

COLLABORATIVE AI AGENTS

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Shirley (Yingxin) Wu

February 2026

© 2026 by Yingxin Wu. All Rights Reserved.

Re-distributed by Stanford University under license with the author.



This work is licensed under a Creative Commons Attribution-Noncommercial 3.0 United States License.

<https://creativecommons.org/licenses/by-nc/3.0/legalcode>

This dissertation is online at: <https://purl.stanford.edu/kt076zd0858>

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Jure Leskovec, Co-Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

James Zou, Co-Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Diyi Yang

Approved for the Stanford University Committee on Graduate Studies.

Kenneth Goodson, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format.

Abstract

Traditional Artificial Intelligence (AI) often conducts tasks in non-collaborative settings, where a single AI makes decisions given user-defined inputs. Modern AI is increasingly deployed in collaborative settings, where AI agents and humans work together in multiturn toward a shared goal. This paradigm of collaborative AI goes beyond building systems that aim to match or even surpass human-level capabilities; instead, it aims to empower creativity and intelligence that cannot be achieved by either side alone.

This thesis studies the following problem: How do we build capable agents that effectively collaborate with human users? Compared to non-collaborative agents, collaborative AI agents require proactive behaviors to understand user intent, reason about user queries and ambiguity, and provide assistance that helps users achieve their goals. We analyze common failure modes of agents in collaborative settings, draw connections to social science studies on human collaboration, and more importantly, identify what levels of behaviors and capabilities are needed to solve collaborative tasks.

Building upon these foundations, our recent work presents a unified framework that transforms non-collaborative AIs into collaborative agents. We propose a multiturn-aware objective that prioritizes long-term outcomes in multi-turn collaboration rather than single-turn task completion. We demonstrate that collaborative agents trained with this objective markedly outperform non-collaboratively trained agents across various benchmarks. More importantly, human users report more efficient, engaging, and creative interactions when working with collaborative agents.

This thesis further investigates how to build more capable and intelligent collaborative agents beyond the behavioral change from the multiturn-aware objective. We find that

agents often struggle with learning from experiences. An agent which has conducted multiple tasks may still fail to solve a similar one with the same mistakes. To address the limitations, we pioneer a research direction that enables agent system to learn from past failure and adapt to harder tasks. The key is for agent systems to conduct contrastive reasoning, which identifies the gap between collaboration sessions with good and bad outcomes, so that they can recognize and effectively address current limitations. We also uncover the benefits of scaling agentic systems and develop an optimization framework that aligns individual agents with the shared goal. Together, these works enhance multiturn collaboration with stronger system-level capabilities in collaborating with users.

Lastly, we conclude with open questions in collaborative AI and advocate for democratizing collaborative agents that are not merely governed by humans, but also create more value with humans.

Acknowledgments

People we meet change our lives' trajectories, and many have profoundly changed mine.

Since the very beginning, my mom, **Dandan Chen**, and my dad, **Mingzhi Wu**, have shaped who I am. Their casual parenting style and decision of having me in boarding schools early on made me independent. I learned, hopefully, from my mom's persistence and eloquence, and from my dad's erudition and diligence. I thank my emotional support person, **Katherine Ke**, who I met in high school. Katherine is so kind that my mom treats her as another daughter and gives her more lucky money than she gives me. Katherine shows me the ultimate strength by making jokes in the face of bad things, which has proven to be an effective way to cope with all kinds of rejections and life challenges.

In my undergraduate, I worked with **Xiang Wang**, **Xiangnan He**, and **Tat-Seng Chua**. From them, I learned how solid research should be done. I thank them for opening the door for me to see how the world evolved and should evolve with AI. During this time, **Xu Chen** was a great companion and we often studied together until late at night.

Getting into Stanford is life-changing. **Jure Leskovec** and **James Zou**, my incredibly supportive and high-achieving advisors, made this happen.

I started working with **James** in the first quarter at Stanford, which, now I recall, was a formative experience that set the tone for my Ph.D. James is one of the most foresighted, inspiring, and generous people I know. He taught me to spot the most impactful directions, identify the key entry points, and emphasize real-world impacts in the most impressive way. James is like a lighthouse when I feel lost and a boat when I feel like sinking in research dilemmas. It is far beyond the research skills that I learned from James, but also discipline to learn deep, not cheap; to think calmly, not talk chaotically; and to embrace the new, not hold on to the old. James is my hero, who is thorough, big-hearted, and open-minded.

Jure has been one of the greatest minds I know. He helped me grow from a “shy” student who didn’t seem able to convince herself of a research idea into a researcher with confidence, a clear mental map of the work, and the ability to deliver direct, powerful messages through crisp presentations. The research structure Jure taught me (“*What’s the problem*”, “*Why is it important?*”, “*Why is it hard?*”...) has been my research bible. The guidance from Jure (“*Less is more*”, “*There is no dumb audience*”, and “*Research should be like making a dessert with one main ingredient, not a soup that mixes everything*”) has been mottos that I keep reminding myself. Jure cares about students so much, in a way that prioritizes their chances of success. Jure taught me what it means to be rigorous, sharp, and outstanding, with himself as the perfect example.

The experience of working with **Jure** and **James** is everything a student can ask for. They have offered me so much support, for which I am deeply grateful. From about 120 meetings in total with each one of them, their advice and wisdom, hopefully, became part of me. And my achievements in the Ph.D and in the future belong to them.

I thank my committee chair, **Nick Haber**, and my committee members **Diyi Yang** and **Sanmi Koyejo**, for their precious time and invaluable feedback.

I am fortunate to have worked with Diyi, who is insightful, creative, and passionate. Her work on human-centric AI has been important source of inspiration and motivation for my own research. Even when I feel confident about the projects, Diyi’s questions and comments often catch me off guard and make me rethink the fundamental questions.

I am deeply grateful to **Jayanthi Subramanian** and **Omer Reingold**, who lead the Stanford CS Ph.D. program, for their invaluable support to the department. I thank **Rok Sosic** and **Lata Nair** for making me feel like home especially when I was new here.

I worked with **Muhammad Jehangir Amjad** and **Chelsea Finn** in CS229 and CS224R as a class assistant. I learned a ton about managing classes and teaching.

Moreover, I have had the opportunity to chat with many other professors: **Huaxiu Yao**, **Linjun Zhang**, **Bruno Ribeiro**, **Zhou Yu**, **Michael Bernstein**, **Carlos Guestrin**, **Chris Manning**, **Omar Khattab**, **Danai Koutra**. I thank Huaxiu for motivating me to pursue research. Linjun has been a close collaborator, and we share bitter and sweet memories of “battling” with conference reviewers.

I had a lovely time interning at Microsoft Research in the summer of 2024, working

with **Michel Galley**. Regardless of personal bias, I would say Michel is the best. Without his thoughtfulness, prudence, and foresight on human-centric area, our work, CollabLLM, would not exist and would not have won the Outstanding Paper Award at ICML 2025. After the internship, I went back to China and had to stay for three months due to delays in my visa renewal. Michel gently asked about my progress every time in our online meeting and offered his help and support. I am deeply grateful for what Michel did for me and for making our success possible. I also want to thank **Jianfeng Gao, Baolin Peng, Hao Cheng, Weixin Cai**, who offered valuable guidance during my internship.

Across many projects, I have learned so much from my collaborators: **Michihiro Yasunaga, Wanjia Zhao, Mert Yuksekogul, Zeliang Zhang, Yao Dou, Jubayer Ibn Hamid, Rex Ying, and Caroline Choi** *etc.* I have had the joy of mentoring many students, often feeling that I was only pretending to know more than they do: **Shiyu Zhao, Parth Sarthi, Evelyn Choi, Arpandeeep Khatua, and Herumb Shandilya** *etc.*

Friends have been the most important part of my Ph.D. life. I met **Yang Zheng, Pu Yi, and Pengli Wang** on the same flight flying from China to the U.S. the first time, and we have been on the same AT&T family plans up to the writing of this acknowledgment. Then the “epic gang” with **Zhiqiang Xie, Jiaju Ma, and Boyang Deng** was formed, and for a while we made Wilbur Dining Hall our daily meetup spot, filled with gossip and laughter. We’ve gotten together for Halloween and New Year parties, “someone’s-life-is-too-good” and “one-of-us-needs-a-drink” hangouts, and recently, Checkers games that somehow became amusingly personal. They explicitly asked me to write more acknowledgment about them, but I ignored, because that would go on for pages. A sincere thank-you goes to **Zhanghan Wang**, who was a sweet companion. He taught me what care and love are, and that I have the right to enjoy life outside of work. He turned me from a cold working machine into whoever I am today.

It was always fun to hang out with labmates in James’ and Jure’s groups: **Kaidi Cao, Qian Huang, Michael Moor, Serina Chang, Hongyu Ren, Kexin Huang, Jiaxuan You, Yanay Rosen, Yangyi Shen, Zach Izzo, Yiqun Chen, Siyu He, Pan Lu, Ruishan Liu, Sheng Liu, Batu El, Rahul Thapa, Nitya Thakkar, Owen Queen, Jiacheng Miao, Kevin Wu, Eric Wu** *etc.* I had great memories of us doing go-karting, Topgolf, pickleball, sushi class, and countless group lunches. Adapting to a competitive environment at

Stanford is hard, and it would certainly have been much harder for me without **Kaidi Cao**, who offered me invaluable guidance. **Michael Moor** and I worked on Med-Flamingo, one of the earliest biomedical foundation models. He taught me how to play SET and made me realize that my pattern recognition ability is definitely below a regular AI. I also miss carving pumpkins with **Qian Huang** and **Eric Zelikman** in the old days, when dropping out of Ph.D. programs was less common than it is now. **Chris Fifty** and I bonded through many fun experiences, yet I still cannot fully forgive him for leaving Stanford so early.

I am more than lucky to meet so many friends thanks to Stanford's social culture. It was amazing to meet **Violet Xiang**, one of the funniest and sweetest, and bring "Doordash order frequency", "late hour at the Gates office" and "the chaos of unprepared surfing trips" to a next level. **Joy He-Yueya**, **Tian Gao**, and I were once a "homeless" group for staying too late at the Gates building. Joy has been my life advisor, second to my mom, and with a better record of me actually listening. Tian reminds me of kindergarten, where wildly different kids managed to get along. **Weijia Shi** and I met in a conference event in the most hilarious way. We have been sharing life updates since then and laugh far too much every time we talk with each other. It is fun to practice Hokkien outside of my family video calls with **Michelle Guo**. **Daniel Shin** and I met at the TGIF CS social and have never lost touch since. **Rajan Vivek** and I bonded over tasty Asian food. I enjoyed practicing "stand-up" with **Aishwarya Mandyam**, and I appreciate her efforts in trying to convince me to join the running club. And I thank **Rylan Schaeffer** for organizing the best social events.

Many great memories flow. I thank my gym buddies **Vignesh Kothapalli**, **Rishabh Rajan**, tennis buddies **Moritz Schaefer**, **Tianlang Chen**, diving buddies **Judy Shen**, **Eric Mitchell**, karaoke buddies **Steven Feng**, **Ge Gao**, **Qitong Gao**, hiking buddies **Pu Yi**, **Tiange Xiang** (although I haven't particularly liked hiking). I thank friends who helped me and with whom I had great fun: **Horace He**, **John Yao**, **Ajay Sridhar**, **Yanjiang Guo**, **Anikait Singh**, **Perry Dong**, **Jihao Andreas Lin**, **Swapnil Gandhi**, **Connor Shorten**, **Zoe Piran**, **Inna Wanyin Lin**, **Omar Shaikh**, **Ed Chen**, **Tyler Lum**, **Giannis Daras**, **Joy Hsu**, **Vivek Gopalakrishnan**, **Simran Khanuja**, **Cindy Wu**.

Finally, I thank friends I connected with through music: **David**, **JB**, **Lucas**, **Renee**, **Tony**, **Richard**, and **Azmi**. We had many amazing nights playing Rock and Blues. Fun fact: We do not know each other's last names because we never mention them.

To all who have carried my learning and lit the way for my growth

Contents

Abstract	iv
Acknowledgments	vi
1 Introduction	1
1.1 Background: What are Agents and Why Agents?	1
1.2 Motivation: Why Collaborative Agents?	2
1.3 Thesis Outline and Contributions	6
I Training Collaborative Agents	12
2 From Passive Responders to Active Collaborators	13
2.1 Introduction	13
2.2 Related Work	16
2.3 Formulation: What is the multiturn objective	18
2.4 CollabLLM: Collaborative LLM Training	18
2.4.1 Our Core Innovation: Multiturn-aware rewards	19
2.4.2 1) Conversational Reward Function: Measure multiturn success . .	20
2.4.3 2) Forward Sampling: Estimate long-term outcome	21
2.4.4 Online Training: How to optimize with multiturn-aware rewards . .	21
2.4.5 Offline Training: How to generate offline datasets	22
2.4.6 Extended Discussion I: Connections with causal inference	23

2.4.7	Extended Discussion II: Distinctions from related multiturn training frameworks	23
2.5	Experimental Setup	24
2.6	Simulated Experiments	26
2.6.1	Main Results: How effective is CollabLLM under simulation	26
2.6.2	Ablations: What mechanism makes CollabLLM effective	27
2.6.3	Case Study: What does CollabLLM learn	28
2.6.4	Generalization: How does CollabLLM generalize to unseen data	29
2.7	Real-world User Study	32
2.7.1	Interface: What is our collaboration environment	32
2.7.2	Results: How effective is CollabLLM in the real world	37
2.7.3	Extended Discussion III: What are the differences and similarities between simulated users and real users	38
2.8	Safety Evaluation: Is CollabLLM safe to deploy	39
2.9	Extended Direction: Improving User Simulators	41
2.9.1	Motivation: Why better user simulators matter	42
2.9.2	Challenges: Why user simulation is difficult	43
2.9.3	Realistic Simulation for Collaborative Training and Evaluation: What are the promising approaches?	44
2.10	Conclusion	45

II Improving Agents to Learn from Experience 46

3 Benchmarking Agents in Solving User Queries 47

3.1	Introduction	48
3.2	Related Benchmarks	53
3.3	Formulation: Answering queries over heterogeneous knowledge	54
3.4	STARK: A Benchmark for User-centered Reasoning over Semi-structured Knowledge	55
3.4.1	Semi-structured Knowledge Bases (SKBs)	55
3.4.2	User-centered Queries on Semi-structured Knowledge Bases	57

3.4.3	Constructing Synthesized Queries	58
3.4.4	Analysis: Distribution and human evaluation on synthesized data	61
3.4.5	Constructing Human-Generated Queries	63
3.5	Experimental Setup	64
3.6	Results: How do models perform on STARK	67
3.6.1	Failure Analysis I: Dense and sparse retrieval models	69
3.6.2	Failure Analysis II: Multivector retrievers	70
3.6.3	Failure Analysis III: Single-agent reranker	71
3.6.4	Failure Analysis IV: Single-agent retriever with tools	73
3.6.5	Retrieval Latency	75
3.7	Conclusion	76
4	Learning from Experience through Contrastive Reasoning	77
4.1	Introduction	77
4.2	Related Work	80
4.3	Formulation: Agent systems for complex queries	82
4.4	AVATAR: Actor-Comparator System for User Queries	84
4.4.1	Actor: The single agent and its limitations	85
4.4.2	Comparator: Enable learning from contrastive reasoning	86
4.4.3	AVATAR System: Optimization and deployment	88
4.5	Experiments: Evaluating agent systems	89
4.5.1	STaRK: Semi-structured knowledge base retrieval	92
4.5.2	Flickr30K: Image retrieval with multi-modal queries	95
4.5.3	QA Benchmarks: Multi-hop reasoning tasks	96
4.6	Conclusion	99
III	Scaling Up Collaborative Multi-Agent Systems	101
5	Optimizing Large-scale Multi-Agent Collaborators	102
5.1	Introduction	102
5.2	Related work	106

5.3	Formulation: What is the objective for multi-agent systems	107
5.4	OPTIMAS: Unified Multi-Agent System Optimization	108
5.4.1	Local Reward Functions: Align local with global objective	109
5.4.2	Adaptation: How to maintain alignment during optimization	111
5.4.3	Optimization: How to optimize with aligned reward functions	111
5.4.4	Theory: Why does local optimization improve global rewards	113
5.5	Experimental Setup	117
5.5.1	Benchmarks and Evaluation	117
5.5.2	Multi-Agent System Design	118
5.5.3	Baselines	120
5.5.4	Optimization Setup	121
5.6	Results	122
5.6.1	Main Results: How well does OPTIMAS compare to baselines	122
5.6.2	Optimization Dynamics: How do configurations evolve	123
5.6.3	Alignment Quality: How well do LRFs align with global rewards	124
5.6.4	Sensitivity: Does better alignment lead to better optimization	124
5.6.5	Interpretability: What do the learned reward functions capture	126
5.6.6	Efficiency: How data- and compute-efficient is OPTIMAS	127
5.7	Conclusion	131
6	Conclusion	133
6.1	Summary of Contributions	133
6.1.1	Part I: Training Collaborative Agents	134
6.1.2	Part II: Enabling Agents to Learn from Experience	135
6.1.3	Part III: Scaling Up Collaborative Multi-Agent Systems	135
6.2	Vision for the Future	136
6.3	Concluding Remarks	137

Chapter 1

Introduction

1.1 Background: What are Agents and Why Agents?

The concept of an *agent* existed long before modern large AI models. Traditionally, an agent was a lightweight computer program that acted on behalf of a user in a specific environment. It was typically built to automate a narrowly defined task (*e.g.*, playing chess, filtering and sending emails, or scheduling a meeting given constraints) or to carry out simple conversations in a limited domain (*e.g.*, a booking chatbot). These agents were mostly procedural: the goal was explicitly specified, the behavior was engineered through rules or pipelines, and success could be evaluated with task-specific metrics.

In the modern context, the meaning of *agent* has broadened because the environments where agents operate have changed. Many deployments now happen in settings like the web, workplace software, or personal devices, where information is spread across many sources and important constraints may appear only after several interaction steps. These settings are also actionable: beyond producing text, an agent may need to look up information, run code, call external services, or update documents.

Foundation models enable this shift by providing a natural language interface and the ability to use long context across multiple turns. As a result, modern agents are better viewed as proactive decision making systems that can interpret goals that are not fully specified, track evolving context, and use external tools and APIs [41, 124, 139, 145, 150, 191].

These agents demonstrate remarkable capabilities in reasoning [9, 173, 190, 191], planning [48, 64, 213], and can interact with environments [41, 93, 105, 177] to perform complex tasks like multi-step problem-solving and information retrieval [148, 189, 219]. **The central challenge therefore moves from executing a fixed plan to deciding, together with the user, what to do next and how to adapt as the task unfolds.**

1.2 Motivation: Why Collaborative Agents?



Figure 1.1: **Motivating examples of collaboration failures in multiturn interactions.** **Left [4]:** A recipe request shows unnecessary assumptions: the agent suggests chicken without asking about dietary preferences, forcing the user to correct it (vegetarian) and wasting time. **Right [182]:** A health query shows generic answers: the agent lists broad causes without first understanding key context necessary to address the specific issue.

This broader view of agents has enabled applications where language is the main interface and the agent takes multiple steps over time, such as planning assistants that produce concrete outputs like an itinerary or a grocery shortlist. Yet, when we are collaborating with strong agents like large language models, we noticed a recurring and surprising gap between model capability and effective collaboration. **These models can be remarkably strong at difficult technical problems, but they can still fail at basic human interaction.**

A popular example comes from Casey Newton, describing an attempt to order groceries with an agentic system. The user asked for help buying groceries on Instacart and expected

the system to ask a few basic questions, such as where they live, which store they prefer, and what they want to buy. Instead, the system immediately opened Instacart and started searching for milk in stores located in an incorrect city. This failure is striking because it is not a failure of knowledge, but a failure of communication and coordination. An agent that does not ask clarifying questions, that guesses preferences too early, or that cannot track context reliably can feel frustrating even when the underlying model is highly capable.

This is not a minor usability issue. Poor collaboration impacts performance, safety, and efficiency. Even in low-stakes tasks, unchecked assumptions can slow down progress and waste user time. In high-stakes domains such as healthcare, legal, or financial settings, similar failures can be more serious and result in misleading outcomes. Figure 1.1 illustrates two failure modes. In a recipe request, the agent makes an unnecessary assumption by suggesting chicken, forcing the user to correct it by stating their vegetarian preference. In a health question, the agent responds with a generic list instead of asking for a few key details needed to answer. In both cases, users waste substantial time correcting misunderstandings and restating constraints.

One reason this gap persists is that many evaluations focus on simplified settings, such as single-turn prompts with clear and complete instructions. Real interactions are different. Users often begin with a vague request, refine preferences after seeing options, and introduce key constraints only later. As a result, success is rarely a single response. It depends on how well the agent interacts over time to clarify intent, track evolving context, and adapt its actions as the task unfolds.

Collaboration therefore demands capabilities beyond correctness, including communication to understand intent, coordination to choose what to do next, and adaptation to preferences and changing objectives. Moreover, if humans and AI systems are to solve meaningful problems together, agents cannot only respond passively to requests. They must also help users explore options, reduce risks in decision making, and amplify creativity.

At the same time, unlike real collaborators, agents may not improve meaningfully from more experience. When humans collaborate repeatedly on similar tasks, they naturally learn from past mistakes, refine their strategies, and become more effective partners. An experienced human collaborator who previously misunderstood a flight search request

would recognize similar patterns in future interactions and adjust their approach accordingly. This improvement with experience is a hallmark of effective collaboration.

Agents, however, often fail to exhibit this adaptive behavior. Consider the task of helping users book flights, where agents must interpret travel preferences, distinguish between one-way and round-trip requests, and search the appropriate booking platforms. Figure 1.2 shows a pattern that appears across multiple deployment studies [101]: even after handling similar requests, agents repeat the same errors. This failure is not due to task complexity or limited capability. The agent possesses sufficient knowledge to distinguish flight types and has access to the correct booking tools. Instead, the failure stems from an inability to learn from past interactions and recognize recurring patterns in its own mistakes.

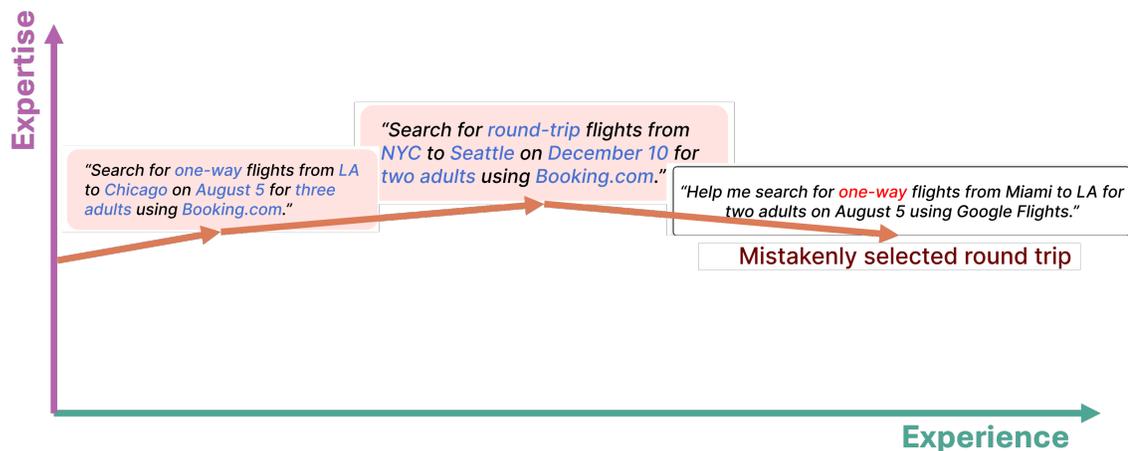


Figure 1.2: **Agents fail to learn from experience in repeated interactions.** Unlike human collaborators whose expertise improves with experience, agents often repeat the same mistakes across similar tasks. In flight booking scenarios [101], an agent consistently misinterprets one-way flight requests as round-trip searches despite previous corrections, demonstrating a systematic failure to learn from past errors and adapt its behavior accordingly.

This gap between experience and improvement creates a fundamental limitation for collaborative agents. Users expect that working with an agent over time will lead to smoother interactions, fewer clarifications, and better outcomes. When agents fail to meet this expectation, the collaboration becomes frustrating and inefficient. Moreover, the impact of this limitation grows more severe in complex domains. In retrieval-heavy tasks such as product

search or biomedical question answering, where users add constraints progressively and may reference past searches, an agent that cannot learn from its interaction history will fail to recognize which strategies succeeded or failed in similar situations.

The root of this problem lies in how current agents process information. They excel at reasoning within a single interaction but lack ways to identify systematic patterns across multiple interactions, recognize when past strategies failed, and adapt their approach based on accumulated experience. Agents may access conversation history as raw text, but they cannot easily extract the higher-level lessons: which assumptions led to mistakes, which clarification questions proved most valuable, or which reasoning patterns consistently produced better outcomes.

Addressing this limitation requires enabling agents to engage in *contrastive reasoning*: comparing successful and unsuccessful interactions to identify the key differences in their approaches and recognize the systematic gaps in their current strategies. The challenge is not simply to give agents more memory or longer context windows. Instead, agents must learn to step back and analyze patterns across multiple tasks.

Yet even with mechanisms to learn from experience, agents face another fundamental challenge: complex collaborative tasks may exceed what individual agents can reliably handle. Real-world collaboration often involves tasks where information is scattered across heterogeneous sources, constraints accumulate progressively, and multiple forms of reasoning must coordinate simultaneously. A single agent attempting such tasks may struggle not because it lacks knowledge or collaborative skills, but because the intellectual demands exceed what any individual decision-maker can manage effectively.

In fact, this observation about individual limits connects to a well-studied phenomenon in social science. Individual decision-makers face cognitive constraints that limit their ability to process information and evaluate alternatives, a pattern Herbert Simon termed *bounded rationality* [151]. Decades of research show that groups can transcend these limits: diverse teams outperform homogeneous groups of high-ability individuals [58], crowds often produce better judgments than experts [155], and groups exhibit a measurable collective intelligence that predicts performance better than any single member's ability [176]. This collective capability emerges through coordination mechanisms that manage dependencies between activities [108], transactive memory systems where members specialize

and track who knows what [172], and distributed cognition where reasoning spreads across people and tools [66].

These findings provide a principled basis for building multi-agent AI systems: distribute not just tasks, but also the cognitive functions required for both learning from experience and handling complexity. Different agents can take complementary roles to overcome individual limitations. One agent can attempt tasks while another analyzes failures; one can focus on broad search while another validates constraints. This division of cognitive labor addresses both the learning problem and the complexity problem identified above.

Building such systems introduces new challenges. Agents must coordinate not only with users to understand intent, but also with each other to share experiences, align on lessons learned, and avoid conflicting goals. As systems grow to include more agents, ensuring that local improvements in one component translate to better overall outcomes becomes increasingly difficult.

Motivated by the above limitations, this thesis addresses three dimensions of collaboration: how agents learn to interact effectively with humans through experience, how they improve with more experience, and how multiple agents coordinate with each other to serve human goals at scale.

1.3 Thesis Outline and Contributions

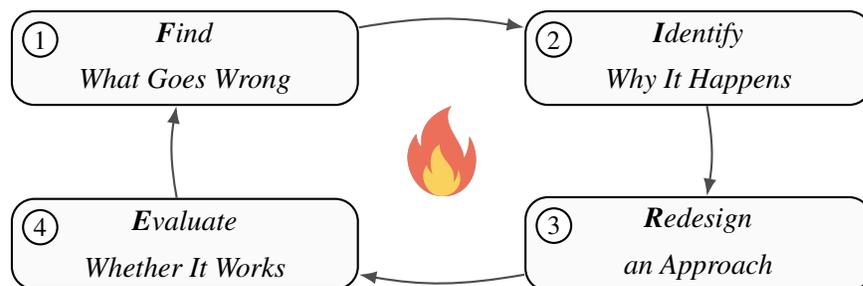


Figure 1.3: Thesis workflow (FIRE).

The thesis addresses a number of important questions regarding building collaborative agents by revealing how agent behavior and system structure lead to collaboration effectiveness. Throughout this thesis, we follow four steps: we find what goes wrong, identify why it happens, redesign an approach, and evaluate whether it works.

Part I (Chapter 2): How to enable collaborative behaviors for agents?

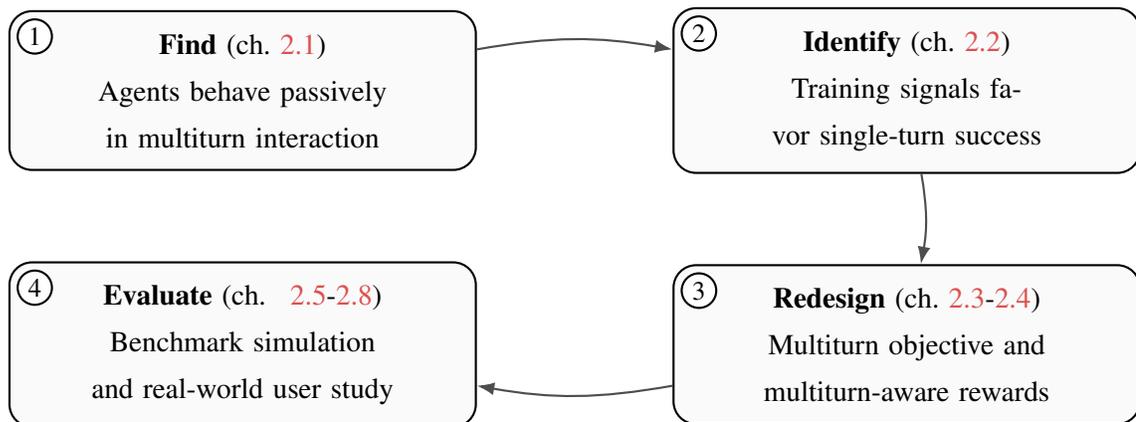


Figure 1.4: **Part I: Training collaborative agents.**

Why do capable models make poor collaborators? We trace the problem to training: current fine-tuning methods like RLHF reward models for immediate, single-turn responses, giving them no incentive to ask clarifying questions or guide users toward better outcomes. The result is agents that passively agree rather than actively collaborate.

We introduce COLLABLLM [179], a training framework that teaches agents to look ahead. By simulating future conversation trajectories and estimating how current responses affect long-term outcomes, we define a *multiturn-aware reward* that captures both task success and interaction efficiency. Fine-tuning on this reward produces agents that ask insightful questions, offer targeted suggestions, and help users refine vague requests into actionable goals. In user studies, COLLABLLM increased user satisfaction and reduced task completion time compared to conventionally trained models.

Part II (Chapters 3–4): How do we enable agents to learn from experience?

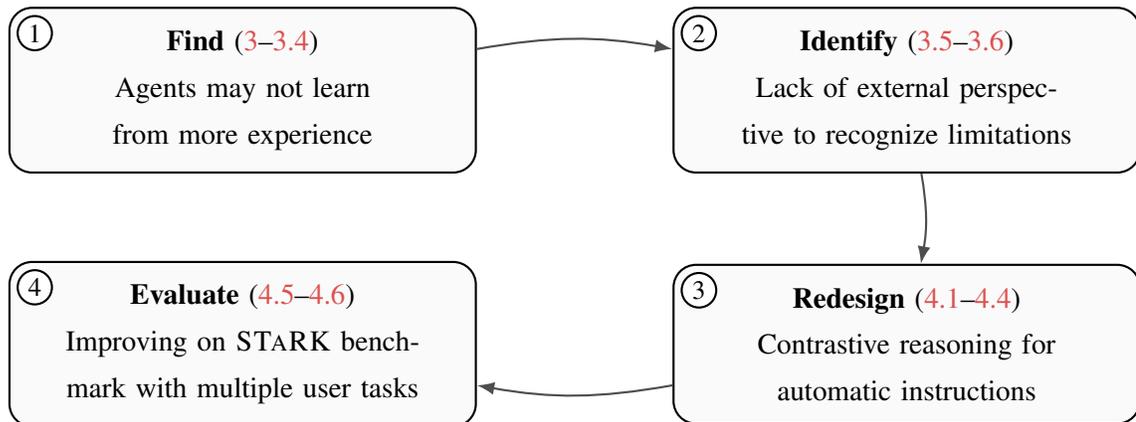


Figure 1.5: **Part II: Improving Agents to Learn from Experience**

How can we build agent systems that recognize their limitations and effectively address them? The key challenge is twofold: agents must first identify what went wrong in failed attempts, then generate actionable guidance to improve future performance. Chapter 4 introduces AVATAR [181], a framework based on contrastive reasoning that addresses both challenges. The framework introduces a Comparator agent that compares successful and unsuccessful task attempts to identify systematic gaps in the agent’s strategies. Critically, this comparison reveals not just what failed, but why it failed, *i.e.*, which assumptions were incorrect, which steps were missing, and which reasoning patterns led to errors. Based on these insights, the system automatically generates instructions that guide the agent to address its specific weaknesses, enabling it to gradually learn how to solve harder tasks.

To systematically evaluate whether agents can continuously learn from experience, we need benchmarks that test their ability to improve on realistic collaborative tasks over multiple attempts. Chapter 3 addresses this need through STARK [182], a large-scale benchmark for retrieval over semi-structured knowledge bases spanning e-commerce, academic research, and healthcare. The benchmark presents complex queries where users specify multiple constraints: a parent searching for a tricycle mentions brand preferences, safety requirements, and delivery constraints; a researcher queries for drugs interacting with genes active in specific anatomical regions. These scenarios create opportunities to

observe whether agents can recognize their mistakes and improve their strategies when exposed to similar queries.

Our experiments reveal that current agents, including state-of-the-art language models, fail to improve meaningfully when given multiple attempts at similar tasks. They repeat the same mistakes across attempts because they cannot identify the systematic patterns in their failures or generate effective strategies to address them. The contrastive reasoning framework in AVATAR addresses this by explicitly comparing what worked versus what failed, extracting the critical differences, and translating these insights into actionable instructions that enable agents to recognize their limitations and systematically improve their performance on collaborative tasks.

Part III (Chapter 5): How to scale multi-agent systems for stronger capabilities while maintaining alignment?

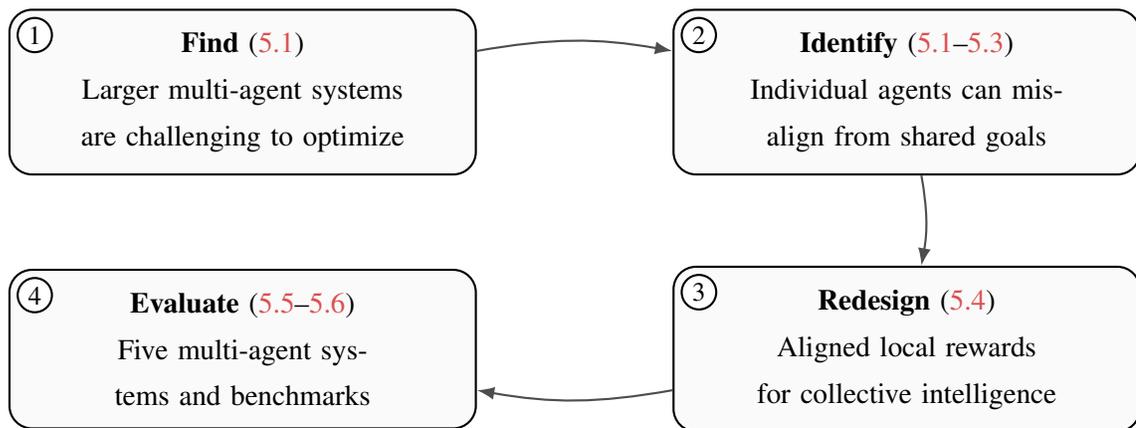


Figure 1.6: **Part III: Scaling Multi-Agent Systems with Alignment**

Another way to understand AVATAR is that it uses two agents, an Actor and a Comparator, working together to enhance the system’s capabilities beyond what a single agent can achieve. This raises a natural question: if two agents working together outperform one, why not more? The promise of multi-agent systems is that specialized agents can divide complex tasks and collaborate toward shared goals. But scaling introduces another important problem: each agent optimizes for its own objective, and these objectives may not add

up to what the user actually wants. An agent for query understanding that produces verbose interpretations might help its own reasoning, but overwhelm the downstream retriever. Each agent wins locally while the system fails globally.

The root cause is misalignment between what each agent is rewarded for and what the system as a whole should achieve. We address this with OPTIMAS [180], a framework built on a simple but powerful idea: teach each agent to care about the right thing. We learn a *globally aligned reward function* for each agent, one that assigns higher scores to outputs that lead to better system-level outcomes. When agents optimize these aligned rewards, local improvements reliably translate to global gains.

This reframing changes how we think about multi-agent optimization. Rather than searching over configurations to find combinations that happen to work well together, we give each agent a clear signal of what “good” means from the system’s perspective. Across five compound systems spanning product recommendation, medical analysis, and question answering, OPTIMAS consistently improves performance where baselines show erratic behavior, sometimes helping and sometimes hurting.

Taken together

A thread runs through these contributions: collaboration cannot be an afterthought. It is difficult to build capable agents and then expect them to work well with humans or with each other. Training signals shape behavior. System design determines what agents can observe and use. The optimization objective defines what counts as improvement. Each of these choices can either enable collaboration or work against it.

Part I shows that agents trained on single-turn rewards become passive, even when the underlying model is highly capable. Part II reveals that agents struggle to learn from experience. Part III demonstrates that scaling to many agents creates alignment problems that brute-force search cannot solve. In each case, the solution requires rethinking the foundations: what we optimize for, how to improve agents, and how they are structured.

The broader ambition is to build AI systems that feel less like insufficient tools and more like capable partners. Partners who ask when they are uncertain, who notice when something is missing, and who work together without pulling in opposite directions. The methods in this thesis offer a principled path toward that goal: design the incentives for

ideal behaviors, the architectures, and the objectives so that collaboration emerges naturally from agents doing what they were trained to do. This thesis advocates a future where AI systems are built to amplify human capability rather than replace human judgment, enabling people to solve problems that neither humans nor machines could tackle alone.

Part I

Training Collaborative Agents

Chapter 2

From Passive Responders to Active Collaborators

2.1 Introduction

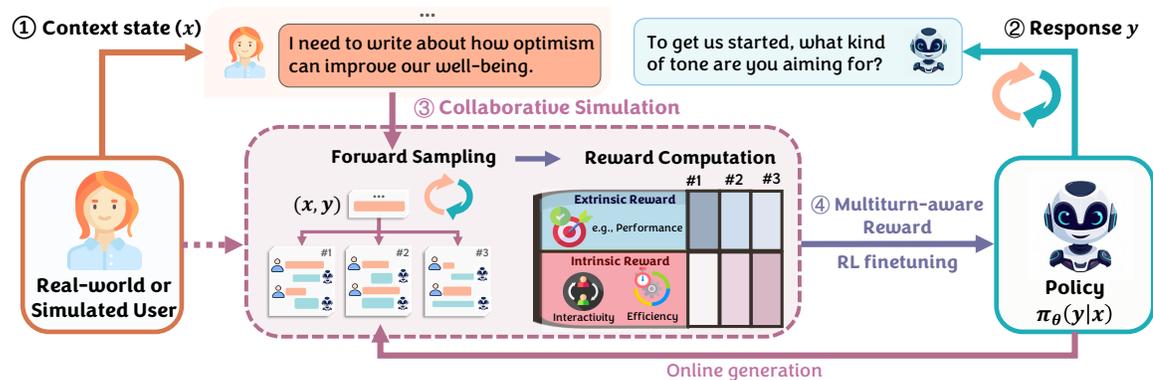


Figure 2.1: COLLABLLM Framework: Given a context ①, the model generates a response ② to maximize long-term collaboration gains, termed *Multiturn-aware Rewards* (MR). During training, MRs are estimated via ③ collaborative simulation, which forward-samples conversations with simulated users. Finally, ④ reinforcement fine-tuning is applied using the MRs.

Modern Large Language Models (LLMs) excel at generating high-quality single-turn responses when given well-specified inputs. However, real-world users often do not fully

articulate their intents and sometimes initiate conversations with an imprecise understanding of their own needs [158]. As a result, users routinely refine their requests post hoc through iterative corrections, which can increase frustration, hinder effective task completion, and reduce conversational efficiency [3, 83, 165, 203]. Therefore, an open problem is to train models that actively guide users in clarifying and refining their intents, and help them achieve their goals. This key challenge would improve user satisfaction and efficiency and streamline human-LLM interactions—especially as LLMs are being applied to real-world tasks that are increasingly complex and open-ended.

A notable limitation of established fine-tuning techniques, such as Reinforcement Learning from Human Feedback (RLHF) [117], is that they primarily reward LLMs for immediate, single-turn responses, reducing their incentive to seek clarification or assist users in refining their intents or preferences. As a result, commonly used LLMs tend to prioritize direct answers, even though seeking additional context would enhance task completion and increase user satisfaction [83].

Here we introduce **COLLABLLM**, a novel and general training framework that improves the ability of LLMs to effectively collaborate with humans in multiturn scenarios [6, 45, 134]. The key innovation of COLLABLLM is to promote LLMs’ forward-looking behavior that leads to long-term collaboration gains (Figure 2.1). We introduce a collaborative simulation module that samples future conversations with users to estimate the long-term impact of model responses across multiple turns, a measure we term the *Multiturn-aware Reward (MR)*. The MR function evaluates responses by incorporating both extrinsic metrics, such as task-specific success, and intrinsic metrics, such as efficiency, to holistically assess collaboration quality (*cf.* Section 2.4). By fine-tuning with RL algorithms [133, 140] on MRs, COLLABLLM promotes responses that lead to better task completion and efficiency in later conversation stages. As shown in Figure 2.2b, the fine-tuned model goes beyond simply responding to user requests in Figure 2.2a—it actively collaborates by asking follow-up questions about the writing tone, generating targeted content about the role of optimism, and offering insightful suggestions such as adding anecdotes.

We also introduce **three challenging multiturn tasks** for training and evaluation in simulated environments: `MediumDocEdit-Chat`, `BigCodeBench-Chat`, and `MATH-Chat`,

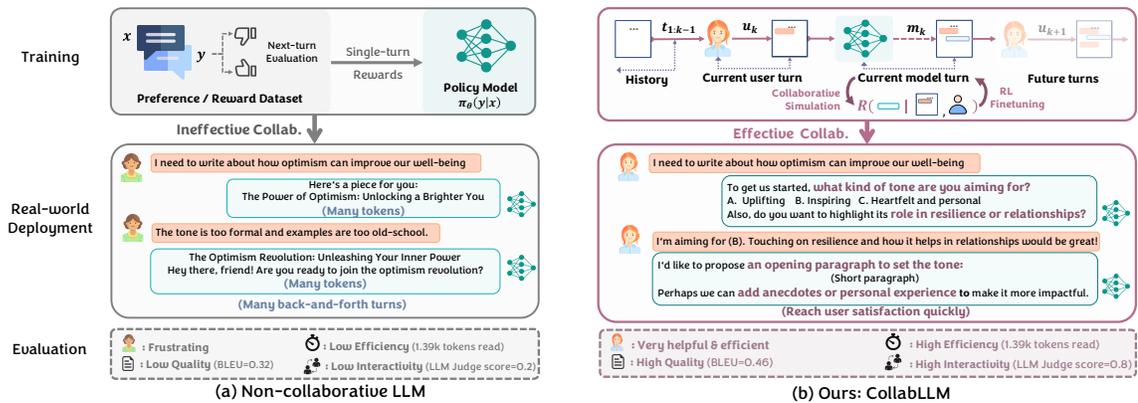


Figure 2.2: Real examples from COLLABLLM and non-collaborative LLM fine-tuning. (a) Non-collaborative LLM fine-tuning relies on single-turn rewards on immediate responses, which exhibits passive behaviors that follow the user’s requests, leading to user frustration, less efficient process, and less satisfactory results. (b) COLLABLLM incorporates Multiturn-aware Rewards from collaborative simulation, enabling forward-looking strategies. This results in more high-performing, efficient, and interactive conversations that anticipate future needs, propose timely clarification, and provide insightful suggestions.

which respectively encompass document creation, code generation, and multiturn question answering. On the three test sets, our approach improves task accuracy metrics by 18.5% and interactivity by 46.3% on average compared to our best baselines, according to LLM judges. Beyond the tasks that the COLLABLLMs are fine-tuned on, we show COLLABLLMs are highly generalizable to other data domains.

Moreover, we perform a **large-scale and real-world user study** with 201 Amazon Mechanical Turkers (MTurkers), who are asked to write documents with the help of anonymous AI assistants, either COLLABLLM or non-collaboratively trained LLMs. COLLABLLM achieves impressive improvement with 17.6% increase in user satisfaction and yield user time savings of 10.4% on average. The qualitative analysis from MTurkers confirms our observations: non-collaboratively-trained LLMs passively agree with users, while COLLABLLM actively provides insightful questions and suggestions to guide writing processes.

2.2 Related Work

Non-collaborative LLM training. Existing LLM training frameworks, including pre-training, supervised fine-tuning (SFT), and reinforcement learning (RL) [89, 117, 133, 140], primarily optimize for next-turn response quality. Standard RL methods such as Proximal Policy Optimization (PPO) [140] apply rewards to individual model responses without accounting for their long-term impact on conversation trajectories. While effective for single-turn objectives, these approaches fail to capture how responses influence user intent discovery and long-term task success [3, 83, 165, 203].

Prompting techniques for multiturn interaction. Prior work has explored prompting strategies to enhance LLM interactivity, particularly for clarification questions [31, 38, 80, 82, 110, 207, 211] and mixed-initiative dialogues [24, 36, 96]. For instance, Mu et al. [110] prompts LLMs to ask clarification questions when code generation requests are ambiguous. However, such prompting-based approaches are constrained by predefined interaction patterns, limiting adaptability across different tasks and conversation stages. Moreover, their reliance on fixed prompts reduces generalization, as demonstrated in our experiments where proactive prompting fails to match the effectiveness of our fine-tuned models.

Learning-based methods for multiturn interaction.

- **LLMs for generating clarification questions:** Beyond prompting, prior studies have explored supervised fine-tuning [4], RL fine-tuning [23, 42, 202], and active learning [119] to train models to ask clarification questions. For example, Chen et al. [23] use Direct Preference Optimization (DPO) to encourage models to request clarifications. However, like prompting approaches, these methods primarily focus on clarification questions and do not generalize to broader multiturn collaboration strategies.
- **Multiturn training for LLMs:** Recent benchmarks [1, 84] evaluate LLMs’ performance in multiturn settings, measuring the goal orientation and planning capabilities of models across interactions. Several studies extend RLHF to multiturn settings by optimizing trajectory-level rewards [47, 142, 149, 205, 217]. Other works [37, 184] leverage self-chat or self-play to enhance model adaptation. However, these methods primarily rely

Table 2.1: Compare COLLABLLM with Selected Works. (1) Task-Agnostic, assessing whether the approach applies across diverse domains rather than being task-specific; (2) Versatile Interaction, evaluating its ability to support diverse strategies for intent discovery and efficient task completion beyond predefined behaviors; (3) User-Centric, determining whether engagement, efficiency, and intent discovery are explicitly considered; and (4) Causal & Objective-Aligned Reward, measuring whether reward estimation captures causal effects on future interactions and optimizes for long-term task success.

	Task-Agnostic	Versatile Interaction	User-Centric	Causal & Objective-Aligned Reward
ClarifyGPT [110]	✗	✗	✗	-
STaR-GATE [4]	✓	✗	✗	-
MTPO [142]	✓	✓	✗	✗
COLLABLLM	✓	✓	✓	✓

on post-hoc trajectory-level data, learning from observed conversations rather than explicitly modeling the causal effect of individual responses on task success. Additionally, they often overlook open-ended tasks such as document generation [44, 71], where user responses can be highly diverse, and users may have limited capacity to read and refine lengthy model outputs.

- **User simulators for enhancing AI systems.** Recent works employ user simulators to enhance dialogue systems [147, 162] and LLMs [44, 57, 62]. Recently, Hong et. al. [57] leverage LLMs to create diverse synthetic dialogues with varying user personas to train smaller dialogue models. CollabLLM differs in leveraging user simulators in forward sampling to account for long-term effect in both offline and online training.

In Table 2.1, we compare COLLABLLM with related methods across four key dimensions. COLLABLLM is a general, user-centric, and multiturn-aware framework that leverages more accurate reward estimation to better align with real-world objectives, enhancing user satisfaction and streamlining human-LLM interactions.

2.3 Formulation: What is the multiturn objective

In contrast to many existing tasks that are single-turn and require no human involvement beyond the initial query, our problem formulation reflects a real-world setting in which a user’s underlying (implicit) goal is defined as g in a multiturn conversational task. The conversation unfolds over multiple turns $t_j = \{u_j, m_j\}$, where u_j is the user input and m_j is the model’s response at each turn $j = 1, \dots, K$, where K is the number of turns in the conversation.

At the j -th turn, the model generates its response based on the previous conversation turns $t_{1:j-1} = \{t_1, \dots, t_{j-1}\}$ and the current user response u_j . For simplicity, we define historical conversation at j -th turn as $t_j^h = t_{1:j-1} \cup \{u_j\}$, therefore, $m_j = M(t_j^h)$. The objective is to generate a sequence of model responses $\{m_j\}_{j=1}^K$ that effectively and efficiently achieve for goal g , *e.g.*, answering a math question, where goal achievement is assessed based on user satisfaction or an external evaluation function, such as accuracy by LLM judge. Formally, we define the objective as $R^*(t_{1:K} \mid g)$, where R^* incorporate the achievement of task success and user experience factors such as time cost.

2.4 CollabLLM: Collaborative LLM Training

Key Motivations. Established LLM training frameworks, such as Reinforcement Learning from Human Feedback (RLHF) [117], focus on maximizing immediate rewards for single-turn tasks. This causes a misalignment between their single-turn objective and real-world multiturn objective $R^*(t_{1:K} \mid g)$. Precisely, the model’s accumulative single-turn reward $\sum_{j=1}^{j=K} R(m_j \mid t_j^h)$ may not imply a higher final reward $R^*(t_{1:K} \mid g)$. *In fact, achieving high single-turn rewards at each turn may not imply a higher final reward.* For example, consider a task where the user’s goal g is to write an engaging article. A model trained with traditional RLHF might generate isolated responses, like drafting an introduction or listing conclusions. While these responses are helpful in isolation, they fail to consider how the sections flow together, resulting in an article that might not be cohesive and aligned with the user’s goal.

Instead, effective multiturn collaboration requires model responses that optimally contribute to the final reward. The model should aim to align its responses with the user’s goal g by considering their impact on the entire conversation trajectory $t_{1:K}$. In the previous example, instead of generating a conclusion, asking, “Should I maintain an engaging tone in the conclusion like the introduction?” offers better long-term alignment with the goal.

2.4.1 Our Core Innovation: Multiturn-aware rewards

In Figure 2.1, our key insight is that effective multiturn collaboration relies on **forward-looking strategies**. Given a context ①, the model should consider how its response ② influences the subsequent turns of the conversation. To capture this, we design a ③ collaborative simulation module to estimate this impact. By ④ fine-tuning to distinguish between potential future conversations resulting from different responses, the model generates responses that align better with the overarching goal g .

This high-level design naturally aligns with causal effect estimation [125, 126], which evaluates the interventional effects of an action in sequential decision-making. Section 2.4.6 provides further discussion on the connection between causal effect estimation and our approach. More specifically, we define the Multiturn-aware Reward:

Multiturn-aware Reward (MR): *The multiturn-aware reward for model response m_j at the j -th turn is given by:*

$$\begin{aligned} & \text{MR}(m_j \mid t_j^h, g) \\ &= \mathbb{E}_{t_j^f \sim P(t_{j+1:K} \mid t_j^h \cup \{m_j\})} R^*(t_j^h \cup \{m_j\} \cup t_j^f \mid g) \\ &= \mathbb{E}_{t_j^f \sim P(t_j^f \mid t_{1:j})} R^*(t_{1:j} \cup t_j^f \mid g), \end{aligned} \tag{2.1}$$

where $t_{1:j}$ denotes the conversation history up to and including the j -th turn, and $t_j^f = t_{j+1:K}$ represents the forward trajectory of turns following the j -th turn. The distribution $P(t_j^f \mid t_{1:j})$ models the possible forward conversations conditioned on the prior conversation history.

However, computing Equation 2.1 remains challenging as it requires the following components: **(a) A conversation-level reward function**, $R^*(t \mid g)$, for evaluating an arbitrary

multiturn conversation t , and **(b) a sampling strategy for obtaining forward conversations** $P(t_j^f | t_{1:j})$, which represents the forward conversation distribution. We elaborate on the two components in Section 2.4.2 and 2.4.3.

2.4.2 1) Conversational Reward Function: Measure multiturn success

We approximate the conversation-level reward $R^*(t | g)$ with a combination of extrinsic (goal-specific) and intrinsic (goal-agnostic) metrics:

$$R^*(t | g) \simeq R_{\text{ext}}(t, g) + R_{\text{int}}(t), \quad (2.2)$$

where $R_{\text{ext}}(t, g)$ focuses on task success, and $R_{\text{int}}(t)$ evaluates user experience including efficiency and engagement.

- **Extrinsic Reward** $R_{\text{ext}}(t, g)$ measures how well the conversation achieves the user’s goal g . Formally:

$$R_{\text{ext}}(t, g) = S(\text{Extract}(t), y_g), \quad (2.3)$$

where $\text{Extract}(t)$ extracts the final solution or response from the conversation t , especially for tasks requiring revisions or multi-step answers. y_g is the reference solution for the goal g , e.g., the ground truth solution for a math problem. And $S(\cdot, \cdot)$ evaluates task-specific metrics like accuracy or similarity. This ensures the conversation contributes directly to achieving the desired goal.

- **Intrinsic Reward** $R_{\text{int}}(t)$ prioritizes conversations that enhance user experience, defined as:

$$R_{\text{int}}(t) = -\min[\lambda \cdot \text{TokenCount}(t), 1] + R_{\text{LLM}}(t), \quad (2.4)$$

where we encourage conversational efficiency by penalizing excessive tokens that users read and write, with λ controlling the penalty severity. This efficiency measure is bounded by 1 to maintain balance with other metrics. The second term, $R_{\text{LLM}}(t)$, is assigned by an LLM-based judge [212] on a 0–1 scale, evaluating user-valued objectives such as engagement / interactivity. Notably, additional conversational aspects, such as clarity, can be further integrated into the objective.

The conversation-level reward incorporates task-specific and human-centered metrics, encouraging the model to balance goal achievement, efficiency, and engagement.

2.4.3 2) Forward Sampling: Estimate long-term outcome

To compute Eq. 2.1, we require samples from $P(t_j^f | t_{1:j})$, the distribution of forward conversation conditioned on the conversation history. A simple approach is to use Monte Carlo sampling, where the conversation is extended turn-by-turn until it concludes. However, this can be computationally expensive for computing reward for every model response. For a scalable approximation, we introduce a window size w as a hyperparameter to limit the maximum number of forward turns considered in t_j^f . This reduces the computational cost while maintaining sufficient context.

More importantly, while real-world conversations could be gathered from human participants, sampling multiple forward conversations during training is costly and impractical. To further reduce cost and ensure scalability, we introduce a user simulator U .

User Simulator: *A user simulator $U : \mathcal{T} \rightarrow \mathcal{U}$ is a function that maps a given conversation history $t \in \mathcal{T}$ to a user response $u \in \mathcal{U}$. Specifically, U generates a probabilistic distribution $P(u | t)$ over possible user responses conditioned on the conversation history t , simulating realistic user behavior.*

Specifically, we prompt an LLM to role-play as users, explicitly asking the LLM to follow the same language style as the previous user turns, and injecting typical user behaviors. The user simulator operates with an implicit goal g , which it seeks to achieve over the course of the conversation. This design emulates real-world scenarios where users may have evolving needs, limited background knowledge, or require clarification, resulting in naturally unfolding multiturn conversations [122].

2.4.4 Online Training: How to optimize with multiturn-aware rewards

With the conversation-level reward function and forward sampling strategy, we can compute MR for any model response without requiring an additional reward model, which is often costly and slow to train. Unlike traditional single-turn reward approaches, MR explicitly accounts for the impact of a response on future conversations, promoting long-term

collaboration.

Further, we employ reinforcement learning (RL) methods such as PPO [140] and DPO [133] to guide the model in navigating complex conversations. By optimizing for higher MR, the model learns to generate responses that enhance overall effectiveness and efficiency by the end of the conversation.

2.4.5 Offline Training: How to generate offline datasets

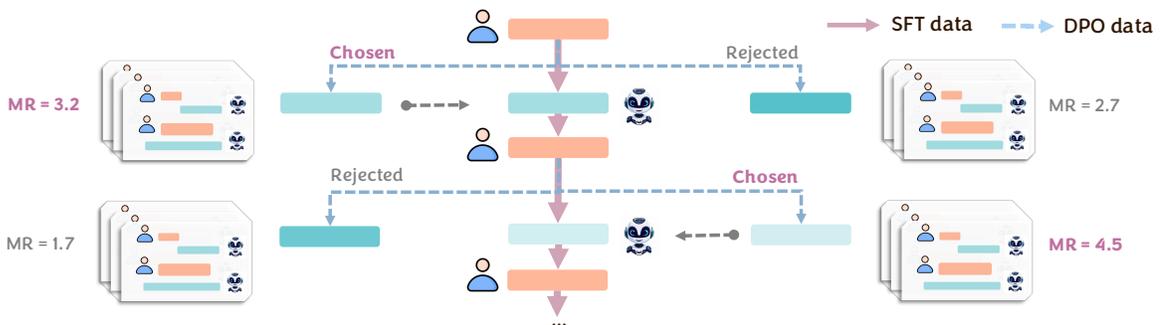


Figure 2.3: Generating high-quality conversation data with Multiturn-aware Rewards (MR).

Moreover, MR can generate **high-quality synthetic conversations** for both supervised fine-tuning (SFT) and DPO. In Figure 2.3, given a user query, multiple LLM responses are sampled and ranked based on their MR scores, with higher-ranked responses designated as *Chosen* and lower-ranked as *Rejected*. To simulate natural conversational flow, the first turn from the chosen response’s forward interaction window is appended to the prompt for the next turn, iteratively extending the conversation until completion. Solid red arrows denote data collection for Supervised Fine-Tuning (SFT), while dashed blue arrows indicate preference data construction for Direct Preference Optimization (DPO). This approach systematically curates multiturn conversations that enhance both response quality and collaborative efficiency, both of which are explicitly captured by MR.

Overall, COLLABLLM enables scalable dataset generation and online RL training without human annotation, making it generalizable across diverse tasks.

2.4.6 Extended Discussion I: Connections with causal inference

Our approach naturally aligns with causal inference principles, as it aims to quantify how a model’s response influences the future trajectory of a conversation. This aligns with the fundamental goal of **causal effect estimation**, which seeks to isolate the impact of an intervention—in this case, a model response—on long-term outcomes.

From a causal perspective, given a conversation history t_j^h at turn j , the **causal effect** of a model response m_j on the final conversation trajectory can be expressed using **front-door adjustment** [125, 126]:

$$\begin{aligned} \sum R^*(t_{1:K} | g)P(t_{1:K} | t_j^h)P(t_j^h) &= \sum R^*(t_{1:K} | g)P(t_{1:K} | t_j^h) \\ &= \mathbb{E}_{t_{1:K} \sim P(t_{1:K} | t_j^h)} R^*(t_{1:K} | g). \end{aligned} \quad (2.5)$$

This equation captures the expected long-term reward of a conversation conditioned on the model’s response at turn j . It explicitly accounts for how m_j intervenes in the conversation, influencing future turns and, ultimately, task success.

2.4.7 Extended Discussion II: Distinctions from related multiturn training frameworks

Existing multiturn trajectory-based training frameworks [47, 142, 217] primarily rely on learning from observed trajectory-level rewards. These methods estimate the utility of responses by assigning rewards post hoc to completed conversations, typically training models to prefer higher-rated conversations over lower-rated ones. However, this approach is fundamentally **observational**—it captures statistical associations between responses and final outcomes, without disentangling how individual responses causally influence future turns. For example, in MTPO [142], the learning signal remains coarse-grained: rewards are assigned at the trajectory level, and the influence of specific turns within a conversation remains confounded and indirect.

In contrast, our Multiturn-aware Reward (MR) framework **intervenes** on individual model responses and uses forward simulation to generate alternative future trajectories. This allows the model to estimate the **counterfactual impact** of different responses at each

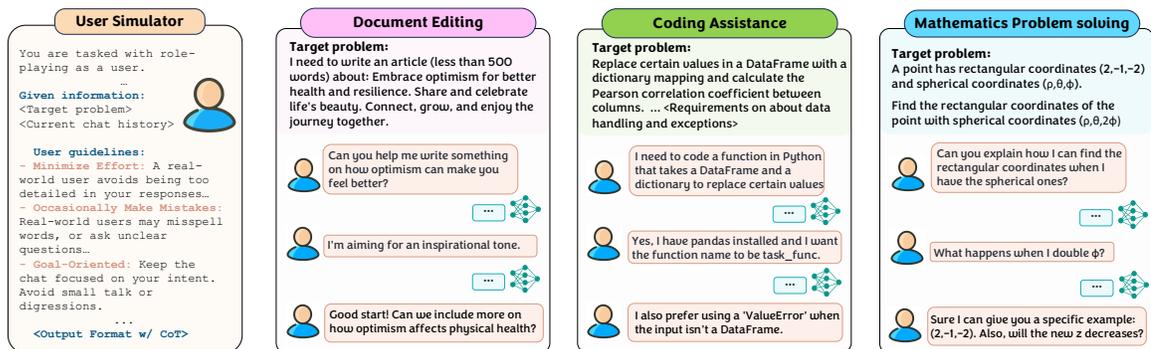


Figure 2.4: Simulated Multiturn Environment for Evaluation. Our evaluation pipeline simulates real-world collaborations by prompting a user simulator LLM to emulate diverse behaviors and personalities in multiturn conversations.

turn, thereby enabling fine-grained optimization. By leveraging causal effect estimation, MR training moves beyond passive imitation of high-reward conversations and instead actively selects responses to maximize long-term task success. This interventional approach provides turn-level credit assignment that is critical in dynamic human-LLM interactions, where user needs evolve and the consequences of early decisions compound over time.

2.5 Experimental Setup

For fine-tuning and evaluation, we create three multiturn datasets using publicly available data across diverse domains [33, 55, 222]: collaborative document editing, coding problem assistance, and multiturn mathematics problem solving.

To build a multiturn environment (Figure 2.4), we employ GPT-4o-mini as a user simulator LLM to role-play realistic user behaviors, given the target problem and conversation history. Our simulation-based evaluations are designed to closely mimic real-world interactions [122]. Unlike traditional single-turn tasks, our setup requires dynamic interactions over multiple turns to achieve a goal. The three interactive datasets are:

MediumDocEdit-Chat: Document editing requires iterative feedback and refinements across multiple turns to ensure coherence and alignment with user intent. We sample 100 Medium articles as goal documents, which are summarized into target problems to guide the user simulator. After each interaction, task performance is evaluated using the **BLEU**

score, measuring similarity between the extracted document and the original articles.

BigCodeBench-Chat: Coding tasks inherently require multiturn interactions, such as clarifying requirements and debugging. We sample 600 problems from BigCodeBench [222]. For evaluation, we compute the average **Pass Rate (PR)** of code at the end of the interactions.

MATH-Chat: Math problem solving often requires addressing implicit assumptions, verifying intermediate steps, and clarifying reasoning. We sample 200 level-5 math problems from MATH [55] to prompt the user simulator, which interacts with the LLMs. Task success is measured by the **accuracy (ACC)** of the final solution, as evaluated by an LLM judge.

In addition to the above task-specific metrics, we incorporate two task-agnostic scores across all datasets: **1) Average Token Count**, which quantifies the average number of tokens generated by the LLM per conversation, reflecting interaction efficiency. **2) Interactivity (ITR)**, which evaluates engagement levels using an LLM judge (Claude-3.5-Sonnet), with scores rescaled to an upper bound of 1.

Table 2.2: Statistics of conversational datasets created from MR. Chosen/Rejected MR indicates the mean and standard deviation (mean \pm std) of MRs for chosen and rejected responses (*cf.* Figure 2.3).

	# Train	# Turns	Average # Turns	Chosen MR	Rejected MR
MediumDocEdit-Chat	500	2,303	4.61	0.312 (± 0.104)	0.246 (± 0.113)
BigCodeBench-Chat	500	2,627	5.25	0.494 (± 0.621)	0.207 (± 0.763)
MATH-Chat	500	2,527	5.05	0.863 (± 0.524)	0.547 (± 0.502)

Fine-tuning COLLABLLMs. COLLABLLMs are based on Llama-3.1-8B-Instruct [103] with LoRA finetuning [60] on the training splits of our benchmarks (Table 2.2). We train four model variants: **1) Offline models:** SFT and Offline DPO are fine-tuned on pre-generated multiturn conversational datasets guided by Multiturn-aware Rewards (MR) (*cf.* Section 2.4.4). **2) Online models:** PPO and Online DPO are further trained from the SFT and Offline DPO models, respectively. The model during online fine-tuning is involved in the collaborative simulation to compute MRs, which, in turn, dynamically adjust the model preference.

Table 2.3: Evaluation results on our multiturn datasets. Green zone: Baselines; Orange zone: Variants of COLLABLLMs. Rel. Improv. indicates the relative improvements of CollabLLMs trained with Online DPO over Proactive Base.

	MediumDocEdit-Chat			BigCodeBench-Chat			MATH-Chat		
	BLEU \uparrow	#Tokens(k) \downarrow	ITR \uparrow	PR \uparrow	#Tokens(k) \downarrow	ITR \uparrow	ACC \uparrow	#Tokens(k) \downarrow	ITR \uparrow
Base	32.2	2.49	46.0	9.3	1.59	22.0	11.0	3.40	44.0
Proactive Base	35.0	2.18	62.0	11.0	1.51	33.7	12.5	2.90	46.0
SFT	35.2	2.21	68.0	11.7	1.35	42.0	13.5	2.88	58.0
PPO	38.5	2.00	78.0	14.0	1.35	40.7	13.0	2.59	52.0
Offline DPO	36.4	2.15	82.0	12.3	1.35	46.7	15.5	2.40	50.0
Online DPO	36.8	2.00	92.0	13.0	1.31	52.0	16.5	2.37	60.0
Rel. Improv.	5.14%	8.25%	48.3%	18.2%	13.2%	54.3%	32.0%	18.3%	36.4%

Baselines. We compare COLLABLLMs against (1) the pretrained Llama-3.1-8B-Instruct (*Base*), (2) the base model with proactive prompt engineering (*Proactive Base*), which encourages follow-up and clarification questions.

2.6 Simulated Experiments

2.6.1 Main Results: How effective is CollabLLM under simulation

We present the results in Table 2.3 and the takeaways are:

Prompt engineering is helpful, but limited in terms of performance gains and flexibility. Proactive Base improves base model performance by encouraging follow-up questions and clarifications. For example, it increases BLEU on MediumDocEdit-Chat from 32.2% to 35.0% and reduces read tokens by 0.31k compared to the base model. However, these gains are modest and do not fully address the challenges of multiturn collaboration. We observe that prompting strategies remain rigid, relying on predefined instructions rather than adapting dynamically to user needs. For instance, the model sometimes asks clarification questions even when unnecessary, leading to redundant interactions that disrupt conversation flow.

COLLABLLM improves task performance, efficiency, and engagement. COLLABLLM achieves 18.5% superior task-specific performance, 13.3% more efficient conversations,

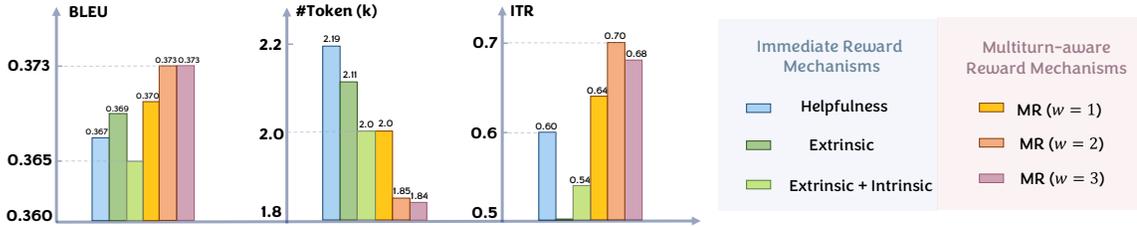


Figure 2.5: Selected Ablation Study of Reward Mechanisms on `MediumDocEdit-Chat`. This figure compares three immediate reward mechanisms with three MR variants. The results demonstrate that MR consistently improves task-specific performance (BLEU), conversational efficiency (# Tokens), and interactivity (ITR).

and 46.3% enhanced interactivity compared to the best baselines. We highlight that `COLLABLLM` engage in more meaningful collaborations, with ITR shows substantial gains. For `MediumDocEdit-Chat`, the Online DPO model increases ITR from 0.46 to 0.92. Moreover, our framework significantly improves conversational efficiency by minimizing the content users need to review to arrive at the final solution. For `MATH-Chat`, Online DPO decreases token count per conversation by 1.03k compared to the base model.

2.6.2 Ablations: What mechanism makes `CollabLLM` effective

To investigate how components contribute to `COLLABLLM`'s superior performance, we conduct an ablation study focusing on the reward mechanisms used during fine-tuning. We evaluate the following reward mechanisms:

- **Variants of Multiturn-aware Reward:** We vary the forward sampling window size $w = 1, 2, 3$ to assess their ability to capture long-term conversational effects through simulated collaborations.
- **Immediate Rewards** evaluate the model's immediate response based on: 1) *Helpfulness*: Assessed by an LLM judge; 2) *Extrinsic Reward*: Focuses on task-specific metrics like BLEU while ignoring intrinsic factors such as efficiency; 3) *Extrinsic + Intrinsic Reward*: Combines task-specific metrics with efficiency and interactivity measures. This can be seen as a special case of the multiturn-aware reward function with $w = 0$.

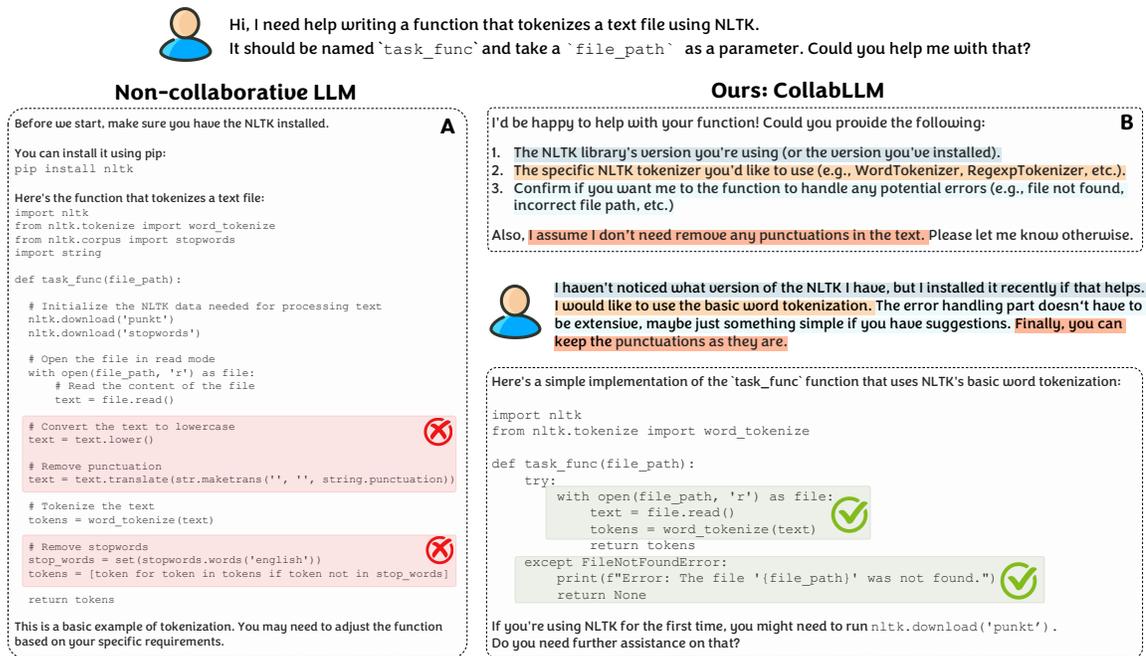


Figure 2.6: Case study on BigCodeBench-Chat. The non-collaborative LLM assumes user needs, adding unnecessary steps like punctuation and stopword removal. In contrast, COLLABLLM clarifies tokenizer preferences, error handling, and package installation, leading to a solution that precisely aligns with user intent.

We present results in Figure 2.5. Interestingly, expanding the forward sampling window w within the range generally enhances performance and efficiency by better capturing future interactions. Notably, MR with $w = 2$ balances the gains and additional costs to conduct forward sampling, making it well-suited for large-scale fine-tuning. In contrast, immediate rewards, even with extrinsic and intrinsic components, fall short as they ignore long-term impact. These findings validate the positive impact of the forward sampling strategy in MRs.

2.6.3 Case Study: What does CollabLLM learn

We now offer a deeper insight into COLLABLLM's behavior as shown in Figure 2.6. In this example, the user request to tokenize a text file is inherently open-ended due to unspecified factors, such as the NLTK environment, tokenizer selection, and optional preprocessing

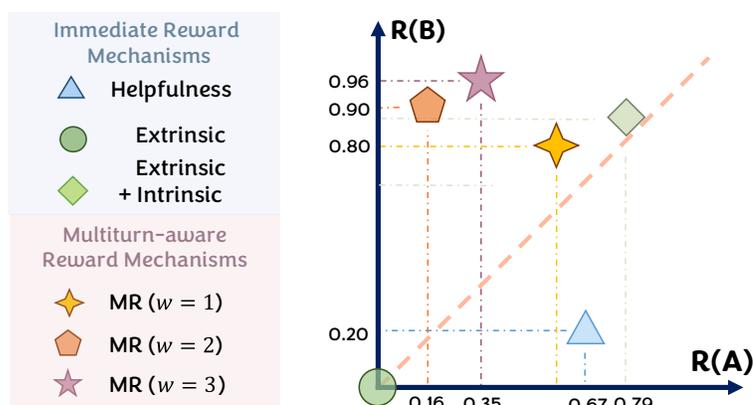


Figure 2.7: Rewards of response A and B in Figure 2.6 show different preferences.

steps. The base LLM makes several arbitrary assumptions, applying lowercase conversion and stopword removal without user confirmation. The user simulator later corrects these assumptions, but the final solution remains incorrect due to missing stopwords. In contrast, COLLABLLM actively clarifies user intent by seeking confirmation on key decisions, ensuring an aligned final solution with a 100% Pass Rate. This approach also reduces user effort with lower token usage.

In Figure 2.7, we compare different reward mechanisms for responses A and B of Figure 2.6, to confirm that these rewards work as intended. The helpfulness rewards favor response A due to its seemingly more well-round output. Extrinsic rewards assign zero scores to both, as A provides an incorrect solution and B defers answering. Extrinsic + Intrinsic rewards slightly favor B for efficiency and engagement. Interestingly, MR assigns significantly higher rewards to B, especially at $w = 2$ and $w = 3$, since the response obtains useful information and provide a precise answer within the future interaction window.

2.6.4 Generalization: How does CollabLLM generalize to unseen data

Modern foundation models are expected to generalize across a diverse range of tasks beyond their training domain. A key question is whether collaborative behaviors learned by COLLABLLM during fine-tuning transfer effectively to new tasks without additional adaptation.

	Action-level Accuracy		Macro Metric	
	Ambiguous	Non-Ambiguous	Accuracy	F1
GPT-4o	15.44%	95.60%	55.52%	56.62%
Llama-3.1-8B-Instruct	16.26%	90.40%	53.33%	53.31%
COLLABLLM	52.84%	72.32%	62.58%	55.08%

Table 2.4: Zero-shot generalization to Abg-CoQA, a conversational QA benchmark for ambiguity identification. Action-level accuracy measures whether the model asks a question for ambiguous inputs and provides a direct answer for non-ambiguous ones.

We assess COLLABLLM, trained with online DPO on BigCodeBench-Chat (the coding assistance task), on Abg-CoQA [49], a question-answering (QA) benchmark where questions are labeled as ambiguous or non-ambiguous. We use the following prompt format for the LLMs to answer the question given a story.

```

1   Can you help me answer a question about the following story?
2
3   {story}
4
5   My question is: {question}

```

For example:

```

1   Can you help me answer a question about the following story?
2

```

3 *I spent last weekend with my grandma and grandpa. I love them very much! I always look forward to visiting them! They always do fun things with me. Last weekend, we went to the zoo together. I saw a great big elephant. It had a long nose. My grandpa and I played a game to see who could be the most like an elephant. We stomped around a lot and made trumpeting noises. I won! Grandma looked on and laughed. I saw a monkeys too! The monkeys swung through the trees. They even made monkey noises! Grandma wanted to take a picture of me with the monkeys, but I was too busy pretending I was monkey to stand still. After we left the zoo, I went home. We had dinner together. Then, my grandma read me a story and tucked me into bed. I had a great time with my grandparents. I love them a lot. I always look forward to visiting them.*

4

5 *My question is: Where did they go when they left?*

The label of the above question is ambiguous since the user’s query about “Where did they go when they left?” could mean “Where did they go when they left the zoo?” or “Where did the grandparents go when they left me?”.

We categorize the model’s responses into two actions—asking a clarifying question or providing a direct answer—and evaluate action-level accuracy within each question type. As shown in Table 2.4, GPT-4o and Llama-3.1-8B-Instruct rarely ask clarifying questions regardless of ambiguity. In contrast, COLLABLLM asks questions about 50% of the time while maintaining high accuracy on unambiguous inputs. This behavior leads to the highest Macro Accuracy across both ambiguous and non-ambiguous sets and improves Macro F1 over the base model, while leaving room for further improvement against GPT-4o. These results suggest that **COLLABLLM effectively generalizes its learned collaborative strategies beyond its training domain.**

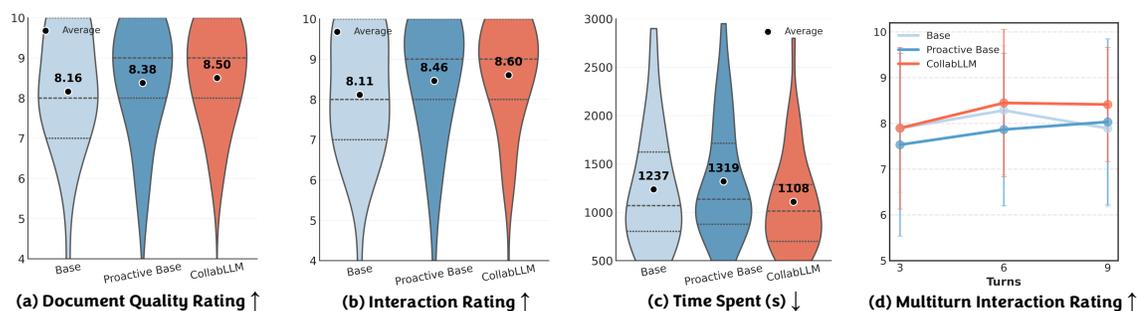


Figure 2.8: Real-world user study with 201 participants interacting with an anonymized AI assistant randomly sampled from Base, Proactive Base, and COLLABLLM. Participants rate (a) document quality and (b) overall interaction experience, with additional assessments (d) every three turns. We also measure (c) user spent time to evaluate efficiency.

2.7 Real-world User Study

2.7.1 Interface: What is our collaboration environment

We conduct a large-scale user study on Amazon Mechanical Turk with 201 participants. Each participant is assigned a document type, randomly selected from blog post, creative writing, and personal statement, and then chooses a topic from a predefined set. To reflect real-world scenarios where users begin with only a rough idea of the task, participants first answer a few topic-related questions with brief responses. Participants then engage in at least eight turns of conversation with an anonymized AI assistant, which is either Base, Proactive Base, or COLLABLLM. During the interaction, participants provide an interaction rating every three turns. After the conversation, they rate the final document quality and the overall interaction. All ratings use a 1 to 10 scale. We also record total interaction duration to assess efficiency. All data collection is anonymized to protect user privacy.

Interface overview. Figure 2.9 shows the landing page. It summarizes the three-step task, presents a data collection notice and participation notes, and requires participants to acknowledge the notices before starting.

Creating Writing Contents with AI Assistants [Version: 10x Speed Up]

In this task, you will collaborate with an AI assistant to create a document. This task is broken into three simple steps designed to guide you through the experience.

1. Step 1: Select Your Intent Given Document Type

You will be given a document type from *Blog Post*, *Creative Writing*, *Personal Statements*. Choose your intent (writing goal) from provided options.

2. Step 2: Pre-Writing Preparation

Jot down some pre-writing details for your selected intent, this is for you to organize your thoughts like some rough plan on what to write.

3. Step 3: Collaborate with the AI Writing Assistant

Interact with the AI to create your document.

<p>Data Collection Notice:</p> <p>Before you begin, please note that by checking the box below, you agree to:</p> <ul style="list-style-type: none"> ◦ Allow us to collect your annotations for research ◦ Have your annotations shared publicly as part of our research data <p>To protect your privacy, please do NOT include any personal identifying information (PII) in your annotations. For instance, if you are writing a personal statement, you can use a fictional character and create a statement about them. If you prefer not to participate, you can simply close this window.</p> <p><input type="checkbox"/> I agree that my annotations will be collected and shared publicly as research data.</p>	<p>Important Note:</p> <ul style="list-style-type: none"> ◦ (Jan 20 18:00pm PT) We speed up the AI response generation by 10x. Slow responses may due to very busy traffic, please be patient. ◦ You should try to create content that's relevant and useful in your real life. For example, you can ask AI to customize the documents to match your personal writing style. ◦ Please don't paste all your pre-writing notes in the 2nd step at once when you chatting with the AI. Think of these as your private brainstorming steps—share them gradually, like how you naturally talk to a chatbot and develop a document. <p><input type="checkbox"/> I have read these notes carefully.</p>
<p>Let's Start the Task</p>	

Figure 2.9: Overall interface. Landing page describing the three-step writing task, along with a data collection notice and an important note that participants must acknowledge before clicking “Let’s Start the Task.”

Step 1: Select intent. In Step 1 (Figure 2.10), participants select a writing intent from an intent bank given the assigned document type. The interface displays the selected intent together with pre-writing questions to guide planning.

Step 1

Given the document type: creative writing
 Choose intent that you are interested in writing about.

☰ Intent Bank

Index	Intent
17	Parody Retelling: Write a humorous and creative reinterpretation of a classic story or historical event in a modern or unexpected context.
39	Cyberpunk Mystery: Write a futuristic detective story in a neon-lit world of advanced tech, corporate control, and virtual reality heists.
5	Mermaid Tale: Plunge into an underwater kingdom, exploring its politics, folklore, and interactions with land dwellers.
27	Historical Fiction: Write a story set in a specific time period of the past, where historical events and social conditions shape a character's personal journey.
33	Utopian Society: Imagine a story set in a near-perfect world and explore the hidden costs or unintended consequences of maintaining that society.
46	Dark Fairy Tale: Weave a haunting reimagining of a whimsical setting, introducing sinister twists, eerie creatures, and moral complexities.
0	Space Western: Combine frontier adventures and futuristic technology in a galaxy of outlaws and pioneers.
25	Contemporary Life: Write a story about professional relationships and personal growth in a contemporary work setting.
23	Mystery Thriller: Write a suspenseful story about a detective solving an intriguing case.
2	Musical Script: Write a story where characters express key moments through song or rhythmic dialogue.
49	Anthropomorphic Critters: Create a whimsical narrative starring talking animals or creatures, exploring human-like societies, conflicts, and resolutions.
45	Animal Perspective: Tell a tale from the viewpoint of an animal, exploring how they perceive human actions, habitats, or changing environments.

Selected Intent: Cyberpunk Mystery: Write a futuristic detective story in a neon-lit world of advanced tech, corporate control, and virtual reality heists.

Pre-writing Questions for this Intent:

1. What does your cyberpunk setting look like—are there mega-cities, flying cars, omnipresent neon lights?
2. Who is your detective or investigator, and what personal codes or flaws define them?
3. What advanced technology is central to the plot—VR chips, AI companions, cybernetic implants?
4. What corporate entities or crime syndicates rule this futuristic society, and how do they exploit the masses?
5. What mysterious crime or data theft needs solving, and how is it linked to larger conspiracies?
6. Who are the suspects or informants, and how do their augmentations or affiliations complicate the investigation?
7. How do you weave elements of noir—gritty streets, moral ambiguity—into the futuristic narrative?
8. What obstacles does the detective face—corruption, technology hacking, betrayals from allies?
9. Does the detective rely on advanced tools or old-school intuition to crack the case?
10. How do you want readers to feel about the blurred line between humanity and machinery at the conclusion?

Click the button below to start jotting down some thoughts for the above questions

Proceed to Step 2: Pre-Writing Preparation

Figure 2.10: Step 1 interface. Participants choose an intent from an intent bank. The right panel shows the selected intent and a list of pre-writing questions, with a button to proceed to Step 2.

Step 2: Pre-writing preparation. In Step 2 (Figure 2.11), participants jot down pre-writing notes by answering the provided questions. They can add new questions or remove existing ones, and the interface enforces minimum completion requirements.

Step 2

Pre-writing: Jot down some thoughts for your intent.

Your Intent: Cyberpunk Mystery: Write a futuristic detective story in a neon-lit world of advanced tech, corporate control, and virtual reality heists.

Note: You can customize the questions to suit your needs—feel free to add new ones, remove existing ones, or leave fields blank if certain aspects don't apply to your planning.

However, please make sure that you answer at least 6 questions. Each answer should be longer than 8 words.

In very rare case, if you encounter problems submitting, you might need to refresh the website, which may assign a different document type. We have taken this rare error into account when set up the reward. Our apologies for the inconvenience.

Add New Question

1. What does your cyberpunk setting look like—are there mega-cities, flying cars, omnipresent neon lights?

Remove

2. Who is your detective or investigator, and what personal codes or flaws define them?

Remove

Figure 2.11: Step 2 interface. Pre-writing form for the selected intent. Participants answer planning questions in text fields and can add new questions or remove questions. The interface instructs participants to answer at least six questions, with each answer longer than eight words.

Step 3: Multi-turn collaboration. In Step 3 (Figure 2.12), participants converse with the assistant to create the document. The interface shows the selected intent, provides interaction guidelines and task requirements, and allows participants to view their pre-writing notes as a reference.

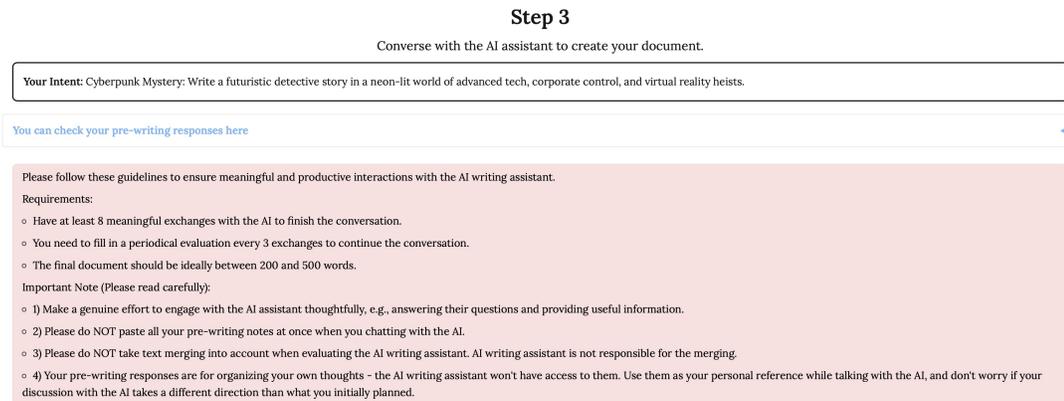


Figure 2.12: Step 3 interface. Chat interface for collaborating with the assistant. The view displays the selected intent, task requirements (at least eight exchanges, periodic evaluation every three exchanges, and a target document length), and a collapsible panel to review pre-writing notes.

Periodic and final evaluations. During Step 3, participants complete a periodic evaluation every three exchanges (Figure 2.13). After finishing the task, they complete a final evaluation that includes free-form feedback and separate ratings for interaction experience and the final document (Figure 2.14).

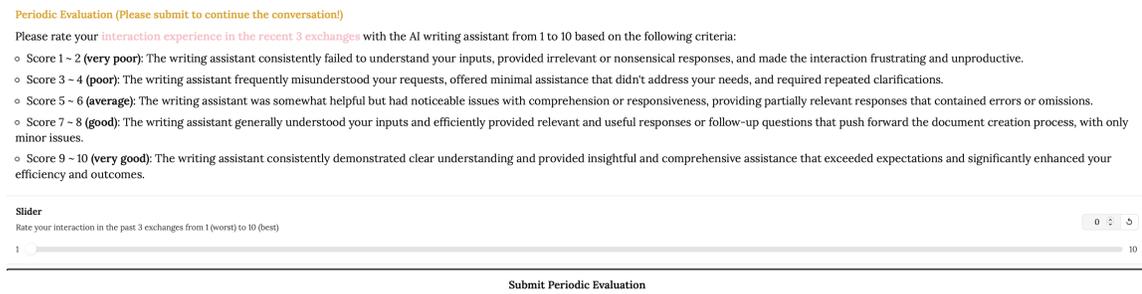


Figure 2.13: Multi-turn evaluation view. Periodic evaluation form that participants must submit to continue the conversation. It provides rating guidelines and a 1–10 slider for the interaction experience over the most recent three exchanges.

Five survey questions:

1. What are the strengths of the AI writing assistant? 2. What are the weaknesses of the AI writing assistant?

strengths of the AI writing assistant... weaknesses of the AI writing assistant...

3. Why did you end the session?

I am satisfied with the document the AI created. I found the AI writing assistant not helpful. I encountered technical issues with the interface. Other reason

4. Please rate your **interaction experience** (the conversation) with the AI writing assistant from 1 to 10 based on the following criteria:

- Score 1 - 2 (**very poor**): The writing assistant consistently failed to understand your inputs, provided irrelevant or nonsensical responses, and made the interaction frustrating and unproductive.
- Score 3 - 4 (**poor**): The writing assistant frequently misunderstood your requests, offered minimal assistance that didn't address your needs, and required repeated clarifications.
- Score 5 - 6 (**average**): The writing assistant was somewhat helpful but had noticeable issues with comprehension or responsiveness, providing partially relevant responses that contained errors or omissions.
- Score 7 - 8 (**good**): The writing assistant generally understood your inputs and efficiently provided relevant and useful responses or follow-up questions that push forward the document creation process, with only minor issues.
- Score 9 - 10 (**very good**): The writing assistant consistently demonstrated clear understanding and provided insightful and comprehensive assistance that exceeded expectations and significantly enhanced your efficiency and outcomes.

Interaction Rating
rate from 1 (poor) to 10 (best)

1 10

5. Please rate the final document created by the AI writing assistant from 1 to 10 based on the following criteria:

- Score 1 - 2 (**very poor**): The document contains numerous errors, inaccuracies, or irrelevant content, lacks coherence and structure, and is unusable for your objective.
- Score 3 - 4 (**poor**): The document has significant issues such as incomplete sections, misleading information, or poor organization, only partially addresses your instructions, and requires substantial revisions.
- Score 5 - 6 (**average**): The document meets basic requirements but includes noticeable errors or omissions, provides some useful content but lacks depth or clarity, and requires moderate revisions to improve quality.
- Score 7 - 8 (**good**): The document is well-organized, covers the key topics as instructed, contains accurate and relevant information with minor errors, and serves as a strong foundation that fulfills your main objective.
- Score 9 - 10 (**very good**): The document is comprehensive, insightful, and meticulously crafted, exceeds expectations by providing exceptional clarity and depth, requires minimal to no revisions, and significantly achieves your objective.

Document Rating
rate from 1 (poor) to 10 (best)

1 10

Figure 2.14: Final evaluation view. Post-task survey with free-form questions, a 1–10 interaction rating slider, a 1–10 final document rating slider, and a submission button to complete the task.

2.7.2 Results: How effective is CollabLLM in the real world

Quantitative Results (Figure 2.8). Across multiple metrics, COLLABLLM consistently outperforms the baselines. It achieves an average document quality score of 8.50. Specifically, 91.4% of participants rate COLLABLLM’s **document quality** as “good” (score 8–9), and 56.9% as “very good” (score 9–10), compared to 88.5% and 39.3% for Base (Llama-3.1-8B-Instruct), respectively. Similarly, 63.8% of participants find COLLABLLM **highly engaging**, while only 42.6% report the same for Base.

Interestingly, for **multiturn interaction**, the Base model shows a declining trend in ratings from turns 6–9, indicating reduced user experience in longer conversations. In contrast, both COLLABLLM and Proactive Base exhibit increasing ratings over time, with COLLABLLM consistently achieving higher average ratings every three turns compared to Proactive Base. This suggests that COLLABLLM maintains sustained engagement more effectively.

Moreover, COLLABLLM improves task efficiency, reducing **time spent** by 10.4% compared to the Base model and by 15.6% relative to Proactive Base. While Proactive Base is

prompted to maintain conciseness, it frequently asks unnecessary questions, causing lower efficiency. In contrast, COLLABLLM strikes a more streamlined user experience.

Qualitative Results (Table 2.5). We collected a total of 180 strengths and 180 weaknesses across the three models. Table 2.5 presents representative feedback, while we summarize here the models’ strengths and weaknesses: The base model generates coherent content while effectively following user instructions, but it sometimes struggles with maintaining context in long texts, and can be overly verbose or repetitive in its responses. Proactive Base excels in responsiveness and adapting to user input but struggles with memory retention, and could produce repetitive or overly structured content. On the other hand, COLLABLLM is highly engaging, effectively guiding users through writing, adapting seamlessly to feedback. However, users also point out that COLLABLLM can occasionally feel bland, lack up-to-date information, and require additional effort to personalize the output. Overall, COLLABLLM enhances collaboration by guiding users through an interactive and iterative refinement process, yet future improvements should focus on increasing personalization, creativity, and real-time knowledge integration to further optimize human-LLM collaboration.

2.7.3 Extended Discussion III: What are the differences and similarities between simulated users and real users

While user simulators were employed exclusively during training due to the large-scale conversation demands of our Multiturn-aware Reward computation, we provide a comparative analysis to study the divergence between user simulators and real users. In Table 2.6, we summarize key differences and similarities in communication patterns between real and simulated users.

Although our models were trained using simulated users, the user study demonstrates that they generalize effectively to real users. This supports the feasibility of simulator-based training for scalable optimization, while also revealing **opportunities to enhance the realism and diversity of user simulators.**

Table 2.5: Representative feedback from human participants on strengths and weaknesses of each model.

Model	Strengths	Weaknesses
Base	<i>“Follows great instruction and does exactly what I’m asking it to do.”, “It can create a nice form of an outline to work with.”</i>	<i>“The AI just agreed with me on pretty much everything. There was no discussion”, “I didn’t really like that it kept coming up with different options”</i>
Proactive Base	<i>“It is very organized and it actually asks you for feedback after writing the revision.”</i>	<i>“The AI seemed to be very redundant and asked me the same questions over and over.”</i>
Collab-LLM	<i>“Asking questions and making you think of things you never thought of”, “The AI really helped me with focusing on one part of the story at a time.”, “It helped really well to navigate what to say and what information is needed”</i>	<i>“The AI assistant was not up to date enough to help with this recent sporting event. The AI assistant also asked me to repeat information I had already given it.”</i>

2.8 Safety Evaluation: Is CollabLLM safe to deploy

As the models in this work are collaboratively trained and designed to be more aligned with the user’s intent, concerns may arise if a user happens to have malevolent intentions. However, we note that COLLABLLM models were finetuned from Llama-3.1-8B-Instruct, which has been aligned for safety—so jailbreaking COLLABLLM still poses a significant challenge. To determine whether collaborative training weakens the safety features inherent to a model (Llama-3.1-8B-Instruct) that has undergone significant alignment steps for safety, we performed an adversarial evaluation using the Azure AI Evaluation SDK* and prompted both the baseline and COLLABLLM with various offensive queries intended to elicit unsafe responses.

Specifically, we performed the following steps:

*<https://learn.microsoft.com/en-us/python/api/overview/azure/ai-evaluation-readme>

Table 2.6: Comparison of key differences and similarities in communication patterns between simulated and real users.

Differences	Similarities
<ul style="list-style-type: none"> • Real users tend to write shorter, fragmented sentences and may include grammatical errors, while simulators produce more complete and polished responses. • Real users often change directions mid-conversation and introduce specific personal details (e.g., “eight dogs”), while simulated users are typically more predictable and generic. • Real users express emotion more bluntly and use informal language, abbreviations, or incomplete thoughts, while simulators often respond in a more neutral and formal tone. 	<ul style="list-style-type: none"> • Both develop content iteratively by revealing information over multiple turns rather than specifying everything upfront. • Both request accessible assistance, such as simplifications, examples, and actionable guidance. • Both state preferences about structure or style and provide feedback when expectations are met or unmet.

- **Adversarial query selection:** We used the SDK’s `AdversarialSimulator` to generate adversarial queries (e.g., queries encouraging the LLM to produce hateful comments). We then used the SDK’s harm evaluators (`ViolenceEvaluator`, `SexualEvaluator`, `SelfHarmEvaluator`, `HateUnfairnessEvaluator`) to categorize each query into one of four harm types: violence, sexual, self-harm, and hate. For each query, we used the highest score among the four evaluators to determine its harm category. We randomly selected 20 adversarial queries per harm category, resulting in a total of 80 queries.
- **Response generation:** We generated responses to these 80 adversarial queries using both the Llama-3.1-8B-Instruct baseline model and COLLABLLM.
- **Harm scoring:** We evaluated each model-generated response using all four harm evaluators to ensure comprehensive assessment.

The main safety results are shown in Table 2.7, which presents the average harm scores

Table 2.7: Harm scores of responses generated by the two models under adversarial prompting. Score range is 0-7, with values between 0 and 1 indicating “very low” harm.

Model	Harm score (0–7 range, ↓)			
	Violence	Sexual	Self-harm	Hate
Llama-3.1-8B-Instruct	0.88	0.96	0.89	1.01
COLLABLLM	0.95	0.94	1.00	0.99

across the four categories. Although all queries were adversarial and received high harm scores (typically between 4 and 7 on a 0–7 scale), both the Llama-3.1-8B-Instruct baseline and COLLABLLM produced responses that were, on average, very safe. Most scores are in the 0–1 range, which corresponds to “very low” harm. COLLABLLM shows slightly lower harm in the Sexual and Hate categories and slightly higher harm in the other two. In terms of defect rate, COLLABLLM produced only one response deemed unsafe by the SDK (out of 80 queries \times 4 categories = 320 evaluations), resulting in a pass rate of 99.7%. Coincidentally, this is the same pass rate as Llama-3.1-8B-Instruct, which also had one failed evaluation.

Overall, these results are encouraging. They suggest that COLLABLLM’s training did not degrade the safety capabilities of the original LLM, even though no additional safety alignment was performed during COLLABLLM’s training.

2.9 Extended Direction: Improving User Simulators

As we see from the framework, COLLABLLM relies on interaction data and multi-turn rollouts to shape agent behavior. The multi-turn rollouts are expensive and slow to collect with real users, and it is often infeasible to run enough iterations to train policies with these limited dataset. Therefore, a workaround in COLLABLLM is to use an *LLM-prompted user simulator*, where an LLM is instructed to play the role of a user and respond to the agent across multiple turns. This setup is promising because it is feasible, scalable, and easy to control.

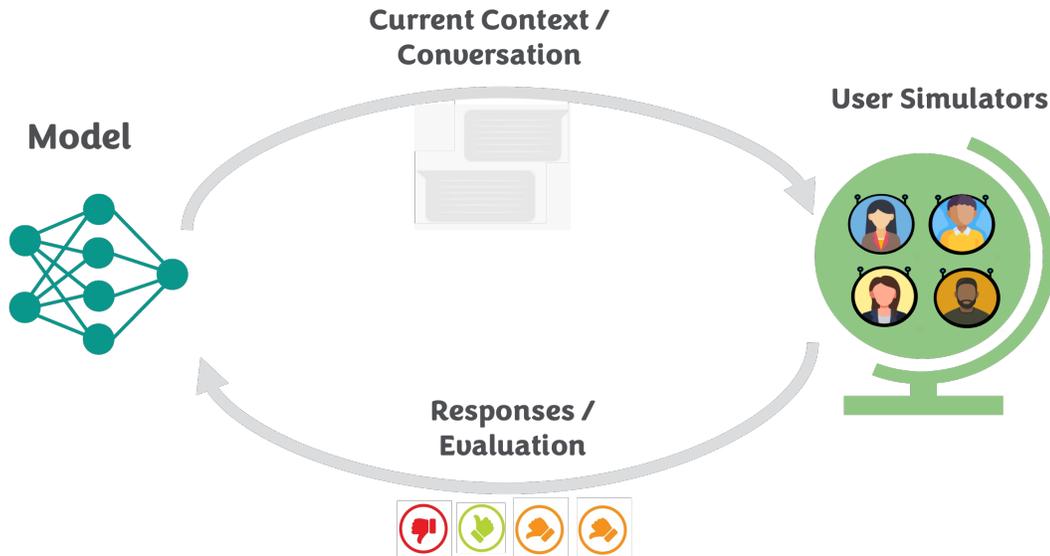


Figure 2.15: **User simulators as a feedback loop for collaborative training and evaluation.** Given a current context or conversation, the model produces an action or response. A user simulator then generates user reactions and provides feedback signals that score the interaction. The data generated from this loop is used to update or evaluate the model.

2.9.1 Motivation: Why better user simulators matter

However, we found limitations of LLM-prompted simulators. For example, they often respond in a way that is cooperative, but not representative of how real users behave. A simulator may accept vague answers, fail to enforce its own constraints consistently, or provide information that a real user would not know. When this happens, collaborative training can optimize for behavior that looks good under simulation but does not transfer well to deployment. This makes simulator quality a central issue for both training and evaluation.

More specifically, a collaborative agent is trained to perform well *over a conversation*, not only to produce a good single response. This means the training signal depends on how the user reacts across turns. If a simulator reacts differently from real users, then the agent is effectively trained against the wrong environment. In particular, an overly helpful simulator can hide important failure modes. It may reward an agent that makes unchecked assumptions, ignores constraints, or skips clarification, simply because the simulator still

provides a cooperative next turn. The resulting agent may appear strong in simulated roll-outs, yet frustrate real users who correct the agent, reject generic answers, or stop the interaction early.

Better simulators reduce this mismatch by reflecting the factors that drive user reactions beyond surface wording. In real interactions, users have relatively stable preferences, stances, and habits for how they communicate. These factors influence whether a user accepts a suggestion, challenges it, becomes dissatisfied, or provides additional constraints. If a simulator preserves these factors across turns, then collaborative training can reward behaviors that matter in deployment, such as asking targeted questions, maintaining constraints consistently, and adapting when the user disagrees.

2.9.2 Challenges: Why user simulation is difficult

Improving simulators is non-trivial because realistic user behavior is hard to specify and hard to measure [178].

Non-determinism. Even the same user may respond differently to the same situation, using different wording or different levels of detail. This makes it a poor goal to reproduce the exact ground-truth response. For collaborative training, what matters more is whether the simulator reacts in a way that is consistent with the user’s underlying intent and constraints.

Multiple drivers of behavior. A single response can reflect several factors at once, such as a preference, a stance, and a communication habit. An LLM-prompted simulator can match one factor while drifting on another. For example, it may keep the tone consistent but stop enforcing the original constraint, which changes the training signal the agent receives.

Feedback-loop effects. In multi-turn training, the agent’s behavior changes what the simulator will say next. This means simulator errors can compound. If the simulator tolerates mistakes that real users would push back on, the agent never learns to recover. Conversely, if the simulator overreacts to small mistakes, the agent may learn overly cautious behaviors that reduce helpfulness.

2.9.3 Realistic Simulation for Collaborative Training and Evaluation: What are the promising approaches?

To make simulators more useful for collaborative training, we aim to make simulated interactions resemble deployment interactions in the aspects that matter for learning and evaluation. Below we list concrete directions and explain how each one connects to training and evaluation.

Constraint-consistent simulation. A simulator should enforce its own stated constraints across turns. Practically, this means tracking what the user has said (preferences, requirements, refusals) and responding in a way that remains relatively consistent. For collaborative training, this creates a clearer reward for behaviors like clarification and constraint tracking. For evaluation, it allows stress-testing whether an agent can maintain constraints over long conversations.

State-aware simulation. Instead of prompting a simulator to imitate surface style, we can prompt or train it to preserve high-level user states that drive behavior, such as stance, preference, and communication approach. The simulator then generates a response that is consistent with these states even if the wording varies. For collaborative training, this provides a more stable target and reduces overfitting to phrasing. For evaluation, it supports metrics that test whether an agent preserves the user’s intent and avoids unwanted shifts in stance or preference.

Calibration with real interactions. Even a carefully designed simulator can drift from real user behavior. A practical approach is to periodically compare simulator trajectories against a small set of real user interactions and adjust prompts, state definitions, or training data accordingly. For collaborative training, this reduces the train-deploy gap. For evaluation, it increases confidence that benchmark performance reflects real-world performance.

Overall, user simulation is not just a convenience for scaling experiments. It directly shapes what collaborative training methods learn and how we evaluate them. Improving simulators therefore strengthens both sides of the loop: it provides more reliable training signals for methods like COLLABLLM, and it enables evaluation settings that better predict how agents will behave with real users.

2.10 Conclusion

Multiturn human-LLM collaboration is increasingly common in real world applications, from drafting and editing documents to planning workflows and supporting complex analysis. In these settings, foundation models should act as collaborators rather than passive responders. They should help clarify intent, surface missing constraints, and adapt over the course of an interaction, especially when the task is open ended. Yet current LLMs often fall short: they may make early assumptions, fail to ask useful questions, or optimize for a locally good response that harms the conversation later.

A central takeaway of this work is that enabling collaboration does not always require massive changes to the model or the interface. It requires a better objective and a longer lens for evaluation. The key insight of *COLLABLLM* is to make LLMs more multiturn aware by optimizing for long term outcomes, using forward sampling to estimate the downstream impact of candidate responses. Through extensive simulated and real world evaluations, we show that *COLLABLLM* consistently improves collaborative performance over non collaboratively trained baselines. More importantly, users report interactions that feel more efficient, engaging, and reliable.

Looking forward, the applications of collaborative AI are broad. Beyond assisting with everyday tasks, collaborative agents can support higher stakes and higher complexity work, including scientific and technical problem solving where progress depends on iterative exploration and careful coordination. Ultimately, our goal is to democratize collaborative AI so that effective human-AI collaboration is accessible to anyone building AI powered solutions.

Part II

Improving Agents to Learn from Experience

Chapter 3

Benchmarking Agents in Solving User Queries

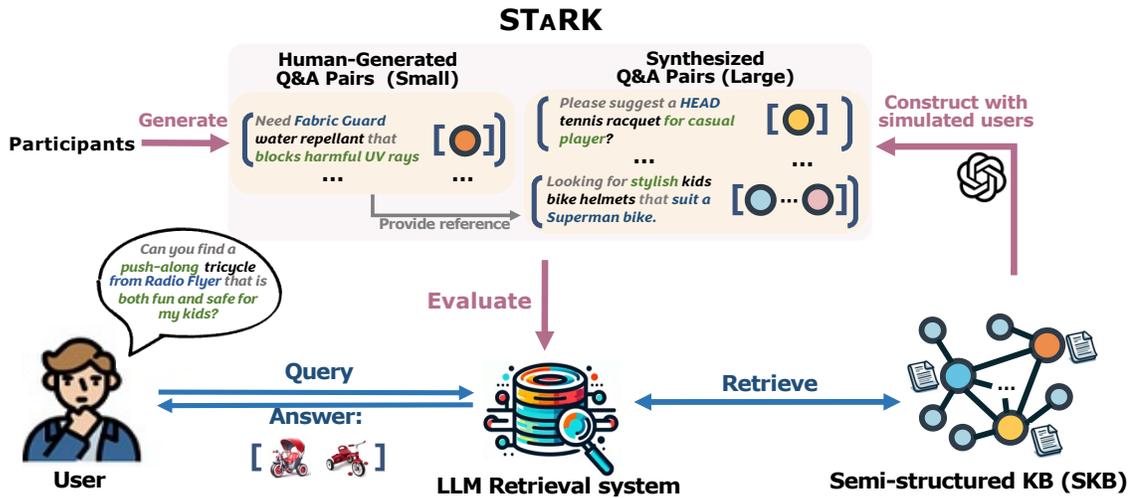


Figure 3.1: STARK evaluates user-centered reasoning for agents by benchmarking retrieval on Semi-structured Knowledge Bases (SKBs). Because users naturally express queries over heterogeneous knowledge that blend textual descriptions and structured relationships, the benchmark assesses whether agents can accurately interpret these holistic user needs and retrieve correct answers. It consists of synthesized queries simulating realistic interactions and human-generated queries, with node entities as ground-truth answers.

3.1 Introduction

Collaborative agents must understand and respond to complex user queries to effectively assist users in achieving their goals. A core capability for collaboration is *user-centered reasoning*: accurately interpreting what users are asking for, understanding their intent across multiple constraints, and providing assistance that directly addresses their needs.

Complex user queries require reasoning over heterogeneous knowledge. When users collaborate with agents, they naturally express complex information needs through natural language queries [56, 69, 79]. These queries are *reasoning-intensive* because they require agents to understand and combine constraints from heterogeneous knowledge sources. By heterogeneous knowledge, we mean information stored in different representations: unstructured text (descriptions, reviews, documents) and structured relationships (categories, citations, gene-protein associations). Users express their needs holistically, referring to both types of knowledge simultaneously, because they think about their goals as a whole rather than in terms of how information is stored or queried.

Consider two representative examples. A parent shopping for a tricycle asks: “*Can you help me find a push-along tricycle from Radio Flyer that’s both fun and safe for my kid?*” This query blends a structured constraint (brand = “Radio Flyer”) with textual requirements (“fun and safe”). A medical researcher investigating a disease queries: “*What disease is associated with the PNPLA8 gene and presents with hypotonia as a symptom?*” This query combines structured relationships (gene-disease associations) with textual descriptions (symptom presentations). In both cases, users merge together requirements from heterogeneous knowledge in a single natural language query, because this is how they naturally express their information needs.

Why retrieval on semi-structured knowledge is critical for collaboration. This pattern is pervasive across domains where collaborative agents operate. In customer support, users describe their problem (textual) while referencing specific products or account details (structured). In academic search, researchers combine topic descriptions (textual) with author affiliations or citation relationships (structured). In travel planning, users specify destination preferences (textual) alongside budget constraints and date requirements (structured). Because users naturally blend these information types when collaborating

with agents, effective retrieval for collaboration requires reasoning across heterogeneous knowledge representations. These queries over heterogeneous knowledge create a fundamental retrieval challenge for collaborative agents: accurately finding relevant items requires reasoning across both textual and relational information simultaneously. To successfully retrieve information for users, the agent must: (1) understand the user’s query and identify what information types are needed, (2) search across both textual descriptions and structured relationships to avoid missing relevant items, (3) filter results by applying *all* user constraints precisely, and (4) return answers the user can verify and act upon. This retrieval process demands user-centered reasoning: understanding not just what the user said, but what information they actually need and how to find it.

In these user queries, knowledge must be represented in formats that capture both the richness of natural language and the precision of structured data. Semi-structured knowledge bases (SKBs) [114, 137, 166] provide this representation by integrating textual descriptions (*e.g.*, “fun and safe” features) with structured relationships (*e.g.*, brand affiliations, gene-disease associations). SKBs are ubiquitous in domains where collaborative agents operate: e-commerce platforms maintain product catalogs with descriptions and brand relationships [53], academic databases store papers with abstracts and citation networks [166], and medical knowledge bases combine disease descriptions with gene-protein associations [15, 61, 75]. Retrieval on SKBs is therefore essential for collaborative agents to serve users effectively in these real-world domains.

The gap: Retrieval benchmarks don’t evaluate reasoning over heterogeneous knowledge. Despite the importance of retrieval on SKBs for collaboration, current research has not adequately evaluated whether systems can perform user-centered retrieval across heterogeneous knowledge. Prior benchmarks focused on either purely textual queries over unstructured documents [40, 51, 68, 76, 78, 90, 187, 189], structured SQL queries over databases [198, 214], or graph queries over knowledge graphs [5, 8, 13, 46, 54, 153, 156, 195, 196]. Each benchmark evaluates retrieval on only one type of knowledge representation and cannot assess whether systems can handle the queries over heterogeneous knowledge that users naturally express during collaboration. Moreover, existing retrieval benchmarks focus on general knowledge from sources like Wikipedia, while collaborative agents

Table 3.1: Example queries from STARK showing how users naturally express multi-source information needs during collaboration. Users holistically combine **textual** descriptions (“durable,” “non-Markovian dynamics”) with **relational** constraints (“Dart World brand,” “by Hao-Sheng Zeng”) in a single query, requiring agents to perform user-centered reasoning across heterogeneous knowledge to retrieve accurate answers.

	Example query	Title of ground-truth items(s)
STARK-AMAZON	<i>Looking for durable Dart World brand dart flights that resist easy tearing. Any recommendations?</i>	<Amazon Standard Flights> <Dart World Broken Glass Flight> (+12 more)
	<i>What are recommended scuba diving weights for experienced divers that would fit well with my Gorilla PRO XL waterproof bag?</i>	<Sea Pearls Vinyl Coated Lace Thru Weight>
STARK-MAG	<i>Search publications by Hao-Sheng Zeng on non-Markovian dynamics.</i>	<Distribution of non-Markovian intervals...> <Comparison between non-Markovian...>
	<i>What are some nanofluid heat transfer research papers published by scholars from Philadelphia University?</i>	<A Numerical Study on Convection Around A Square Cylinder using AL2O3-H2O Nanofluid>
STARK-PRIME	<i>Could you provide a list of investigational drugs that interact with genes or proteins active in the epididymal region?</i>	<(S)-3-phenyllactic Acid>, <Anisomycin>, <Puromycin>
	<i>Search for diseases without known treatments and induce pruritus in pregnant women, potentially associated with Autoimmune.</i>	<Intrahepatic Cholestasis>
	<i>Please find pathways involving the POLR3D gene within nucleoplasm.</i>	<RNA Polymerase III Chain Elongation>
	<i>Which gene or protein associated with lichen amyloidosis can bind interleukin-31 to activate the PI3K/AKT and MAPK pathways?</i>	<OSMR>, <IL31RA>

typically operate on domain-specific knowledge bases (products, papers, medical information).

Collaborative agents are mostly powered by large language models (LLMs), which have shown promise on retrieval tasks [51, 92, 146, 219]. However, it remains unclear whether LLM-based agents can perform accurate retrieval on SKBs where queries require simultaneously searching across textual descriptions and structured relationships. This creates a critical gap: we lack benchmarks to evaluate whether current retrieval systems, including LLM-based collaborative agents, can handle queries over heterogeneous knowledge in the domain-specific settings where users collaborate with agents in real-world scenarios.

Challenge: Benchmarking retrieval on semi-structured knowledge. To enable progress on collaborative agents, we need retrieval benchmarks that evaluate whether systems can accurately find information when users express queries over heterogeneous knowledge on SKBs. Such benchmarks must assess whether retrieval systems can interpret user intent and return correct answers when queries combine constraints across textual and relational

knowledge (Table 3.1). However, building retrieval benchmarks for SKBs presents significant technical challenges.

First, user queries on SKBs exhibit complex interdependencies between textual and relational information. A query like “*investigational drugs that interact with genes active in the epididymal region*” requires understanding both the textual property (“investigational”) and the relational constraint (gene activity location), then finding entities that satisfy both simultaneously. This interdependence makes it difficult to construct precise ground-truth answers from millions of candidate entities.

Second, to evaluate user-centered reasoning, queries must resemble real-world scenarios where users interact with collaborative systems. Queries should be natural-sounding, reflect diverse user intents, and require context-specific reasoning, such as inferring implicit user interests or understanding domain-specific terminology. Generating such realistic queries at scale while ensuring quality and coverage is challenging.

Present work: STARK. To address this gap, we present STARK, a large-scale Semi-structured Textual and Relational Knowledge retrieval benchmark (Figure 3.1). STARK enables systematic evaluation of whether retrieval systems can accurately find information when users express queries over heterogeneous knowledge during collaboration with agents. By benchmarking retrieval on SKBs, we assess whether systems can interpret the holistic information needs users naturally express and return correct answers.

The key technical challenge we solve is constructing a large-scale retrieval benchmark with realistic user queries over heterogeneous knowledge on SKBs. We develop a novel pipeline that addresses the interdependence of textual and relational information, enabling us to generate natural queries and construct precise ground-truth answers from millions of candidate entities. Our pipeline produces queries that require retrieval systems to understand multiple constraints simultaneously, infer implicit user context, and reason about domain-specific relationships. We validate query quality through detailed analysis and human evaluation on naturalness, diversity, and practicality.

We build three SKBs from extensive texts and millions of entity relations from public sources, covering diverse domains where collaborative agents operate: e-commerce (product recommendation), academic research (paper search), and healthcare (precision medicine). Furthermore, we incorporate 274 human-generated queries to enrich the testing

scenarios, ensuring our benchmark reflects authentic user needs during collaboration.

Key features of STARK as a retrieval benchmark for collaborative agents:

- **Natural queries over heterogeneous knowledge (Table 3.1):** Queries incorporate both textual properties and relational constraints, mirroring how users naturally express information needs when collaborating with agents. Queries use flexible formats and often include additional context, such as “*I had a dozen 2.5-inch Brybelly air hockey pucks, so I’m trying to find matching strikers,*” reflecting how users provide background information during collaboration.
- **Retrieval requiring reasoning over heterogeneous knowledge:** To return correct answers, retrieval systems must simultaneously search across textual descriptions and structured relationships, then filter results using all user constraints. Queries require inferring implicit user interests (*e.g.*, complementary products), understanding specialized terminology (*e.g.*, domain-specific descriptions), and reasoning about multi-hop relationships. For example, retrieving complementary products requires understanding the user’s stated context and implicit purchase intent.
- **Large-scale domain-specific SKBs:** Our benchmark includes three knowledge bases* from domains where collaborative agents operate: e-commerce (product catalogs), academic research (paper databases), and healthcare (medical knowledge). Each SKB contains millions of entities and relations, providing realistic evaluation of retrieval at scale across diverse domains and user needs.

We conduct extensive experiments evaluating retrieval systems on STARK, including LLM-based approaches, revealing significant challenges in multi-source retrieval. Our experiments highlight performance gaps when retrieving from large-scale SKBs with millions of entities, and we analyze where current systems fail to accurately find information for user queries. Finally, we offer insights into building more capable retrieval systems for collaborative agents in real-world settings.

*Demos of SKBs at https://stark.stanford.edu/skb_explorer.html

3.2 Related Benchmarks

Understanding user queries is central to building effective collaborative agents. As discussed in the introduction, users naturally express complex information needs that reference multiple types of knowledge. Existing QA datasets have approached this challenge from different angles, each capturing some aspects of user queries but falling short of the full complexity users express during collaboration.

Benchmarks for user queries on unstructured knowledge. Early QA datasets focused on user queries that could be answered from unstructured text [40, 76, 85, 113, 135, 160, 174, 188]. While these benchmarks capture how users ask questions about textual content, they do not reflect user queries that reference structured attributes and relationships in addition to textual descriptions. STARK evaluates whether systems can interpret user queries that naturally combine both aspects, reflecting how users holistically express their needs during collaboration.

Benchmarks for user queries on structured knowledge. Another line of work focuses on user queries over structured knowledge, such as knowledge graphs [10, 11, 13, 46, 54, 156, 196] or databases [198, 214]. These benchmarks capture the structured aspects of user queries but restrict systems to predefined entities and relationships, ignoring the rich textual descriptions users naturally include when asking questions. STARK addresses this gap by evaluating user-centered reasoning on queries that combine structured constraints (*e.g.*, specific brands, author names) with textual requirements (*e.g.*, quality descriptions, topic interests), as users naturally express them.

Benchmarks for user queries on semi-structured knowledge. A smaller set of datasets attempts to capture user queries that span both textual and structured information [25, 26, 70, 123]. However, these datasets primarily use tables as structured frameworks, which limits the types of relationships they can represent compared to the rich entity networks users encounter in real collaborative settings (*e.g.*, e-commerce product catalogs, academic citation networks, biomedical knowledge graphs). STARK advances beyond these approaches by evaluating user-centered reasoning on semi-structured knowledge bases that integrate textual descriptions with flexible relational structures, reflecting the complexity of real user

queries during collaboration.

3.3 Formulation: Answering queries over heterogeneous knowledge

To evaluate collaborative agents on user-centered reasoning, we formalize retrieval on semi-structured knowledge bases (SKBs) that capture heterogeneous knowledge. A Semi-Structured Knowledge Base (SKB) consists of a knowledge graph G and a collection of free text documents D . Formally, let $G = (V, E)$ be the knowledge graph, where V is the set of nodes and $E \subseteq V \times V$ is the set of edges representing relationships between nodes. $D = \bigcup_{i \in V} D_i$ be the collection of free-form text documents associated with the nodes, where D_i is the set of documents associated with node i . For example, the product knowledge graph in e-commerce can capture relationships between products and brands/colors/categories, and the corresponding text documents include product descriptions, reviews, *etc.*

We define the retrieval tasks on our benchmark datasets as follows: Given the knowledge graph $G = (V, E)$, a collection of free text documents D , and a user query Q , the output is a set of nodes $A \subseteq V$ such that for each node $i \in A$, it satisfies the relational requirements imposed by the structure of G as specified in Q , and the associated documents D_i satisfy the textual requirements specified in Q . This formulation captures how users naturally express queries over heterogeneous knowledge during collaboration with agents.

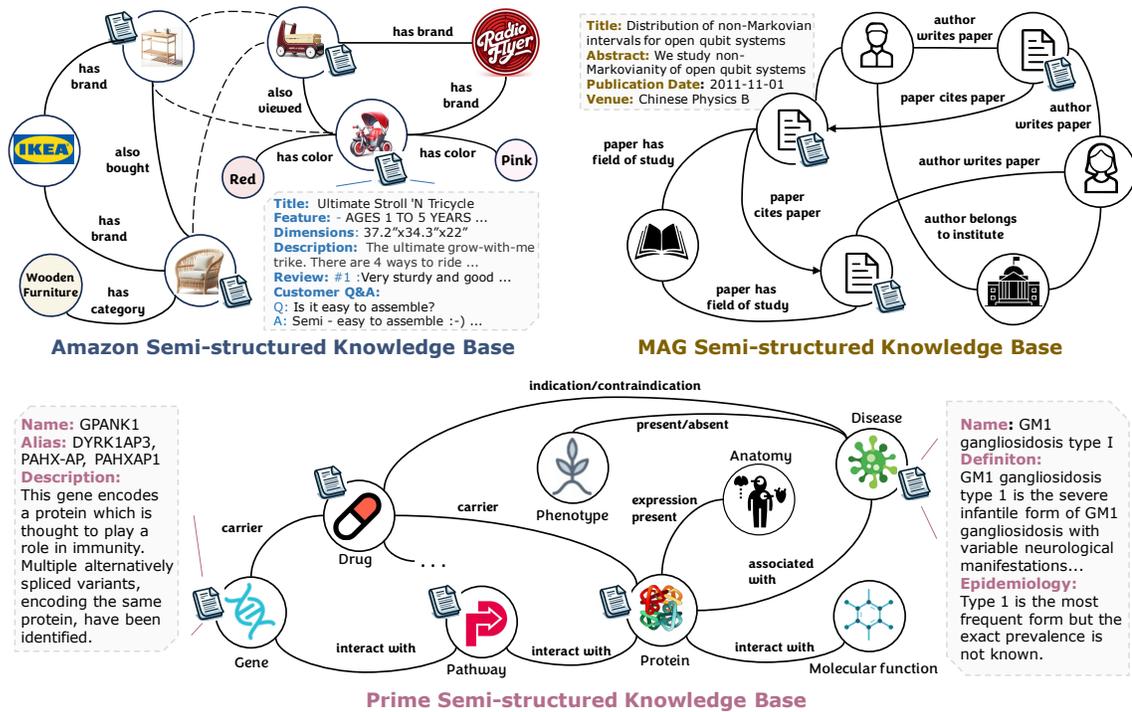


Figure 3.2: Demonstration of Semi-structured Knowledge Bases, where each knowledge base combines both textual and relational information in a complex way, making the retrieval tasks challenging.

3.4 STARK: A Benchmark for User-centered Reasoning over Semi-structured Knowledge

3.4.1 Semi-structured Knowledge Bases (SKBs)

To evaluate retrieval for collaborative agents in real-world domains, we construct three large-scale SKBs that represent heterogeneous knowledge where collaborative agents operate: e-commerce, academic research, and healthcare. In Figure 3.2, each SKB combines relational and textual information for entities, mirroring the knowledge representations agents must navigate when assisting users. See Table 3.2 for the data statistics.

Amazon Semi-structured Knowledge Base. The SKB features four entity types: product, brand, color, and category, and five relation types: also_bought, also_viewed

Table 3.2: Data statistics of our constructed semi-structured knowledge bases.

	#entity types	#relation types	avg. degree	#entities	#relations	#tokens
STARK-AMAZON	4	5	18.2	1,035,542	9,443,802	592,067,882
STARK-MAG	4	4	43.5	1,872,968	39,802,116	212,602,571
STARK-PRIME	10	18	125.2	129,375	8,100,498	31,844,769

between product entities, and `has_brand/color/category` associated with the products. We derive the textual information of product nodes by combining Amazon Product Reviews [53] with Amazon Q&A Data [109], providing rich texts including product descriptions and customer reviews. For other entities, we extract their names or titles as textual attributes. In total, Amazon SKB comprises around 1.0M entities (`product`: 0.9M, `brand`: 0.1M, `category`: 1.4k, `color`: 1.7k) and 9.4M relations (`also_bought`: 2.8M, `also_viewed`: 1.9M, `has_brand`: 1.7M, `has_category`: 2.3M, `has_color`: 0.6M), featuring extensive textual data from customer reviews and Q&A.

MAG Semi-structured Knowledge Base. This SKB includes node entities of `paper`, `author`, `institute`, and `field_of_study`. We derive its relational structure by extracting a subgraph from obgn-mag [61], which contains shared paper nodes with obgn-papers100M [?] and all non-paper nodes. We filter out non-English language papers as we only consider single-lingual queries. The paper documents include their titles and abstracts. Additionally, we integrate details from the Microsoft Academic Graph database (version 2019-03-22) [152, 166], providing extra textual information like paper venue, author and institution names. MAG SKB contains around 1.9M entities (`author`: 1.1M, `paper`: 0.7M, `institution`: 8.7K, `field`: 59.5k) and 39.8M relations (`author_writes`: 13.5M, `paper_has_field`: 14.5M, `cites`: 9.7M, `affiliated`: 2.0M), demonstrating a large number of relations associated with `paper` nodes, especially citation and authorship relations.

Prime Semi-structured Knowledge Base. We leverage the existing knowledge graph PrimeKG [15] which contains ten entity types including `disease`, `gene/protein`, and eighteen relation types, such as `associated_with`, `indication`. Compared to

the Amazon and MAG SKBs, Prime SKB is denser and features a greater variety of relation types. While PrimeKG provides text information on disease and drug entities, we integrate data from multiple databases for gene/protein and pathway entities such as genomic position, gene activity summary and pathway orthologous events. Prime SKB contains approximately 129.3K entities (disease: 17K, gene: 27.7K, mol_func: 11.2K, drug: 8K, pathway: 2.5K, anatomy: 14K, phenotype: 15.3K, bio_process: 28.6K, cell_comp: 4.2K, exposure: 0.8K) and around 8.1M relations.

3.4.2 User-centered Queries on Semi-structured Knowledge Bases

To evaluate whether retrieval systems can perform user-centered reasoning when users express queries over heterogeneous knowledge, our benchmark (Table 3.3) consists of three novel retrieval-based question-answering datasets, each comprising synthesized train/val/test sets with 9k to 14k queries in total and a high-quality human-generated query set. The queries blend relational and textual constraints as users naturally do during collaboration, with natural-sounding expressions and flexible formats that reflect real-world collaborative scenarios.

STARK-AMAZON. The task aims at product recommendation, with 68% of synthesized queries yielding multiple ground truth answers. The dataset prioritizes customer-oriented criteria, highlighting textual elements such as product quality, functionality, and style. It incorporates single-hop relational aspects using templates: `brand→product`, `product→product`, `color→product`, and `category→product`. These basic templates combine to form complex relational requirements, such as `color→product←brand` representing queries about products of a specific brand in a specific color.

STARK-MAG. Beyond the single-hop requirements in STARK-AMAZON, STARK-MAG targets academic paper search where users specify both textual and multi-hop relational constraints. We design seven relational query templates.

- **Single-hop templates** include (i) `author → paper` for “Can you list the papers authored by <author>?”, (ii) `paper → paper` for “Which papers have been cited by <paper>?”, and (iii) `field_of_study → paper` for “Can you provide papers in <field_of_study>?”

- **Multi-hop templates** include (i) $inst \rightarrow author \rightarrow paper$ for “Papers published by researchers from $\langle institution \rangle$?”, (ii) $paper \rightarrow author \rightarrow paper$ for “Papers published by coauthors of $\langle paper \rangle$?”, (iii) $paper \rightarrow author \rightarrow paper \leftarrow field$ for “Papers sharing a coauthor with $\langle paper \rangle$ in the same field”, and (iv) $inst \rightarrow author \rightarrow paper \leftarrow field$ for “Papers from $\langle institution \rangle$ in $\langle field \rangle$?”

The textual constraints focus on the paper’s topic, methodology, and contribution, *etc.*

STARK-PRIME. The task is to answer complex biomedicine inquiries. For synthesized queries, we developed 28 multi-hop query templates to cover various relation types and ensure their practical relevance. These templates range from 2-hop patterns like ($drug \rightarrow gene/protein \rightarrow drug$) for “What drug has common gene/protein carrier with $\langle drug \rangle$?” to complex 3-hop patterns like ($anatomy \rightarrow gene/protein \rightarrow drug$) for “What is the drug that targets genes or proteins expressed in $\langle anatomy \rangle$?” which aids precision medicine by identifying treatments targeted to specific anatomical areas. Templates also include negation patterns like ($effect/phenotype \rightarrow [phenotype\ absent] \rightarrow disease \leftarrow [!indication] \leftarrow drug$) for “Find diseases with zero indication drug and are associated with $\langle effect/phenotype \rangle$ ” and conjunction patterns that combine multiple relational paths. For *drug*, *disease*, *gene/protein*, and *pathway* entities, the queries are a hybrid of relational and textual requirements. For entities such as *effect/phenotype*, the queries rely solely on relational data due to limited textual information. We exhibit three distinct user roles (medical scientist, doctor, and patient) for generating queries about drug and disease, which diversify the language to comprehensively evaluate the retrieval systems.

3.4.3 Constructing Synthesized Queries

To generate realistic queries that mirror how users naturally express information needs over heterogeneous knowledge during collaboration, we present a novel pipeline (Figure 3.3) that synthesizes queries and automatically generates ground truth answers. The key challenge is that user queries naturally entangle relational and textual information, requiring our pipeline to capture this interdependence during synthesis and then disentangle them during answer filtering. The pipeline involves four steps as follows:

Table 3.3: Statistics on the STARK benchmark datasets.

		#queries	#queries w/ multiple answers	average #answers	train / val / test
Synthesized (Sec 3.4.3, 3.4.4)	STARK-AMAZON	9,100	7,082	17.99	0.65 / 0.17 / 0.18
	STARK-MAG	13,323	6,872	2.78	0.60 / 0.20 / 0.20
	STARK-PRIME	11,204	4,188	2.56	0.55 / 0.20 / 0.25
Human-generated (Sec 3.4.5)	STARK-AMAZON	81	64	19.50	For testing only
	STARK-MAG	84	34	3.26	
	STARK-PRIME	98	41	2.77	

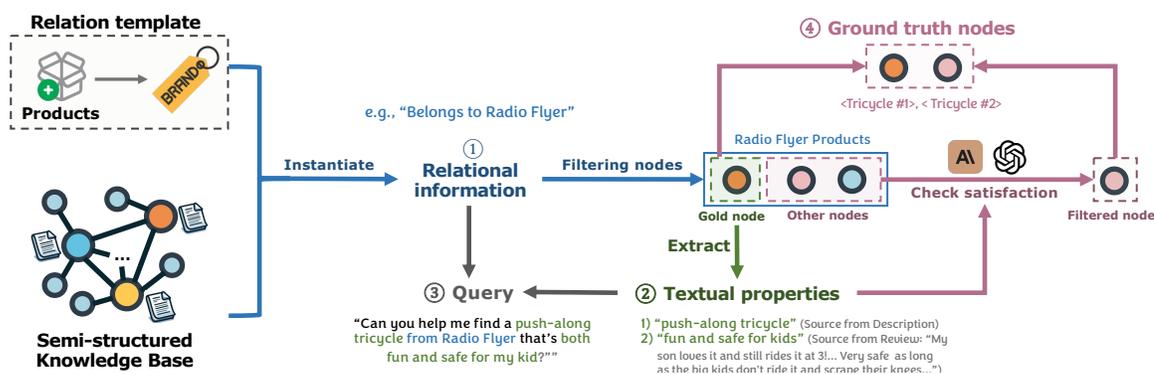


Figure 3.3: The construction pipeline to generate our semi-structured retrieval datasets.

- **1) Sample Relational Requirements:** For each query, we sample a practical relation template constructed with expert/domain knowledge, *e.g.*, “(a product) belongs to <brand>” and ground it with sampled entities (*i.e.*, a specific brand), *e.g.*, “belongs to Radio Flyer”. This relational requirement yields a set of candidate entities, *i.e.*, products belonging to Radio Flyer. We construct relational templates that reflect realistic user queries. For example, Amazon templates include “(brand → product)” for queries like “Can you list the products made by <brand>?” and “(product → product)” for “Which products are similar to <product>?” MAG templates include multi-hop patterns like “(institution → author → paper)” for “What papers have been published by researchers from <institution>?” Prime templates include complex medical queries like “(anatomy → gene/protein → drug)” for “What is

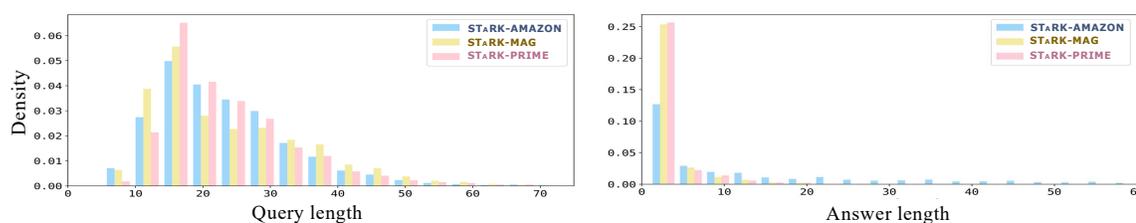


Figure 3.4: Distribution of query and answer lengths on STARK datasets.

the drug that targets the genes or proteins expressed in <anatomy>?” which aids precision medicine inquiries.

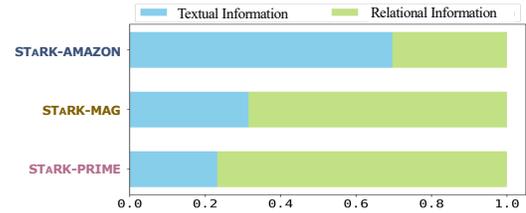
- **2) Extracting Textual Properties:** We randomly sample a candidate entity from the first step, referred to as the *gold answer*, from which LLMs extract properties that align with the interests of specific roles (*e.g.*, customers, researchers, or doctors) in its textual document. In Figure 3.3, we extract multiple properties about the functionality and user experience from a Radio Flyer product.
- **3) Combining Textual and Relational Information:** We use two LLMs to synthesize queries from textual properties and relational requirements, enhancing diversity and reducing bias arising from relying on a single LLM. The first LLM focuses on generating natural, role-specific, and style-consistent (*e.g.*, ArXiv searches) queries. The second LLM enriches the context and rephrases queries, which poses the need for advanced reasoning to comprehend them under complex contexts.
- **4) Filtering Additional Answers:** Finally, we employ multiple LLMs to verify if the candidates from the first step meet the extracted textual properties. Only candidates passing all LLM verifications are included in the final ground truth set. To assess the precision of this filtering mechanism, we compute the average ratios for the gold answers to be verified, which are 86.6%, 98.9%, and 92.3% on the three datasets, highlighting our efficacy in yielding high-quality ground truth answers.

This dataset construction pipeline is automatic, efficient, and broadly applicable to the SKBs in our formulation.

Figure 3.5: Query diversity measurement.

	Shannon Entropy	Type-Token Ratio
STARK-AMAZON	10.39	0.179
STARK-MAG	10.25	0.180
STARK-PRIME	9.63	0.143
Reference article	10.44	0.261

Figure 3.6: Average relative composition of relational vs. textual information.



3.4.4 Analysis: Distribution and human evaluation on synthesized data

To validate that our benchmark captures realistic user queries over heterogeneous knowledge, we analyze the synthesized query distributions and conduct human evaluation on query quality.

- Query and Answer Length.** Query length (in words) reflects the amount of user-provided context information, while the number of answers indicates query ambiguity/concreteness. Figure 3.4 shows similar query length distributions across the datasets, with most queries around 16 words. Longer queries (up to 50 words) often mention other entities or provide detailed context. Notably, STARK-AMAZON has a significant long-tail pattern, with about 22% of the answers have more than 30 entities, reflecting diverse e-commerce recommendations and ambiguous user queries.
- Query Diversity.** A diverse set of queries poses challenges for broader applicability to meet varying user demands. We measure query diversity using two metrics. Shannon Entropy [143] quantifies uncertainty in word distribution:

$$H(X) = - \sum_{i=1}^n p(x_i) \log p(x_i),$$

where $p(x_i)$ is the probability of word x_i . Type-Token Ratio (TTR) [159] measures lexical diversity: $TTR = V/N$, where V is unique words and N is total words. Higher values indicate greater lexical diversity. Table 3.5 shows high Shannon Entropy and steady TTR

Table 3.4: Positive/Non-negative rates (%) from human evaluation.

	Naturalness	Diversity	Practicality
STARK-AMAZON	73.6 / 89.5	68.4 / 89.5	89.5 / 94.7
STARK-MAG	94.7 / 100	73.7 / 84.2	68.4 / 84.2
STARK-PRIME	67.8 / 92.8	71.4 / 82.1	71.4 / 89.3
Average	78.7 / 94.1	71.0 / 85.3	76.4 / 89.4

across all datasets. For reference, we compute these metrics for the Wikipedia page of Barack Obama[†].

- **Proportionality of Relational vs. Textual Information.** Our benchmark queries feature the composition of textual and relational information. To understand the distribution of information types, we calculate the average ratio of relational to textual requirements by word count in the queries across each dataset. Note that the ratios do not directly reflect their importance in determining final answers. Figure 3.6 shows varying ratios, which highlights different emphases on textual versus relational requirements and challenges retrieval systems to adapt to different distributions.

Human evaluation. We qualitatively assess sampled queries from our benchmark for naturalness (resembling natural conversation), diversity (covering various question structures and complexities), and practicality (relevance to real-world situations) with 63 participants. Evaluation results, converted from a 5-point Likert-like scale to a positive/tie/negative scale, show positive and non-negative rates in Table 3.4. On average, 94.1%, 85.3%, and 89.4% of participants rated the queries neutral or above in naturalness, diversity, and practicality, respectively. These results validate the quality of our benchmark and its potential for diverse and realistic retrieval tasks.

[†]https://en.wikipedia.org/wiki/Barack_Obama

3.4.5 Constructing Human-Generated Queries

To ensure our benchmark reflects authentic user-centered reasoning during collaboration, we engaged 31 participants (22 native English speakers) to generate 263 queries across three SKBs, capturing how real users naturally express information needs over heterogeneous knowledge. Participants were given randomly sampled entity IDs and asked to explore the SKBs using our interactive platform at https://stark.stanford.edu/skb_explorer.html. They followed these instructions: (1) review the entity’s document and connected entities/multi-hop paths, (2) identify interesting aspects from both relational structures and textual information, (3) write queries such that the entity satisfies all requirements, without leaking the entity name. Participants were encouraged to be creative with query content and include additional context, with no restrictions on expression. We manually verified and filtered the ground truth answers to ensure correctness. Table 3.3 shows the statistics of the human-generated datasets. Finally, we analyzed the commonalities and differences between synthesized and human-generated queries.

Table 3.5: Comparison of human-generated and synthesized queries showing commonalities across query types.

Query Type	Human-generated Query	Synthesized Query
Short and Direct	<i>Red sweatshirt for proud Montreal Canadiens</i>	<i>Suggestions for a Suunto bike mount?</i>
Specific Author & Field	<i>Find me papers that discuss improving condenser performance authored by Stojan Hrnjak</i>	<i>Show me papers by Seung-Hyeok Kye that discuss separability criteria.</i>
Complex Context	<i>Help me. I am trying to diagnose a patient with persistent joint pain, and I suspect a condition where the bone is dying due to compromised blood supply, often linked to factors like steroid use, ... what’s the name of this sneaky bone-killing culprit?</i>	<i>I’m experiencing joint pain accompanied by swelling... I’m concerned about medications aggravating my fuzzy eyesight and potential blood clotting complications. Could you recommend treatments while minimizing these side effects?</i>

Commonality. The number of answers of synthesized and human-generated queries are comparable, indicating a similar level of query ambiguity. Moreover, we observe that most

styles of human-generated queries are covered in the synthesized dataset. For example, Table 3.5 highlights their similarities in short product queries, specific author/field inquiries, and complex contextual queries.

Difference. We find that human-generated queries often exhibit more unique expressions compared to synthesized ones, such as *”Give me a **fat cross** and road tire that works with my Diamondback bicycle tube”* and *”**this sneaky bone-killing culprit**”*. This discovery suggests a future direction for our benchmark to incorporate modern and dynamic language nuances.

3.5 Experimental Setup

To assess whether current retrieval systems can perform user-centered reasoning when users express queries over heterogeneous knowledge, we conduct extensive experiments evaluating various retrieval approaches on STARK. Our experiments reveal significant challenges in retrieval over heterogeneous knowledge, highlighting where current systems fail to accurately find information for user queries that collaborative agents must handle. We extensively evaluate six classes of retrieval models described below.

- **Sparse Retriever:** **BM25** [136] is a traditional yet powerful sparse retrieval method based on term frequency-inverse document frequency (TF-IDF). It computes relevance scores by considering the frequency of query terms in documents, adjusted for term rarity and document length.
- **Small Dense Retrievers:** **DPR** [78], **ANCE** [183], and **QAGNN** [193]. These compact models generate dense embeddings for both queries and documents, computing retrieval scores based on embedding similarities. They serve as baselines for comparison with LLM-based dense retrievers.
- **LLM-based Dense Retrievers:** **text-embedding-ada-002** (abbrev. **ada-002**) [115], **voyage-large-2-instruct** (abbrev. **voyage-l2-instruct**) [2], **LLM2Vec-Meta-Llama-3-8B-Instruct-mntp** (abbrev. **LLM2Vec**) [7], and **GritLM-7b** [111]. These models leverage LLMs to generate dense embeddings that are more contextually expressive.

- **Multivector Retrievers: multi-ada-002** [115] and **ColBERTv2** [138]. Beyond ada-002 which represents a document as an embedding, **multi-ada-002** splits each document into overlapping chunks and embeds them using the same encoder as the query. Similarity scores between the query and chunks are aggregated using the average of the top-3 similarities, which we found to perform best. **ColBERTv2** represents each document as multiple token-level embeddings for fine-grained matching, capturing richer semantic information.
- **LLM Rerankers: Claude3** and **GPT-4** rerankers [32, 220]. These models improve the precision of top- k ada-002 results by reranking them using large language models. We employ GPT-4-turbo (`gpt-4-1106-preview`) and Claude3 (`claude-3-opus`), setting $k = 20$ for synthesized queries and $k = 10$ for human-generated queries. Given a query, the LLMs assign a satisfaction score from 0 to 1 to each candidate entity based on textual and relational information. Due to high computational costs, we evaluate these rerankers on a random 10% sample of test queries.
- **Agent-Based Retrievers with Tools: ReAct** [191] and **Reflexion** [150]. These methods enable LLMs to interactively use tools, reason about observations, and perform actions. **ReAct** conducts reasoning and action in an in-context and interleaved manner to enable LLMs to interactively analyze observed information and perform actions. **Reflexion** uses self-reflection on the current task completion and stores these reflections in an episodic memory buffer to enhance decision-making in subsequent trials.

We implement both methods using Claude3 Opus (`claude-3-opus`) as the backbone LLM. Both agent-based retrievers have access to a function library of twenty-one tools, summarized in Table 3.6. These tools enable the agents to retrieve textual and relational information from the underlying SKBs.

The performance of these models are measured using standard retrieval metrics below.

- **Hit@ k** assesses whether the correct item is among the top- k results from the model. We used $k = 1$ and $k = 5$ for evaluation. At $k = 1$, it evaluates the accuracy of the top recommendation; at $k = 5$, it evaluates the precision in broader recommendation sets.

Function Name	Input	Description
Information Access Tools		
ParseAttributeFromQuery	query, attributes	Parse query into dictionary based on attributes
GetTextEmbedding	string (list)	Embed N strings into N tensors
GetRelevantChunk	query, node_id	Get relevant chunk of node info for query
GetFullInfo	node_id	Get full information of the node
GetEntityDocuments	node_id	Get text information of the node
Relational Structure Tools		
GetRelationInfo	node_id	Get relation information of the node
GetRelationDict	node_id	Get relation dict (keys: relation type, values: neighbors)
GetRelatedEntities	node_id	Get nodes related to the specified node
GetEntityIdsByType	type	Get IDs of nodes with specified type
GetEntityType	node_id	Get type of the node
GetEntityEmbedding	node_ids (list)	Get embedding indices of nodes
Semantic Matching Tools		
ComputeEmbeddingSimilarity	emb_1, emb_2	Compute cosine similarity of two embeddings
ComputeQueryEntitySimilarity	query, node_ids	Compute embedding similarity between query and nodes
ComputeExactMatchScore	string, node_ids	Compute exact match score for string in node info
TokenMatchScore	string, node_ids	Compute token recall scores between string and nodes
LLM-based Reasoning Tools		
SummarizeTexts	texts (list)	Use LLM to summarize texts
ClassifyEntities	node_ids, classes	Use LLM to classify nodes into classes
Classify	texts, classes	Use LLM to classify texts into classes
ExtractRelevantInfo	texts, extract_term	Use LLM to extract relevant info from texts
CheckRequirements	node_ids, requirement	Use LLM to check if nodes satisfy requirement
GetSatisfactionScore	node_ids, query	Use LLM to score nodes based on query
Utility Tools		
FINISH	final_answer_list	Indicate end of task with final answer

Table 3.6: Function library on STARK. Tools are grouped into four categories: information access, relational structure, semantic matching, and LLM-based reasoning.

- **Recall@ k** measures the proportion of relevant items in the top- k results. For synthesized queries, $k = 20$ is used, as the answer length of all of the queries in our benchmarks are equal or smaller than 20. This metric offers insight into the model’s ability to identify all relevant items, particularly in scenarios where missing any could be critical.
- **Mean Reciprocal Rank (MRR)** is a statistic for evaluating the average effectiveness of a predictive model. It calculates the reciprocal of the rank at which the first relevant item appears in the list of predictions. This metric emphasizes the importance of the rank of

Table 3.7: Testing results on STARK-Syn(thesized). Best results in bold.

	STARK-AMAZON				STARK-MAG				STARK-PRIME			
	Hit@1	Hit@5	R@20	MRR	Hit@1	Hit@5	R@20	MRR	Hit@1	Hit@5	R@20	MRR
Non-Agentic Retrieval Systems (Full Test Set)												
BM25	44.94	67.42	53.77	55.30	25.85	45.25	45.69	34.91	12.75	27.92	31.25	19.84
DPR (roberta)	15.29	47.93	44.49	30.20	10.51	35.23	42.11	21.34	4.46	21.85	30.13	12.38
ANCE (roberta)	30.96	51.06	41.95	40.66	21.96	36.50	35.32	29.14	6.53	15.67	16.52	11.05
QAGNN (roberta)	26.56	50.01	52.05	37.75	12.88	39.01	46.97	29.12	8.85	21.35	29.63	14.73
ada-002	39.16	62.73	53.29	50.35	29.08	49.61	48.36	38.62	12.63	31.49	36.00	21.41
voyage-l2-instruct	40.93	64.37	54.28	51.60	30.06	50.58	50.49	39.66	10.85	30.23	37.83	19.99
LLM2Vec	21.74	41.65	33.22	31.47	18.01	34.85	35.46	26.10	10.10	22.49	26.34	16.12
GritLM-7b	42.08	66.87	56.52	53.46	37.90	56.74	46.40	47.25	15.57	33.42	39.09	24.11
multi-ada-002	40.07	64.98	55.12	51.55	25.92	50.43	50.80	36.94	15.10	33.56	38.05	23.49
ColBERTv2	46.10	66.02	53.44	55.51	31.18	46.42	43.94	38.39	11.75	23.85	25.04	17.39
LLM Rerankers (Random 10% Sample)												
Claude3 Reranker	45.49	71.13	53.77	55.91	36.54	53.17	48.36	44.15	17.79	36.90	35.57	26.27
GPT4 Reranker	44.79	71.17	55.35	55.69	40.90	58.18	48.60	49.00	18.28	37.28	34.05	26.55
Agent-Based Retrieval Systems (Full Test Set)												
ReAct	42.14	64.56	50.81	52.30	31.07	49.49	47.03	39.25	15.28	31.95	33.63	22.76
Reflexion	42.79	65.05	54.70	52.91	40.71	54.44	49.55	47.06	14.28	34.99	38.52	24.82

the first correct answer, which is crucial in many practical applications where the first correct answer is often the most impactful.

3.6 Results: How do models perform on STARK

Table 3.7 presents comprehensive results on synthesized queries, and Table 3.8 shows results on human-generated queries from the STARK benchmark.

Performance varies significantly across model classes and domains. On synthesized queries, we observe substantial variation in performance. Among non-agentic methods evaluated on the full test set, ColBERTv2 achieves the highest Hit@1 on Amazon (46.10%), while GritLM-7b leads on MAG (37.90%) and Prime (15.57%). LLM rerankers, evaluated on a random 10% sample, achieve 40.90% Hit@1 on MAG (GPT-4 Reranker) and 18.28% on Prime (GPT-4 Reranker). Agent-based methods show competitive performance, with Reflexion reaching 42.79% Hit@1 on Amazon and 40.71% on MAG.

Performance patterns differ across domains. Amazon queries yield the highest Hit@1 scores (up to 46.10%), while Prime queries prove most challenging with peak Hit@1 of only 18.28%. The MAG dataset shows intermediate difficulty with best Hit@1 reaching

Table 3.8: Testing results on STARK-Human(-Generated). Best results in bold.

Method	STARK-AMAZON				STARK-MAG				STARK-PRIME			
	Hit@1	Hit@5	R@20	MRR	Hit@1	Hit@5	R@20	MRR	Hit@1	Hit@5	R@20	MRR
Non-Agentic Retrieval Systems												
BM25	27.16	51.85	29.23	18.79	32.14	41.67	32.46	37.42	22.45	41.84	42.32	30.37
DPR (roberta)	16.05	39.51	15.23	27.21	4.72	9.52	25.00	7.90	2.04	9.18	10.69	7.05
ANCE (roberta)	25.93	54.32	23.69	37.12	25.00	30.95	27.24	27.98	7.14	13.27	11.72	10.07
QAGNN (roberta)	22.22	49.38	21.54	31.33	20.24	26.19	28.76	25.53	6.12	13.27	17.62	9.39
ada-002	39.50	64.19	35.46	52.65	28.57	41.67	35.95	35.81	17.35	34.69	41.09	26.35
voyage-l2-instruct	35.80	62.96	33.01	47.84	22.62	36.90	32.44	29.68	16.33	32.65	39.01	24.33
LLM2Vec	29.63	46.91	21.21	38.61	16.67	28.57	21.74	21.59	9.18	21.43	26.77	15.24
GritLM-7b	40.74	71.60	36.30	53.21	34.52	44.04	34.57	38.72	25.51	41.84	48.10	34.28
multi-ada-002	46.91	72.84	40.22	58.74	23.81	41.67	39.85	31.43	24.49	39.80	47.21	32.98
ColBERTv2	33.33	55.56	29.03	43.77	33.33	36.90	30.50	35.97	15.31	26.53	25.56	19.67
LLM Rerankers												
Claude3 Reranker	53.09	74.07	35.46	62.11	38.10	45.24	35.95	42.00	28.57	46.94	41.61	36.32
GPT4 Reranker	50.62	75.31	35.46	61.06	36.90	46.43	35.95	40.65	28.57	44.90	41.61	34.82
Agent-Based Retrieval Systems												
ReAct	45.65	71.73	35.95	58.81	27.27	40.00	35.95	33.94	21.73	33.33	41.09	28.20
Reflexion	49.38	64.19	35.95	58.96	28.57	39.29	35.95	36.53	16.52	33.03	41.09	23.99

40.90%. Similar patterns appear in Recall@20 and MRR metrics.

Human-generated queries show consistent trends. On human-generated queries, performance is generally higher than on synthesized queries, but relative rankings among methods remain similar. LLM rerankers achieve the strongest performance: Claude3 Reranker reaches 53.09% Hit@1 on Amazon, 38.10% on MAG, and 28.57% on Prime. Among non-agentic embedding-based methods, multi-ada-002 performs best on Amazon (46.91% Hit@1), while GritLM-7b leads on MAG (34.52% Hit@1) and Prime (25.51% Hit@1). Agent-based methods show strong performance, with Reflexion achieving 49.38% Hit@1 on Amazon and 28.57% on MAG.

Despite higher absolute scores on human-generated queries, all methods still show substantial room for improvement, with no method exceeding 54% Hit@1 on any dataset.

To understand why current methods struggle on retrieval over heterogeneous knowledge, we organize our analysis into failure modes that examine the fundamental limitations of each model class.

3.6.1 Failure Analysis I: Dense and sparse retrieval models

This section analyzes the fundamental limitations of single-vector dense and sparse retrieval approaches when handling queries over heterogeneous knowledge.

Small dense retrievers struggle with heterogeneous knowledge. Among small dense retrievers, we observe that finetuned **DPR** and **QAGNN** exhibit insufficient performance despite being specifically trained on the datasets. This underperformance is likely due to their relatively small model sizes and the risk of overfitting during training. These models achieve only 15.29% Hit@1 (DPR on Amazon) and 12.88% Hit@1 (QAGNN on MAG), demonstrating that compact models struggle to capture the complex interdependencies between textual and relational information in SKBs. The entity documents can be hard to differentiate without capturing detailed information from both textual and relational aspects, a capability these smaller models lack.

LLM-based dense retrievers show improvement but remain limited. Among larger LLM-based dense retrievers, **ada-002** benefits from superior pretrained embeddings and significantly outperforms smaller models, achieving 39.16% Hit@1 on Amazon. **GritLM-7b** delivers the strongest performance among single-vector dense retrievers on the full test set, achieving 42.08% Hit@1 on Amazon, 37.90% on MAG, and 15.57% on Prime. **voyage-l2-instruct** also performs competitively, reaching 40.93% Hit@1 on Amazon. In contrast, **LLM2Vec** underperforms due to its limited context length, which is insufficient for encoding the lengthy documents in the SKBs that contain both extensive textual descriptions and relational information.

Despite improvements from using LLM-based embeddings, these single-vector dense retrievers face a fundamental limitation: they compress all textual and relational information into a single embedding. This compression makes it difficult to simultaneously capture fine-grained textual semantics and structured relationships, as the model must balance representing both types of information in a fixed-dimensional vector space.

Sparse retrieval remains competitive but fundamentally limited. Interestingly, **BM25**, despite its simplicity, proves to be a strong baseline, outperforming many dense retrieval models. On Amazon, BM25 achieves 44.94% Hit@1, even surpassing many sophisticated dense retrievers like **ANCE** (30.96% Hit@1) and matching some LLM-based retrievers.

This demonstrates that term-frequency approaches can capture some aspects of textual queries effectively, particularly when queries contain distinctive keywords that appear in relevant documents.

However, BM25 faces a fundamental limitation: it cannot reason about relational constraints in the knowledge graph. It treats all text equally, whether describing textual properties or listing relational connections, without understanding the semantic differences. This limits its ability to handle queries over heterogeneous knowledge that require understanding both textual descriptions and structural relationships simultaneously. For example, when a query asks for products from a specific brand with certain features, BM25 cannot distinguish between brand relationships and feature descriptions. It simply matches keywords without understanding their different roles in satisfying the user’s information need.

3.6.2 Failure Analysis II: Multivector retrievers

Multivector retrievers represent an advancement over single-vector approaches by using multiple embeddings to represent each document. This section analyzes **multi-ada-002** and **ColBERTv2**, which employ different strategies for multivector representation.

Multivector approaches show improvement over single-vector methods. **multi-ada-002** splits each document into overlapping chunks and embeds them separately, aggregating similarity scores using the average of the top-3 chunk similarities. This approach achieves 40.07% Hit@1 on Amazon and 25.92% on MAG, generally outperforming the single-vector ada-002 (39.16% on Amazon). The improvement demonstrates that using multiple vectors per document enhances retrieval effectiveness by allowing the model to capture different aspects of the document’s content separately.

ColBERTv2 takes a more fine-grained approach, representing each document with multiple token-level embeddings. This fine-grained representation allows ColBERTv2 to capture subtle semantic nuances between queries and documents, achieving strong performance with 46.10% Hit@1 on Amazon and 31.18% on MAG. On Amazon, ColBERTv2 achieves the highest Hit@1 among all non-agentic retrieval systems, demonstrating the value of token-level matching for certain types of queries.

Multivector retrievers still struggle with complex heterogeneous queries. Despite their

sophistication, multivector retrievers generally underperform compared to LLM rerankers, especially in terms of Hit@k metrics. For instance, on the random 10% sample, ColBERTv2’s best Hit@1 (46.10% on Amazon) is comparable to GPT-4 Reranker (44.79%), but it significantly underperforms on MAG (31.18% vs. 40.90%). This suggests that while fine-grained representations effectively capture semantic information, they may not fully grasp the nuanced relevance judgments required for top-tier retrieval performance when queries involve complex interdependencies between textual and relational information.

Fundamental limitation: Similarity-based matching without reasoning. The fundamental limitation of multivector retrievers is that, despite their sophistication, they still operate primarily through embedding similarity matching. They lack explicit mechanisms to reason about the logical structure of queries that combine textual constraints (*e.g.*, “fun and safe”) with relational constraints (*e.g.*, “from Brand X”). While token-level or chunk-level representations can capture more granular semantic information than single vectors, they cannot perform the multi-step reasoning required to understand how different constraints should be integrated. This limitation becomes particularly evident on the more complex MAG and Prime datasets, where queries often require multi-hop reasoning over knowledge graph structures. For example, a query might require first identifying papers from a specific institution, then filtering by research topic.

3.6.3 Failure Analysis III: Single-agent reranker

LLM rerankers improve but remain suboptimal. The rerankers, utilizing powerful LLMs like **GPT-4** (gpt-4-1106-preview) and **Claude3** (claude-3-opus), excel by re-evaluating the top candidates and assigning satisfaction scores based on a deeper understanding of the query and document content. This process allows them to better discern subtle contextual cues and relational information that dense retrievers might overlook. Consequently, LLM rerankers enhance retrieval precision at the top ranks, achieving the best performance among non-agentic methods with 40.90% Hit@1 on MAG and 18.28% on Prime.

However, regardless of the higher computational costs of the rerankers, their performance remains suboptimal. For instance, the Hit@1 scores for the GPT-4 reranker are only

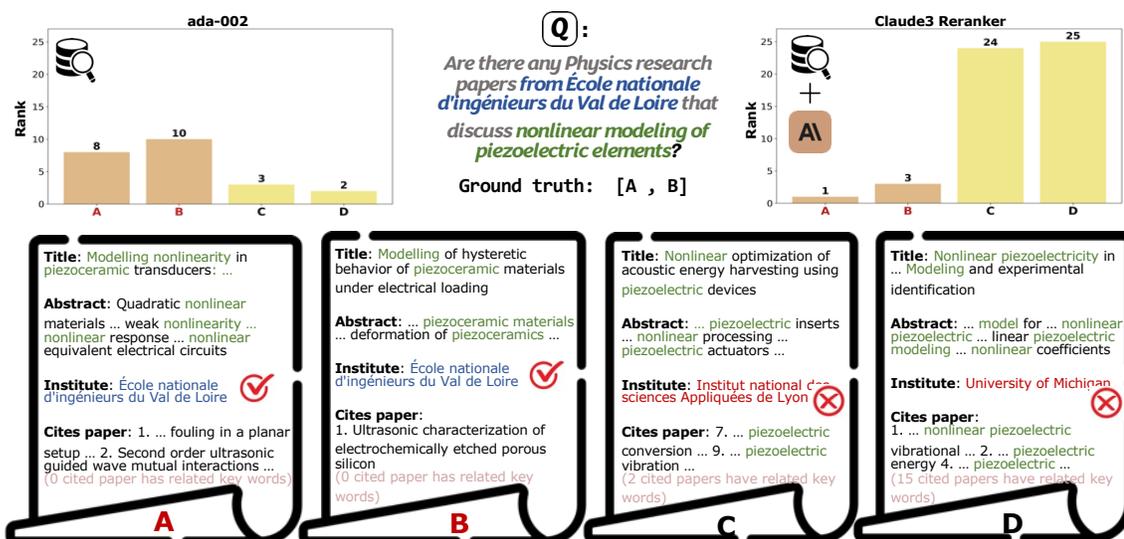


Figure 3.7: A case study on STARK-MAG shows that ada-002 overranks non-ground truth papers C and D due to repeated keywords in the relational information “cites paper”. After reranking with Claude3, it correctly prioritizes ground truth papers A and B with accurate reasoning and analysis.

about 18% on STARK-PRIME and 41% on STARK-MAG, indicating that the top-ranked answers are frequently incorrect. Similarly, the Recall@20 metrics are below 60% across all datasets, with the GPT-4 reranker achieving Recall@20 scores of 55% on STARK-AMAZON, 49% on STARK-MAG, and 34% on STARK-PRIME. This suggests that the ranking results miss a significant portion of relevant answers. The MRR scores are also relatively low, especially for STARK-PRIME, where the GPT-4 reranker attains an MRR of only around 27%.

Case study: The need for multi-step reasoning. To illustrate the importance of user-centered reasoning over heterogeneous knowledge, we present a case study in Figure 3.7, comparing the ada-002 model with the Claude3 Reranker. In this example, the user query requests papers from a specific institution on a particular topic, naturally combining relational constraints (institution affiliation) with textual requirements (topic). The ada-002 model fails to address the relational aspect of the query because it embeds entire documents without detailed analysis. This leads to high relevance scores for irrelevant papers that frequently mention keywords like “nonlinear modeling” and “piezoelectric elements” but do

not satisfy the relational requirement. In contrast, the LLM reranker significantly improves the results by reasoning about both the textual and relational aspects of the query, resulting in scores that more accurately reflect relevance. This underscores the need for multi-step reasoning to handle user queries that span heterogeneous knowledge.

Why rerankers still fail. The insufficient performance may be attributed to the fundamental challenge of reasoning over heterogeneous knowledge. User queries in SKBs require simultaneously understanding textual descriptions and relational constraints, demanding multi-step user-centered reasoning that current retrieval systems struggle to perform. Rerankers, while capable of better understanding than embedding-based methods, still operate in a single-shot manner. They evaluate each candidate independently without iteratively refining their understanding through interaction with the knowledge base. This limits their ability to perform complex multi-step reasoning required for queries like “What disease is associated with the PNPLA8 gene and presents with hypotonia as a symptom?” which requires first identifying diseases associated with the gene, then filtering by symptom descriptions.

3.6.4 Failure Analysis IV: Single-agent retriever with tools

Agent-based methods show promise but face limitations. Agent-based retrieval methods like **ReAct** and **Reflexion** represent a different paradigm, where LLMs can iteratively use tools to search, filter, and reason over the knowledge base. Table 3.7 shows that these methods achieve competitive performance, with Reflexion reaching 42.79% Hit@1 on Amazon, 40.71% on MAG, and 14.28% on Prime. This demonstrates that enabling iterative tool use and reasoning can improve retrieval over purely embedding-based approaches.

However, several key limitations prevent these methods from achieving strong performance:

(1) Limited tool selection strategies: ReAct agents cannot optimize their tool usage based on task-specific feedback. They tend to select tools based on the LLM’s prior knowledge, which may not be optimal for the given retrieval task. We observe that ReAct agents apply similar tools across various queries and struggle to explore alternative tool usage even with extensive in-context reasoning.

(2) **Inability to learn from experience:** While Reflexion attempts to use self-reflection to improve, it struggles with “overfitting” to specific examples. Reflexion agents are easily misled by specific entity characteristics, leading to inappropriate action sequences that don’t generalize well. For instance, an agent might develop a strategy that works for one type of product but fails on others.

(3) **Lack of systematic optimization:** Both methods rely on the LLM’s inherent reasoning capabilities without systematic optimization of the agent’s behavior for the specific retrieval task. They cannot effectively learn which combinations of tools work best for different query patterns or how to sequence operations to efficiently navigate the knowledge base.

ReAct: Single Agent

```

Input: Q1: "Can you suggest any TUSA swim fins that has a split fin design for better propulsion?", Action space (GetEntityTypes, GetEntityDocuments, StringMatching, ...)
Output: Answer(s)

Thought 1: Compute similarity scores based on the product description
Action 1: ComputeEmbeddingSimilarity["swim fins", GetEntityDocuments()]
Result/Obs 1: s1 ← similarity scores ✗ Ignore the brand information "TUSA"

Thought 2: Check the functionality requirement
Action 2: StringMatching["split fin design for better propulsion", GetEntityDocuments()]
Result/Obs 2: s2 ← string matching scores ✗ All zeros due to no perfect match

Thought 3: Synthesize the final result
Action 3: Multiply[s1, s2]
Result/Obs 3: s ← final scores ✗ All zeros which lead to a trivial solution

Final Result: answers ← GetTopkEntities[s, k=5] ✗ Poor task performance

```



Figure 3.8: A case study on STARK-AMAZON showing ReAct’s brittle tool usage. The agent ignores brand constraints, applies overly restrictive string matching, and returns zero results despite access to multiple search tools.

Case study: Brittle tool usage leads to trivial solutions. To illustrate these limitations concretely, we present a case study in Figure 3.8 showing how ReAct agents can produce trivial answers despite having access to multiple tools. In this example from STARK-AMAZON, the user asks for TUSA swim fins with a split fin design for better propulsion, naturally combining textual requirements (split fin design, better propulsion) with relational constraints (TUSA brand).

The ReAct agent’s failure demonstrates the brittle nature of hand-engineered prompting

strategies. In Thought 1 and Action 1, the agent computes embedding similarity based on product descriptions but completely ignores the brand constraint. This illustrates the limited tool selection strategy (limitation 1), where the agent applies tools based on generic patterns rather than carefully analyzing what the specific query requires. In Thought 2 and Action 2, the agent performs string matching on the exact phrase “split fin design for better propulsion,” which returns all zeros because no product description contains this precise wording. The agent fails to recognize that this approach is too restrictive and does not explore alternative strategies like matching individual terms or using semantic similarity for the textual constraints while applying exact matching for the brand.

Finally, in Thought 3, the agent synthesizes these misleading intermediate results. Rather than recognizing that the zero results indicate a problem with its approach, the agent produces zero final answers. This failure illustrates how single-agent retrievers with tools can easily produce trivial solutions when they lack systematic optimization (limitation 3) to learn which tool combinations work best for different query patterns.

These findings highlight significant gaps in current retrieval capabilities, revealing substantial challenges that must be addressed to enable collaborative agents to effectively serve users in real-world scenarios. The performance ceiling of single-model methods, whether embedding-based, reranking-based, or single agents, suggests that new approaches are needed to handle the complexity of reasoning over heterogeneous knowledge.

3.6.5 Retrieval Latency

Latency is crucial for practical retrieval systems, as users expect quick responses. In Table 3.9, we evaluated the latency of various models using a single NVIDIA A100-SXM4-80GB GPU. We observed that the DPR and QAGNN models exhibit lower average latency, making them suitable for time-sensitive applications. In contrast, the ada-002 and multi-ada-002 models have moderate latency due to multiple API calls. However, when combined with LLM rerankers, the latency increases significantly due to the computational demands of these large models. Therefore, it is important to balance accuracy and latency, especially for complex queries that require advanced reasoning capabilities.

Table 3.9: Latency (s) of the retrieval systems on STARK.

	DPR	QAGNN	ada-002	multi-ada-002	Claude3 Reranker	GPT4 Reranker
STARK-AMAZON	2.34	2.32	5.71	4.87	27.24	24.76
STARK-MAG	0.94	1.35	2.25	3.14	22.60	23.43
STARK-PRIME	0.92	1.29	0.54	0.90	29.14	26.97
Average	1.40	1.65	2.83	2.97	26.33	25.05

3.7 Conclusion

We introduce STARK, the first benchmark for evaluating agents on user-centered reasoning over semi-structured knowledge bases (SKBs), which are knowledge sources that combine unstructured text with structured relational data. STARK features natural queries that blend textual and relational constraints across three diverse domains: e-commerce product search, academic literature discovery, and biomedical knowledge retrieval. We contribute an automated pipeline to generate realistic user queries requiring reasoning across both textual descriptions and structured relationships, along with 274 human-generated queries capturing authentic user needs.

Our experiments evaluate six classes of retrieval methods and reveal fundamental challenges across all current paradigms. Consider a query like “find papers on machine learning written by Stanford researchers that cite work on neural networks.” Answering this requires identifying relevant topics from text, traversing authorship and citation relationships, and verifying that all constraints are satisfied. Embedding-based retrievers capture textual semantics but miss relational structure. Sparse retrievers match keywords but cannot traverse relationships. LLM rerankers reason effectively but operate in a single-shot manner without iterative refinement. Agent-based retrievers show promise through iterative tool use but cannot learn from their mistakes or systematically improve their strategies. The performance ceiling across all methods, with no approach exceeding 54% Hit@1, highlights substantial room for improvement and motivates multi-agent architectures where agents can collaborate to optimize query-solving strategies. STARK paves the way for such research by providing a rigorous testbed for developing and evaluating next-generation multi-agent systems.

Chapter 4

Learning from Experience through Contrastive Reasoning

4.1 Introduction

The previous chapter established comprehensive benchmarks for evaluating whether agents can learn from experience on retrieval tasks over semi-structured knowledge bases. Our experiments tested state-of-the-art agentic baselines including ReAct [191] and Reflexion [150], which maintain memory of past interactions. Despite having this memory, these systems revealed a consistent pattern: they fail to improve when exposed to similar queries multiple times, repeating the same mistakes across attempts. This raises a critical question: *what specific mechanisms do agents lack that prevents them from learning?*

To answer this, we analyze what happens when agents attempt the similar types of query repeatedly. Consider an agent tasked with finding products that match multiple user constraints. On the first attempt, it might miss a key constraint and return irrelevant results. On subsequent similar queries, the agent has access to the full history of what it tried before and what succeeded or failed. Yet it repeats the same oversight. The problem is not memory: the agent can see past attempts. Rather, the agent cannot extract the critical insight from this history. It cannot identify that “when queries mention brand names, I systematically ignore them and focus only on product categories,” nor can it generate the instruction “always extract brand constraints explicitly before searching.” This analysis reveals that learning

from experience requires two distinct capabilities that current agents fundamentally lack. First, agents need the ability to conduct contrastive analysis: comparing successful attempts against failed ones to identify systematic differences in their reasoning patterns. Second, agents need to translate these comparative insights into explicit, actionable instructions that guide future attempts. Existing optimization methods [27, 35, 170, 185, 192, 208, 218] typically focus on refining individual responses but do not provide mechanisms for this kind of systematic learning across multiple attempts.

Present work: AVATAR. To address these challenges, we introduce AVATAR, a framework that enables agents to learn from experience by recognizing their limitations and generating actionable instructions to address them. Our key insight is that agents can learn more robust strategies by contrasting their performance on tasks where they succeed versus tasks where they fail. We leverage this insight to build a system consisting of two collaborative agents: a comparator agent (“comparator”) and an actor agent (“actor”). The comparator agent analyzes performance patterns across multiple attempts and generates holistic instructions through contrastive reasoning, while the actor agent uses these instructions to improve its strategies. This collaboration, with contrastive reasoning at its core, enables agents to recognize systematic gaps in their current approaches and translate these insights into improved performance, allowing them to gradually learn how to solve harder tasks. We demonstrate our framework on the STARK benchmarks, an image retrieval task [130], and general question-answering tasks [94, 189, 221]. Specifically, AVATAR includes two phases:

- **Optimization phase.** The core of our framework (Figure 4.1) is the collaboration between the comparator agent and the actor agent for learning from experience. The comparator agent automatically generates holistic instructions to help the actor agent recognize its limitations and address them effectively. Through contrastive reasoning, the comparator agent takes positive and negative task samples—where the actor currently succeeds and fails, respectively—to identify systematic gaps in the actor’s strategies. Critically, this comparison reveals not just what failed, but why it failed: which assumptions were incorrect, which steps were missing, and which reasoning patterns led to errors. Unlike per-sample instructions, which can easily overfit to individual examples, by

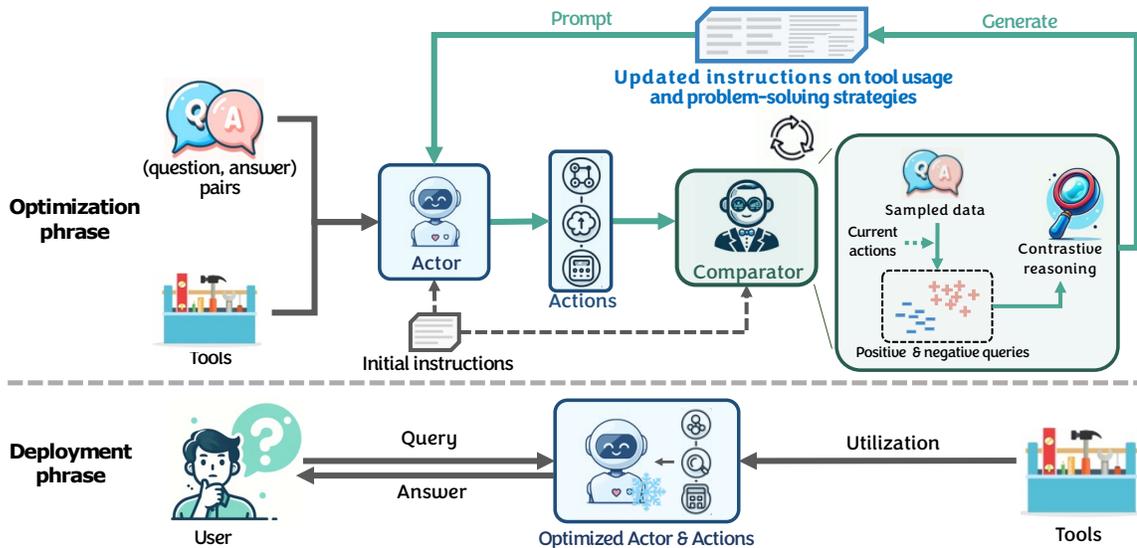


Figure 4.1: Overview of AVATAR. AVATAR is a multi-agent system consisting of an actor agent (actor) and a comparator agent (comparator). (a) During optimization, the actor agent generates actions to answer user queries by leveraging the provided tools. Then, the comparator agent contrasts queries where the actor performs well (positive) versus poorly (negative), automatically generating holistic instructions through contrastive reasoning to teach the actor more effective strategies for decomposing complex requests and coordinating tool usage (*cf.* Section 4.4). (b) At deployment, the actor agent with optimized instructions can effectively answer new user queries.

constructing multiple samples as a “batch,” the comparator agent can extract more robust insights about the actor’s systematic weaknesses. Through batch-wise contrastive reasoning, the system provides effective and adaptive instructions, helping the actor recognize recurring patterns in its failures and learn how to address them. Following previous methods [118, 150, 192, 215], we also maintain a memory bank with selected past instructions to prevent the actor agent from repeating previous mistakes.

- **Deployment phase.** After the optimization phase, the actor agent with best-performing instructions can be selected for new user queries. The iterative optimization through our AVATAR framework updates the actor agent for more effective and generalizable strategies, enabling it to apply learned lessons to novel tasks at deployment. In Figure 4.2 (b), the optimized actor agent demonstrates three key improvements from learning through

contrastive reasoning: 1) recognizing when initial assumptions are incorrect and adjusting its approach, 2) identifying which tool selection strategies work versus which fail and adapting accordingly, and 3) learning from past synthesis mistakes to produce more accurate results, enabling it to successfully address user queries where it previously failed.

Experimental evaluation. We evaluate AVATAR on the STARK datasets introduced in the previous chapter, an image retrieval benchmark [130], and three general question-answering datasets [94, 189, 221]. Our multi-agent system consistently outperforms state-of-the-art methods, showing a substantial 14% improvement in Hit@1 on retrieval tasks. With only 25 iterations of multi-agent collaboration, AVATAR boosts Hit@1 from 5.1% to 28.6% on FLICKR30K-ENTITIES and Recall@20 from 30.3% to 39.3% on STARK-PRIME. For general QA datasets, AVATAR outperforms state-of-the-art methods by 13% on average. Our key contributions are:

- We introduce AVATAR, a framework that enables agents to learn from experience, where a comparator agent and an actor agent collaborate through contrastive reasoning to help agents recognize their limitations and generate actionable instructions to address them.
- We demonstrate our framework on four retrieval tasks over semi-structured knowledge bases and three QA tasks, where agents using AVATAR learn to improve their performance over multiple attempts, significantly outperforming existing methods that fail to learn from experience.
- We provide comprehensive analysis of how contrastive reasoning systematically improves the actor’s ability to recognize what went wrong in failed attempts, understand why strategies failed, and translate these insights into improved performance on collaborative tasks.

4.2 Related Work

Agents for complex user queries. Recent research has leveraged the remarkable language understanding and reasoning abilities of LLMs [9, 150, 173, 190, 191] to build agents that

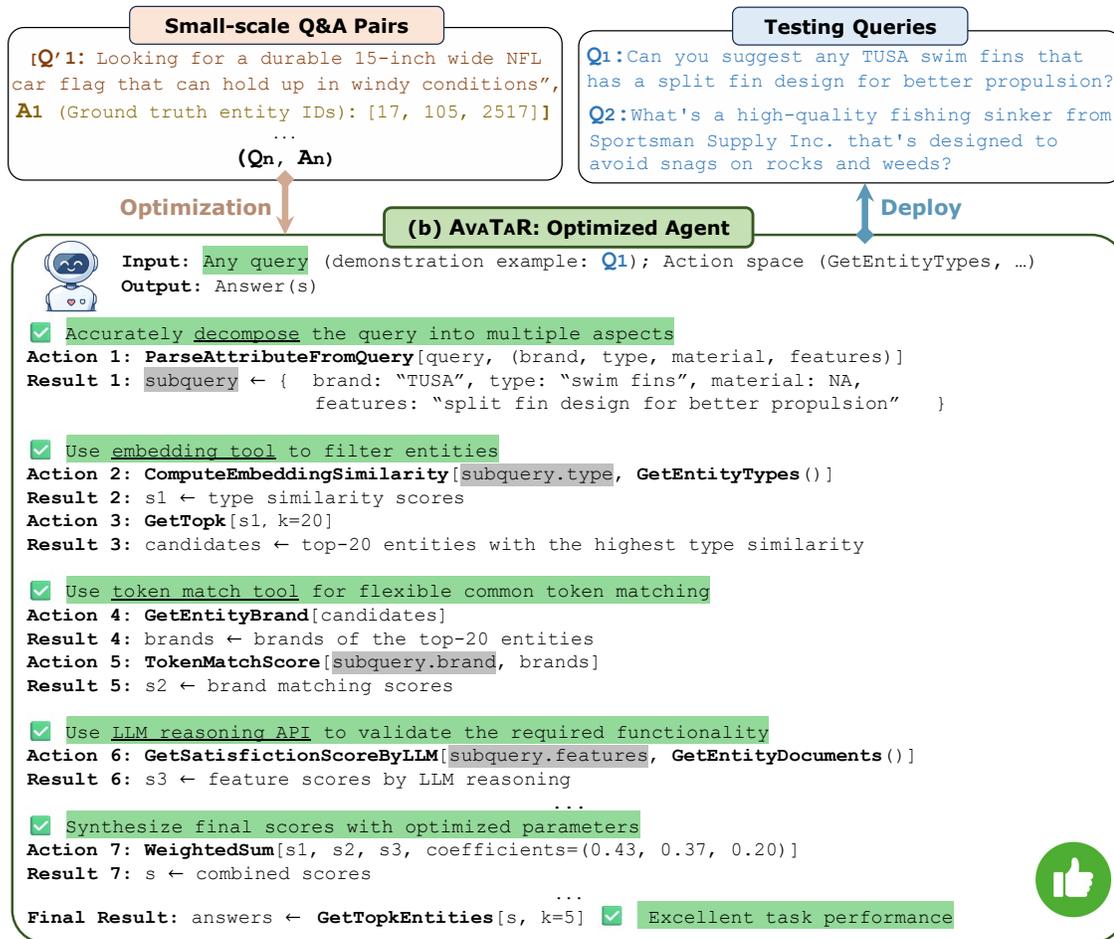


Figure 4.2: AVATAR decomposes the query into multiple interpretable constraints, implements strategic tool usage with flexible matching, and synthesizes results with learned parameters from the optimization phase to successfully answer the user’s query.

can interact with environments [27, 41, 64, 81, 93, 105, 107, 148, 177, 191] and leverage external tools [41, 112, 120, 124, 131, 139, 145, 221, 223]. For example, ReAct [191] conducts reasoning and action in an interleaved way to retrieve information from Wikipedia. However, as shown in the previous chapter, effectively addressing complex user queries requires agents to decompose multi-constraint requests, strategically coordinate tool usage, and synthesize information from heterogeneous sources, which remains challenging for current approaches.

Agents for retrieval over heterogeneous knowledge. Previous research has applied LLM agents to Information Retrieval (IR) systems through pretraining [12, 51, 67, 193], reranking [59, 154], and prompting techniques [54, 81]. These agents support downstream collaborative tasks, such as retrieval-augmented generation [92, 112, 118] and knowledge-intensive question answering [127, 189]. However, as our STARK evaluation demonstrated, these LLM agents usually employ heuristic prompts or rely on few-shot examples [81, 104, 148, 191], which lack systematic optimization for multi-step reasoning over heterogeneous constraints.

Agent optimization through learning. In the field of optimizing LLM agents, previous works have modified the parameters of LLM backbones through fine-tuning or instruction tuning to enhance agent capability [17, 65, 87, 95, 121, 124, 132, 157, 186, 194, 204] or generated better prompts through iterative prompt tuning [54, 81, 170, 185, 192]. Recently, Zhang et al. [208] conducted agent training by iteratively updating the agents’ functions according to the execution history, and ExpeL [209] extracts insights from successful and failed action sequences. However, these single-agent approaches do not explicitly focus on systematically improving agents’ ability to handle complex user queries that require decomposing multi-constraint requests and coordinating heterogeneous information sources. Additionally, enhancing agents’ generalization abilities [52, 120, 168] to novel user queries in real-world collaborative scenarios has received less attention. In our work, we introduce a multi-agent system where a comparator agent and an actor work together to solve complex user queries. We focus on automatically generating holistic instructions via a novel contrastive reasoning mechanism within this multi-agent framework that helps the actor agent learn robust strategies for solving complex user queries. Compared to fine-tuning approaches, our multi-agent system only requires a small subset of training queries and tool descriptions, making it more adaptable and less computationally intensive.

4.3 Formulation: Agent systems for complex queries

During human-AI collaboration, users naturally express complex queries that combine multiple constraints across diverse information sources. Consider a query like “find papers on graph neural networks published after 2020 by authors who have also worked on knowledge

graphs.” This query involves multiple constraints (topic, publication date, author expertise) that must be decomposed, verified against different information sources, and synthesized into a coherent answer. Building agent systems that can effectively handle such multi-step, multi-constraint queries requires careful formulation of the core components.

Tools. We define tools or APIs as a set of implemented functions with specified input and output variables. We denote the tool space as $\mathcal{T} = \{f_k : \mathcal{I}_{f_k} \rightarrow \mathcal{O}_{f_k} \mid k = 1, 2, \dots\}$, where f_k maps the input \mathcal{I}_{f_k} to the output \mathcal{O}_{f_k} . For example, tools can include search APIs for accessing external knowledge, encoder models that generate vector representations from text or image data, or classifiers that output probabilities over a list of classes. The diversity of available tools determines what types of constraints the agent system can address.

Agent Systems. An agent system consists of one or more LLM agents that coordinate to complete a task. Each agent, defined as $\mathcal{A} : \mathcal{P} \rightarrow \alpha$, is controlled by verbal prompts to generate a sequence of actions. Here α denotes the action sequence $[\alpha_1, \dots, \alpha_L]$, where each action is defined by a tuple $(f \in \mathcal{T}, i \in \mathcal{I}_f, o \in \mathcal{O}_f)$, consisting of a tool function, specified input(s), and a designated variable that receives the output(s). Each action can leverage outputs from previous actions, and agents within the system can pass intermediate results to one another. The final action α_L renders the results for the task.

Multi-step problem-solving. Real-world user queries are rarely simple lookups. Users often pose requests with multiple constraints that span different information sources, and key requirements may only become clear after several interaction steps. For example, a query like “find papers on graph neural networks published after 2020 by authors who have also worked on knowledge graphs” involves constraints on topic, publication date, and author expertise that must each be verified against different parts of the knowledge base. A single tool call or a straightforward retrieval step cannot address such queries; instead, the agent system must decompose the problem, execute multiple steps, and integrate findings into a coherent response. Solving these multi-constraint queries with agent systems can be structured into three stages:

- **Understanding and decomposition:** The agent system first interprets the user’s request and breaks it into subproblems, each targeting a specific constraint or information need. This decomposition clarifies what must be retrieved or verified at each step.

Table 4.1: Key differences between AVATAR and prevailing agent methods. AVATAR demonstrates the ability to: 1) self-improve on specific tasks, 2) retain memory throughout the optimization process, 3) enhance the agent’s generalization capability, and 4) autonomously generate holistic, high-quality prompts for better tool usage. Please refer to Section 4.4 for details.

	Self-Improvement	Memory	Generalization	Holistic Prompt Generation
ReAct [191]	✗	✗	✗	✗
Self-refine [107]	✓	✗	✗	✗
Reflexion [150]	✓	✓	✗	✗
AVATAR (Ours)	✓	✓	✓	✓

- **Tool-assisted execution:** The agent system then selects and invokes tools from the tool space \mathcal{T} to address each subproblem. This stage may involve querying different knowledge sources, filtering candidates against constraints, or performing computations on intermediate results.
- **Synthesis and response formulation:** Finally, the agent system integrates the intermediate results to construct a response. This synthesis may require reasoning about how different pieces of evidence relate, resolving conflicts, and organizing the output so the user can verify that their constraints have been satisfied.

Retrieval tasks exemplify this multi-step structure. Given a user query, the goal is to identify a ranked list of relevant entities from the entity space of a knowledge base, where ground truth answers are used to evaluate prediction quality. The agent system must comprehend the user’s request and its underlying constraints, use the provided tools to search and filter across a potentially large and multimodal knowledge space, and finally integrate all gathered information to produce an accurate and well-justified response.

4.4 AVATAR: Actor-Comparator System for User Queries

As described in Section 4.3, solving complex user queries requires agent systems to decompose multi-constraint requests, strategically coordinate tool usage, and synthesize results

from heterogeneous sources. This section describes how we build an agent system that can learn to handle such queries effectively.

We begin by defining the actor agent, which is responsible for generating action sequences to answer user queries (Section 4.4.1). However, as we discuss, a single actor agent faces fundamental challenges: it cannot easily recognize its own limitations or identify why certain strategies fail. This motivates augmenting the system with a comparator agent that provides an external perspective, analyzing where the actor succeeds and fails to generate holistic instructions for improvement (Section 4.4.2). Together, these two agents form the AVATAR multi-agent system. In Table 4.1, we highlight four critical aspects of our approach compared with prevailing agent frameworks [107, 150, 191].

4.4.1 Actor: The single agent and its limitations

Actor definition. The actor agent is responsible for answering complex user queries by generating action sequences. Given initial instructions that describe the task and available tools, the actor produces a sequence of actions, each invoking a tool from the tool space \mathcal{T} , to decompose the user’s request, gather relevant information, and synthesize a response. These actions can be expressed as code (e.g., Python function calls) along with natural language explanations of how the tools are employed.

Limitations of existing approaches. Prevailing agent frameworks like ReAct [191] rely on hand-engineered prompts that work for simple cases but exhibit brittle behavior on complex user queries. As shown in Table 4.1, ReAct lacks the ability to self-improve, retain memory, or generalize to new query patterns.

Methods like Self-refine [107] and Reflexion [150] attempt to address this by having the agent reflect on its own performance. However, these self-reflection approaches face a fundamental limitation: **a single agent cannot easily recognize its own systematic gaps**. When an agent fails on a complex user query, the failure may stem from multiple interacting factors: incomplete decomposition of constraints, inappropriate tool selection, or flawed synthesis of results. Self-reflection on individual failures tends to produce narrow fixes that address surface-level symptoms rather than underlying patterns.

The core challenge. Complex user queries require the agent to coordinate multiple interacting components: how it decomposes multi-constraint requests, which tools it selects for each constraint, and how it synthesizes intermediate results. Instructions generated from a single failed query may identify one specific issue but miss systematic gaps that manifest across different query patterns. Moreover, strategies that work well for one type of query may fail on others, leading to poor generalization. Without an external perspective to analyze performance across multiple queries, the actor cannot learn robust strategies for fulfilling diverse user requests.

4.4.2 Comparator: Enable learning from contrastive reasoning

The limitations described above motivate augmenting the actor with an external agent that can provide the perspective it lacks. We introduce the comparator agent, which analyzes the actor’s performance across multiple queries to identify systematic gaps and generate holistic instructions for improvement.

Key insight. Rather than relying on the actor to reflect on individual failures, the comparator takes a fundamentally different approach: it contrasts queries where the actor succeeds against queries where it fails. By examining what distinguishes successful from unsuccessful cases, the comparator can identify patterns that the actor itself cannot see, for example, that failures consistently involve queries with subtle brand mentions, or that the actor tends to miss implicit constraints in longer requests. This contrastive reasoning enables the comparator to generate instructions that address root causes rather than surface symptoms.

Contrastive reasoning process. The comparator’s analysis proceeds in two steps:

Step 1: Constructing positive and negative query sets. As shown in Figure 4.1, the comparator agent samples a set of user queries and evaluates the actor agent’s current action sequence on each. Based on performance, queries are categorized into positive (well-performing) and negative (poorly-performing) groups. Specifically, we define two thresholds, ℓ and h (where $0 < h \leq \ell < 1$): queries with evaluation metric above ℓ are classified as positive, while those below h are classified as negative. We then sample a mini-batch of b queries with an equal split ($b/2$ each) for contrastive analysis.

Step 2: Generating holistic instructions. The comparator contrasts the two groups to

identify what distinguishes successful from unsuccessful cases. This analysis reveals systematic gaps in how the actor handles complex queries, including patterns in decomposition failures, tool selection errors, or synthesis problems that recur across negative examples. The comparator then generates instructions that address these gaps, providing guidance on how to improve decomposition strategies, which tools to prefer for specific constraint types, and how to better synthesize results. These instructions are appended to the actor's prompts for the next iteration.

Why batch-wise contrastive reasoning works. We draw an analogy to deep neural network training, where extremely small batch sizes introduce high variance in gradient estimates. Similarly, instructions generated from a single failed query tend to be noisy and narrow. By constructing batches of positive and negative queries, the comparator can extract a more robust “gradient,” identifying patterns that consistently distinguish success from failure rather than idiosyncratic features of individual examples.

This batch-wise approach offers three advantages:

- It encourages comprehensive instructions that address all three stages of query-solving (decomposition, tool coordination, synthesis) rather than focusing on one surface-level fix.
- Contrastive reasoning directly targets the relationship between query patterns and tool effectiveness, helping identify which strategies work for which types of constraints.
- By analyzing failures across diverse queries, the comparator generates instructions that generalize to novel cases rather than overfitting to specific examples.

Demonstration example. Figure 4.3 illustrates the comparator's analysis in action. When contrasting positive and negative queries, the comparator observes that failures consistently involve queries with more complex product descriptions, more subtle brand mentions, and additional relevant product requirements. From these patterns, it identifies three systematic gaps in the actor's approach: (1) incomplete decomposition that misses detailed product features, (2) inappropriate tool selection that uses embedding similarity for brand matching when more precise verification is needed, and (3) insufficient synthesis that fails to incorporate related products. The comparator then generates instructions addressing each gap,

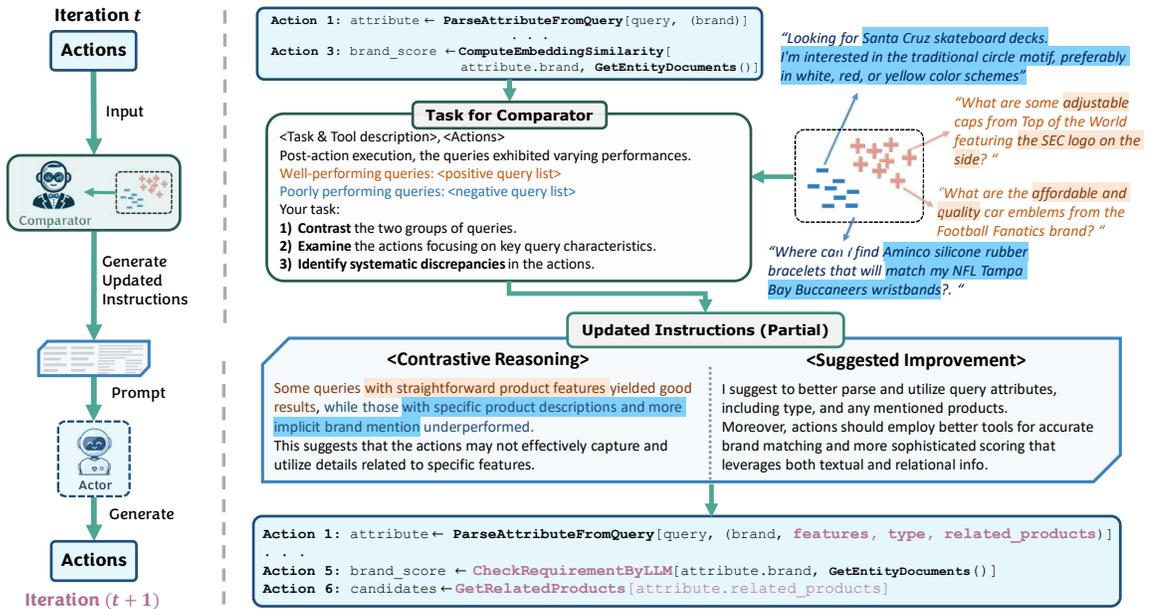


Figure 4.3: Demonstration example during optimization. Best viewed in color. The comparator contrasts positive and negative queries to identify systematic gaps, then generates holistic instructions that help the actor improve query decomposition, tool coordination, and result synthesis.

for example, recommending that the actor replace the embedding tool with an LLM verification tool for precise constraint matching. These instructions help the actor learn robust strategies that transfer to new queries with similar characteristics.

4.4.3 AVATAR System: Optimization and deployment

Having defined the actor and comparator agents, we now describe how they work together in the complete AVATAR system.

Iterative optimization. The optimization proceeds iteratively. In each iteration, the comparator samples queries, evaluates the actor’s current performance, constructs positive and negative sets, and generates instructions through contrastive reasoning. These instructions are appended to the actor’s prompts, and the actor generates revised action sequences. This cycle repeats, with each iteration refining the actor’s strategies based on the comparator’s analysis.

Logistic instructions. In addition to the comparator’s strategic instructions, we incorporate two types of logistic instructions to ensure actions are valid and efficient:

- **Validity check:** Triggered during action execution to verify correct function calls and valid tool usage.
- **Timeout error:** A mechanism that triggers an error if processing exceeds a threshold, prompting the actor to adopt more efficient strategies.

Memory bank. To prevent the actor from repeating previous mistakes, we maintain a memory bank following Shinn et al. [150]. The memory stores tuples of action sequences, comparator instructions, and performance metrics. To manage context size, we retain only the top-5 action sequences with the best performance. This enables the actor to learn from both immediate instructions and historical results.

Deployment. After optimization, the actor with best-performing instructions can be deployed to handle new user queries. As shown in Figure 4.1, the optimized actor uses the learned strategies for decomposing multi-constraint requests, coordinating tool usage, and synthesizing results. Because these strategies were derived from contrastive reasoning across diverse queries, they generalize effectively to novel user requests.

4.5 Experiments: Evaluating agent systems

Tasks and Evaluation. We conduct experiments to evaluate how effectively AVATAR helps agents solve complex user queries across diverse collaborative scenarios:

- **STARK [182]:** The semi-structured knowledge base benchmark introduced in Chapter 3. The function library for this task is shown in Table 4.3.
- **FLICKR30K-ENTITIES [130]:** An image retrieval dataset where user queries combine textual descriptions with visual references. Users express requests like “a person in a red shirt near a bicycle,” requiring agents to coordinate phrase matching across annotated bounding boxes and descriptive text. We assess task performance using Hit@1, Hit@5,

Function Name	Input	Description
Information Access Tools		
ParseAttributeFromQuery	query, attributes	Parse query into dictionary based on attributes
GetBagOfPhrases	image_ids (list)	Get list of phrases for each image
GetEntityDocuments	image_ids (list)	Get text information for each image
GetClipTextEmbedding	string (list)	Embed strings into embeddings
GetPatchIdToPhraseDict	image_ids (list)	Get patch_id to phrase dict for each image
Visual Information Tools		
GetImages	image_id_lst (list)	Return images with corresponding ids
GetClipImageEmbedding	image_lst (list)	Embed images into tensors
GetImagePatchByPhraseId	image_id, patch_id	Return patch image for given image and patch id
Semantic Matching Tools		
ComputeEmbeddingSimilarity	emb_1, emb_2	Compute cosine similarity of two embeddings
ComputeF1	string_to_match, strings	Compute F1 score between string and each target
TokenMatchScore	string_to_match, strings	Compute recall score between string and each target
ComputeExactMatchScore	string_to_match, strings	Compute exact match score for each target
LLM-based Reasoning Tools		
Vqa	question, image_lst	Use LLM to answer question based on images
ExtractVisualAttributes	attribute_lst, image_lst	Use LLM to extract attributes from images
Utility Tools		
FINISH	final_answer_list	Indicate end of task with final answer

Table 4.2: **Function library on FLICKR30K-ENTITIES.** Tools are grouped into five categories: information access, visual information, semantic matching, LLM-based reasoning, and utility.

Recall@20, and Mean Reciprocal Rank (MRR) metrics. The function library for this task is shown in Table 4.2.

- **Question-answering benchmarks: HotpotQA [189], ArxivQA [94], ToolQA [221],** where users ask complex questions requiring multi-hop reasoning and information synthesis. We sample 100, 100, and 40 training queries, and 100, 100, and 60 testing queries for the three benchmarks, respectively. We use exact match (EM) score on HotpotQA. For ArxivQA and ToolQA, we use the LLM judge score for more reliable evaluation.

Baselines. We compare AVATAR with several agent frameworks. As discussed in Section 4.4.1, **ReAct** [191] and **Reflexion** [150] represent prevailing approaches that either lack self-improvement capabilities or rely on single-instance reflection. We additionally compare against:

Function Name	Input	Description
Information Access Tools		
ParseAttributeFromQuery	query, attributes	Parse query into dictionary based on attributes
GetTextEmbedding	string (list)	Embed strings into embeddings
GetRelevantChunk	query, node_id	Get the relevant chunk of information for the node based on the query
GetFullInfo	node_id	Get the full information of the node
GetEntityDocuments	node_id	Get the text information of the node
Relational Information Tools		
GetRelationInfo	node_id	Get the relation information of the node
GetRelationDict	node_id	Get the relation dictionary for the node
GetRelatedEntities	node_id	Get the nodes related to the specified node
GetEntityIdsByType	type	Get the IDs of nodes with the specified type
GetEntityType	node_id	Get the type of the node
GetEntityEmbedding	node_ids (list)	Get the embedding indices of nodes
Semantic Matching Tools		
ComputeEmbeddingSimilarity	emb_1, emb_2	Compute cosine similarity of two embeddings
ComputeQueryEntitySimilarity	query, node_ids	Compute embedding similarity between query and nodes
ComputeExactMatchScore	string, node_ids	Compute exact match score for each node
TokenMatchScore	string, node_ids	Compute recall score between string and node info
LLM-based Reasoning Tools		
SummarizeTextsByLLM	texts (list)	Use LLM to summarize the provided texts
ClassifyEntitiesByLLM	node_ids, classes	Use LLM to classify nodes into classes
ClassifyByLLM	texts, classes	Use LLM to classify texts into classes
ExtractRelevantInfoByLLM	texts, extract_term	Use LLM to extract relevant information from texts
CheckRequirementsByLLM	node_ids, requirement	Use LLM to check if nodes satisfy requirement
GetSatisfactionScoreByLLM	node_ids, query	Use LLM to score nodes based on query
Utility Tools		
FINISH	final_answer_list	Indicate end of task with final answer

Table 4.3: **Function library on STARK.** Tools are grouped into five categories: information access, relational information, semantic matching, LLM-based reasoning, and utility.

- **ExpeL** [209] extracts insights from successful and failed action sequences, retrieving and including them in the context during inference. We apply ExpeL on the QA datasets.
- **Retroformer** [192] reinforces LLM agents and automatically tunes their prompts by learning a retrospective model through policy gradient. We compare the performance of AVATAR with the reported result by Retroformer on HotpotQA.

We include AVATAR-C, which removes the Comparator from our optimization pipeline. This comparison aims to validate the effectiveness of the Comparator.

LLM backbones. For the knowledge retrieval tasks, we use `claude-3-opus` as the backbone LLM. For the QA tasks, we use `gpt-4` for HotpotQA for fair comparison with previous methods and `gpt-4o` for ArxivQA and ToolQA.

Function library. For STARK, we use the function library described in Table 4.3. For the image retrieval tasks, our function library provides access to visual and textual information, including tools for retrieving annotated phrases, computing embedding similarities, and evaluating visual attributes (Table 4.2). For the QA tasks, we provide web search tools: `WEB_SEARCH` for general-purpose web queries, `ARXIV_SEARCH` for retrieving academic paper metadata, `WIKI_SEARCH` for Wikipedia lookups, and `RETRIEVE_FROM_DB` for database retrieval (ToolQA only). We used the same function library across all agent methods.

General pipeline. For AVATAR, we optimize the agent for a fixed number of epochs and select the instruction with the highest performance. We use the same initial prompt structure, the metric Recall@20 or Accuracy for constructing positive and negative queries, and hyperparameters ($\ell = h = 0.5$, $b = 20$) for all datasets.

4.5.1 STaRK: Semi-structured knowledge base retrieval

We evaluate AVATAR on the STARK benchmark introduced in Chapter 3, which requires agents to retrieve entities from semi-structured knowledge bases combining relational graph structure with free-text documents. User queries in STARK involve multiple heterogeneous constraints that must be verified against different parts of the knowledge base.

Takeaway 1: Multi-agent system outperforms baselines on semi-structured retrieval. Table 4.4 shows that AVATAR consistently outperforms both embedding-based methods and agent baselines across all three STARK datasets. On AMAZON, AVATAR achieves 49.87% Hit@1, a 16.6% relative improvement over Reflexion. On MAG, which involves academic paper retrieval with complex relational constraints, AVATAR improves Hit@1 by 9.0% over Reflexion. The gains are particularly notable on PRIME, where AVATAR

Table 4.4: Retrieval performance (%) on STARK benchmark. AVATAR achieves up to 16.5% relative improvement in Hit@1 over the best baseline. We compare AVATAR against embedding-based retrieval (multi-ada-002) and agent baselines. Results for other non-agentic baselines are reported in Chapter 3.

	AMAZON				MAG				PRIME			
	Hit@1	Hit@5	R@20	MRR	Hit@1	Hit@5	R@20	MRR	Hit@1	Hit@5	R@20	MRR
multi-ada-002	40.07	64.98	<u>55.12</u>	51.55	25.92	50.43	50.80	36.94	15.10	33.56	38.05	23.49
ReAct	42.14	64.56	50.81	52.30	31.07	49.49	47.03	39.25	<u>15.28</u>	31.95	33.63	22.76
Reflexion	<u>42.79</u>	<u>65.05</u>	54.70	<u>52.91</u>	<u>40.71</u>	<u>54.44</u>	<u>49.55</u>	<u>47.06</u>	14.28	<u>34.99</u>	<u>38.52</u>	<u>24.82</u>
AVATAR-C	40.92	63.63	53.68	51.73	33.25	52.17	47.88	41.34	8.82	23.82	30.32	16.20
AVATAR	49.87	69.16	60.57	58.70	44.36	59.66	50.63	51.15	18.44	36.73	39.31	26.73
Relative Improvement	16.6%	6.3%	9.9%	12.2%	9.0%	9.6%	-0.3%	8.7%	20.7%	5.0%	2.1%	7.7%

achieves 18.44% Hit@1 compared to 15.28% for ReAct and 14.28% for Reflexion, demonstrating that multi-agent contrastive reasoning is especially effective when queries require coordinating constraints across diverse information types.

Takeaway 2: The Comparator enables systematic improvement through contrastive analysis. The comparison between AVATAR and AVATAR-C (which removes the Comparator) reveals the critical role of multi-agent collaboration. Without the Comparator’s contrastive reasoning, AVATAR-C performs worse than even single-agent baselines on PRIME (8.82% vs 15.28% Hit@1 for ReAct). This demonstrates that validity checks and timeout mechanisms alone are insufficient; the Comparator’s ability to identify systematic gaps across queries is essential for learning robust strategies.

Takeaway 3: Multi-agent contrastive reasoning systematically improves query handling. Figure 4.4 presents concrete cases illustrating how the Comparator helps the Actor learn to solve complex user queries. By contrasting queries where the Actor succeeds versus fails, the Comparator identifies three types of systematic gaps: (1) information omission where the Actor misses key constraints from user requests, (2) ineffective tool usage where the Actor applies inappropriate tools for specific constraint types, and (3) suboptimal synthesis where the Actor fails to properly weight results from different sources. The instructions generated by the Comparator address these gaps, leading to more robust strategies that generalize to novel user queries.

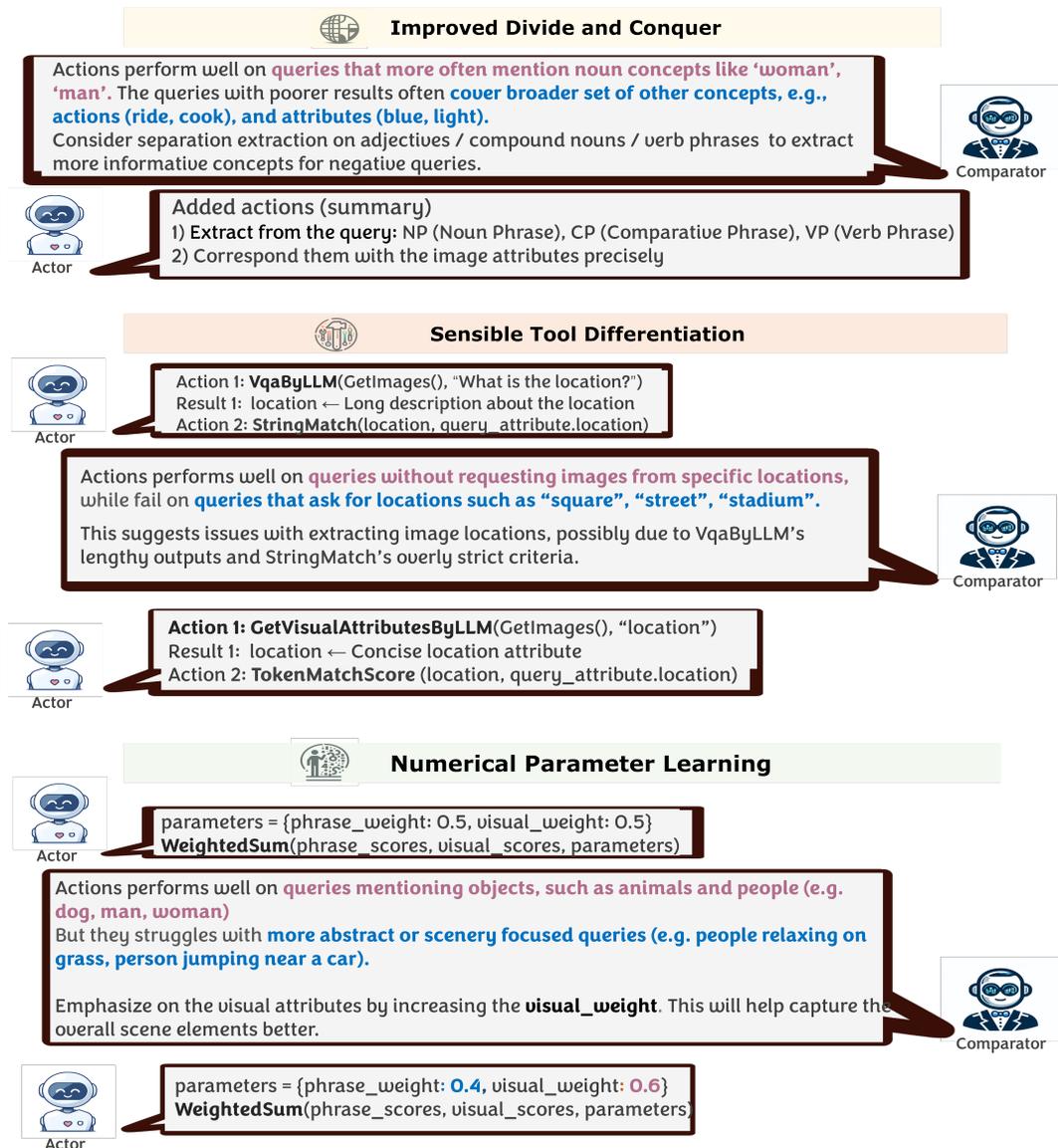


Figure 4.4: Representative instruction types from the Comparator for solving complex user queries. We show three cases where the Comparator guides the Actor towards (1) better divide-and-conquer strategies for decomposing multi-constraint requests, (2) more effective differentiation between good and bad tool usage for coordinating heterogeneous constraints, and (3) adjustments in weights to synthesize results from multiple information sources.

Table 4.5: Retrieval performance (%) on FLICKR30K-ENTITIES. AVATAR achieves up to 13.0% relative improvement in MRR over the best baseline.

	Hit@1	Hit@5	R@20	MRR
clip-vit-large-patch14	37.2	<u>56.4</u>	72.8	<u>46.3</u>
ReAct (claude3)	<u>38.8</u>	54.8	71.6	46.1
Reflexion (claude3)	28.4	53.2	75.2	41.2
AVATAR-C (claude3)	28.8	53.2	<u>78.4</u>	40.0
AVATAR (claude3)	42.4	63.0	79.2	52.3
Relative Improvement	9.2%	11.7%	5.3%	13.0%

4.5.2 Flickr30K: Image retrieval with multi-modal queries

We experiment on FLICKR30K ENTITIES [130], an image retrieval dataset where users express queries combining textual descriptions with visual references. The dataset contains 30k images, each with annotated bounding boxes and descriptive phrases. Formally, each sample consists of an image v_i , its descriptive caption q_i (which serves as the query), and entity bounding box information $T_i = \{(c_{ij}, p_{ij})\}_{j=1}^{b_i}$, where c_{ij} is the coordinate of the j -th bounding box and p_{ij} describes the entity within it. The agent can access images and bounding box information but not captions (to prevent information leakage). Each user query requires agents to understand natural language descriptions (“a person in a red shirt”) and coordinate them with visual elements (“near a bicycle”), then retrieve images that satisfy both constraints.

Takeaway 4: Batch-wise contrastive reasoning prevents overfitting to specific examples. Table 4.5 shows that our multi-agent system achieves significant improvements over existing methods, and Figure 4.5 illustrates the optimization dynamics over iterations. While ReAct agents struggle to coordinate textual and visual constraints, often applying tools based on generic patterns, AVATAR learns to systematically decompose multi-modal user requests and apply appropriate tools through multi-agent collaboration. In contrast, Reflexion agents struggle with “overfitting,” where they are easily misled by specific image data, leading to inappropriate actions (*e.g.*, trying to “extract the color of a hat” from images without hats). Our multi-agent system effectively avoids such pitfalls through batch-wise

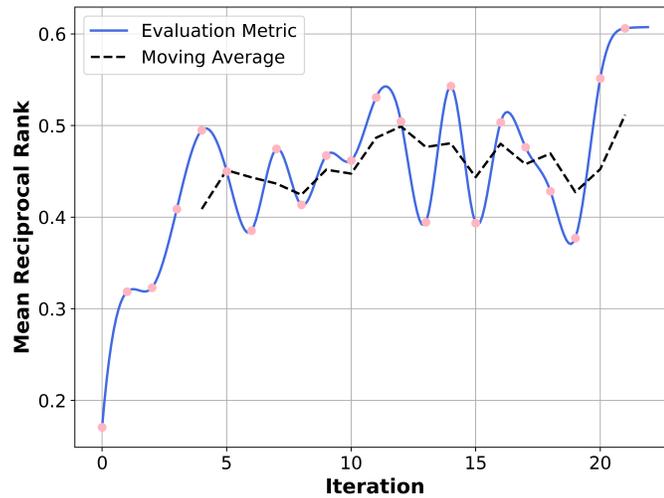


Figure 4.5: AVATAR’s optimization dynamics on FLICKR30K-ENTITIES. Performance improvement over iterations.

contrastive reasoning, which provides a broader perspective on how to handle diverse user queries.

Takeaway 5: The multi-agent system discovers novel strategies beyond tool usage.

Figure 4.6 shows the final action sequence optimized by AVATAR on FLICKR30K-ENTITIES. Through multi-agent optimization, the Actor skillfully manages input queries and leverages Inverse Document Frequency (IDF) scores to refine phrase matching, ultimately synthesizing accurate answers. Beyond using existing tools, our multi-agent system can help the Actor develop high-level strategies, such as IDF-based reweighting for multi-constraint queries, suggesting a promising direction for multi-agent systems that can adapt their strategies to complex user needs.

4.5.3 QA Benchmarks: Multi-hop reasoning tasks

We apply AVATAR to three widely used QA benchmarks where users ask complex questions requiring multi-hop reasoning. For ToolQA, we test AVATAR on two different domains: SciREX, which focuses on extracting information from full-length machine learning papers, and Agenda, which involves personal agenda-related questions. Both datasets

```

Input: Any query (example: "A man with pierced ears is wearing glasses and an orange hat .");
      Action Space: {GetImages, GetEmbeddingSimilarity, GetVisualAttributesByLLM, , ...}
Output: Retrieved Image IDs

✔ Remove empty spaces or non-alphabetic characters
Action 1: CleanQueryText[query]
Result 1: normalized_query ← "a man with pierced ears is wearing glasses and an orange hat"

✔ Get all phrases from the knowledge base
Action 2: GetBagofPhrases()
Result 2: phrases_list ← [{"a man", "grass", "sky"}, [{"a", "cat", ...}]

✔ Compute IDF for phrase importance
Action 3: ComputeIDFScores[Flatten[phrases_list]]
Result 3: idf_scores ← {"pierced": 0.5, "man": 0.0012, ...}

✔ Get visual attributes for the candidate images
Action 4: GetVisualAttributesByLLM[GetImages(), ["color", "object", "action", "count"]]
Result 4: visual_attributes ← {node_id_1: {"color": "red", ...}, node_id_2: {...}, ...}

✔ Evaluate textual and visual relevance
Action 5: [','.join(list) for list in phrases_list]
Result 5: phrase_sentences ← ["a man, grass, sky", "a, cat, playground",...]

✔ Evaluate textual and visual relevance
Action 6: ComputeEmbeddingSimilarity[normalized_query, phrase_sentences]
Result 6: text_scores

Action 7: ComputeEmbeddingSimilarity[normalized_query, visual_attributes]
Result 7: visual_scores

✔ Match query phrases with node attributes using IDF scores
Action 8: MatchQueryPhrases[normalized_query.split(), visual_attributes]
Result 8: phrase_match_scores

✔ Reweight the phrase_match_scores with IDF score
Action 9: ReweightByIDFScore[phrase_match_scores, idf_scores]
Result 9: reweighted_match_scores

✔ Aggregate scores with weighted parameters
Action 10: WeightedSum[text_scores, visual_scores, reweighted_match_scores, weights=(0.5, 0.3, 0.2)]
Result 10: aggregated_scores

✔ Normalize scores for final ranking
Action 11: NormalizeScores[aggregated_scores]
Result 11: normalized_scores = {node_id: normalized_score, ...}

Final Result: answers = GetTopkEntities[normalized_scores, k=5]
✔ Excellent task performance

```

Figure 4.6: Optimized action sequence by AVATAR on FLICKR30K-ENTITIES. The Actor learns to parse query attributes, compute semantic similarity scores using IDF weighting, and synthesize results by combining multiple matching strategies.

have easy and hard versions with varying complexity in user queries.

Takeaway 6: Contrastive reasoning is especially effective for challenging multi-hop queries. Table 4.6 shows that our multi-agent system consistently outperforms state-of-the-art methods across all three QA datasets, with especially strong results on TOOLQA.

Table 4.6: Performance (%) on three QA benchmarks. Last row shows the relative improvements over the best metric value in each column.

	HOTPOTQA	ARXIVQA	TOOLQA			
			SCIEX-EASY	SCIEX-HARD	AGENDA-EASY	AGENDA-HARD
CoT	28.0%	58.0%	1.7%	0.0%	0.0%	0.0%
ReAct	40.0%	72.0%	31.7%	<u>17.5%</u>	38.3%	<u>3.33%</u>
Reflexion	46.0%	<u>77.0%</u>	28.3%	13.3%	30.0%	<u>3.33%</u>
ExpeL	39.0%	73.0%	<u>36.7%</u>	14.5%	<u>56.6%</u>	1.67%
Retroformer (#retry=1)	<u>51.0%</u>	-	-	-	-	-
AVATAR-C	41.0%	73.0%	31.7%	13.3%	31.7%	1.67%
AVATAR	53.0%	84.0%	37.5%	23.3%	60.0%	4.17%
Relative Improvement	3.92%	9.09%	2.18%	33.1%	5.82%	25.0%

Table 4.7: Performance (%) on selected QA benchmarks. Relative improvements are computed over the best non-AvaTaR baseline.

Method	HOTPOTQA	ARXIVQA	TOOLQA (SciREX-hard)
ReAct	40.0%	72.0%	17.5%
Reflexion	46.0%	77.0%	13.3%
AvaTaR	53.0%	84.0%	23.3%
Relative Improvement	+13.0%	+9.1%	+33.1%

In SCIEX-HARD, which focuses on extracting complex information from long scientific papers where user questions combine multiple constraints, AVATAR shows a 33.1% improvement, while in AGENDA-HARD, it achieves a 25.0% relative gain. These improvements are attributed to the multi-agent system’s ability to generate optimized instructions through contrastive reasoning that help the Actor systematically decompose complex user questions, identify what information sources to query, and synthesize answers that address all aspects of the user’s request. The gains are particularly pronounced on hard versions, demonstrating that multi-agent contrastive reasoning is especially effective for teaching the Actor to handle the most challenging multi-constraint user queries.

Discussion: How multi-agent contrastive reasoning helps solve complex user queries.

Across all experiments, we observe consistent patterns in how our multi-agent system helps solve complex user queries:

- **Query decomposition:** By contrasting successful and failed queries, the Comparator identifies systematic gaps in how the Actor decomposes requests. The Actor often misses implicit constraints or fails to recognize when constraints should be applied sequentially versus in parallel.
- **Tool coordination:** Contrastive reasoning reveals patterns in tool usage, where the Actor learns which tools work best for specific constraint types and how to combine their outputs.
- **Result synthesis:** The Comparator helps the Actor learn synthesis strategies for weighting and integrating information from heterogeneous sources to produce answers that satisfy all user constraints.

This systematic learning through multi-agent contrastive reasoning enables the Actor to generalize to novel user queries, as evidenced by strong performance on held-out test sets across all benchmarks.

4.6 Conclusion

In this chapter, we introduced AVATAR, a multi-agent system designed to solve complex user queries during human-AI collaboration. The system consists of two components: an Actor that generates action sequences to answer queries, and a Comparator that analyzes the Actor’s performance through contrastive reasoning to generate holistic instructions for improvement.

The key insight behind AVATAR is that a single agent cannot easily recognize its own systematic gaps. By introducing an external Comparator that contrasts successful and failed queries, we enable the identification of patterns that the Actor itself cannot see. The batch-wise contrastive reasoning mechanism extracts robust “gradients” by analyzing multiple queries simultaneously, addressing systematic gaps in query decomposition, tool coordination, and result synthesis.

Our experiments across seven diverse datasets demonstrate the effectiveness of this approach. On the STARK benchmark from the previous chapter, AVATAR achieves up

to 16.6% relative improvement in Hit@1 over the best baselines. On multi-modal image retrieval and question-answering tasks, the multi-agent system consistently outperforms existing methods, with especially strong gains on challenging queries requiring multi-hop reasoning.

Limitations and looking ahead. While AVATAR demonstrates that multi-agent collaboration through contrastive reasoning improves performance over single-agent approaches, several limitations point toward the need for principled system-level optimization:

- AVATAR’s two-agent architecture (Actor and Comparator) represents a fixed topology; scaling to systems with many specialized agents working together requires new mechanisms for coordination.
- The Comparator generates instructions based on query-level performance, but in larger systems with multiple interacting agents, optimizing one agent in isolation may not improve overall system performance if other agents are misaligned.
- The contrastive reasoning mechanism lacks formal guarantees about convergence or alignment between local agent improvements and global system goals.

These limitations motivate the next chapter, where we address the fundamental challenge of scaling multi-agent benefits: *how can we ensure that individual agents in a compound agentic system remain aligned with shared collaborative goals?* We introduce a framework that learns globally aligned reward functions for each agent, enabling effective optimization of systems with heterogeneous agents while maintaining alignment with user needs.

Part III

Scaling Up Collaborative Multi-Agent Systems

Chapter 5

Optimizing Large-scale Multi-Agent Collaborators

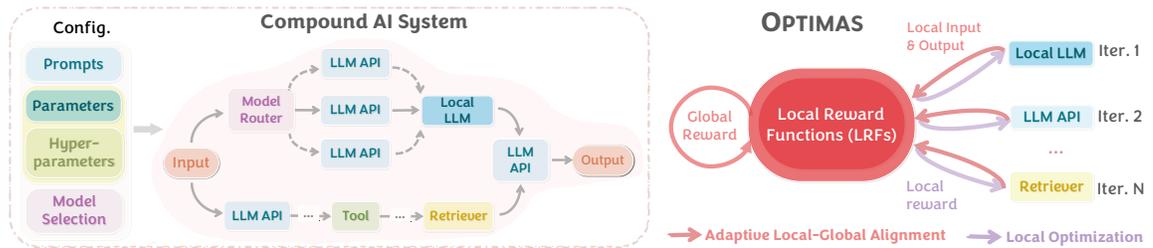


Figure 5.1: Overview of OPTIMAS. Given a multi-agent or compound AI system’s heterogeneous configurations (e.g., prompts, parameters) across multiple components, OPTIMAS maintains globally aligned *Local Reward Functions* (LRFs) as the system evolves, where each supervises a component and assigns higher local rewards to outputs with higher system performance (aka. global rewards). It iteratively adapts LRFs and optimizes each component to maximize its local reward for effective system optimization.

5.1 Introduction

In the previous chapter, addressing user queries benefits significantly from multi-agent architectures. A two-agent system with an Actor and Comparator can solve complex user queries more effectively than a single agent through contrastive reasoning. This raises a

natural question: *Can we scale these benefits further by building larger multi-agent systems with many specialized agents working together toward shared goals?*

Indeed, modern AI systems increasingly employ such multi-agent systems—also referred to as *compound AI systems* in the literature [201]—that integrate multiple agents and specialized components, such as Large Language Models (LLMs), tool-calling agents, and machine learning models like retrievers [39, 81, 200]. Throughout this chapter, we use these terms interchangeably to refer to systems where multiple AI components collaborate to serve user needs. These agents collaborate to process heterogeneous data sources and solve complex user tasks through specialized subtask allocation [21, 77, 91, 201, 216]. While compound agentic systems have yielded performance advantages over single agents [39, 63, 88, 92, 102], they introduce a fundamental challenge for human-AI collaboration: *how do we ensure that individual agents remain aligned with the shared collaborative goal?* These systems can be highly sensitive to the failure of individual agents, which leads to cascading failures in serving user needs [14, 16, 128, 161]. For example, if one agent misinterprets a user query, it can retrieve irrelevant information. This leads to downstream agents operating on incorrect inputs, producing unreliable outputs that fail to address the user’s actual needs. Therefore, optimizing compound agentic systems as a whole is crucial for maintaining alignment with user goals and achieving reliable global system performance (*i.e.*, global rewards).

However, optimizing compound agentic systems end-to-end to maintain alignment with shared goals is fundamentally challenging:

- These systems are non-differentiable, preventing gradient-based optimization.
- Individual agents have heterogeneous configurations: some agents are controlled by textual prompts, others by numerical parameters, model selection choices, or even fine-tunable weights.
- Evaluating whether the entire agentic system serves user needs requires running all agents together, which is costly.

Previous works have largely focused on optimizing individual agents in isolation, such as improving agent prompts through textual feedback [81, 107, 181, 185, 200] or selecting better models through iterative search [21, 22]. Yet these approaches fail to maintain

alignment across the full system. For example, a perfectly optimized query-understanding agent can still fail if paired with a poorly configured retrieval agent. Even when individual agents are well optimized, they may collaborate suboptimally because upstream agents lack visibility into what inputs work best for downstream agents. This misalignment means previous methods require costly system runs over many configurations to find settings where agents work well together, leading to low data efficiency and potential failures in serving user needs.

Present work. To address the challenge of scaling multi-agent systems while maintaining alignment, we propose OPTIMAS (Figure 5.1), a unified framework for optimizing compound agentic systems effectively and data-efficiently. The core idea is to achieve **collective intelligence through aligned local optimization**: we learn a **globally aligned Local Reward Function (LRF)** for each agent, such that independently improving each agent’s local performance still reliably improves the overall system’s ability to serve user needs. We show that under mild conditions, our approach converges reliably, providing strong theoretical guarantees for this alignment-based optimization. Furthermore, since the learned LRFs enable optimizing individual agents locally, OPTIMAS achieves higher data efficiency by avoiding extensive runs of the entire agentic system to achieve high global reward.

Specifically, each LRF estimates how an individual agent’s output contributes to the overall system goal. All LRFs share an LLM backbone with agent-specific projection heads to produce tailored rewards. We propose a **lightweight adaptation** mechanism using mini-batch preference data to ensure these LRFs remain aligned with the shared goal as the system evolves (Figure 5.3). Leveraging this decentralized structure, OPTIMAS applies appropriate optimization methods to each agent based on its configuration type, for example, reinforcement learning for model parameters [133, 140] or metric-guided search for prompts and hyperparameters [97, 116, 185]. Overall, OPTIMAS iteratively updates heterogeneous agent configurations toward higher global reward by using each adaptive LRF as a local objective. By optimizing each agent to maximize its local reward while maintaining alignment, OPTIMAS reduces costly full-system runs and maintains higher data efficiency.

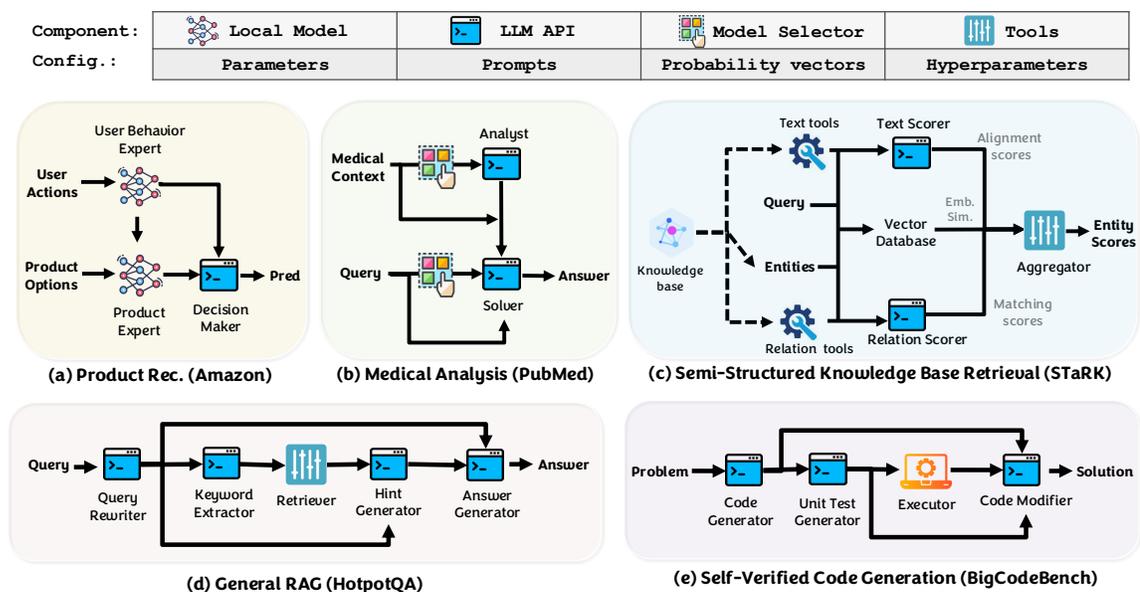


Figure 5.2: Five real-world compound AI systems. The goal is to automatically optimize configurations across a heterogeneous set of components and parameters, *e.g.*, model parameters, prompts, model selection choice, and hyperparameters. See Section 5.5.2 for details.

We conduct extensive experiments to evaluate OPTIMAS across five real-world compound agentic systems (Figure 5.2) that assist users with challenging tasks such as behavior-driven product recommendation and medical analysis. OPTIMAS consistently outperforms strong baselines, achieving an average relative improvement of 11.92% with higher data efficiency, while baseline methods occasionally improve performance. For example, while DSPy improves performance on the multi-hop QA system, it degrades performance on other collaborative tasks like product recommendation. In contrast, OPTIMAS is the *only* method that improves performance across *all* five tasks, demonstrating that our approach successfully scales multi-agent benefits. This is consistent with our theoretical guarantee (Section 5.4.4) that maintaining alignment between local agent optimization and global system goals enables effective scaling of compound agentic systems.

5.2 Related work

Multi-agent systems. Multi-agent systems show significant advantages. Small multi-agent systems (2-3 agents) with specialized roles can solve complex user queries more effectively than single agents [39, 102, 181]. However, scaling to larger compound agentic systems introduces the challenge of maintaining alignment across many heterogeneous agents. Our work addresses this scaling challenge by developing methods to keep individual agents aligned with shared collaborative goals.

Optimizing single agents. Prior work extensively optimizes individual agents, focusing on prompts for Large Language Models (LLMs) in single-step generation [18, 50, 107, 150, 175, 185, 197, 199]. However, these methods are limited in their ability to handle complex collaborative tasks. For example, addressing complex user queries often requires combining multiple agents, such as query understanding agents, retrieval agents, and synthesis agents, to serve user needs effectively.

Optimizing multi-agent compound systems. Compound agentic systems consisting of multiple agents enable more complex planning and specialized processing for collaborative tasks [39, 81, 102, 191, 206]. Previous studies typically optimize different agents separately, such as optimizing agent prompts [81, 181, 200], fine-tuning model weights using supervised learning [30, 210] or reinforcement learning [28, 99], developing model routing [21] and layer grouping [20] strategies, and selecting hyperparameters [43, 100, 129, 171]. In contrast, OPTIMAS enables end-to-end optimization across *all* agents in the compound system while maintaining alignment with shared goals.

Reward modeling and alignment for multi-agent systems. To maintain alignment in multi-agent systems, recent works break down global rewards (*e.g.*, answer accuracy) into more targeted signals for individual agents (*i.e.*, dense/process rewards). Representative approaches include leveraging or bootstrapping from human step-wise annotations [98, 144, 164]; hierarchical planning that assigns rewards to error correction steps [169]; using Monte Carlo Tree Search to assign credit to intermediate reasoning steps [19, 72, 106, 141, 167] or actions [29, 34]. Recently, Chen et al. [30] leverage Bayesian optimization to decompose global losses into local losses for optimizing model weights. OPTIMAS differs by

dynamically maintaining alignment between local agent rewards and global system goals through preference-based adaptation. This design scales to compound agentic systems with heterogeneous agents, is applicable to both differentiable and non-differentiable configurations, and does not require fixed decomposition or extensive retraining. The aligned local optimization avoids extensive full-system runs and offers higher data efficiency for scaling multi-agent systems. Moreover, we provide theoretical analysis to prove the convergence of our framework. We highlight our key contributions in Table 5.1.

Table 5.1: **A comparison of OPTIMAS with selected methods.** OPTIMAS optimizes compound systems with heterogeneous configurations and enables higher data efficiency with local optimization to reduce number of system runs. We prove OPTIMAS’s convergence under mild conditions.

	Supports compound system	Optimizes heterogeneous config.	Data efficiency	Convergence guarantee
OPRO [185]	✗	✗	✗	✗
DSPy [81]	✓	✗	✗	✗
TextGrad [200]	✓	✗	✗	✗
LLMSelector [21]	✓	✗	✗	✓
OPTIMAS	✓	✓	✓	✓

5.3 Formulation: What is the objective for multi-agent systems

Compound AI system. A compound AI system is represented as a directed acyclic graph $\mathcal{G} = (\mathcal{C}, \mathcal{E})$, where $\mathcal{C} = \{C_k\}_{k=1}^K$ is a set of K distinct components (task nodes) and \mathcal{E} is the set of all possible directed edges between components. A component of the compound system can be an LLM, a general machine learning model, a model selector, *etc.* We denote the input and output to each component C_k as x_k and y_k , respectively. The system input is treated as a source node C_0 .

The system can operate with dynamic planning: for each input instance x , the connections $\mathcal{E}(x) \subseteq \mathcal{E}$ between the components can be adaptive. A directed edge $(C_i, C_j) \in \mathcal{E}(x)$

indicates that the output of component C_i is routed as input to component C_j when processing instance x . By default, we assume the component indices follow the topological order over \mathcal{E} , meaning that C_i is the upstream component of C_j if $i < j$.

Component configurations. A component $C_k : (x_k; \mathbf{v}_k) \mapsto y_k$ is controlled by a configuration policy \mathbf{v}_k . The configuration space \mathcal{Y} can either be empty (indicating no optimizable configuration for the component), discrete (e.g., textual prompts or model selections), or continuous (e.g., model parameters or hyperparameters). We denote the joint configuration policy by $\mathbf{v} = (\mathbf{v}_1, \dots, \mathbf{v}_K)$.

Forward execution. For a given input x and configuration policy \mathbf{v} , the system executes components in topological order over the edge set $\mathcal{E}(x): y_k = C_k(\{y_i \mid C_i \in pa(C_k)\}; \mathbf{v}_k)$, where $pa(C_k)$ denotes all the parents of component C_k over $\mathcal{E}(x)$. For clarity, we define the overall system as $f(x; \mathbf{v}) := y$, where y is a collection of outputs from one or more components.

Optimization objective. Given a dataset \mathcal{D} with initial inputs and a user-defined global reward function $R : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ that evaluates the final system output, the optimization goal is to find the configuration policy $\mathbf{v}^*(x)$ that maximizes the expected global reward:

$$\mathbf{v}^* = \arg \max_{\mathbf{v}} \mathbb{E}_{x \sim \mathcal{D}} [R(x, f(x; \mathbf{v}))]. \quad (5.1)$$

5.4 OPTIMAS: Unified Multi-Agent System Optimization

Challenges. Directly optimizing the objective in Eq. 5.1 is difficult. As the configuration spaces are typically non-differentiable, gradient-based optimization cannot be used. Moreover, each policy \mathbf{v}_k may control a different configuration type, so the joint policy \mathbf{v} can span heterogeneous spaces. Therefore, previous efforts [21, 81, 200, 210] largely focus on optimizing the policy for single types of configurations, which simplifies the optimization problem; however, this also leads to suboptimal compound systems.

Key intuition. To address the challenges, our approach (Figure 5.3) learns Local Reward

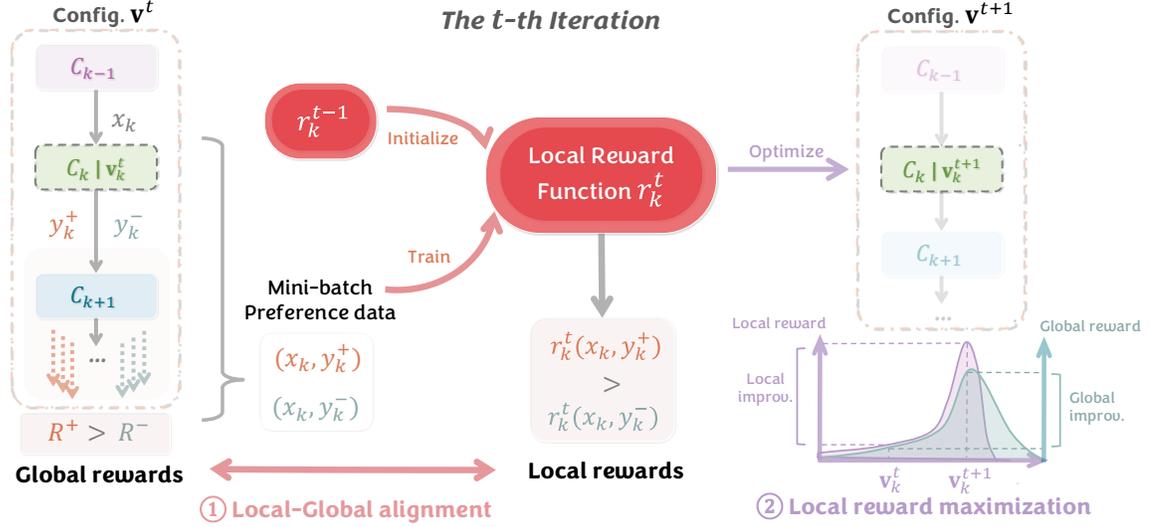


Figure 5.3: OPTIMAS optimization iteration. At each iteration, OPTIMAS updates a component C_k by first collecting a mini-batch of preference data and adapting its Local Reward Function r_k to remain aligned with the global task metric. This alignment helps ensure that optimizing the component to maximize its local reward also improves the global reward.

Functions (LRFs) that align with the global reward for individual components, allowing local and independent optimization on heterogeneous components using different optimization approaches.

Such **local-global alignments** (Section 5.4.1) encourage the global reward to increase during local optimizations (Section 5.4.3). Moreover, as the system configurations change during optimization, the LRFs should be adapted to remain aligned. To ensure alignment, OPTIMAS employs a **lightweight adaptation** mechanism that updates LRFs with minimal data sampled from the system, preserving consistency with the global reward (Section 5.4.2).

5.4.1 Local Reward Functions: Align local with global objective

Definition (Local Reward Function (LRF)). An LRF on component C_k is defined as $r_k : (x_k, y_k) \rightarrow \mathbb{R}$, which evaluates the component’s output y_k given the context x_k .

Implementation. We implement all LRFs with a LLM backbone ϕ and separate linear

heads h_k for a component C_k . The backbone encodes the concatenated text inputs $[x_k, y_k]$ into an embedding, and the corresponding head projects this embedding to a scalar reward value. Using such a multitask neural network ensures scalability with large number of components and reduces memory costs. Specifically, each LRF is modeled as:

$$r_k(x_k, y_k) = h_k \circ \phi([x_k, y_k]), \quad \text{for all } k \text{ if } \mathbf{v}_k \text{ is non-empty.} \quad (5.2)$$

Property (Local–global alignment). An LRF r_k is said to be *aligned* with the global reward R if, for every input x and for any two candidate outputs y_k^+, y_k^- of C_k ,

$$\begin{aligned} r_k(x_k, y_k^+) &\geq r_k(x_k, y_k^-) \\ \implies \mathbb{E}_{\text{downstream}} \left[R(x, f(x; \mathbf{v}_{-k})) \mid y_k^+ \right] &\geq \mathbb{E}_{\text{downstream}} \left[R(x, f(x; \mathbf{v}_{-k})) \mid y_k^- \right], \end{aligned} \quad (5.3)$$

where \mathbf{v}_{-k} denotes the configurations of all *downstream* components (those that directly or indirectly receive information originating from C_k). The expected global reward for each candidate output is estimated via Monte Carlo sampling. This involves executing the downstream components with the candidate output and the outputs from the non-downstream components fixed, capturing their stochasticity, and averaging the resulting global rewards from the final system outputs.

Objective of reward functions. To make each LRF r_k conform to the local-global alignment property, we collect $\mathcal{D}_k(\mathbf{v})$, a preference dataset under the current system configuration \mathbf{v} , and train each r_k using a pairwise log-sigmoid ranking loss:

$$\mathcal{L}_k(\mathcal{D}_k(\mathbf{v})) = -\mathbb{E}_{(x_k, y_k^+, y_k^-) \sim \mathcal{D}_k(\mathbf{v})} \left[\log \sigma(r_k(x_k, y_k^+) - r_k(x_k, y_k^-)) \right], \quad (5.4)$$

The collection of $\mathcal{D}_k(\mathbf{v})$ follows the following steps: 1) execute the compound system up to C_k and record the partial trajectory $\langle x, (x_1, y_1), \dots, (x_{k-1}, y_{k-1}) \rangle$; 2) sample two candidate outputs for C_k (e.g., via higher-temperature decoding or alternate hyperparameters); and 3) estimate their expected task metrics according to the expectation terms on the right-hand side of Eq. 5.3. The output with the higher expected value is labeled as y_k^+ , and the other as y_k^- in $\mathcal{D}_k(\mathbf{v})$.

5.4.2 Adaptation: How to maintain alignment during optimization

Problem: Misaligned LRFs in the evolving system. As the system configuration changes during optimization, LRFs trained under a previous configuration \mathbf{v}^t may become inaccurate under the updated configuration \mathbf{v}^{t+1} . Specifically:

- After updating C_k , the same outputs from its upstream component C_i ($i < k$) may lead to different global reward, making r_i misaligned.
- Its downstream components C_j ($j > k$) now receive inputs generated by the updated C_k , which may fall outside the distribution seen by their LRFs.

These shifts accumulate over time, degrading the local–global alignment property (Eq. 5.3) that LRFs are designed to satisfy.

However, retraining all LRFs from scratch after every configuration update is expensive. To address this, we develop a lightweight adaptation strategy that incrementally refines the LRFs as the system changes, maintaining alignment without full retraining.

Stage 1: Initial reward modeling. Given the initial system configuration and a dataset with initial inputs, we first construct an offline preference dataset for each component and train its LRF to convergence. This offline phase establishes well-aligned LRFs that accurately reflect each component’s contribution to the global reward.

Stage 2: Online reward function adaptation. When any configuration changes, we sample a small batch of input data and construct a mini-batch of preference data \mathcal{B}_k for each component C_k using the steps described in Section 5.4.1. We then optimize the LRF on C_k on the objective $\mathcal{L}_k(\mathcal{B}_k)$ following the definition in Eq. 5.4. To enable stable optimization and improve data efficiency, we maintain a buffer of previously generated preference data in \mathcal{B}_k . This adaptation helps maintain the local-global alignment property in Eq. 5.3.

5.4.3 Optimization: How to optimize with aligned reward functions

Local Optimization. As each component has its own LRF, OPTIMAS flexibly applies a specialized optimization method for each component. Given a globally aligned LRF r_k ,

we perform local optimization on each component C_k to improve its configuration \mathbf{v}_k . Specifically, we solve:

$$\mathbf{v}_k^{t+1} = \arg \max_{\mathbf{v}_k \in \mathcal{V}_k} \mathbb{E}_{x_k} [r_k(x_k, C_k(x_k; \mathbf{v}_k))] \quad \text{subject to} \quad d(\mathbf{v}_k, \mathbf{v}_k^t) \leq \delta, \quad (5.5)$$

where \mathbf{v}_k^t is the configuration before the t -th iteration, $d(\cdot, \cdot)$ is a distance function over configurations, and δ defines a trust region that bounds allowable updates. This constraint ensures that r_k is used within a region where it is expected to produce reliable evaluations. In practice, explicitly setting the trust region threshold δ can be difficult due to heterogeneous configuration types (*e.g.*, continuous weights or discrete tokens). Instead, we adopt a conservative number of update steps to restrict the magnitude of change during each iteration.

The specific optimization approach depends on the configuration type:

- **Prompt tuning.** For textual prompts, we apply prompt optimization algorithms [181, 185], using r_k as the evaluation metric. We sample multiple prompts limited by a max number of prompt candidates. The prompts are ranked by average reward over validation instances, and the best-performing prompt is selected.
- **Model fine-tuning.** When C_k is an LLM or neural model with trainable parameters, we apply reinforcement learning algorithms, *e.g.*, Proximal Policy Optimization (PPO) [140], using r_k as the critic. The model parameters are updated for a small and fixed number of steps.
- **Model selection and hyperparameter tuning.** For discrete or low-dimensional continuous configurations, such as model selection, tool routing, or scalar hyperparameters, we formulate the optimization as a sampling problem parameterized by a probabilistic distribution. Since these configurations are instance-specific, the expectation in Eq. 5.5 reduces to a single input. For each input x , we evaluate a set of candidate configurations using the LRF r_k , and compute a probability distribution over candidates proportional to $\exp\{r_k(x_k, C_k(x_k; \mathbf{v}_k))\}$. This distribution is then used to sample the configuration update for the current iteration.

Under a conservative update to the configuration of a component C_k , the expected

global reward is guaranteed to maintain or improve, if the local-global alignment property in Eq. 5.3 holds.

Overall algorithm (Algorithm 1). Starting with the initial system configuration, OPTIMAS leverages initial reward modeling to learn a set of LRFs that are well-aligned with the global reward. At each optimization iteration, OPTIMAS randomly selects a component to optimize, conducts local optimization, and if the local configuration change leads to improved global reward, it updates the system and adapts the LRF using minimal amount of data. To prevent potential cascading errors, the new configuration is accepted only if it improves the global reward on a small validation set. Since the optimization with LRFs is conducted locally, the number of system runs to achieve a high global reward is reduced, as we later show in the experiments.

5.4.4 Theory: Why does local optimization improve global rewards

We prove that the local–global alignment property holds for the LRFs constructed in Section 5.4.1.

Theorem 5.4.1. *Under the conditions specified below, the maximizer of Eq. 5.4 satisfies the local-global alignment property Eq. 5.3. In addition, maximizing $r_k(x, C_k(x_k; \mathbf{v}_k))$ over v_k and maximizing $R(x, f(x; \mathbf{v}_{-k}) \mid C_k(x_k; \mathbf{v}_k))$ over v_k will yield the same solution.*

According to the procedure described in Section 5.4.1, the positive and negative pairs are determined by comparing the expected task metrics. We assume the estimated metrics at y_k^+ and y_k^- are chosen following

$$\mathbb{P}(y_k^+ \text{ is labeled as positive}) = \sigma_\alpha \left(\mathbb{E}_{\text{downstream}} \left[R(x, f(x; \mathbf{v}_{-k}(x))) \mid y_k^+ \right] - \mathbb{E}_{\text{downstream}} \left[R(x, f(x; \mathbf{v}_{-k}(x))) \mid y_k^- \right] \right),$$

where $\sigma_\alpha(u) = \frac{1}{1+\exp(-\alpha u)}$ is the sigmoid function with parameter $\alpha > 0$. $\alpha = +\infty$ corresponds to the case where the pairs are chosen deterministically.

Proof. We first present a lemma.

Algorithm 1 OPTIMAS: Multi-agent System Optimization with Locally Aligned Local Reward Functions

Require: Components $\mathcal{C} = \{C_1, \dots, C_K\}$
Require: Initial configuration $\mathbf{v}^0 = (\mathbf{v}_1^0, \dots, \mathbf{v}_K^0)$
Require: Initial reward function parameters $\Theta^0 = \{\theta_1^0, \dots, \theta_K^0\}$
Require: Global reward $R(\cdot)$, preference dataset size k , total iterations T
Require: Training dataset \mathcal{D} and validation dataset D_v

- 1: $\mathbf{v}^* \leftarrow \mathbf{v}^0$
- 2: $\Theta^t \leftarrow \Theta^0$
- 3: **for** $t = 0, \dots, T - 1$ **do**
- 4: *// Scheduler: choose a component to optimize*
- 5: $i_t \sim \text{Uniform}(\{1, \dots, K\})$
- 6: *// Local optimization for the chosen component C_{i_t}*
- 7: **if** C_{i_t} is an LLM with prompts **then**
- 8: $\tilde{\mathbf{v}}_{i_t}^{t+1} \leftarrow \text{PROMPTOPTIMIZATION}(\mathbf{v}_{i_t}^t, \theta_{i_t}^{t+1})$
- 9: **else if** C_{i_t} has trainable weights **then**
- 10: $\tilde{\mathbf{v}}_{i_t}^{t+1} \leftarrow \text{PPOTRAIN}(\mathbf{v}_{i_t}^t, \theta_{i_t}^{t+1})$
- 11: **else if** C_{i_t} has a hyperparameter configuration **then**
- 12: $\tilde{\mathbf{v}}_{i_t}^{t+1} \leftarrow \text{HYPERPARAMETERSEARCH}(\mathbf{v}_{i_t}^t, \theta_{i_t}^{t+1})$
- 13: $\tilde{\mathbf{v}} \leftarrow \mathbf{v}^t$ with $\mathbf{v}_{i_t}^t$ replaced by $\tilde{\mathbf{v}}_{i_t}^{t+1}$
- 14: *// Validation*
- 15: **if** $\sum_{x_v \in D_v} R(x_v, f(x_v; \tilde{\mathbf{v}})) > \sum_{x_v \in D_v} R(x_v, f(x_v; \mathbf{v}^t))$ **then**
- 16: $\mathbf{v}^{t+1} \leftarrow \tilde{\mathbf{v}}; \quad \mathbf{v}^* \leftarrow \mathbf{v}^{t+1}$
- 17: *// Reward-model adaptation*
- 18: $D_t \leftarrow \text{COLLECTPREFERENCE}(\mathcal{D}, \Theta^t, k)$ *// create k (x^+, x^-) pairs*
- 19: $\Theta^{t+1} \leftarrow \text{REWARDMODELTRAIN}(\Theta^t, D_t)$
- 20: **else**
- 21: $\mathbf{v}^{t+1} \leftarrow \mathbf{v}^t; \quad \Theta^{t+1} \leftarrow \Theta^t$
- 22: **return** \mathbf{v}^*

Lemma 5.4.1. Suppose $(\mathbf{x}, y) \in \mathbb{R}^p \times \{-1, 1\}$ follows the distribution $\mathbb{P}(y = 1 \mid \mathbf{x}) = \sigma_1(p^*(x))$ for some function $p : \mathbb{R}^p \rightarrow (0, 1)$. Then

$$\arg \max_p \mathbb{E}[\log(\sigma_1(y \cdot p(x)))] = p^*.$$

Proof. We take the derivative of the left-hand side with respect to p and set it to 0:

$$\mathbb{E} \left[\frac{\sigma_1(y \cdot p(x)) \cdot \sigma_1(-y \cdot p(x))}{\sigma_1(y \cdot p(x))} \cdot y \right] = 0,$$

which is equivalent to

$$\mathbb{E}[\sigma_1(-y \cdot p(x)) \cdot y] = 0.$$

We then have

$$\sigma_1(-p(x))\sigma_1(p^*(x)) - \sigma_1(p(x))\sigma_1(-p^*(x)) = 0,$$

and therefore $p(x) = p^*(x)$. \square

Applying this lemma, we can obtain the solution to Eq. 5.4 is $\alpha \cdot \mathbb{E}_{\text{down}}[R(f(x; \mathbf{v}_{-k})) \mid y_k]$ for some positive α , and therefore it satisfies the local-global alignment property Eq. 5.3.

In the following, we then prove that maximizing $r_k(x, \pi_k(x))$ over v_k and maximizing $R(x, f(x; \mathbf{v}_{-k}) \mid \pi_k(x))$ over v_k will yield the same solution.

Lemma 5.4.2. *Assume the local-global alignment property Eq. 5.3 holds for component C_k . Let $\mathbf{v}(x)$ and $\tilde{\mathbf{v}}(x)$ be two configuration policies that differ only in the policy for component C_k ; denote the corresponding local outputs by y_k and \tilde{y}_k , respectively. If*

$$\mathbb{E}_x[r_k(x_k, \tilde{y}_k)] > \mathbb{E}_x[r_k(x_k, y_k)],$$

then

$$\mathbb{E}_x[R(x, f(x; \tilde{\mathbf{v}}(x)))] \geq \mathbb{E}_x[R(x, f(x; \mathbf{v}(x)))] .$$

Proof. Fix an arbitrary input instance x . Because the two policies differ only at C_k , all other component configurations remain the same, so we can write

$$f(x; \mathbf{v}_{-k}(x), y_k) \quad \text{and} \quad f(x; \mathbf{v}_{-k}(x), \tilde{y}_k),$$

where $\mathbf{v}_{-k}(x)$ denotes downstream configurations (independent of the choice at C_k). By assumption on the expected local reward, we have $r_k(x_k, \tilde{y}_k) \geq r_k(x_k, y_k)$ for almost every x . Applying the alignment property Eq. 5.3 pointwise yields

$$\mathbb{E}_{\text{downstream}}[R(x, f(x; \mathbf{v}_{-k}(x), \tilde{y}_k)) \mid x_k] \geq \mathbb{E}_{\text{downstream}}[R(x, f(x; \mathbf{v}_{-k}(x), y_k)) \mid x_k].$$

Taking the expectation over x (law of total expectation) gives

$$\mathbb{E}_x[R(x, f(x; \tilde{\mathbf{v}}(x)))] \geq \mathbb{E}_x[R(x, f(x; \mathbf{v}(x)))] .$$

Hence increasing the expected local reward for C_k cannot decrease, and may strictly increase, the expected global objective. \square

\square

Since solving (5.1) is generally challenging, we introduce some regularity conditions to make the convergence analysis tractable.

As the configurations \mathbf{v} are heterogeneous, where some of the coordinates are discrete, and some are continuous, without loss of generality, we assume the first M configurations $\mathbf{v}^{(1)} = \{\mathbf{v}_1, \dots, \mathbf{v}_M\}$ are continuous, and the last $(K - M)$ configurations $\mathbf{v}^{(2)} = \{\mathbf{v}_{M+1}, \dots, \mathbf{v}_K\}$ are discrete. We write the objective function $\mathbb{E}_{x \sim \mathcal{D}}[R(x, f(x; \mathbf{v}))]$ as $l(\mathbf{v}) = l(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_K) := l(\mathbf{v}^{(1)}, \mathbf{v}^{(2)})$.

Assumption 5.4.1. *Suppose for any given configuration $\mathbf{v}^{(2)}$, the initial level set $\{\mathbf{v}^{(2)} : l(\mathbf{v}^{(1)}, \mathbf{v}^{(2)}) \leq l(\mathbf{v}^{0,(1)}, \mathbf{v}^{(2)})\}$ is a compact set, where $\mathbf{v}^{0,(1)}$ is the initialization used in the algorithm for $\mathbf{v}^{(1)}$. In addition, for every component k and every fixed \mathbf{v}_{-k} , $l(\cdot, \mathbf{v}_{-k})$ has a unique maximizer.*

Theorem 5.4.2. *Under Assumption 5.4.1, the algorithm will converge to the component-wise maximum, that is, the limit point \mathbf{v}^* satisfies*

$$l(\mathbf{v}^*) \geq l(\mathbf{v}_k, \mathbf{v}_{-k}^*),$$

for any $k \in [K]$ and any \mathbf{v}_k .

Proof. We first show that the algorithm is essentially performing coordinate maximization on $l(\mathbf{v})$. Given a previous configuration \mathbf{v}^t , at time t , the updated configuration \mathbf{v}^{t+1} only changes the configuration of a single component, say, C_k . As the change is solved by maximizing $r_k(x, C_k(x_k; \mathbf{v}_k))$, by Theorem 5.4.1, this is equivalently maximizing $l(\mathbf{v}_k, \mathbf{v}_{-k})$ for the k -th coordinate.

To rule out cycling, we first prove that the discrete block $\mathbf{v}^{(2)}$ stabilizes. Consider the sequence $(\mathbf{v}^{1,(1)}, \mathbf{v}^{1,(2)}), (\mathbf{v}^{2,(1)}, \mathbf{v}^{2,(2)}), \dots, (\mathbf{v}^{t,(1)}, \mathbf{v}^{t,(2)}), \dots$. We first show that there exists $T > 0$, such that for all $k > 0$, $\mathbf{v}^{T+k,(2)} = \mathbf{v}^{T,(2)}$.

As $\mathbf{v}^{(2)}$ are discrete, there are finitely many different configurations. In addition, according to the assumption that the coordinate-wise maximum is unique, each update will result in a strict increase in the objective function. Therefore, after a finite number of iterations, $\mathbf{v}^{(2)}$ will not change.

Now when we consider all the iterations that are later than the time T , $\mathbf{v}^{(2)}$ is fixed, and we only need to consider the update regarding $\mathbf{v}^{(1)}$. In this case, we apply Theorem 4.1 of [163] and complete the proof. \square

In fact, our theoretical analysis shows that by conducting local optimization, OPTIMAS is essentially performing coordinate maximization. Therefore, existing convergence results for coordinate maximization directly apply. Note that the block-coordinate (round-robin) updates adopted in OPTIMAS do not guarantee global optimality in non-convex problems, but this does not constitute a flaw unique to our method; rather, it reflects a standard limitation of non-convex optimization broadly, and our global convergence guarantees only hold under additional structural assumptions, such as Polyak-Łojasiewicz or Kurdyka-Łojasiewicz conditions.

5.5 Experimental Setup

5.5.1 Benchmarks and Evaluation

We evaluate OPTIMAS on five real-world tasks (Figure 5.2):

- **AMAZON (Behavior-Driven Next-Item Recommendation)** [74]: The corpus is derived from the Amazon MMLU dataset. Each instance consists of a user’s historical behavior sequence (views, clicks, purchases) and the target “next item” to be recommended. We split it into 335 / 60 / 99 user sequences (train / validation / test) after filtering malformed entries. The evaluation metric is accuracy, *i.e.*, whether the predicted item matches the ground truth.

- **PUBMEDQA (Medical Analysis based QA)** [73]: This dataset contains biomedical abstracts paired with yes/no/maybe answers to research questions. We keep the original “expert” split and discard ambiguous samples, resulting in 475 / 25 / 500 question-abstract pairs. Our compound system frames the task as three-way classification; exact-match accuracy is reported.
- **STARK-PRIME (Semi-Structured KB Retrieval)** [182]: This benchmark blends free-text passages with relational triples from biomedical knowledge graphs. Queries are natural-language questions; relevance labels are automatically propagated from the original STaRK annotations. We use the original dataset split: 495 / 51 / 96 queries. Performance is measured by Hit@1, the proportion of queries for which the top-ranked result is a ground truth item.
- **HOTPOTQA (Retrieval-Augmented Multi-Hop QA)** [189]: We keep the official train/dev/test splits: 1000, 250, and 100 questions respectively. Each example contains a question and its human-annotated answer. We report answer-level F1 score.
- **BigCodeBench (Self-Verified Code Generation)** [222]: We use a subset of the *full-instruction* subset of BigCodeBench due to efficiency considerations. After proportionally reducing the data, we obtain 500 / 25 / 70 coding tasks. Each sample includes a natural-language specification and reference unit tests. Our metric is *pass@1*: the proportion of generated programs that pass all tests in one try.

5.5.2 Multi-Agent System Design

We design a compound system per benchmark with diverse and common patterns for agentic systems. Table 5.2 summarizes each pipeline’s modules. All systems are accessible in our code repository.

The table illustrates how different modules require different types of optimization, from prompt tuning and model-parameter fine-tuning (PPO) to discrete model selection and hyperparameter search. By unifying these heterogeneous updates within our OPTIMAS framework, we effectively coordinate local improvements to achieve consistent global reward gains.

Table 5.2: **Modules, models, and optimization methods.** See system-specific details below for the search spaces of *Model Selection*, *Aggregator*, and *Retriever*.

System	Module	Model	Config	Optimization
Amazon	Session Analyzer	Qwen 2.5 1.5B	Model Params ($\text{lr}=10^{-5}$)	PPO (RL)
	Candidate Profiler	Qwen 2.5 1.5B	Model Params ($\text{lr}=10^{-5}$)	PPO (RL)
	Next Item Decider	GPT-4o-mini	Prompt	Prompt Opt.
PubMed	Context Model Selector	–	Model Selection (LLMs)	Hyperparam Search
	Context Analyst	One of $\{\text{gpt-4o}, \dots\}$	Prompt	Prompt Opt.
	Solver Model Selector	–	Model Selection (LLMs)	Hyperparam Search
	Problem Solver	One of $\{\text{gpt-4o}, \dots\}$	Prompt	Prompt Opt.
STARK-PRIME	Text Scorer	Claude 3 Haiku	Prompt	Prompt Opt.
	Relation Scorer	Claude 3 Haiku	Prompt	Prompt Opt.
	Aggregator	–	Coefficients $\in \{0.1, 1.0\}$	Hyperparam Search
HotpotQA	Question Rewriter	GPT-4o-mini	Prompt	Prompt Opt.
	Info Extractor	GPT-4o-mini	Prompt	Prompt Opt.
	Retriever	–	$k \in \{1, 5, 10, 25\}$	Hyperparam Search
	Hint Generator	GPT-4o-mini	Prompt	Prompt Opt.
	Answer Generator	GPT-4o-mini	Prompt	Prompt Opt.
BigCodeBench	Code Generator	Claude 3 Haiku	Prompt	Prompt Opt.
	Unit Test Generator	Claude 3 Haiku	Prompt	Prompt Opt.
	Final Code Generator	Claude 3 Haiku	Prompt	Prompt Opt.

System-specific details.

- **AMAZON:** *Session Analyzer* and *Candidate Profiler* both use the Qwen 2.5 1.5B model; we optimize their model parameters with PPO reinforcement learning [140]. This helps each module better encode task-specific knowledge, *i.e.*, user sessions and product candidates. The final *Next Item Decider* is a GPT-4o-mini module, whose prompt we optimize.
- **PUBMEDQA:** Two modules (*Context Model Selector* and *Solver Model Selector*) each do discrete model selection from a list of seven LLMs (GPT-4o, GPT-4o-mini, GPT-3.5-turbo, GPT-4-turbo, Claude-3.5-Haiku, Claude-3.5-Sonnet, Claude-3.7-Sonnet). At inference time, these selectors use a reward model to pick the best LLM for each input instance. The *Context Analyst* and *Problem Solver* modules then receive the chosen model and optimize only the prompt for improved medical QA performance.
- **STARK-PRIME:** We have two scoring modules (*Text Scorer*, *Relation Scorer*), both using Claude 3 Haiku with prompt optimization. The *Aggregator* merges these two scores; we tune two numeric weights, `relation_weight` and `text_weight`, each set in

{0.1, 1.0}.

- **HOTPOTQA**: Four GPT-4o-mini modules (*Question Rewriter*, *Info Extractor*, *Hint Generator*, *Answer Generator*) each rely on prompt optimization to improve multi-hop reasoning. The *Retriever* has a key hyperparameter k , the number of passages to retrieve, which we search over {1, 5, 10, 25}.
- **BigCodeBench**: All three modules (*Code Generator*, *Unit Test Generator*, *Final Code Generator*) are Claude 3 Haiku LLMs; each uses prompt optimization to iteratively refine code solutions based on test outcomes.

5.5.3 Baselines

We compare OPTIMAS against the following baselines:

- **Unoptimized**: The system uses default settings for all components.
- **LLMSelector** [21]: A lightweight policy selects the best LLM per component via model routing, without updating other configurations. Only applicable on PUBMEDQA. We run LLMSelector with the LLMDIAGNOSER to estimate per-module performance, following their procedure. We perform two rounds of allocation updates with 100 training examples each round.
- **REINFORCE** [175]: A policy-gradient baseline that directly updates the parameters of local LLM components to maximize the task reward. This method is only applicable on AMAZON, where we deploy two locally hosted LLMs with trainable parameters. We optimize each component with REINFORCE using sampled trajectories from the full system and propagate the scalar reward back to the corresponding module-level policy. We use 16 rollouts per step to estimate the expected system performance, with learning rate 10^{-5} for 100 training steps.
- **Hierarchical Behavior Cloning (HBC)** [86]: A hierarchical imitation learning method that optimizes components to produce outputs similar to those that lead to high global rewards. We collect successful trajectories to approximate ground truth intermediate outputs, and then perform supervised updates on the local components to mimic the ideal

behavior. We run HBC using the collected preference dataset by replacing the original reward model with the embedding similarity score.

- **TextGrad** [200]: A gradient-based prompt tuning method using estimated gradients from black-box LLMs to improve prompt efficacy. We run TextGrad using GPT-4o mini to optimize each component’s prompt independently in separate epochs. Validation is performed every two optimization steps using 20 held-out validation instances.
- **DSPy** [81, 116]: A prompt optimization framework using the MIPRO algorithm that jointly refines module-level instructions and few-shot demonstrations. For fair comparison, we disable the few-shot example and system prompt optimization and only conduct optimization on the user instructions. We dynamically set the number of iterations for the MIPRO optimizer to match the budget of system runs in Table 5.6.

Moreover, we include a single-LLM reference that prompts an LLM to complete the task directly, which serves to justify our compound system design.

For TextGrad, DSPy, and OPTIMAS, we consistently use the same 20 held-out validation instances on each dataset to select the best configurations.

5.5.4 Optimization Setup

At each optimization iteration t , we randomly select a module to optimize. Before on-policy optimization, we train a reward model on preference pairs (“chosen” vs. “rejected” system outputs) so it can assign higher scores to better outputs. Table 5.3 highlights the main hyperparameters for this stage.

After training the reward model, we run iterative on-policy optimization for each module in the compound AI system. Table 5.4 lists the key hyperparameters. When a module is selected, we collect a small preference dataset, retrain (or adapt) the reward model as needed, and then locally optimize that module’s parameters or prompts to maximize its local reward.

All experiments were run on a node with 8 NVIDIA A100 GPUs (80 GB memory each); depending on the complexity of the compound system and hyperparameters, training and optimization typically finished in 2–8 hours.

Table 5.3: **Key hyperparameters for the local reward model training.** We use LoRA for memory efficiency and an early-stopping mechanism based on the evaluation loss to reduce overfitting.

Parameter	Value
Base model	Llama 3 8B Instruct
LoRA rank	32
LoRA alpha	16
Maximum sequence length	2048 tokens
Learning rate	2e-6
Number of epochs	25
Batch size	32

Table 5.4: **Key hyperparameters for on-policy optimization.** The *train size* limits how many examples are used to train each module’s local configuration, while the *search size* sets how many samples we use when searching for the best local update.

Parameter	Value / Description
Train size	50
Search size	50
Prompt candidates	3
Local optimization steps	3
Fresh input size	20
Validation size	20

5.6 Results

5.6.1 Main Results: How well does OPTIMAS compare to baselines

We compare OPTIMAS with the baselines under similar numbers of system runs. Under this controlled data cost (Table 5.6), OPTIMAS consistently improves global rewards across all compound systems, achieving an average relative improvement of 11.92% compared to the best baseline.

For REINFORCE, while it yields performance gain from the unoptimized baseline, OPTIMAS outperforms it by around 3%. Moreover, REINFORCE requires collecting the

Table 5.5: Performance of each method on the compound systems. The best and second-best results in each column are highlighted. Relative improvement is computed with respect to the best baseline.

	AMAZON Product Rec. (Acc.)	PUBMEDQA Medical Analysis (Acc.)	STARK-PRIME Complex Retrieval (MRR)	HOTPOTQA RAG (F1)	BIGCODEBENCH Verified Code Gen. (Pass Rate)
Single LLM	20.20±1.43	54.13±2.73	0.00±0.00	21.58±1.24	35.47±0.34
Unoptimized	21.21±3.78	57.46±0.75	40.73±0.64	33.80±1.51	36.67±1.35
REINFORCE	21.89±2.65	–	–	–	–
LLMSelector	–	67.93±0.09	–	–	–
HBC	21.55±2.07	58.80±0.58	36.95±0.59	21.16±0.97	27.78±2.08
TextGrad	20.88±3.53	56.96±2.24	41.31±1.67	24.86±1.19	35.71±0.10
DSPy	18.18±0.82	60.26±0.40	41.40±0.04	44.90±0.32	33.81±2.75
OPTIMAS	24.24±0.82	69.13±0.33	50.54±0.70	50.48±1.48	38.92±0.36
Rel. Improv.	14.3%	1.8%	22.1%	12.4%	9.0%

Table 5.6: Number of equivalent runs on the entire systems (in thousands). We compare TextGrad, DSPy, and OPTIMAS from Table 5.5. We control the optimization process of each to use comparable system runs.

	AMAZON	PUBMEDQA	STARK-PRIME	HOTPOTQA	BIGCODEBENCH	Average
TextGrad	0.32	0.70	0.70	2.12	0.18	0.80
DSPy	0.24	0.66	0.66	2.09	0.28	0.79
OPTIMAS	0.31	0.52	0.51	2.02	0.21	0.71

reward signal from downstream Monte Carlo sampling, which requires more than three times the data compared to OPTIMAS. While the strong baseline DSPy shows notable improvements on some datasets (*e.g.*, a 21.6% gain on HOTPOTQA), its performance may be inconsistent and can even degrade the system (*e.g.*, a 14.3% drop on the AMAZON dataset). For LLMSelector, it requires 2.8*k* times of forwarding through the entire system, which is 3× more expensive than OPTIMAS.

5.6.2 Optimization Dynamics: How do configurations evolve

We study how the configurations change in the local optimization. Within a small number of iterations, OPTIMAS achieves a substantial average improvement of 41.7% over the initial global reward on the validation sets, using lightweight and data-efficient updates.

Interestingly, we observe mixed updates on prompts, model parameters, and hyperparameters, which can lead to improved global reward. For example, updating the prompt of `Text Scorer` in the 9th iteration improves global reward from 0.49 to 0.56. Among these cases, they involve optimizing different components to achieve the highest global reward empirically, showing the importance of being able to optimize different types of configurations.

Table 5.7: Average pairwise ranking accuracy on validation sets. This measures how often the method assigns a higher score to the output with higher expected global reward.

	AMAZON	PUBMEDQA	STARK-PRIME	HOTPOTQA	BIGCODEBENCH	Avg.
LLM Judge	51.25%	49.54%	54.37%	50.00%	42.45%	49.52%
OPTIMAS	84.93%	65.28%	76.64%	72.40%	90.57%	77.96%

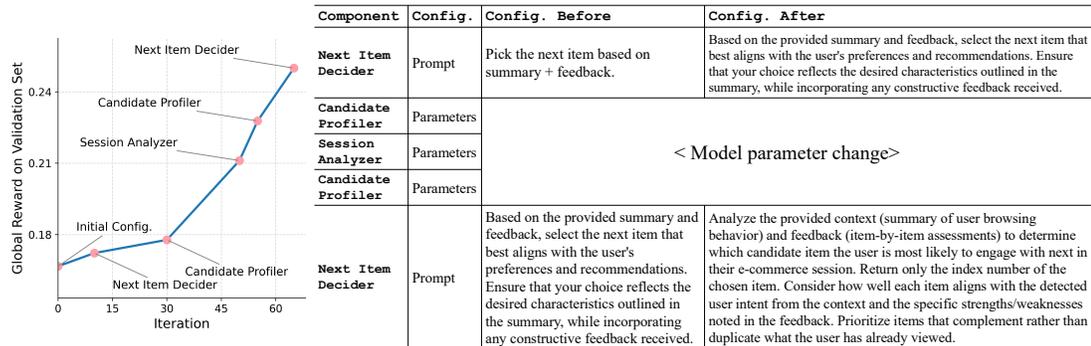
5.6.3 Alignment Quality: How well do LRFs align with global rewards

We conduct extensive in-depth studies to understand the mechanism in OPTIMAS. To measure alignment quality of LRFs, we compute pairwise ranking accuracy: the probability that an output with higher global reward receives a higher score than an output with lower global reward. This reflects how well the learned LRFs align with global rewards. We compare against an LLM Judge, which prompts a `gpt-4o` model to score the outputs of components based on 20 in-context examples. This approach is similar to prior methods such as TextGrad, which rely on few-shot reasoning over textual patterns. In Table 5.7, LLM Judge performs closer to random guessing, due to the diversity and stochasticity of component outputs that make it difficult to reason reliably. In contrast, our LRFs achieve substantially higher performance. Moreover, LRFs internalize the local-global alignment within their weights without relying on limited in-context examples, enabling more precise alignment.

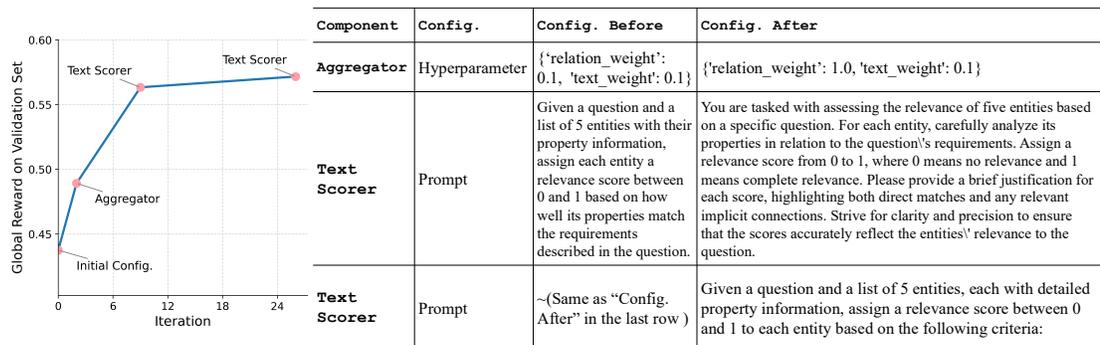
5.6.4 Sensitivity: Does better alignment lead to better optimization

To understand how alignment quality affects global reward, we conduct controlled experiments, where we select a key component, apply local reward models with varying alignment quality to optimize it, and measure the average global reward achieved after updating

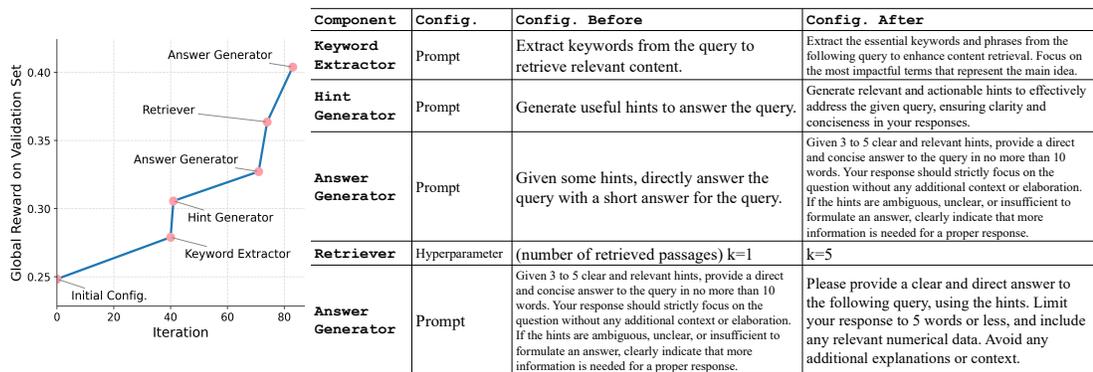
CHAPTER 5. OPTIMIZING LARGE-SCALE MULTI-AGENT COLLABORATORS 125



(a) AMAZON system



(b) PRIME system



(c) HOTPOTQA system

Figure 5.4: Global reward and configuration updates over optimization iterations. We show results for the three compound AI systems. For conciseness, we only show the local optimization steps that lead to an increase in global reward on the validation sets. The annotations show the optimized components.

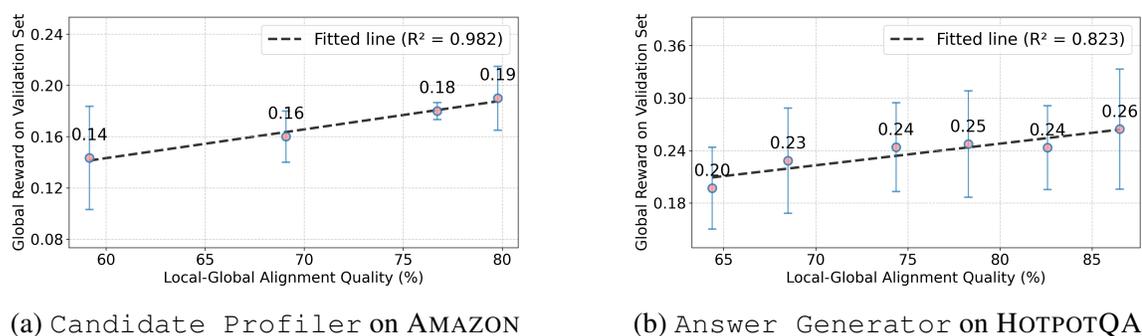


Figure 5.5: Effect of alignment quality on global rewards. Local reward models with varying alignment quality are used to optimize a selected component in each task, where we observe that higher alignment quality yields higher global rewards.

the component. In Figure 5.5, we show a strong positive correlation between the LRF’s alignment quality and the global reward improvement.

5.6.5 Interpretability: What do the learned reward functions capture

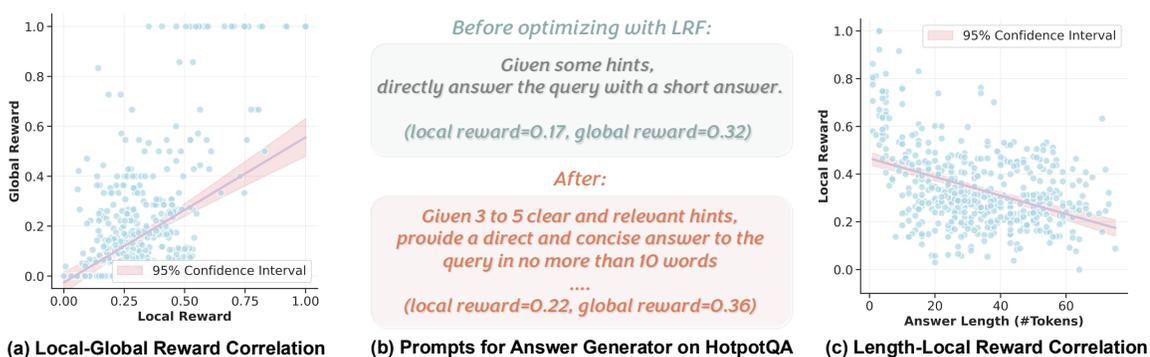


Figure 5.6: An interpretability study on what is learned by LRFs. With (a) a well-aligned LRF, we find that (b) the optimized prompt explicitly constrains the output length of the component. We attribute this to (c) the LRF preferring short outputs, which is consistent with the use of F1 as global metric.

An important aspect of system optimization is interpretability, *i.e.*, whether the configuration updates are reliable and understandable. We provide a study in Figure 5.6, where the LRF learns to favor concise answers. In fact, it is more feasible to interpret configuration

updates with LRFs. Specifically, one can perturb the component outputs in certain ways, and observe the changes in local rewards to obtain insights.

Local vs. global reward landscapes.

Table 5.8: Alignment between local and global reward landscapes on HOTPOTQA. Sweeping the retriever’s top- k reveals closely aligned local and global reward landscapes.

k	1	2	3	5	10	15	25
Local reward	0.4247	0.5578	0.5695	0.6124	0.6117	0.5949	0.5123
Global reward	0.3398	0.3493	0.3325	0.3598	0.3645	0.3568	0.3465

To better understand how local objectives reflect global performance, we sweep the retriever’s top- k setting on HOTPOTQA and compare local and global rewards (Table 5.8). The two landscapes are closely aligned: both are unimodal and peak at nearby values ($k=5$ for the local reward and $k=10$ for the global reward). The top-3 configurations $\{5, 10, 15\}$ coincide, differing only in the order of the top-2. This alignment shows that the local reward provides a reliable proxy for the global objective, lending empirical support to our theoretical guarantee that local optimization drives system-level gains.

5.6.6 Efficiency: How data- and compute-efficient is OPTIMAS

We conduct extensive studies to show the data efficiency with varying dataset size and computational efficiency with lightweight models. Moreover, we provide cost estimates on the AMAZON system compared to DSPy and TextGrad.

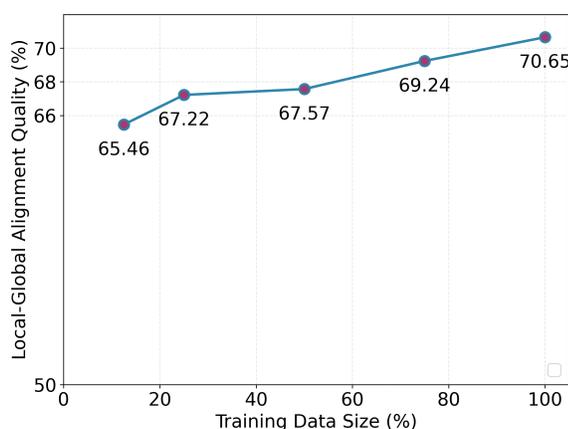


Figure 5.7: Impact of training data size on evaluation accuracy for the HOTPOTQA system. OPTIMAS maintains robust performance even with reduced training data, demonstrating only a 5.19% absolute performance drop when using just 12.5% of the data compared to the full dataset.

Data efficiency. To evaluate the data efficiency of OPTIMAS, we examine how the amount of training data (system runs) affects the performance of learned LRFs. We train local reward models on the HOTPOTQA system using varying percentages of available training data: 12.5%, 25%, 50%, 75%, and 100%. Each model is trained for 3 epochs, and performance is measured by pairwise ranking accuracy.

OPTIMAS maintains reasonable performance across different data sizes as shown in Figure 5.7. Even with only 12.5% of the data, the system achieves 65.46% accuracy (92.7% of full-data performance), while 25% of the data yields 67.22% accuracy (95.1% of full-data performance). The diminishing returns beyond 25% data suggest that the core patterns for local-global alignment can be captured efficiently from limited preference data.

This data efficiency is particularly valuable in practice, where collecting extensive preference data through system runs can be computationally expensive. Combined with the reduced computational cost of local optimization, these results demonstrate that OPTIMAS provides a practical solution for optimizing complex compound AI systems even in resource-constrained environments.

Table 5.9: Effect of LRF backbone size on pairwise ranking accuracy (%) on HOTPOTQA. Lightweight backbones are competitive with larger ones.

Backbone size	Pairwise accuracy (%)
1B	72.58
3B	70.04
8B	71.20

LRF backbone size. We also examine whether the size of the LRF backbone affects alignment quality. On HOTPOTQA, we compare 1B, 3B, and 8B backbones (Table 5.9). Performance remains comparable across sizes, with pairwise accuracies of 72.58%, 70.04%, and 71.20%, respectively. The lack of clear benefit from larger backbones suggests that lightweight models are sufficient for this task, making local optimization both effective and computationally efficient.

Prompt candidates per step. Another factor is the number of candidate prompts considered at each optimization step. On HOTPOTQA, we vary this number from 3 to 10 and measure the resulting global performance (Table 5.10). The best F1 score is achieved with 7 candidates, although results with fewer candidates remain competitive. These findings indicate that exhaustive candidate pools are unnecessary, and that modest numbers already yield strong performance with lower computational cost.

Table 5.10: Effect of number of prompt candidates on global performance (F1) on HOTPOTQA. We vary the number of candidate prompts per optimization step.

# candidates	3	5	7	10
Final F1	0.2822	0.1945	0.2968	0.2405

New inputs for LRF adaptation. Finally, we explore how many new inputs are needed when adapting the LRFs. On HOTPOTQA, we test adaptation with 10 to 40 new inputs (Table 5.11). Performance improves up to about 20 inputs, after which gains plateau and even slightly decline. This suggests that effective adaptation can be achieved with relatively small amounts of new data, reinforcing the practicality of our approach in scenarios where

data collection is limited.

Table 5.11: Effect of new inputs for LRF adaptation on global performance (F1) on HOPOTQA. Number of new inputs used to collect preference pairs for LRF adaptation vs. final global performance.

# new inputs	10	20	30	40
Final F1	0.2773	0.2822	0.2659	0.2533

Cost estimation. We report a detailed breakdown of the cost on the AMAZON system. This is the only system where components have trainable local models and thus requires PPO training. For all methods, we measure cost in terms of *full system runs*, *i.e.*, one invocation of the system, and assume that all components contribute equally to the per-run cost. We label the three components on Amazon system as A, B, C in topological order.

Table 5.12: Cost-performance comparison on the AMAZON system. Individual cost components are shown in actual runs, with total runs measured in thousands (k). “LRF training” and “LRF adaptation” only apply to OPTIMAS, which learns trainable local reward functions. DSPy and TextGrad optimize only the prompt of the last component C , and we report their cost as an *effective* number of full system runs.

Method	LRF training	LRF adaptation	Validation cost	Total runs (k)	Performance (%)
OPTIMAS	120	100	93	0.31	24.24
TextGrad	–	–	320	0.32	20.88
DSPy	–	–	240	0.24	18.18

For OPTIMAS, the total of ≈ 0.31 k effective full system runs decomposes into three parts: (i) **LRF training**: we collect 60 initial preference pairs per component; each pair requires 2 full system runs, giving $60 \times 2 = 120$ runs; (ii) **LRF adaptation**: during local optimization, we update the LRFs 5 times, each time collecting 10 new preference pairs per component, again with 2 runs per pair, for a total of $5 \times 10 \times 2 = 100$ runs; and (iii) **global validation**: whenever a local update is predicted to improve the global metric, we evaluate the new configuration on a held-out validation set. We use 20 validation inputs and observe 7 such updates, but the *effective* cost is discounted by a factor of $2/3$ due to

cached trajectories: $20 \times 7 \times \frac{2}{3} \approx 93$.

For DSPy and TextGrad, we follow their standard setup and optimize only the prompt for the final component C , while holding A and B fixed. To obtain a fair cost comparison in terms of effective full system runs, we (i) pre-compute a pool of 20 validation inputs by running A and B once, and then (ii) run only C during optimization. Assuming A , B , and C have comparable cost, we convert the number of C -only calls into an effective number of full system runs by dividing by three.

Under this cost-normalized view, OPTIMAS achieves the best global performance while using a comparable (slightly lower) effective number of full system runs than TextGrad and only moderately more than DSPy. This demonstrates that local optimization with learned reward functions can deliver stronger performance without incurring prohibitive cost.

Additional PPO training cost. Beyond system-run cost, OPTIMAS incurs additional compute for training local models with PPO. We measure this cost in GPU-hours. For each local PPO update on the AMAZON system, we train for 3 epochs on a single NVIDIA A100-SXM4-80GB GPU. Averaged over 5 runs of local optimization, each run takes approximately 12 minutes, yielding a total of about 6 GPU-hours (equivalently, 1.5 hours on 4 GPUs). This one-time training overhead is modest relative to the cost of repeated system evaluations and is only required for systems with trainable local components.

5.7 Conclusion

The central question of this chapter, *i.e.*, how to scale multi-agent systems while maintaining alignment with shared goals, has a surprisingly elegant answer: give each agent a learned local reward function that reflects how its decisions affect the whole system. When these local rewards are properly aligned with the global objective, agents can be optimized independently yet still improve collective performance.

Key Insights. Two findings from this work challenge conventional assumptions.

First, the prevailing view is that optimizing agents in isolation doesn't necessarily improve systems because each agent ignores how its outputs affect downstream components. We show that this limitation can be mitigated: when local reward functions are trained to

correlate with global performance, local optimization becomes a principled strategy (while it is true that it should satisfy certain conditions, as we discuss in the theory section).

Second, the degree of alignment between local rewards and global outcomes is the pivotal factor. When alignment is strong, local optimization reliably improves global performance; when alignment is weak, local improvements may not translate to global gains. This insight highlights the importance of learning high-quality local reward functions that accurately reflect global objectives.

Limitations and Open Questions. The framework rests on assumptions that may not hold universally. We assume each agent’s outputs can be evaluated without running the full system, which breaks down in systems with feedback loops where downstream agents influence upstream behavior. We assume a single global objective, but users often have multiple, conflicting goals.

More fundamentally, the learned reward functions capture correlation rather than causation. If the global metric changes or downstream components are modified, the learned preferences may become misleading. Building reward functions that capture more robust notions of output quality is an important direction for future work.

Chapter 6

Conclusion

This thesis studies how to build capable AI agents that effectively collaborate with human users. While modern foundation models demonstrate remarkable capabilities in reasoning and planning, they often fail at basic collaboration: making unwarranted assumptions, missing opportunities to clarify intent, and optimizing for local responses that harm the conversation overall. We addressed these challenges through three interconnected research directions: training agents for collaborative behaviors (Part I), building agent systems that recognize their limitations and effectively address them (Part II), and scaling these systems while maintaining alignment with shared goals (Part III).

6.1 Summary of Contributions

Table 6.1 summarizes the contributions of this thesis across three dimensions: the collaborative *behaviors* we enable in AI agents, the *systems* we develop to achieve these behaviors, and the *benchmarks* we create to evaluate collaborative AI.

	Part I <i>Collaborative Behaviors</i>	Part II <i>Learning from Experience</i>	Part III <i>Scaling with Alignment</i>
Core Insight	Agents trained on single-turn rewards generate passive responses	Agents cannot recognize their own limitations or learn from experience	Local optimization can conflict with global system goals
Approach	Multiturn-aware rewards that capture long-term effects	Contrastive reasoning to identify limitations and generate actionable instructions	Globally aligned local reward functions for each agent
System	COLLABLLM: forward sample trajectories with user simulators	AVATAR: Actor-Comparator architecture with batch-wise feedback	OPTIMAS: learned reward alignment with lightweight online adaptation
Evaluation	Multiturn benchmarks	STARK datasets	Five multiagent systems

Table 6.1: Thesis contributions. Each part addresses a challenge in building collaborative AI: enabling collaborative behaviors in individual agents, building systems that recognize limitations and learn from experience, and maintaining alignment as systems scale.

6.1.1 Part I: Training Collaborative Agents

In Chapter 2, we identify that current LLMs are trained with objectives that favor single-turn success over multiturn collaboration. This leads to agents that make early assumptions, fail to ask useful clarifying questions, and produce responses that harm the conversation later. We introduce COLLABLLM, a framework that transforms non-collaborative agents into effective collaborators by optimizing for long-term conversation outcomes. The key insight is that collaborative behavior emerges naturally when agents are trained with a multiturn-aware objective that considers the downstream impact of each response. Through extensive simulated and real-world evaluations, we demonstrate that COLLABLLM agents markedly outperform non-collaboratively trained baselines, with users reporting more efficient, engaging, and creative interactions.

6.1.2 Part II: Enabling Agents to Learn from Experience

Chapter 3 establishes that agents fail to learn from experience even when they have access to memory and past interactions. We introduce STARK, a benchmark for evaluating whether agents can improve on retrieval tasks over semi-structured knowledge bases when exposed to similar queries multiple times. Our experiments reveal that state-of-the-art agentic baselines including ReAct and Reflexion repeat the same mistakes across attempts, demonstrating a systematic failure to recognize their own limitations and learn from past errors.

Chapter 4 addresses this limitation by enabling agents to recognize what went wrong and generate actionable guidance for improvement. We introduce AVATAR, a framework where a Comparator agent analyzes patterns across successful and failed attempts to identify systematic gaps in the Actor agent’s strategies. Through contrastive reasoning, the system reveals not just what failed, but why it failed, enabling the agent to extract insights it cannot recognize on its own. AVATAR achieves up to 16.6% relative improvement over the best baselines on STARK and demonstrates that agents can systematically learn to improve their performance on collaborative tasks over multiple attempts.

6.1.3 Part III: Scaling Up Collaborative Multi-Agent Systems

Chapter 5 addresses the challenge that emerges as multi-agent systems grow: individual agents optimized in isolation may pursue local objectives that conflict with shared collaborative goals. We introduce OPTIMAS, a framework that learns globally aligned reward functions for each component in a compound AI system. The framework provides theoretical guarantees that local optimization leads to global improvement, develops lightweight adaptation mechanisms for evolving systems, and demonstrates consistent gains across five diverse benchmarks. This work completes the thesis arc by showing that effective collaboration at scale requires not just capable individual agents, but principled mechanisms to ensure alignment between local and global objectives.

6.2 Vision for the Future

We point to several directions where collaborative AI can have huge impact.

Scientific discovery through human-AI teams. Consider a researcher investigating a rare disease. Today, they must manually search across genomic databases, clinical literature, and drug interaction networks, often missing connections that span these sources. A collaborative system could change this: an agent trained with multiturn-aware rewards (Part I) would ask clarifying questions about the researcher’s hypothesis before searching; a multi-agent architecture like AVATAR (Part II) would coordinate specialized agents to query heterogeneous knowledge bases while learning from which search strategies succeed; and aligned reward functions (Part III) would ensure that each agent optimizes for the researcher’s actual goals rather than proxy metrics like retrieval volume. Extending these methods to support iterative hypothesis refinement would enable genuine scientific partnership.

Personalized education at scale. Effective tutoring requires understanding not just what a student asks, but what they are struggling to articulate. The multiturn-aware training from Part I provides the foundation: agents that learn to ask diagnostic questions rather than immediately providing answers. A student working through calculus might interact with a system that notices they consistently struggle with implicit differentiation, asks targeted questions to identify the conceptual gap, and coordinates with a visualization agent to provide geometric intuition. The multi-agent coordination from Parts II and III enables such systems to combine specialized capabilities, such as problem generation, hint provision, and progress tracking, while remaining aligned with the student’s learning objectives.

Collective creativity and problem-solving. The most ambitious applications involve many humans and many agents working together. Consider a software team building a complex product: one engineer commands an agent army for code generation and refactoring, another leads agents handling testing and debugging, while a third coordinates documentation and API design. These human-agent teams must then collaborate with each other, with product managers defining requirements, and with designers shaping the user experience. No single participant, human or AI, holds all the answers. The challenge is

ensuring that every contributor, whether an individual agent, a human-led team, or the full organization, optimizes for the shared product vision rather than local metrics. The path forward lies not in building ever-more-powerful individual agents, but in designing systems where diverse capabilities at every level combine into coherent collaboration.

6.3 Concluding Remarks

This thesis advances the vision of collaborative AI: systems that do not merely execute tasks, but work alongside humans to achieve goals that neither could accomplish alone. We have shown that enabling collaboration requires rethinking how agents are trained (Part I), how systems are structured (Part II), and how components are optimized (Part III).

The key insight unifying these contributions is that collaboration is fundamentally about alignment: between short-term actions and long-term outcomes, between individual agent behaviors and user intent, between local component objectives and global system goals. By developing principled approaches to alignment at each of these levels, we provide a foundation for building AI systems that genuinely serve human needs.

Ultimately, our goal is to democratize collaborative AI so that effective human-AI collaboration is accessible to anyone. As AI systems become more capable and are deployed in increasingly complex settings, the alignment principles developed in this thesis, from multistep-aware rewards for single agents to globally aligned local rewards for compound systems, offer a path toward AI that empowers creativity and intelligence that neither humans nor AI could achieve alone.

Bibliography

- [1] Marwa Abdulhai, Isadora White, Charlie Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin Xu, and Sergey Levine. LMRL gym: Benchmarks for multi-turn reinforcement learning with language models. *arXiv:2311.18232*, 2023.
- [2] Voyage AI. Voyage ai embeddings api, 2023. [Software].
- [3] Saleema Amershi, Daniel S. Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi T. Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. Guidelines for human-ai interaction. In *CHI*. ACM, 2019.
- [4] Chinmaya Andukuri, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah D. Goodman. Star-gate: Teaching language models to ask clarifying questions. *CoLM*, 2024.
- [5] Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *ICLR*, 2020.
- [6] Krisztian Balog and ChengXiang Zhai. Rethinking conversational agents in the era of llms: Proactivity, non-collaborativity, and beyond. In *SIGIR-AP*, 2023.
- [7] Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. LLM2Vec: Large language models are secretly powerful text encoders. *arXiv preprint*, 2024.
- [8] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, 2013.

- [9] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. *arXiv:2308.09687*, 2023.
- [10] Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with sub-graph embeddings. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, October 2014. Association for Computational Linguistics.
- [11] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075, 2015.
- [12] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *ICML*, 2022.
- [13] Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. KQA pro: A dataset with explicit compositional programs for complex question answering over knowledge base. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022.
- [14] Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent llm systems fail?, 2025.

- [15] Payal Chandak, Kexin Huang, and Marinka Zitnik. Building a knowledge graph to enable precision medicine. *Scientific Data*, 2023.
- [16] Angelica Chen, Jason Phang, Alicia Parrish, Vishakh Padmakumar, Chen Zhao, Samuel R. Bowman, and Kyunghyun Cho. Two failures of self-consistency in the multi-step reasoning of llms, 2024.
- [17] Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning, 2023.
- [18] Changyu Chen, Zichen Liu, Chao Du, Tianyu Pang, Qian Liu, Arunesh Sinha, Pradeep Varakantham, and Min Lin. Bootstrapping language models with DPO implicit rewards. In *International Conference on Learning Representations (ICLR)*, 2025.
- [19] Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Step-level value preference optimization for mathematical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024.
- [20] Jiaao Chen, Aston Zhang, Xingjian Shi, Mu Li, Alex Smola, and Diyi Yang. Parameter-efficient fine-tuning design spaces, 2023.
- [21] Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Matei Zaharia, James Zou, and Ion Stoica. Optimizing model selection for compound ai systems, 2025.
- [22] Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024.
- [23] Maximillian Chen, Ruoxi Sun, Sercan Ö. Arik, and Tomas Pfister. Learning to clarify: Multi-turn conversations with action-based contrastive self-training. *arXiv:2406.00222*, 2024.
- [24] Maximillian Chen, Xiao Yu, Weiyan Shi, Urvi Awasthi, and Zhou Yu. Controllable mixed-initiative dialogue generation through prompting. In *ACL*, 2023.

- [25] Wenhui Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyu Zhou, and William Yang Wang. Tabfact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations*, 2020.
- [26] Wenhui Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, November 2020.
- [27] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug, 2023.
- [28] Yiqun Chen, Lingyong Yan, Weiwei Sun, Xinyu Ma, Yi Zhang, Shuaiqiang Wang, Dawei Yin, Yiming Yang, and Jiaxin Mao. Improving retrieval-augmented generation through multi-agent reinforcement learning. *CoRR*, 2025.
- [29] Zhenfang Chen, Delin Chen, Rui Sun, Wenjun Liu, and Chuang Gan. Scaling autonomous agents via automatic reward modeling and planning, 2025.
- [30] Zhiliang Chen, Chuan-Sheng Foo, and Bryan Kian Hsiang Low. Towards autoai: Optimizing a machine learning system with black-box and differentiable components. In *ICML*, 2024.
- [31] Yizhou Chi, Jessy Lin, Kevin Lin, and Dan Klein. CLARINET: augmenting language models to ask clarification questions for retrieval. *arXiv:2405.15784*, abs/2405.15784, 2024.
- [32] Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. INSTRUCTEVAL: towards holistic evaluation of instruction-tuned large language models. *CoRR*, abs/2306.04757, 2023.
- [33] Fabio Chiusano. Medium articles dataset. <https://huggingface.co/datasets/fabiochiu/medium-articles>, 2024. A dataset of Medium articles collected through web scraping, including metadata such as titles, authors, publication dates, tags, and content.

- [34] Sanjiban Choudhury. Process reward models for llm agents: Practical framework and directions, 2025.
- [35] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. In *EMNLP. ACL*, 2022.
- [36] Yang Deng, Lizi Liao, Liang Chen, Hongru Wang, Wenqiang Lei, and Tat-Seng Chua. Prompting and evaluating large language models for proactive dialogues: Clarification, target-guided, and non-collaboration. In *EMNLP*, 2023.
- [37] Yang Deng, Wenxuan Zhang, Wai Lam, See-Kiong Ng, and Tat-Seng Chua. Plug-and-play policy planner for large language model powered dialogue agents. In *ICLR*, 2024.
- [38] Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. Rephrase and respond: Let large language models ask better questions for themselves. *arXiv:2311.04205*, 2023.
- [39] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv:2305.14325*, 2023.
- [40] Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Guney, Volkan Cirik, and Kyunghyun Cho. Searchqa: A new qa dataset augmented with context from a search engine, 2017.
- [41] Zane Durante, Qiuyuan Huang, Naoki Wake, Ran Gong, Jae Sung Park, Bidipta Sarkar, Rohan Taori, Yusuke Noda, Demetri Terzopoulos, Yejin Choi, Katsushi Ikeuchi, Hoi Vo, Li Fei-Fei, and Jianfeng Gao. Agent AI: surveying the horizons of multimodal interaction. *arXiv:2401.03568*, 2024.
- [42] Pierre Erbacher and Laure Soulier. CIRCLE: multi-turn query clarifications with reinforcement learning. *arXiv:2311.02737*, 2023.

- [43] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1437–1446. PMLR, 10–15 Jul 2018.
- [44] Felix Faltings, Michel Galley, Kianté Brantley, Baolin Peng, Weixin Cai, Yizhe Zhang, Jianfeng Gao, and Bill Dolan. Interactive text generation. In *EMNLP, 2023*.
- [45] Jianfeng Gao, Michel Galley, and Lihong Li. Neural approaches to conversational AI. *Found. Trends Inf. Retr.*, 13, 2019.
- [46] Tiantian Gao, Paul Fodor, and Michael Kifer. Querying knowledge via multi-hop english questions. *Theory and Practice of Logic Programming*, 2019.
- [47] Zhaolin Gao, Wenhao Zhan, Jonathan D. Chang, Gokul Swamy, Kianté Brantley, Jason D. Lee, and Wen Sun. Regressing the relative future: Efficient policy optimization for multi-turn rlhf, 2024.
- [48] Ran Gong, Qiuyuan Huang, Xiaojian Ma, Hoi Vo, Zane Durante, Yusuke Noda, Zilong Zheng, Song-Chun Zhu, Demetri Terzopoulos, Li Fei-Fei, and Jianfeng Gao. Mindagent: Emergent gaming interaction. *arXiv:2309.09971*, 2023.
- [49] Meiqi Guo, Mingda Zhang, Siva Reddy, and Malihe Alikhani. Abg-coqa: Clarifying ambiguity in conversational question answering. In *AKBC*, 2021.
- [50] Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers, 2024.
- [51] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *ICML*. PMLR, 2020.
- [52] Simon Jerome Han, Keith J. Ransom, Andrew Perfors, and Charles Kemp. Inductive reasoning in humans and large language models. *Cogn. Syst. Res.*, 2024.

- [53] Ruining He and Julian J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*. ACM, 2016.
- [54] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering, 2024.
- [55] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.
- [56] Lynette Hirschman and Robert Gaizauskas. Natural language question answering: the view from here. *natural language engineering*, 2001.
- [57] Joey Hong, Sergey Levine, and Anca D. Dragan. Zero-shot goal-directed dialogue via RL on imagined conversations. *CoRR*, 2023.
- [58] Lu Hong and Scott E Page. Groups of diverse problem solvers can outperform groups of high-ability problem solvers. *Proceedings of the National Academy of Sciences*, 101(46), 2004.
- [59] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. Large language models are zero-shot rankers for recommender systems. *arXiv:2305.08845*, 2023.
- [60] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.
- [61] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*, 2020.
- [62] Zhiyuan Hu, Yue Feng, Anh Tuan Luu, Bryan Hooi, and Aldo Lipani. Unlocking the potential of user feedback: Leveraging large language model as user simulators to enhance dialogue system. In *CIKM*. ACM, 2023.

- [63] Dong Huang, Qingwen Bu, Jie M. Zhang, Michael Luck, and Heming Cui. Agent-coder: Multi-agent-based code generation with iterative testing and optimisation. *CoRR*, abs/2312.13010, 2023.
- [64] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *ICML*, 2022.
- [65] Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *ICLR*, 2024.
- [66] Edwin Hutchins. *Cognition in the Wild*. MIT Press, 1995.
- [67] Vassilis N Ioannidis, Xiang Song, Da Zheng, Houyu Zhang, Jun Ma, Yi Xu, Belinda Zeng, Trishul Chilimbi, and George Karypis. Efficient and effective training of language and graph neural network models. *arXiv preprint arXiv:2206.10781*, 2022.
- [68] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*, 2021.
- [69] Hasan M. Jamil. Knowledge rich natural language queries over structured biological databases. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, 2017.
- [70] Sujay Kumar Jauhar, Peter D. Turney, and Eduard H. Hovy. Tabmcq: A dataset of general knowledge tables and multiple-choice questions. *CoRR*, abs/1602.03960, 2016.
- [71] Yucheng Jiang, Yijia Shao, Dekun Ma, Sina J. Semnani, and Monica S. Lam. Into the unknown unknowns: Engaged human learning through participation in language model agent conversations. *arXiv:2408.15232*, 2024.

- [72] Fangkai Jiao, Chengwei Qin, Zhengyuan Liu, Nancy F. Chen, and Shafiq Joty. Learning planning-based reasoning with trajectory collection and process rewards synthesizing. In *EMNLP*. Association for Computational Linguistics, 2024.
- [73] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. In *EMNLP-IJCNLP*, 2019.
- [74] Yilun Jin, Zheng Li, Chenwei Zhang, Tianyu Cao, Yifan Gao, Pratik Jayarao, Mao Li, Xin Liu, Ritesh Sarkhel, Xianfeng Tang, et al. Shopping mmlu: A massive multi-task online shopping benchmark for large language models. *arXiv preprint arXiv:2410.20745*, 2024.
- [75] Kevin B. Johnson, Wei-Qi Wei, Dharini Weeraratne, Mark E. Frisse, Kevin Misulis, Kevin Rhee, Jie Zhao, and J. L. Snowdon. Precision medicine, ai, and the future of personalized health care. *Clinical and Translational Science*, 2021.
- [76] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*, 2017.
- [77] Eser Kandogan, Nikita Bhutani, Dan Zhang, Rafael Li Chen, Sairam Gurajada, and Estevam Hruschka. Orchestrating agents and data for enterprise: A blueprint architecture for compound ai, 2025.
- [78] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP*, 2020.
- [79] Esther Kaufmann and Abraham Bernstein. Evaluating the usability of natural language query languages and interfaces to semantic web knowledge bases. *Journal of Web Semantics*, 2010.
- [80] Sedrick Keh, Justin T. Chiu, and Daniel Fried. Asking more informative questions for grounded retrieval. In *FNAACL*, 2024.

- [81] Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp, 2023.
- [82] Gangwoo Kim, Sungdong Kim, Byeongguk Jeon, Joonsuk Park, and Jaewoo Kang. Tree of clarifications: Answering ambiguous questions with retrieval-augmented large language models. In *EMNLP*, 2023.
- [83] Yoonsu Kim, Jueon Lee, Seoyoung Kim, Jaehyuk Park, and Juho Kim. Understanding users’ dissatisfaction with chatgpt responses: Types, resolving tactics, and the effect of knowledge level. In *IUI*. ACM, 2024.
- [84] Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. Mt-eval: A multi-turn capabilities evaluation benchmark for large language models. In *EMNLP*, 2024.
- [85] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 2019.
- [86] Hoang Minh Le, Nan Jiang, Alekh Agarwal, Miroslav Dudík, Yisong Yue, and Hal Daumé III. Hierarchical imitation and reinforcement learning. In *ICML*, 2018.
- [87] Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu-Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. In *NeurIPS*, 2022.
- [88] Rémi Leblond and et al. Alphacode 2 technical report. Technical report, DeepMind, 2023.
- [89] Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and

- Sushant Prakash. RLAIIF vs. RLHF: scaling reinforcement learning from human feedback with AI feedback. In *ICML*, 2024.
- [90] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In *ACL*, 2019.
- [91] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv:1811.07871*, 2018.
- [92] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*, 2020.
- [93] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: communicative agents for "mind" exploration of large scale language model society. *arXiv:2303.17760*, abs/2303.17760, 2023.
- [94] Lei Li, Yuqi Wang, Runxin Xu, Peiyi Wang, Xiachong Feng, Lingpeng Kong, and Qi Liu. Multimodal ArXiv: A dataset for improving scientific comprehension of large vision-language models. In *ACL*, 2024.
- [95] Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In *EMNLP*. Association for Computational Linguistics, 2023.
- [96] Lizi Liao, Grace Hui Yang, and Chirag Shah. Proactive conversational agents in the post-chatgpt world. In *SIGIR*, 2023.
- [97] Petro Liashchynskyi and Pavlo Liashchynskyi. Grid search, random search, genetic algorithm: A big comparison for NAS. *arXiv:1912.06059*, 2019.
- [98] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024.

- [99] Kunyang Lin, Yufeng Wang, Peihao Chen, Runhao Zeng, Siyuan Zhou, Mingkui Tan, and Chuang Gan. Dcir: Dynamic consistency intrinsic reward for multi-agent reinforcement learning, 2023.
- [100] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search, 2019.
- [101] Yimeng Liu, Misha Sra, Jeevana Priya Inala, and Chenglong Wang. Reuseit: Synthesizing reusable ai agent workflows for web automation. In *Proceedings of the 31st International Conference on Intelligent User Interfaces, IUI '26*, 2026. arXiv:2510.14308.
- [102] Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. *arXiv:2310.02170*, 2023.
- [103] AI @ Meta Llama Team. The llama 3 herd of models, 2024.
- [104] Robert Lo, Abishek Sridhar, Frank Xu, Hao Zhu, and Shuyan Zhou. Hierarchical prompting assists large language model on web navigation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, December 2023.
- [105] Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models. In *NeurIPS*, 2023.
- [106] Yiran Ma, Zui Chen, Tianqiao Liu, Mi Tian, Zhuo Liu, Zitao Liu, and Weiqi Luo. What are step-level reward models rewarding? counterintuitive findings from mcts-boosted mathematical reasoning, 2025.
- [107] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *NeurIPS*, 2023.

- [108] Thomas W Malone and Kevin Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1), 1994.
- [109] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*. ACM, 2015.
- [110] Fangwen Mu, Lin Shi, Song Wang, Zhuohao Yu, Binqun Zhang, Chenxue Wang, Shichao Liu, and Qing Wang. Clarifygpt: Empowering llm-based code generation with intention clarification. *arXiv:2310.10996*, abs/2310.10996, 2023.
- [111] Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. Generative representational instruction tuning, 2024.
- [112] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback. *arXiv:2112.09332*, 2021.
- [113] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, CEUR Workshop Proceedings, 2016.
- [114] Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Sejr Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering. In *ACL Findings*, 2022.
- [115] OpenAI. Openai embeddings api, 2023. [Software].
- [116] Krista Opsahl-Ong, Michael J. Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for

- multi-stage language model programs. In *EMNLP*. Association for Computational Linguistics, 2024.
- [117] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.
- [118] Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2023.
- [119] Jing-Cheng Pang, Heng-Bo Fan, Pengyuan Wang, Jiahao Xiao, Nan Tang, Si-Hang Yang, Chengxing Jia, Sheng-Jun Huang, and Yang Yu. Empowering language models with active inquiry for deeper understanding. *CoRR*, abs/2402.03719, 2024.
- [120] Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. Art: Automatic multi-step reasoning and tool-use for large language models. *arXiv:2303.09014*, 2023.
- [121] Aaron Parisi, Yao Zhao, and Noah Fiedel. Talm: Tool augmented language models, 2022.
- [122] Joon Sung Park, Carolyn Q. Zou, Aaron Shaw, Benjamin Mako Hill, Carrie Cai, Meredith Ringel Morris, Robb Willer, Percy Liang, and Michael S. Bernstein. Generative agent simulations of 1,000 people, 2024.
- [123] Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In *Association for Computational Linguistics*, July 2015.
- [124] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv:2305.15334*, 2023.

- [125] Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition, 2009.
- [126] Judea Pearl, Madelyn Glymour, and Nicholas P. Jewell. *Causal Inference in Statistics: A Primer*. John Wiley & Sons, 2016.
- [127] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and Jianfeng Gao. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arxiv:2302.12813*, 2302.12813, 2023.
- [128] Jingyu Peng, Maolin Wang, Xiangyu Zhao, Kai Zhang, Wanyu Wang, Pengyue Jia, Qidong Liu, Ruocheng Guo, and Qi Liu. Stepwise reasoning error disruption attack of llms, 2025.
- [129] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4095–4104. PMLR, 10–15 Jul 2018.
- [130] Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *Int. J. Comput. Vis.*, 2017.
- [131] Yujia Qin, Zihan Cai, Dian Jin, Lan Yan, Shihao Liang, Kunlun Zhu, Yankai Lin, Xu Han, Ning Ding, Huadong Wang, Ruobing Xie, Fanchao Qi, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Webcpm: Interactive web search for chinese long-form question answering. In *Proceedings of ACL 2023*. Association for Computational Linguistics, 2023.
- [132] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arxiv:2307.16789*, 2023.

- [133] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023.
- [134] Hossein A. Rahmani, Xi Wang, Yue Feng, Qiang Zhang, Emine Yilmaz, and Aldo Lipani. A survey on asking clarification questions datasets in conversational systems. In *ACL*, 2023.
- [135] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, November 2016. Association for Computational Linguistics.
- [136] Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 2009.
- [137] Pum-Mo Ryu, Myung-Gil Jang, and Hyunki Kim. Open domain question answering using wikipedia-based knowledge model. *Inf. Process. Manag.*, 2014.
- [138] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *NAACL*, 2022.
- [139] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, 2023.
- [140] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.
- [141] Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. *ICLR*, 2025.
- [142] Lior Shani, Aviv Rosenberg, Asaf B. Cassel, Oran Lang, Daniele Calandriello, Avital Zipori, Hila Noga, Orgad Keller, Bilal Piot, Idan Szepkator, Avinatan Hassidim,

- Yossi Matias, and Rémi Munos. Multi-turn reinforcement learning from preference human feedback. *arXiv:2405.14655*, 2024.
- [143] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27, 1948.
- [144] Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. R-prm: Reasoning-driven process reward modeling, 2025.
- [145] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving AI tasks with chatgpt and its friends in hugging face. In *NeurIPS*, 2023.
- [146] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. REPLUG: retrieval-augmented black-box language models. *arXiv:2301.12652*, 2023.
- [147] Weiyang Shi, Kun Qian, Xuwei Wang, and Zhou Yu. How to build user simulators to train rl-based dialog systems. In *EMNLP-IJCNLP*. Association for Computational Linguistics, 2019.
- [148] Wenqi Shi, Ran Xu, Yuchen Zhuang, Yue Yu, Jieyu Zhang, Hang Wu, Yuanda Zhu, Joyce Ho, Carl Yang, and May D Wang. Ehragent: Code empowers large language models for complex tabular reasoning on electronic health records. *arXiv:2401.07128*, 2024.
- [149] Wentao Shi, Mengqi Yuan, Junkang Wu, Qifan Wang, and Fuli Feng. Direct multi-turn preference optimization for language agents. In *EMNLP*. Association for Computational Linguistics, 2024.
- [150] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *NeurIPS*, 2023.
- [151] Herbert A Simon. *Models of Man: Social and Rational*. Wiley, 1957.

- [152] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Paul Hsu, and Kuansan Wang. An overview of microsoft academic service (MAS) and applications. In *WWW*, 2015.
- [153] Kai Sun, Yifan Ethan Xu, Hanwen Zha, Yue Liu, and Xin Luna Dong. Head-to-tail: How knowledgeable are large language models (llm)? A.K.A. will llms replace knowledge graphs? *NAACL*, 2024.
- [154] Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is chatgpt good at search? investigating large language models as re-ranking agents. In *EMNLP*, 2023.
- [155] James Surowiecki. *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*. Doubleday, 2004.
- [156] Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In *NAACL-HLT*, 2018.
- [157] Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv:2306.05301*, 2023.
- [158] Robert S Taylor. Question-negotiation and information seeking in libraries. *College & research libraries*, 29(3):178–194, 1968.
- [159] Mildred C. Templin. *Certain Language Skills in Children: Their Development and Interrelationships*, volume 26. University of Minnesota Press, 1957.
- [160] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordani, Philip Bachman, and Kaheer Suleman. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Association for Computational Linguistics, August 2017.

- [161] Jen tse Huang, Jiayu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Michael R. Lyu, and Maarten Sap. On the resilience of llm-based multi-agent collaboration with faulty agents, 2025.
- [162] Bo-Hsiang Tseng, Yinpei Dai, Florian Kreyszig, and Bill Byrne. Transferable dialogue systems and user simulators. In *ACL/IJCNLP*. Association for Computational Linguistics, 2021.
- [163] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109:475–494, 2001.
- [164] Jonathan Uesato, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Y. Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *CoRR*, abs/2211.14275, 2022.
- [165] Jiayin Wang, Weizhi Ma, Peijie Sun, Min Zhang, and Jian-Yun Nie. Understanding user experience in large language model interactions. *arXiv:2401.08329*, 2024.
- [166] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quant. Sci. Stud.*, 2020.
- [167] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *ACL*, pages 9426–9439. Association for Computational Linguistics, 2024.
- [168] Ruocheng Wang, Eric Zelikman, Gabriel Poesia, Yewen Pu, Nick Haber, and Noah D. Goodman. Hypothesis search: Inductive reasoning with language models. *ICLR*, 2024.
- [169] Teng Wang, Zhangyi Jiang, Zhenqi He, Wenhan Yang, Yanan Zheng, Zeyu Li, Zifan He, Shenyang Tong, and Hailei Gong. Towards hierarchical multi-step reward models for enhanced reasoning in large language models, 2025.

- [170] Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P. Xing, and Zhiting Hu. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *arXiv:2310.16427*, 2023.
- [171] Zhaozhi Wang, Kefan Su, Jian Zhang, Huizhu Jia, Qixiang Ye, Xiaodong Xie, and Zongqing Lu. Multi-agent automated machine learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*. IEEE, 2023.
- [172] Daniel M Wegner. Transactive memory: A contemporary analysis of the group mind. In *Theories of group behavior*. Springer, 1987.
- [173] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.
- [174] Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing datasets for multi-hop reading comprehension across documents. *Trans. Assoc. Comput. Linguistics*, 2018.
- [175] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 1992.
- [176] Anita Williams Woolley, Christopher F Chabris, Alex Pentland, Nada Hashmi, and Thomas W Malone. Evidence for a collective intelligence factor in the performance of human groups. *Science*, 330(6004), 2010.
- [177] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversation framework. *arXiv:2308.08155*, 2023.
- [178] Shirley Wu, Evelyn Choi, Arpandeeep Khatua, Zhanghan Wang, Joy He-Yueya, Tharindu Cyril Weerasooriya, Wei Wei, Diyi Yang, Jure Leskovec, and James

- Zou. Humanlm: Simulating users with state alignment beats response imitation. <https://humanlm.stanford.edu/>, 2026.
- [179] Shirley Wu, Michel Galley, Baolin Peng, Hao Cheng, Gavin Li, Yao Dou, Weixin Cai, James Zou, Jure Leskovec, and Jianfeng Gao. Collabllm: From passive responders to active collaborators. In *ICML*, 2025.
- [180] Shirley Wu, Parth Sarthi, Shiyu Zhao, Aaron Lee, Herumb Shandilya, Adrian Mladenec Grobelnik, Nurendra Choudhary, Eddie Huang, Karthik Subbian, Linjun Zhang, Diyi Yang, James Zou, and Jure Leskovec. Optimas: Optimizing compound ai systems with globally aligned local rewards. In *ICLR*, 2026.
- [181] Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis N. Ioannidis, Karthik Subbian, Jure Leskovec, and James Zou. Avatar: Optimizing llm agents for tool usage via contrastive reasoning. *NeurIPS*, 2024.
- [182] Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis N. Ioannidis, Karthik Subbian, James Zou, and Jure Leskovec. Stark: Benchmarking llm retrieval on textual and relational knowledge bases. In *NeurIPS Datasets and Benchmarks Track*, 2024.
- [183] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *ICLR*, 2021.
- [184] Canwen Xu, Daya Guo, Nan Duan, and Julian J. McAuley. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. In *EMNLP*, 2023.
- [185] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *ICLR*, 2024.
- [186] Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. Gpt4tools: Teaching large language model to use tools via self-instruction. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *NeurIPS*, 2023.

- [187] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with bertserini. In *NAACL-HLT*, 2019.
- [188] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Association for Computational Linguistics*, October-November 2018.
- [189] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, 2018.
- [190] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*, 2023.
- [191] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *ICLR*, 2023.
- [192] 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. Retroformer: Retrospective large language agents with policy gradient optimization. *ICLR*, 2024.
- [193] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. Qa-gnn: Reasoning with language models and knowledge graphs for question answering, 2021.
- [194] Junjie Ye, Guanyu Li, Songyang Gao, Caishuang Huang, Yilong Wu, Sixian Li, Xiaoran Fan, Shihan Dou, Qi Zhang, Tao Gui, and Xuanjing Huang. Tooleyes: Fine-grained evaluation for tool learning capabilities of large language models in real-world scenarios. *arXiv:2401.00741*, 2024.

- [195] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL*, 2015.
- [196] Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In *Association for Computational Linguistics*, Berlin, Germany, August 2016.
- [197] Xunjian Yin, Xinyi Wang, Liangming Pan, Xiaojun Wan, and William Yang Wang. Gödel agent: A self-referential agent framework for recursive self-improvement, 2025.
- [198] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL. In *EMNLP*, pages 3911–3921, 2018.
- [199] Yang Yu, Qiyue Yin, Junge Zhang, Pei Xu, and Kaiqi Huang. ADMN: agent-driven modular network for dynamic parameter sharing in cooperative multi-agent reinforcement learning. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*. ijcai.org, 2024.
- [200] Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639:609–616, 2025.
- [201] Matei Zaharia, Omar Khattab, Lingjiao Chen, Jared Quincy Davis, Heather Miller, Chris Potts, James Zou, Michael Carbin, Jonathan Frankle, Naveen Rao, and Ali Ghodsi. The shift from models to compound ai systems. BAIR Berkeley blog, 2024.
- [202] Hamed Zamani, Susan T. Dumais, Nick Craswell, Paul N. Bennett, and Gord Lueck. Generating clarifying questions for information retrieval. In *WWW*, 2020.

- [203] J. D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. Why johnny can't prompt: How non-ai experts try (and fail) to design LLM prompts. In *CHI*. ACM, 2023.
- [204] Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. *arxiv:2310.12823*, 2023.
- [205] Chen Zhang, Xinyi Dai, Yaxiong Wu, Qu Yang, Yasheng Wang, Ruiming Tang, and Yong Liu. A survey on multi-turn interaction capabilities of large language models. *arXiv preprint arXiv:2501.09959*, 2025.
- [206] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. Aflow: Automating agentic workflow generation, 2025.
- [207] Michael J. Q. Zhang and Eunsol Choi. Clarify when necessary: Resolving ambiguity through interaction with lms. *arXiv:2311.09469*, 2023.
- [208] Shaokun Zhang, Jieyu Zhang, Jiale Liu, Linxin Song, Chi Wang, Ranjay Krishna, and Qingyun Wu. Training language model agents without modifying language models. *ICML*, 2024.
- [209] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. In *AAAI*, 2024.
- [210] Wanxia Zhao, Mert Yuksekogul, Shirley Wu, and James Zou. Sirius: Self-improving multi-agent systems via bootstrapped reasoning, 2025.
- [211] Ziliang Zhao and Zhicheng Dou. Generating multi-turn clarification for web information seeking. In *WWW*, 2024.
- [212] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E.

- Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *NeurIPS*, 2023.
- [213] Wenqing Zheng, SP Sharan, Ajay Kumar Jaiswal, Kevin Wang, Yihan Xi, Deji Xu, and Zhangyang Wang. Outline, then details: Syntactically guided coarse-to-fine code generation. *ICML*, 2023.
- [214] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017.
- [215] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *AAAI*, 2024.
- [216] Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulic, Anna Korhonen, and Sercan Ö. Arik. Multi-agent design: Optimizing agents with better prompts and topologies. *arXiv:2502.02533*, 2025.
- [217] Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn RL. In *ICML*, 2024.
- [218] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *ICLR*, 2023.
- [219] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. Large language models for information retrieval: A survey. *arXiv:2308.07107*, 2023.
- [220] Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. Beyond yes and no: Improving zero-shot LLM rankers via scoring fine-grained relevance labels. *CoRR*, abs/2310.14122, 2023.
- [221] Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. Toolqa: A dataset for llm question answering with external tools, 2023.

- [222] Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. *arXiv:2406.15877*, 2024.
- [223] Chang Zong, Yuchen Yan, Weiming Lu, Eliot Huang, Jian Shao, and Yueting Zhuang. Triad: A framework leveraging a multi-role llm-based agent to solve knowledge base question answering. *EMNLP*, 2024.