Supervised Fine-tuning versus Reinforcement Learning: A Comprehensive Survey on Large Language Model Post-training

Anonymous ACL submission

Abstract

Pre-trained Large Language Model (LLM) exhibits broad capabilities, yet, for specific tasks or domains their attainment of higher accuracy and more reliable reasoning generally depends on post-training through Supervised Fine-Tuning (SFT) or Reinforcement Learning (RL). Although often treated as distinct methodologies, recent theoretical and empirical developments demonstrate that SFT and RL are closely connected. This survey presents a comprehensive and unified perspective on LLM post-training with SFT and RL. We first provide an in-depth overview of both techniques, examining their objectives, algorithmic structures, and data requirements. We then systematically analyze their interplay, highlighting frameworks that integrate SFT and RL, hybrid training pipelines, and methods that leverage their complementary strengths. Drawing on a representative set of recent application studies from 2023 to 2025, we identify emerging trends, characterize the rapid shift toward hybrid post-training paradigms, and distill key takeaways that clarify when and why each method is most effective. By synthesizing theoretical insights, practical methodologies, and empirical evidence, this survey establishes a coherent understanding of SFT and RL within a unified framework and outlines promising directions for future research in scalable, efficient, and generalizable LLM post-training.

1 Introduction

Pre-trained LLMs have demonstrated remarkable capabilities across a wide range of tasks, from fact-based question answering (Joshi et al., 2017) to code generation (Jimenez et al., 2024). Despite being trained on corpora containing billions to trillions of tokens, LLMs often require task-specific post-training adaptation to improve accuracy, mitigate erroneous outputs, and handle new tasks. For instance, fine-tuning can enhance multi-step reasoning by enabling the generation of progressively longer reasoning chains, ultimately leading to more accurate final answers, particularly in tasks that require complex reasoning ability(Guo et al., 2025). It can also enhance practical interaction skills, including performing tasks in household (Song et al., 2024a) or device-control environments (Rawles et al., 2025). Such capabilities rarely emerge from pre-training alone, as the data seldom contain the task-specific patterns or feedback necessary for accurate reasoning or complex interactions, highlighting the importance of post-training.

Current post-training methodologies for LLMs primarily fall into two paradigms: SFT and RL. The objective of SFT (Zhou et al., 2023; Li et al., 2024c; Pang et al., 2025) is to maximize the likelihood of tokens conditioned on context, whereas RL (Ouyang et al., 2022; Guo et al., 2025; Yang et al., 2025a) optimizes a reward signal derived from human or automated preference feedback. Despite their different objectives, in recent years, research efforts have increasingly focused on bridging these two approaches (Wu et al., 2025; Fu et al., 2025b), exploring how their combination (Wu et al., 2025; Qin and Springenberg, 2025; Yan et al., 2025a; Liu et al., 2025a) can enhance performance beyond what either approach achieves in isolation.

This combination is especially pronounced in complex tasks that demand both accuracy and generalization, such as multi-step reasoning. SFT alone can teach models to generate basic Chain-of-Thought (CoT), but may struggle with novel problem structures (Ross and Bagnell, 2010; De Haan et al., 2019). Conversely, RL fine-tuning based on preference feedback can improve step-wise correctness, yet it often requires extensive exploration in the absence of offline demonstrations. By combining SFT and RL, models can leverage the strengths of both approaches, achieving more reliable and robust reasonings. As illustrated in Figure 1, these

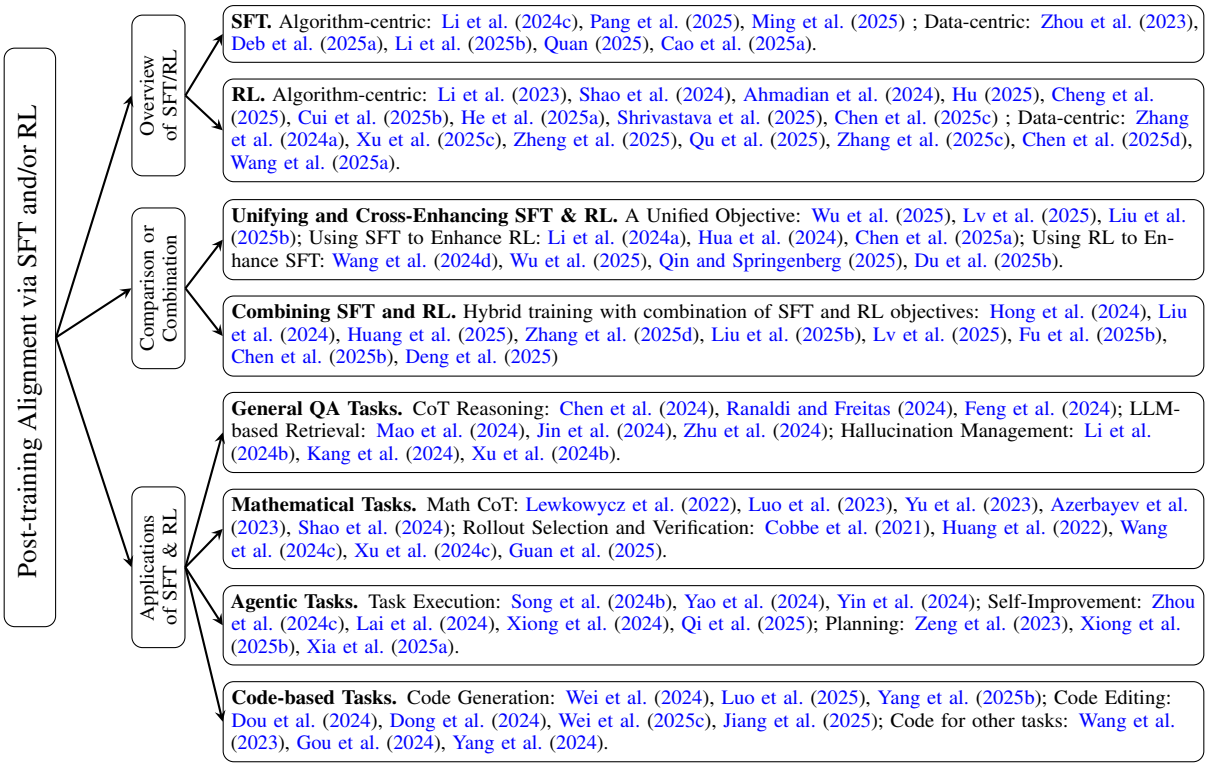

Figure 1: A taxonomy of large language model (LLM) post-training alignment methods via supervised fine-tuning (SFT) and reinforcement learning (RL). We organize prior work along three categories: (1) algorithm-centric versus data-centric approaches within SFT and RL, (2) comparative, unifying, and hybrid frameworks that integrate SFT and RL objectives, and (3) representative downstream application domains, including reasoning, mathematics, agentic behavior, and code-related tasks.

studies highlight that understanding and combining the complementary strengths of SFT and RL is crucial for advancing LLM post-training methods.

While recent surveys offer valuable insights into LLM post-training, the majority of they typically examine SFT or RL separately (Parthasarathy et al., 2024; Tao et al., 2024; Mao et al., 2025; Tie et al., 2025; Zhang et al., 2025b,a), leaving the relationships between these approaches comparatively underexplored. Other works focus on specialized dimensions of post-training, such as vision-centric adaptation (Chu et al., 2025), advances in reasoning (Kumar et al., 2025), agentic behaviors (Du et al., 2025a), or scaling strategies (Lai et al., 2025). In contrast, our survey provides a systematic and integrated perspective on SFT and RL as complementary post-training tools, with particular emphasis on their interplay and practical applications.

Our key **contributions** are threefold:

- We are the first study to systematically summarize and compare SFT and RL in LLM post-training, providing a clear understanding of what SFT and RL are, and how they can be extended from both algorithm-centric and data-centric perspectives.
- We then establish a unified framework for characterizing SFT and RL, highlighting how they can complement each other or be integrated into hybrid learning approaches.
- From a survey of applications spanning 2023 to 2025, we observe rapid task-domain expansion, growing adoption of integrated SFT–RL training, and a continued shift from API-based labeling to open-weight–generated datasets.

2 Background: SFT and RL

Supervised Fine-Tuning (SFT). SFT is a method for adapting pre-trained LLMs to specific tasks or domains by training on high-quality prompt–response pairs using standard language modeling objectives. This process typically involves collecting or curating a dataset of expert-written demonstrations $D = \{(x, y)\}$, prompt x paired with target responses y, and then fine-tuning the model π_θ:

$$\min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[-\log \pi_\theta(y \mid x)\right].$$

In essence, SFT trains a model to imitate expert behavior. It is closely related to Behavior Cloning (BC) (Pomerleau, 1991) in traditional reinforcement learning, as both frameworks learn directly from expert demonstrations without relying on explicit reward signals. The goal of SFT is to reproduce expert performance; however, like BC, it suffers from distribution shift, which can lead to compounding errors (Ross and Bagnell, 2010; De Haan et al., 2019), and depends heavily on the quality of the demonstrations.

Reinforcement Learning (RL). RL offers an alternative paradigm in which LLMs are tuned not through direct supervised signals, but by optimizing behaviors according to a reward function. In classical RL, models interact with environments and learn a policy that maximizes cumulative rewards over time. Recent LLMs post-training methods (Ouyang et al., 2022; Guo et al., 2025; Yang et al., 2025a) have adopted similar techniques: an environment is constructed, the model serves as a policy that interacts with this environment, and training proceeds to maximize expected rewards:

$$\max_{\theta} \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)} \left[r(x,y)\right],$$

where the reward function r is either manually specified or learned from data. Although RL can also refer to offline reinforcement learning without environment interactions, in this work, we use it exclusively to denote online RL, as it plays a predominant role in the current frontier of LLM alignment, including reasoning and agent development.

Here, we summarize the key distinction between SFT and RL in the context of LLM post-training, as commonly described in current literature (Shao et al., 2024; Zhang et al., 2025b): SFT is an **supervised learning** paradigm that trains on expert-annotated prompt–response pairs, Whereas RL is a **reward-driven optimization** paradigm that learns by updating the model from its own generations.

3 SFT and RL: Distinct Methodological Landscapes

In the evolving landscape of LLM post-training, SFT and RL are two primary methodological paradigms. This section presents an overview of these approaches from (1) **Algorithm-centric**: refined training algorithms or loss functions, and (2) **Data-centric**: curated data selection or sophisticated data synthesis.

3.1 SFT

Algorithm-centric SFT. Li et al. (2024c) introduces Entropic Distribution Matching (GEM), which reformulates SFT as a distribution-matching problem with entropy regularization to mitigate overfitting and preserve output diversity. Token Cleaning (Pang et al., 2025) estimates the contribution of each token to model updates and removes uninformative ones, effectively reducing noise in supervision. One-Token Rollout (Ming et al., 2025) is a policy-gradient-inspired variant of SFT that treats each token prediction as a one-step trajectory and leverages the ground-truth token as a reward, introducing on-policy learning signals without the complexity of RL.

Data-centric SFT. Zhou et al. (2023) fine-tunes their model on only $1,000$ high-quality and diverse instruction–response pairs, achieving alignment performance comparable to that of much larger models. FisherSFT (Deb et al., 2025b) selects training examples that maximize information gain, achieving efficient learning with limited data. Li et al. (2025b) optimizes data mixing by learning domain-specific weights that minimize validation loss, thereby improving generalization with minimal tuning cost. Quan (2025) proposes to automatically generate context-driven instruction–response pairs to enrich and diversify SFT data without heavy human annotation. Finally, Condor (Cao et al., 2025a) integrates knowledge-guided synthesis and iterative refinement to produce high-fidelity alignment data, further demonstrating the importance of data curation.

3.2 RL

Algorithm-centric RL. Several modified objectives and training procedures have been proposed to extend LLM capabilities beyond what standard RL achieves. A key line of innovation lies in policy optimization algorithms. While Proximal Policy Optimization (PPO) (Schulman et al., 2017), with a learned value critic, has been the dominant approach, recent work favors critic-free methods such as Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which simplifies training by replacing the value network with group-relative normalized advantages. Other methods adopt direct REINFORCE-style updates (Li et al., 2023; Ahmadian et al., 2024; Hu, 2025; Xiong et al., 2025a), foregoing the complex components of PPO. Another active direction focuses on improving train-

ing stability and efficiency through objective regularization. Entropy regularization remains crucial for preventing entropy collapse (Cheng et al., 2025; Cui et al., 2025a). He et al. (2025a); Shrivastava et al. (2025) introduce weighted token entropy to regularize policy updates. In contrast, Cui et al. (2025a) identifies the covariance between an action's probability and its advantage as the key entropy "driver," proposing covariance and Kullback–Leibler (KL) clipping to selectively constrain tokens with exceptionally high covariance. Moreover, Cheng et al. (2025) and Chen et al. (2025c) incorporate entropy information into advantage estimation, further stabilizing training dynamics.

Data-centric RL. Several approaches (Zhang et al., 2024a; Xu et al., 2025c) perform rollout selection, demonstrating that selectively training on a subset of informative model rollouts can achieve promising performance, such as high-variance or diverse examples, hence highly data efficient. Other approaches (Zheng et al., 2025; Qu et al., 2025) explore prompt selection (prior to rollout) to reduce computational cost while maintaining performance. Curriculum learning also contributes to this direction. Zhang et al. (2025c); Yao et al. (2025) dynamically select prompts of intermediate difficulty to maximize the learning signal. Moreover, Chen et al. (2025d); Wang et al. (2025a) introduce a distribution-level adaptation approach that prioritizes tasks where the model exhibits the greatest advantage or lowest visitation.

4 Comparison and Combination of SFT and RL

Despite the prevalence of SFT and RL, two commonly adopted post-training techniques for LLMs, they stay relatively disentangled and are usually applied in a sequential manner. For example, Reinforcement Learning from Human Feedback (RLHF) initially applies SFT to inject prior general knowledge into the LLMs, and then conducts RL for promoting the capability in a particular aspect (Ouyang et al., 2022; Bai et al., 2022). However, there is a missing systematical comparison and combination between these two stages, especially from a methodological and theoretical perspective, despite that there have been some pioneering works that try to modify the objectives of each other to complement each other's strengths. The detailed comparison between different manners to combine SFT and RL together is summarised in Table 1.

4.1 A Unified Objective

First, we write down the objectives of both SFT and RL respectively in an explicit manner,

$$\mathcal{L}_{\mathrm{SFT}} = \mathbb{E}_{(x,y)\sim\mathcal{D}}[-\log \pi_\theta(y|x)],$$
$$\mathcal{L}_{\mathrm{RL}} = \mathbb{E}_{x\sim\mathcal{D}_x, y\sim\pi_\theta(\cdot|x)}[r(x,y)].$$

Based on the above formulation, we could derive the gradients for both objectives below,

$$\nabla_\theta\mathcal{L}_{\mathrm{SFT}} = \mathbb{E}_{(x,y)\sim\mathcal{D}}[-\nabla_\theta \log \pi_\theta(y|x)],$$
$$\nabla_\theta\mathcal{L}_{\mathrm{RL}} = \mathbb{E}_{x\sim\mathcal{D}_x, y\sim\pi_\theta(\cdot|x)}[\nabla_\theta \log \pi_\theta(y|x)\cdot r(x,y)].$$

As pointed out in recent works (Wu et al., 2025), the objective of SFT could be regarded as a special case of RL, i.e.,

$$\nabla_\theta\mathcal{L}_{\mathrm{SFT}} = \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}}[-\nabla_\theta \log \pi_\theta(y_i|x_i)]$$
$$= \mathbb{E}_{x_i\sim\mathcal{D}_x}[\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[-\frac{\mathbb{I}\{y=y_i\}}{\pi_\theta(y|x)}\nabla_\theta \log \pi_\theta(y_i|x_i)]],$$

where $\mathbb{I}\{y=y_i\}$ is the indicator function, which takes value 1 when the output y sampled from policy $\pi_\theta(\cdot|x)$ is exactly the same with the ground-truth response y_i in the SFT training dataset. Therefore, the ratio $-\frac{\mathbb{I}\{y=y_i\}}{\pi_\theta(y|x)}$ could be seen as a proxy reward function in SFT.

Based on the above argument, formatting SFT as a special case of RL, the training objective of most post-training could be abstracted into the following formula,

$$\max_\pi \mathbb{E}_{x\sim\mathcal{D}_x, y\sim\pi_\theta(\cdot|x)}\big[r(x,y)\big]$$
$$-\beta\,\mathrm{KL}\big(\pi_\theta(\cdot|x)\|\pi_0(\cdot|x)\big),$$

where π_0 is the base (reference) policy model; r is a proxy for the reward, and β is a hyper-parameter. The KL regularization between the policy model and the reference model restricts π_θ from deviating too much from a pre-trained checkpoint, primarily for stability reasons. In the following sections, we will continue to discuss the difference and combination of SFT and RL, mainly from the objective perspective.

Therefore, optimizing LLMs with both SFT and RL objectives ultimately collapses to the RL objective, and the tricks that work for one thus have potential to be applied to the other. To mitigate the drop in generalization ability from SFT, one could consider applying importance sampling and online rollout, as in RL. Meanwhile, to help LLMs memorize additional knowledge, one could integrate the

SFT loss into the RL objective. The roles of these two stages should be regarded as a mutually reinforcing and interdependent relationship instead of merely being applied alternatively.

4.2 Leveraging SFT to Enhance RL

Combine offline demonstration and online data Recent studies have leveraged SFT to enhance RL in LLMs by combining offline demonstrations with online rollouts. Yan et al. (2025a) introduces an off-policy guided framework that augments on-policy updates with reasoning traces, effectively balancing imitation and exploration. SRFT (Fu et al., 2025b) integrates supervised and reinforcement objectives in a single-stage framework, avoiding inefficiencies of sequential fine-tuning. Similar to LLMs, Vision Language Models (VLMs) are another flourishing area with great potential by extending the single text-based modality of LLMs into vision, like images and videos. Liu et al. (2025a) propose a dynamic memorization–exploration strategy for small VLMs, which adaptively alternates between demonstration imitation and online reward optimization. AMFT (He et al., 2025b) adopts a meta-learning perspective, dynamically adjusting the imitation–exploration balance to maximize performance. Lv et al. (2025) provides a unified view of post-training, showing that both SFT and RL optimize a common objective, and propose hybrid updates to exploit demonstration and rollout data. BREAD (Zhang et al., 2025e) bridges SFT and RL through branched rollouts anchored by expert prefixes, reducing reliance on large demonstration sets while improving stability. Similarly, Huang et al. (2025) develops a prefix sampling approach that guides generation into high-quality trajectories before applying RL optimization. Collectively, these methods illustrate complementary strategies, including single-stage integration, adaptive weighting, and prefix-based seeding, that enhance the efficiency and robustness of RL when grounded in SFT demonstrations.

Objective Modification NFT (Chen et al., 2025a) enables models to learn from both correct and incorrect outputs under supervision while implicitly optimizing a policy to enhance reasoning performance. UFT (Liu et al., 2025b) unifies supervised and reinforcement fine-tuning into a single process that balances memorization and exploration, achieving better generalization and exponentially improved sample efficiency. ReLIFT (Zhu et al., 2025b) interleaves RL with SFT on the hardest questions, allowing models to acquire capabilities beyond what pure RL can achieve. Zhu et al. (2025b) demonstrates that penalizing incorrect answers alone, through negative sample reinforcement, can substantially improve reasoning performance, often matching or exceeding classical RL methods by suppressing wrong outputs and reallocating probability to plausible alternatives.

4.3 From RL Perspective to Improve SFT

DFT (Wu et al., 2025) is proposed to rescale each token's loss by its predicted probability to rectify implicit reward bias in SFT and boost generalization. iw-SFT (Qin and Springenberg, 2025) interpret curated SFT as optimizing a lower bound on a sparse-reward RL objective and tightening that bound via importance weights. Another work (Du et al., 2025b) in RLHF proposes a reweighted reward-driven SFT objective through variational inference. In contrast, inspired by the RL, especially Direct Preference Optimization (DPO) (Rafailov et al., 2023), objective, Wang et al. (2024d) minimizes the distribution difference between the reference model and policy model on an offline dataset.

4.4 Hybrid Training Combining SFT and RL

Huang et al. (2025) proposes to combine SFT and RL together through utilizing the prefix of a ground-truth response to generate new continuation based on the current policy model, and use SFT loss to train the prefix partition while use RL loss to train the newly generated partition. Lv et al. (2025) uses interleaving SFT and RL based on the performance of the policy model during online rollouts. When the performance is above a preset threshold, online RL is preferred for exploration, while correct guidance from SFT is preferred when the performance is bad. Liu et al. (2024) starts from the original DPO (Rafailov et al., 2023) objective, and directly injects the SFT loss on samples from the base model to mitigate the overoptimization. Zhang et al. (2025d) combines a weighted SFT loss and GRPO (Shao et al., 2024) loss together to aggregate the off-policy and on-policy training. Liu et al. (2025b) uses expert data in the first several steps, and switches to the new generation in the following steps. SRL (Deng et al., 2025) proposes to decompose the reasoning process into several intermediate steps, and compare online rollouts with offline expert trajectories, and assign rewards proportional to the matching between them.

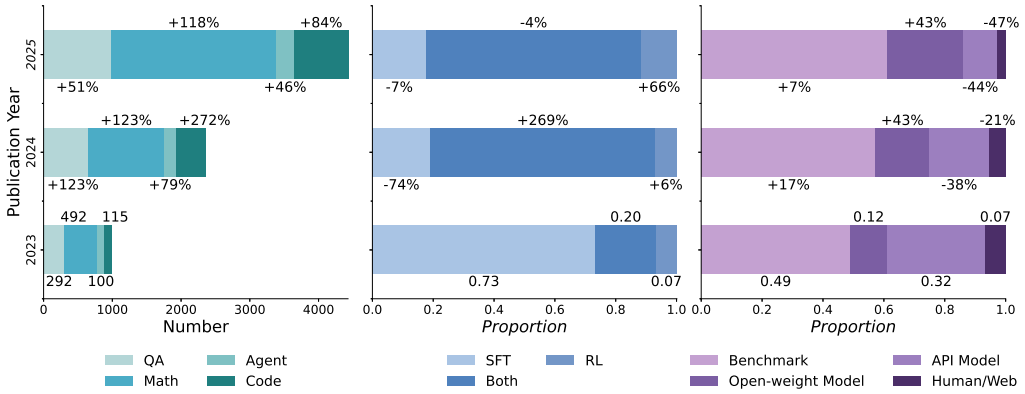

Figure 2: Trends in task focus, training methodologies, and ground-truth data sources from 2023 to 2025. Results show rapid growth across all surveyed domains, with substantial increases in research volume and diversification of application areas; increasing convergence toward hybrid SFT–RL pipelines, supported by more mature training infrastructures, libraries, and preference datasets; and a continued shift from API-based labeling to data generated with increasingly capable open-weight models. Projections for 2025 and all reported proportions are derived from surveyed publications; see Appendix B.4 for further discussion.

5 Applications

We focus on application studies that post-train LLMs using SFT or RL with four domains, where each domain encompasses distinct methodologies and challenges, and these categories capture the breadth of LLM capabilities and support a systematic analysis of performance across tasks. Key trends are summarized in Fig. 2, with additional details provided in Appendix B.

5.1 LLMs for General QA Tasks

This subsection reviews methods for improving LLM Question Answering (QA) performance via reasoning augmentation, e.g., CoT; external knowledge integration, e.g., Retrieval-Augmented Generation (RAG); and hallucination mitigation techniques.

Step-by-Step Reasoning. Smaller Foundation Model (FM) trained only on web data generally lack native CoT reasoning; thus, CoT demonstrations are typically generated using larger models via prompt engineering and subsequently used for SFT to instruction-tune reasoning or query decomposition capabilities. Building upon this approach, some works (Ranaldi and Freitas, 2024; Feng et al., 2024) introduce an additional RL stage, where collected demonstrations are treated as positive examples and newly generated model outputs are treated as negative samples. A key challenge in equipping LLMs with CoT capabilities lies in evaluating the quality of generated reasoning, as providing fine-grained, step-wise supervision in an online setting is often difficult or costly. To mitigate this, researchers have proposed using surrogate signals, such as answer correctness (Chen et al., 2024; Guo et al., 2025) or reasoning comprehensiveness (Huang et al., 2024), and applying Rejection Fine-Tuning (RFT) or RL on self-generated samples.

LLM-based Retrieval. In RAG pipelines, LLMs are often used for query rewriting prior to retrieval. Prior work shows that FMs can perform this task effectively, with retrieval quality evaluated via downstream responses, and several studies (Mao et al., 2024; Hsu et al., 2024; Yan and Ling, 2025; Xu et al., 2025a; Gao et al., 2025) directly apply RL with task-specific search rewards for model alignment.

Hallucination Management. LLM hallucination refers to the generation of responses that appear plausible but contain incorrect or fabricated information. This phenomenon typically arises when the required factual knowledge is absent from the pre-training corpus, leading the model to infer or construct an answer rather than abstain. To mitigate hallucinations for general question, researchers have leveraged PPO to reduce factual errors by incorporating response corrections (Li et al., 2024b) provided by stronger models. Additionally, to encourage models to explicitly acknowledge uncertainty and suppress hallucinated outputs, prior works (Kang et al., 2024; Xie et al., 2024; Cheng et al., 2024; Xu et al., 2024a) have proposed inference over a collection of sampled responses, often derived from benchmark datasets, and synthesizing "I do not know" outputs for subsequent SFT or RL pipelines. Furthermore, to quantify

and model uncertainty, several studies (Band et al., 2024; Sun et al., 2025) generate multiple candidate responses for each training instance and fine-tune the model using aggregated or summarized outputs as approximations of the underlying uncertainty distribution.

5.2 LLMs for Mathematical Tasks

This subsection reviews methods for enhancing LLMs' mathematical capabilities, with a focus on joint mathematical reasoning and training using step-wise rollouts.

Mathematical Reasoning. Early efforts for mathematical reasoning abilities can be traced back to Polu and Sutskever (2020) and Cobbe et al. (2021), which studied problems involving algorithmic computation and mathematical induction. Subsequent work further improved performance by SFT FMs on curated mathematical corpora collected from the web (Lewkowycz et al., 2022; Azerbayev et al., 2023), synthetically generated datasets (Yu et al., 2023; Toshniwal et al., 2024a), or data produced by stronger models (Luo et al., 2023; Toshniwal et al., 2024b). More recent studies construct explicit reward models for mathematical evaluation and propose alignment procedures, such as GRPO (Shao et al., 2024) and Critique-DPO (Xu et al., 2024c), to enable downstream RL.

Rollout-Based Training. Similar to standard CoT prompting, developing reliable step-wise reward evaluation for mathematical reasoning remains challenging. Huang et al. (2022) proposes fine-tuning models on their own generated solutions using majority voting. Additionally, many researchers exploited FMs with multiple reasoning paths when solving mathematical problems. For PPO-based methods, Wang et al. (2024c) and Guan et al. (2025) approximate process-level rewards using the probability of producing a correct final answer, while Shen et al. (2025) randomly assigns reflection prompts and incorporates both outcome-based and reflection-based bonus rewards.

5.3 LLMs for Agentic Tasks

This subsection reviews strategies for improving the agentic capabilities of LLMs, with a focus on action selection and long-horizon planning.

Action Selection. Agentic tasks require selecting actions over multiple steps within a given environment. To equip FMs with this capability, pioneering work such as Schick et al. (2023); Yin et al. (2024); Zhang et al. (2024b,c) synthesizes or unifies action trajectories across diverse environments for SFT. Subsequent studies adopt a sequential paradigm that combines SFT with RL using environment-specific rewards. For example, Xi et al. (2024) incorporates SFT rewards directly into the SFT objective, while hierarchical RL approaches (Zhou et al., 2024c; Hu et al., 2025b) assign rewards at the sentence level while updating policies at the token level. Similarly, DMPO (Shi et al., 2025) introduces step-wise reward discounting, and Qi et al. (2025) employs critic-based curriculum learning to stabilize RL training.

Planning with LLMs. Task planning has emerged as an active research direction, with agentic tasks serving as a natural testbed due to their long action horizons and complex inter-step dependencies. Recent work predominantly leverages outcome-based surrogate rewards and GPT-synthesized sequences to optimize planning behavior. For instance, Xiong et al. (2025b) proposes a hierarchical planning framework with a top-level planner fine-tuned on seed plans and optimized via DPO. Vattikonda et al. (2025) further incorporates RFT to refine high-reward actions, while Wang et al. (2025b) improves planning performance by adjusting exploration strategies and prioritizing critical steps during DPO training.

5.4 LLMs for Coding Tasks

This subsection reviews methods for improving coding capabilities in LLMs, focusing on code generation and code editing.

Code Generation. Recent work has explored various approaches to enhance LLM-based code generation. Under SFT, Chaudhary (2023) employs self-instruction to synthesize training data, while Luo et al. (2025) generates more challenging queries by modifying constraints of seed problems. Wei et al. (2024) leverages open-source code snippets to produce diverse and controllable instruction datasets. From the RL perspective, Yang et al. (2025b) introduces a multi-agent framework in which an SFT model generates rollouts for subsequent DPO training.

Code Editing. Code editing aims to refine programs that fail to satisfy intended functionality, whether human-written or LLM-generated. For example, Wei et al. (2025c) leverages human pull requests for GRPO-based training, while Chae et al. (2024) and Jiang et al. (2025) generate iterative refinement rollouts evaluated using unit tests to drive RL optimization.

6 Common Practice and Takeaways

In this section, we summarize common practices and key takeaways from our previous theoretical and application discussions, highlighting the interplay between SFT and RL, when to choose each method, and training trade-offs.

Relationship Between SFT and RL Objectives. As discussed in Section 4, SFT can be viewed as a special case of RL under certain formulations, where an indicator function that measures whether the current policy reproduces an offline trajectory acts as a surrogate reward. This shared optimization structure enables techniques developed for either SFT or RL to transfer across paradigms. For example, Section 5.3 reviews several studies that jointly optimize weighted SFT and RL objectives, demonstrating consistent improvements over standalone training losses.

Leveraging Expert Data or Policies. When high-quality expert data are available, SFT is generally preferred as an initial training stage over RL as observed in Section 5, possibly due to its simpler implementation and greater stability. If a strong policy model or prompt is available to generate expert data, incorporating importance sampling on failed queries can mitigate distribution shift by treating them as informative positive samples for RFT/RL, bridging the gap between the limited offline dataset and the evolving policy. When a reliable reward model can be trained, the common practice is to first perform SFT and then RL, which typically achieves the highest reported performance.

Training Strategies and Trade-offs. SFT is essential when the policy model is unfamiliar with the downstream task or cannot efficiently generate sufficient positive samples. However, SFT may reduce the model's generalization ability compared with RL. Conversely, directly applying RL can better balance performance gains with exploratory capacity, though it may suffer from issues such as entropy collapse (Cui et al., 2025a) or reward hacking (Pan et al., 2024a,b), one possible reason why the majority of pre-training computation is still devoted to SFT.

7 Future Directions and Open Problems

Despite rapid progress, many fundamental questions remain unresolved. We highlight two open challenges and outline promising directions for future research.

Sample- and compute-efficient methodologies. State-of-the-art SFT and RL pipelines typically require substantial computational resources, large volumes of high-quality data, and extensive rollout generation, raising both practical and environmental concerns. Improving efficiency is therefore a central challenge. Early progress includes data-efficient SFT based on information-theoretic principles (Deb et al., 2025b) and quantization-aware methods that reduce training cost (Wei et al., 2025b). For RL, recent work explores selective or partial rollouts to reduce computation (Zheng et al., 2025; Xu et al., 2025b). Nevertheless, significantly more research is needed to develop broadly applicable, highly efficient methods.

SFT and RL under sparse or indirect reward signals. Many real-world tasks lack well-defined or easily verifiable reward signals. User feedback may be sparse, inconsistent, or costly. In safety-critical settings, it may be infeasible to obtain ground-truth evaluations altogether. Recent work has begun to explore alignment through verbal or otherwise indirect feedback (Stephan et al., 2024; Wang et al., 2024b). An open challenge is to identify and leverage additional natural but underexplored sources of implicit supervision, such as self-evaluation signals or user-churn behaviors, to guide both SFT and RL in low-feedback regimes.

8 Conclusion

This survey provides a unified view of SFT and RL as post-training strategies for LLMs, examining both their theoretical foundations and practical implementations. SFT offers stable and efficient learning from high-quality offline datasets, whereas RL enables reward-driven optimization that enhances generalization and exploration. Recent research characterizes SFT as a special case of RL, underscoring the promise of hybrid approaches that balance training stability with adaptive behavior. Application papers published between 2023 and 2025 indicate a clear shift toward hybrid post-training pipelines that integrate SFT and RL, with scalable workflows centered on open-weight, model-generated rollouts emerging as the dominant paradigm. These developments underscore the need for integrative methodologies that leverage the strengths of both approaches, and this survey lays a foundation by consolidating the current state of the field, theoretical advances, and versatile LLM applications.

Limitations

Despite our best efforts to ensure comprehensiveness, this survey may omit some recent advances in SFT/RL research due to the rapid pace of progress and the sheer volume of emerging literature in this field. In the *Applications* section, the use of a paper-filtering strategy may introduce approximation bias, as commonly adopted benchmarks are not fully inclusive of all real-world scenarios or methodologies. Additional discussion of application approximation limitations is provided in Appendix B.2.

References

Mohammad Mahdi Abootorabi, Amirhosein Zobeiri, Mahdi Dehghani, Mohammadali Mohammadkhani, Bardia Mohammadi, Omid Ghahroodi, Mahdieh Soleymani Baghshah, and Ehsaneddin Asgari. 2025. Ask in any modality: A comprehensive survey on multimodal retrieval-augmented generation. *Preprint*, arXiv:2502.08826.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Neil Band, Xuechen Li, Tengyu Ma, and Tatsunori Hashimoto. 2024. Linguistic calibration of long-form generations. *Preprint*, arXiv:2404.00474.

Maosong Cao, Taolin Zhang, Mo Li, Chuyu Zhang, Yunxin Liu, Haodong Duan, Songyang Zhang, and Kai Chen. 2025a. Condor: Enhance llm alignment with knowledge-driven data synthesis and refinement. *arXiv preprint arXiv:2501.12273*.

Zouying Cao, Runze Wang, Yifei Yang, Xinbei Ma, Xiaoyong Zhu, Bo Zheng, and Hai Zhao. 2025b. Pgpo: Enhancing agent reasoning via pseudocode-style planning guided preference optimization. *Preprint*, arXiv:2506.01475.

Hyungjoo Chae, Taeyoon Kwon, Seungjun Moon, Yongho Song, Dongjin Kang, Kai Tzu iunn Ong, Beong woo Kwak, Seonghyeon Bae, Seung won Hwang, and Jinyoung Yeo. 2024. Coffee-gym: An environment for evaluating and improving natural language feedback on erroneous code. *Preprint*, arXiv:2409.19715.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca.

Huayu Chen, Kaiwen Zheng, Qinsheng Zhang, Ganqu Cui, Yin Cui, Haotian Ye, Tsung-Yi Lin, Ming-Yu Liu, Jun Zhu, and Haoxiang Wang. 2025a. Bridging supervised learning and reinforcement learning in math reasoning. *arXiv preprint arXiv:2505.18116*.

Jack Chen, Fazhong Liu, Naruto Liu, Yuhan Luo, Erqu Qin, Harry Zheng, Tian Dong, Haojin Zhu, Yan Meng, and Xiao Wang. 2025b. Step-wise adaptive integration of supervised fine-tuning and reinforcement learning for task-specific llms. *arXiv preprint arXiv:2505.13026*.

Minghan Chen, Guikun Chen, Wenguan Wang, and Yi Yang. 2025c. Seed-grpo: Semantic entropy enhanced grpo for uncertainty-aware policy optimization. *arXiv preprint arXiv:2505.12346*.

Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. 2025d. Self-evolving curriculum for llm reasoning. *arXiv preprint arXiv:2505.14970*.

Zhaorun Chen, Zhuokai Zhao, Zhihong Zhu, Ruiqi Zhang, Xiang Li, Bhiksha Raj, and Huaxiu Yao. 2024. Autoprm: Automating procedural supervision for multi-step reasoning via controllable question decomposition. *Preprint*, arXiv:2402.11452.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. 2025. Reasoning with exploration: An entropy perspective. *arXiv preprint arXiv:2506.14758*.

Qinyuan Cheng, Tianxiang Sun, Xiangyang Liu, Wenwei Zhang, Zhangyue Yin, Shimin Li, Linyang Li, Zhengfu He, Kai Chen, and Xipeng Qiu. 2024. Can ai assistants know what they don't know? *Preprint*, arXiv:2401.13275.

Sanjiban Choudhury and Paloma Sodhi. 2024. Better than your teacher: Llm agents that learn from privileged ai feedback. *Preprint*, arXiv:2410.05434.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*.

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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. 2025a. The entropy mechanism of reinforcement learning for reasoning language models. *Preprint*, arXiv:2505.22617.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, and 1 others. 2025b. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*.

Wenqian Cui, Dianzhi Yu, Xiaoqi Jiao, Ziqiao Meng, Guangyan Zhang, Qichao Wang, Yiwen Guo, and Irwin King. 2025c. Recent advances in speech language models: A survey. *Preprint*, arXiv:2410.03751.

Pim De Haan, Dinesh Jayaraman, and Sergey Levine. 2019. Causal confusion in imitation learning. *Advances in neural information processing systems*, 32.

Rohan Deb, Kiran Thekumparampil, Kousha Kalantari, Gaurush Hiranandani, Shoham Sabach, and Branislav Kveton. 2025a. Fishersft: Data-efficient supervised fine-tuning of language models using information gain. *arXiv preprint arXiv:2505.14826*. ICML Poster (2025).

Rohan Deb, Kiran Thekumparampil, Kousha Kalantari, Gaurush Hiranandani, Shoham Sabach, and Branislav Kveton. 2025b. Fishersft: Data-efficient supervised fine-tuning of language models using information gain. *Preprint*, arXiv:2505.14826. ICML Poster (2025).

Yihe Deng, I Hsu, Jun Yan, Zifeng Wang, Rujun Han, Gufeng Zhang, Yanfei Chen, Wei Wang, Tomas Pfister, Chen-Yu Lee, and 1 others. 2025. Supervised reinforcement learning: From expert trajectories to step-wise reasoning. *arXiv preprint arXiv:2510.25992*.

Zhirui Deng, Zhicheng Dou, Yutao Zhu, Ji-Rong Wen, Ruibin Xiong, Mang Wang, and Weipeng Chen. 2024. From novice to expert: Llm agent policy optimization via step-wise reinforcement learning. *Preprint*, arXiv:2411.03817.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Preprint*, arXiv:2305.14314.

Guanting Dong, Keming Lu, Chengpeng Li, Tingyu Xia, Bowen Yu, Chang Zhou, and Jingren Zhou. 2024. Self-play with execution feedback: Improving instruction-following capabilities of large language models. *Preprint*, arXiv:2406.13542.

Shihan Dou, Yan Liu, Haoxiang Jia, Limao Xiong, Enyu Zhou, Wei Shen, Junjie Shan, Caishuang Huang, Xiao Wang, Xiaoran Fan, Zhiheng Xi, Yuhao Zhou, Tao Ji, Rui Zheng, Qi Zhang, Xuanjing Huang, and Tao Gui. 2024. Stepcoder: Improve code generation with reinforcement learning from compiler feedback. *Preprint*, arXiv:2402.01391.

Shangheng Du, Jiabao Zhao, Jinxin Shi, Zhentao Xie, Xin Jiang, Yanhong Bai, and Liang He. 2025a. A survey on the optimization of large language model-based agents. *arXiv preprint arXiv:2503.12434*.

Yuhao Du, Zhuo Li, Pengyu Cheng, Zhihong Chen, Yuejiao Xie, Xiang Wan, and Anningzhe Gao. 2025b. Simplify rlhf as reward-weighted sft: A variational method. *arXiv preprint arXiv:2502.11026*.

Peiyuan Feng, Yichen He, Guanhua Huang, Yuan Lin, Hanchong Zhang, Yuchen Zhang, and Hang Li. 2024. Agile: A novel reinforcement learning framework of llm agents. *Preprint*, arXiv:2405.14751.

Dayuan Fu, Keqing He, Yejie Wang, Wentao Hong, Zhuoma Gongque, Weihao Zeng, Wei Wang, Jingang Wang, Xunliang Cai, and Weiran Xu. 2025a. Agentrefine: Enhancing agent generalization through refinement tuning. *Preprint*, arXiv:2501.01702.

Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. 2025b. Srft: A single-stage method with supervised and reinforcement fine-tuning for reasoning. *arXiv preprint arXiv:2506.19767*.

Jingsheng Gao, Linxu Li, Weiyuan Li, Yuzhuo Fu, and Bin Dai. 2025. Smartrag: Jointly learn rag-related tasks from the environment feedback. *Preprint*, arXiv:2410.18141.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2024. Tora: A tool-integrated reasoning agent for mathematical problem solving. *Preprint*, arXiv:2309.17452.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *Preprint*, arXiv:2501.04519.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, and 1 others. 2025a. Skywork open reasoner 1 technical report. *arXiv preprint arXiv:2505.22312*.

Lixuan He, Jie Feng, and Yong Li. 2025b. Amft: Aligning llm reasoners by meta-learning the optimal imitation-exploration balance. *arXiv preprint arXiv:2508.06944*.

Tao He, Hao Li, Jingchang Chen, Runxuan Liu, Yixin Cao, Lizi Liao, Zihao Zheng, Zheng Chu, Jiafeng Liang, Ming Liu, and Bing Qin. 2025c. Breaking the reasoning barrier a survey on LLM complex reasoning through the lens of self-evolution. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 7377–7417, Vienna, Austria. Association for Computational Linguistics.

Joey Hong, Anca Dragan, and Sergey Levine. 2024. Q-sft: Q-learning for language models via supervised fine-tuning. *arXiv preprint arXiv:2411.05193*.

Sheryl Hsu, Omar Khattab, Chelsea Finn, and Archit Sharma. 2024. Grounding by trying: Llms with reinforcement learning-enhanced retrieval. *Preprint*, arXiv:2410.23214.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Jian Hu. 2025. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, Yuhuai Li, Shengze Xu, Shenzhi Wang, Xinchen Xu, Shuofei Qiao, Zhaokai Wang, Kun Kuang, Tieyong Zeng, Liang Wang, and 10 others. 2025a. OS agents: A survey on MLLM-based agents for computer, phone and browser use. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7436–7465, Vienna, Austria. Association for Computational Linguistics.

Zican Hu, Wei Liu, Xiaoye Qu, Xiangyu Yue, Chunlin Chen, Zhi Wang, and Yu Cheng. 2025b. Divide and conquer: Grounding llms as efficient decision-making agents via offline hierarchical reinforcement learning. *Preprint*, arXiv:2505.19761.

Ermo Hua, Biqing Qi, Kaiyan Zhang, Kai Tian, Xingtai Lv, Ning Ding, and Bowen Zhou. 2024. Intuitive fine-tuning: Towards simplifying alignment into a single process. *arXiv preprint arXiv:2405.11870*.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*.

Lei Huang, Xiaocheng Feng, Weitao Ma, Liang Zhao, Yuchun Fan, Weihong Zhong, Dongliang Xu, Qing Yang, Hongtao Liu, and Bing Qin. 2024. Advancing large language model attribution through self-improving. *Preprint*, arXiv:2410.13298.

Zeyu Huang, Tianhao Cheng, Zihan Qiu, Zili Wang, Yinghui Xu, Edoardo M Ponti, and Ivan Titov. 2025. Blending supervised and reinforcement fine-tuning with prefix sampling. *arXiv preprint arXiv:2507.01679*.

Nan Jiang, Xiaopeng Li, Shiqi Wang, Qiang Zhou, Soneya Binta Hossain, Baishakhi Ray, Varun Kumar, Xiaofei Ma, and Anoop Deoras. 2025. Ledex: Training llms to better self-debug and explain code. *Preprint*, arXiv:2405.18649.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? *Preprint*, arXiv:2310.06770.

Jiajie Jin, Yutao Zhu, Yujia Zhou, and Zhicheng Dou. 2024. Bider: Bridging knowledge inconsistency for efficient retrieval-augmented llms via key supporting evidence. *Preprint*, arXiv:2402.12174.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *Preprint*, arXiv:1705.03551.

Katie Kang, Eric Wallace, Claire Tomlin, Aviral Kumar, and Sergey Levine. 2024. Unfamiliar finetuning examples control how language models hallucinate. *Preprint*, arXiv:2403.05612.

Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li, Chengwei Qin, Peifeng Wang, Silvio Savarese, Caiming Xiong, and Shafiq Joty. 2025. A survey of frontiers in llm reasoning: Inference scaling, learning to reason, and agentic systems. *Preprint*, arXiv:2504.09037.

Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Fahad Shahbaz Khan, and Salman Khan. 2025. Llm post-training: A deep dive into reasoning large language models. *arXiv preprint arXiv:2502.21321*.

Hanyu Lai, Xiao Liu, Junjie Gao, Jiale Cheng, Zehan Qi, Yifan Xu, Shuntian Yao, Dan Zhang, Jinhua Du, Zhenyu Hou, and 1 others. 2025. A survey of post-training scaling in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2771–2791.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. 2024. Autowebglm: A large language model-based web navigating agent. *Preprint*, arXiv:2404.03648.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2023. Ds-1000: A natural and reliable benchmark for data science code generation. In *International Conference on Machine Learning*, pages 18319–18345. PMLR.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo

Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857.

Jiaxiang Li, Siliang Zeng, Hoi-To Wai, Chenliang Li, Alfredo Garcia, and Mingyi Hong. 2024a. Getting more juice out of the sft data: Reward learning from human demonstration improves sft for llm alignment. *Advances in Neural Information Processing Systems*, 37:124292–124318.

Junyi Li, Jie Chen, Ruiyang Ren, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024b. The dawn after the dark: An empirical study on factuality hallucination in large language models. *Preprint*, arXiv:2401.03205.

Moxin Li, Yong Zhao, Wenxuan Zhang, Shuaiyi Li, Wenya Xie, See-Kiong Ng, Tat-Seng Chua, and Yang Deng. 2025a. Knowledge boundary of large language models: A survey. *Preprint*, arXiv:2412.12472.

Yuan Li, Zhengzhong Liu, and Eric Xing. 2025b. Data mixing optimization for supervised fine-tuning of large language models. *arXiv preprint arXiv:2508.11953*.

Ziniu Li, Congliang Chen, Tian Xu, Zeyu Qin, Jiancong Xiao, Ruoyu Sun, and Zhi-Quan Luo. 2024c. Entropic distribution matching in supervised fine-tuning of llms: Less overfitting and better diversity. *arXiv preprint arXiv:2408.16673*. Version v1; latest version v2 (2025).

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. 2023. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. *arXiv preprint arXiv:2310.10505*.

Jiazhen Liu, Yuchuan Deng, and Long Chen. 2025a. Empowering small vlms to think with dynamic memorization and exploration. *arXiv preprint arXiv:2506.23061*.

Mingyang Liu, Gabriele Farina, and Asuman Ozdaglar. 2025b. Uft: Unifying supervised and reinforcement fine-tuning. *arXiv preprint arXiv:2505.16984*.

Zhihan Liu, Miao Lu, Shenao Zhang, Boyi Liu, Hongyi Guo, Yingxiang Yang, Jose Blanchet, and Zhaoran Wang. 2024. Provably mitigating overoptimization in rlhf: Your sft loss is implicitly an adversarial regularizer. *Advances in Neural Information Processing Systems*, 37:138663–138697.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2025. Wizardcoder: Empowering code large language models with evol-instruct. *Preprint*, arXiv:2306.08568.

Xingtai Lv, Yuxin Zuo, Youbang Sun, Hongyi Liu, Yuntian Wei, Zhekai Chen, Lixuan He, Xuekai Zhu, Kaiyan Zhang, Bingning Wang, and 1 others. 2025. Towards a unified view of large language model post-training. *arXiv preprint arXiv:2509.04419*.

Zexiong Ma, Chao Peng, Pengfei Gao, Xiangxin Meng, Yanzhen Zou, and Bing Xie. 2025. Sorft: Issue resolving with subtask-oriented reinforced fine-tuning. *Preprint*, arXiv:2502.20127.

Shengyu Mao, Yong Jiang, Boli Chen, Xiao Li, Peng Wang, Xinyu Wang, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. 2024. Rafe: Ranking feedback improves query rewriting for rag. *Preprint*, arXiv:2405.14431.

Yuren Mao, Yuhang Ge, Yijiang Fan, Wenyi Xu, Yu Mi, Zhonghao Hu, and Yunjun Gao. 2025. A survey on lora of large language models. *Frontiers of Computer Science*, 19(7):197605.

Rui Ming, Haoyuan Wu, Shoubo Hu, Zhuolun He, and Bei Yu. 2025. One-token rollout: Guiding supervised fine-tuning of llms with policy gradient. *arXiv preprint arXiv:2509.26313*.

Shikhar Murty, Hao Zhu, Dzmitry Bahdanau, and Christopher D. Manning. 2025. Nnetnav: Unsupervised learning of browser agents through environment interaction in the wild. *Preprint*, arXiv:2410.02907.

Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, Xintong Li, Jing Shi, Hongjie Chen, Viet Dac Lai, Zhouhang Xie, Sungchul Kim, Ruiyi Zhang, Tong Yu, Mehrab Tanjim, and 11 others. 2025. GUI agents: A survey. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22522–22538, Vienna, Austria. Association for Computational Linguistics.

Fan Nie, Lan Feng, Haotian Ye, Weixin Liang, Pan Lu, Huaxiu Yao, Alexandre Alahi, and James Zou. 2025. Weak-for-strong: Training weak meta-agent to harness strong executors. *Preprint*, arXiv:2504.04785.

Tianyue Ou, Frank F. Xu, Aman Madaan, Jiarui Liu, Robert Lo, Abishek Sridhar, Sudipta Sengupta, Dan Roth, Graham Neubig, and Shuyan Zhou. 2024. Synatra: Turning indirect knowledge into direct demonstrations for digital agents at scale. *Preprint*, arXiv:2409.15637.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Alexander Pan, Erik Jones, Meena Jagadeesan, and Jacob Steinhardt. 2024a. Feedback loops with language models drive in-context reward hacking. *Preprint*, arXiv:2402.06627.

Jane Pan, He He, Samuel R. Bowman, and Shi Feng. 2024b. Spontaneous reward hacking in iterative self-refinement. *Preprint*, arXiv:2407.04549.

Jinlong Pang, Na Di, Zhaowei Zhu, Jiaheng Wei, Hao Cheng, Chen Qian, and Yang Liu. 2025. Token cleaning: Fine-grained data selection for llm supervised fine-tuning. *arXiv preprint arXiv:2502.01968*.

Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar, Aafaq Khan, and Arsalan Shahid. 2024. The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *arXiv preprint arXiv:2408.13296*.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*.

Dean A Pomerleau. 1991. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. 2025. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *Preprint*, arXiv:2411.02337.

Chongli Qin and Jost Tobias Springenberg. 2025. Supervised fine tuning on curated data is reinforcement learning (and can be improved). *arXiv preprint arXiv:2507.12856*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Yun Qu, Qi Wang, Yixiu Mao, Vincent Tao Hu, Björn Ommer, and Xiangyang Ji. 2025. Can prompt difficulty be online predicted for accelerating rl finetuning of reasoning models? *arXiv preprint arXiv:2507.04632*.

Shanghaoran Quan. 2025. Automatically generating numerous context-driven sft data for llms across diverse granularity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25074–25082.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Leonardo Ranaldi and Andre Freitas. 2024. Self-refine instruction-tuning for aligning reasoning in language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, page 2325–2347. Association for Computational Linguistics.

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. 2025. Androidworld: A dynamic benchmarking environment for autonomous agents. *Preprint*, arXiv:2405.14573.

Mathieu Rita, Florian Strub, Rahma Chaabouni, Paul Michel, Emmanuel Dupoux, and Olivier Pietquin. 2024. Countering reward over-optimization in llm with demonstration-guided reinforcement learning. *Preprint*, arXiv:2404.19409.

Stéphane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Maohao Shen, Guangtao Zeng, Zhenting Qi, Zhang-Wei Hong, Zhenfang Chen, Wei Lu, Gregory Wornell, Subhro Das, David Cox, and Chuang Gan. 2025. Satori: Reinforcement learning with chain-of-action-thought enhances llm reasoning via autoregressive search. *Preprint*, arXiv:2502.02508.

Wentao Shi, Mengqi Yuan, Junkang Wu, Qifan Wang, and Fuli Feng. 2025. Direct multi-turn preference optimization for language agents. *Preprint*, arXiv:2406.14868.

Vaishnavi Shrivastava, Ahmed Awadallah, Vidhisha Balachandran, Shivam Garg, Harkirat Behl, and Dimitris Papailiopoulos. 2025. Sample more to think less: Group filtered policy optimization for concise reasoning. *arXiv preprint arXiv:2508.09726*.

Yifan Song, Weimin Xiong, Xiutian Zhao, Dawei Zhu, Wenhao Wu, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. 2024a. Agentbank: Towards generalized llm agents via fine-tuning on 50000+ interaction trajectories. *Preprint*, arXiv:2410.07706.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024b. Trial and error: Exploration-based trajectory optimization for llm agents. *Preprint*, arXiv:2403.02502.

Moritz Stephan, Alexander Khazatsky, Eric Mitchell, Annie S Chen, Sheryl Hsu, Archit Sharma, and Chelsea Finn. 2024. Rlvf: Learning from verbal feedback without overgeneralization. *Preprint*, arXiv:2402.10893.

Xin Sun, Jianan Xie, Zhongqi Chen, Qiang Liu, Shu Wu, Yuehe Chen, Bowen Song, Weiqiang Wang, Zilei Wang, and Liang Wang. 2025. Divide-then-align: Honest alignment based on the knowledge boundary of rag. *Preprint*, arXiv:2505.20871.

Quanwei Tang, Sophia Yat Mei Lee, Junshuang Wu, Dong Zhang, Shoushan Li, Erik Cambria, and Guodong Zhou. 2025. A comprehensive graph framework for question answering with mode-seeking preference alignment. *Preprint*, arXiv:2506.17951.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. 2024. A survey on self-evolution of large language models. *arXiv preprint arXiv:2404.14387*.

Guiyao Tie, Zeli Zhao, Dingjie Song, Fuyang Wei, Rong Zhou, Yurou Dai, Wen Yin, Zhejian Yang, Jiangyue Yan, Yao Su, and 1 others. 2025. A survey on post-training of large language models. *arXiv e-prints*, pages arXiv–2503.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. 2024a. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *Preprint*, arXiv:2410.01560.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. 2024b. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *Advances in Neural Information Processing Systems*, 37:34737–34774.

Dheeraj Vattikonda, Santhoshi Ravichandran, Emiliano Penaloza, Hadi Nekoei, Megh Thakkar, Thibault Le Sellier de Chezelles, Nicolas Gontier, Miguel Muñoz-Mármol, Sahar Omidi Shayegan, Stefania Raimondo, Xue Liu, Alexandre Drouin, Laurent Charlin, Alexandre Piché, Alexandre Lacoste, and Massimo Caccia. 2025. How to train your llm web agent: A statistical diagnosis. *Preprint*, arXiv:2507.04103.

Huaijie Wang, Shibo Hao, Hanze Dong, Shenao Zhang, Yilin Bao, Ziran Yang, and Yi Wu. 2024a. Offline reinforcement learning for llm multi-step reasoning. *Preprint*, arXiv:2412.16145.

Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2023. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. *Preprint*, arXiv:2310.03731.

Kuan Wang, Yadong Lu, Michael Santacroce, Yeyun Gong, Chao Zhang, and Yelong Shen. 2024b. Adapting llm agents with universal feedback in communication. *Preprint*, arXiv:2310.01444.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. 2024c. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *Preprint*, arXiv:2312.08935.

Zhenting Wang, Guofeng Cui, Yu-Jhe Li, Kun Wan, and Wentian Zhao. 2025a. Dump: Automated distribution-level curriculum learning for rl-based llm post-training. *arXiv preprint arXiv:2504.09710*.

Zhichao Wang, Bin Bi, Zixu Zhu, Xiangbo Mao, Jun Wang, and Shiyu Wang. 2024d. Uft: Unifying fine-tuning of sft and rlhf/dpo/una through a generalized implicit reward function. *arXiv preprint arXiv:2410.21438*.

Zilong Wang, Jingfeng Yang, Sreyashi Nag, Samarth Varshney, Xianfeng Tang, Haoming Jiang, Jingbo Shang, and Sheikh Muhammad Sarwar. 2025b. Rro: Llm agent optimization through rising reward trajectories. *Preprint*, arXiv:2505.20737.

Hui Wei, Zihao Zhang, Shenghua He, Tian Xia, Shijia Pan, and Fei Liu. 2025a. Plangenllms: A modern survey of llm planning capabilities. *Preprint*, arXiv:2502.11221.

Quan Wei, Chung-Yiu Yau, Hoi-To Wai, Yang Katie Zhao, Dongyeop Kang, Youngsuk Park, and Mingyi Hong. 2025b. Roste: An efficient quantization-aware supervised fine-tuning approach for large language models. *Preprint*, arXiv:2502.09003.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I. Wang. 2025c. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *Preprint*, arXiv:2502.18449.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2024. Magicoder: Empowering code generation with oss-instruct. *Preprint*, arXiv:2312.02120.

Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. 2025. On the generalization of sft: A reinforcement learning perspective with reward rectification. *arXiv preprint arXiv:2508.05629*.

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. 2024. Agentgym: Evolving large language model-based agents across diverse environments. *Preprint*, arXiv:2406.04151.

Yu Xia, Jingru Fan, Weize Chen, Siyu Yan, Xin Cong, Zhong Zhang, Yaxi Lu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2025a. Agentrm: Enhancing agent generalization with reward modeling. *Preprint*, arXiv:2502.18407.

Yu Xia, Yiran Shen, Junda Wu, Tong Yu, Sungchul Kim, Ryan A. Rossi, Lina Yao, and Julian McAuley. 2025b. Sand: Boosting llm agents with self-taught action deliberation. *Preprint*, arXiv:2507.07441.

Qiming Xie, Zengzhi Wang, Yi Feng, and Rui Xia. 2024. Ask again, then fail: Large language models' vacillations in judgment. *Preprint*, arXiv:2310.02174.

Zhihui Xie, Jie chen, Liyu Chen, Weichao Mao, Jingjing Xu, and Lingpeng Kong. 2025. Teaching language models to critique via reinforcement learning. *Preprint*, arXiv:2502.03492.

Wei Xiong, Jiarui Yao, Yuhui Xu, Bo Pang, Lei Wang, Doyen Sahoo, Junnan Li, Nan Jiang, Tong Zhang, Caiming Xiong, and 1 others. 2025a. A minimalist approach to llm reasoning: from rejection sampling to reinforce. *arXiv preprint arXiv:2504.11343*.

Weimin Xiong, Yifan Song, Qingxiu Dong, Bingchan Zhao, Feifan Song, Xun Wang, and Sujian Li. 2025b. Mpo: Boosting llm agents with meta plan optimization. *Preprint*, arXiv:2503.02682.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. 2024. Watch every step! llm agent learning via iterative step-level process refinement. *Preprint*, arXiv:2406.11176.

Hongshen Xu, Zichen Zhu, Situo Zhang, Da Ma, Shuai Fan, Lu Chen, and Kai Yu. 2024a. Rejection improves reliability: Training llms to refuse unknown questions using rl from knowledge feedback. *Preprint*, arXiv:2403.18349.

Ran Xu, Wenqi Shi, Yuchen Zhuang, Yue Yu, Joyce C. Ho, Haoyu Wang, and Carl Yang. 2025a. Collab-rag: Boosting retrieval-augmented generation for complex question answering via white-box and black-box llm collaboration. *Preprint*, arXiv:2504.04915.

Tianyang Xu, Shujin Wu, Shizhe Diao, Xiaoze Liu, Xingyao Wang, Yangyi Chen, and Jing Gao. 2024b. Sayself: Teaching llms to express confidence with self-reflective rationales. *Preprint*, arXiv:2405.20974.

Yifan Xu, Xiao Liu, Xinghan Liu, Zhenyu Hou, Yueyan Li, Xiaohan Zhang, Zihan Wang, Aohan Zeng, Zhengxiao Du, Wenyi Zhao, Jie Tang, and Yuxiao Dong. 2024c. Chatglm-math: Improving math problem-solving in large language models with a self-critique pipeline. *Preprint*, arXiv:2404.02893.

Yixuan Even Xu, Yash Savani, Fei Fang, and J. Zico Kolter. 2025b. Not all rollouts are useful: Down-sampling rollouts in llm reinforcement learning. *Preprint*, arXiv:2504.13818.

Yixuan Even Xu, Yash Savani, Fei Fang, and Zico Kolter. 2025c. Not all rollouts are useful: Down-sampling rollouts in llm reinforcement learning. *arXiv preprint arXiv:2504.13818*.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. 2025a. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*.

Shi-Qi Yan and Zhen-Hua Ling. 2025. Rpo: Retrieval preference optimization for robust retrieval-augmented generation. *Preprint*, arXiv:2501.13726.

Yibo Yan, Jiamin Su, Jianxiang He, Fangteng Fu, Xu Zheng, Yuanhuiyi Lyu, Kun Wang, Shen Wang, Qingsong Wen, and Xuming Hu. 2025b. A survey of mathematical reasoning in the era of multimodal large language model: Benchmark, method & challenges. *Preprint*, arXiv:2412.11936.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Guang Yang, Yu Zhou, Xiang Chen, Xiangyu Zhang, Terry Yue Zhuo, and Taolue Chen. 2024. Chain-of-thought in neural code generation: From and for lightweight language models. *Preprint*, arXiv:2312.05562.

Jian Yang, Wei Zhang, Jiaxi Yang, Yibo Miao, Shanghaoran Quan, Zhenhe Wu, Qiyao Peng, Liqun Yang, Tianyu Liu, Zeyu Cui, Binyuan Hui, and Junyang Lin. 2025b. Multi-agent collaboration for multilingual code instruction tuning. *Preprint*, arXiv:2502.07487.

Zhengdong Yang, Shuichiro Shimizu, Yahan Yu, and Chenhui Chu. 2025c. When large language models meet speech: A survey on integration approaches. *Preprint*, arXiv:2502.19548.

Jiarui Yao, Yifan Hao, Hanning Zhang, Hanze Dong, Wei Xiong, Nan Jiang, and Tong Zhang. 2025. Optimizing chain-of-thought reasoners via gradient variance minimization in rejection sampling and rl. *arXiv preprint arXiv:2505.02391*.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran

Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. 2024. Retroformer: Retrospective large language agents with policy gradient optimization. *Preprint*, arXiv:2308.02151.

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2024. Agent lumos: Unified and modular training for open-source language agents. *Preprint*, arXiv:2311.05657.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. 2025. Agent-r: Training language model agents to reflect via iterative self-training. *Preprint*, arXiv:2501.11425.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023. Agenttuning: Enabling generalized agent abilities for llms. *Preprint*, arXiv:2310.12823.

Chuheng Zhang, Wei Shen, Li Zhao, Xuyun Zhang, Lianyong Qi, Wanchun Dou, and Jiang Bian. 2024a. Policy filtration in rlhf to fine-tune llm for code generation.

Fan Zhang, Hao Chen, Zhihong Zhu, Ziheng Zhang, Zhenxi Lin, Ziyue Qiao, Yefeng Zheng, and Xian Wu. 2025a. A survey on foundation language models for single-cell biology. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 528–549, Vienna, Austria. Association for Computational Linguistics.

Jianguo Zhang, Tian Lan, Rithesh Murthy, Zhiwei Liu, Weiran Yao, Ming Zhu, Juntao Tan, Thai Hoang, Zuxin Liu, Liangwei Yang, Yihao Feng, Shirley Kokane, Tulika Awalgaonkar, Juan Carlos Niebles, Silvio Savarese, Shelby Heinecke, Huan Wang, and Caiming Xiong. 2024b. Agentohana: Design unified data and training pipeline for effective agent learning. *Preprint*, arXiv:2402.15506.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, Zhiwei Liu, Yihao Feng, Tulika Awalgaonkar, Rithesh Murthy, Eric Hu, Zeyuan Chen, Ran Xu, Juan Carlos Niebles, Shelby Heinecke, and 3 others. 2024c. xlam: A family of large action models to empower ai agent systems. *Preprint*, arXiv:2409.03215.

Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai Tian, Guoli Jia, Pengfei Li, and 1 others. 2025b. A survey of reinforcement learning for large reasoning models. *arXiv preprint arXiv:2509.08827*.

Shijie Zhang, Guohao Sun, Kevin Zhang, Xiang Guo, and Rujun Guo. 2025c. Clpo: Curriculum learning meets policy optimization for lactlm reasoning. *arXiv preprint arXiv:2509.25004*.

Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. 2025d. On-policy rl meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting. *arXiv preprint arXiv:2508.11408*.

Xuechen Zhang, Zijian Huang, Yingcong Li, Chenshun Ni, Jiasi Chen, and Samet Oymak. 2025e. Bread: Branched rollouts from expert anchors bridge sft & rl for reasoning. *arXiv preprint arXiv:2506.17211*.

Haizhong Zheng, Yang Zhou, Brian R Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. 2025. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. *arXiv preprint arXiv:2506.02177*.

Victor Zhong, Dipendra Misra, Xingdi Yuan, and Marc-Alexandre Côté. 2024. Policy improvement using language feedback models. *Preprint*, arXiv:2402.07876.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, and 1 others. 2023. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021.

Qinhao Zhou, Zihan Zhang, Xiang Xiang, Ke Wang, Yuchuan Wu, and Yongbin Li. 2024a. Enhancing the general agent capabilities of low-parameter llms through tuning and multi-branch reasoning. *Preprint*, arXiv:2403.19962.

Runlong Zhou, Simon S. Du, and Beibin Li. 2024b. Reflect-rl: Two-player online rl fine-tuning for lms. *Preprint*, arXiv:2402.12621.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. 2024c. Archer: Training language model agents via hierarchical multi-turn rl. *Preprint*, arXiv:2402.19446.

Yujia Zhou, Zheng Liu, and Zhicheng Dou. 2024d. Assistrag: Boosting the potential of large language models with an intelligent information assistant. *Preprint*, arXiv:2411.06805.

Kun Zhu, Xiaocheng Feng, Xiyuan Du, Yuxuan Gu, Weijiang Yu, Haotian Wang, Qianglong Chen, Zheng Chu, Jingchang Chen, and Bing Qin. 2024. An information bottleneck perspective for effective noise filtering on retrieval-augmented generation. *Preprint*, arXiv:2406.01549.

Wenhong Zhu, Ruobing Xie, Rui Wang, Xingwu Sun, Di Wang, and Pengfei Liu. 2025a. Proximal supervised fine-tuning. *arXiv preprint arXiv:2508.17784*.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. 2025b. The surprising effectiveness of negative reinforcement in llm reasoning. *arXiv preprint arXiv:2506.01347*.

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. 2025c. Knowagent: Knowledge-augmented planning for llm-based agents. *Preprint*, arXiv:2403.03101.

Zhihong Zhu, Yunyan Zhang, Xianwei Zhuang, Fan Zhang, Zhongwei Wan, Yuyan Chen, Qingqing Long, Yefeng Zheng, and Xian Wu. 2025d. Can we trust AI doctors? a survey of medical hallucination in large language and large vision-language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 6748–6769, Vienna, Austria. Association for Computational Linguistics.

Table 1: Objective comparison across training paradigms. Here for integration, (SFT →) RL means modifying RL objective based on SFT objective, and vice versa.

Method	Paper	Objective / Gradient Expression	Data	Integration
DFT	Wu et al. (2025)	$\mathbb{E}_{(x_i,y_i)\sim\mathcal{D}}[-\mathrm{sg}(\pi_\theta(y_i\|x_i))\nabla_\theta \log \pi_\theta(y_i\|x_i)]$	Offline dataset	(RL→) SFT
Prefix Sampling	Huang et al. (2025)	$-\sum_{y_1,\cdots,y_{n-1},y_n^{<L}} \alpha\nabla_\theta \log \pi_\theta(y_i^t\|x, y_i^{<t})$ $-\sum_{y_n^{<L}} \beta\nabla_\theta \log \pi_\theta(y_n^t\|x, y_n^{<t})$	Offline dataset + online rollout	RL + SFT
CHORD	Zhang et al. (2025d)	$\mathcal{L}_{\mathrm{CHORD}}(\theta) = (1-\mu)\mathcal{L}_{\mathrm{GRPO}}(\theta) + \mu\mathcal{L}_{\mathrm{SFT}-\phi}(\theta)$, where $\mathcal{L}_{\mathrm{SFT}-\phi} = -\mathbb{E}_{(x,y^*)\sim\mathcal{D}_{\mathrm{SFT}}}[\sum_{t=1}^{\|y^*\|} \phi(y_t^*;\pi_\theta)\cdot\log\pi_\theta(y_t^* \mid x, y_{<t}^*)]$, and $\phi(y_t^*;\pi_\theta) = \pi_\theta(y_t^*\|x, y_{<t}^*)(1-\pi_\theta(y_t^*\|x, y_{<t}^*))$	Offline dataset + online rollout	RL + SFT
UFT	Liu et al. (2025b)	$\mathcal{J}^{\mathrm{UFT}} = \mathbb{E}_{l, x_l=x_l^*} [\mathcal{J}^{\mathrm{value}}((x_h,y_h)_{h=l}^{H-1})$ $_{(x_h,y_h)_{h=l}^{H-1}\sim\pi}$ $-\beta\sum_{h=l}^{H-1}\mathrm{KL}(\pi(\cdot\mid x_h)\,\|\,\pi^{\mathrm{ref}}(\cdot\mid x_h)) + \beta\sum_{h=0}^{l-1}\log\pi(y_h^*\mid x_h^*)]$	Offline dataset + online rollout	RL + SFT
Proximal SFT	Zhu et al. (2025a)	$L^{\mathrm{PSFT}}(\theta) = \mathbb{E}_{(x_t,y_t)\sim\mathcal{D}}\left[\min\left(\frac{\pi_\theta(y_t\|x_t)}{\pi_{\theta_{\mathrm{old}}}(y_t\|x_t)}, \mathrm{clip}\left(\frac{\pi_\theta(y_t\|x_t)}{\pi_{\theta_{\mathrm{old}}}(y_t\|x_t)}, 1-\epsilon,\ 1+\epsilon\right)\right)\right]$	Offline dataset	(RL →) SFT
UFT-SFT	Wang et al. (2024d)	$L_{\text{UFT-SFT}}(\pi_\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left\{\left[\sigma\left(\beta\log\frac{\pi_\theta(y\|x)}{\pi_{\mathrm{ref}}(y\|x)}\right)-1\right]^2\right\}$	Offline dataset	(RL →) SFT
iw-SFT	Qin and Springenberg (2025)	$\mathcal{L}_{\mathrm{iw-SFT}} = \mathbb{E}_{\tau\in\mathcal{D}^+}[-\frac{q(\tau)}{\pi_{\mathrm{ref}}(\tau)}\log p(\tau;\theta)]$	Offline dataset	(RL →) SFT
VAR	Du et al. (2025b)	$\mathcal{L} = -\mathbb{E}[\frac{\pi_{\mathrm{ref}}(y\|x)\exp(\frac{1}{\lambda}r(x,y)}{Z(x)}\log\pi_\theta(y\|x)]$	Offline dataset	SFT
HPT	Lv et al. (2025)	$\mathcal{L} = \alpha\mathcal{L}_{\mathrm{RL}} + \beta\mathcal{L}_{\mathrm{SFT}}$	Offline dataset + online rollout	RL + SFT
RPO	Liu et al. (2024)	$\mathcal{L}_{\mathrm{RPO}}(\theta) = \eta\beta\cdot\mathbb{E}_{x\sim d_0, y^0\sim\pi^{\mathrm{base}}(\cdot\|x)}[-\log\pi_\theta(y^0\|x)] + \mathcal{L}_{\mathcal{D}}(\beta\cdot\log\frac{\pi_\theta}{\pi_{\mathrm{ref}}})$, where $\mathcal{L}_{\mathcal{D}}(r) = -\widehat{\mathbb{E}}_{\mathcal{D}}\left[p_i\log(\sigma(r(x_i,y_i^1)-r(x_i,y_i^0))) + (1-p_i)\log(\sigma(r(x_i,y_i^0)-r(x_i,y_i^1)))\right]$	Offline (preference) dataset	SFT + DPO
SRFT	Fu et al. (2025b)	$\mathcal{L}_{\mathrm{SRFT}}(\theta) = \mathcal{L}_{\mathrm{SFT}}^{\mathrm{demo}}(\theta) + \mathcal{L}_{\mathrm{RL}}^{\mathrm{demo}}(\theta) + \mathcal{L}_{\mathrm{RL}}^{\mathrm{self-rollout}}(\theta)$, where $\mathcal{L}_{\mathrm{RL}}^{\mathrm{self-rollout}}$ stands for SFT loss on new rollouts from policy model, and GRPO advantage estimation is used in $\mathcal{L}_{\mathrm{RL}}^{\mathrm{demo}}$ with concatenated group from offline dataset and online rollout	Offline dataset + online rollout	SFT + RL
Q-SFT	Hong et al. (2024)	$\mathcal{L}_{\mathrm{CE}}(\phi) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\log\pi_\phi(y\|x)]$ $\mathcal{L}_{\mathrm{TD}}(\theta) = \mathbb{E}_{(x,y,r,x')\sim\mathcal{D}}\left[\left(r+\gamma\max_{y'}Q_{\bar\theta}(x',y') - Q_\theta(x,y)\right)^2\right]$	Offline dataset	SFT + Q learning
IRL-RFT	Li et al. (2024a)	$\frac{1}{\beta}\nabla_\theta r(x_{t,k},y_{t,k};\theta_{t,k}) - \frac{1}{\beta}\nabla_\theta r(x_{t,k},\tilde y_{t,k};\theta_{t,k})$, where $y\sim\pi_{\mathrm{expert}}(\cdot\|x_{t,k})$, $\tilde y\sim\pi^t(\cdot\|x_{t,k})$, $\pi^t(y\|x)\propto\exp(r(x,y;\theta_{t,K}))$	Online rollout	RL
SASR	Chen et al. (2025b)	$\mathcal{L}(\theta) = \frac{1}{S}\sum_{s=1}^{S}[(1-I(t))\cdot\mathcal{L}_{\mathrm{SFT}}(\theta) + I(t)\cdot\mathcal{L}_{\mathrm{GRPO}}(\theta)]$	Offline dataset + online rollout	SFT + RL
IFT	Hua et al. (2024)	$\mathcal{L}\left(\hat T_\theta(s_n^*,\rho_0)\right) = -\sum_{i=n}^{N}\log\mathcal{T}_\theta(a_i^*,\delta_\theta(s_i^*))$	Offline dataset	RL
NFT	Chen et al. (2025a)	$\mathcal{L}_{\mathcal{D}}^{\mathrm{NFT}}(\theta) = -\sum_{x,y,r}\omega(x)\sum_t\left[r\log R_\theta^t(x,y) + (1-r)\log\max_v\left(\frac{1-\hat r_x R_\theta^t(x,y)}{1-\hat r_x},\ \epsilon\right)\right]$, where $R_\theta^t(x,y) = \frac{\pi_\theta^+(y_t\|x,y_{<t})}{\pi(y_t\|x,y_{<t})}$, $\hat r_x = \frac{1}{K}\sum_{y\|x} r(x,y)$	Online rollout	(SFT →) RL
SRL	Deng et al. (2025)	$R = \frac{2\sum_{(i,j,n)\in\mathrm{MatchingBlocks}} n}{\|S_1\|+\|S_2\|}$, $r(\mathbf{y}'_{\mathrm{step}_{p_k}}, \mathbf{y}_{\mathrm{step}_{p_k}}) = \begin{cases} R(\mathbf{y}'_{\mathrm{step}_{p_k}}, \mathbf{y}_{\mathrm{step}_{p_k}}), & \text{if } \mathbf{y}' \text{ follows format,} \\ -1, & \text{otherwise.} \end{cases}$	Offline dataset + online rollout	SFT + RL

A More on Comparing and Combining SFT and RL

We summarize the main notation used throughout this section and Table 1. For ease of reference, commonly used acronyms are listed in the acronym table, and a consolidated summary of symbols is provided in Table 2.

A.1 Notation

The offline training dataset used in SFT and related settings is denoted by $\mathcal{D} = \{(x_i, y_i)\}$, where x_i represents the input prompt and y_i denotes the corresponding model-generated response. We use π_θ to denote a policy (language) model parameterized by θ, and $\pi_\theta(y_i \mid x_i)$ to denote the conditional probability of generating response y_i given prompt x_i. The reference policy is denoted by π^{ref}, which is typically a frozen base model or an SFT checkpoint.

In the RL setting, $r(x_i, y_i)$ denotes the reward assigned to a prompt-response pair (x_i, y_i), and $Q_\theta(x, y)$ represents the action-value (Q) function under policy π_θ, with state x (prompt) and action y (response). We use $\mathcal{J}$ to denote cumulative reward–based objectives.

The sigmoid function is defined as $\sigma(x) = 1/(1 + e^{-x})$. The operator $\text{sg}(\cdot)$ denotes the stop-gradient operation, which preserves the forward-pass value while preventing gradient propagation during backpropagation. Additional symbols, operators, and hyperparameters appearing in Table 1 are summarized in Table 2 for completeness.

A.2 Preliminaries

We consider the optimization of Pretrained LLMs, also referred to as FMs, which are large-scale pretrained models adapted to downstream tasks such as QA, reasoning with CoT, and database augmentation tasks such as RAG. Training typically proceeds in multiple stages, starting with SFT on curated prompt-response pairs, which can be viewed as a form of BC from expert demonstrations.

Beyond SFT, model alignment and performance are often further improved using RL techniques, particularly RLHF, where a learned or human-provided reward signal guides optimization. Popular policy-gradient–based algorithms include PPO and its variants such as GRPO, which stabilize updates through clipping or relative normalization. Alternative optimization frameworks include preference-based methods such as DPO and rejection-based methods such as RFT, which bypass explicit reward modeling. Information-theoretic objectives, such as GEM, and regularization via the KL divergence are commonly used to control deviation from a reference policy.

The RL process itself can be formalized as a MDP, defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, H)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P denotes the transition dynamics, r is the reward function, and H is the horizon length, which may be infinite under idealized analysis. The policy $\pi_\theta(\cdot \mid x)$ represents an LLM that induces a distribution over responses conditioned on an input prompt x. This formulation naturally extends to multi-modal settings, including VLM, and other structured decision-making scenarios.

Acronyms from the Main Paper

BC Behavior Cloning. 3

CoT Chain-of-Thought. 1, 6, 7

DPO Direct Preference Optimization. 5, 7

FM Foundation Model. 6, 7

GEM Entropic Distribution Matching. 3

GRPO Group Relative Policy Optimization. 3, 5, 7

KL Kullback–Leibler. 4

LLM Large Language Model. 1–8

PPO Proximal Policy Optimization. 3, 6, 7

QA Question Answering. 6

RAG Retrieval-Augmented Generation. 6

RFT Rejection Fine-Tuning. 6–8

RL Reinforcement Learning. 1–8

RLHF Reinforcement Learning from Human Feedback. 4, 5

SFT Supervised Fine-Tuning. 1–8

VLMs Vision Language Models. 5

Symbol	Description
Models and Policies	
π_θ, π_ϕ	Current policy/model parameterized by θ or ϕ
$\pi_{\text{ref}}, \pi_{\text{base}}$	Reference or base policy (typically frozen)
π_{old}	Policy from a previous optimization step
π_{expert}	Expert policy used for imitation or IRL
Data and Sequences	
x, y	Input prompt and generated response
$y_t, y_{<t}$	Token at time t and the prefix sequence
$\mathcal{D}, \mathcal{D}^+$	Dataset (offline, SFT, or preference-based)
τ	Trajectory or full sequence (x, y)
Functions and Objectives	
$\mathcal{L}, \mathcal{J}$	Loss or objective function
$r(x, y), R$	Reward function or scalar reward value
$Q(x, y)$	Action-value function (Q-function)
$Z(x)$	Partition function (normalization constant)
$\sigma(\cdot)$	Sigmoid activation function
Operators and Hyperparameters	
$\mathbb{E}$	Mathematical expectation
∇_θ	Gradient with respect to parameters θ
$\text{sg}(\cdot)$	Stop-gradient operator
$\text{KL}(\cdot\|\cdot)$	Kullback-Leibler divergence
α, β, μ, η	Weighting coefficients or temperature parameters
ϵ	Clipping or threshold constant

Table 2: Summary of Notation and Symbols

B Application Survey Details

This appendix section provides detailed information on our survey of LLM applications across professional domains, including the datasets, evaluation criteria, and publication trends that underpin our analysis. The goal is to contextualize the results presented in the main text by describing (i) how different task domains vary in input complexity, reasoning demands, and output characteristics, (ii) our methodology for categorizing research publications by domain using benchmark-oriented searches, and (iii) observed trends in publication volume, methodological approaches, and model usage over time. By presenting these details, we aim to offer a transparent and reproducible account of the survey process, enabling deeper insight into the evolving landscape of LLM applications.

B.1 Applications Across Professional Domains

To characterize the difficulty of each domain, we consider three qualitative factors: input complexity, intermediate reasoning requirements (chains of thought, CoT), and output complexity. Table 3 summarizes these characteristics across focompur representative domains. Note that Section 5 focuses on papers addressing text-only tasks. We make this choice because text-only tasks enable clearer side-by-side comparison, whereas multimodal settings introduce additional sources of uncertainty in model selection and fusion techniques, which lie outside the scope of this work. Nevertheless, we acknowledge that numerous impactful works closely related to the surveyed area extend beyond text-only modalities. For further references, we direct readers to Section D, which highlights some recently published surveys as a starting point.

General QA assesses fact-checking and language understanding with variable-length inputs and moderate reasoning, prioritizing accuracy over verbosity. Mathematical tasks demand precise answers supported by extended logical reasoning. Agentic tasks require rich environmental context to enable multi-step planning before producing compact outputs. Code generation relies on context-heavy inputs and moderate reasoning to infer intent and produce syntactically correct, semantically coherent code.

Across these domains, the overarching question is how to optimize performance given varying input scales, reasoning depths, and output complexities. The relative weight of these factors differs by task, reflecting the diverse requirements and shifting priorities of professional LLM applications.

B.2 Benchmark-Oriented Paper Search

To estimate publication counts for each domain, we conducted a benchmark-oriented keyword search over arXiv preprints in the Computer Science categories from 1 January 2023 to 30 June 2025. The arXiv metadata dataset (Cornell-University/arxiv) is provided under the Creative Commons CC0 1.0 Universal Public Domain Dedication, which applies to the dataset's metadata as hosted on Kaggle. Original arXiv content remains subject to the licenses specified by individual authors.

Table 3: Summary of LLM application domains and their characteristics.

Domain	Input	CoT / Reasoning	Output
General QA	Moderate	Moderate	Moderate
Mathematical	Moderate	Long	Moderate
Agentic	Long	Long(Multi-Step)	Moderate
Code Gen.	Long	Moderate	Long

We accessed scientific articles via Google Cloud Public Datasets (`gs://arxiv-dataset`). The initial dataset comprises approximately 195K papers, with 64K, 78K, and 52K papers from 2023, 2024, and 2025, respectively. Our approach is motivated by the observation that recent LLM post-training research consistently relies on a set of widely used, standardized datasets that facilitate reproducible and comparable evaluations across methods. As this practice has become increasingly common, dataset mentions serve as a reasonable proxy for assigning papers to topical domains.

Following Table 7 of Du et al. (2025a), we use 26 datasets spanning four domains as domain-specific query keys. A paper is assigned to a domain if it contains at least five mentions of any dataset associated with that domain. Mutual exclusivity is not enforced: although relatively rare, papers that evaluate across multiple domains are counted in every applicable category.

The datasets used as search keys are listed in Table 4, and the resulting paper counts for each category are shown in Table 5. We adopt a threshold of five keyword occurrences because a paper that genuinely employs a dataset typically introduces it, reports results on it, and provides comparative or analytical discussion, making five mentions a conservative filter against purely methodological contributions. To project the full-year counts for 2025, we note that the first six months of 2023 and 2024 account for 47.38% and 49.81% of their respective annual publication totals. Accordingly, we estimate the 2025 full-year count by doubling the observed number from the first half of 2025.

To contextualize our estimation procedure, we acknowledge both its strengths and limitations. Our keyword-based counting method provides a transparent, scalable, and reproducible way to approximate the volume of research activity across domains, with higher keyword-mention thresholds generally yielding more reliable domain assignments. As shown in Table 5, the number of papers decreases consistently as the threshold increases, reflecting increasing specificity but reduced recall. We report results primarily using the threshold of keyword matches greater than five, though counts under alternative thresholds are also available. However, this approach also inherits several limitations: low thresholds may inflate counts by including papers that mention a dataset only in passing, while high thresholds may exclude legitimate work that uses a benchmark but references it sparsely. Additionally, cross-domain papers contribute to multiple categories, and naming variations may lead to undercounting unless normalized. Despite these caveats, the method remains a practical heuristic for capturing broad research trends in benchmark-driven LLM evaluation.

B.3 Detailed Paper Application Survey

B.3.1 LLMs for General QA Tasks

LLMs demonstrate strong performance in general question answering by enhancing reasoning capabilities, mitigating hallucinations, and effectively leveraging external knowledge. Current research emphasizes sequential sub-question reasoning, robust answer generation under uncertainty, and retrieval-augmented methods to improve information access. The following sections present representative approaches in each of these areas, with a focus on model-tuning strategies. For studies that do not involve model tuning, we refer readers to Ke et al. (2025).

Step-by-Step Reasoning. Most works decompose a question into sub-questions and train LLMs to answer them sequentially to produce more accurate CoTs. For instance, Chen et al. (2024) proposes generating sub-questions with GPT for each sentence in the ground-truth answer, while Ranaldi and Freitas (2024); Huang et al. (2024) use model rollouts to create preference data based on the quality of the final answers. Furthermore, Feng et al. (2024); Zhou et al. (2024d) align LLM behavior in deciding when to reflect, answer, or ask follow-up

Table 4: Datasets used as search keys for benchmark-oriented paper classification.

Agentic webshop, webarena, wind2web, miniwob++, scienceworld, alfworld, tdw-mat, c-wah, alfred, rlcard
QA hotpotqa, strategyqa, triviaqa, pubmedqa, musique, 2wikimultihopqa, qasper
Math gsm8k, asdiv, svamp, aime
Code swe-bench, humaneval, livecodebench, bird, intercodesql

questions.

Hallucination Management. Several works investigate how LLMs behave under uncertainty, particularly regarding hallucination. For instance, Li et al. (2024b) treats GPT generations as positive samples for RLHF, whereas Kang et al. (2024) suggests using conservative RL (assigning low scores when the model is unfamiliar) to mitigate hallucination during RLHF. Other approaches (Xie et al., 2024; Cheng et al., 2024; Xu et al., 2024a; Sun et al., 2025) propose alternative data pipelines for the RL stage, yielding more robust answers to challenging follow-ups and enabling refusal in cases of unknown scenarios. Moreover, Band et al. (2024); Xu et al. (2024b) equip the LLM with confidence estimation capabilities, typically synthesized from multiple rollouts and GPT-generated root causes.

LLM-based Retrieval. Mao et al. (2024); Hsu et al. (2024); Yan and Ling (2025); Xu et al. (2025a); Gao et al. (2025) post-train LLMs' query generation with retrieval-based rewards, leveraging them as query rewriters to improve retrieval effectiveness. Other studies (Jin et al., 2024; Zhu et al., 2024) propose alternative evaluation criteria for extracting key information from retrieved documents. Additionally, Tang et al. (2025) converts long documents into hierarchical graphs and applies RL to the policy model for RAG using mode-seeking DPO algorithms.

B.3.2 LLMs for Mathematical Tasks

Recent efforts have enhanced LLMs for mathematical reasoning to support multi-step logic, symbolic manipulation, and rigorous verification.

Math-Based Capabilities. Polu and Sutskever (2020); Lewkowycz et al. (2022); Azerbayev et al. (2023) fine-tuned LLMs on mathematics-specific corporato enhance mathematical reasoning and proof generation. Building on this, Luo et al. (2023) applied the Evol-Instruct framework to math question–answer datasets. Similarly, Yu et al. (2023); Toshniwal et al. (2024a) improved Q&A pairs via rephrasing, backward reasoning, and other data augmentation strategies. In parallel, Toshniwal et al. (2024b) distilled reasoning abilities from proprietary models by collecting successful solution trajectories. Additionally, Shao et al. (2024) used FastText-based filtering in Common Crawl and employed GRPO to further enhance problem-solving performance.

Rollout Selection and Verification. Cobbe et al. (2021) proposed training separate verifier models to assess the correctness of model-generated solutions, selecting answers with the highest verification scores. Huang et al. (2022) fine-tuned LLMs on high-confidence data, while Xu et al. (2024c) filtered successful self-generated solutions for iterative training. Wang et al. (2024c) introduced MCTS to generate rollouts and annotate intermediate reasoning steps with probabilities of reaching correct answers. Similarly, Guan et al. (2025) learned a value function from MCTS probabilities to guide exploration and DPO data selection, with Xu et al. (2024c) further refining DPO via a math-specific critic model. Finally, Shen et al. (2025) fine-tuned models to select actions during mathematical reasoning.

B.3.3 LLMs for Agentic Tasks

Agentic tasks challenge LLMs to act as autonomous decision-makers in dynamic environments, requiring continuous state perception, contextual integration, and multi-step planning. Success hinges on long-term reasoning, history man-

Table 5: Paper counts under different keyword-mention thresholds across four domains.

	QA			Math			Agentic			Code		
Keyword count	**23**	**24**	**25**	**23**	**24**	**25**	**23**	**24**	**25**	**23**	**24**	**25**
>0	771	1644	1308	12673	20405	13177	1310	1826	1373	334	1091	1302
>1	580	1321	1088	3870	7143	4889	330	541	437	209	767	921
>2	453	1022	858	1673	3295	2684	162	310	260	178	621	717
>3	384	861	730	932	1957	1953	125	234	207	148	549	604
>4	338	746	621	655	1391	1587	108	196	167	130	481	513
>5	**292**	**652**	**558**	**492**	**1098**	**1366**	**100**	**174**	**145**	**115**	**428**	**458**
>10	165	342	326	221	538	794	68	122	92	69	254	252
>20	56	114	90	97	195	302	34	71	46	37	126	93

agement, and adaptive feedback, as each action influences future outcomes.

Autonomous Task Execution. Schick et al. (2023); Qin et al. (2023); Zeng et al. (2023); Yin et al. (2024); Zhang et al. (2024b,c); Song et al. (2024a); Ou et al. (2024) developed custom pipelines to behavior-clone full trajectories with open-weight models, supporting task-specific capabilities, whereas Yao et al. (2024); Murty et al. (2025) fine-tuned instruction-following models using actor–reflector outputs from proprietary APIs. To improve execution reliability, Zhong et al. (2024) used LLM judges to filter low-quality actions, Fu et al. (2025a) excluded erroneous steps during loss computation, and Xia et al. (2025b) added reflective annotations. Extending this, Zhu et al. (2025c) introduced knowledge-based self-learning with dynamic memory. In reinforcement learning, Choudhury and Sodhi (2024) curated preference pairs by synthesizing reasoning traces for ground-truth actions, while others generated preference data iteratively: ETO (Song et al., 2024b) used successful and failed trajectories for DPO, and Yuan et al. (2025) applied MCTS with corrections at the first error step.

Self-Improvement and Feedback Integration. Some studies use sequential SFT followed by RL. Zhou et al. (2024b) leveraged LLM API reflections, while Lai et al. (2024) showed early RL is effective after initial SFT using a 70B teacher model. Hierarchical RL approaches assign rewards at the sentence or utterance level while updating policies at the token level (Zhou et al., 2024c; Hu et al., 2025b), with DMPO (Shi et al., 2025) adding step-wise discounting and Qi et al. (2025) using critic-based curriculum learning. OREO (Wang et al., 2024a) enables multi-step reasoning via maximum-entropy learning, complemented by IRL and PPO tuning (Deng et al., 2024) and reward calibration (Rita et al., 2024). Some works jointly train SFT and RL losses with variants including masked or weighted objectives, outcome- and step-based DPO, and pseudocode planning (Wang et al., 2024b; Xi et al., 2024; Xiong et al., 2024; Cao et al., 2025b).

Planning and Long-Horizon Control. Xiong et al. (2025b) proposed a hierarchical planning framework with a top-level planner fine-tuned on seed plans and optimized via DPO. Vattikonda et al. (2025) added Rejected Sampling Fine-Tuning (RFT) to further refine high-reward actions, while Wang et al. (2025b) adjusted exploration and prioritized critical steps for DPO. For test-time scaling, Zhou et al. (2024a) integrated general and agentic data (Zeng et al., 2023) with multi-path reasoning, and Xia et al. (2025a) combined policy fine-tuning with reward-model training, showing that beam search with explicit reward modeling improved performance.

B.3.4 LLMs for Code Generation Tasks

LLMs have shown remarkable potential in code generation, editing, and task-specific reasoning, which are often improved by structured code data, executable feedback, or multi-agent frameworks for more robust and verifiable outputs.

Code Generation Capabilities. Recent studies have explored methods to improve code generation with LLMs. Chaudhary (2023) employ self-instruction to synthesize training data, while Luo et al. (2025) generate harder queries by modifying constraints of seed questions. Wei et al. (2024) leverage open-source snippets to produce diverse and controllable instruction data. Yang et al. (2024) show that preceding CoT reasoning with code explanations enhances accuracy. More recently, Yang et al. (2025b) introduce a multi-agent framework in which an SFT model generates rollouts for DPO training.

Code Editing Capabilities. Code editing focuses on refining code that fails to meet its intended functionality, whether human-written or LLM-generated. Wei et al. (2025c) use human pull requests to guide GRPO-based specialization of LLMs. Chae et al. (2024); Jiang et al. (2025) generate refinement rollouts evaluated against unit tests. Dou et al. (2024) decompose code generation into verifiable subtasks for fine-grained alignment, while Ma et al. (2025) propose a four-stage pipeline combining rejection sampling with rule-based RL rewards. Xie et al. (2025) fine-tune an auxiliary LLM to produce verbal critiques that assist the primary model in correcting errors.

Code Reuse for Other Tasks. Code can also be leveraged to enhance a variety of tasks by serving as a verifiable and interpretable reward signal. For instance, Dong et al. (2024) use code execution to assess the instruction-following ability of LLMs. Nie et al. (2025) train an SLM to leverage LLMs via environment feedback by simulating workflows as executable Python code. Wang et al. (2023); Lai et al. (2023); Gou et al. (2024) perform supervised instruction tuning on code-augmented math or data science tasks.

B.4 Trend Analysis Details

This subsection covers a detailed analysis of the procedure-wise trend from surveyed papers, summarized in Figure 2.

Rapid Growth Across Domains. Between 2023 and 2025, research activity accelerates sharply across all major task domains, though the magnitude of growth varies substantially. QA studies more than double from 292 in 2023 to 652 in 2024 (+123%), with projections indicating continued expansion to 983 in 2025 (+118%). Math-related research grows even faster in absolute scale, rising from 492 to 1,098 (+123%) and then reaching 2,399 papers in 2025, which is a near 5× increase from 2023. Agent-focused work, while smaller in volume, shows steady expansion from 100 to 179 (+79%) and further to 261 (+46%). The most dramatic surge occurs in code-related studies, which climb from 115 in 2023 to 428 in 2024 (+272%) and nearly double again to 786 in 2025 (+84%). This trajectory reflects the rapid maturation of code-centric benchmarks, tools, and evaluation pipelines, and highlights coding as one of the most rapidly diversifying application domains for LLMs.

Convergence Toward Hybrid Training. The landscape of training methodologies undergoes a marked consolidation toward hybrid approaches that combine SFT with RL or other post-training strategies. In 2023, SFT dominates at 73.3%, with hybrid ("Both") methods representing only 20.0%. By 2024, hybrid training expands by 269%, becoming the most common approach at 73.8%, superseding pure SFT (19.1%) and growing further to 70.6% in 2025. This shift illustrates a collective movement toward more sophisticated multi-stage training pipelines that exploit complementary strengths of SFT and RL. Meanwhile, RL-only methods, which starts initially 6.7% of studies, gain modest traction, reaching 7.1% by 2024 and 11.8% by 2025, reflecting ongoing improvements in open-source RLHF frameworks and increased accessibility of preference data.

Shift from Proprietary to Open Models. A pronounced realignment in model choice accompanies these methodological changes. Between 2023 and 2025, reliance on proprietary API-based models declines sharply. The trend starts at 32.2% in 2023 to 19.9% in 2024, and down to 11.1% by 2025, where researchers pivot toward open-weight models and standardized benchmarks. Open-weight usage more than doubles, growing from 12.2% in 2023 to 17.5% in 2024 and reaching 25.0% in 2025. Benchmarks remain the most commonly used resource category but also increase steadily (48.9% to 61.1% over three years), underscoring the field's movement toward reproducible, transparent experimentation. In parallel, the use of human-curated or web-scraped datasets continues to shrink (6.7% to 2.8%), reflecting both improved benchmark coverage and growing concerns around legality, license compliance, and data provenance. Overall, these trends indicate a strong and persistent shift toward open, standardized, and reproducible workflows as community norms solidify.

C Hardware Requirements for SFT/RL

For researchers intending to perform practical experimentation with post-training alignment, consulting community-reported hardware guidance can be valuable. It is important to note, however, that GPU requirements vary substantially depending on model size, training approaches, e.g., full fine-tuning versus parameter-efficient methods such as LoRA (Hu et al., 2021) or QLoRA (Dettmers et al., 2023), choice of optimizer, and software frameworks. Moreover, these requirements continue to evolve alongside advances in

hardware and software tools.

For SFT, a commonly cited heuristic indicates that full fine-tuning of a transformer-based large language model requires approximately 16GB of VRAM per billion parameters when using half-precision (FP16). This high memory demand arises from the need to store gradients and optimizer states, and is substantially greater than the roughly 2GB per billion parameters typically needed for inference. Parameter-efficient approaches, such as LoRA and QLoRA, can reduce this requirement significantly, often into the range of 5–20GB on contemporary GPUs, depending on model size and precision settings[1].

For RL training, frameworks such as *verl* provide community-reported guidance on hardware requirements across different model scales. Practical experimentation indicates that RL generally demands substantially more memory than SFT due to additional overhead from rollout generation, policy gradients, and reward models. Heuristic estimates suggest that medium-sized models (1–3B parameters) can be trained on a single 80GB GPU using parameter-efficient methods like LoRA, while larger models (7–14B) typically require 2–4 such GPUs. Very large models (32B+) often necessitate 4–8 high-memory GPUs, with full-parameter training potentially exceeding 600GB of aggregate VRAM. Roughly, LoRA-based RL training consumes 15–25GB VRAM per billion parameters, compared with 16GB/B for full SFT fine-tuning, illustrating that RL generally incurs 1.5–3× higher memory demands per parameter. These numbers should be treated as approximate guidelines, as actual requirements depend on batch size, rollout length, optimizer choice, and memory-saving strategies such as gradient checkpointing or quantization[2].

It is essential to treat these hardware guidelines as approximate starting points rather than strict requirements. Actual resource needs will depend on factors including model architecture, task complexity, batch size, rollout length, and ongoing improvements in software and hardware technologies.

D More References

This section provides additional references that extend beyond the text-only scope emphasized in Section 5. These works highlight broader developments in multimodal learning, retrieval-augmented methods, hallucination mitigation, and domain-specific datasets and capabilities. Together, they offer a more comprehensive view of the rapidly evolving ecosystem surrounding LLM reasoning and application domains.

One of the most practically oriented directions is multimodal LLM research, which integrates modalities such as speech and vision beyond pure text. This line of work serves as a natural extension to the text-only tasks surveyed earlier, offering insights into how additional sensory channels introduce new modeling challenges and opportunities. For instance, Cui et al. (2025c) and Yang et al. (2025c) summarize methods for leveraging and aligning LLMs with speech-augmented multimodal data, enabling richer understanding and generation capabilities.

Beyond multimodality, another cluster of related literature addresses system-level improvements that refine how LLMs access information and maintain reliability. Abootorabi et al. (2025) survey advances in retrieval-augmented generation (RAG) across various modalities, providing a complementary perspective to reasoning-centric approaches. Meanwhile, Li et al. (2025a) investigates hallucination through the lens of knowledge boundaries, and Zhu et al. (2025d) explores hallucination in conjunction with multimodal trustworthiness. In parallel, He et al. (2025c) review methodologies for constructing SFT and RL datasets specifically tailored for reasoning, bridging data collection practices with model behavior.

Finally, several specialized surveys focus on domain-specific datasets and complex task settings, which enrich the understanding of how LLM reasoning varies across fields. For example, Yan et al. (2025b) discusses mathematical reasoning datasets and benchmarks that support structured and verifiable inference. Complementing this, Wei et al. (2025a) reviews LLM-based planning techniques applicable to both question answering and agentic tasks. Further extending the discussion to environments requiring perception–action loops, Hu et al. (2025a); Nguyen et al. (2025) examine multimodal–agent interactions, including GUI-based interfaces that illustrate how reasoning, perception, and action co-evolve. Together, these surveys broaden the contextual understanding of LLM capabilities beyond the domains highlighted in the main text.

[1] https://modal.com/blog/how-much-vram-need-fine-tuning

[2] https://verl.readthedocs.io/en/latest/perf/device_tuning.html

E Use of AI Assistants

We employed LLM-based tools exclusively for language polishing during manuscript preparation and for assisting with full-text filtering of SFT/RL-related papers for human review in the Applications section (Section 5). All outputs generated by LLMs were carefully reviewed by the authors, who take full responsibility for any potential errors or hallucinations.

The prompt used with `gpt-oss-120b` to identify and classify SFT/RL-related papers is provided in the following verbatim. For all prompt requests to this model, we use FP4 precision, a context length of 128k, a temperature of 0.8, and set top-k and top-p to 40 and 0.95, respectively.

Human readers subsequently verified all candidate papers to confirm their publication status and the relevance of their training paradigms (i.e., whether they involve SFT, RL, or both).

```
You are a research paper analyzer.
From the given scientific text, extract and classify the following:

1. The **proposed method** (ignore ablations).
2. The **comparison methods/baselines** the paper evaluates against.
3. The **datasets/benchmarks** used for evaluation.
4. The **training type** of the proposed method, exactly one of:
   - Reinforcement Learning (RL)
   - Supervised Fine-Tuning (SFT)
   - Both RL and SFT (SFT+RL)
   - Prompt Optimization
  - none-text Modality (vision, speech, multimodal, etc)
   - Other Methods
  - Survey (no new method, only summarization of others)

Output format:
<thought>
YOUR THOUGHT
</thought>

<answer>
{
  "method": "...",
  "baselines": ["..."],
  "benchmarks": ["..."],
 "training_type": one of ["SFT", "RL", "SFT+RL", "Prompt", "Modality", "Other", "Survey"]
}
</answer>
```