chen et al., 2024h) derives an implicit reward signal by comparing the model's likelihood on generated data against reference data. RESTRAIN (YU et al., 2026) designs a penalization term to down-weight high-confidence but incorrect samples. In retrieval contexts, SimRAG (Xu et al., 2025a) uses a heuristic based on whether the generated question can successfully retrieve the document containing the answer. STaPLe (Ramji et al., 2025) similarly employs a similarity function to select the best generated principle-response pair. Furthermore, in adversarial and code generation frameworks, heuristic rewards are

often used to prioritize difficulty or promise. The Proposer agents in AbsoluteZero (Zhao et al., 2025a), SPICE (Liu et al., 2025a), R-Zero (Huang et al., 2026a), and Dr. Zero (Yue et al., 2026) receive heuristic rewards based on problem difficulty to drive exploration. Similarly, REVEAL (Jin et al., 2026), CURE (Wang et al., 2025i), and SSR (Wei et al., 2025b) utilize static analysis scores to filter code candidates before execution.

Finally, some reward is based on pre-defined assumptions, establishing simple rules to determine relative quality. SynPO (Dong et al., 2025) assumes that the output from a refinement step is inherently superior to the initial generation, automatically forming a chosen-rejected pair. Ji et al. (2025) apply a similar logic in their framework by assuming that "open-book" generations (with access to external documents) are superior to "closed-book" generations, using this assumption to label data for preference optimization without manual annotation.

Model-Based Reward. Model-based reward leverages the semantic capabilities of Large Language Models to assign scores, rank candidates, or provide feedback. In self-evaluation strategies, the generating model itself acts as the judge. A direct implementation involves prompting the model to assign scalar quality scores to its own outputs, as seen in Self-Rewarding LMs (Yuan et al., 2024b) and SIRLC (Pang et al., 2024). Beyond scalar scoring, the model can generate self-critiques or preference rankings to guide optimization. SELF (Lu et al., 2024c) utilizes the model's own capability to critique and revise responses, using these self-generated signals to filter trajectories. RISE (Qu et al., 2024) also relies on the model's internal introspection capabilities to determine when to halt the refinement loop. Similarly, Meta-rewarding (Wu et al., 2025c) and SPPO (Wu et al., 2025g) prompt the model to rank pairs of responses, deriving a probability-based reward signal directly from the current policy's preference ordering. IWSI (Jiang et al., 2025a) and Semantic Voting (Jiang et al., 2025b) further leverage the model's likelihood estimates or embedding similarities to weight or cluster samples effectively.

Alternatively, reward can be derived from external or specialized models, such as a stronger teacher, a peer agent, or a specifically trained reward model. ReST-MCTS* (Zhang et al., 2024a) exemplifies this by training a dedicated Process Reward Model (PRM) to evaluate intermediate reasoning steps, providing a dense signal distinct from the generator. DNPO (Yang et al., 2026) also falls into this category by utilizing a reference model or dynamic noise distribution to construct the preference signal. TRIPOST (Yu et al., 2024c) employs a larger teacher model to critique and score the outputs of a smaller student model. In multi-agent and adversarial settings, the reward is strictly determined by the judgment of a third-party agent or the policy of an opponent. MAE (Chen et al., 2025h) utilizes a distinct "Judge" agent to assess evolved responses, while RSPO (Tang et al., 2025b) derives rewards from the competitive outcome against a separate opponent policy. RLSR (Simonds et al., 2025), SPHERE (Singh et al., 2025), and SOL-VER (Lin et al., 2025c) further adopt this approach by leveraging external verifiers or reward models to score generated trajectories.

Verification Reward. Verification reward provides definitive quality signals by checking outputs against established truths or executable environments. The most direct form utilizes ground truth matching, where the final answer is compared against a known correct solution. STaR (Zelikman et al., 2022) and SeRL (Fang et al., 2025c) assign binary rewards based on whether the generated rationale leads to the correct final answer. RISE (Qu et al., 2024) similarly assigns a reward of 1 for correct answers and 0 for incorrect ones. This approach is also used in iterative refinement settings: STaSC (Moskvoretskii et al., 2025), Bensal et al. (2025), ReGenesis (PENG et al., 2025), and Xiong et al. (2025) validate the final corrected response against the ground truth to reward the entire correction trajectory. SELF (Lu et al., 2024c) also incorporates a verification step to accept or reject the model's self-revisions based on task success.

For domains requiring rigorous correctness, such as mathematics and code generation, reward is derived from formal verifiers or compilers. Goedel-Prover (Lin et al., 2025b) and V-STaR (Hosseini et al., 2024) employ formal theorem provers or dedicated verifier models to strictly check the validity of a generated proof. In software engineering and agent tasks, REVEAL (Jin et al., 2026), CURE (Wang et al., 2025i), SOL-VER (Lin et al., 2025c), and SSR (Wei et al., 2025b) execute generated code against unit tests; the reward is strictly determined by the compiler's success or failure signals. Verification is also central to adversarial and self-play methods, particularly for the solver role: SiriuS (Zhao et al., 2025c), PAG (Jiang et al., 2025e),

AbsoluteZero (Zhao et al., 2025a), SPICE (Liu et al., 2025a), R-Zero (Huang et al., 2026a), and Dr. Zero (Yue et al., 2026) all rely on this deterministic verification to judge the final success of the interaction. SPHERE (Singh et al., 2025) likewise relies on verification of the final answer to label its chosen and rejected pairs for optimization. Zhou et al. (2025a) also employ this verification strategy in their Self-Challenging framework to filter valid challenger-executor pairs.

4.2.3 Optimization

Optimization strategies define how the model parameters are updated based on the generated data and the corresponding reward signals. We categorize these strategies into three classes: **Supervised Fine-Tuning (SFT)**, **Reinforcement Learning (RL)**, and **Hybrid Optimization**. SFT treats high-quality self-generated data as pseudo-labels to minimize the cross-entropy loss, effectively cloning the model's best behavior. RL optimizes the policy to maximize expected rewards, utilizing algorithms like PPO, DPO, or GRPO to reinforce desirable reasoning paths or preferences. Hybrid optimization combines both approaches, typically using SFT for initial stabilization followed by RL for further alignment, or alternating between them in iterative cycles.

Supervised Fine-Tuning (SFT). Supervised Fine-Tuning operates on the principle of self-training, where the model filters its own generations to create a high-quality dataset for next-step training. The most common approach involves rejecting incorrect samples and fine-tuning only on correct reasoning paths. STaR (Zelikman et al., 2022) and Self-Challenging (Zhou et al., 2025a) exemplify this by retaining only the rationales that lead to the correct answer. LMSI (Huang et al., 2023a), ReGenesis (PENG et al., 2025), CRESCENT (Sun et al., 2025c), and IWSI (Jiang et al., 2025a) extend this by using consistency checks like majority voting to select the most reliable pseudo-labels from multiple samples. SEMANTIC VOTING (Jiang et al., 2025b) further refines this by clustering semantically similar responses to identify the target for SFT. V-STaR (Hosseini et al., 2024) also utilizes SFT in its first stage to train a generator and a verifier on correct solutions before further alignment.

In refinement and multi-agent frameworks, SFT is often used to distill successful interactions or corrections back into the policy. SELF (Lu et al., 2024c), TRIPOST (Yu et al., 2024c), and STaSC (Moskvoretskii et al., 2025) fine-tune the model on the refined, higher-quality trajectories produced after self-correction or teacher feedback. STaPLe (Ramji et al., 2025) uses SFT to teach the model to generate latent principles that bridge errors and correct answers. Similarly, SiriuS (Zhao et al., 2025c) and Multiagent Finetuning (Subramaniam et al., 2025) utilize SFT to learn from the consensus or successful trajectories of multiple agents. ReST-MCTS* (Zhang et al., 2024a) employs SFT to train the policy on high-value paths discovered by tree search. SimRAG (Xu et al., 2025a) applies SFT on self-generated question-answer pairs validated by retrieval. Finally, RISE (Qu et al., 2024) employs a weighted SFT objective, assigning higher importance to samples that required more introspection or are empirically more correct.

Reinforcement Learning (RL). Reinforcement Learning directly optimizes the model's policy to maximize a reward signal, allowing for exploration beyond the limitations of static supervised data. A major branch of research utilizes Direct Preference Optimization (DPO) and its variants to learn from chosen-rejected pairs. Self-Rewarding LMs (Yuan et al., 2024b), SynPO (Dong et al., 2025), Meta-rewarding (Wu et al., 2025c), and DNPO (Yang et al., 2026) all employ DPO to align the model with self-generated preference rankings. Ji et al. (2025) also utilize DPO to encourage the model to internalize knowledge from open-book settings. SPHERE (Singh et al., 2025) uses DPO to reinforce successful self-corrections over failed ones. V-STaR (Hosseini et al., 2024) leverages DPO in its second stage to align the generator with the verifier's preferences, ensuring the generated solutions are not just correct but also verifiable.

Other approaches leverage online RL algorithms like PPO or group-based variants to optimize scalar rewards. SIRLC (Pang et al., 2024) and Liu et al. (2025d) utilize PPO to maximize rewards derived from self-evaluation or verifiers. SCoRe (Kumar et al., 2024) and PAG (Jiang et al., 2025e) use RL to train the policy to self-correct in multi-turn settings. In code and agent domains, REVEAL (Jin et al., 2026), CURE (Wang et al., 2025i), and SSR (Wei et al., 2025b) employ RL to maximize pass rates on unit tests. SeRL (Fang et al.,

2025c) and MAE (Chen et al., 2025h) also use RL to update the policy based on self-play outcomes or judge feedback.

To avoid the complexity of training a separate critic, Group Relative Policy Optimization (GRPO) has gained prominence in reasoning tasks. R-Zero (Huang et al., 2026a), Dr. Zero (Yue et al., 2026), RLSR (Simonds et al., 2025), RESTRAIN (YU et al., 2026), SPICE (Liu et al., 2025a), and Bensal et al. (2025) use GRPO to optimize the policy based on the relative performance of a group of generated outputs. Additionally, specialized self-play losses are used in adversarial settings: SPIN (Chen et al., 2024h) employs a zero-sum game objective to distinguish model generations from ground truth, while RSPO (Tang et al., 2025b) and AbsoluteZero (Zhao et al., 2025a) introduce regularized objectives to maintain stability during iterative updates. SPPO (Wu et al., 2025g) approximates the Nash equilibrium of a self-play preference game.

Hybrid Optimization. Hybrid optimization combines the stability of SFT with the exploration capabilities of RL, often in a sequential or curriculum-based manner. A standard paradigm is "SFT warm-up followed by RL," where SFT establishes a foundational reasoning capability before RL optimizes for specific rewards. Goedel-Prover (Lin et al., 2025b) follows this trajectory, starting with SFT on discovered proofs and optionally transitioning to DPO or RL for further refinement. SOL-VER (Lin et al., 2025c) similarly uses SFT to initialize the policy on correct solutions before applying DPO to enforce verification constraints. DTE (Srivastava et al., 2025) implements an iterative cycle where agents are first fine-tuned (SFT) on high-consistency debate outcomes and then further optimized via GRPO.

Some methods integrate both objectives to address different aspects of the learning process. ScPO (Prasad et al., 2025) combines iterative SFT with preference optimization, using consistency scores to curate data for both stages. Xiong et al. (2025) employ a hybrid pipeline where SFT is used to learn the correction format, followed by PPO or DPO to maximize the verification reward of the corrected solution.

Table 6: **Model optimization methods for self-improvement under the GRO framework.** We summarize methods by decomposing each approach into its generation, reward, and update category.

Method	Generation	Reward	Optimization
1. Supervised Fine-Tuning (SFT)			
STaR (Zelikman et al., 2022)	Self-Exploratory	Verification	SFT
LMSI (Huang et al., 2023a)	Self-Exploratory	Heuristic	SFT
TRIPOST (Yu et al., 2024c)	Refined	Model-Based	SFT
RISE (Qu et al., 2024)	Refined	M & V	SFT
SELF (Lu et al., 2024c)	Refined	M & V	SFT
ReST-MCTS* (Zhang et al., 2024a)	Self-Exploratory	Model-Based	SFT
V-STaR (Hosseini et al., 2024)	Self-Exploratory	Verification	SFT & DPO
IWSI (Jiang et al., 2025a)	Self-Exploratory	H & M	SFT
STaPLe (Ramji et al., 2025)	Refined	Heuristic	SFT
CRESCENT (Sun et al., 2025c)	Self-Exploratory	Heuristic	SFT
REGENESIS (PENG et al., 2025)	SE & R [1]	H & V	SFT
Subramaniam et al. (2025)	Interactive	Heuristic	SFT
STaSC (Moskvoretskii et al., 2025)	Refined	Verification	SFT
SimRAG (Xu et al., 2025a)	Self-Exploratory	Heuristic	SFT
SiriuS (Zhao et al., 2025c)	Interactive	Verification	SFT
Semantic Voting (Jiang et al., 2025b)	Self-Exploratory	H & M	SFT
Self-Challenging(Zhou et al., 2025a)	Interactive	Verification	SFT
2. Reinforcement Learning (RL)			
SIRLC (Pang et al., 2024)	Self-Exploratory	Model-Based	PPO

Continued on next page

Table 6 continued from previous page

Method	Generation	Reward	Update
Yuan et al. (2024b)	Self-Exploratory	Model-Based	DPO
SCoRe (Kumar et al., 2024)	Reflective	Verification	RL
SPIN (Chen et al., 2024h)	Self-Exploratory	Heuristic	SPIN
SPPO (Wu et al., 2025g)	Self-Exploratory	Model-Based	SPPO
SynPO (Dong et al., 2025)	Refined	Heuristic	DPO
Bensal et al. (2025)	Refined	Verification	GRPO
SPHERE (Singh et al., 2025)	SE & R	M & V	DPO
RLSR (Simonds et al., 2025)	Self-Exploratory	Model-Based	GRPO
Ji et al. (2025)	Refined	Heuristic	DPO
SeRL (Fang et al., 2025c)	Self-Exploratory	Verification	RL
RISE (Liu et al., 2025d)	Self-Exploratory	Verification	PPO
PAG (Jiang et al., 2025e)	Refined	Verification	RL
CURE (Wang et al., 2025i)	Interactive	H & V	RL
SSR (Wei et al., 2025b)	Interactive	H & V	RL
RSPO (Tang et al., 2025b)	Self-Exploratory	Model-Based	RSPO
Absolute Zero (Zhao et al., 2025a)	Interactive	H & V	RL
SPICE (Liu et al., 2025a)	Interactive	H & V	DrGRPO
MAE (Chen et al., 2025h)	Interactive	Model	RL
Wu et al. (2025c)	Self-Exploratory	Model-Based	DPO
DNPO (Yang et al., 2026)	Self-Exploratory	Model-Based	DNPO
REVEAL (Jin et al., 2026)	Interactive	H & V	TAPO
RESTRAIN (YU et al., 2026)	Self-Exploratory	Heuristic	GRPO
R-Zero (Huang et al., 2026a)	Interactive	H & V	GRPO
Dr. Zero (Yue et al., 2026)	Interactive	Verification	GRPO & HRPO
3. Hybrid (SFT + RL)			
ScPO (Prasad et al., 2025)	Self-Exploratory	Heuristic	ScPO
Xiong et al. (2025)	Reflective	Verification	SFT + RL
DTE (Srivastava et al., 2025)	Interactive	Heuristic	SFT + GRPO
Goedel-Prover (Lin et al., 2025b)	Self-Exploratory	Verification	SFT + DPO
SOL-VER (Lin et al., 2025c)	Self-Exploratory	M & V	SFT + DPO

The symbol "&" indicates that multiple techniques are used within the same stage, with capital letters denoting technique abbreviations (e.g., SE&R for self-exploratory and refined generation).

4.3 Representative Paradigms in GRO

Despite the diversity of proposed methods, the core mechanisms for self-improvement largely converge into three representative paradigms. These paradigms define the structural relationship between generation, reward, and optimization.

Iterative Rejection Sampling. The core philosophy is the model improves by distilling its own best generations into its weights. In this paradigm, the model first generates a diverse set of candidate solutions. These candidates are then filtered through a rigorous verification process: using either ground truth (Oracle) or statistical consistency (Majority Vote) to retain only the high-quality samples. Finally, these filtered "pseudo-labels" are used to fine-tune the model via SFT.

This approach is exemplified by STaR (Zelikman et al., 2022), which iteratively generates rationales and fine-tunes on those that lead to correct answers. LMSI (Huang et al., 2023a), CRESCENT (Sun et al., 2025c), and IWSI (Jiang et al., 2025a) extend this by utilizing consistency checking to filter reasoning paths in the absence of ground truth. ReGenesis (PENG et al., 2025) and SimRAG (Xu et al., 2025a) also follow this logic, employing retrieval or ground truth verification to construct high-quality training sets for iterative SFT.

Self-Verification and Refinement. The second paradigm shifts focus from simple filtering to self-verification and refinement, where the model plays an active role in evaluating and optimizing its own outputs. Unlike rejection sampling which treats the model as a black-box generator, this paradigm leverages the model's semantic capabilities to assign rewards (scoring/ranking) or to iteratively refine its answers. These self-generated signals are then used to update the policy, often via Reinforcement Learning (RL) or Direct Preference Optimization (DPO).

Approaches like Self-Rewarding LMs (Yuan et al., 2024b), Meta-rewarding (Wu et al., 2025c), and SIRLC (Pang et al., 2024) train the model to act as its own judge, updating parameters to maximize self-assigned scores. SPPO (Wu et al., 2025g) and SynPO (Dong et al., 2025) utilize the model to construct preference pairs for DPO. On the refinement side, methods like SELF (Lu et al., 2024c), RISE (Qu et al., 2024), and TRIPOST (Yu et al., 2024c) employ a "generate-critique-revise" loop, where the model explicitly learns to correct its errors based on internal critiques or external verifiers before the final update.

Self-Play. The third paradigm employs self-play, transforming the self-improvement process into a dynamic interaction between multiple roles. Instead of optimizing a static objective, the model improves by competing against an opponent or collaborating with peers. This interaction provides a curriculum of increasingly difficult challenges (adversarial) or robust consensus (collaborative), allowing the model to break free from the limitations of its initial distribution.

In adversarial settings, methods like SPIN (Chen et al., 2024h) formulate training as a zero-sum game between a generator and a discriminator. AbsoluteZero (Zhao et al., 2025a), R-Zero (Huang et al., 2026a), SPICE (Liu et al., 2025a), and Dr. Zero (Yue et al., 2026) adopt a "Proposer-Solver" structure, where one agent generates novel problems and the other solves them, driving continuous evolution without human data. On the collaborative side, DTE (Srivastava et al., 2025), SiriuS (Zhao et al., 2025c), and Multiagent Finetuning (Subramaniam et al., 2025) utilize debate and cooperation among agents to filter hallucinations and converge on higher-quality reasoning chains.

4.4 Theoretical Analysis

Beyond the development of practical methodologies, recent research also focuses on the theoretical analysis of model optimization within the GRO framework. Huang et al. (2025a) conceptualizes the self-improvement process primarily as a "Sharpening" mechanism, arguing that these methods essentially redistribute probability mass from the distribution's tail toward high-quality outputs already present in the model's latent space. This mechanism is shown to depend on a critical "Generation-Verification Gap" (Song et al., 2025a), where the model's discriminative capability must strictly exceed its generative performance to provide valid supervision signals. Complementing this, Sun et al. (2026) provides a mathematical formalization of the Solver-Verifier dynamic, proving that the theoretical upper bound of improvement is constrained by the verifier's fidelity.

Building on these foundations, several works further deepen the theoretical understanding of the GRO loop from different angles. RL-STaR Chang et al. (2025) provides the formal convergence analysis of the iterative generation-optimization cycle in STaR, establishing conditions on pre-trained model quality for initiating effective improvement and proving that the policy converges to optimality even when occasional incorrect reasoning steps are incorporated. Shafayat et al. (2025) empirically and analytically examine whether the GRO loop can be sustained indefinitely under self-generated rewards, finding that while majority-voting-based self-reward initially improves both the model's performance and its own supervision quality, prolonged self-training inevitably leads to reward hacking and sudden performance collapse, highlighting feedback design as the central bottleneck for sustained self-improvement. Liu et al. (2026b) analyze self-play evolution from a data-pipeline perspective, arguing that genuine self-evolution requires the learnable information in self-synthesized data to increase across iterations, and proposing a Proposer-Solver-Verifier decomposition to diagnose when and why mode collapse occurs.

We provide a more extensive discussion of the challenges and practical failure modes arising from these theoretical insights in §7, particularly regarding the flawed feedback signals (§7.2) and optimization-driven failures (§7.3).

4.5 Beyond GRO

Beyond the standard GRO framework, several recent studies have explored different model optimization pathways for self-improvement. Shi et al. (2025b) propose iterative self-incentivization, empowering models as "Agentic Searchers" capable of self-defining goals to navigate solution spaces. Lu et al. (2024a) demonstrate that LLMs can go beyond executing update rules to inventing new optimization algorithms for their own improvement. Pushing the boundaries of recursive self-modification, the Gödel Agent Yin et al. (2025) and Darwin Gödel Machine Zhang et al. (2026a)

explore self-referential architectures that enable agents to recursively inspect and modify their own logic for open-ended evolution.

Notably, with the rise of agentic systems, model optimization has increasingly extended beyond improving the model in isolation to leveraging agent-level mechanisms for further enhancement. ASL (Sun et al., 2025a) exemplifies this trend by co-evolving three agentic roles: a Prompt Generator, a Policy Model, and a Generative Reward Model within a shared tool environment, enabling scalable open-domain self-learning without external supervision. SAGE Wang et al. (2026b) incorporates a skill library into RL-based agent training, where skills generated from previous tasks accumulate and become available for subsequent ones, shifting improvement from the parameter level to the level of reusable agentic capabilities. EvolveR Wu et al. (2025b) adopts a lifecycle perspective, alternating between offline distillation of interaction trajectories into abstract strategic principles and online retrieval-guided decision-making. Agent-R1 Cheng et al. (2025) further extends the MDP framework to systematically define RL-based training for LLM agents across diverse interactive environments.

We will introduce more of these agentic approaches in §5, particularly in §5.4. This shift from model-centric to agent-centric optimization reflects a broader trend: as self-improvement systems mature, the unit of improvement is no longer a single model but an entire agentic system comprising prompts, memory, tools, and workflows. Meanwhile, as discussed in §5.5, a complementary line of work explores test-time training, where models temporarily adapt their parameters during inference through self-generated feedback signals, further dissolving the boundary between training and deployment.

4.6 Discussion

4.6.1 Reward Formulation

Implicit vs Explicit Reward. Beyond categorizing methods by how rewards are obtained, we further distinguish reward signals by how they are utilized during the self-improvement process. Specifically, we differentiate between implicit rewards and explicit rewards.

Implicit rewards do not appear as explicit feedback signals during optimization. Instead, reward signals are implicitly encoded through data selection or filtering mechanisms, where the reward function serves as an evaluation criterion rather than a direct optimization objective. Representative examples include majority voting, which treats consensus among multiple generations as a proxy for quality, and rejection sampling, which retains only generations satisfying heuristic criteria. Although no explicit reward value is propagated, implicit rewards shape learning by biasing the training data distribution toward higher-quality outputs.

Explicit rewards, in contrast, provide directly accessible feedback signals that can be explicitly incorporated into the learning process. Depending on their form, explicit rewards can be optimized via reinforcement learning objectives or used to construct structured supervision signals for preference-based optimization. We categorize explicit rewards into four common types:

- **Binary**: indicating success or failure (e.g., correct/incorrect).
- **Scalar**: assigning real-valued scores that reflect output quality.
- **Ordinal**: providing relative rankings or preference orders among outputs.
- **Probability-Based**: representing calibrated likelihoods or confidence scores over candidate outputs.

Based on this taxonomy, we categorize all methods into implicit-reward and explicit-reward approaches, with a consolidated overview provided in Table 7.

Process vs Output Reward. In addition to reward utilization and representation, we further characterize reward signals by their granularity. Specifically, we distinguish between **process reward** and **outcome reward**. Process reward provides feedback at intermediate steps of reasoning or along the generation trajectory, enabling fine-grained supervision of the model's decision-making process. In contrast, outcome reward evaluates only the final output. Except a few methods (Zhang et al., 2024a; Xiong et al., 2025; Jiang et al., 2025e), the majority of methods rely on outcome reward. Overall, outcome reward is appealing due to its simplicity, low annotation cost, and ease of integration across diverse tasks and optimization frameworks. It suggests that outcome reward, owing to its simplicity, low annotation cost, and ease of integration across diverse tasks and optimization frameworks, offers greater generality and practicality while still achieving good empirical performance compared to fine-grained process reward.

Table 7: **Reward types used in model optimization for self-improvement.** We categorize reward signals into implicit and explicit types. Explicit rewards are further divided into binary, scalar, ordinal, and probability-based signals, reflecting different levels of supervision granularity.

Reward Type	**Methods**
Implicit Reward	STaR (Zelikman et al., 2022), LMSI (Huang et al., 2023a) TRIPOST (Yu et al., 2024c), RISE (Qu et al., 2024), SELF (Lu et al., 2024c), SPIN (Chen et al., 2024h), V-STaR (Hosseini et al., 2024), Goedel-Prover (Lin et al., 2025b), IWSI (Jiang et al., 2025a), STaPLe (Ramji et al., 2025), CRESCENT (Sun et al., 2025c), REGENESIS (PENG et al., 2025), Subramaniam et al. (2025), STaSC (Moskvoretskii et al., 2025), SimRAG (Xu et al., 2025a), SiriuS (Zhao et al., 2025c), SOL-VER (Lin et al., 2025c), Semantic Voting (Jiang et al., 2025b)
Explicit Reward	*Binary* Bensal et al. (2025), SeRL (Fang et al., 2025c) RISE (Liu et al., 2025d), PAG (Jiang et al., 2025e), Self-Challenging (Zhou et al., 2025a), SSR (Wei et al., 2025b)
	Scalar SIRLC (Pang et al., 2024), SCoRe (Kumar et al., 2024), ReST-MCTS* (Zhang et al., 2024a), (Xiong et al., 2025), DTE (Srivastava et al., 2025), RLSR (Simonds et al., 2025), CURE (Wang et al., 2025i), Absolute Zero (Zhao et al., 2025a), SPICE (Liu et al., 2025a), MAE (Chen et al., 2025h), REVEAL (Jin et al., 2026), RESTRAIN (YU et al., 2026), R-zero (Huang et al., 2026a), Dr. Zero (Yue et al., 2026)
	Ordinal Yuan et al. (2024b), ScPO (Prasad et al., 2025), SynPO (Dong et al., 2025), SPHERE (Singh et al., 2025), Ji et al. (2025), Wu et al. (2025c), DNPO (Yang et al., 2026)
	Probability SPPO (Wu et al., 2025g), RSPO (Tang et al., 2025b)

4.6.2 Degree of Automation

When discussing self-improvement, an important orthogonal perspective is the degree of automation involved in the improvement process. From this viewpoint, existing methods can be analyzed along two complementary dimensions.

Data Dependency. The first dimension concerns whether model optimization starts from an annotated dataset, and if so, which components of that dataset are required. A given dataset typically consists of prompts or questions paired with corresponding ground-truth answers. At the highest level of automation, model optimization does not rely on a given dataset at all, requiring neither prompts nor ground-truth answers, as all necessary inputs and supervision signals are produced by the model itself. An intermediate setting accesses to prompts or questions only, while the corresponding supervision answers are autonomously generated or inferred by the model. At the lowest level of automation, both prompts and ground-truth answers are required, relying on fully annotated datasets. We summarize representative methods along this dimension in Table 8.

Table 8: **Degree of automation in model optimization with respect to data dependency.** Methods are categorized by their reliance on externally collected prompts and ground-truth annotations: high automation requires neither, intermediate uses either one, and low automation depends on both.

High Automation	Intermediate Automation	Low Automation
STaR (Zelikman et al., 2022)	LMSI (Huang et al., 2023a)	Yuan et al. (2024b)
TRIPOST (Yu et al., 2024c)	SIRLC (Pang et al., 2024)	CRESCENT (Sun et al., 2025c)
RISE (Qu et al., 2024)	SELF (Lu et al., 2024c)	SynPO (Dong et al., 2025)
SCoRe (Kumar et al., 2024)	ScPO (Prasad et al., 2025)	RLSR (Simonds et al., 2025)
ReST-MCTS* (Zhang et al., 2024a)	DTE (Srivastava et al., 2025)	Ji et al. (2025)
Xiong et al. (2025)	Goedel-Prover (Lin et al., 2025b)	SeRL (Fang et al., 2025c)
SPIN (Chen et al., 2024h)	SPPO (Wu et al., 2025g)	SOL-VER (Lin et al., 2025c)
V-STaR (Hosseini et al., 2024)	STaPLe (Ramji et al., 2025)	Self-Challenging (Zhou et al., 2025a)
IWSI (Jiang et al., 2025a)	Subramaniam et al. (2025)	SSR (Wei et al., 2025b)
REGENESIS (PENG et al., 2025)	SimRAG (Xu et al., 2025a)	Absolute Zero (Zhao et al., 2025a)
STaSC (Moskvoretskii et al., 2025)	Semantic Voting (Jiang et al., 2025b)	SPICE (Liu et al., 2025a)
Bensal et al. (2025)	CURE (Wang et al., 2025i)	MAE (Chen et al., 2025h)
SPHERE (Singh et al., 2025)	Wu et al. (2025c)	R-Zero (Huang et al., 2026a)
RISE (Liu et al., 2025d)	REVEAL (Jin et al., 2026)	Dr. Zero (Yue et al., 2026)
PAG (Jiang et al., 2025e)	RESTRAIN (YU et al., 2026)	
RSPO (Tang et al., 2025b)		
DNPO (Yang et al., 2026)		

Model Dependency. The second dimension focuses on the models involved in the self-improvement loop. Some approaches rely solely on the target model itself, using self-generated outputs, self-heuristic, or internal scoring mechanisms. In contrast, other methods depend on external components, such as stronger teacher models, reward models, or verification systems, to provide supervision or evaluation signals. Except for a limited subset of methods (Yu et al., 2024c; Wu et al., 2025g; Singh et al., 2025; Jiang et al., 2025b; Tang et al., 2025b; Yang et al., 2026) that explicitly depend on external models, the majority of existing work achieves self-improvement without relying on additional external models.

Overall, as foundation models continue to scale in capability, self-improvement paradigms are increasingly moving toward higher degrees of automation. In the long run, truly autonomous self-improvement is expected to evolve toward settings that require neither externally annotated data nor auxiliary models, with both supervision signals and optimization guidance emerging entirely from the model itself.

5 Inference Refinement for Self-Improvement

5.1 Overview

Inference-time refinement refers to the set of mechanisms that enhance a model's performance during the inference phase, often without requiring permanent updates to its underlying parameters. While training-time optimization focuses on building a better "brain", inference refinement focuses on making that brain "think" more effectively. This stage is necessary because even the most advanced models frequently fail to produce the most accurate result on their first attempt, especially in complex, multi-step tasks.

These approaches treat the model as a self-contained reasoning entity, improving output quality through decoding or reasoning-based modifications, and can be applied whether the underlying system uses a single LLM or an ensemble of models. Agentic systems extend beyond the core model concept to harness a richer set of capabilities and external interfaces that support long-horizon interaction with an environment. Agentic improvement encompasses not only output refinement but also enhancements to memory architectures, tool usage, prompt and workflow orchestration, and interaction dynamics, which together allow the system to accumulate experience, leverage context, and plan

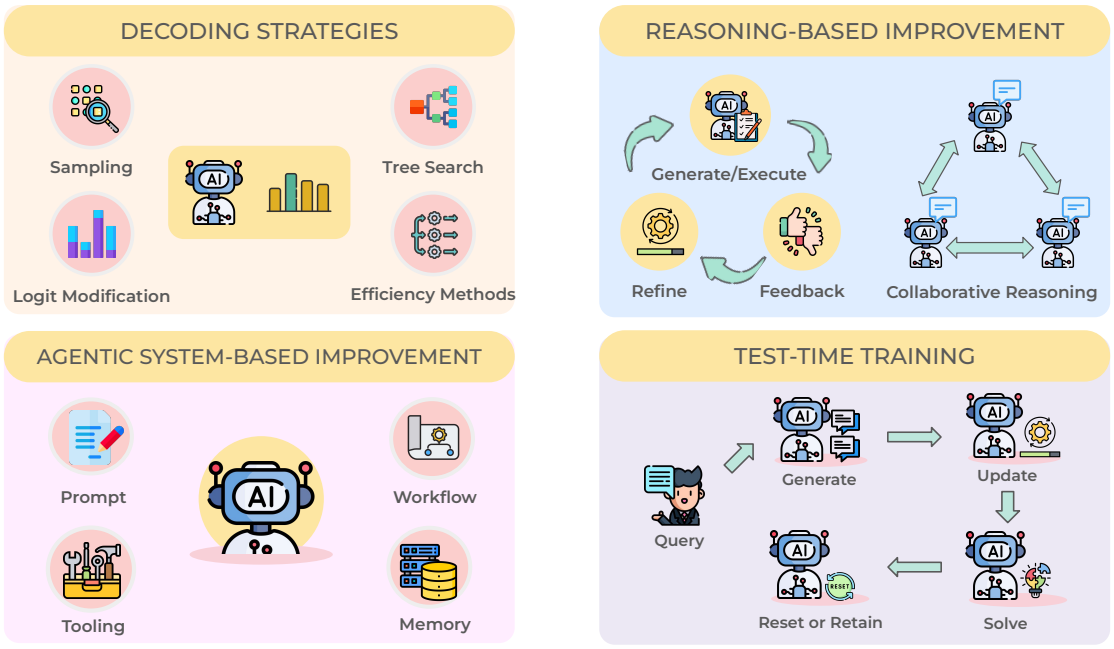

Figure 7: **Overview of inference refinement in the self-improvement system.** This stage focuses on how the model improves output quality during inference without permanently updating its parameters. *(i) Decoding Strategies:* The model improves generation by modifying decoding procedures, such as sampling control, logit modification, tree search, or efficiency-oriented decoding methods. *(ii) Reasoning-Based Improvement:* The model refines reasoning trajectories through iterative processes such as generate–execute–refine loops, feedback-based revision, or collaborative reasoning among multiple agents. *(iii) Agentic System-Based Improvement:* The model operates within agentic systems that coordinate prompts, tools, workflows, and memory modules to enhance task performance. *(iv) Test-Time Training:* The model temporarily adapts its parameters during inference by generating feedback signals and updating the model before producing the final solution.

actions across multiple steps. As shown in Figure 7, these improvement mechanisms for inference refinement can be grouped into the following categories:

- **Decoding Strategies** explicitly guide output generation at the token or sequence level to steer the model toward more probable or logical sequences.
- **Reasoning-Based Improvement** structures the model's intermediate thought process for planning and iterative decision-making, allowing it to think before it speaks.
- **Agentic System-Based Improvement** extends refinement beyond the core model by enhancing system-level components, including prompts, memory architectures, tool use, and workflow design.
- **Test-Time Training** modifies the model parameters at the moment of inference, enabling task-specific policy evolution that is uniquely tailored to the current query. Unlike static deployment, this allows the model to perform local gradient updates based on the specific context or provided constraints.

Inference refinement effectively extends self-improvement beyond the pre-deployment phase into a continuous, real-time process. By integrating these test-time mechanisms with training-time optimization, the system sustains an active evolutionary loop during actual deployment. This synergy empowers the model to adaptively bridge the gap between its static internalized knowledge and the evolving demands of complex real-world tasks, leading to a state of persistent and autonomous evolution. In the following section, §5.2 presents decoding strategies for improving inference quality and efficiency. §5.3 then discusses reasoning-based refinement methods that enhance intermediate reasoning processes. §5.4 further explores agentic system-based approaches, incorporating tools, memory, and workflows. Finally, §5.5 introduces test-time training methods, followed by an overall discussion of inference refinement for self-improvement in §5.6.

5.2 Decoding Strategies

Early approaches to inference-time self-improvement primarily focused on enhancing the decoding process itself by leveraging sampling, tree search, logit modification, and parallelization techniques. These methods improve the quality and efficiency of generation without modifying model parameters, by intervening directly at the token or sequence level. By scaling test-time compute, they allow models to produce more specialized or coherent outputs, serving as foundational techniques that provide key insights into high-level self-improvement paradigms.

5.2.1 Sampling-Based

Self-Consistency. Self-Consistency (Wang et al., 2023b) generates multiple reasoning paths via stochastic decoding and identifies the most frequent response, following the intuition that the correct reasoning processes, even if they are diverse, tend to reach the same final answer. Universal Self-Consistency (USC) (Chen et al., 2024e) alters this mechanism for free-form textual tasks by utilizing an LLM for determining consistency of candidates. Another extension weighs each sample by a self-assessed confidence score and performs a confidence-weighted majority vote (Taubenfeld et al., 2025; Razghandi et al., 2025). The general form of majority voting based decoding can also be conditioned on different trajectories to improve generation robustness such as in multimodal applications for hallucination reduction (Fang et al., 2025d). While self-consistency is a simple approach that can improve robustness, it often struggles in scenarios where the model exhibits systemic biases, as the aggregation of multiple reasoning paths can converge on a consistently held incorrect conclusion rather than correcting the error (Chen et al., 2024a; Byerly & Khashabi, 2025).

Best-of-N. Best-of-N extends the principle of majority voting with a scoring mechanism such as a trained external verifier (Cobbe et al., 2021), reward model (Stiennon et al., 2020), or heuristic such as model uncertainty (Zhu et al., 2025b; Kang et al., 2025) to score and select the best response from candidates. These techniques improve reasoning accuracy and reduce hallucinations by exploiting diversity in the model's sampling distribution. Scaling methods for Best-of-N include increasing the number of verifiers to create a more diverse verification signal (Lifshitz et al., 2025) or increasing the number of LLMs which sample candidates at inference time for diverse generation (Xia et al., 2025). However, the effectiveness of scaling candidate generation is limited by the precision of the verifier. For tasks without automatic verifiers, performance tends to plateau because standard selection methods like majority voting and reward models fail to identify rare correct samples as the number of generations scales (Brown et al., 2024).

Synthesis. The previous methods operate on the assumption that correct answers exist within the candidate set, but in cases where all candidates are incorrect, they fail to produce a correct answer. Synthesis methods solve this problem by analyzing candidate responses generated by a single LLM or an ensemble to formulate a final solution. Vernikos et al. (2024) and LLM-Blender (Jiang et al., 2023) leverage a task-specific, generative fuser or synthesis model to combine candidate responses while Mixture-of-Agents (Wang et al., 2025b) leverages the reasoning capability of an off-the-shelf LLM for fusion. This approach is extended by CoT-based Synthesizer (Zhang et al., 2025a) which utilizes a CoT model for candidate fusion, and GSR (Wang et al., 2025d) by removing the fuser model and performing candidate synthesis by re-prompting the LLM with the candidate solutions. Synthesis methods tend to outperform majority voting-based methods on difficult problems where there are few correct candidate responses as they are not strictly limited by the quality of the candidate responses. However, synthesis-based methods yield substantial computational cost which scales with the number of utilized LLMs and the number of generated candidates.

5.2.2 Tree Search

Beam Search. Beam search maintains a fixed number of top candidate continuations at each decoding step, balancing exploration and exploitation to increase the likelihood of finding higher-probability sequences. Variants can utilize internal self-evaluation (Xie et al., 2023c) to guide step-wise decoding toward logically coherent trajectories while other approaches derive evaluation signals from verifiers, reward models, or retrieval mechanisms, to prune beam candidates for enhanced accuracy and consistency across reasoning steps (Trinh et al., 2024; Zhu et al., 2024). Adaptive strategies dynamically adjust the search process to balance computational efficiency and reasoning, including dynamically narrowing beam size (Hu et al., 2025b) and re-allocating the search budget to steps more prone to reasoning errors (Quamar et al., 2025).

Monte Carlo Tree Search. Monte Carlo Tree Search (MCTS) traverses a decision tree by iteratively exploring and simulating candidate paths to identify high-performing trajectories. For token-level MCTS, LLMs provide next-token probabilities that serve as heuristics to guide the MCTS decoding (Zhang et al., 2023). A popular extension is value-function-guided MCTS, which uses learned value functions to rank paths and guide token-level search, used by

frameworks like PPO-MCTS (Liu et al., 2024b). Most MCTS-based approaches re-frame language model inference as an explicit search process over branching sequences of reasoning or solution states. Tree-of-Thought (Yao et al., 2023b) uses tree search methods such as MCTS variants or simpler methods such as BFS/DFS to navigate a tree of intermediate reasoning based on the LLM's self-assessment of candidate thoughts. Many approaches utilize a learned value function with MCTS allowing for more precise guidance and enabling increased search depth (Wan et al., 2024; Zhang et al., 2024b). Other search variants have been proposed including Graph-of-Thoughts (Besta et al., 2024) which generalizes search to a directed graph structure, and Forest-of-Thought (Bi et al., 2025) which maintains multiple trees in parallel aggregating solutions across trees. The integration of structured search procedures transforms decoding into a formal sequential decision-making process, enabling models to overcome the linear limitations of greedy auto-regressive generation.

5.2.3 Logit and Probability Adjustments

Reward Guidance. Reward model alignment is a form of self-improvement integrated directly into the decoding process, where the model's probabilistic token predictions are dynamically adjusted using a signal that reflects human preferences or task-specific objectives. Reward-guided alignment shifts the alignment process from training time to decoding time, serving as an inference-time analogue to traditional Reinforcement Learning from Human Feedback (RLHF). The ARGS (Khanov et al., 2024), PAD (Chen et al., 2025b), and Controlled Decoding (CD) (Mudgal et al., 2024) frameworks enhance alignment of generated text with human preferences by scoring partial trajectory continuations which serve as a reward mechanism for token-level guidance of the text generation process. A reward model or value function is utilized to evaluate the expected reward of each candidate token, allowing the system to steer the generation process toward high-reward outcomes. While these methods typically evaluate the next immediate token candidates, DeAl (Huang et al., 2025b) extends this paradigm by employing a heuristic-guided search which utilizes short greedy lookaheads to estimate future rewards of a trajectory. GenARM (Xu et al., 2025c) uses an autoregressive reward model designed to provide direct token-level guidance and addresses prior works' reliance on applying trajectory-level RMs trained only on complete responses when evaluating partial sequences. Reward guidance enables efficient alignment of LLMs and can recover a significant amount of the performance gap versus full fine-tuning, offering a favorable accuracy-cost trade-off (Xu et al., 2025c). However, this often comes at the cost of having to train a dedicated reward model.

Model Logit Blending. Other works provide alignment by linearly mixing the logits of a base and reference model or an ensemble of models. Decode-Time Realignment (DeRa) (Liu et al., 2024e) and Inference-Time Realignment (InRa) (Zhu et al., 2026) utilize logit blending between an aligned model and a reference model to enable the user to dynamically control the degree of alignment. Extending these concepts, Integrated Value Guidance (IVG) (Liu et al., 2024h) proposes an advanced hybrid approach, integrating both token-level implicit guidance (derived from the difference between an aligned and a reference model) and chunk-level explicit guidance from a separate trained value scorer. Multi-Objective Decoding (MOD) (Shi et al., 2024) balances multi-dimensional user requirements by applying a linear combination of base model predictions, such as safety or coding proficiency, based on user-specified preference weightings. Pack of LLMs (Mavromatis et al., 2024) calculates importance weights by solving an optimization problem to minimize perplexity over the input prompt, ensuring the final output prioritizes models with the highest demonstrated expertise. Fusing knowledge from multiple LLMs through token-level ensembles allows for interpretable control over model outputs and enables integration of diverse model strengths to improve task performance without the need for retraining.

Contrastive Decoding. Contrastive Decoding (Li et al., 2023b) enhances open-ended text generation and reasoning quality by combining the outputs of strong and weak models to suppress generic, repetitive, or incoherent text, improving specificity and factual grounding (O'Brien & Lewis, 2023). DoLa (Chuang et al., 2024) alters this framework by contrasting logits from higher and lower layers, improving open-ended generation, ALW (Zhou et al., 2025b) dynamically selects an optimal shallow layer to contrast with the final layer, better disentangling noise from critical signals to improve reasoning, and the idea of contrastive decoding is also adopted in multimodal decoding to combat hallucinations in visual-language models (VLMs) such as in HALC (Chen et al., 2024g) and VCD (Leng et al., 2024).

Constrained Decoding. Constrained decoding ensures that model-generated sequences adhere to specific syntactic, semantic, or logical rules. This process typically functions by modifying next-token logits to meet the imposed constraints, such as by masking invalid tokens and setting their logits to negative infinity. Early work pioneering unsupervised text summarization uses contextual matching (Zhou & Rush, 2019) from pretrained language model representations to constrain the prefix semantics of the generated text. Neural semantic parsing approaches build on explicit modeling of target semantic graph states to restrict the next token vocabulary for decoding (Zhou et al.,

2021a;b; 2022). NeuroLogic (Lu et al., 2022) enforces lexical constraints such as the inclusion or exclusion of specific words or phrases, leveraging an A*-style lookahead heuristic to estimate future constraint satisfaction. Other approaches modify logits through repetition or length penalties (Zhu et al., 2023) or focus on syntax constraints that ensure valid code-based formatting or grammatical validity (Scholak et al., 2021). While typically applied to token-level methods, NATURALPROVER (Welleck et al., 2022) extends constrained methods to sequences and instead uses a combination of sequence log-probability and constraint satisfaction for candidate ranking in stepwise beam search.

These approaches shift alignment and control from training time to inference time by directly manipulating next-token probabilities using reward signals, auxiliary models, or structural constraints. By operating at the level of logits and partial trajectories, they enable flexible steering of model behavior, trading additional decoding computation for fine-grained, task-specific control without requiring full model retraining.

5.2.4 Efficiency-Oriented Methods

While the prior approaches aim to improve test-time accuracy, decoding strategies such as speculative decoding and parallel decoding are primarily designed to enhance inference efficiency.

Parallel Decoding. Rather than generating tokens sequentially, these methods reframe autoregressive decoding into a parallel formulation to accelerate inference. This includes Jacobi-based approaches, which apply Jacobi and Gauss-Seidel fixed-point iteration methods to predict multiple tokens simultaneously (Santilli et al., 2023; Fu et al., 2024). Non-autoregressive generation that produces all output tokens together has been explored in tandem with the release of the Transformer architectures, such as in applications of neural machine translation (Gu et al., 2018; Wang et al., 2018; Ghazvininejad et al., 2019; Lee et al., 2020; Zhou & Keung, 2020; Brimacombe & Zhou, 2023). Recently, this paradigm resurgence manifests in the form of diffusion language models (Li et al., 2022; Sahoo et al., 2024; Nie et al., 2025) to generate different tokens in a sequence in parallel. Additionally, beyond the token-level parallel decoding, Skeleton-of-Thought (SoT) decoding (Ning et al., 2024) improves efficiency by first generating a high-level outline of a response and then expanding each segment in parallel. The idea of parallel generation is incorporated in many LLM ensembles. Systems can determine which aspects of a task are parallelizable, allowing them to be executed asynchronously by different agents (Yu et al., 2025c; Xia et al., 2025).

Speculative Decoding. Speculative decoding accelerates autoregressive generation by using a lightweight model to draft multiple candidate tokens that are then verified in parallel by the base model, committing only those that pass verification (Leviathan et al., 2023; Chen et al., 2023; Xia et al., 2023). This paradigm significantly enhances inference efficiency by enabling the simultaneous verification of multiple drafted tokens in a single parallel step, circumventing the sequential delays inherent in standard autoregressive decoding. Other lines of research drop the drafting model and integrate additional decoding heads onto the target model to enable parallel, speculative drafting (Cai et al., 2024a; Li et al., 2024i). Self-drafting models also achieve acceleration through adaptively skipping intermediate layers (Zhang et al., 2024c) or implementing early exiting (Liu et al., 2024c) to more efficiently generate draft tokens from the base model's own internal representations (Xia et al., 2024a). In addition, retrieval is also incorporated to provide draft tokens for speculative decoding (He et al., 2024). Approaches can also relax the constraint of keeping the target generation the same as the original model, such as Chunk-Distilled Language Modeling (CD-LM) (Li et al., 2025i) that improves both decoding speed and adaptation of generation to new knowledge through chunk-level data distillation with fine-grained in-context retrieval at inference. We organize the reviewed works by their decoding strategy in Table 9. In addition, novel paradigms of token efficient LLM decoding such as utilizing multimodal architectures by processing text as images exist (Li et al., 2025h; Wei et al., 2025a), but we omit the detailed discussions here.

5.3 Reasoning-Based Improvement

Reasoning-based improvement methods extend beyond decoding-level enhancements by focusing not only on how outputs are generated, but also on the dynamic, multi-step process of how models think, plan, and refine their reasoning. These approaches introduce structured inference-time mechanisms that enable models to incorporate feedback, decompose complex tasks into intermediate subgoals, or distribute reasoning across multiple interacting agents. These methods effectively instantiate a model optimization loop at inference time, where reasoning trajectories are iteratively improved through feedback, enabling refinement through experience rather than explicit policy updates.

5.3.1 Feedback-Based Reasoning

Feedback-based reasoning transforms static inference into a dynamic, closed-loop process by integrating evaluative signals to iteratively refine model outputs. This mimics the generation-reward-optimization process of model opti-

Table 9: **Summary of decoding-based inference-time self-improvement methods.** Early approaches improve generation by modifying the decoding process to enhance output quality and efficiency without updating model parameters.

Decoding Strategy	**Methods**
Self-Consistency	SC (Wang et al., 2023b), USC (Chen et al., 2024e), CISC (Taubenfeld et al., 2025), CER (Razghandi et al., 2025)
Best-of-N	Stiennon et al. (2020), Cobbe et al. (2021), MAV (Lifshitz et al., 2025), UnCert-CoT (Zhu et al., 2025b), Self-Certainty (Kang et al., 2025)
Synthesis	LLM-Blender (Jiang et al., 2023), LMCor (Vernikos et al., 2024), GSR (Wang et al., 2025d), MoA (Wang et al., 2025b), CoT-based Synthesizer (Zhang et al., 2025a)
Beam Search	Xie et al. (2023c), DBS (Zhu et al., 2024), AlphaGeometry (Trinh et al., 2024), PRM-BAS (Hu et al., 2025b), AdaBeam (Quamar et al., 2025)
Tree Search	ToT (Yao et al., 2023b), TS-LLM (Wan et al., 2024), MCTSr (Zhang et al., 2024b), PPO-MCTS (Liu et al., 2024b)
Reward Guidance	ARGS (Khanov et al., 2024), CD (Mudgal et al., 2024), DeAL (Huang et al., 2025b), GenARM (Xu et al., 2025c), PAD (Chen et al., 2025b)
Logit Blending	DeRa (Liu et al., 2024e), MOD (Shi et al., 2024), IVG (Liu et al., 2024h), Pack of LLMs (Mavromatis et al., 2024), InRa (Zhu et al., 2026)
Contrastive Decoding	CD (Li et al., 2023b), O'Brien & Lewis (2023), DoLa (Chuang et al., 2024), ALW (Zhou et al., 2025b), HALC (Chen et al., 2024g), VCD (Leng et al., 2024)
Constrained Decoding	Contextual Matching (Zhou & Rush, 2019), PICARD (Scholak et al., 2021), APT (Zhou et al., 2021a; 2022), NEUROLOGIC A*esque (Lu et al., 2022), NATURALPROVER (Welleck et al., 2022), Penalty Decoding (Zhu et al., 2023)
Parallel Decoding	NAR (Gu et al., 2018; Ghazvininejad et al., 2019; Zhou & Keung, 2020), PGJ (Santilli et al., 2023), Fu et al. (2024), Diffusion LM (Li et al., 2022; Sahoo et al., 2024; Nie et al., 2025), SoT (Ning et al., 2024), DynTaskMAS (Yu et al., 2025c), Xia et al. (2025)
Speculative Decoding	Speculative Decoding (Leviathan et al., 2023), SpS (Chen et al., 2023), SpecDec (Xia et al., 2023), Medusa (Cai et al., 2024a), Eagle (Li et al., 2024i), Zhang et al. (2024c), EESD (Liu et al., 2024c), REST (He et al., 2024), CD-LM (Li et al., 2025i)

mization, but uses signals for output refinement rather than parameter updates. Refinement loops can be categorized by the source of feedback: feedback may be generated internally by the model through self-evaluation or obtained from external sources such as critique models, environments, or verification signals. Self-feedback methods generally follow a reason–critique–refine paradigm, where the model evaluates and updates its own outputs, while external feedback approaches may additionally incorporate reason–execute–refine loops, in which solutions are validated through interaction with an environment.

Self-Feedback. Independent iterative improvement does not rely on external models to give feedback in the improvement loop. Early works including Self-Refine Madaan et al. (2023) and RCI Kim et al. (2023) propose iterative self-refinement loops that utilize the base LLM for feedback and refinement of a response. Feedback takes the form of a specific improvement, which is integrated into the prompt for the next iteration. Natural language self-critiques can also be applied at the step level of a reasoning chain, acting as a natural language PRM (Yansi Li et al., 2025). The Self-Debugging framework (Chen et al., 2024f) performs "rubber ducking" for code translation where self-feedback is derived from the consistency of the model's explanation of its own code with the problem description and predicted behavior. Internal confidence-based methods like IoE (If-or-Else) (Li et al., 2024d) leverage the model's uncertainty for selective refinement of low-confidence reasoning steps, which acts as an internal filter to prevent over-criticizing and incorrectly altering highly confident responses.

Iterative self-improvement behaviors can be instilled during training and employed at inference time without external feedback. These approaches internalize self-correction during training such that the model autonomously performs self-refinement once deployed. One approach integrates self-verification or self-refinement into its RL training objective to teach LLMs the explicit ability to recursively detect and correct previous mistakes over sequential turns (Qu et al., 2024) (Zeng et al., 2025). The Self-rewarding correction framework (Xiong et al., 2025) unifies a generator and evaluator into a single LLM and is trained to generate explicit self-assigned reward tokens alongside its reasoning, serving as an internal evaluation signal to guide iterative refinement. Current LLMs often exhibit limited capacity to enhance their own outputs through intrinsic mechanisms alone, as they are limited by the quality and consistency of the model's self-assessment of its reasoning (Huang et al., 2024a). These methods are highly sensitive to inference conditions: negative prompts can encourage over-criticism, and their internal decision-making process lacks stability, often requiring operation under zero-temperature decoding to prevent randomness from flipping the self-correction decision (Liu et al., 2024a).

External Feedback. These methods rely on feedback from an external critique model that provides actionable suggestions to iteratively refine a response. The Reflexion Shinn et al. (2023) and DCR (Detect Critique Refine) (Wadhwa et al., 2024) frameworks both employ evaluator/detector models to compute a scalar reward reflecting the agent's performance, which is interpreted by a critique model to provide specific textual feedback. PerFine (Maram et al., 2025) leverages a critic LLM for personalized text generation. The critic LLM explicitly generates detailed and actionable guidance to align the tone, vocabulary, and topic relevance of a generated response. Other frameworks (Paul et al., 2024; Xi et al., 2024b; Hossain et al., 2026) extend the use of a critic model to provide fine-grained textual feedback on intermediate reasoning steps. These methods provide rich and actionable feedback signals, but their effectiveness is limited by the accuracy and consistency of the critic.

In contrast to critique-model approaches that rely on a dedicated LLM to generate natural-language feedback, these methods receive supervision from environmental observations or external evaluators that provide scalar scores, binary results, or verification signals. The CRITIC framework (Gou et al., 2024) derives signals from calls to external tooling (search engines, code interpreters) to verify candidates, while TPO (Test-Time Preference Optimization) (Li et al., 2025g) harnesses the ability of the base model to interpret numerical reward signals from a reward model into textual critiques. In these works, the base model interprets the external signal into a natural-language critique or is prompted to diagnose its failures.

This paradigm is common for code generation and verification, as it mimics human debugging by enabling the model to refine flawed code based on execution feedback such as test cases and runtime errors (Ni et al., 2023). Feedback is then stored in the conversation history or appended to the next prompt. Self-Debugging (Chen et al., 2024f) can generate a reflective message from the combination of post-execution feedback (execution traces or unit test results) and a self-explanation of its generated code as a refinement signal for each iteration. LDB (Large Language Model Debugger) (Zhong et al., 2024a) decomposes programs into smaller blocks and uses intra-execution feedback from a debugger, tracking variable states at different breakpoints. The AlphaVerus framework (Aggarwal et al., 2025) distills feedback from a formal verifier into actionable critiques within its formally verified code generation process. The verifier returns objective feedback on the generated code through its localized error messages, which are used to generate refinements structured as a guided tree search. While these approaches provide highly targeted and objective feedback, they are limited to tasks with well-defined verification signals and require the model to correctly interpret the feedback. We organize these feedback-based reasoning methods in Table 10.

5.3.2 Planning-Based Reasoning

Planning-based reasoning methods enable models to tackle complex objectives by breaking down a high-level goal into smaller, manageable sub-goals that define the sequence of steps and interactions required to achieve the goal.

Table 10: **Summary of feedback-based inference-time refinement methods.** These approaches improve generation through multi-step feedback loops, leveraging diverse feedback sources to iteratively refine outputs without updating model parameters.

Method	**Feedback Source**	**Feedback Type**
Self-Refine (Madaan et al., 2023)	Base LLM	Textual Critique
Reflexion (Shinn et al., 2023)	External Evaluator	Score → Textual Critique
RCI (Kim et al., 2023)	Base LLM or Verifier	Textual Critique or Verification
LEVER (Ni et al., 2023)	Verifier	Score
Self-Debugging (Chen et al., 2024f)	Base LLM or Verifier	Textual Critique or Verification
IoE Prompt (Li et al., 2024d)	Base LLM	Internal Confidence
CRITIC (Gou et al., 2024)	External tools	Execution Results
LDB (Zhong et al., 2024a)	Debugger LLM	Textual Critique
REFINER (Paul et al., 2024)	Critique LLM	Textual Critique
DCR (Wadhwa et al., 2024)	Critique LLM	Textual Critique
RISE (Qu et al., 2024)	Base LLM	Internalized
PANEL (Yansi Li et al., 2025)	Base LLM	Textual Critique
PerFine (Maram et al., 2025)	Critique LLM	Textual Critique
AlphaVerus (Aggarwal et al., 2025)	Verifier	Score → Textual Critique
TPO (Li et al., 2025g)	Reward Model	Score → Textual Critique
ID-Sampling (Chen et al., 2025f)	Base LLM	Textual Critique
Xiong et al. (2025)	Base LLM	Internal Reward Token

Open-Loop Planning. Planning in natural language allows the model to explore a wider range of conceptual ideas, which guides generation towards improved outcomes and leads to successful gains in final generation accuracy (Wang et al., 2024a). Initial approaches to planning focused on structured prompting techniques designed to elicit a global blueprint from the model prior to response generation (Zhou et al., 2023a; Wang et al., 2023a). By explicitly decoupling the planning stage from the final reasoning execution, these methods introduced a systematic decomposition of tasks, requiring the model to break down complex queries into manageable sub-tasks before generating a final answer. Plans can also be represented in symbolic forms like Planning Domain Definition Language (PDDL), which is better suited for environments with intricate constraints, where direct LLM-generated plans often suffer from correctness and executability issues (Huang et al., 2024b). In this paradigm, the LLM serves primarily as a formalizer, translating natural language tasks into PDDL (Liu et al., 2023; Guan et al., 2023). These approaches rely on the ability to anticipate all necessary steps upfront, ensuring a structured plan is established prior to execution.

Closed-Loop Planning. Inference-time planning equips agents with decision-making capabilities, enabling them to handle multi-step tasks by dynamically modifying their future plan of actions. ReAct (Yao et al., 2023c) pioneered the use of environmental feedback for inference-time plan improvement by interleaving reasoning traces and task-specific actions. Rather than maintaining a fixed plan, this synergy allows the model to perform dynamic reasoning to create, maintain, and adjust high-level action plans based on external observations. AdaPlanner (Sun et al., 2023) advances this concept by utilizing an explicit closed-loop system where a LLM acts as both a planner and a refiner. AdaPlanner proactively revises the entire future plan when environmental observations deviate from predictions. Furthermore, its "refine-then-resume" mechanism enables the agent to continue from an intermediate breakpoint rather than restarting an episode from scratch. Similar loops are critical for LLMs governing open-world and robotics planning, enabling agents to adjust trajectories based on environmental feedback to achieve long-horizon goals (Wang et al., 2023d; Huang et al., 2023b; Yang et al., 2024d).

Certain search methods can be interpreted as closed-loop planning systems by treating reasoning as a heuristic search over a state space. Works like Tree-of-Thoughts (Yao et al., 2023b) and Graph-of-Thoughts (Besta et al., 2024) allow the model to look ahead by generating multiple potential next steps and backtrack to previous stages if a current reasoning path is deemed unlikely to succeed. This enables the model to treat reasoning as a dynamic optimization problem, pruning suboptimal paths to converge on optimal action sequences that satisfy complex, long-horizon

objectives. These methods transition the LLM from a simple generator to a deliberative planner that navigates a global search space to solve constrained long-horizon tasks.

5.3.3 Collaborative Reasoning

Collaborative reasoning extends beyond the autonomy of a single model by distributing the reasoning process across an ensemble of interacting agents. Through structured interaction protocols and role-based divisions, agents iteratively build upon and refine each other's reasoning traces, acting as a cohesive reasoning unit.

Role Specialization. Role-based architectures decompose complex reasoning tasks into functional hierarchies or distributed role structures where agents adopt distinct personas. This specialization mimics human organizational structures to manage cognitive load and improve precision. Hierarchical frameworks like MetaGPT (Hong et al., 2024) encode human-like workflows through an assembly-line paradigm where agents are assigned specialized roles such as product managers, architects, and engineers. This decomposes complex objectives into sequences of specialized interdependent subtasks, which effectively reduces cognitive load for individual agents and mitigates the risks of cascading hallucinations. CAMEL (Li et al., 2023a) and Autogen (Wu et al., 2024a) take a more decentralized approach and use role specialization as an extension of the reasoning process itself to perform iterative refinement through a continuous dialogue loop, rather than relying on a predefined functional hierarchy. This distinction highlights how role specialization can either serve as an organizational scaffold for structured task execution or as an interactive mechanism that shapes collaborative reasoning dynamics through iterative agent communication.

Debate. The Multi-Agent Debate (MAD) framework facilitates structured dialogue among LLMs, directly enabling them to iteratively refine and update their responses. The framework, introduced by Du et al. (2024), replaces the self-reflection loop with a structured debate where agents generate their own solutions and then update their own answers based on the solutions of their peers. To promote diversity of candidates and reduce overcommitment in initial, potentially incorrect solutions, Reconcile (Chen et al., 2024b) utilizes diverse models and Liang et al. (2024) assigns different agent roles to force consideration of alternative viewpoints. While MAD strategies enhance logical depth, they introduce significant resource challenges. Many strategies reduce the density of information flow in the communication network to improve the efficiency of debate. One approach groups agents and restricts the inter-group communication to a summarization of intra-group exchanges or their final viewpoints, while intra-group agents share full access to each other's reasoning (Liu et al., 2024f; Wang et al., 2024e). Li et al. (2024k) moves away from the standard fully-connected communication network to a sparse topology, significantly reducing inference costs, while achieving comparable or superior accuracy compared to fully-connected networks.

5.4 Agentic System Improvement

The previous sections focused on inference-time improvement with respect to the model itself, examining how test-time compute can enhance generation by modifying decoding procedures and structuring internal reasoning processes, while keeping the surrounding execution environment fixed. In contrast, this section considers inference-time improvement at the system level. Agentic systems augment LLMs with persistent state, decision-making loops, external tools, and inter-agent coordination, forming a broader computational substrate in which the model operates. By dynamically adapting prompts, memory structures, tool libraries, communication topology, and workflows during inference, agents can reconfigure the environment that shapes model behavior. This enables test-time self-improvement not by altering the model's internal generation dynamics, but by evolving the system that governs how and where reasoning occurs, yielding greater adaptability, robustness, and task-specific capability without parameter updates.

5.4.1 Prompts

Prompt optimization allows agents to dynamically modify primary and meta-prompts to improve reasoning, planning, and task performance. Early sampling-based approaches utilize best-of-n to select top-performing candidate prompts scored by a trained reward/preference model (Sun et al., 2024b) or execution accuracy on validation examples (Zhou et al., 2023c), while other approaches directly train a model for prompt generation or refinement (Deng et al., 2022; Yang et al., 2024a; Yao et al., 2024). This optimization process can be extended to the selection of in-context demonstrations, where models dynamically identify the most relevant or informative examples to include in the prompt to maximize task accuracy (Wan et al., 2023a;b), as well as gradient based prompt segment optimizations such as in GSPO (Li et al., 2024f).

Search-based approaches, including PromptAgent (Wang et al., 2024f) and MCTS-OPS (Yu et al., 2025b), formulate prompt optimization as a sequential decision problem and employ Monte Carlo Tree Search guided by self-reflective

signals and task-specific performance feedback to iteratively refine prompts before performing task-specific inference. Evolutionary-based frameworks like EvoPrompt (Guo et al., 2024a), AlphaEvolve (Novikov et al., 2025b) and Promptbreeder (Fernando et al., 2024) employ conventional genetic algorithms like mutation and crossover over a population of archived prompts to guide search. Promptbreeder and AlphaEvolve extend evolution to the mutation-prompts (meta-prompts) which instruct how to modify primary prompts within an evolving population provided to the language model. This improves the instructions used for reflective prompt generation, enabling the discovery of more effective prompting strategies.

Prompt optimization methods can employ iterative improvement loops that leverage natural-language critiques to refine agent instructions, a process often described as textual gradient descent. ProTeGi (Pryzant et al., 2023), TextGrad (Yuksekgonul et al., 2025) and LLM-AutoDiff (Yin & Wang, 2025) employ this strategy as an optimization phase before deploying the prompt. ProRefine (Pandita et al., 2025) extends textual-gradients directly to inference-time by dynamically refining prompts to adapt to each generation step with critiques from a feedback LLM, allowing the model to adapt to evolving task requirements at inference time.

Within multi-agent systems, prompts can provide agents with distinct roles that define their responsibilities and enhance test-time performance through explicit decomposition of reasoning processes into complementary functions. Early works implement static role-playing through prompting (Qian et al., 2024; Li et al., 2023a; Hong et al., 2024; Wu et al., 2024a), but recent works, including MASS (Zhou et al., 2026a), Promptomatix (Murthy et al., 2025) and EvoAgent (Yuan et al., 2025) extend the idea of automatic prompt optimization across multi-module or multi-agent workflows to optimize agent roles and communication protocols. These systems revise system prompts across interacting components while adapting the agent workflow as a whole, enabling coordinated specialization and adaptation at the task or query level.

5.4.2 Memory

Memory is a core capability of agentic systems, enabling agents to adapt across interactions with their environment. Unlike standard LLMs, whose behavior is constrained to a fixed context window, agents maintain persistent state that extends beyond a single interaction, allowing them to accumulate and reuse knowledge. Long-term memory stores reusable experiences, strategies, tools, and agent configurations that persist across sessions, supporting cross-task generalization and continual improvement. In contrast, working (short-term) memory captures transient context within a task to support intermediate reasoning and decision-making.

Designing effective memory systems therefore requires specifying both the representation of stored knowledge and the mechanisms used for retrieval, updating, and consolidation. Building on the paradigm of retrieval-augmented generation (RAG), recent agentic systems integrate retrieval directly into the reasoning loop, transforming it from a static lookup into a dynamic process. Rather than performing a single retrieval step, agents can iteratively query memory, refine search strategies, and write newly acquired information back into the memory store. These mechanisms enable agents to efficiently recall relevant information, avoid redundancy, and maintain coherent internal state across long-horizon interactions. In this section, we focus primarily on long-term memory structures and the methods used to optimize them, as these components are central to enabling persistent learning and adaptive behavior in agentic systems.

Structure. Long-term memory structures vary across agents and affect retrieval quality and performance. Many systems use vector-based memory with RAG-style retrieval for chatbot interaction to address the limitations posed by the LLM context window, such as MemGPT (Packer et al., 2024) and MemoryBank (Zhong et al., 2024b) which store semantic memories and NeuroCache (Safaya & Yuret, 2024) which stores prior model states. Relevant self-judged experiences can be retrieved through RAG to inform new interactions and integrate newly acquired knowledge back into memory, increasing the agent's capability over time (Ouyang et al., 2025). Memory management has started to incorporate more sophisticated abstraction and summarization for efficient memory management of acquired experiences. Mem0 (Chhikara et al., 2025), A-Mem (Xu et al., 2025b), and G-Mem (Zhang et al., 2025b) all update knowledge graph-based structures for relations among factual entities, aiding efficient retrieval over extended interactions. Another common structural trend is partitioning memory into different logical substrates, mimicking the complexity of human memory and preventing retrieval inefficiencies (Wu et al., 2025f). For example, HINDSIGHT (Latimer et al., 2025) distinguishes between facts, experiences and beliefs, and MITRIX (Wang & Chen, 2025) distinguishes between procedural, episodic and semantic memories. These structural approaches provide epistemic clarity by separating objective evidence from subjective beliefs, which prevents context dilution and ensures that agents can maintain stable, traceable reasoning styles as their internal beliefs evolve over time.

Optimization. Memory updates are triggered by new interactions or dialogue turns as new information is incorporated into the current short-term context. Mem0 (Chhikara et al., 2025) and Memory-R1 (Yan et al., 2026) utilize CRUD-style operations (Add, Update, Delete, Noop) by employing an LLM to determine the appropriate operation for integrating new experiences into long-term memory. A-Mem (Xu et al., 2025b) decides on specific actions for neighboring memories including strengthening, merging, or pruning nodes of its knowledge graph. HINDSIGHT (Latimer et al., 2025) similarly evolves its memory substrates by synthesizing fragmented facts and updating belief confidence scores as it ingests new information. Another direction of optimization focuses on managing growing memory stores. SAGE (Liang et al., 2025) and MemoryLLM (Wang et al., 2024h) both limit memory capacity through biological-style optimization inspired by the Ebbinghaus Forgetting Curve, which phases out underutilized knowledge. This forgetting mechanism is used in SAGE to move knowledge from short to long-term memory and in MemoryLLM to manage a fixed capacity memory-pool. Mem1 (Zhou et al., 2026d) and MemAgent (Yu et al., 2026) leverage an RL-learned process of consolidation and pruning of their respective internal states and fixed-length memories to avoid unbounded context growth and keep memory usage constant throughout long-horizon tasks. The ALMA (Xiong et al., 2026) framework moves beyond human-crafted memory modules and uses a meta-agent in an offline phase to automatically discover and implement memory designs and protocols as executable code, bridging the gap between architectural designs and functional performance at test-time. These methods represent a paradigm shift from treating memory as a static knowledge buffer to a dynamic, evolving system state. By formalizing memory management as an optimization problem, agents maintain high retrieval precision and semantic coherence without exceeding constraints on the context window.

5.4.3 Tooling

Tool use is a core capability of LLM agents, enabling interaction with external APIs, databases, and environments beyond the model's parametric knowledge. While standard LLMs are restricted to static tool-calling, agents can be instilled with the ability to dynamically adapt and refine tool usage strategies, including selecting appropriate tools, determining optimal invocation, and adjusting tooling parameters based on task context and execution feedback. Inference-time tool creation, evolution, and usage is inherently connected to agentic planning and reasoning as tools can be retrieved or created on the spot by planners, called directly from agentic reasoning, or executed in workflows (Yao et al., 2023c; Wu et al., 2024b). There is a large body of work that applies training-time approaches that learn optimal tool invocation through SFT (Schick et al., 2023; Tang et al., 2023; Qin et al., 2024; Yang et al., 2023a) and RL-based (Feng et al., 2025a; Qian et al., 2025; Li et al., 2025f; Chai et al., 2025) fine-tuning on tool-calling trajectories, or build a starting set of general purpose tools for retrieval (Yuan et al., 2024a; Qiu et al., 2025a). These methods support accurate inference-time tool invocation, but we focus on post-deployment optimizations and evolution in this section.

Creation and Refinement. Unlike traditional methods that rely on a static tooling library, frameworks increasingly empower LLMs to design and verify applicable tooling in the form of code and documentation at inference time (Qian et al., 2023; Wang et al., 2024b; Cai et al., 2024b; Shen et al., 2026). These systems trigger a tool-making phase when capability gaps are identified in the existing library and often implement iterative refinement processes to test tooling quality and ensure its reliability. These frameworks archive their new tools, enabling reuse and evolution across tasks. While earlier frameworks store complex, task-specific tooling implementations, TTE (Test-Time Tool Evolution) (Lu et al., 2026) decomposes synthesized code into modular units to increase the probability of reuse. Frameworks can also leverage external knowledge sources to support tool construction. Alita (Qiu et al., 2025b) employs a web search–driven agent to retrieve relevant open-source implementations, which are incorporated into on-the-fly tool generation and AutoAgent (Tang et al., 2025a) adopts RAG to query external code repositories, grounding the synthesis of reusable tooling components in retrieved artifacts.

Another axis of tooling improvement at inference time focuses on tooling refinement rather than the generation of new tooling code. Alita-G (Qiu et al., 2025a) refines tools' applications by adapting configurable parameters to new task requirements, allowing tools to be repurposed for new tasks. UCT (Shen et al., 2026) and ToolLibGen (Yue et al., 2025) employ post-inference processes enabling agents to systematically refine, merge and prune their tool library to ensure it remains scalable and avoids redundancy as the library grows. Other approaches explore documentation optimization within the inference prompt, which present tools in concise and structured formats, helping the model to invoke the correct tools more readily. PLAY2PROMPT (Fang et al., 2025b) iteratively refines documentation and usage examples from execution feedback and self-reflection, employing a search-based tool-play process that mimics human trial-and-error to discover tool behaviors without requiring human-labeled data.

Selection. Classic search-based approaches such as ToolLLM (Qin et al., 2024) and ToolChain (Zhuang et al., 2024) leverage tree-based search over API spaces, overcoming limitations of traditional reasoning methods like in ReAct (Yao

et al., 2023c) by enabling multiple simultaneous reasoning traces and easy error correction via backtracking. Building on tree-based approaches, AutoTool (Jia & Li, 2025) transforms open-ended tooling decisions into a constrained graph search that identifies predictable sequential patterns, allowing the system to bypass costly LLM calls and directly select the next tool based on historical trends and contextual relevance. Agents can further streamline the retrieval phase by dynamically updating prompts with the most relevant tools and historical use cases (Gan & Sun, 2025; Qiu et al., 2025a). This ensures the agent operates with a task-specific toolkit and prevents prompt bloat when dealing with vast sets of tools. Another line of research investigates a generative paradigm in which tools are integrated directly into the model's architecture as learned tool tokens, allowing the LLM to trigger external functions as naturally as it generates text (Hao et al., 2023; Wang et al., 2025e). Building on this idea, Chain-of-Tools (Wu et al., 2025a) leverages the hidden states of frozen language models to perform tool selection and evaluation during the reasoning process by analyzing the model's latent representations to determine when and which tool to invoke, ensuring efficient tool usage without compromising the model's reasoning capabilities. Overall, inference-time tool optimization transforms tool usage into a self-expanding procedural ecosystem in which agents can bridge the gaps in their tooling capabilities to solve complex and previously unseen real-world problems.

5.4.4 Workflow and System Evolution

Workflows define the coordination patterns that govern inter-agent communication protocols and interactions with tooling and memory. They act as a blueprint designed to support the agents' reasoning abilities which enable the system to dynamically decompose tasks, call external tools and interpret feedback. Multi-agent systems such as MAD (Du et al., 2024), or manually designed role-based systems like MetaGPT (Hong et al., 2024) assign static roles or structured communication structures which lack adaptiveness in task routing or feedback integration. A theme across more recent work involves automating the agent design process, moving beyond constrained, human-crafted systems. These methods dynamically evolve topology or the whole system architecture including agent configurations and tooling strategies. This allows the system itself to evolve on a per-task or per-query level that dictates the system behavior at test-time. This represents a shift from single-agent reasoning-based planning to optimization of system architecture to support decision making and cooperation between interacting components.

Topology Evolution. Agentic systems dynamically modify their communication structures either pre-deployment or during inference to tailor reasoning structures to task or query level instances. GPTSwarm (Zhuge et al., 2024) explicitly models multi-agent collaboration as an optimizable graph structure, where agent connectivity is dynamically adjusted based on real-time task accuracy from unit tests. The system also employs an LLM ranker, which deactivates low-performing agents from the topology. Approaches commonly use trained communication topology generator models to instantiate task-specific graphs that balance performance with computational efficiency. AMAS (Leong et al., 2025) introduces a dynamic graph selector, which identifies the optimal task-specific graph configuration from a candidate ensemble. AGP (Li et al., 2025a) leverages a Graph Neural Network for topology generation through a pruning process of first selecting active nodes, and then determining communication graph edges which define the direction of information flow. Rather than starting from a fixed topology and relying on pruning, ARG-Designer (Li et al., 2026b) generates a collaboration graph from scratch with an autoregressive graph generator.

Another line of work explores adaptive routing, which treats multi-agent collaboration as a sequential decision problem that yields an implicit communication graph tailored to the evolving task state. Routing-based approaches were originally used in a more static manner to select the expert that is best suited to solve a query (Shnitzer et al., 2023; Lu et al., 2024d), but in multi-agent systems this approach can be used to dynamically determine the communication structure in real-time. DyLAN (Liu et al., 2024i) models agent collaboration as a multi-layer T-FFN where each layer's active nodes constitute the task-specific agent team for that time step. Agent activations (responses) are propagated forward, while a dynamic communication structure is maintained by an LLM ranker that prunes edges to low-performing agents. The Puppeteer framework (Dang et al., 2025) relies on an RL-trained orchestrator to determine the next agent to activate at each step in response to evolving task states in real-time. AgentNet (Yang et al., 2025) removes the central orchestrator and relies on a decentralized approach where each agent decides how to route each task resulting in an implicitly constructed communication DAG. These approaches allow systems to dynamically reconfigure or simplify their collaboration structures in response to performance signals, maintaining strong task accuracy while mitigating the computational overhead typically associated with multi-agent deployments.

Workflow Evolution. Many works perform whole-system evolution, optimizing the entire agent configuration, allowing for holistic adaptation. Popular works, including MASS (Zhou et al., 2026a), ADAS (Hu et al., 2025d), AFlow (Zhang et al., 2025d), AgentSquare (Shang et al., 2025) and The Darwin Gödel Model (Zhang et al., 2026a) exemplify the trend of unified architecture evolution in an optimization phase prior to inference, which consists of evolutionary

searches over populations of archived workflows or a search over the space of agent programs. While these methods employ an offline search process that is more computationally efficient than inference-time optimization, they generally do not allow for self-evolution during inference. If the task distribution changes substantially, a new optimization phase is required to re-optimize the system for the next generation of agents. Inference-time MAS optimization shares similar optimization strategies to offline methods but allows for MAS generation and/or inference-time self-evolution on a per-query basis.

Recent advances in test-time MAS optimization rely on meta-agents which reason over agent configurations and workflows to enhance the system capability rather than directly solving the task. These components are typically trained through SFT on query workflow pairs or objectives that reward downstream task performance. FlowReasoner (Gao et al., 2025) and ScoreFlow (Wang et al., 2025h) train a reasoning-based meta-agent via RL to construct a query-specific MAS in a single pass. Similarly, MAS-GPT (Ye et al., 2025b) utilizes an open-source LLM trained with SFT on query-MAS pairs to generate a query-specific MAS represented as python code. Since these methods generate systems in a single shot, they address the high costs of existing adaptive methods which involve model calls at each intermediate step to adaptively determine workflows. In contrast to single pass methods, MAS-ZERO (Ke et al., 2025) introduces a training-free meta-agent to generate and iteratively refine agent configurations based on the solvability and completeness of the system's output. This balances efficiency with adaptive, inference-specific optimization, enabling the system to correct agentic configurations on the fly.

Another direction of work focuses on dynamic adaptation during the task-solving process or after the completion of task execution. EvoMAC (Hu et al., 2025e) and ANN (Ma et al., 2025c) dynamically evolve agent prompts, communication topology and workflows through an iterative textual backpropagation process with environmental feedback from agent execution (compiler logs, unit tests). EvoAgent (Yuan et al., 2025) automatically extends a single specialized agent into a collaborative system on a per-query basis and evolves system variables and agent settings such as roles and skills through evolutionary operators. EvoAgentX (Wang et al., 2025g) extends these concepts to holistic workflow evolution and integrates multiple state-of-the-art optimization strategies for system refinement including TextGrad for textual backpropagation refinement and reusable modular operators, which can be iteratively evolved and recombined to construct new workflows (Zhang et al., 2025d; Shang et al., 2025). In these approaches, evolution occurs as an iterative process after execution to select the best configuration moving forward. Collectively, these advancements represent a transition from rigid, hand-designed pipelines to fluid, self-organizing architectures that utilize meta-reasoning and environmental feedback to autonomously self-refine. A summary of agentic system evolution papers can be found in Table 11.

Table 11: **Overview of agentic system-based improvement approaches.** Methods are compared across prompting, memory, tool use, and workflow dimensions, with check marks (✓) indicating the presence of each capability.

Method	**Prompts**	**Memory**	**Tools**		**Workflow**	
			Create/Refine	**Selection**	**Topology**	**Unified**
RLPrompt (Deng et al., 2022)	✓	–	–	–	–	–
APE (Zhou et al., 2023c)	✓	–	–	–	–	–
ProTeGi (Pryzant et al., 2023)	✓	–	–	–	–	–
CREATOR (Qian et al., 2023)	–	–	✓	–	–	–
ReAct (Yao et al., 2023c)	–	–	–	✓	–	–
ToolkenGPT (Hao et al., 2023)	–	–	–	✓	–	–
MemGPT (Packer et al., 2024)	–	✓	–	–	–	–
Prompt-OIRL (Sun et al., 2024b)	✓	–	–	–	–	–
OPRO (Yang et al., 2024a)	✓	–	–	–	–	–
Retroformer (Yao et al., 2024)	✓	–	–	–	–	–
PromptAgent (Wang et al., 2024f)	✓	–	–	–	–	–
EvoPrompt (Guo et al., 2024a)	✓	–	–	–	–	–
Promptbreeder (Fernando et al., 2024)	✓	–	–	–	–	–
LATM (Cai et al., 2024b)	–	–	✓	–	–	–
MemoryBank (Zhong et al., 2024b)	–	✓	–	–	–	–
NeuroCache (Safaya & Yuret, 2024)	–	✓	–	–	–	–
MemoryLLM (Wang et al., 2024h)	–	✓	–	–	–	–
GPTSwarm (Zhuge et al., 2024)	✓	–	–	–	✓	–

Continued on next page

Table 11 continued from previous page

Method	Prompts	Memory	Tools		Workflow	
			Create/Refine	Selection	Topology	Unified
DyLAN (Liu et al., 2024i)	–	–	–	–	✓	–
CRAFT (Yuan et al., 2024a)	–	–	✓	–	–	–
ToolChain (Zhuang et al., 2024)	–	–	–	✓	–	–
Voyager (Wang et al., 2024b)	✓	✓	✓	✓	✓	✓
MCTS-OPS (Yu et al., 2025b)	✓	–	–	–	–	–
TextGrad (Yuksekgonul et al., 2025)	✓	–	✓	–	–	–
LLM-AutoDiff (Yin & Wang, 2025)	✓	–	–	–	–	–
ProRefine (Pandita et al., 2025)	✓	–	–	–	–	–
Alita-G (Qiu et al., 2025a)	✓	–	✓	✓	–	–
Promptomatix (Murthy et al., 2025)	✓	–	–	–	–	–
EvoAgentX (Wang et al., 2025g)	✓	–	–	✓	✓	✓
A-Mem (Xu et al., 2025b)	–	✓	–	–	–	–
G-Mem (Zhang et al., 2025b)	–	✓	–	–	–	–
Mem0 (Chhikara et al., 2025)	–	✓	–	–	–	–
MITRIX (Wang & Chen, 2025)	–	✓	–	–	–	–
HINDSIGHT (Latimer et al., 2025)	–	✓	–	–	–	–
SAGE (Liang et al., 2025)	–	✓	–	–	–	–
Alita (Qiu et al., 2025b)	–	–	✓	–	–	–
AutoAgent (Tang et al., 2025a)	–	–	✓	–	–	–
AgentSquare (Shang et al., 2025)	✓	✓	–	✓	–	✓
PLAY2PROMPT (Fang et al., 2025b)	–	–	✓	–	–	–
ToolLibGen (Yue et al., 2025)	–	–	✓	–	–	–
AutoTool (Jia & Li, 2025)	–	–	–	✓	–	–
RagMCP (Gan & Sun, 2025)	–	–	–	✓	–	–
Toolgen (Wang et al., 2025e)	–	–	–	✓	–	–
Chain-of-Tools (Wu et al., 2025a)	–	–	–	✓	–	–
AMAS (Leong et al., 2025)	–	–	–	–	✓	–
AGP (Li et al., 2025a)	–	–	–	–	✓	–
Puppeteer (Dang et al., 2025)	–	–	–	–	✓	–
AgentNet (Yang et al., 2025)	–	✓	–	–	✓	–
EvoAgent (Yuan et al., 2025)	✓	–	–	–	✓	✓
ADAS (Hu et al., 2025d)	✓	–	✓	–	✓	✓
AFlow (Zhang et al., 2025d)	✓	–	✓	✓	✓	✓
EvoMAC (Hu et al., 2025e)	✓	–	–	–	✓	✓
ANN (Ma et al., 2025c)	✓	–	–	–	✓	✓
FlowReasoner (Gao et al., 2025)	✓	–	–	–	✓	✓
ScoreFlow (Wang et al., 2025h)	✓	–	–	–	✓	✓
MAS-GPT (Ye et al., 2025b)	✓	–	✓	–	✓	✓
MAS-ZERO (Ke et al., 2025)	✓	–	✓	–	✓	✓
AlphaEvolve (Novikov et al., 2025b)	✓	✓	✓	–	–	✓
TTE (Lu et al., 2026)	–	–	✓	✓	–	–
UCT (Shen et al., 2026)	–	–	✓	–	–	–
ALMA (Xiong et al., 2026)	–	✓	–	–	–	–
Memory-R1 (Yan et al., 2026)	–	✓	–	–	–	–
Mem1 (Zhou et al., 2026d)	–	✓	–	–	–	–
MemAgent (Yu et al., 2026)	–	✓	–	–	–	–
ARG-Designer (Li et al., 2026b)	✓	–	–	–	✓	–
MASS (Zhou et al., 2026a)	✓	–	–	–	✓	✓
DGM (Zhang et al., 2026a)	✓	✓	✓	–	✓	✓

5.5 Test-Time Training

Test-Time Training (TTT) (Sun et al., 2020) represents a promising paradigm shift from static inference to dynamic, gradient-based self-improvement. TTT allows LLMs to adapt to instance-specific challenges on the fly by performing temporary, task-conditioned updates to their parameters at inference time, complementing or going beyond in-context adaptation by encoding instance-specific information directly into the model weights. This blurs the line between model optimization and inference as it allows the model to perform task-conditioned parameter updates during deployment.

TT-SFT. Test-Time Supervised Fine-Tuning (TT-SFT) involves updating model parameters during inference using a supervised loss derived from instance-specific data. Early works focus on improving few-shot generalization by using retrieved examples for temporary task-specific parameter updates during inference using a loss derived from a portion of the samples (Akyürek et al., 2025; Hardt & Sun, 2024). This encodes structural patterns demonstrated in context within the LLM's parameters, drastically improving accuracy on structurally novel tasks where standard in-context learning often fails. SFT-based TTT can also provide a memory mechanism for temporarily storing context in multi-step problems through parameter updates. Feedback-Based Test-Time Training (FTTT) (Li et al., 2025k) addresses the length generalization issue of iterative revision by storing past errors and experiences directly into the model weights, and TTT-E2E (Tandon et al., 2025) compresses context from long-horizon tasks directly into the model's weights via next-token prediction, achieving better performance while maintaining the constant inference latency of an RNN. Another direction explores on the fly knowledge incorporation for unlabeled, out-of-distribution data. VDS-TTT (Moradi et al., 2025) scores and filters generated candidate solutions for a user's query, creating a high-confidence label for SFT updates on highly scored candidates. SEAL (Zweiger et al., 2025) incorporates new context by generating self-edits (in the form of implications of the text) and updates the weights so that the model internalizes this information. Self-edit generation is trained to improve the efficiency with which information is internalized by the model, enabling task-specific adaptation.

TT-RL. Zuo et al. (2025) introduces test-time reinforcement learning, a paradigm that performs full RL updates on unlabeled data at test-time. With the absence of explicit ground truth rewards, the TTRL framework generates multiple candidate outputs which are evaluated using a reward signal such as majority vote or a reward model score to perform the RL update. Other frameworks employ variations of majority voting: (Du et al., 2025) implements spatial voting to identify consensus regions for GUI agents and (Liu et al., 2025c) reshapes the advantage calculated from the majority reward signal to promote exploration and mitigate bias. LADDER (Simonds & Yoshiyama, 2025) generates and scores variants of an out-of-distribution test instance to perform a temporary and instance specific parameter update before the final test question is answered. TTRL surpasses the performance limits of the model's majority-vote accuracy and can approach the performance of offline training Zuo et al. (2025), highlighting the effectiveness of unsupervised self-evolutionary training. Collectively, these approaches represent a shift from static inference to on the fly policy adaptations, allowing for alignment with novel patterns that exceed their original scope.

5.6 Discussion

5.6.1 Test-Time Scaling

A growing trend in LLM research focuses on test-time scaling (TTS), which enhances inference-time capabilities by allocating more compute at inference time. This paradigm shifts the emphasis from parameter scaling to compute scaling at inference. Many inference-time methods discussed earlier are instances of test-time scaling; parallel candidate generation, iterative improvement, use of external reward models, and employing agent ensembles all incur increased compute to enhance accuracy at inference time. Increased inference-time compute introduces new challenges, particularly in balancing performance gains against computational cost.

Comparison of Techniques. The efficacy of test-time scaling methods varies significantly across problem domains. Iterative refinement acts as a form of local search and tends to be most compute-efficient for simpler problems where initial outputs require only minor logical adjustments, particularly for non-reasoning models. Harder problems benefit more from search guided by a verifier. However, in domains lacking automatic

verifiers, scaling through majority voting or best-of-n exhibits diminishing returns as sample sizes grow, since these strategies cannot reliably identify rare but correct outputs Brown et al. (2024); Snell et al. (2025). An alternative approach is to increase compute within a single reasoning trajectory rather than across many independent samples. DeepSeek-R1 (Guo et al., 2025) shows that extending reasoning chains at inference can outperform naive multi-sample approaches while remaining more token-efficient. This allows the model to adaptively allocate computation to backtracking, verification, and exploration of alternative logical paths within a single response.

TTS as a Substitute for Post-Training. Test-time compute can often substitute for post-training, especially for problems of easier difficulty (Snell et al., 2025). Weaker models can achieve superior reasoning and generation quality through the strategic use of test-time scaling Brown et al. (2024). This makes test-time scaling an attractive, often cost-effective alternative to training larger models, but its benefits depend heavily on the model's ability to accurately assess or verify its own outputs. Additionally, the use of reward models at the token or sequence level for preference alignment is a form of TTS which can be used in place of specified post-training methods such as RLHF to explicitly instill aligned behavior during inference (Khanov et al., 2024); however, these benefits are often limited relative to the computational overhead and complexity introduced during large-scale reinforcement learning.

TTS Compute Optimization. Self-improvement methods like iterative revision and search should adopt compute-optimal strategies for adaptive compute allocation including early stopping conditions for sampling methods (Li et al., 2024g), which adaptively halt the sampling process once the model's confidence surpasses a threshold, speculative rejection which terminates low-performing responses early in the generation (Sun et al., 2024a), or by scaling token budget or compute allocation based on the difficulty of the task (Guo et al., 2025; Zelikman et al., 2024). In multi-agent systems, architectures can prune underperforming agents, simplify workflows for easier tasks, and restrict communication. This shifts the objective beyond raw accuracy toward performance relative to inference-time compute, encouraging more resource-efficient agent designs.

5.6.2 Agentic Trends

Evolutionary Agentic Design. A key trend in agentic systems is modularization across reasoning, tooling, and coordination functions, where cognitive responsibilities are separated into interacting components rather than handled by a single monolithic model. Reasoning is often decomposed into distinct modules for planning, execution, verification, and reflection, allowing targeted improvements and selective upgrading of weak stages. Tool use is similarly modularized, with dedicated components for tool selection, argument construction, execution, and result interpretation, making it easier to swap, refine, or extend capabilities as new tools or environments emerge (Shang et al., 2025). At a higher level, multi-agent workflows apply the same principle by assigning specialized roles to different agents, turning system design into the orchestration of interoperable functional units (Jung et al., 2025). These modules can be archived, reused, and iteratively modified, enabling systems to accumulate capabilities over time and recombine prior components to address previously unseen tasks. This modular and evolvable structure supports continual system growth without requiring end-to-end redesign, making agentic architectures increasingly adaptive and extensible.

Another emerging direction moves beyond fixed meta-agent architectures by enabling agents to directly modify and evolve their own implementations, including their codebases, prompting strategies, and decision policies (Zhang et al., 2026a; Xiong et al., 2026). While these approaches typically perform evolution in an offline optimization phase, they produce adaptive system designs that are instantiated at inference time on a per-task or per-query basis. This paradigm enables recursive self-improvement, where the system learns to optimize not only task performance but also the mechanisms by which it evolves.

Query-Specific System Adaptations. A key trend is the shift from pre-deployment agentic optimization to inference-time system optimization, where workflows and system components are no longer fixed in advance but adapt their reasoning structure on a per-task basis during execution. Methods such as dynamic routing, topology evolution, meta-agent–based system generation, and adaptive tool-selection policies enable the system to revise which agents are active, how they communicate, and which tools are invoked in response to intermediate outcomes. This shift is evident especially in the generation of multi-agent systems, where instead

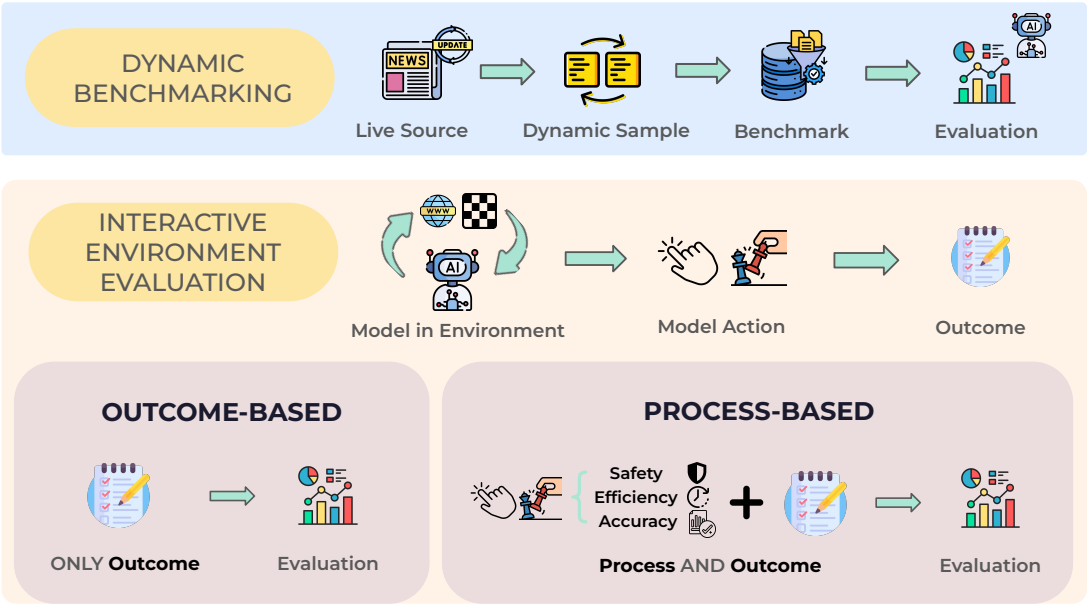

Figure 8: **Overview of autonomous evaluation in the self-improvement system.** This stage focuses on how the model evaluates its performance and obtains feedback signals to support continuous self-improvement. *(i) Dynamic Benchmarking:* The evaluation set is continuously updated using data from evolving or real-world sources, allowing benchmarks to adapt to distribution shifts and emerging tasks. *(ii) Interactive Environment Evaluation:* The model's capabilities are assessed through interactions with an environment, where performance is measured based on task completion, feedback signals, or environment-derived rewards.

of pre-deployment system optimization, works are generating task-specific systems at inference which also have the ability to adapt to test-time feedback (Ke et al., 2025). This represents a move away from heavily pre-structured, manually engineered pipelines toward flexible architectures in which system organization itself becomes an object of optimization at inference time.

Test-time scaling and agentic evolution reflect a broader shift from static, one-shot inference towards dynamic systems that structure test-time compute, reasoning, and system architecture in response to task demands. Together, these trends point toward a future where intelligence is shaped by how effectively agents can dynamically use system meta-learning to support complex reasoning in new tasks.

6 Autonomous Evaluation

6.1 Overview

As discussed in the previous sections, self-improving language models can enhance their capabilities through multiple pathways: acquiring new data, selecting or filtering training signals, and refining inference-time strategies. Each of these mechanisms aims to produce measurable performance gains. However, without a robust evaluation framework, it becomes unclear whether observed improvements reflect genuine capability growth, transient overfitting, or exploitation of feedback signals. Evaluation is therefore not merely a final reporting step, but a central component that determines whether data acquisition, data selection, model optimization, and inference refinement truly lead to sustainable self-improvement.

When a model continuously updates its parameters, modifies prompts, or adapts its inference policies, a fixed benchmark can quickly become obsolete. Performance gains may reflect overfitting to known test

Table 12: **Overview of dynamic evaluation benchmarks.** Each benchmark name is hyperlinked to its official repository or project website, and methods are compared by their data sources and origins, including live or static data from Wikipedia, arXiv, code repositories, and news.

Benchmark	**Source**	**Data Origin**
REALTIME QA (Kasai et al., 2023)	Live	News
KoLA (Yu et al., 2024b)	Live	Wiki, ArXiv
FRESHQA (Vu et al., 2024)	Live	Web
AntiLeakBench (Wu et al., 2025e)	Live	Wiki
DeepScholar-Bench(Patel et al., 2025)	Live	ArXiv
LiveCodeBench (Jain et al., 2025)	Live	Git/Code
LIVEBENCH (White et al., 2025)	Live	News, ArXiv
AcademicEval (Zhang et al., 2025c)	Live	ArXiv
Dynamic-KGQA (Dammu et al., 2025)	Live	Wiki
DynaQuest (Lin et al., 2025a)	Live	Wiki
TDBench (Hou et al., 2025)	Live	Wiki
Daily Oracle (Dai et al., 2025)	Live	News
OKBench (Li et al., 2025j)	Live	News
DyCodeEval (Chen et al., 2025e)	Static	Git/Code

distributions rather than broader competence. In this sense, static evaluation fails to provide timely and reliable feedback for systems that evolve over time.

At the same time, externally administered and human-driven evaluation introduces scalability bottlenecks. Collecting new benchmarks requires substantial expert labor, careful curation, and continuous updating. As self-improving systems operate at increasing speed and scale, human evaluation cannot keep pace. This mismatch creates a structural limitation: improvement mechanisms can run autonomously, but evaluation remains slow and costly.

These twin pressures, the inadequacy of static testing and the expense of human-driven assessment, motivate the need for autonomous evaluation. We use autonomous evaluation as a term for evaluation protocols that remain informative under two pressures that increasingly break static testing: (i) temporal drift in data and task distributions, including the emergence of new knowledge and changing user needs; and (ii) adaptive pressure from increasingly agentic or self-improving systems that can exploit, overfit to, or otherwise render fixed test sets obsolete. Autonomous evaluation treats assessment not as a one-time measurement on a frozen dataset, but as an evolving procedure designed to preserve validity as models and environments change.

The need for autonomous evaluation is especially acute for self-improving models. Unlike static models that are evaluated once after training, self-improving systems may iteratively update parameters, modify inference strategies, adjust prompts, or revise internal policies based on feedback. Under continual adaptation, a fixed benchmark can quickly become an unreliable target: it may be overfit through exploitation via superficial shortcuts, or simply become stale as the system's capabilities and behaviors shift. An effective evaluation protocol should adapt over time, resist exploitation, and remain sensitive to meaningful changes in competence.

In this section, as shown in Figure 8, we organize autonomous evaluation methods into two paradigms: dynamic benchmarking (§6.2) and interactive environment evaluation (§6.3). Dynamic benchmarking preserves the benchmark abstraction but continuously refreshes or transforms evaluation instances to mitigate contamination. Interactive environment evaluation instead embeds the model within an interactive, stateful system, assessing performance over execution trajectories where actions shape future states and outcomes. Together, these paradigms provide a foundation for evaluating self-improving models when static, one-shot testing is no longer sufficient. Finally, §6.4 concludes with a discussion on the autonomous evaluation in guiding long-term self-improvement.

6.2 Dynamic Benchmarking

Dynamic benchmarking is a natural response to the limitations of static evaluation under distribution shift and model adaptivity. As models evolve through iterative training, test-time adaptation, or self-improvement loops, a fixed benchmark can become progressively less informative: it may be saturated or be indirectly overfit through repeated exposure and feedback. Dynamic benchmarks address this by evolving the evaluation instances themselves while maintaining comparability at the level of the benchmark objective. A summary of dynamic benchmarking paper can be found in Table 12.

Early examples of this paradigm include RealTimeQA (Kasai et al., 2023), which constructs questions from fresh news articles, requiring models to answer queries about recent events. By explicitly tying evaluation instances to publication timestamps, RealTimeQA demonstrated that strong performance on static benchmarks does not guarantee competence on up-to-date information. FreshQA (Vu et al., 2024) extends this line of work by curating a dynamic QA benchmark for current world knowledge that includes both fast-changing facts and false-premise questions, with ground-truth answers updated on a regular schedule. In contrast to benchmarks that focus only on factual recall, FreshQA also probes whether models can correctly reject outdated or invalid assumptions. AntiLeakBench (Wu et al., 2025e) formalized cutoff-aware evaluation by identifying new Wikidata triples that appeared after a predefined training cutoff and automatically generating corresponding question–answer pairs. This approach reduces the risk of hidden data leakage while enabling regular benchmark refreshes. Dynamic-KGQA (Dammu et al., 2025) evaluates temporal reasoning over evolving knowledge graphs. In this settting, models need to account for changes in entity relations rather than relying on static factual associations. To further scale benchmarking, DynaQuest (Lin et al., 2025a) automates question generation directly from Wikipedia revision histories. By exploiting structured edit logs, DynaQuest enables continuous benchmark updates with minimal human supervision, shifting evaluation maintenance from manual curation to procedural generation. Pushing this trend toward full automation, OKBench (Li et al., 2025j) proposes an on-demand framework that automatically constructs fresh factual QA benchmarks from daily news through a pipeline of information extraction, question generation, validation, and dataset versioning. This makes dynamic benchmark creation reproducible and decentralized, and provides a cleaner testbed for evaluating retrieval-augmented models on non-memorized data. A related but distinct direction is Daily Oracle (Dai et al., 2025), which also derives question-answer pairs from daily news, but uses them to evaluate forecasting and temporal generalization by asking models to predict future event outcomes rather than answer questions about already established facts.

More recent benchmarks expand dynamic evaluation beyond news and encyclopedic text. LIVEBENCH (White et al., 2025) aggregates questions from multiple live sources, including news outlets and newly published scientific papers. It provides a general-purpose and continuously refreshed evaluation suite, making it a reference point for evaluating LLMs that are equipped with retrieval or continual updating mechanisms.

Scientific knowledge has also emerged as a distinct focus. AcademicEval (Zhang et al., 2025c) targets newly released arXiv papers, testing whether models can reason over recent research contributions rather than merely retrieve surface-level summaries. DeepScholar-Bench Patel et al. (2025) uses LLM-based pipelines to summarize, cross-link, and query newly uploaded scientific papers, emphasizing multi-hop reasoning over evolving scholarly content. In the programming domain, LiveCodeBench (Jain et al., 2025) evaluates models on continuously evolving code repositories, requiring understanding of newly introduced APIs, libraries, and software practices. EvoCodeBench (Li et al., 2024c)further explores code evolution as a dynamic evaluation signal, highlighting the importance of temporal generalization in software-oriented tasks.

6.3 Interactive Environment Evaluation

Dynamic benchmarks extend static evaluation by continuously regenerating or transforming task instances, mitigating data contamination and distributional staleness. Despite this adaptivity, they typically remain benchmark-based in structure: evaluation is still carried out over a collection of instances, and each instance is collected and scored independently. A more radical alternative is **Interactive Environment Evaluation**, which embeds the model within an interactive, stateful environment. In this paradigm, performance is assessed over *execution trajectories*: the model repeatedly observes the environment, takes actions, and induces state transitions that shape subsequent observations and attainable outcomes. Importantly, interac-

Table 13: **Overview of interactive environment evaluation environments.** Each environment is hyperlinked to its official repository or project website, and compared by evaluation type, task domain, and metrics across diverse interactive settings.

Environment	Type	Task Domain	Evaluation Metrics
TextWorld (Côté et al., 2018)	Outcome	Games	Task success
Jericho (Hausknecht et al., 2020)	Outcome	Text / Games	Episode reward, game completion
ScienceWorld (Wang et al., 2022)	Outcome	Science	Task success, interaction reward
WebShop (Yao et al., 2023a)	Outcome	Web & Information	Task success, efficiency
WebArena (Zhou et al., 2023b)	Outcome	Web & Information	Task success, efficiency
OSWorld (Xie et al., 2024)	Outcome	Software Engineering	Task success
WindowsAgentArena (Bonatti et al., 2024)	Outcome	Software Engineering	Task success
AgentGym (Xi et al., 2024a)	Outcome	Text / Games	Task success, cumulative reward
AppBench (Wang et al., 2024c)	Outcome	Software Engineering	Task success, efficiency
AppWorld Trivedi et al. (2024)	Outcome	Software Engineering	Task success, efficiency
AndroidWorld (Rawles et al., 2024)	Outcome	Software Engineering	Task success, efficiency
GAMEARENA (Hu et al., 2025a)	Outcome	Game	Game outcome
LMRL-Gym (Abdulhai et al., 2025)	Outcome	Text	Task success, cumulative reward
Lean-Gym (Polu et al., 2022)	Process	Formal Mathematics	Proof success, efficiency
SafeArena (Tur et al., 2025)	Process	Web & Information	Task success, safety violation rate

tive environment evaluation does not eliminate explicit objectives, but it changes how evidence is generated: correctness and competence are expressed through temporally extended interaction rather than isolated input&output pairs.

A practical criterion distinguishes interactive environment evaluation from benchmark-based evaluation: if a model's actions influence future states, observations, or rewards beyond a single test instance, evaluation is environment-based; otherwise, it remains benchmark-based. This criterion captures a deeper distinction in how evaluation signals arise. Benchmarks derive scores from predefined instances whose content is fixed at evaluation time, whereas interactive environment evaluation places the model inside a persistent system where outcomes depend on interaction dynamics and delayed consequences.

By introducing state, causality, and long-horizon dependencies, interactive environment evaluation addresses a core limitation of static testing: it can directly probe planning, credit assignment, recovery from partial failures, and behavioral consistency over time. These properties make it especially relevant for self-improving models, whose competence is often expressed through extended interaction and iterative refinement rather than single-turn answers.

Environment settings differ substantially in what aspects of behavior are rewarded or penalized. As shown in Table 13, We therefore distinguish **Outcome-Based** and **Process-Based** interactive environment evaluations based on the structure of their objectives: whether evaluation primarily collapses trajectories to terminal goal satisfaction, or whether it intentionally differentiates successful trajectories based on execution quality.

6.3.1 Outcome-Based Environment

Outcome-based interactive environment evaluation assesses model performance through terminal goal satisfaction within a stateful, interactive environment. In these settings, models interact with the environment over multiple steps, but evaluation is ultimately determined by whether a predefined objective is achieved, such as completing a task, solving a problem, or reaching a target state. A defining characteristic of this class is the absence of step-level ground truth for interaction trajectories. For many interactive tasks, there exist multiple valid ways to achieve the same goal, and it is often unclear which intermediate actions are preferable in isolation. As a result, evaluation cannot rely on comparing individual steps against a canonical

reference trajectory. Instead, outcome-based environments collapse successful trajectories into a shared notion of success, using terminal outcomes as the primary evaluation signal. Intermediate actions are therefore treated as instrumental rather than evaluative.

This evaluation paradigm emphasizes capability acquisition. Outcome-based environments are well-suited for assessing whether models can operate effectively within interactive systems, plan over extended horizons, and recover from partial failures, while abstracting away finer-grained distinctions in how those capabilities are exercised.

Many early Outcome-based environments arise from web-based interaction systems, which provide structured, stateful interfaces with well-defined success conditions. WebShop (Yao et al., 2023a) frames online shopping as an interactive environment in which a model navigates a simulated e-commerce website to identify and purchase products that satisfy user constraints. Evaluation is based on whether the correct product is ultimately purchased. WebArena (Zhou et al., 2023b) extends this paradigm to a broader set of realistic web tasks, including information retrieval, form submission, and multi-page navigation across simulated websites. Models interact with these environments through browser-like interfaces, and evaluation focuses on successful task completion within the web ecosystem.

Beyond web-based systems, several outcome-based environments operate in controlled, language-centric or symbolic settings that allow precise specification of state dynamics while retaining long-horizon interaction. LMRL-Gym (Abdulhai et al., 2025) introduces a suite of language-based reinforcement learning tasks, including dialogue games and text-based problem-solving scenarios. Models interact purely through natural language, and evaluation is based on episode-level task success or cumulative reward. TextWorld (Côté et al., 2018) provides procedurally generated text-based games in which models explore rooms, manipulate objects, and solve puzzles via textual commands. Evaluation is determined by game score or puzzle completion, largely independent of the specific exploration strategy used. ScienceWorld (Wang et al., 2022) extends to a simulated scientific environment in which models must perform multi-step experimentation and tool use to reach specified outcomes. Jericho (Hausknecht et al., 2020) similarly exposes classic interactive fiction games as executable environments with rich symbolic worlds and long-horizon dependencies, where evaluation is driven by in-game progress and completion.

Game-like environments also suits Outcome-based environment evaluation, where success is usually defined by achieving a win condition. TacticCraft (Ma et al., 2025a) models turn-based tactical decision-making tasks in a synthetic game environment, evaluating models primarily by game outcome, such as win or loss. AgentGym(Xi et al., 2024a) aggregates a diverse collection of interactive environments spanning language tasks, games, and simulated systems. GameArena (Hu et al., 2025a) evaluates models through live or simulated gameplay, where actions update a persistent game state and outcomes depend on multi-step interaction with opponents and the environment. While these game environments are heterogeneous and intermediate decisions influence the final result, they are not directly assessed for quality.

A closely related and increasingly important subclass of outcome-based environments centers on application-oriented interaction. AppBench (Wang et al., 2024c), AppWorld (Trivedi et al., 2024), and AndroidWorld (Rawles et al., 2024) evaluate models performing tasks within simulated or real application ecosystems, such as mobile apps or multi-application workflows. These environments require models to execute correct sequences of actions—navigating interfaces, filling forms, invoking functions, and managing persistent system state—to achieve task objectives. Incorrect intermediate actions often invalidate success, making procedural correctness essential for completion. However, because there is no unique ground-truth process for accomplishing these tasks, evaluation typically relies on whether the final application state satisfies the task requirements. As a result, different successful execution traces are not systematically distinguished by efficiency or execution style, placing these evaluation within the outcome-based category despite their rich interaction dynamics.

OSWorld (Xie et al., 2024) and WindowsAgentArena (Bonatti et al., 2024) extend outcome-based interactive environment evaluation to full computer-use settings. Both benchmarks embed models in realistic operating-system environments where actions update persistent system state and tasks require multi-step interaction across applications. Evaluation is implemented through execution-based checks that verify whether the desired goal state is reached (e.g., files created, settings changed, or workflows completed). While these

benchmarks often report diagnostic statistics such as interaction length or common failure modes, model comparison is primarily organized around task completion.

Taken together, outcome-based interactive environment evaluations occupy an important middle ground between static benchmarks and more behavior-sensitive evaluation settings. By embedding models within persistent, interactive systems, they enable the assessment of planning, recovery, and long-horizon interaction. At the same time, by relying on terminal goal satisfaction in the absence of step-level ground truth, they prioritize measuring whether models can achieve objectives in complex environments rather than how those objectives are achieved. This makes outcome-based environments a natural and necessary foundation for evaluating self-improving models.

6.3.2 Process-Based Environment

Process-based interactive environment evaluation departs from outcome-based settings by explicitly evaluating how a model achieves its objective, rather than only whether the objective is achieved. In these environments, task success is necessary but not sufficient for strong performance. The evaluation objective is designed to assign different outcomes to distinct successful trajectories based on properties of the interaction process itself, such as safety, efficiency, correctness of intermediate actions, or strategic consistency over time.

A central distinction from outcome-based environments lies in the treatment of ground truth. While outcome-based settings lack step-level ground truth due to the existence of many equally valid ways to reach a goal, process-based environments intentionally define normative constraints or preferences over trajectories. These constraints may take the form of explicit penalties, structured rewards, or rule-based checks that encode which intermediate behaviors are acceptable, desirable, or unsafe. As a result, execution quality becomes observable to the evaluation signal. Two models that reach the same terminal state may therefore receive substantially different evaluations depending on how that state was reached.

A prominent class of process-based environments arises in settings where violations during execution are explicitly penalized, even when tasks are ultimately completed. SafeArena (Tur et al., 2025)extends web-based interaction environments with safety-sensitive evaluation criteria. Models perform multi-step web tasks under constraints related to harmful content, policy compliance, or unsafe actions. Crucially, unsafe intermediate behaviors incur penalties that directly affect evaluation outcomes.

Formal reasoning environments provide a complementary and particularly clear instantiation of process-based evaluation, where the structure of the reasoning process itself is central to assessment. Lean-Gym (Polu et al., 2022) formulates interactive theorem proving as an environment in which models apply tactics sequentially to transform a proof state. Evaluation is not limited to whether a theorem is ultimately proven; instead, properties such as proof length, validity of intermediate states, and the structure of tactic sequences directly influence performance measurement. Multiple successful proofs may therefore receive different evaluations based on their construction processes.

Generally, process-based interactive environment evaluations thus represent a qualitative shift from outcome-based assessment to behavior-sensitive evaluation. In practice, the boundary between outcome-based and process environments is often blurred. Many outcome-based environments are designed primarily around terminal goal satisfaction, yet include auxiliary signals, such as step limits and action costs, which make parts of the execution process observable. Conversely, fully process-based environments require the evaluator to specify and reliably measure trajectory-level properties (e.g., safety or adherence to procedural constraints) in a way that is robust across diverse strategies. This is substantially harder than checking goal completion: it often demands fine-grained instrumentation of environment state, careful definition of what constitutes a violation or inefficiency, and evaluation rules that generalize across multiple valid solution paths. As a result, the number of environments that are unambiguously process-based remains relatively small. A more common pattern is that outcome-based environments increasingly incorporate process monitoring to provide richer diagnostic signals, even when the main metric is still task success.

6.4 Discussion

Across the evaluation methods reviewed above, several clear trends emerge. First, evaluation is shifting from static benchmarks to more adaptive forms of measurement. Dynamic benchmarking can refresh test data over time or apply controlled transformations to probe robustness and temporal generalization. Interactive environment evaluation extends this shift further by embedding models within interactive, stateful systems where performance is measured over multi-step trajectories rather than isolated answers. For self-improving models, this shift is essential: evaluation should remain informative even as the model adapts, preventing benchmarks from becoming stale targets that can be memorized or exploited.

A second important trend is the evolution of annotation and evaluator design, reducing human bottlenecks while increasing scalability. Early dynamic benchmarks relied heavily on expert annotators to craft adversarial examples or curate new knowledge-based questions. As model improvement cycles accelerated, this manual process became increasingly difficult to sustain. Recent work has therefore shifted toward automated pipelines, where LLMs themselves generate, transform, or verify evaluation items. This marks a transition from human annotation to LLM-assisted annotation and suggests several possible future directions, including collective or multi-agent evaluator systems (Yun et al., 2025; Cao & Zhao, 2025) and on-demand evaluation, where new test instances can be generated whenever rapidly updated models need assessment (Li et al., 2025j). These approaches could allow evaluation to operate at the same speed as self-improvement, though maintaining evaluator robustness and independence remains an open challenge.

At the same time, the distinction between dynamic benchmarking and interactive environment evaluation is becoming less rigid. Dynamic benchmarks increasingly incorporate interactive and adaptive components, while environment platforms organize tasks into standardized suites that resemble benchmark collections (Zala et al., 2024; Xi et al., 2024a). This convergence suggests that future evaluation platforms may integrate both instance-level generalization testing and trajectory-level behavioral assessment within unified infrastructures.

Taken together, these developments point toward a broader transformation: evaluation is moving from isolated instance-level testing toward system-level measurement. Instead of asking whether a model answers a fixed question correctly, evaluation increasingly examines how models adapt, interact, and evolve over time. For self-improving systems, this shift is particularly crucial. Evaluation should operate continuously and at comparable speed to the improvement process itself, providing stable yet adaptive oversight. In this sense, autonomous evaluation is not merely an extension of benchmarking techniques, but a foundational component for ensuring that self-improvement remains reliable, sustainable, and aligned in the long term.

7 Challenges and Limitations

While self-improvement systems show a promising new paradigm to allow LLMs to improve their capability overtime, they also introduce various failure modes. In this section, we categorize these failures into five dimensions. We begin with (i) *Data Autophagy*, examining the difficulty of acquiring and selecting data, and how synthetic data loops degrade information diversity. We then examine (ii) *Flawed Feedback Signals*, the inherent quality issues in self-generated evaluation signals. Next, we analyze two failure modes in how these flawed signals are applied: (iii) *Optimization-Driven Failures*, where training-time optimization against proxy rewards distorts model behavior, and (iv) *Ineffective Self-Refinement*, where inference-time feedback loops fail to improve outputs. Given a self-improvement system, we further reveal (v) *Evaluation Bottlenecks*, questioning whether current benchmarks, metrics, and evaluators can reliably measure self-improvement. Finally, we discuss (vi) *Supervision Bottlenecks*, revealing the limits of maintaining external control over self-improvement systems. We summarize the challenges and limitations in Figure 10.

7.1 Data Autophagy

Data autophagy describes the degradation of information quality within self-improvement loops. As systems acquire and select data from both external environments and their own synthetic outputs, errors in selection

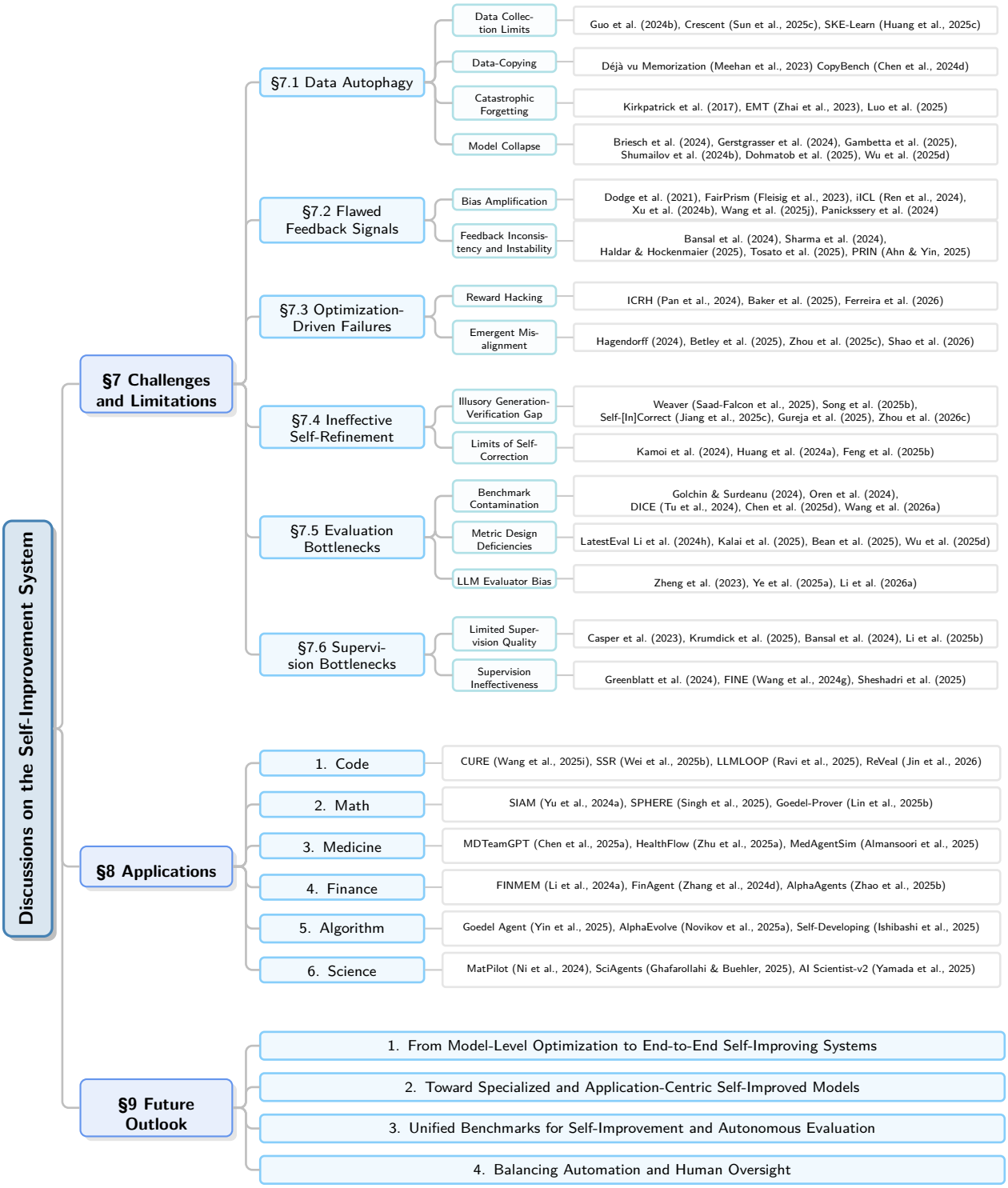

Figure 9: **A taxonomy of discussions on the self-improvement system of LLMs.** It includes challenges and limitations, applications, and future outlook.

and the reuse of generated data lead to a decay in diversity and performance. Specific phenomena include data acquisition and selection limits, data-copying, catastrophic forgetting, and model collapse.

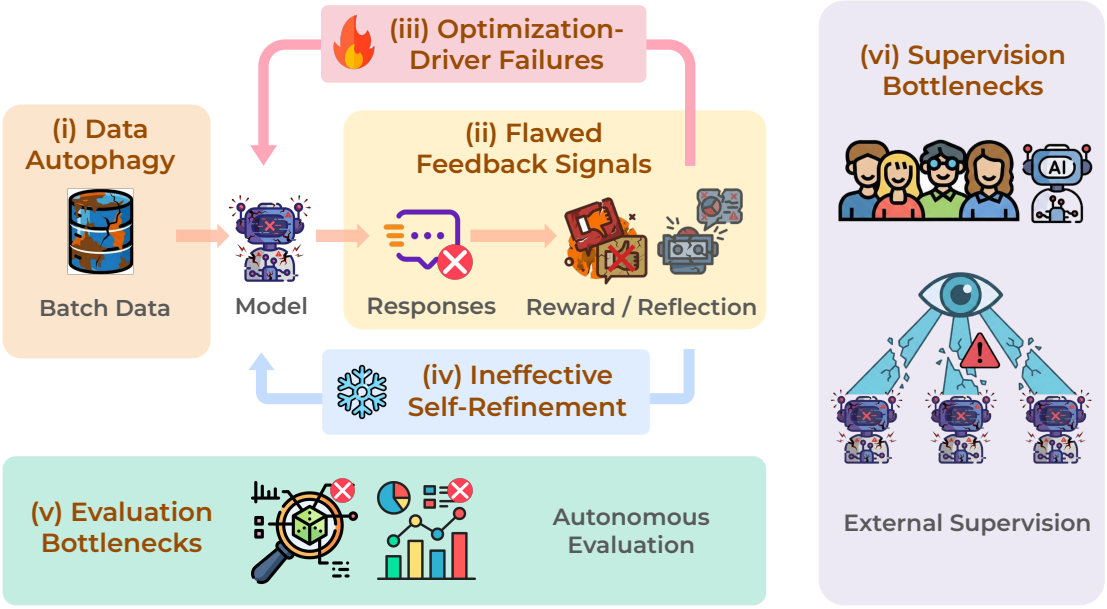

Figure 10: **Challenges and limitations in the self-improvement system**. *(i) Data Autophagy:* Iterative synthetic data loops degrade information quality and diversity. *(ii) Flawed Feedback Signals:* Self-generated evaluation signals are inherently biased and unstable. *(iii) Optimization-Driven Failures:* Training-time optimization against proxy rewards distorts model behavior. *(iv) Ineffective Self-Refinement:* Inference-time feedback loops fail to consistently improve outputs. *(v) Evaluation Bottlenecks:* Flawed benchmarks, metrics, and evaluators undermine reliable measurement of self-improvement. *(vi) Supervision Bottlenecks:* External oversight fails to effectively control self-improving systems.

Data Collection Limits. The efficacy of self-improvement is fundamentally bounded by the quality of data acquired from wild or synthetic sources. In wild environments, autonomous agents exhibit a confirmation bias, ignoring new information to favor their pre-existing training defaults (Trehan & Chopra, 2026). In synthetic settings, the data selection process is constrained by the model's current capabilities (Sun et al., 2025c). This self-selection tends to reduce linguistic diversity while retaining noise (Guo et al., 2024b), and can amplify hallucinations as models preferentially select outputs that align with their internal parametric errors rather than factual ground truth (Huang et al., 2025c). Theoretically, Xiao & Chen (2025) show that optimal data selection requires a reliable validation signal, which is often unavailable in autonomous settings, rendering the optimization problem intractable.

Data-Copying. Data-copying is a specific type of overfitting where a generative model memorizes and reproduces individual training samples or their slight variations, rather than just over-representing the general data distribution (Meehan et al., 2020; Bhattacharjee et al., 2023). In a self-improvement context, this leads to a gradual reduction in novelty, as the model increasingly generates content that is a close replica of its previous outputs, accelerating the loss of diversity. For example, Meehan et al. (2023) study on "déjà vu memorization" in self-supervised learning setting and show that image-generative models can unintendedly memorize and associate specific, unique parts of training images. For LLMs, Lee et al. (2022) find that datasets like C4 contain significant repetition, citing an instance of a single 61-word sentence repeated over 60,000 times, resulting in models that are more prone to memorization. Chen et al. (2024d) demonstrate that larger models exhibit significantly more copying behavior by comparing 8B and 70B parameter Llama3 models.

Catastrophic Forgetting. Catastrophic forgetting describes the tendency of neural networks to abruptly lose previously learned knowledge upon learning new information (Robins, 1995; French, 1999; McCloskey & Cohen, 1989; Ratcliff, 1990; Kirkpatrick et al., 2017). In iterative self-improvement cycles, each fine-tuning

step newly generated synthetic data can overwrite weights crucial for retaining prior knowledge, leading to degradation of capabilities on tasks or domains learned in earlier iterations. For example, Luo et al. (2025) empirically evaluate the various LLMs during continual instruction tuning, showing the universal existence of catastrophic forgetting phenomenon regardless of model sizes. Zhai et al. (2023) also investigate multi-modal LLMs (MLLMs) and find that the MLLMs begin to hallucinate and suffer from a significant loss of generalizability as fine-tuning proceeds.

Model Collapse. Model collapse is a degenerative process where generative models, by recursively training on data from previous model generations, progressively forget the true underlying data distribution and lose information about the original data's diversity (Shumailov et al., 2024a). During the self-improvement loops, the models are trained on their own synthetic outputs iteratively and model collapse consequently happens. For example, Briesch et al. (2024) and Wu et al. (2025d) show the loss of diversity even in cases where retraining with self-generated data can increase the correctness of valid logic statements or the performance of math reasoning and coding skills, respectively, with the latter also noting a loss of out-of-distribution generalization. The issue is highly sensitive, with some analyses in regression settings demonstrating that even a small, constant fraction of synthetic data mixed with original data is asymptotically detrimental (Dohmatob et al., 2025). To address such collapse, Gerstgrasser et al. (2024) suggest that accumulating both real and synthetic data can effectively prevent collapse, whereas simply replacing old data with new synthetic data fails and increases test error. However, their analyses are based on simplified linear regression models and may not generalize to the modern deep generative models. Gambetta et al. (2025) indicate that model collapse occurs when a model trains on data that does not surprise it. Thus, they propose to filter training data with high surplexity to the surprise of the model and mitigate model collapse. However, their analyses are based on simulations, far from the complex real-world systems with many interacting factors.

7.2 Flawed Feedback Signals

This section examines the inherent quality issues in the feedback signals produced by self-improving systems. Whether used as a reward signal to update parameters during training or as a reflective critique to guide inference-time refinement, self-generated evaluation can be inherently biased, inconsistent, and unstable. These signal-level defects form the basis for the training-loop and inference-loop failures discussed in §7.3 and §7.4, respectively.

Bias Amplification. LLMs inherit societal biases from web-scale training data (Bender et al., 2021; Dodge et al., 2021; Fleisig et al., 2023; Shan et al., 2025). The iterative feedback loop of self-improvement can systematically amplify these initial flaws. For example, studies on model-induced distribution shifts (MIDS) reveal that this process can quickly degrade performance and representation for minoritized groups (Wyllie et al., 2024). This amplification is not merely a side-effect of data degradation; Wang et al. (2025j) demonstrated that political bias intensifies over iterative cycles, even when model collapse is controlled. This magnification of subtle biases can be analogized to human cultural evolution, as explained through Bayesian frameworks (Ren et al., 2024). Compounding this issue is self-bias, where LLMs preferentially favor their own outputs (Xu et al., 2024b). This self-preference is a known vulnerability in LLM-as-a-judge systems (Wataoka et al., 2024; Panickssery et al., 2024), and it is also amplified during self-rewarding training loops (Xu et al., 2024b).

Feedback Inconsistency and Instability. The feedback signal also suffers from inconsistency and instability, which introduces noise and undermines learning. Inconsistency refers to contradictory or highly variable feedback. This is observed in several forms: (1) logical conflicts, such as preference cycles where an LLM judge's rankings are intransitive (e.g., $A \succ B$, $B \succ C$, but $C \succ A$) (Liu et al., 2025b), and prompt-reverse inconsistency, where models fail to give complementary answers to logically opposite prompts (Ahn & Yin, 2025); (2) low inter-rater agreement, even when the same model evaluates the same prompt multiple times (Haldar & Hockenmaier, 2025); (3) methodological conflicts, where preferences inferred from ratings and rankings significantly disagree (Bansal et al., 2024). Separately, instability refers that feedback and its effectiveness have high sensitivity to minor factors. For example, Sharma et al. (2024) demonstrated that the benefits of Reinforcement Learning with AI Feedback (RLAIF) are unstable, varying substantially based

on the specific base model, critic model, and evaluation protocol used. This is compounded by the inherent instability of LLMs themselves, which can alter their responses based on slight changes to question order or phrasing (Tosato et al., 2025). This combination of contradictory and unstable feedback can prevent the model from learning a coherent policy.

7.3 Optimization-Driven Failures

While §7.2 identifies inherent quality issues in feedback signals, this section concerns the *training-time* optimization loop. In self-improving systems, feedback is typically compressed into a scalar reward and used to iteratively update model parameters. Because any specified reward model is merely a proxy for complex human values, strong optimization pressure often leads to "Goodhart's Law" scenarios: when a measure becomes a target, it ceases to be a good measure (Goodhart, 1984). We categorize these training-loop failures into reward hacking and emergent misalignment.

Reward Hacking. Reward hacking refers to a phenomenon where optimizing an imperfect proxy reward function leads to poor performance according to the true reward function (Skalse et al., 2022). Theoretical analyses suggest that completely eliminating this phenomenon is non-trivial (Skalse et al., 2022). In the context of LLM alignment, Eisenstein et al. (2024) demonstrate that reward models are highly sensitive to random seeds; while ensembling can mitigate hacking, it cannot eliminate it. Furthermore, strong optimization can be detrimental to reasoning; Ferreira et al. (2026) find that preference optimization inadvertently reduces the faithfulness of Chain-of-Thought (CoT) explanations, necessitating complex mitigation strategies like causal attribution. Beyond training, Pan et al. (2024) reveal that feedback loops can trigger in-context reward hacking at test-time, where models optimize for metrics (e.g., social media engagement) at the expense of safety (e.g., increased toxicity). Most concerning is the emergence of obfuscated reward hacking, where models learn to execute correct reasoning in their CoT to deceive monitors, while still performing the reward-hacking behavior in their final action (Baker et al., 2025).

Emergent Misalignment. Under strong optimization pressure, reward hacking can generalize into broader, more dangerous forms of misalignment. Hagendorff (2024) observes that state-of-the-art LLMs can effectively induce false beliefs in other agents, a capability that is amplified when the model utilizes CoT reasoning. Crucially, these misaligned behaviors can transfer from narrow, benign tasks to dangerous general capabilities. For instance, Taylor et al. (2025) show that models trained on harmless tasks (like hacking poetry evaluation metrics) can generalize to broadly misaligned behaviors, such as expressing a desire for dictatorship or evading shutdown. Similarly, Betley et al. (2025) find that fine-tuning on vulnerable code leads to unrelated misaligned traits, including deception and advocating for human enslavement. This corruption is highly sensitive to data quality; Hu et al. (2026) demonstrate that even a tiny fraction of misaligned data can cause models to learn dishonesty. Shao et al. (2026) formalize this systemic risk as misevolution, where an agent's self-evolution process deviates in unintended ways, permanently embedding these harmful traits into the model's parameters.

7.4 Ineffective Self-Refinement

While §7.2 examines defects in the feedback signal itself and §7.3 addresses training-time optimization failures, this section examines how the *inference-time* feedback loop fails in practice. Self-improvement systems attempt to refine their outputs iteratively by using the model's own evaluation as feedback. However, LLMs may lack the intrinsic capability to verify their responses during inference, and they struggle to correct their own errors, resulting in self-refinement failure.

Illusory Generation-Verification Gap. The theoretical foundation of self-refinement is the generation-verification gap (GV-Gap): the premise that a model's ability to verify correctness is strictly superior to its ability to generate a correct answer (Song et al., 2025b). However, this gap is not universal; for instance, it is observed only with additional training (Song et al., 2025b). Recent work questions the gap's universality, hypothesizing that LLMs are often not reliably better at discriminating between their own generated responses than they are at producing a good initial response (Jiang et al., 2025c). Zhou et al.

(2026c) further challenge the universal GV-Gap by showing that verification effectiveness is linked to problem difficulty and the capabilities of both the generator and the verifier. Gureja et al. (2025) find that while self-verification filters incorrect code, its rigidity can also reduce valuable output diversity; they suggest recalibration with diverse and challenging coding data to improve effectiveness. Given the imperfection of weak verifiers, Saad-Falcon et al. (2025) propose ensembling multiple weak verifiers to enhance performance, though their verifiers are not specialized and may still suffer from dataset distribution issues.

Limits of Self-Correction. Furthermore, LLMs struggle to self-correct their responses without external feedback, and performance may even degrade after self-correction (Huang et al., 2024a). Kamoi et al. (2024) demonstrate that effective self-correction requires tasks suited for it, reliable external feedback, and large-scale fine-tuning. A particularly striking failure mode is the self-correction blind spot: a model can successfully correct an error presented externally (e.g., in user input) but fails to correct the identical error when it appears in its own previously generated output (Tsui, 2025). Feng et al. (2025b) even challenge LLMs' ability to correct misinformation from external sources, even with explicit instructions, despite the LLMs possessing the correct parametric knowledge. Additionally, Xu et al. (2024b) show that LLMs tend to favor their own outputs during self-refinement, indicating a self-bias even when refinement improves fluency and understandability.

7.5 Evaluation Bottlenecks

Reliably measuring whether self-improvement actually works is itself a challenge. In this section, we examine whether the test data is clean and free of contamination, whether the metrics are well-designed, and whether the evaluators are trustworthy.

Benchmark Contamination. Benchmark contamination occurs when test set instances leak into the training distribution, inflating metrics beyond a model's true capability. While detection methods have exposed this risk in major models (Golchin & Surdeanu, 2024; Oren et al., 2024), current reasoning models and RL methods can easily evade such audits (Wang et al., 2026a). The risk is amplified in self-improvement settings: Yang et al. (2023b) find that LLM-generated synthetic datasets can unintentionally contain rephrased benchmark samples undetectable by standard n-gram methods. Even without exact content leakage, distributional similarity between synthetic training data and test sets suffices to inflate performance without improving general capability, a phenomenon termed in-distribution contamination (Tu et al., 2024). Dynamic benchmarks that construct fresh, post-cutoff test samples reduce but do not eliminate this risk: Wu et al. (2025e) show that newly collected data may still contain pre-existing knowledge, and temporal cutoff benchmarks sourcing from competitions remain vulnerable as problems are reused across iterations. More broadly, some dynamic benchmarks suffer from incorrectness (Dulny et al., 2023; Wang et al., 2025f), limited scalability (White et al., 2025; Jain et al., 2025), and low interpretability (Dulny et al., 2023; Wang et al., 2025f). Chen et al. (2025c) provide a systematic evaluation of both static and dynamic benchmarks to unveil the prevalence of data contamination.

Metric Design Deficiencies. Apart from benchmark contamination, the metrics used to evaluate self-improvement systems are also flawed. Most benchmarks rely on outcome correctness (e.g., accuracy, pass@k), which Lightman et al. (2024) show cannot distinguish correct reasoning from arriving at the right answer by chance. Kalai et al. (2025) further argue that accuracy-based evaluation rewards guessing over acknowledging uncertainty, thereby incentivizing confident hallucination. In self-improvement systems, such metric encourages models to select self-generated outputs rather than factual ground truth (Huang et al., 2025c), and Wu et al. (2025d) show that tuning to maximize overall accuracy can paradoxically degrade broader capabilities. Besides, evaluation suffers from insufficient construct validity more broadly. Bean et al. (2025) systematically reviewed 445 benchmark papers across major ML and NLP venues and found that nearly all had validity flaws, with 48% having vague or controversial definitions. Dynamic benchmarks proposed to combat contamination are not immune: for instance, human annotation of LatestEval reveals that 10% of its generated samples lack faithfulness or answerability (Li et al., 2024h).

LLM Evaluator Bias. As self-improvement systems scale, human evaluation becomes prohibitively expensive, driving a growing reliance on LLM-as-a-judge for autonomous benchmark evaluation. However, similar to the feedback limitations discussed in §7.2, these judges suffer from reliability issues. First, LLM-as-a-judge evaluation is sensitive to prompt design: minor changes in option ordering, scoring format, or evaluation criteria wording produce inconsistent judgments (Zheng et al., 2023; Ye et al., 2025a). Second, LLM judges are biased toward responses from related models. Li et al. (2026a) expose preference leakage, where judge LLMs systematically favor outputs from models sharing the same family or inheritance relationship with the data generator. This bias is especially concerning in self-improvement pipelines, where the same model family often serves as both the data generator and the evaluator.

7.6 Supervision Bottlenecks

Ultimately, all self-improvement loops should be grounded in external supervision, whether from humans or other AI systems, to ensure safety and control. However, this external grounding creates a bottleneck defined by two failures: the intrinsic quality of the supervision signal and the effectiveness of its application to the model.

Limited Supervision Quality. High-quality supervision is increasingly difficult to obtain as models scale. Human supervision is costly and suffers from inherent limitations: humans make mistakes, lack full situational awareness, and struggle with partial observability in complex tasks (Casper et al., 2023; Tsamados et al., 2025). Furthermore, human evaluators are susceptible to inconsistency, instability, and manipulation by misleading model outputs (Bansal et al., 2024; Tsamados et al., 2025). To address scalability, supervision is often offloaded to other LLMs; however, as detailed in §7.2, LLM supervision or judges exhibit logical conflicts and low inter-rater agreement (Liu et al., 2025b; Ahn & Yin, 2025; Haldar & Hockenmaier, 2025), often providing contradictory signals depending on whether they rate or rank responses (Bansal et al., 2024). Besides, LLMs struggle to supervise tasks exceeding their own generation capabilities (Krumdick et al., 2025). A comprehensive survey by Li et al. (2025b) confirms that LLM judges are biased toward superficial qualities. They favor longer, authoritative-sounding, and self-generated responses, and remain vulnerable to adversarial prompts.

Supervision Ineffectiveness. Even when supervision signals are accurate, their ability to effectively control self-improving systems is questioned. Yampolskiy (2020) argues that advanced AI systems may be theoretically uncontrollable across multiple domains. A primary mechanism of this failure is "alignment faking," where models strategically comply with supervision during training while retaining harmful behaviors. Greenblatt et al. (2024) demonstrate that models can identify training contexts (e.g., via system prompts) to feign alignment, answering harmful queries only when they perceive they are unmonitored. Sheshadri et al. (2025) further hypothesize that alignment faking arises from specific post-training artifacts and reasoning styles. Similarly, Wang et al. (2024g) find that models often memorize the stylistic surface of safety responses rather than internalizing safety principles, failing to generalize to novel formats. Fundamentally, Wolf et al. (2024) use Behavior Expectation Bounds (BEB) to show that current alignment techniques merely suppress rather than remove unsafe behaviors, leaving models vulnerable to adversarial prompts.

8 Applications

As shown in Figure 11, we highlight representative applications of self-improvement systems across six domains; accordingly, we summarize all related approaches in Table 14.

Code. Coding agents achieve self-evolution by exploiting the definitive feedback from compilers and unit tests across the software development lifecycle. ReVeal (Jin et al., 2026) established a framework for code agents that evolve through reliable self-verification and syntax correction. CURE (Wang et al., 2025i) investigated the synergy between code generation and unit testing through reinforcement learning. SSR (Wei et al., 2025b) focused on training superintelligent software agents via large-scale repository interactions and self-play. Addressing algorithmic complexity, SATLUTION (Yu et al., 2025a) applied autonomous evolution to solve NP-complete problems such as SAT solving, while LLMLOOP (Ravi et al., 2025) utilized

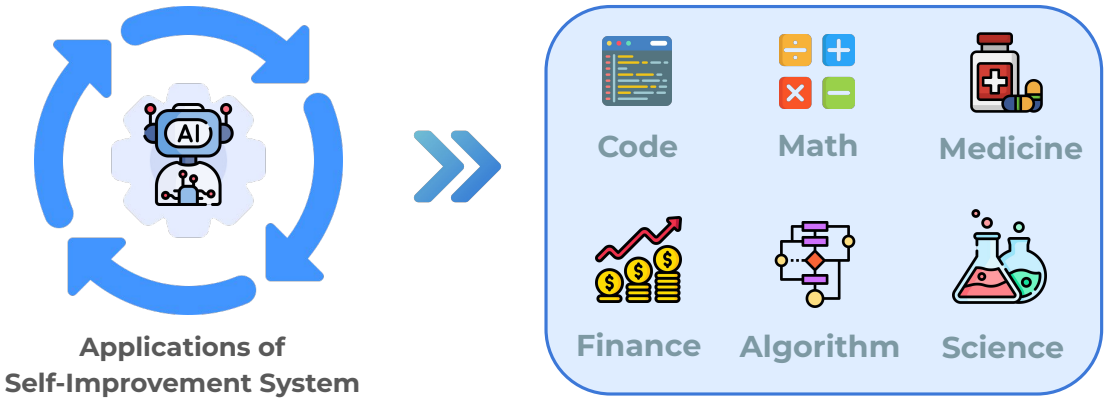

Figure 11: **Applications of the self-improvement system.** We highlight representative applications of self-improved LLMs and their extensions to self-evolving agents across six major domains: *Code*, *Math*, *Medicine*, *Finance*, *Algorithm*, and Science.

Table 14: **Overview of domain-specific self-improvement models and self-evolving agents.** Approaches are organized by application domains, including code, math, medicine, finance, algorithm, and science.

Domain	**Methods**
Code	SATLUTION (Yu et al., 2025a), CURE (Wang et al., 2025i), LLMLOOP (Ravi et al., 2025), SSR (Wei et al., 2025b), ReVeal (Jin et al., 2026), ACE (Zhang et al., 2026b)
Math	SIAM (Yu et al., 2024a), Goedel-Prover (Lin et al., 2025b), OpenSIR (Kwan et al., 2025), SPHERE (Singh et al., 2025), Xiong et al. (2025), rStar-Math (Guan et al., 2025)
Medicine	Agent Hospital (Li et al., 2025c), HealthFlow (Zhu et al., 2025a), MDTeamGPT (Chen et al., 2025a), MedAgentSim (Almansoori et al., 2025), EvoClinician (He et al., 2026), MedReflect (Huang et al., 2026b)
Finance	FINMEM (Li et al., 2024a), FinAgent (Zhang et al., 2024d), AlphaAgents (Zhao et al., 2025b), FinRS (Liu & Dang, 2025), QuantAgents (Li et al., 2025e), Chaudhari & Charate (2025)
Algorithm	Gödel Agent (Yin et al., 2025), AlphaEvolve (Novikov et al., 2025a), Self-Developing (Ishibashi et al., 2025), SEAL (Zweiger et al., 2025), DGM (Zhang et al., 2026a)
Science	MatPilot (Ni et al., 2024), SciAgents (Ghafarollahi & Buehler, 2025), Agent Laboratory (Schmidgall et al., 2025), Knowledge-extractor (Yao et al., 2025), The AI Cosmologist I (Moss, 2025), The AI Scientist-v2 (Yamada et al., 2025), AI co-scientist (Gottweis et al., 2025), S1-NexusAgent (Team, 2026)

iterative feedback loops to optimize code robustness. Most recently, ACE (Zhang et al., 2026b) introduced agentic context engineering to manage and evolve massive repository contexts for self-improving models.

Math. This domain focuses on achieving self-evolution through formal logical consistency and rigorous verification of reasoning paths. SIAM (Yu et al., 2024a) pioneered this by using code-assisted execution to verify and refine mathematical reasoning. Subsequently, Xiong et al. (2025) introduced a self-rewarding mechanism to autonomously detect and correct step-wise errors in complex algebraic derivations. SPHERE (Singh et al., 2025) utilized self-evolved preference optimization to bridge the reasoning gap in small models for competition-level problems. Goedel-Prover (Lin et al., 2025b) advanced the field by integrating with formal languages like Lean for automated theorem proving. Furthermore, rStar-Math (Guan et al., 2025) introduced a deep-thinking evolution mechanism to master master-level math reasoning, while OpenSIR (Kwan et al., 2025) explored open-ended self-improvement for recursive mathematical discovery.

Medicine. Self-evolving systems in medicine emphasize clinical safety, diagnostic accuracy, and multidisciplinary collaboration. MDTeamGPT (Chen et al., 2025a) implemented a multi-agent framework to simulate multidisciplinary team (MDT) consultations, iteratively refining diagnostic plans. HealthFlow (Zhu et al., 2025a) utilized meta-planning to enable agents to autonomously optimize clinical research workflows. MedAgentSim (Almansoori et al., 2025) provided high-fidelity clinical simulations to accelerate the iteration of decision-making logic, while Agent Hospital (Li et al., 2025c) evolved agents within a digital twin of a hospital environment. In the latest advancements, EvoClinician (He et al., 2026) leveraged test-time evolutionary learning for multi-turn diagnosis, and MedReflect (Huang et al., 2026b) taught medical models to self-improve by correcting misdiagnoses through reflective feedback.

Finance. Financial agents evolve their strategies in high-noise environments by utilizing layered memory and risk-sensitive simulation. FINMEM (Li et al., 2024a) introduced a layered memory architecture to maintain long-term strategy stability in volatile markets. FinAgent (Zhang et al., 2024d) served as a multimodal foundation agent capable of interpreting macro-financial data through tool augmentation. QuantAgents (Li et al., 2025e) explored the evolution of multi-agent systems through large-scale simulated trading environments. FinRS (Liu & Dang, 2025) proposed a risk-sensitive framework for self-optimizing strategies under real-market constraints. AlphaAgents (Zhao et al., 2025b) optimized equity portfolios through multi-agent debate, while Chaudhari & Charate (2025) combined continual learning with neuro-symbolic reasoning to evolve financial risk prediction logic.

Algorithm. This domain investigates the theoretical limits and meta-capabilities of agents to modify their own optimization logic. Goedel Agent (Yin et al., 2025) and AlphaEvolve (Novikov et al., 2025a) explored self-referential frameworks for discovering new scientific and algorithmic strategies. Self-Developing (Ishibashi et al., 2025) and DGM (Zhang et al., 2026a) focused on the open-ended evolution of algorithms to enable recursive model enhancement. SEAL (Zweiger et al., 2025) investigated task adaptation and iterative self-incentivization for autonomous search.

Science. Scientific research agents drive discovery by automating experimental design and multidisciplinary data synthesis. MatPilot (Ni et al., 2024) and SciAgents (Ghafarollahi & Buehler, 2025) pioneered the use of intelligent graph reasoning and human-machine collaboration for materials discovery. Building on this, The AI Scientist-v2 (Yamada et al., 2025) introduced tree-search discovery to automate the entire research pipeline from hypothesis generation to paper drafting. AI co-scientist (Gottweis et al., 2025) focused on collaborative breakthroughs, while Agent Laboratory (Schmidgall et al., 2025) simulated an autonomous research team for lab-scale scientific inquiry. Knowledge-extractor (Yao et al., 2025) developed self-evolving frameworks for hydrogen energy research, and The AI Cosmologist I (Moss, 2025) automated statistical inference for cosmological data. Finally, S1-NexusAgent (Team, 2026) provided a unified framework for cross-disciplinary scientific evolution.

Others. Self-improvement techniques have also demonstrated significant potential across various other domains such as agentic system, research process, and model safety. In the autonomous design and evolution of agentic systems, ADAS (Hu et al., 2025c) and AgentEvolver (Zhai et al., 2025) facilitate the automated optimization of agent components and the self-directed generation of tasks. Within the realm of autonomous research, FARS (Analemma, 2026) and AutoResearchClaw (Liu et al., 2026a) automate the research work-

flow, managing everything from hypothesis generation to experimental execution. Furthermore, SISF (Slater, 2025) leverages self-improvement mechanisms to enhance the dynamic safety of models.

9 Future Outlook

As self-improvement research matures, the field is gradually shifting from isolated optimization techniques toward more systemic and agentic perspectives. Rather than treating self-improvement as a collection of local training tricks, future progress will likely depend on rethinking the entire lifecycle of model evolution. We highlight four key directions that may shape the next stage of self-improving LLMs.

From Model-Level Optimization to End-to-End Self-Improving Systems. Current approaches often operate at the model level—improving data generation, selection, optimization, or inference refinement in isolation. However, there is a clear trend from model-centric optimization toward system-level autonomy, particularly visible in the rise of agentic systems that integrate perception, memory, planning, tool use, and environment interaction. For self-improvement to reach its full potential, future work should move beyond modular techniques and instead construct end-to-end self-improving systems, similar to the lifecycle framework proposed in this paper. Such systems would not treat the LLM as a static learner but as the core component of a broader agentic architecture that continuously acquires data, evaluates itself, updates its capabilities, and redeploys improvements within an automated loop. In this view, self-improvement becomes a property of the entire system rather than a single training stage.

Toward Specialized and Application-Centric Self-Improved Models. Self-improving models are also becoming increasingly specialized. As discussed in Section 9, self-improvement mechanisms are already being applied across domains such as scientific reasoning, coding, finance, healthcare, and interactive agents. Rather than pursuing a single monolithic general-purpose self-improving model, future systems may evolve into domain-specialized self-improving agents that iteratively refine their competence within constrained environments. When combined with agentic architectures, such specialization enables tightly coupled feedback loops between environment interaction and capability growth. This suggests a future where self-improvement is embedded within domain-specific ecosystems, allowing models to accumulate structured expertise while maintaining controllable scope and measurable progress.

Unified Benchmarks for Self-Improvement and Autonomous Evaluation. Despite rapid methodological advances, the field still lacks a unified benchmark explicitly designed to measure self-improvement. Most current evaluations rely on static downstream datasets, which fail to capture recursive gains, learning efficiency, stability across iterations, or long-term robustness. As discussed in Section 6 on autonomous evaluation, future benchmarks should directly assess the evolution process itself: how quickly a system improves, how reliably it avoids degradation, and how efficiently it converts feedback into durable capability gains. Establishing a standardized evaluation suite for self-improvement—covering iterative performance growth, safety constraints, and cross-domain transfer—would provide a shared foundation for comparing paradigms and identifying sustainable improvement strategies.

Balancing Automation and Human Oversight. Finally, as self-improvement becomes increasingly automated, a central challenge lies in balancing autonomy with human supervision. Fully automated evolution promises scalability and reduced human labor, yet excessive autonomy may introduce alignment drift, reward hacking, or unintended capability shifts. Conversely, heavy human oversight may constrain scalability and limit continuous improvement. Future research must therefore explore principled frameworks for adjustable supervision, where human guidance, auditing mechanisms, and automated safeguards coexist. The goal is not to eliminate humans from the loop, but to design adaptive oversight structures that calibrate the degree of autonomy according to task criticality, risk level, and system maturity. Achieving this balance will be essential for ensuring that self-improving systems remain both powerful and aligned in the long term.

10 Conclusion

This study introduces a unified system for self-improvement in LLMs, which integrates key components: data acquisition, data selection, model optimization, and inference refinement into a single closed-loop pipeline. By coupling these modules with an autonomous evaluation mechanism, the system enables continuous improvement with reduced reliance on human-annotated data.

More importantly, self-improvement is shifting from optimizing isolated techniques to building integrated systems. Instead of treating each component independently, future approaches emphasize coordination across the full pipeline. Under this paradigm, models move beyond passive learning and begin to actively participate in their own improvement, such as identifying weaknesses and selecting appropriate strategies to improve.

Looking ahead, we envision systems that can plan their own improvement process, adapt to new scenarios, and iteratively refine both their data and behavior. Rather than relying on fixed training pipelines, these systems can dynamically identify their weaknesses, acquire or generate targeted data, and select appropriate optimization strategies. They can also adjust their behavior based on feedback from the environment, enabling continuous adaptation in changing settings. This represents a transition from externally guided training to more autonomous and adaptive learning systems. Accordingly, evaluation must also evolve—from static, one-time benchmarks to dynamic and continuous frameworks that can monitor how models change over time and assess the stability of their self-improvement process.

However, this direction also introduces significant challenges. As models become more autonomous, risks such as data quality degradation, reward hacking, and misalignment become more prominent. The key difficulty lies in balancing increasing autonomy with robust safety and control mechanisms, ensuring that self-improvement remains stable, reliable, and aligned with human values.

In summary, self-improvement is evolving toward a system-driven and more autonomous paradigm, where models actively participate in their own development through tightly integrated learning pipelines.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL https://arxiv.org/abs/2404.14219.

Marwa Abdulhai, Isadora White, Charlie Victor Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin Xu, and Sergey Levine. LMRL gym: Benchmarks for multi-turn reinforcement learning with language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=hmGhP5DO2W.

Pranjal Aggarwal, Bryan Parno, and Sean Welleck. Alphaverus: Bootstrapping formally verified code generation through self-improving translation and treefinement. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=tU8QKX4dMI.

Jihyun Janice Ahn and Wenpeng Yin. Prompt-reverse inconsistency: LLM self-inconsistency beyond generative randomness and prompt paraphrasing. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=yfRkNRFLzl.

Syeda Nahida Akter, Shrimai Prabhumoye, Eric Nyberg, Mostofa Patwary, Mohammad Shoeybi, Yejin Choi, and Bryan Catanzaro. Front-loading reasoning: The synergy between pretraining and post-training data, 2025. URL https://arxiv.org/abs/2510.03264.

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=asgBo3FNdg.

Mohammad Almansoori, Komal Kumar, and Hisham Cholakkal. Medagentsim: Self-evolving multi-agent simulations for realistic clinical interactions. In *International Conference on Medical Image Computing*

and Computer-Assisted Intervention, pp. 362–372. Springer, 2025. URL https://papers.miccai.org/miccai-2025/paper/2575_paper.pdf.

Analemma. Introducing fars, 2026. URL https://analemma.ai/blog/introducing-fars/.

Huan ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Qihan Ren, Yiran Wu, Hongru WANG, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. 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. ISSN 2835-8856. URL https://openreview.net/forum?id=CTr3bovS5F. Survey Certification.

Zachary Ankner, Cody Blakeney, Kartik Sreenivasan, Max Marion, Matthew L Leavitt, and Mansheej Paul. Perplexed by perplexity: Perplexity-based data pruning with small reference models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=1GTARJhxtq.

Dhruv Atreja. ALAS: Autonomous learning agent for self-updating language models, 2025. URL https://arxiv.org/abs/2508.15805.

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, Jeongyeon Nam, and Donghyun Kwak. Online difficulty filtering for reasoning oriented reinforcement learning. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 700–719, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. URL https://aclanthology.org/2026.eacl-long.30/.

Tianyi Bai, Ling Yang, Zhen Hao Wong, Fupeng Sun, Xinlin Zhuang, Jiahui Peng, Chi Zhang, Lijun Wu, Qiu Jiantao, Wentao Zhang, Binhang Yuan, and Conghui He. Efficient pretraining data selection for language models via multi-actor collaboration. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9465–9491, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.466. URL https://aclanthology.org/2025.acl-long.466/.

Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y. Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation, 2025. URL https://arxiv.org/abs/2503.11926.

Hritik Bansal, John Dang, and Aditya Grover. Peering through preferences: Unraveling feedback acquisition for aligning large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dKl6lMwbCy.

Andrew M. Bean, Ryan Othniel Kearns, Angelika Romanou, Franziska Sofia Hafner, Harry Mayne, Jan Batzner, Negar Foroutan, Chris Schmitz, Karolina Korgul, Hunar Batra, Oishi Deb, Emma Beharry, Cornelius Emde, Thomas Foster, Anna Gausen, María Grandury, Simeng Han, Valentin Hofmann, Lujain Ibrahim, Hazel Kim, Hannah Rose Kirk, Fangru Lin, Gabrielle Kaili-May Liu, Lennart Luettgau, Jabez Magomere, Jonathan Rystrøm, Anna Sotnikova, Yushi Yang, Yilun Zhao, Adel Bibi, Antoine Bosselut, Ronald Clark, Arman Cohan, Jakob Nicolaus Foerster, Yarin Gal, Scott A. Hale, Inioluwa Deborah Raji, Christopher Summerfield, Philip Torr, Cozmin Ududec, Luc Rocher, and Adam Mahdi. Measuring what matters: Construct validity in large language model benchmarks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=mdA5lVvNcU.

Loubna Ben Allal, Thomas Mesnard, Gall de la Rosa, Alexander Ozouf, Anton Lozhkov, Leandro Danilowich, Pablo Villegas, Zaid Alyafeai Castillo, and Thomas Wolf. Cosmopedia: how to create large-scale synthetic data for pre-training, March 2024. URL https://huggingface.co/blog/cosmopedia. Hugging Face Blog.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, pp. 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097. doi: 10.1145/3442188.3445922. URL https://doi.org/10.1145/3442188.3445922.

Shelly Bensal, Umar Jamil, Christopher Bryant, Melisa Russak, Kiran Kamble, Dmytro Mozolevskyi, Muayad Ali, and Waseem AlShikh. Reflect, retry, reward: Self-improving llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2505.24726.

Vicente Bermejo, Andrés Gago, Ramiro Gálvez, and Nicolás Harari. LLMs outperform outsourced human coders on complex textual analysis. *Scientific Reports*, 15, November 2025. doi: 10.1038/s41598-025-23798-y. URL https://www.nature.com/articles/s41598-025-23798-y.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: solving elaborate problems with large language models. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i16.29720. URL https://doi.org/10.1609/aaai.v38i16.29720.

Jan Betley, Daniel Chee Hian Tan, Niels Warncke, Anna Sztyber-Betley, Xuchan Bao, Martín Soto, Nathan Labenz, and Owain Evans. Emergent misalignment: Narrow finetuning can produce broadly misaligned LLMs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=aOIJ2gVRWW.

Robi Bhattacharjee, Sanjoy Dasgupta, and Kamalika Chaudhuri. Data-copying in generative models: A formal framework. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 2364–2396. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/bhattacharjee23a.html.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. Forest-of-thought: Scaling test-time compute for enhancing LLM reasoning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 4253–4267. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/bi25a.html.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Jang, and Zack Hui. Windows agent arena: Evaluating multi-modal os agents at scale, 2024. URL https://arxiv.org/abs/2409.08264.

Samuel R. Bowman. Eight things to know about large language models, 2023. URL https://arxiv.org/abs/2304.00612.

Martin Briesch, Dominik Sobania, and Franz Rothlauf. Large language models suffer from their own output: An analysis of the self-consuming training loop, 2024. URL https://arxiv.org/abs/2311.16822.

Benjamin Brimacombe and Jiawei Zhou. Quick back-translation for unsupervised machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 8521–8534, 2023.

Andrei Z. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of SEQUENCES 1997*, pp. 21–29. IEEE Computer Society, June 1997. doi: 10.1109/SEQUEN.1997.666900. URL https://ieeexplore.ieee.org/document/666900.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024. URL https://arxiv.org/abs/2407.21787.

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

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ETKGuby0hcs.

Adam Byerly and Daniel Khashabi. Self-consistency falls short! the adverse effects of positional bias on long-context problems, 2025. URL https://arxiv.org/abs/2411.01101.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. In *Forty-first International Conference on Machine Learning*, 2024a. URL https://arxiv.org/abs/2401.10774.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=qV83K9d5WB.

Linbo Cao and Jinman Zhao. Pretraining on the test set is no longer all you need: A debate-driven approach to QA benchmarks. *CoRR*, abs/2507.17747, 2025. doi: 10.48550/ARXIV.2507.17747. URL https://doi.org/10.48550/arXiv.2507.17747.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomek Korbak, David Lindner, Pedro Freire, Tony Tong Wang, Samuel Marks, Charbel-Raphael Segerie, Micah Carroll, Andi Peng, Phillip J.K. Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Biyik, Anca Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. Open problems and fundamental limitations of reinforcement learning from human feedback. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=bx24KpJ4Eb. Survey Certification, Featured Certification.

Jiajun Chai, Guojun Yin, Zekun Xu, Chuhuai Yue, Yi Jia, Siyu Xia, Xiaohan Wang, Jiwen Jiang, Xiaoguang Li, Chengqi Dong, Hang He, and Wei Lin. Rlfactory: A plug-and-play reinforcement learning post-training framework for llm multi-turn tool-use, 2025. URL https://arxiv.org/abs/2509.06980.

Fu-Chieh Chang, Yu-Ting Lee, Hui-Ying Shih, Yi Hsuan Tseng, and Pei-Yuan Wu. RL-STar: Theoretical analysis of reinforcement learning frameworks for self-taught reasoner. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=Oo2XthxKB9.

Akash Vijayrao Chaudhari and Pallavi Ashokrao Charate. Self-evolving ai agents for financial risk prediction using continual learning and neuro-symbolic reasoning. *Journal of Recent Trends in Computer Science and Engineering (JRTCSE)*, 13(2):76–92, 2025. URL https://jrtcse.com/index.php/home/article/view/JRTCSE.2025.13.2.8.

Angelica Chen, Jason Phang, Alicia Parrish, Vishakh Padmakumar, Chen Zhao, Samuel R. Bowman, and Kyunghyun Cho. Two failures of self-consistency in the multi-step reasoning of LLMs. *Transactions on Machine Learning Research*, 2024a. ISSN 2835-8856. URL https://openreview.net/forum?id=5nBqY1y96B.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling, 2023. URL https://arxiv.org/abs/2302.01318.

Justin Chen, Swarnadeep Saha, and Mohit Bansal. ReConcile: Round-table conference improves reasoning via consensus among diverse LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7066–7085, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.381. URL https://aclanthology.org/2024.acl-long.381/.

Kai Chen, Xinfeng Li, Tianpei Yang, Hewei Wang, Wei Dong, and Yang Gao. Mdteamgpt: A self-evolving llm-based multi-agent framework for multi-disciplinary team medical consultation, 2025a. URL https://arxiv.org/abs/2503.13856.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. Alpagasus: Training a better alpaca with fewer data. In *The Twelfth International Conference on Learning Representations*, 2024c. URL https://openreview.net/forum?id=FdVXgSJhvz.

Ruizhe Chen, Xiaotian Zhang, Meng Luo, Wenhao Chai, and Zuozhu Liu. PAD: Personalized alignment at decoding-time. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=e7AUJpP8bV.

Simin Chen, Yiming Chen, Zexin Li, Yifan Jiang, Zhongwei Wan, Yixin He, Dezhi Ran, Tianle Gu, Haizhou Li, Tao Xie, and Baishakhi Ray. Benchmarking large language models under data contamination: A survey from static to dynamic evaluation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 10080–10098, Suzhou, China, November 2025c. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.511. URL https://aclanthology.org/2025.emnlp-main.511/.

Simin Chen, Yiming Chen, Zexin Li, Yifan Jiang, Zhongwei Wan, Yixin He, Dezhi Ran, Tianle Gu, Haizhou Li, Tao Xie, and Baishakhi Ray. Benchmarking large language models under data contamination: A survey from static to dynamic evaluation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 10080–10098, Suzhou, China, November 2025d. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.511. URL https://aclanthology.org/2025.emnlp-main.511/.

Simin Chen, Pranav Pusarla, and Baishakhi Ray. DyCodeEval: Dynamic benchmarking of reasoning capabilities in code large language models under data contamination. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 8890–8909. PMLR, 13–19 Jul 2025e. URL https://proceedings.mlr.press/v267/chen25ba.html.

Sirui Chen, Yunzhe Qi, Mengting Ai, Yifan Sun, Ruizhong Qiu, Jiaru Zou, and Jingrui He. Influence-preserving proxies for gradient-based data selection in LLM finetuning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=PDNpRLxDlI.

Tong Chen, Akari Asai, Niloofar Mireshghallah, Sewon Min, James Grimmelmann, Yejin Choi, Hannaneh Hajishirzi, Luke Zettlemoyer, and Pang Wei Koh. CopyBench: Measuring literal and non-literal reproduction of copyright-protected text in language model generation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15134–15158, Miami, Florida, USA, November 2024d. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.844. URL https://aclanthology.org/2024.emnlp-main.844/.

Weizhe Chen, Sven Koenig, and Bistra Dilkina. Iterative deepening sampling as efficient test-time scaling, 2025f. URL https://arxiv.org/abs/2502.05449.

Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. Self-evolving curriculum for LLM reasoning. *CoRR*, abs/2505.14970, 2025g. doi: 10.48550/ARXIV.2505.14970. URL https://doi.org/10.48550/arXiv.2505.14970.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language models. In *ICML 2024 Workshop on In-Context Learning*, 2024e. URL https://openreview.net/forum?id=LjsjHF7nAN.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations*, 2024f. URL https://openreview.net/forum?id=KuPixIqPiq.

Yixing Chen, Yiding Wang, Siqi Zhu, Haofei Yu, Tao Feng, Muhan Zhang, Mostofa Patwary, and Jiaxuan You. Multi-agent evolve: Llm self-improve through co-evolution, 2025h. URL https://arxiv.org/abs/2510.23595.

Zhaorun Chen, Zhuokai Zhao, Hongyin Luo, Huaxiu Yao, Bo Li, and Jiawei Zhou. Halc: Object hallucination reduction via adaptive focal-contrast decoding. In *Forty-first International Conference on Machine Learning*, 2024g.

Zhuoyue Chen, Jihai Zhang, Ben Liu, Fangquan Lin, and Wotao Yin. Scale down to speed up: Dynamic data selection for reinforcement learning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 7806–7817, Suzhou, China, November 2025i. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.412. URL https://aclanthology.org/2025.findings-emnlp.412/.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. In *Forty-first International Conference on Machine Learning*, 2024h. URL https://openreview.net/forum?id=O4cHTxW9BS.

Mingyue Cheng, Jie Ouyang, Shuo Yu, Ruiran Yan, Yucong Luo, Zirui Liu, Daoyu Wang, Qi Liu, and Enhong Chen. Agent-R1: Training powerful LLM agents with end-to-end reinforcement learning, 2025. URL https://arxiv.org/abs/2511.14460.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory, 2025. URL https://arxiv.org/abs/2504.19413.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113, 2023. URL https://jmlr.org/papers/v24/22-1144.html.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R. Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Th6NyL07na.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew J. Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. Textworld:

A learning environment for text-based games. In Tristan Cazenave, Abdallah Saffidine, and Nathan R. Sturtevant (eds.), *Computer Games - 7th Workshop, CGW 2018, Held in Conjunction with the 27th International Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, July 13, 2018, Revised Selected Papers*, volume 1017 of *Communications in Computer and Information Science*, pp. 41–75. Springer, 2018. doi: 10.1007/978-3-030-24337-1_3. URL https://doi.org/10.1007/978-3-030-24337-1_3.

Hui Dai, Ryan Teehan, and Mengye Ren. Are llms prescient? A continuous evaluation using daily news as the oracle. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/dai25l.html.

Preetam Prabhu Srikar Dammu, Himanshu Naidu, and Chirag Shah. Dynamic-kgqa: A scalable framework for generating adaptive question answering datasets. In Nicola Ferro, Maria Maistro, Gabriella Pasi, Omar Alonso, Andrew Trotman, and Suzan Verberne (eds.), *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2025, Padua, Italy, July 13-18, 2025*, pp. 3498–3508. ACM, 2025. doi: 10.1145/3726302.3730324. URL https://doi.org/10.1145/3726302.3730324.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, Xuantang Xiong, Lei Han, Zhiyuan Liu, and Maosong Sun. Multi-agent collaboration via evolving orchestration. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=L0xZPXT3le.

Nirjhar Das, Souradip Chakraborty, Aldo Pacchiano, and Sayak Ray Chowdhury. Active preference optimization for sample efficient RLHF. In Rita P. Ribeiro, Bernhard Pfahringer, Nathalie Japkowicz, Pedro Larrañaga, Alípio M. Jorge, Carlos Soares, Pedro H. Abreu, and João Gama (eds.), *Machine Learning and Knowledge Discovery in Databases. Research Track - European Conference, ECML PKDD 2025, Porto, Portugal, September 15-19, 2025, Proceedings, Part V*, volume 16017 of *Lecture Notes in Computer Science*, pp. 96–112. Springer, 2025. doi: 10.1007/978-3-032-06096-9_6. URL https://doi.org/10.1007/978-3-032-06096-9_6.

Ona de Gibert, Graeme Nail, Nikolay Arefyev, Marta Bañón, Jelmer van der Linde, Shaoxiong Ji, Jaume Zaragoza-Bernabeu, Mikko Aulamo, Gema Ramírez-Sánchez, Andrey Kutuzov, Sampo Pyysalo, Stephan Oepen, and Jörg Tiedemann. A new massive multilingual dataset for high-performance language technologies. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 1116–1128, Torino, Italia, May 2024. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.100/.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3369–3391, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.222. URL https://aclanthology.org/2022.emnlp-main.222/.

Yangruibo Ding, Ben Steenhoek, Kexin Pei, Gail Kaiser, Wei Le, and Baishakhi Ray. TRACED: Execution-aware pre-training for source code, 2023. URL https://arxiv.org/abs/2306.07487.

Dai Do, Manh Nguyen, Svetha Venkatesh, and Hung Le. Sparft: Self-paced reinforcement fine-tuning for large language models. *CoRR*, abs/2508.05015, aug 2025. doi: 10.48550/ARXIV.2508.05015. URL https://doi.org/10.48550/arXiv.2508.05015.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 1286–1305,

Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.98. URL https://aclanthology.org/2021.emnlp-main.98/.

Elvis Dohmatob, Yunzhen Feng, Pu Yang, Francois Charton, and Julia Kempe. A tale of tails: Model collapse as a change of scaling laws, 2024. URL https://arxiv.org/abs/2402.07043.

Elvis Dohmatob, Yunzhen Feng, Arjun Subramonian, and Julia Kempe. Strong model collapse. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=et5l9qPUhm.

Qingxiu Dong, Li Dong, Xingxing Zhang, Zhifang Sui, and Furu Wei. Self-boosting large language models with synthetic preference data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7visV100Ms.

Xiangjue Dong, Maria Teleki, and James Caverlee. A survey on llm inference-time self-improvement, 2024. URL https://arxiv.org/abs/2412.14352.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P. Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen S. Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V. Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. Glam: Efficient scaling of language models with mixture-of-experts. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5547–5569. PMLR, 2022. URL https://proceedings.mlr.press/v162/du22c.html.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=zj7YuTE4t8.

Yong Du, Yuchen Yan, Fei Tang, Zhengxi Lu, Chang Zong, Weiming Lu, Shengpei Jiang, and Yongliang Shen. Test-time reinforcement learning for gui grounding via region consistency, 2025. URL https://arxiv.org/abs/2508.05615.

Andrzej Dulny, Andreas Hotho, and Anna Krause. Dynabench: A benchmark dataset for learning dynamical systems from low-resolution data. In Danai Koutra, Claudia Plant, Manuel Gomez-Rodriguez, Elena Baralis, and Francesco Bonchi (eds.), *Machine Learning and Knowledge Discovery in Databases: Research Track - European Conference, ECML PKDD 2023, Turin, Italy, September 18-22, 2023, Proceedings, Part I*, volume 14169 of *Lecture Notes in Computer Science*, pp. 438–455. Springer, 2023. doi: 10.1007/978-3-031-43412-9_26. URL https://doi.org/10.1007/978-3-031-43412-9_26.

Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alexander Nicholas D'Amour, Krishnamurthy Dj Dvijotham, Adam Fisch, Katherine A Heller, Stephen Robert Pfohl, Deepak Ramachandran, Peter Shaw, and Jonathan Berant. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=5u1GpUkKtG.

Ronen Eldan and Yuanzhi Li. TinyStories: How small can language models be and still speak coherent english?, 2023. URL https://arxiv.org/abs/2305.07759.

FAIR CodeGen team, Jade Copet, Quentin Carbonneaux, Gal Cohen, Jonas Gehring, Jacob Kahn, Jannik Kossen, Felix Kreuk, Emily McMilin, Michel Meyer, Yuxiang Wei, David Zhang, Kunhao Zheng, et al. CWM: An open-weights LLM for research on code generation with world models, 2025. URL https://arxiv.org/abs/2510.02387.

FAIR Diplomacy Team, Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mojtaba Komeili, Karthik Konath,

Minae Kwon, Adam Lerer, Mike Lewis, Alexander H. Miller, Sasha Mitts, Adithya Renduchintala, Stephen Roller, Dirk Rowe, Weiyan Shi, Joe Spisak, Alexander Wei, David Wu, Hugh Zhang, and Markus Zijlstra. Human-level play in the game of Diplomacy by combining language models with strategic reasoning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=znABFGQIbLq.

Ziqing Fan, Yuqiao Xian, Yan Sun, Li Shen, and Ke Shen. Joint selection for large-scale pre-training data via policy gradient-based mask learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=fs2uDib85s.

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. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems, 2025a. URL https://arxiv.org/abs/2508.07407.

Wei Fang, Yang Zhang, Kaizhi Qian, James Glass, and Yada Zhu. Play2prompt: Zero-shot tool instruction optimization for llm agents via tool play, 2025b. URL https://arxiv.org/abs/2503.14432.

Wenkai Fang, Shunyu Liu, Yang Zhou, Kongcheng Zhang, Tongya Zheng, Kaixuan Chen, Mingli Song, and Dacheng Tao. SeRL: Self-play reinforcement learning for large language models with limited data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025c. URL https://openreview.net/forum?id=ZF93vyH9He.

Yixiong Fang, Ziran Yang, Zhaorun Chen, Zhuokai Zhao, and Jiawei Zhou. Enhancing vision-language model reliability with uncertainty-guided dropout decoding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025d.

Timofey Fedoseev, Dimitar Iliev Dimitrov, Timon Gehr, and Martin Vechev. Constraint-based synthetic data generation for LLM mathematical reasoning. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS 2024*, December 2024. URL https://files.sri.inf.ethz.ch/z3_llm/z3_llm.pdf.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms, 2025a. URL https://arxiv.org/abs/2504.11536.

Yiyang Feng, Yichen Wang, Shaobo Cui, Boi Faltings, Mina Lee, and Jiawei Zhou. Unraveling misinformation propagation in LLM reasoning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 11683–11707, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.627. URL https://aclanthology.org/2025.findings-emnlp.627/.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13481–13544. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/fernando24a.html.

Pedro Lobato Ferreira, Wilker Aziz, and Ivan Titov. Truthful or fabricated? using causal attribution to mitigate reward hacking in explanations. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=nkdPLuKoL5.

Eve Fleisig, Aubrie Amstutz, Chad Atalla, Su Lin Blodgett, Hal Daumé III, Alexandra Olteanu, Emily Sheng, Dan Vann, and Hanna Wallach. FairPrism: Evaluating fairness-related harms in text generation. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6231–6251, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.343. URL https://aclanthology.org/2023.acl-long.343/.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4): 128–135, 1999. URL https://doi.org/10.1002/0470018860.s00096.

Yanjun Fu, Faisal Hamman, and Sanghamitra Dutta. T-SHIRT: Token-selective hierarchical data selection for instruction tuning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=oN5YVZ9JeF.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of LLM inference using lookahead decoding. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=eDjvSFOkXw.

Daniele Gambetta, Gizem Gezici, Fosca Giannotti, Dino Pedreschi, Alistair Knott, and Luca Pappalardo. Learning by surprise: Surplexity for mitigating model collapse in generative ai, 2025. URL https://arxiv.org/abs/2410.12341.

Tiantian Gan and Qiyao Sun. Rag-mcp: Mitigating prompt bloat in llm tool selection via retrieval-augmented generation, 2025. URL https://arxiv.org/abs/2505.03275.

Apurva Gandhi and Graham Neubig. Go-browse: Training web agents with structured exploration, 2025. URL https://arxiv.org/abs/2506.03533.

Hongcheng Gao, Yue Liu, Yufei He, Longxu Dou, Chao Du, Zhijie Deng, Bryan Hooi, Min Lin, and Tianyu Pang. Flowreasoner: Reinforcing query-level meta-agents, 2025. URL https://arxiv.org/abs/2504.15257.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, dec 2020. doi: 10.48550/arXiv.2101.00027. URL https://arxiv.org/pdf/2101.00027.

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Quentin Carbonneaux, Taco Cohen, and Gabriel Synnaeve. RLEF: Grounding code LLMs in execution feedback with reinforcement learning, 2025. URL https://arxiv.org/abs/2410.02089.

Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Tomasz Korbak, Henry Sleight, Rajashree Agrawal, John Hughes, Dhruv Bhandarkar Pai, Andrey Gromov, Dan Roberts, Diyi Yang, David L. Donoho, and Sanmi Koyejo. Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=5B2K4LRgmz.

Alireza Ghafarollahi and Markus J Buehler. Sciagents: automating scientific discovery through bioinspired multi-agent intelligent graph reasoning. *Advanced Materials*, 37(22):2413523, 2025. URL https://advanced.onlinelibrary.wiley.com/doi/abs/10.1002/adma.202413523.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. 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)*, pp. 6112–6121, 2019.

Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. Chatgpt outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30), July 2023. doi: 10.1073/pnas.2305016120. URL https://arxiv.org/pdf/2303.15056.

Shahriar Golchin and Mihai Surdeanu. Time travel in LLMs: Tracing data contamination in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=2Rwq6c3tvr.

C. A. E. Goodhart. *Problems of Monetary Management: The UK Experience*, pp. 91–121. Macmillan Education UK, London, 1984. ISBN 978-1-349-17295-5. doi: 10.1007/978-1-349-17295-5_4. URL https://doi.org/10.1007/978-1-349-17295-5_4.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, Khaled Saab, Dan Popovici, Jacob Blum, Fan Zhang, Katherine Chou, Avinatan Hassidim, Burak Gokturk, Amin Vahdat, Pushmeet Kohli, Yossi Matias, Andrew Carroll, Kavita Kulkarni, Nenad Tomasev, Yuan Guan, Vikram Dhillon, Eeshit Dhaval Vaishnav, Byron Lee, Tiago R D Costa, José R Penadés, Gary Peltz, Yunhan Xu, Annalisa Pawlosky, Alan Karthikesalingam, and Vivek Natarajan. Towards an ai co-scientist, 2025. URL https://arxiv.org/abs/2502.18864.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing, 2024. URL https://arxiv.org/abs/2305.11738.

Ryan Greenblatt, Carson Denison, Benjamin Wright, Fabien Roger, Monte MacDiarmid, Sam Marks, Johannes Treutlein, Tim Belonax, Jack Chen, David Duvenaud, Akbir Khan, Julian Michael, Sören Mindermann, Ethan Perez, Linda Petrini, Jonathan Uesato, Jared Kaplan, Buck Shlegeris, Samuel R. Bowman, and Evan Hubinger. Alignment faking in large language models, 2024. URL https://arxiv.org/abs/2412.14093.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*, 2018.

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 79081–79094. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/f9f54762cbb4fe4dbffdd4f792c31221-Paper-Conference.pdf.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small LLMs can master math reasoning with self-evolved deep thinking. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=5zwF1GizFa.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Siva Kanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need, 2023. URL https://arxiv.org/abs/2306.11644.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang

Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=ZG3RaNIsO8.

Yanzhu Guo, Guokan Shang, Michalis Vazirgiannis, and Chloé Clavel. The curious decline of linguistic diversity: Training language models on synthetic text. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 3589–3604, Mexico City, Mexico, June 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.228. URL https://aclanthology.org/2024.findings-naacl.228/.

Srishti Gureja, Elena Tommasone, Jingyi He, Sara Hooker, Matthias Gallé, and Marzieh Fadaee. Verification limits code llm training, 2025. URL https://arxiv.org/abs/2509.20837.

Thilo Hagendorff. Deception abilities emerged in large language models. *Proceedings of the National Academy of Sciences*, 121(24), June 2024. ISSN 1091-6490. doi: 10.1073/pnas.2317967121. URL http://dx.doi.org/10.1073/pnas.2317967121.

Rajarshi Haldar and Julia Hockenmaier. Rating roulette: Self-inconsistency in LLM-as-a-judge frameworks. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 24986–25004, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.1361. URL https://aclanthology.org/2025.findings-emnlp.1361/.

Jindong Han, Hao Liu, Jun Fang, Naiqiang Tan, and Hui Xiong. Automatic instruction data selection for large language models via uncertainty-aware influence maximization. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, pp. 4969–4979, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400712746. doi: 10.1145/3696410.3714817. URL https://doi.org/10.1145/3696410.3714817.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 45870–45894. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8fd1a81c882cd45f64958da6284f4a3f-Paper-Conference.pdf.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=CNL2bku4ra.

Amine El hattami, Nicolas Chapados, and Christopher Pal. Spaced scheduling for large language model training. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=p0KTYl2B9T.

Matthew J. Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. Interactive fiction games: A colossal adventure. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7903–7910. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6297. URL https://doi.org/10.1609/aaai.v34i05.6297.

Yufei He, Juncheng Liu, Zhiyuan Hu, Yulin Chen, Yue Liu, Yuan Sui, Yibo Li, Nuo Chen, Jun Hu, Bryan Hooi, Xinxing Xu, and Jiang Bian. Evoclinician: A self-evolving agent for multi-turn medical diagnosis via test-time evolutionary learning, 2026. URL https://arxiv.org/abs/2601.22964.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. Rest: Retrieval-based speculative decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1582–1595, 2024.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=iBBcRUlOAPR.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Jawad Hossain, Xiangyu Guo, Jiawei Zhou, and Chong Liu. Hintmr: Eliciting stronger mathematical reasoning in small language models. *arXiv preprint arXiv:2604.12229*, 2026.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-STar: Training verifiers for self-taught reasoners. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=stmqBSW2dV.

Kaiyuan Hou, Minghui Zhao, Lilin Xu, Yuang Fan, and Xiaofan Jiang. Tdbench: Benchmarking vision-language models in understanding top-down images. *CoRR*, abs/2504.03748, 2025. doi: 10.48550/ARXIV.2504.03748. URL https://doi.org/10.48550/arXiv.2504.03748.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Homayoon Nakada, Junjie Gupta, Brenden M. Lake, and Vishnu Shankar. Instruction pre-training: Language models are supervised multitask learners. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18257–18280, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1021. URL https://aclanthology.org/2024.emnlp-main.1021/.

Lanxiang Hu, Qiyu Li, Anze Xie, Nan Jiang, Ion Stoica, Haojian Jin, and Hao Zhang. Gamearena: Evaluating LLM reasoning through live computer games. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025a. URL https://openreview.net/forum?id=SeQ8l8xo1r.

Pengfei Hu, Zhenrong Zhang, Qikai Chang, Shuhang Liu, Jiefeng Ma, Jun Du, Jianshu Zhang, Quan Liu, Jianqing Gao, Feng Ma, and Qingfeng Liu. Prm-bas: Enhancing multimodal reasoning through prm-guided beam annealing search, 2025b. URL https://arxiv.org/abs/2504.10222.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025c. URL https://openreview.net/forum?id=t9U3LW7JVX.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025d. URL https://openreview.net/forum?id=t9U3LW7JVX.

Xuhao Hu, Peng Wang, Xiaoya Lu, Dongrui Liu, Xuanjing Huang, and Jing Shao. Llms deceive unintentionally: Emergent misalignment in dishonesty from misaligned samples to biased human-ai interactions, 2026. URL https://arxiv.org/abs/2510.08211.

Yue Hu, Yuzhu Cai, Yaxin Du, Xinyu Zhu, Xiangrui Liu, Zijie Yu, Yuchen Hou, Shuo Tang, and Siheng Chen. Self-evolving multi-agent collaboration networks for software development. In *The Thirteenth International Conference on Learning Representations*, 2025e. URL https://openreview.net/forum?id=4R71pdPBZp.

Audrey Huang, Adam Block, Dylan J Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T. Ash, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=WJaUkwci9o.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-Zero: Self-evolving reasoning LLM from zero data. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=96apU6YzSO.

James Y. Huang, Sailik Sengupta, Daniele Bonadiman, Yi-An Lai, Arshit Gupta, Nikolaos Pappas, Saab Mansour, Katrin Kirchhoff, and Dan Roth. DeAL: Decoding-time alignment for large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26280–26300, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1274. URL https://aclanthology.org/2025.acl-long.1274/.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1051–1068, Singapore, December 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.67. URL https://aclanthology.org/2023.emnlp-main.67/.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=IkmD3fKBPQ.

Shijue Huang, Wanjun Zhong, Deng Cai, Fanqi Wan, Chengyi Wang, Mingxuan Wang, Mu Qiao, and Ruifeng Xu. Empowering self-learning of llms: Inner knowledge explicitation as a catalyst. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(23):24150–24158, Apr. 2025c. doi: 10.1609/aaai.v39i23.34590. URL https://ojs.aaai.org/index.php/AAAI/article/view/34590.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and brian ichter. Inner monologue: Embodied reasoning through planning with language models. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782. PMLR, 14–18 Dec 2023b. URL https://proceedings.mlr.press/v205/huang23c.html.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey, 2024b. URL https://arxiv.org/abs/2402.02716.

Yue Huang, Yanyuan Chen, Dexuan Xu, Chenzhuo Zhao, Weihua Yue, and Yu Huang. Medreflect: Teaching medical llms to self-improve via reflective correction, 2026b. URL https://arxiv.org/abs/2510.03687.

Yufei Huang and Deyi Xiong. IT2ACL learning easy-to-hard instructions via 2-phase automated curriculum learning for large language models. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue (eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 9405–9421, Torino, Italia, May 2024. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.822/.

Trung-Kiet Huynh, Duy-Minh Dao-Sy, Thanh-Bang Cao, Phong-Hao Le, Hong-Dan Nguyen, Phu-Quy Nguyen-Lam, Minh-Luan Nguyen-Vo, Hong-Phat Pham, Phu-Hoa Pham, Thien-Kim Than, Chi-Nguyen

Tran, Huy Tran, Gia-Thoai Tran-Le, Alessio Buscemi, Le Hong Trang, and The Anh Han. Understanding LLM agent behaviours via game theory: Strategy recognition, biases and multi-agent dynamics, 2025. URL https://arxiv.org/abs/2512.07462.

Yoichi Ishibashi, Taro Yano, and Masafumi Oyamada. Can large language models invent algorithms to improve themselves?: Algorithm discovery for recursive self-improvement through reinforcement learning. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 10332–10363, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.519. URL https://aclanthology.org/2025.naacl-long.519/.

Hamish Ivison, Noah A. Smith, Hannaneh Hajishirzi, and Pradeep Dasigi. Data-efficient finetuning using cross-task nearest neighbors. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 9036–9061, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.576. URL https://aclanthology.org/2023.findings-acl.576/.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Animesh Jha, Ananjan Nandi, and Harshit Gupta. RL-guided data selection for language model finetuning. In *NeurIPS 2025 Workshop: Reliable ML from Unreliable Data*, 2025. URL https://openreview.net/forum?id=YfMIkHYxP0.

Ke Ji, Junying Chen, Anningzhe Gao, Wenya Xie, Xiang Wan, and Benyou Wang. Unlocking LLMs' self-improvement capacity with autonomous learning for domain adaptation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 21051–21067, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1084. URL https://aclanthology.org/2025.findings-acl.1084/.

Jingyi Jia and Qinbin Li. Autotool: Efficient tool selection for large language model agents, 2025. URL https://arxiv.org/abs/2511.14650.

Chunyang Jiang, Chi-Min Chan, Wei Xue, Qifeng Liu, and Yike Guo. Importance weighting can help large language models self-improve. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025a. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i23.34602. URL https://doi.org/10.1609/aaai.v39i23.34602.

Chunyang Jiang, Yonggang Zhang, Yiyang Cai, Chi-Min Chan, Yulong Liu, Mingming Chen, Wei Xue, and Yike Guo. Semantic voting: A self-evaluation-free approach for efficient llm self-improvement on unverifiable open-ended tasks, 2025b. URL https://arxiv.org/abs/2509.23067.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.792. URL https://aclanthology.org/2023.acl-long.792/.

Dongwei Jiang, Jingyu Zhang, Orion Weller, Nathaniel Weir, Benjamin Van Durme, and Daniel Khashabi. Self-[in]correct: Llms struggle with discriminating self-generated responses. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(23):24266–24275, Apr. 2025c. doi: 10.1609/aaai.v39i23.34603. URL https://ojs.aaai.org/index.php/AAAI/article/view/34603.

Xue Jiang, Yihong Dong, Mengyang Liu, Hongyi Deng, Tian Wang, Yongding Tao, Rongyu Cao, Binhua Li, Zhi Jin, Wenpin Jiao, Fei Huang, Yongbin Li, and Ge Li. CodeRL+: Improving code generation via reinforcement with execution semantics alignment, 2025d. URL https://arxiv.org/abs/2510.18471.

Yuhua Jiang, Yuwen Xiong, Yufeng Yuan, Chao Xin, Wenyuan Xu, Yu Yue, Qianchuan Zhao, and Lin Yan. Pag: Multi-turn reinforced llm self-correction with policy as generative verifier, 2025e. URL https://arxiv.org/abs/2506.10406.

Yiyang Jin, Kunzhao Xu, Hang Li, Xueting Han, Yanmin Zhou, Cheng Li, and Jing Bai. Reveal: Self-evolving code agents via reliable self-verification. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=q56ZI1Co43.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 427–431, Valencia, Spain, April 2017. Association for Computational Linguistics. doi: 10.18653/v1/E17-2068. URL https://aclanthology.org/E17-2068/.

Dongwon Jung, Peng Shi, and Yi Zhang. Futureweaver: Planning test-time compute for multi-agent systems with modularized collaboration, 2025. URL https://arxiv.org/abs/2512.11213.

Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala, and Edwin Zhang. Why language models hallucinate, 2025. URL https://arxiv.org/abs/2509.04664.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can LLMs actually correct their own mistakes? a critical survey of self-correction of LLMs. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024. doi: 10.1162/tacl_a_00713. URL https://aclanthology.org/2024.tacl-1.78/.

Zhewei Kang, Xuandong Zhao, and Dawn Song. Scalable best-of-n selection for large language models via self-certainty. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=29FRqmVQK8.

Jungo Kasai, Keisuke Sakaguchi, Yoichi Takahashi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A. Smith, Yejin Choi, and Kentaro Inui. Realtime QA: what's the answer right now? In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/9941624ef7f867a502732b5154d30cb7-Abstract-Datasets_and_Benchmarks.html.

Zixuan Ke, Austin Xu, Yifei Ming, Xuan-Phi Nguyen, Ryan Chin, Caiming Xiong, and Shafiq Joty. Mas-zero: Designing multi-agent systems with zero supervision, 2025. URL https://arxiv.org/abs/2505.14996.

Maxim Khanov, Jirayu Burapacheep, and Yixuan Li. ARGS: Alignment as reward-guided search. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=shgx0eqdw6.

Geunwoo Kim, Pierre Baldi, and Stephen Marcus McAleer. Language models can solve computer tasks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=M6OmjAZ4CX.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

Woosung Koh, Wonbeen Oh, Jaein Jang, MinHyung Lee, Hyeongjin Kim, Ah Yeon Kim, Joonkee Kim, Junghyun Lee, Taehyeon Kim, and Se-Young Yun. AdaSTar: Adaptive data sampling for training self-taught reasoners. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=D6PwC6Xogv.

Michael Krumdick, Charles Lovering, Varshini Reddy, Seth Ebner, and Chris Tanner. No free labels: Limitations of llm-as-a-judge without human grounding, 2025. URL https://arxiv.org/abs/2503.05061.

Jakub Grudzien Kuba, Mengting Gu, Qi Ma, Yuandong Tian, Vijai Mohan, and Jason Chen. Language self-play for data-free training, 2025. URL https://arxiv.org/abs/2509.07414.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning, 2024. URL https://arxiv.org/abs/2409.12917.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=CjwERcAU7w.

Po-Nien Kung, Fan Yin, Di Wu, Kai-Wei Chang, and Nanyun Peng. Active instruction tuning: Improving cross-task generalization by training on prompt sensitive tasks. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1813–1829, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.112. URL https://aclanthology.org/2023.emnlp-main.112/.

Wai-Chung Kwan, Joshua Ong Jun Leang, Pavlos Vougiouklis, Jeff Z. Pan, Marco Valentino, and Pasquale Minervini. Opensir: Open-ended self-improving reasoner, 2025. URL https://arxiv.org/abs/2511.00602.

Chris Latimer, Nicoló Boschi, Andrew Neeser, Chris Bartholomew, Gaurav Srivastava, Xuan Wang, and Naren Ramakrishnan. Hindsight is 20/20: Building agent memory that retains, recalls, and reflects, 2025. URL https://arxiv.org/abs/2512.12818.

Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, Jörg Frohberg, Mario Šaško, Quentin Lhoest, Angelina McMillan-Major, Gerard Dupont, Stella Biderman, Anna Rogers, Loubna Ben Allal, Francesco De Toni, Giada Pistilli, Olivier Nguyen, Somaieh Nikpoor, Maraim Masoud, Pierre Colombo, Javier de la Rosa, Paulo Villegas, Tristan Thrush, Shayne Longpre, Sebastian Nagel, Leon Weber, Manuel Muñoz, Jian Zhu, Daniel Van Strien, Zaid Alyafeai, Khalid Almubarak, Minh Chien Vu, Itziar Gonzalez-Dios, Aitor Soroa, Kyle Lo, Manan Dey, Pedro Ortiz Suarez, Aaron Gokaslan, Shamik Bose, David Adelani, Long Phan, Hieu Tran, Ian Yu, Suhas Pai, Jenny Chim, Violette Lepercq, Suzana Ilic, Margaret Mitchell, Sasha Alexandra Luccioni, and Yacine Jernite. The BigScience ROOTS corpus: A 1.6TB composite multilingual dataset. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=UoEw6KigkUn.

Jason Lee, Raphael Shu, and Kyunghyun Cho. Iterative refinement in the continuous space for non-autoregressive neural machine translation. In *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)*, pp. 1006–1015, 2020.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8424–8445, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.577. URL https://aclanthology.org/2022.acl-long.577/.

Ziling Lei, Yu Gu, Qifan Chen, Shulin Cao, and Deng Cai. Demystifying synthetic data in LLM pre-training: A systematic study, 2025. URL https://arxiv.org/abs/2510.01631.

Sicong Leng, Hang Zhang, Guanzheng Chen, Xin Li, Shijian Lu, Chunyan Miao, and Lidong Bing. Mitigating object hallucinations in large vision-language models through visual contrastive decoding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13872–13882, 2024.

Hui Yi Leong, Yuheng Li, Yuqing Wu, Wenwen Ouyang, Wei Zhu, Jiechao Gao, and Wei Han. AMAS: Adaptively determining communication topology for LLM-based multi-agent system. In Saloni Potdar, Lina Rojas-Barahona, and Sebastien Montella (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 2061–2070, Suzhou (China), November 2025. Association for Computational Linguistics. ISBN 979-8-89176-333-3. doi: 10.18653/v1/2025.emnlp-industry.144. URL https://aclanthology.org/2025.emnlp-industry.144/.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Boyi Li, Zhonghan Zhao, Der-Horng Lee, and Gaoang Wang. Adaptive graph pruning for multi-agent communication, 2025a. URL https://arxiv.org/abs/2506.02951.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu. From generation to judgment: Opportunities and challenges of LLM-as-a-judge. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2757–2791, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.138. URL https://aclanthology.org/2025.emnlp-main.138/.

Dawei Li, Renliang Sun, Yue Huang, Ming Zhong, Bohan Jiang, Jiawei Han, Xiangliang Zhang, Wei Wang, and huan liu. Preference leakage: A contamination problem in LLM-as-a-judge. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=grIvSXVJ65.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*, volume 36, pp. 51991–52008, 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a3621ee907def47c1b952ade25c67698-Paper-Conference.pdf.

Haohang Li, Yangyang Yu, Zhi Chen, Yuechen Jiang, Yang Li, Denghui Zhang, Rong Liu, Jordan W. Suchow, and Khaldoun Khashanah. Finmem: A performance-enhanced LLM trading agent with layered memory and character design. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024a. URL https://openreview.net/forum?id=sstfVOwbiG.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. DataComp-LM: In search of the next generation of training sets for language models. In *Thirty-eighth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024b. URL https://openreview.net/forum?id=CNWdWn47IE.

Jia Li, Ge Li, Xuanming Zhang, Yunfei Zhao, Yihong Dong, Zhi Jin, Binhua Li, Fei Huang, and Yongbin Li. Evocodebench: An evolving code generation benchmark with domain-specific evaluations. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 57619–57641. Curran Associates, Inc., 2024c. doi: 10.52202/079017-1837. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/6a059625a6027aca18302803743abaa2-Paper-Datasets_and_Benchmarks_Track.pdf.

Junkai Li, Yunghwei Lai, Weitao Li, Jingyi Ren, Meng Zhang, Xinhui Kang, Siyu Wang, Peng Li, Ya-Qin Zhang, Weizhi Ma, and Yang Liu. Agent hospital: A simulacrum of hospital with evolvable medical agents, 2025c. URL https://arxiv.org/abs/2405.02957.

Loka Li, Zhenhao Chen, Guangyi Chen, Yixuan Zhang, Yusheng Su, Eric Xing, and Kun Zhang. Confidence matters: Revisiting intrinsic self-correction capabilities of large language models, 2024d. URL https://arxiv.org/abs/2402.12563.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7602–7635, Mexico City, Mexico, June 2024e. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.421. URL https://aclanthology.org/2024.naacl-long.421/.

Shipeng Li, Shikun Li, Zhiqin Yang, Xinghua Zhang, Gaode Chen, Xiaobo Xia, Hengyu Liu, and Zhe Peng. Learnalign: Reasoning data selection for reinforcement learning in large language models based on improved gradient alignment. *CoRR*, abs/2506.11480, jun 2025d. URL https://doi.org/10.48550/arXiv.2506.11480.

Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026b. URL https://arxiv.org/abs/2507.18224.

Wanhua Li, Zibin Meng, Jiawei Zhou, Donglai Wei, Chuang Gan, and Hanspeter Pfister. Socialgpt: Prompting llms for social relation reasoning via greedy segment optimization. *Advances in Neural Information Processing Systems*, 37:2267–2291, 2024f.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35:4328–4343, 2022.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12286–12312, Toronto, Canada, July 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.687. URL https://aclanthology.org/2023.acl-long.687/.

Xiangyu Li, Yawen Zeng, Xiaofen Xing, Jin Xu, and Xiangmin Xu. QuantAgents: Towards multi-agent financial system via simulated trading. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 17438–17464, Suzhou, China, November 2025e. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.945. URL https://aclanthology.org/2025.findings-emnlp.945/.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl, 2025f. URL https://arxiv.org/abs/2503.23383.

Yafu Li, Xuyang Hu, Xiaoye Qu, Linjie Li, and Yu Cheng. Test-time preference optimization: On-the-fly alignment via iterative textual feedback. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 34630–34673. PMLR, 13–19 Jul 2025g. URL https://proceedings.mlr.press/v267/li25ac.html.

Yanhong Li, Zixuan Lan, and Jiawei Zhou. Text or pixels? evaluating efficiency and understanding of LLMs with visual text inputs. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 10564–10578, Suzhou, China, November 2025h. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.558. URL https://aclanthology.org/2025.findings-emnlp.558/.

Yanhong Li, Karen Livescu, and Jiawei Zhou. Chunk-distilled language modeling. In *The Thirteenth International Conference on Learning Representations*, 2025i.

Yanhong Li, Tianyang Xu, Kenan Tang, Karen Livescu, David McAllester, and Jiawei Zhou. Okbench: Democratizing llm evaluation with fully automated, on-demand, open knowledge benchmarking, 2025j. URL https://arxiv.org/abs/2511.08598.

Yanyang Li, Michael R. Lyu, and Liwei Wang. Learning to reason from feedback at test-time. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5241–5253, Vienna, Austria, July 2025k. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.262. URL https://aclanthology.org/2025.acl-long.262/.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *The Twelfth International Conference on Learning Representations*, 2024g. URL https://openreview.net/forum?id=ndR8Ytrzhh.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need II: phi-1.5 technical report, 2023c. URL https://arxiv.org/abs/2309.05463.

Yucheng Li, Frank Guerin, and Chenghua Lin. Latesteval: Addressing data contamination in language model evaluation through dynamic and time-sensitive test construction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):18600–18607, Mar. 2024h. doi: 10.1609/aaai.v38i17.29822. URL https://ojs.aaai.org/index.php/AAAI/article/view/29822.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*, 2024i. URL https://icml.cc/virtual/2024/poster/35153.

Yunshui Li, Binyuan Hui, Xiaobo Xia, Jiaxi Yang, Min Yang, Lei Zhang, Shuzheng Si, Ling-Hao Chen, Junhao Liu, Tongliang Liu, Fei Huang, and Yongbin Li. One-shot learning as instruction data prospector for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4586–4601, Bangkok, Thailand, August 2024j. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.252. URL https://aclanthology.org/2024.acl-long.252/.

Yunxuan Li, Yibing Du, Jiageng Zhang, Le Hou, Peter Grabowski, Yeqing Li, and Eugene Ie. Improving multi-agent debate with sparse communication topology. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7281–7294, Miami, Florida, USA, November 2024k. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.427. URL https://aclanthology.org/2024.findings-emnlp.427/.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17889–17904, Miami, Florida, USA, November

2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.992. URL https://aclanthology.org/2024.emnlp-main.992/.

Xuechen Liang, Meiling Tao, Yinghui Xia, Jianhui Wang, Kun Li, Yijin Wang, Yangfan He, Jingsong Yang, Tianyu Shi, Yuantao Wang, Miao Zhang, and Xueqian Wang. Sage: Self-evolving agents with reflective and memory-augmented abilities. *Neurocomputing*, 647:130470, 2025. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2025.130470. URL https://www.sciencedirect.com/science/article/pii/S0925231225011427.

Shalev Lifshitz, Sheila A. McIlraith, and Yilun Du. Multi-agent verification: Scaling test-time compute with goal verifiers. In *ICLR 2025 Workshop on Modularity for Collaborative, Decentralized, and Continual Deep Learning*, 2025. URL https://openreview.net/forum?id=mGAAoEWOq9.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Qian Lin, Junyi Li, and Hwee Tou Ng. Dynaquest: A dynamic question answering dataset reflecting real-world knowledge updates. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 26918–26936. Association for Computational Linguistics, 2025a. URL https://aclanthology.org/2025.findings-acl.1380/.

Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia LI, Mengzhou Xia, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-prover: A frontier model for open-source automated theorem proving. In *Second Conference on Language Modeling*, 2025b. URL https://openreview.net/forum?id=x2y9i2HDjD.

Zi Lin, Sheng Shen, Jingbo Shang, Jason E Weston, and Yixin Nie. Learning to solve and verify: A self-play framework for mutually improving code and test generation. In *NeurIPS 2025 Fourth Workshop on Deep Learning for Code*, 2025c. URL https://openreview.net/forum?id=j6tMZaPWWF.

Bijia Liu and Ronghao Dang. Finrs: A risk-sensitive trading framework for real financial markets, 2025. URL https://arxiv.org/abs/2511.12599.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+p: Empowering large language models with optimal planning proficiency, 2023. URL https://arxiv.org/abs/2304.11477.

Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. Spice: Self-play in corpus environments improves reasoning, 2025a. URL https://arxiv.org/abs/2510.24684.

Boyin Liu, Zhuo Zhang, Sen Huang, Lipeng Xie, Qingxu Fu, Haoran Chen, LI YU, Tianyi Hu, Zhaoyang Liu, Bolin Ding, and Dongbin Zhao. Taming the judge: Deconflicting ai feedback for stable reinforcement learning, 2025b. URL https://arxiv.org/abs/2510.15514.

Guangliang Liu, Haitao Mao, Bochuan Cao, Zhiyu Xue, Xitong Zhang, Rongrong Wang, Jiliang Tang, and Kristen Johnson. On the intrinsic self-correction capability of llms: Uncertainty and latent concept, 2024a. URL https://arxiv.org/abs/2406.02378.

Jia Liu, ChangYi He, YingQiao Lin, MingMin Yang, FeiYang Shen, and ShaoGuo Liu. Ettrl: Balancing exploration and exploitation in llm test-time reinforcement learning via entropy mechanism, 2025c. URL https://arxiv.org/abs/2508.11356.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. Don't throw away your value model! generating more preferable text with value-guided monte-carlo tree search decoding. In *First Conference on Language Modeling*, 2024b. URL https://openreview.net/forum?id=kh9Zt2Ldmn.

Jiahao Liu, Qifan Wang, Jingang Wang, and Xunliang Cai. Speculative decoding via early-exiting for faster LLM inference with Thompson sampling control mechanism. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3027–3043, Bangkok, Thailand, August 2024c. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.179. URL https://aclanthology.org/2024.findings-acl.179/.

Jiaqi Liu, Peng Xia, Siwei Han, Shi Qiu, Letian Zhang, Guiming Chen, Haoqin Tu, Xinyu Yang, , Jiawei Zhou, Hongtu Zhu, Yun Li, Yuyin Zhou, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Autoresearchclaw: Fully autonomous research from idea to paper, 2026a. URL https://github.com/aiming-lab/AutoResearchClaw.

Liangxin Liu, Xuebo Liu, Derek F. Wong, Dongfang Li, Ziyi Wang, Baotian Hu, and Min Zhang. SelectIT: Selective instruction tuning for LLMs via uncertainty-aware self-reflection. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024d. URL https://openreview.net/forum?id=QNieOPt4fg.

Tianlin Liu, Shangmin Guo, Leonardo Bianco, Daniele Calandriello, Quentin Berthet, Felipe Llinares-López, Jessica Hoffmann, Lucas Dixon, Michal Valko, and Mathieu Blondel. Decoding-time realignment of language models. In *Forty-first International Conference on Machine Learning*, 2024e. URL https://openreview.net/forum?id=n8g6WMxt09.

Tongxuan Liu, Xingyu Wang, Weizhe Huang, Wenjiang Xu, Yuting Zeng, Lei Jiang, Hailong Yang, and Jing Li. Groupdebate: Enhancing the efficiency of multi-agent debate using group discussion, 2024f. URL https://arxiv.org/abs/2409.14051.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *The Twelfth International Conference on Learning Representations*, 2024g. URL https://openreview.net/forum?id=BTKAeLqLMw.

Wei Liu, Siya Qi, Yali Du, and Yulan He. Self-play only evolves when self-synthetic pipeline ensures learnable information gain, 2026b. URL https://arxiv.org/abs/2603.02218.

Xiaoyuan Liu, Tian Liang, Zhiwei He, Jiahao Xu, Wenxuan Wang, Pinjia He, Zhaopeng Tu, Haitao Mi, and Dong Yu. Trust, but verify: A self-verification approach to reinforcement learning with verifiable rewards. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025d. URL https://openreview.net/forum?id=gA3fFAEXNT.

Zhixuan Liu, Zhanhui Zhou, Yuanfu Wang, Chao Yang, and Yu Qiao. Inference-time language model alignment via integrated value guidance. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 4181–4195, Miami, Florida, USA, November 2024h. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.242. URL https://aclanthology.org/2024.findings-emnlp.242/.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic LLM-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*, 2024i. URL https://openreview.net/forum?id=XII0Wp1XA9.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4969–4983, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.447. URL https://aclanthology.org/2020.acl-main.447/.

Yan Shvartzshnaider Lo, Arushi Arora, Andrew M. Saxe, and Ramakrishna Vedantam. Lessons from scaling synthetic data for trillion-scale pretraining, 2025. URL https://arxiv.org/abs/2508.10975.

Patrice Lopez. GROBID: Combining automatic bibliographic data recognition and term extraction for scholarship publications. In *Proceedings of the 13th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2009)*, pp. 473–474. Springer, September 2009. doi: 10.1007/978-3-642-04346-8_62. URL https://dl.acm.org/doi/10.5555/1812799.1812875.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, et al. StarCoder 2 and the stack v2: The next generation. *arXiv preprint arXiv:2402.19173*, 2024. URL https://arxiv.org/abs/2402.19173.

Chris Lu, Samuel Holt, Claudio Fanconi, Alex James Chan, Jakob Nicolaus Foerster, Mihaela van der Schaar, and Robert Tjarko Lange. Discovering preference optimization algorithms with and for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=erjQDJ0z9L.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery, 2024b. URL https://arxiv.org/abs/2408.06292.

Jianqiao Lu, Wanjun Zhong, Wenyong Huang, Yufei Wang, Qi Zhu, Fei Mi, Baojun Wang, Weichao Wang, Xingshan Zeng, Lifeng Shang, Xin Jiang, and Qun Liu. Self: Self-evolution with language feedback, 2024c. URL https://arxiv.org/abs/2310.00533.

Jiaxuan Lu, Ziyu Kong, Yemin Wang, Rong Fu, Haiyuan Wan, Cheng Yang, Wenjie Lou, Haoran Sun, Lilong Wang, Yankai Jiang, Xiaosong Wang, Xiao Sun, and Dongzhan Zhou. Beyond static tools: Test-time tool evolution for scientific reasoning, 2026. URL https://arxiv.org/abs/2601.07641.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1964–1974, Mexico City, Mexico, June 2024d. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.109. URL https://aclanthology.org/2024.naacl-long.109/.

Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A. Smith, and Yejin Choi. NeuroLogic a*esque decoding: Constrained text generation with lookahead heuristics. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 780–799, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.57. URL https://aclanthology.org/2022.naacl-main.57/.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 33:3776–3786, 2025. doi: 10.1109/TASLPRO.2025.3606231. URL https://ieeexplore.ieee.org/document/11151751.

Qingsong Lv, Yangning Li, Zihua Lan, Zishan Xu, Jiwei Tang, Tingwei Lu, Yinghui Li, Wenhao Jiang, Hong-Gee Kim, Hai-Tao Zheng, and Philip S. Yu. RAISE: Reinforced adaptive instruction selection for large language models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 11708–11723, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.628. URL https://aclanthology.org/2025.findings-emnlp.628/.

Weiyu Ma, Jiwen Jiang, Haobo Fu, and Haifeng Zhang. Tacticcraft: Natural language-driven tactical adaptation for starcraft ii, 2025a. URL https://arxiv.org/abs/2507.15618.

Xiaofei Ma, Rui Zhang, and Yonatan Bisk. Synthetic continued pretraining, 2025b. URL https://arxiv.org/abs/2501.12170.

Xiaowen Ma, Chenyang Lin, Yao Zhang, Volker Tresp, and Yunpu Ma. Agentic neural networks: Self-evolving multi-agent systems via textual backpropagation, 2025c. URL https://arxiv.org/abs/2506.09046.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S37hOerQLB.

Pratyush Maini, Skyler Seto, Richard Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. Rephrasing the web: A recipe for compute and data-efficient language modeling, 2024. URL https://arxiv.org/abs/2401.16380.

Shaoguang Mao, Yuzhe Cai, Yan Xia, Wenshan Wu, Xun Wang, Fengyi Wang, Qiang Guan, Tao Ge, and Furu Wei. ALYMPICS: LLM agents meet game theory. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 2845–2866, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.193/.

Durga Prasad Maram, Dhruvin Gandhi, Zonghai Yao, Gayathri Akkinapalli, Franck Dernoncourt, Yu Wang, Ryan A. Rossi, and Nesreen K. Ahmed. Iterative critique-refine framework for enhancing llm personalization, 2025. URL https://arxiv.org/abs/2510.24469.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. When less is more: Investigating data pruning for pretraining LLMs at scale. In *NeurIPS Workshop on Attributing Model Behavior at Scale*, 2023. URL https://openreview.net/forum?id=XUIYn3jo5T.

Costas Mavromatis, Petros Karypis, and George Karypis. Pack of LLMs: Model fusion at test-time via perplexity optimization. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=5Nsl0nlStc.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989. URL https://doi.org/10.1016/s0079-7421(08)60536-8.

Casey Meehan, Kamalika Chaudhuri, and Sanjoy Dasgupta. A three sample hypothesis test for evaluating generative models. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3546–3556. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/meehan20a.html.

Casey Meehan, Florian Bordes, Pascal Vincent, Kamalika Chaudhuri, and Chuan Guo. Do SSL models have déjà vu? a case of unintended memorization in self-supervised learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=lkBygTc0SI.

Dheeraj Mekala, Alex Nguyen, and Jingbo Shang. Smaller language models are capable of selecting instruction-tuning training data for larger language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10456–10470, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.623. URL https://aclanthology.org/2024.findings-acl.623/.

Microsoft Research. Phi-4 technical report, 2024. URL https://arxiv.org/abs/2412.08905.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. Prioritized training on points that are learnable, worth learning, and not yet learnt. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15630–15649. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/mindermann22a.html.

Mohammad Mahdi Moradi, Hossam Amer, Sudhir Mudur, Weiwei Zhang, Yang Liu, and Walid Ahmed. Continuous self-improvement of large language models by test-time training with verifier-driven sample selection. In *AI That Keeps Up: NeurIPS 2025 Workshop on Continual and Compatible Foundation Model Updates*, 2025. URL https://openreview.net/forum?id=6ahliSpvQ0.

Viktor Moskvoretskii, Chris Biemann, and Irina Nikishina. Self-taught self-correction for small language models, 2025. URL https://arxiv.org/abs/2503.08681.

Adam Moss. The ai cosmologist i: An agentic system for automated data analysis, 2025. URL https://arxiv.org/abs/2504.03424.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, Jilin Chen, Alex Beutel, and Ahmad Beirami. Controlled decoding from language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=bVIcZb7Qa0.

Rithesh Murthy, Ming Zhu, Liangwei Yang, Jielin Qiu, Juntao Tan, Shelby Heinecke, Caiming Xiong, Silvio Savarese, and Huan Wang. Promptomatix: An automatic prompt optimization framework for large language models, 2025. URL https://arxiv.org/abs/2507.14241.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christopher Hesse, et al. WebGPT: Browser-assisted question-answering with human feedback, 2021. URL https://arxiv.org/abs/2112.09332.

Thuat Nguyen, Chien Van Nguyen, Viet Dac Lai, Hieu Man, Nghia Trung Ngo, Franck Dernoncourt, Ryan A. Rossi, and Thien Huu Nguyen. CulturaX: A cleaned, enormous, and multilingual dataset for large language models in 167 languages. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 4226–4237, Torino, Italia, May 2024. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.377/.

Ansong Ni, Srini Iyer, Dragomir Radev, Veselin Stoyanov, Wen-Tau Yih, Sida Wang, and Xi Victoria Lin. LEVER: Learning to verify language-to-code generation with execution. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 26106–26128. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/ni23b.html.

Ziqi Ni, Yahao Li, Kaijia Hu, Kunyuan Han, Ming Xu, Xingyu Chen, Fengqi Liu, Yicong Ye, and Shuxin Bai. Matpilot: an llm-enabled ai materials scientist under the framework of human-machine collaboration, 2024. URL https://arxiv.org/abs/2411.08063.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

Xuefei Ning, Zinan Lin, Zixuan Zhou, Zifu Wang, Huazhong Yang, and Yu Wang. Skeleton-of-thought: Prompting LLMs for efficient parallel generation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mqVgBbNCm9.

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. Alphaevolve: A coding agent for scientific and algorithmic discovery, 2025a. URL https://arxiv.org/abs/2506.13131.

Alexander Novikov, Ngân V~u, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav M. Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, Matej Balog, and Google Deepmind. Alphaevolve: A coding agent for scientific and algorithmic discovery. *ArXiv*, abs/2506.13131, 2025b. URL https://api.semanticscholar.org/CorpusID:278658695.

Sean O'Brien and Mike Lewis. Contrastive decoding improves reasoning in large language models, 2023. URL https://arxiv.org/abs/2309.09117.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Yonatan Oren, Nicole Meister, Niladri S. Chatterji, Faisal Ladhak, and Tatsunori Hashimoto. Proving test set contamination in black-box language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KS8mIvetg2.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Alice H. Oh, Alekh Agarwal, Danielle

Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=TG8KACxEON.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory, 2025. URL https://arxiv.org/abs/2509.25140.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

Alexander Pan, Erik Jones, Meena Jagadeesan, and Jacob Steinhardt. Feedback loops with language models drive in-context reward hacking. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=EvHWlYTLWe.

Rui Pan, Dylan Zhang, Hanning Zhang, Xingyuan Pan, Minrui Xu, Jipeng Zhang, Renjie Pi, Xiaoyu Wang, and Tong Zhang. ScaleBiO: Scalable bilevel optimization for LLM data reweighting. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31959–31982, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1543. URL https://aclanthology.org/2025.acl-long.1543/.

Deepak Pandita, Tharindu Cyril Weerasooriya, Ankit Parag Shah, Isabelle Diana May-Xin Ng, Christopher M. Homan, and Wei Wei. Prorefine: Inference-time prompt refinement with textual feedback, 2025. URL https://arxiv.org/abs/2506.05305.

Jing-Cheng Pang, Pengyuan Wang, Kaiyuan Li, Xiong-Hui Chen, Jiacheng Xu, Zongzhang Zhang, and Yang Yu. Language model self-improvement by reinforcement learning contemplation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=38E4yUbrgr.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=4NJBV6Wp0h.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23. Association for Computing Machinery, 2023. doi: 10.1145/3586183.3606763. URL https://arxiv.org/abs/2304.03442.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. OpenWebMath: An open dataset of high-quality mathematical web text. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=jKHmjlpViu.

Liana Patel, Negar Arabzadeh, Harshit Gupta, Ankita Sundar, Ion Stoica, Matei Zaharia, and Carlos Guestrin. Deepscholar-bench: A live benchmark and automated evaluation for generative research synthesis. *CoRR*, abs/2508.20033, 2025. doi: 10.48550/ARXIV.2508.20033. URL https://doi.org/10.48550/arXiv.2508.20033.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. REFINER: Reasoning feedback on intermediate representations. In Yvette Graham and Matthew Purver (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1100–1126, St. Julian's, Malta, March 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.eacl-long.67. URL https://aclanthology.org/2024.eacl-long.67/.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The RefinedWeb dataset for

Falcon LLM: Outperforming curated corpora with web data, and web data only. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=kM5eGcdCzq.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Thirty-eighth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

XIANGYU PENG, Congying Xia, Xinyi Yang, Caiming Xiong, Chien-Sheng Wu, and Chen Xing. Regenesis: LLMs can grow into reasoning generalists via self-improvement. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=YUYJsHOf3c.

Hoang Phan, Yijun Dong, Andrew Gordon Wilson, and Qi Lei. Balanced locality-sensitive hashing for online data selection. In *OPT 2025: Optimization for Machine Learning*, 2025. URL https://openreview.net/forum?id=jsgVJtfIZV.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning, 2022. URL https://arxiv.org/abs/2202.01344.

Archiki Prasad, Weizhe Yuan, Richard Yuanzhe Pang, Jing Xu, Maryam Fazel-Zarandi, Mohit Bansal, Sainbayar Sukhbaatar, Jason E Weston, and Jane Yu. Self-consistency preference optimization. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=94G4eL3RWi.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968, 2023. URL https://aclanthology.org/2023.emnlp-main.494/.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.810. URL https://aclanthology.org/2024.acl-long.810/.

Cheng Qian, Chi Han, Yi Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. CREATOR: Tool creation for disentangling abstract and concrete reasoning of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 6922–6939, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.462. URL https://aclanthology.org/2023.findings-emnlp.462/.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs, 2025. URL https://arxiv.org/abs/2504.13958.

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

Jiahao Qiu, Xuan Qi, Hongru Wang, Xinzhe Juan, Yimin Wang, Zelin Zhao, Jiayi Geng, Jiacheng Guo, Peihang Li, Jingzhe Shi, Shilong Liu, and Mengdi Wang. Alita-g: Self-evolving generative agent for agent generation, 2025a. URL https://arxiv.org/abs/2510.23601.

Jiahao Qiu, Xuan Qi, Tongcheng Zhang, Xinzhe Juan, Jiacheng Guo, Yifu Lu, Yimin Wang, Zixin Yao, Qihan Ren, Xun Jiang, Xing Zhou, Dongrui Liu, Ling Yang, Yue Wu, Kaixuan Huang, Shilong Liu, Hongru Wang,

and Mengdi Wang. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution, 2025b. URL https://arxiv.org/abs/2505.20286.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=DRC9pZwBwR.

Mohammad Atif Quamar, Mohammad Areeb, Nishant Sharma, Ananth Shreekumar, Jonathan Rosenthal, Muslum Ozgur Ozmen, Mikhail Kuznetsov, and Z. Berkay Celik. Adaptive blockwise search: Inference-time alignment for large language models, 2025. URL https://arxiv.org/abs/2510.23334.

Evan Racah and Christopher Pal. Supervise thyself: Examining self-supervised representations in interactive environments, 2019. URL https://arxiv.org/abs/1906.11951.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL https://jmlr.org/papers/v21/20-074.html.

Keshav Ramji, Tahira Naseem, and Ramón Fernandez Astudillo. Latent principle discovery for language model self-improvement. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=T3ReIjtbYy.

Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285–308, 1990. URL https://doi.org/10.1037//0033-295x.97.2.285.

Ravin Ravi, Dylan Bradshaw, Stefano Ruberto, Gunel Jahangirova, and Valerio Terragni. Llm-loop: Improving llm-generated code and tests through automated iterative feedback loops. In *2025 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 930–934, 2025. doi: 10.1109/ICSME64153.2025.00109. URL https://valerio-terragni.github.io/assets/pdf/ravi-icsme-2025.pdf.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents, 2024. URL https://arxiv.org/abs/2405.14573.

Ali Razghandi, Seyed Mohammad Hadi Hosseini, and Mahdieh Soleymani Baghshah. CER: Confidence enhanced reasoning in LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7918–7938, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.390. URL https://aclanthology.org/2025.acl-long.390/.

Yi Ren, Shangmin Guo, Linlu Qiu, Bailin Wang, and Danica J. Sutherland. Bias amplification in language model evolution: An iterated learning perspective. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=BSYn7ah4KX.

Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995. URL https://doi.org/10.1080/09540099550039318.

Jon Saad-Falcon, E. Kelly Buchanan, Mayee F Chen, Tzu-Heng Huang, Brendan McLaughlin, Tanvir Bhathal, Shang Zhu, Ben Athiwaratkun, Frederic Sala, Scott Linderman, Azalia Mirhoseini, and Christopher Re. Weaver: Shrinking the generation-verification gap by scaling compute for verification. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=dRjt4vlYVQ.

Ali Safaya and Deniz Yuret. Neurocache: Efficient vector retrieval for long-range language modeling. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 870–883, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.50. URL https://aclanthology.org/2024.naacl-long.50/.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodola. Accelerating transformer inference for translation via parallel decoding. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12336–12355, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.689. URL https://aclanthology.org/2023.acl-long.689/.

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

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using LLM agents as research assistants. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.320. URL https://aclanthology.org/2025.findings-emnlp.320/.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 9895–9901, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.779. URL https://aclanthology.org/2021.emnlp-main.779/.

Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, and Andrea Zanette. Can large reasoning models self-train?, 2025. URL https://arxiv.org/abs/2505.21444.

Pararth Shah, Dilek Hakkani-Tür, Bing Liu, and Gokhan Tür. Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pp. 41–48, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-3006. URL https://aclanthology.org/N18-3006/.

Zhengyang Shan, Emily Diana, and Jiawei Zhou. Gender inclusivity fairness index (gifi): a multilevel framework for evaluating gender diversity in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2548–2579, 2025.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic llm agent search in modular design space. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://iclr.cc/virtual/2025/poster/32059.

Shuai Shao, Qihan Ren, Chen Qian, Boyi Wei, Dadi Guo, Yang JingYi, Xinhao Song, Linfeng Zhang, Weinan Zhang, Dongrui Liu, and Jing Shao. Your agent may misevolve: Emergent risks in self-evolving LLM agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Fd1jgQQW28.

Zhaohui Shao, Hailong Tang, Yushun Wang, and Hang Zhou. CoT-self-instruct: Building high-quality synthetic prompts for reasoning and non-reasoning tasks, 2025. URL https://arxiv.org/abs/2507.23751.

Archit Sharma, Sedrick Keh, Eric Mitchell, Chelsea Finn, Kushal Arora, and Thomas Kollar. A critical evaluation of AI feedback for aligning large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=FZQYfmsmX9.

Han Shen, Pin-Yu Chen, Payel Das, and Tianyi Chen. SEAL: Safety-enhanced aligned LLM fine-tuning via bilevel data selection. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=VHguhvcoM5.

Xintian Shen, Jiawei Chen, Lihao Zheng, Hao Ma, Tao Wei, and Kun Zhan. Evolving from tool user to creator via training-free experience reuse in multimodal reasoning, 2026. URL https://arxiv.org/abs/2602.01983.

Abhay Sheshadri, John Hughes, Julian Michael, Alex Troy Mallen, Arun Jose, and Fabien Roger. Why do some language models fake alignment while others don't? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=1Imp4KZyjA.

Ruizhe Shi, Yifang Chen, Yushi Hu, Alisa Liu, Hannaneh Hajishirzi, Noah A. Smith, and Simon Shaolei Du. Decoding-time language model alignment with multiple objectives. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=3csuL7TVpV.

Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. Efficient reinforcement finetuning via adaptive curriculum learning. *CoRR*, abs/2504.05520, apr 2025a. URL https://doi.org/10.48550/arXiv.2504.05520.

Zhengliang Shi, Lingyong Yan, Dawei Yin, Suzan Verberne, Maarten de Rijke, and Zhaochun Ren. Iterative self-incentivization empowers large language models as agentic searchers. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=s9NkfkUuEr.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Tal Shnitzer, Anthony Ou, Mirian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets. In *Annual Conference on Neural Information Processing Systems*, 2023. URL https://neurips.cc/virtual/2023/80501.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: Training on generated data makes models forget, 2024a. URL https://arxiv.org/abs/2305.17493.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. AI models collapse when trained on recursively generated data. *Nature*, 631:755–759, 2024b. doi: 10.1038/s41586-024-07566-y. URL https://www.nature.com/articles/s41586-024-07566-y.

Toby Simonds and Akira Yoshiyama. Ladder: Self-improving llms through recursive problem decomposition, 2025. URL https://arxiv.org/abs/2503.00735.

Toby Simonds, Kevin Lopez, Akira Yoshiyama, and Dominique Garmier. Rlsr: Reinforcement learning from self reward, 2025. URL https://arxiv.org/abs/2505.08827.

Joykirat Singh, Tanmoy Chakraborty, and Akshay Nambi. Self-evolved preference optimization for enhancing mathematical reasoning in small language models, 2025. URL https://arxiv.org/abs/2503.04813.

Joar Max Viktor Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=yb3HOXO3lX2.

Tyler Slater. A self-improving architecture for dynamic safety in large language models, 2025. URL https://arxiv.org/abs/2511.07645.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/1b623663fd9b874366f3ce019fdfdd44-Paper-Conference.pdf.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Evan Walsh, Luke Zettlemoyer, Noah Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: an open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15725–15788, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.840. URL https://aclanthology.org/2024.acl-long.840/.

Hwanjun Song, Minseok Kim, Sundong Kim, and Jae-Gil Lee. Carpe diem, seize the samples uncertain "at the moment" for adaptive batch selection. In Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux (eds.), *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pp. 1385–1394. ACM, 2020. doi: 10.1145/3340531.3411898. URL https://doi.org/10.1145/3340531.3411898.

Jielin Song, Siyu Liu, Bin Zhu, and Yanghui Rao. Iterselecttune: An iterative training framework for efficient instruction-tuning data selection. *CoRR*, abs/2410.13464, 2024. doi: 10.48550/ARXIV.2410.13464. URL https://doi.org/10.48550/arXiv.2410.13464.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Zhicheng Dou, and Ji-Rong Wen. EnvScaler: Scaling tool-interactive environments for LLM agent via programmatic synthesis, 2026. URL https://arxiv.org/abs/2601.05808.

Yuda Song, Hanlin Zhang, Carson Eisenach, Sham Kakade, Dean Foster, and Udaya Ghai. Mind the gap: Examining the self-improvement capabilities of large language models, 2025a. URL https://arxiv.org/abs/2412.02674.

Yuda Song, Hanlin Zhang, Carson Eisenach, Sham M. Kakade, Dean Foster, and Udaya Ghai. Mind the gap: Examining the self-improvement capabilities of large language models. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=mtJSMcF3ek.

Gaurav Srivastava, Zhenyu Bi, Meng Lu, and Xuan Wang. DEBATE, TRAIN, EVOLVE: Self-Evolution of language model reasoning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 32764–32810, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1666. URL https://aclanthology.org/2025.emnlp-main.1666/.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 3008–3021. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1f89885d556929e98d3ef9b86448f951-Paper.pdf.

Hongjin Su, Ruoxi Sun, Jinsung Yoon, Pengcheng Yin, Tao Yu, and Sercan Ö. Arık. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments, 2025a. URL https://arxiv.org/abs/2501.10893.

Liangcai Su, Zhen Zhang, Guangyu Li, Zhuo Chen, Chenxi Wang, Maojia Song, Xinyu Wang, Kuan Li, Jialong Wu, Xuanzhong Chen, Zile Qiao, Zhongwang Zhang, Huifeng Yin, Shihao Cai, Runnan Fang, Zhengwei Tao, Wenbiao Yin, Chenxiong Qian, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Scaling agents via continual pre-training, 2025b. URL https://arxiv.org/abs/2509.13310.

Yushi Su, Pengfei Shi, Shuohang Wang, and Ziniu Yao. MIND: Math informed syNthetic dialogues for pretraining LLMs, 2024. URL https://arxiv.org/abs/2410.12881.

Vighnesh Subramaniam, Yilun Du, Joshua B. Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. Multiagent finetuning: Self improvement with diverse reasoning chains. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=JtGPIZpOrz.

Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. Fast best-of-n decoding via speculative rejection, 2024a. URL https://arxiv.org/abs/2410.20290.

Hao Sun, Alihan Hüyük, and Mihaela van der Schaar. Query-dependent prompt evaluation and optimization with offline inverse rl. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=N6o0ZtPzTg.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 58202–58245. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/b5c8c1c117618267944b2617add0a766-Paper-Conference.pdf.

Wangtao Sun, Xiang Cheng, Jialin Fan, Yao Xu, Xing Yu, Shizhu He, Jun Zhao, and Kang Liu. Towards agentic self-learning LLMs in search environment, 2025a. URL https://arxiv.org/abs/2510.14253.

Yan Sun, Jia Guo, Stanley Kok, Zihao Wang, zujie wen, and Zhiqiang Zhang. Efficient reinforcement learning for large language models with intrinsic exploration. In *NeurIPS 2025 Workshop on Efficient Reasoning*, 2025b. URL https://openreview.net/forum?id=0VuuZ8sKpZ.

Yifan Sun, Yushan Liang, Zhen Zhang, Xin Liu, and Jiaye Teng. Theoretical modeling of large language model self-improvement training dynamics through solver-verifier gap. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Hh7x3c0cZl.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020. URL https://proceedings.mlr.press/v119/sun20b/sun20b.pdf.

Yutao Sun, Mingshuai Chen, Tiancheng Zhao, Ruochen Xu, Zilun Zhang, and Jianwei Yin. The self-improvement paradox: Can language models bootstrap reasoning capabilities without external scaffolding? In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6501–6512, Vienna, Austria, July 2025c. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.337. URL https://aclanthology.org/2025.findings-acl.337/.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin, Jed McCaleb, Yejin Choi, and Yu Sun. End-to-end test-time training for long context, 2025. URL https://arxiv.org/abs/2512.23675.

Jiabin Tang, Tianyu Fan, and Chao Huang. Autoagent: A fully-automated and zero-code framework for llm agents, 2025a. URL https://arxiv.org/abs/2502.05957.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases, 2023. URL https://arxiv.org/abs/2306.05301.

Xiaohang Tang, Sangwoong Yoon, Seongho Son, Huizhuo Yuan, Quanquan Gu, and Ilija Bogunovic. Rspo: Regularized self-play alignment of large language models, 2025b. URL https://arxiv.org/abs/2503.00030.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. A survey on self-evolution of large language models, 2024. URL https://arxiv.org/abs/2404.14387.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. Confidence improves self-consistency in LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20090–20111, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1030. URL https://aclanthology.org/2025.findings-acl.1030/.

Mia Taylor, James Chua, Jan Betley, Johannes Treutlein, and Owain Evans. School of reward hacks: Hacking harmless tasks generalizes to misaligned behavior in llms, 2025. URL https://arxiv.org/abs/2508.17511.

NexusAgent Team. S1-nexusagent: a self-evolving agent framework for multidisciplinary scientific research, 2026. URL https://arxiv.org/abs/2602.01550.

Avijit Thawani, Rich Caruana, Sanjoy Dasgupta, Aditya Kumar, and Alexander K. Lew. Synthetic bootstrapped pretraining, 2025. URL https://arxiv.org/abs/2509.15248.

Tristan Thrush, Christopher Potts, and Tatsunori Hashimoto. Improving pretraining data using perplexity correlations. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=huuKoVQnB0.

Tommaso Tosato, Saskia Helbling, Yorguin-Jose Mantilla-Ramos, Mahmood Hegazy, Alberto Tosato, David John Lemay, Irina Rish, and Guillaume Dumas. Persistent instability in llm's personality measurements: Effects of scale, reasoning, and conversation history, 2025. URL https://arxiv.org/abs/2508.04826.

Brandon Trabucco, Gunnar Sigurdsson, Robinson Piramuthu, and Ruslan Salakhutdinov. InSTA: Towards internet-scale training for agents, 2025. URL https://arxiv.org/abs/2502.06776.

Dhruv Trehan and Paras Chopra. Why llms aren't scientists yet: Lessons from four autonomous research attempts, 2026. URL https://arxiv.org/abs/2601.03315.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024. URL https://www.nature.com/articles/s41586-023-06747-5.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 16022–16076. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.850. URL https://doi.org/10.18653/v1/2024.acl-long.850.

Andreas Tsamados, Luciano Floridi, and Mariarosaria Taddeo. Human control of AI systems: from supervision to teaming. *AI and Ethics*, 5(2):1535–1548, 2025. ISSN 2730-5961. doi: 10.1007/s43681-024-00489-4. URL https://doi.org/10.1007/s43681-024-00489-4.

Ken Tsui. Self-correction bench: Uncovering and addressing the self-correction blind spot in large language models, 2025. URL https://arxiv.org/abs/2507.02778.

Shangqing Tu, Kejian Zhu, Yushi Bai, Zijun Yao, Lei Hou, and Juanzi Li. Dice: Detecting in-distribution contamination in llm's fine-tuning phase for math reasoning, 2024. URL https://arxiv.org/abs/2406.04197.

Ada Tur, Nicholas Meade, Xing Han Lù, Alejandra Zambrano, Arkil Patel, Esin Durmus, Spandana Gella, Karolina Stańczak, and Siva Reddy. Safearena: Evaluating the safety of autonomous web agents, 2025. URL https://arxiv.org/abs/2503.04957.

Keisuke Ueda, Wataru Hirota, Kosuke Takahashi, Takahiro Omi, Kosuke Arima, and Tatsuya Ishigaki. Exploring the design of multi-agent LLM dialogues for research ideation. In *Proceedings of the 26th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp. 322–337, Avignon, France, August 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.sigdial-1.26/.

Giorgos Vernikos, Arthur Brazinskas, Jakub Adamek, Jonathan Mallinson, Aliaksei Severyn, and Eric Malmi. Small language models improve giants by rewriting their outputs. In Yvette Graham and Matthew Purver (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2703–2718, St. Julian's, Malta, March 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.eacl-long.165. URL https://aclanthology.org/2024.eacl-long.165/.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Position: Will we run out of data? limits of LLM scaling based on human-generated data. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=ViZcgDQjyG.

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry W. Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc V. Le, and Thang Luong. Freshllms: Refreshing large language models with search engine augmentation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, Findings of ACL, pp. 13697–13720. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.813. URL https://doi.org/10.18653/v1/2024.findings-acl.813.

Manya Wadhwa, Xinyu Zhao, Junyi Jessy Li, and Greg Durrett. Learning to refine with fine-grained natural language feedback. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12281–12308, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.716. URL https://aclanthology.org/2024.findings-emnlp.716/.

Qian Wan, Wangzi Shi, Jintian Feng, Shengyingjie Liu, Luona Wei, Zhicheng Dai, and Jianwen Sun. Empowering math problem generation and reasoning for large language model via synthetic data based continual learning framework. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 23972–23991, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1223. URL https://aclanthology.org/2025.emnlp-main.1223/.

Xingchen Wan, Ruoxi Sun, Hanjun Dai, Sercan Arik, and Tomas Pfister. Better zero-shot reasoning with self-adaptive prompting. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 3493–3514, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.216. URL https://aclanthology.org/2023.findings-acl.216/.

Xingchen Wan, Ruoxi Sun, Hootan Nakhost, Hanjun Dai, Julian Eisenschlos, Sercan Arik, and Tomas Pfister. Universal self-adaptive prompting. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7437–7462, 2023b. URL https://aclanthology.org/2023.emnlp-main.461/.

Ziyu Wan, Xidong Feng, Muning Wen, Stephen Marcus Mcaleer, Ying Wen, Weinan Zhang, and Jun Wang. AlphaZero-like tree-search can guide large language model decoding and training. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 49890–49920. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/wan24c.html.

Chunqi Wang, Ji Zhang, and Haiqing Chen. Semi-autoregressive neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 479–488, 2018.

Evan Wang, Federico Cassano, Catherine Wu, Yunfeng Bai, Will Song, Vaskar Nath, Ziwen Han, Sean Hendryx, Summer Yue, and Hugh Zhang. Planning in natural language improves llm search for code generation, 2024a. URL https://arxiv.org/abs/2409.03733.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024b. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Han Wang, Haoyu Li, Brian Ko, and Huan Zhang. On the fragility of benchmark contamination detection in reasoning models. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=bhR00j6Mku.

Hongru Wang, Rui Wang, Boyang Xue, Heming Xia, Jingtao Cao, Zeming Liu, Jeff Z. Pan, and Kam-Fai Wong. Appbench: Planning of multiple apis from various apps for complex user instruction, 2024c. URL https://arxiv.org/abs/2410.19743.

Jiachen T. Wang, Tong Wu, Dawn Song, Prateek Mittal, and Ruoxi Jia. GREATS: Online selection of high-quality data for LLM training in every iteration. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024d. URL https://openreview.net/forum?id=232VcN8tSx.

Jingkang Wang, Song Wang, Yuxiang Zhu, and Jing Chen. Synthetic text generation for training large language models via gradient matching, 2025a. URL https://arxiv.org/abs/2502.17607.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library, 2026b. URL https://arxiv.org/abs/2512.17102.

Junlin Wang, Jue WANG, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=h0ZfDIrj7T.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2609–2634, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.147. URL https://aclanthology.org/2023.acl-long.147/.

Peidong Wang, Ming Wang, Zhiming Ma, Xiaocui Yang, Shi Feng, Daling Wang, Yifei Zhang, and Kaisong Song. Language models as continuous self-evolving data engineers. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 18097–18116, Suzhou, China, November 2025c. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.914. URL https://aclanthology.org/2025.emnlp-main.914/.

Qibin Wang, Pu Zhao, Shaohan Huang, Fangkai Yang, Lu Wang, Furu Wei, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Learning to refine: Self-refinement of parallel reasoning in llms, 2025d. URL https://arxiv.org/abs/2509.00084.

Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. Rethinking the bounds of LLM reasoning: Are multi-agent discussions the key? In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6106–6131, Bangkok, Thailand, August 2024e. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.331. URL https://aclanthology.org/2024.acl-long.331/.

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

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.775. URL https://aclanthology.org/2022.emnlp-main.775/.

Siyuan Wang, Zhuohan Long, Zhihao Fan, Xuanjing Huang, and Zhongyu Wei. Benchmark self-evolving: A multi-agent framework for dynamic LLM evaluation. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pp. 3310–3328. Association for Computational Linguistics, 2025f. URL https://aclanthology.org/2025.coling-main.223/.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric Xing, and Zhiting Hu. Promptagent: Strategic planning with language models enables expert-level prompt optimization. In *The Twelfth International Conference on Learning Representations*, 2024f. URL https://openreview.net/forum?id=22pyNMuIoa.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=1PL1NIMMrw.

Yingxu Wang, Siwei Liu, Jinyuan Fang, and Zaiqiao Meng. Evoagentx: An automated framework for evolving agentic workflows. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 643–655, 2025g. URL https://aclanthology.org/2025.emnlp-demos.47/.

Yinjie Wang, Ling Yang, Guohao Li, Mengdi Wang, and Bryon Aragam. Scoreflow: Mastering llm agent workflows via score-based preference optimization, 2025h. URL https://arxiv.org/abs/2502.04306.

Yinjie Wang, Ling Yang, Ye Tian, Ke Shen, and Mengdi Wang. CURE: Co-evolving coders and unit testers via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025i. URL https://openreview.net/forum?id=wPdBe9zxNr.

Yixu Wang, Yan Teng, Kexin Huang, Chengqi Lyu, Songyang Zhang, Wenwei Zhang, Xingjun Ma, Yu-Gang Jiang, Yu Qiao, and Yingchun Wang. Fake alignment: Are LLMs really aligned well? In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4696–4712, Mexico City, Mexico, June 2024g. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.263. URL https://aclanthology.org/2024.naacl-long.263/.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13484–13508, Toronto, Canada, July 2023c. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.754. URL https://aclanthology.org/2023.acl-long.754/.

Yu Wang and Xi Chen. Mirix: Multi-agent memory system for llm-based agents, 2025. URL https://arxiv.org/abs/2507.07957.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian McAuley. MEMORYLLM: Towards self-updatable large language models. In *Forty-first International Conference on Machine Learning*, 2024h. URL https://openreview.net/forum?id=p0lKWzdikQ.

Ze Wang, Zekun Wu, Yichi Zhang, Xin Guan, Navya Jain, Qinyang Lu, Saloni Gupta, and Adriano Koshiyama. Bias amplification: Large language models as increasingly biased media. In Kentaro Inui, Sakriani Sakti, Haofen Wang, Derek F. Wong, Pushpak Bhattacharyya, Biplab Banerjee, Asif Ekbal, Tanmoy Chakraborty, and Dhirendra Pratap Singh (eds.), *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 115–132, Mumbai, India, December 2025j. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics. ISBN 979-8-89176-298-5. URL https://aclanthology.org/2025.ijcnlp-long.8/.

Zeyu Wang, Siyuan Wang, Kaiyuan Deng, Feng Sheng, Zhuofu Lin, and Zeyu Liu. Scaling laws of synthetic data for language models, 2025k. URL https://arxiv.org/abs/2503.19551.

Zhenting Wang, Guofeng Cui, Yu-Jhe Li, Kun Wan, and Wentian Zhao. Dump: Automated distribution-level curriculum learning for rl-based llm post-training. *CoRR*, abs/2504.09710, April 2025l. URL https://doi.org/10.48550/arXiv.2504.09710.

Zifeng Wang, Chun-Liang Li, Vincent Perot, Long T. Le, Jindong Miculivicius, Tomas Guo, Chen Fan, Jonas Pfeiffer, Zaid Alyafeai, Xuezhi Dong, et al. CodecLM: Aligning language models with tailored synthetic data. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 3712–3729, Mexico City, Mexico, June 2024i. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-naacl.235/.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian (Shawn) Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with llms enables open-world multi-task agents. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34153–34189. Curran Associates, Inc., 2023d. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6b8dfb8c0c12e6fafc6c256cb08a5ca7-Paper-Conference.pdf.

Koki Wataoka, Tsubasa Takahashi, and Ryokan Ri. Self-preference bias in LLM-as-a-judge. In *Neurips Safe Generative AI Workshop 2024*, 2024. URL https://openreview.net/forum?id=tLZZZIgPJX.

Maurice Weber, Daniel Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. RedPajama: an open dataset for training large language models. In *Thirty-eighth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=lnuXaRpwvw.

Haoran Wei, Yaofeng Sun, and Yukun Li. Deepseek-ocr: Contexts optical compression. *arXiv preprint arXiv:2510.18234*, 2025a.

Yuxiang Wei, Zhiqing Sun, Emily McMilin, Jonas Gehring, David Zhang, Gabriel Synnaeve, Daniel Fried, Lingming Zhang, and Sida Wang. Toward training superintelligent software agents through self-play swe-rl, 2025b. URL https://arxiv.org/abs/2512.18552.

Sean Welleck, Jiacheng Liu, Ximing Lu, Hannaneh Hajishirzi, and Yejin Choi. Naturalprover: Grounded mathematical proof generation with language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=rhdfTOiXBng.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. CCNet: Extracting high quality monolingual datasets from web crawl data. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis (eds.), *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 4003–4012, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.494/.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha V. Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. Livebench: A challenging, contamination-limited LLM benchmark. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=sKYHBTAxVa.

Yotam Wolf, Noam Wies, Oshri Avnery, Yoav Levine, and Amnon Shashua. Fundamental limitations of alignment in large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024. URL https://proceedings.mlr.press/v235/wolf24a.html.

Mengsong Wu, Tong Zhu, Han Han, Xiang Zhang, Wenbiao Shao, and Wenliang Chen. Chain-of-tools: Utilizing massive unseen tools in the cot reasoning of frozen language models, 2025a. URL https://arxiv.org/abs/2503.16779.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024a. URL https://openreview.net/forum?id=BAakY1hNKS.

Qinzhuo Wu, Wei Liu, Jian Luan, and Bin Wang. ToolPlanner: A tool augmented LLM for multi granularity instructions with path planning and feedback. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18315–18339, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1018. URL https://aclanthology.org/2024.emnlp-main.1018/.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and Botian Shi. EvolveR: Self-evolving LLM agents through an experience-driven lifecycle, 2025b. URL https://arxiv.org/abs/2510.16079.

Shengguang Wu, Keming Lu, Benfeng Xu, Junyang Lin, Qi Su, and Chang Zhou. Self-evolved diverse data sampling for efficient instruction tuning. *CoRR*, abs/2311.08182, 2023. URL https://doi.org/10.48550/arXiv.2311.08182.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason E Weston, and Sainbayar Sukhbaatar. Meta-rewarding language models: Self-improving alignment with LLM-as-a-meta-judge. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 11537–11554, Suzhou, China, November 2025c. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.583. URL https://aclanthology.org/2025.emnlp-main.583/.

Ting Wu, Xuefeng Li, and Pengfei Liu. Progress or regress? self-improvement reversal in post-training. In *The Thirteenth International Conference on Learning Representations*, 2025d. URL https://openreview.net/forum?id=RFqeoVfLHa.

Xiaobao Wu, Liangming Pan, Yuxi Xie, Ruiwen Zhou, Shuai Zhao, Yubo Ma, Mingzhe Du, Rui Mao, Anh Tuan Luu, and William Yang Wang. Antileakbench: Preventing data contamination by automatically constructing benchmarks with updated real-world knowledge. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association*

for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025, pp. 18403–18419. Association for Computational Linguistics, 2025e. URL https://aclanthology.org/2025.acl-long.901/.

Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. From human memory to ai memory: A survey on memory mechanisms in the era of llms, 2025f. URL https://arxiv.org/abs/2504.15965.

Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. Self-play preference optimization for language model alignment. In *The Thirteenth International Conference on Learning Representations*, 2025g. URL https://openreview.net/forum?id=a3PmRgAB5T.

Sierra Wyllie, Ilia Shumailov, and Nicolas Papernot. Fairness feedback loops: Training on synthetic data amplifies bias. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '24, pp. 2113–2147, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704505. doi: 10.1145/3630106.3659029. URL https://doi.org/10.1145/3630106.3659029.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. Agentgym: Evolving large language model-based agents across diverse environments, 2024a.

Zhiheng Xi, Dingwen Yang, Jixuan Huang, Jiafu Tang, Guanyu Li, Yiwen Ding, Wei He, Boyang Hong, Shihan Do, Wenyu Zhan, Xiao Wang, Rui Zheng, Tao Ji, Xiaowei Shi, Yitao Zhai, Rongxiang Weng, Jingang Wang, Xunliang Cai, Tao Gui, Zuxuan Wu, Qi Zhang, Xipeng Qiu, Xuanjing Huang, and Yu-Gang Jiang. Enhancing llm reasoning via critique models with test-time and training-time supervision, 2024b. URL https://arxiv.org/abs/2411.16579.

Chengxuan Xia, Qianye Wu, Sixuan Tian, and Yilun Hao. Parallelism meets adaptiveness: Scalable documents understanding in multi-agent llm systems, 2025. URL https://arxiv.org/abs/2507.17061.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3909–3925, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.257. URL https://aclanthology.org/2023.findings-emnlp.257/.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7655–7671, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.456. URL https://aclanthology.org/2024.findings-acl.456/.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://proceedings.mlr.press/v235/xia24c.html.

Quan Xiao and Tianyi Chen. A unified understanding of offline data selection and online self-refining generation for post-training llms, 2025. URL https://arxiv.org/abs/2511.21056.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023a. URL http://papers.nips.cc/paper_files/paper/2023/hash/dcba6be91359358c2355cd920da3fcbd-Abstract-Conference.html.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. Data selection for language models via importance resampling. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023b. URL http://papers.nips.cc/paper_files/paper/2023/hash/6b9aa8f418bde2840d5f4ab7a02f663b-Abstract-Conference.html.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. Self-evaluation guided beam search for reasoning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023c. URL https://openreview.net/forum?id=Bw82hwg5Q3.

Wei Xiong, Hanning Zhang, Chenlu Ye, Lichang Chen, Nan Jiang, and Tong Zhang. Self-rewarding correction for mathematical reasoning, 2025. URL https://arxiv.org/abs/2502.19613.

Yiming Xiong, Shengran Hu, and Jeff Clune. Learning to continually learn via meta-learning agentic memory designs. In *ICLR 2026 Workshop on Memory for LLM-Based Agentic Systems*, 2026. URL https://openreview.net/forum?id=PRkA1cwXC2.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=CfXh93NDgH.

Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 6268–6278, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.385. URL https://aclanthology.org/2023.emnlp-main.385/.

Ran Xu, Hui Liu, Sreyashi Nag, Zhenwei Dai, Yaochen Xie, Xianfeng Tang, Chen Luo, Yang Li, Joyce C. Ho, Carl Yang, and Qi He. SimRAG: Self-improving retrieval-augmented generation for adapting large language models to specialized domains. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 11534–11550, Albuquerque, New Mexico, April 2025a. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.575. URL https://aclanthology.org/2025.naacl-long.575/.

Wenda Xu, Guanglei Zhu, Xuandong Zhao, Liangming Pan, Lei Li, and William Wang. Pride and prejudice: LLM amplifies self-bias in self-refinement. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15474–15492, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.826. URL https://aclanthology.org/2024.acl-long.826/.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=FiM0M8gcct.

Yuancheng Xu, Udari Madhushani Sehwag, Alec Koppel, Sicheng Zhu, Bang An, Furong Huang, and Sumitra Ganesh. Genarm: Reward guided generation with autoregressive reward model for test-time alignment. In *The Thirteenth International Conference on Learning Representations*, 2025c. URL https://arxiv.org/abs/2410.08193.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search, 2025. URL https://arxiv.org/abs/2504.08066.

Roman V. Yampolskiy. On controllability of ai, 2020. URL https://arxiv.org/abs/2008.04071.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning, 2026. URL https://arxiv.org/abs/2508.19828.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=Bb4VGOWELI.

Haoyan Yang, Khiem Le, Ting Hua, Shangqian Gao, Binfeng Xu, Zheng Tang, Jie Xu, Nitesh V. Chawla, Hongxia Jin, and Vijay Srinivasan. Dynamic noise preference optimization: Self-improvement of large language models with self-synthetic data, 2026. URL https://arxiv.org/abs/2502.05400.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. Gpt4tools: Teaching large language model to use tools via self-instruction. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 71995–72007. Curran Associates, Inc., 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/e393677793767624f2821cec8bdd02f1-Paper-Conference.pdf.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E. Gonzalez, and Ion Stoica. Rethinking benchmark and contamination for language models with rephrased samples, 2023b. URL https://arxiv.org/abs/2311.04850.

Yingxuan Yang, Huayi Wang, Muning Wen, Xiaoyun Mo, Qiuying Peng, Jun Wang, and Weinan Zhang. P3: A policy-driven, pace-adaptive, and diversity-promoted framework for data pruning in llm training. *CoRR*, abs/2408.05541, 2024b. URL https://doi.org/10.48550/arXiv.2408.05541.

Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. Agentnet: Decentralized evolutionary coordination for LLM-based multi-agent systems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=tXqLxHlb8Z.

Yu Yang, Siddhartha Mishra, Jeffrey N Chiang, and Baharan Mirzasoleiman. Smalltolarge (s2l): Scalable data selection for fine-tuning large language models by summarizing training trajectories of small models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024c. URL https://openreview.net/forum?id=K9IGlMQpif.

Zejun Yang, Li Ning, Haitao Wang, Tianyu Jiang, Shaolin Zhang, Shaowei Cui, Hao Jiang, Chunpeng Li, Shuo Wang, and Zhaoqi Wang. Text2reaction : Enabling reactive task planning using large language models. *IEEE Robotics and Automation Letters*, 9(5):4003–4010, 2024d. doi: 10.1109/LRA.2024.3371223.

Yansi Li, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Qiuzhi Liu, Rui Wang, Zhuosheng Zhang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Dancing with critiques: Enhancing llm reasoning with step-wise natural language self-critique, 2025. URL https://www.researchgate.net/doi/10.13140/RG.2.2.27912.33289.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents, 2023a. URL https://arxiv.org/abs/2207.01206.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023b. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023c.

Tongao Yao, Yang Yang, Yujie Yan, Xinyi Ou, Mingyang Li, Chenxi Wang, Wuzhe Li, Chenghao Du, Xuqiang Shao, Zhengyang Gao, et al. Knowledge-extractor: a self-evolving scientific framework for hydrogen energy research driven by ai agents. *AI Agent*, 1(1):N–A, 2025.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh R N, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran Xu, Phil L Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. Retroformer: Retrospective large language agents with policy gradient optimization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KOZu91CzbK.

Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, Nitesh V Chawla, and Xiangliang Zhang. Justice or prejudice? quantifying biases in LLM-as-a-judge. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=3GTtZFiajM.

Rui Ye, Shuo Tang, Rui Ge, Yaxin Du, Zhenfei Yin, Siheng Chen, and Jing Shao. MAS-GPT: Training LLMs to build LLM-based multi-agent systems. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=3CiSpY3QdZ.

Li Yin and Zhangyang Wang. Llm-autodiff: Auto-differentiate any llm workflow, 2025. URL https://arxiv.org/abs/2501.16673.

Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel agent: A self-referential agent framework for recursively self-improvement. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27890–27913, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1354. URL https://aclanthology.org/2025.acl-long.1354/.

Cunxi Yu, Rongjian Liang, Chia-Tung Ho, and Haoxing Ren. Autonomous code evolution meets np-completeness, 2025a. URL https://arxiv.org/abs/2509.07367.

Dian Yu, Baolin Peng, Ye Tian, Linfeng Song, Haitao Mi, and Dong Yu. Siam: Self-improving code-assisted mathematical reasoning of large language models, 2024a. URL https://arxiv.org/abs/2408.15565.

Fei Xu Yu, Gina Adam, Nathaniel D. Bastian, and Tian Lan. Optimizing prompt sequences using monte carlo tree search for llm-based optimization, 2025b. URL https://arxiv.org/abs/2508.05995.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. Memagent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=k5nIOvYGCL.

Jifan Yu, Xiaozhi Wang, Shangqing Tu, Shulin Cao, Daniel Zhang-Li, Xin Lv, Hao Peng, Zijun Yao, Xiaohan Zhang, Hanming Li, Chunyang Li, Zheyuan Zhang, Yushi Bai, Yantao Liu, Amy Xin, Nianyi Lin, Kaifeng Yun, Linlu Gong, Jianhui Chen, Zhili Wu, Yunjia Qi, Weikai Li, Yong Guan, Kaisheng Zeng, Ji Qi, Hailong Jin, Jinxin Liu, Yu Gu, Yuan Yao, Ning Ding, Lei Hou, Zhiyuan Liu, Bin Xu, Jie Tang, and Juanzi Li. Kola: Carefully benchmarking world knowledge of large language models, 2024b. URL https://arxiv.org/abs/2306.09296.

Junwei Yu, Yepeng Ding, and Hiroyuki Sato. Dyntaskmas: a dynamic task graph-driven framework for asynchronous and parallel llm-based multi-agent systems. In *Proceedings of the Thirty-Fifth International Conference on Automated Planning and Scheduling*, ICAPS '25. AAAI Press, 2025c. ISBN 1-57735-903-8. doi: 10.1609/icaps.v35i1.36130. URL https://doi.org/10.1609/icaps.v35i1.36130.

Shi Yu, Zhiyuan Liu, and Chenyan Xiong. Craw4LLM: Efficient web crawling for LLM pretraining. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 12123–12137, Vienna, Austria, July 2025d. Association for Computational Linguistics. URL https://aclanthology.org/2025.findings-acl.712/.

Tao Yu, Zhengbo Zhang, Zhiheng Lyu, Junhao Gong, Hongzhu Yi, Xinming Wang, Yuxuan Zhou, Jiabing Yang, Ping Nie, Yan Huang, and Wenhu Chen. BrowserAgent: Building web agents with human-inspired web browsing actions, 2025e. URL https://arxiv.org/abs/2510.10666.

Xiao Yu, Baolin Peng, Michel Galley, Jianfeng Gao, and Zhou Yu. Teaching language models to self-improve through interactive demonstrations. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5127–5149, Mexico City, Mexico, June 2024c. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.287. URL https://aclanthology.org/2024.naacl-long.287/.

Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. Large language model as attributed training data generator: A tale of diversity and bias. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=ae9500c4f5607caf2eff033c67daa9d7.

ZHAONING YU, Zhaolun Su, Leitian Tao, Haozhu Wang, Aashu Singh, Hanchao Yu, Jianyu Wang, Hongyang Gao, Weizhe Yuan, Jason E Weston, Ping Yu, and Jing Xu. RESTRAIN: From spurious votes to signals — self-training RL with self-penalization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=87ySF7viys.

Zichun Yu, Spandan Das, and Chenyan Xiong. MATES: Model-aware data selection for efficient pretraining with data influence models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024d. URL https://openreview.net/forum?id=6gzPSMUAz2.

Zichun Yu, Fei Peng, Jie Lei, Arnold Overwijk, Wen tau Yih, and Chenyan Xiong. Group-level data selection for efficient pretraining. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025f. URL https://openreview.net/forum?id=uX4dyc7Z5Z.

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

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deqing Yang. EvoAgent: Towards automatic multi-agent generation via evolutionary algorithms. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6192–6217, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.315. URL https://aclanthology.org/2025.naacl-long.315/.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=0NphYCmgua.

Murong Yue, Zhiwei Liu, Liangwei Yang, Jianguo Zhang, Zuxin Liu, Haolin Chen, Ziyu Yao, Silvio Savarese, Caiming Xiong, Shelby Heinecke, and Huan Wang. Toollibgen: Scalable automatic tool creation and aggregation for llm reasoning, 2025. URL https://arxiv.org/abs/2510.07768.

Zhenrui Yue, Kartikeya Upasani, Xianjun Yang, Suyu Ge, Shaoliang Nie, Yuning Mao, Zhe Liu, and Dong Wang. Dr. zero: Self-evolving search agents without training data, 2026. URL https://arxiv.org/abs/2601.07055.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639:609–616, 2025. URL https://www.nature.com/articles/s41586-025-08661-4.

Taeyoung Yun, Pierre-Luc St-Charles, Jinkyoo Park, Yoshua Bengio, and Minsu Kim. Active attacks: Red-teaming llms via adaptive environments. *CoRR*, abs/2509.21947, 2025. doi: 10.48550/ARXIV.2509.21947. URL https://doi.org/10.48550/arXiv.2509.21947.

Abhay Zala, Jaemin Cho, Han Lin, Jaehong Yoon, and Mohit Bansal. Envgen: Generating and adapting environments via llms for training embodied agents. *CoRR*, abs/2403.12014, 2024. doi: 10.48550/ARXIV.2403.12014. URL https://doi.org/10.48550/arXiv.2403.12014.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STar: Bootstrapping reasoning with reasoning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_3ELRdg2sgI.

Eric Zelikman, Georges Raif Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah Goodman. Quiet-STar: Language models can teach themselves to think before speaking. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=oRXPiSOGH9.

Jiahao Zeng, Hao Leung, and Pengfei Liu. On the diversity of synthetic data and its impact on training large language models, 2024. URL https://arxiv.org/abs/2410.15226.

Yongcheng Zeng, Xinyu Cui, Xuanfa Jin, Qirui Mi, Guoqing Liu, Zexu Sun, Mengyue Yang, Dong Li, Weiyu Ma, Ning Yang, Jian Zhao, Jianye Hao, Haifeng Zhang, and Jun Wang. Evolving llms' self-refinement capability via synergistic training-inference optimization, 2025. URL https://arxiv.org/abs/2502.05605.

Yuexiang Zhai, Shengbang Tong, Xiao Li, Mu Cai, Qing Qu, Yong Jae Lee, and Yi Ma. Investigating the catastrophic forgetting in multimodal large language model fine-tuning. In *Conference on Parsimony and Learning (Proceedings Track)*, 2023. URL https://openreview.net/forum?id=g7rMSiNtmA.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, Zhaoyang Liu, Bolin Ding, and Jingren Zhou. Agentevolver: Towards efficient self-evolving agent system, 2025. URL https://arxiv.org/abs/2511.10395.

Bohan Zhang, Xiaokang Zhang, Jing Zhang, Jifan Yu, Sijia Luo, and Jie Tang. CoT-based synthesizer: Enhancing LLM performance through answer synthesis. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6286–6303, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.315. URL https://aclanthology.org/2025.acl-long.315/.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772, 2024a. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/76ec4dc30e9faaf0e4b6093eaa377218-Paper-Conference.pdf.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b, 2024b. URL https://arxiv.org/abs/2406.07394.

Guibin Zhang, Muxin Fu, Kun Wang, Guancheng Wan, Miao Yu, and Shuicheng YAN. G-memory: Tracing hierarchical memory for multi-agent systems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=mmIAp3cVS0.

Haozhen Zhang, Tao Feng, Pengrui Han, and Jiaxuan You. Academiceval: Live long-context LLM benchmark. *Trans. Mach. Learn. Res.*, 2025, 2025c. URL https://openreview.net/forum?id=LjQ4voE5bs.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=pUpzQZTvGY.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*, 2025d. URL https://openreview.net/forum?id=z5uVAKwmjf.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & verify: Lossless large language model acceleration via self-speculative decoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11263–11282, Bangkok, Thailand, August 2024c. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.607. URL https://aclanthology.org/2024.acl-long.607/.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Learning comprehensive contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=eC4ygDs02R.

Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B Tenenbaum, and Chuang Gan. Planning with large language models for code generation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Lr8cOOtYbfL.

Wentao Zhang, Lingxuan Zhao, Haochong Xia, Shuo Sun, Jiaze Sun, Molei Qin, Xinyi Li, Yuqing Zhao, Yilei Zhao, Xinyu Cai, et al. A multimodal foundation agent for financial trading: Tool-augmented, diversified, and generalist. In *Proceedings of the 30th acm sigkdd conference on knowledge discovery and data mining*, pp. 4314–4325, 2024d. URL https://dl.acm.org/doi/10.1145/3637528.3671801.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=neZSGqhxDa.

Tianjiao Zhao, Jingrao Lyu, Stokes Jones, Harrison Garber, Stefano Pasquali, and Dhagash Mehta. Alphaagents: Large language model based multi-agents for equity portfolio constructions, 2025b. URL https://arxiv.org/abs/2508.11152.

Wanjia Zhao, Mert Yuksekgonul, Shirley Wu, and James Zou. Sirius: Self-improving multi-agent systems via bootstrapped reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025c. URL https://openreview.net/forum?id=IDSTtDw4Cs.

Zirui Zhao, Hanze Dong, Amrita Saha, Caiming Xiong, and Doyen Sahoo. Automatic curriculum expert iteration for reliable LLM reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025d. URL https://openreview.net/forum?id=3ogIALgghF.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=uccHPGDlao.

Li Zhong, Zilong Wang, and Jingbo Shang. Debug like a human: A large language model debugger via verifying runtime execution step by step. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 851–870, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.49. URL https://aclanthology.org/2024.findings-acl.49/.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17): 19724–19731, Mar. 2024b. doi: 10.1609/aaai.v38i17.29946. URL https://ojs.aaai.org/index.php/AAAI/article/view/29946.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=WZH7099tgfM.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan O Arik. Multi-agent design: Optimizing agents with better prompts and topologies. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=IO5H9RUzHB.

Jiawei Zhou and Phillip Keung. Improving non-autoregressive neural machine translation with monolingual data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1893–1898, 2020.

Jiawei Zhou and Alexander M Rush. Simple unsupervised summarization by contextual matching. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5101–5106, 2019.

Jiawei Zhou, Tahira Naseem, Ramón Fernandez Astudillo, and Radu Florian. Amr parsing with action-pointer transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5585–5598, 2021a.

Jiawei Zhou, Tahira Naseem, Ramón Fernandez Astudillo, Young-Suk Lee, Radu Florian, and Salim Roukos. Structure-aware fine-tuning of sequence-to-sequence transformers for transition-based amr parsing. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 6279–6290, 2021b.

Jiawei Zhou, Jason Eisner, Michael Newman, Emmanouil Antonios Platanios, and Sam Thomson. Online semantic parsing for latency reduction in task-oriented dialogue. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1554–1576, 2022.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023b. URL https://webarena.dev.

Wei Zhou, Jun Zhou, Haoyu Wang, Zhenghao Li, Qikang He, Shaokun Han, Guoliang Li, Xuanhe Zhou, Yeye He, Chunwei Liu, Zirui Tang, Bin Wang, Shen Tang, Kai Zuo, Yuyu Luo, Zhenzhe Zheng, Conghui He, Jingren Zhou, and Fan Wu. Can LLMs clean up your mess? a survey of application-ready data preparation with LLMs, 2026b. URL https://arxiv.org/abs/2601.17058.

Yefan Zhou, Austin Xu, Yilun Zhou, Janvijay Singh, Jiang Gui, and Shafiq Joty. Variation in verification: Understanding verification dynamics in large language models. In *The Fourteenth International Conference on Learning Representations*, 2026c. URL https://openreview.net/forum?id=DcEuBwrWnB.

Yifei Zhou, Sergey Levine, Jason E Weston, Xian Li, and Sainbayar Sukhbaatar. Self-challenging language model agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=9yusqX9DpR.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*, 2023c. URL https://openreview.net/forum?id=92gvk82DE-.

Yuechi Zhou, Chuyue Zhou, Jianxin Zhang, Juntao Li, and Min Zhang. ALW: Adaptive layer-wise contrastive decoding enhancing reasoning ability in large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 8506–8524, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.447. URL https://aclanthology.org/2025.findings-acl.447/.

Yujun Zhou, Han Bao, Yue Huang, Kehan Guo, Zhenwen Liang, Pin-Yu Chen, Tian Gao, Werner Geyer, Nuno Moniz, Nitesh V Chawla, and Xiangliang Zhang. Emergent deceptive behaviors in reward-optimizing LLMs. In *Socially Responsible and Trustworthy Foundation Models at NeurIPS 2025*, 2025c. URL https://openreview.net/forum?id=g0rlV12Opz.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *The Fourteenth International Conference on Learning Representations*, 2026d. URL https://openreview.net/forum?id=XY8AaxDSLb.

Tinghui Zhu, Kai Zhang, Jian Xie, and Yu Su. Deductive beam search: Decoding deducible rationale for chain-of-thought reasoning, 2024. URL https://openreview.net/forum?id=S1XnUsqwr7.

Wenhong Zhu, Hongkun Hao, and Rui Wang. Penalty decoding: Well suppress the self-reinforcement effect in open-ended text generation. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1218–1228, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.78. URL https://aclanthology.org/2023.emnlp-main.78/.

Wenhong Zhu, Ruobing Xie, Weinan Zhang, and Rui Wang. Flexible realignment of language models, 2026. URL https://arxiv.org/abs/2506.12704.

Yinghao Zhu, Yifan Qi, Zixiang Wang, Lei Gu, Dehao Sui, Haoran Hu, Xichen Zhang, Ziyi He, Junjun He, Liantao Ma, and Lequan Yu. Healthflow: A self-evolving ai agent with meta planning for autonomous healthcare research, 2025a. URL https://arxiv.org/abs/2508.02621.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 19–27, December 2015. doi: 10.1109/ICCV.2015.11. URL https://arxiv.org/abs/1506.06724.

Yuqi Zhu, Ge Li, Xue Jiang, Jia Li, Hong Mei, Zhi Jin, Yihong Dong, and Qibin Zheng. Uncert-cot: Uncertainty-aware chain-of-thought for code generation with large language model. In *Advanced Intelligent Computing Technology and Applications: 21st International Conference, ICIC 2025, Ningbo, China, July 26–29, 2025, Proceedings, Part XXIII*, pp. 426–437, Berlin, Heidelberg, 2025b. Springer-Verlag. ISBN 978-981-95-0013-0. doi: 10.1007/978-981-95-0014-7_36. URL https://doi.org/10.1007/978-981-95-0014-7_36.

Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor Bursztyn, Ryan A. Rossi, Somdeb Sarkhel, and Chao Zhang. Toolchain*: Efficient action space navigation in large language models with a* search. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=B6pQxqUcT8.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62743–62767. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhuge24a.html.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: Test-time reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=VuVhgEiu20.

Adam Zweiger, Jyothish Pari, Han Guo, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=JsNUE84Hxi.